



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 © 2024 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

.....	xvi
Cos'è il servizio gestito per Apache Flink? .....	1
Scelta del servizio gestito per Apache Flink o del servizio gestito per Apache Flink Studio .....	1
Scelta delle API Apache Flink da utilizzare in Managed Service for Apache Flink .....	3
Scelta di un'API Flink .....	3
Nozioni di base .....	4
Come funziona .....	6
Programmazione dell'applicazione Apache Flink .....	6
API DataStream .....	6
API Table .....	7
Creazione dell'applicazione del servizio gestito per Apache Flink .....	8
Creazione di applicazioni .....	8
Creazione del codice dell'applicazione del servizio gestito per Apache Flink .....	8
Creazione dell'applicazione del servizio gestito per Apache Flink .....	10
Avvio dell'applicazione del servizio gestito per Apache Flink .....	11
Verifica dell'applicazione del servizio gestito per Apache Flink .....	11
Esecuzione delle applicazioni .....	12
Status dell'applicazione e del processo .....	12
Carichi di lavoro in batch .....	14
Risorse dell'applicazione .....	14
Risorse dell'applicazione del servizio gestito per Apache Flink .....	14
Risorse applicative Apache Flink .....	15
API DataStream .....	16
DataStream Connettori API .....	16
Operatori dell'API DataStream .....	30
Timestamp dell'API DataStream .....	31
API Table .....	32
Connettori dell'API Table .....	32
Attributi temporali dell'API Table .....	34
Uso di Python .....	34
Programmazione di un'applicazione .....	35
Creazione di un'applicazione .....	38
Monitoraggio .....	39
Proprietà di runtime .....	41

Utilizzo delle proprietà di runtime nella console .....	41
Utilizzo delle proprietà di runtime nella CLI .....	42
Accesso alle proprietà di runtime in un'applicazione del servizio gestito per Apache Flink .....	44
Tolleranza agli errori .....	45
Configurazione della creazione di checkpoint .....	46
Esempi di API per la creazione di checkpoint .....	47
Snapshot .....	49
Dimensionamento .....	55
Configurazione del parallelismo delle applicazioni e della KPU ParallelismPer .....	55
Allocazione delle unità di elaborazione Kinesis .....	56
Aggiornamento del parallelismo dell'applicazione .....	57
Dimensionamento automatico .....	58
Tagging .....	60
Aggiunta di tag quando viene creata un'applicazione .....	61
Aggiunta o aggiornamento di tag per un'applicazione esistente .....	62
Elenco dei tag per un'applicazione .....	62
Rimozione di tag da un'applicazione .....	63
Utilizzo di CloudFormation con il servizio gestito per Apache Flink .....	63
Prima di iniziare .....	63
Scrittura di una funzione Lambda .....	63
Creazione di un ruolo Lambda .....	65
Richiamo della funzione Lambda .....	66
Richiamo della funzione Lambda .....	66
Pannello di controllo di Apache Flink .....	72
Accesso al pannello di controllo Apache Flink dell'applicazione .....	73
Versioni di rilascio .....	75
Servizio gestito da Amazon per Apache Flink versione 1.15.2 .....	75
Modifiche al servizio gestito da Amazon per Apache Flink con Apache Flink 1.15 .....	76
Componenti .....	77
Notebook Studio .....	79
Creazione di un notebook Studio .....	80
Analisi interattiva dei dati di streaming .....	81
Interpreti Flink .....	82
Variabili dell'ambiente tabellare Apache Flink .....	83
Implementazione come applicazione con stato durevole .....	83
Criteri Scala/Python .....	85

Criteri SQL .....	85
Autorizzazioni IAM .....	86
Connettori e dipendenze .....	86
Connettori predefiniti .....	86
Dipendenze e connettori personalizzati .....	88
Funzioni definite dall'utente .....	89
Considerazioni sulle funzioni definite dall'utente .....	90
Abilitare la creazione di checkpoint .....	92
Impostazione dell'intervallo di checkpoint .....	92
Impostazione del tipo di checkpoint .....	92
Utilizzo di AWS Glue .....	92
Proprietà tabella .....	93
Esempi e tutorial .....	95
Tutorial sulla creazione di un notebook Studio .....	95
Tutorial sull'implementazione come applicazione con stato durevole .....	115
Esempi .....	118
Risoluzione dei problemi .....	131
Arresto di un'applicazione bloccata .....	131
Implementazione come applicazione con stato durevole in un VPC senza accesso a Internet .....	131
eploy-as-app Dimensione D e riduzione del tempo di costruzione .....	132
Annullare processi .....	134
Riavvio dell'interprete Apache Flink .....	135
Appendice: creazione di policy IAM personalizzate .....	135
AWS Glue .....	136
CloudWatch Registri .....	136
Flussi Kinesis .....	137
Cluster Amazon MSK .....	140
Guida introduttiva (API) DataStream .....	141
Componenti dell'applicazione .....	141
Prerequisiti .....	142
Fase 1: impostazione di un account .....	142
Registrarsi per creare un Account AWS .....	142
Creazione di un utente amministratore .....	143
Concessione dell'accesso programmatico .....	144
Fase successiva .....	145

Fase 2: configurazione della AWS CLI .....	146
Fase successiva .....	147
Fase 3: creazione di un'applicazione .....	147
Creazione di due flussi di dati Amazon Kinesis .....	148
Scrittura di record di esempio nel flusso di input .....	148
Download ed esame del codice Java di streaming di Apache Flink .....	150
Compilazione del codice dell'applicazione .....	150
Caricamento del codice Java di streaming di Apache Flink .....	151
Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink .....	152
Fase successiva .....	164
Fase 4: eliminazione .....	164
Eliminazione dell'applicazione del servizio gestito per Apache Flink .....	165
Eliminazione del flusso di dati Kinesis .....	165
Eliminazione del bucket e degli oggetti Amazon S3 .....	165
Eliminazione delle risorse IAM .....	165
CloudWatch Elimina le tue risorse .....	166
Fase successiva .....	166
Fase 5: passaggi successivi .....	166
Nozioni di base (API Table) .....	168
Componenti dell'applicazione .....	168
Prerequisiti .....	169
Crea un'applicazione .....	169
Creazione delle risorse dipendenti .....	169
Scrittura di record di esempio nel flusso di input .....	171
Download ed esame del codice Java di streaming di Apache Flink .....	172
Compilazione del codice dell'applicazione .....	174
Caricamento del codice Java di streaming di Apache Flink .....	174
Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink .....	175
Fase successiva .....	180
Eliminazione .....	180
Eliminazione dell'applicazione del servizio gestito per Apache Flink .....	180
Eliminazione del cluster Amazon MSK .....	181
Eliminazione del VPC .....	181
Eliminazione del bucket e degli oggetti Amazon S3 .....	181
Eliminazione delle risorse IAM .....	181
CloudWatch Elimina le tue risorse .....	182

Fase successiva .....	182
Fasi successive .....	182
Nozioni di base (Python) .....	183
Nozioni di base su Pyflink: l'interprete Python per Apache   Amazon Web Services .....	183
Componenti dell'applicazione .....	183
Prerequisiti .....	184
Crea un'applicazione .....	184
Creazione delle risorse dipendenti .....	185
Scrittura di record di esempio nel flusso di input .....	186
Download ed esame del codice Java di streaming Apache Flink .....	188
Aggiunta di dipendenze di terze parti alle app Python .....	189
Caricamento del codice Python di streaming di Apache Flink .....	191
Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink .....	192
Fase successiva .....	197
Eliminazione .....	197
Eliminazione dell'applicazione del servizio gestito per Apache Flink .....	198
Eliminazione del flusso di dati Kinesis .....	198
Eliminazione del bucket e degli oggetti Amazon S3 .....	198
Eliminazione delle risorse IAM .....	199
CloudWatch Elimina le tue risorse .....	199
Nozioni di base (Scala) .....	200
Creazione delle risorse dipendenti .....	200
Scrittura di record di esempio nel flusso di input .....	201
Download ed esame del codice dell'applicazione .....	203
Per creare e compilare il codice dell'applicazione .....	204
Creazione ed esecuzione dell'applicazione (console) .....	205
Creazione dell'applicazione .....	205
Configurazione dell'applicazione .....	206
Modifica della policy IAM .....	208
Esecuzione dell'applicazione .....	209
Interruzione dell'applicazione .....	210
Creazione ed esecuzione dell'applicazione (CLI) .....	210
Creazione di una policy di autorizzazione .....	210
Creazione di una policy IAM .....	212
Creazione dell'applicazione .....	213
Avvio dell'applicazione .....	214

Interruzione dell'applicazione .....	215
Aggiungi un'opzione di registrazione CloudWatch .....	216
Aggiornamento delle proprietà di ambiente .....	216
Aggiornamento del codice dell'applicazione .....	217
Pulizia .....	218
Eliminazione dell'applicazione del servizio gestito per Apache Flink .....	218
Eliminazione dei flussi di dati Kinesis .....	218
Eliminazione del bucket e degli oggetti Amazon S3 .....	219
Eliminazione delle risorse IAM .....	219
Eliminazione delle risorse CloudWatch .....	219
Utilizzo di Apache Beam .....	220
Utilizzo di Apache Beam con il servizio gestito per Apache Flink .....	220
Funzionalità Beam .....	220
Creazione di un'applicazione utilizzando Apache Beam .....	221
Creazione delle risorse dipendenti .....	221
Scrittura di record di esempio nel flusso di input .....	222
Download ed esame del codice dell'applicazione .....	223
Compilazione del codice dell'applicazione .....	224
Caricamento del codice Java di streaming di Apache Flink .....	225
Creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink .....	225
Eliminazione .....	229
Fasi successive .....	230
Workshop di formazione, laboratori e implementazioni di soluzioni .....	232
Sviluppo a livello locale di applicazioni Apache Flink prima dell'implementazione sul servizio gestito per Apache Flink per Apache Flink .....	232
Rilevamento di eventi con il servizio gestito per Apache Flink Studio .....	232
Soluzione per lo streaming di dati AWS .....	233
Laboratorio clickstream .....	233
Dimensionamento personalizzato .....	233
CloudWatch Cruscotto .....	234
MSK Amazon .....	234
Altro servizio gestito per soluzioni Apache Flink su GitHub .....	234
Utilità .....	235
Snapshot manager .....	235
Analisi comparativa .....	235
Esempi .....	236



DataStream Esempi di API .....	236
Finestra a cascata .....	237
Finestra scorrevole .....	246
Sink S3 .....	256
Replica MSK .....	270
Consumatore EFO .....	277
Sink Kinesis Data Firehose .....	288
Multi-account .....	304
Truststore personalizzato .....	313
Esempi di Python .....	322
Finestra a cascata .....	322
Finestra scorrevole .....	333
Sink S3 .....	343
Esempi di Scala .....	355
Finestra a cascata .....	355
Finestra scorrevole .....	372
Sink S3 .....	390
Sicurezza .....	407
Protezione dei dati .....	408
Crittografia dei dati .....	408
Identity and Access Management .....	409
Destinatari .....	409
Autenticazione con identità .....	410
Gestione dell'accesso con policy .....	414
Funzionamento del servizio gestito da Amazon per Apache Flink con IAM .....	416
Esempi di policy basate su identità .....	424
Risoluzione dei problemi .....	427
Prevenzione del problema "confused deputy" tra servizi .....	429
Monitoraggio .....	431
Convalida della conformità .....	431
FedRAMP .....	432
Resilienza .....	432
Ripristino di emergenza .....	432
Controllo delle versioni .....	433
Sicurezza dell'infrastruttura .....	433
Best practice di sicurezza .....	434

Implementazione dell'accesso con privilegi minimi .....	434
Uso di ruoli IAM per accedere ad altri servizi Amazon .....	434
Implementazione della crittografia lato server in risorse dipendenti .....	435
Utilizzato per monitorare CloudTrail le chiamate API .....	435
Registrazione e monitoraggio .....	436
Registrazione .....	437
Interrogazione dei log con Logs Insights CloudWatch .....	437
Monitoraggio .....	437
Configurazione della registrazione .....	439
Configurazione della CloudWatch registrazione tramite la console .....	440
Configurazione della CloudWatch registrazione tramite la CLI .....	440
Livelli di monitoraggio dell'applicazione .....	445
Registrazione di best practice .....	446
Risoluzione dei problemi relativi alla registrazione .....	447
Fase successiva .....	447
Analisi dei log .....	447
Esecuzione di una query di esempio .....	448
Query di esempio .....	449
Metriche e dimensioni nel servizio gestito per Apache Flink .....	451
Parametri di applicazioni .....	452
Parametri del connettore del flusso di dati Kinesis .....	482
Parametri del connettore Amazon MSK .....	483
Parametri di Apache Zeppelin .....	485
Visualizzazione delle metriche CloudWatch .....	486
Metriche .....	487
Parametri personalizzati .....	489
Allarmi .....	493
Scrittura di messaggi personalizzati .....	505
Scrivi nei CloudWatch log usando Log4J .....	505
Scrivi nei log usando CloudWatch SLF4J .....	506
Uso di AWS CloudTrail .....	507
Servizio gestito per Apache Flink Information in CloudTrail .....	507
Informazioni sulle voci del file di log del servizio gestito da per Apache Flink .....	508
Prestazioni .....	511
Risoluzione dei problemi relativi alle prestazioni .....	511
Il percorso dei dati .....	511

Soluzioni per la risoluzione dei problemi .....	512
Best practice per ottimizzare le prestazioni .....	514
Gestire correttamente il dimensionamento .....	514
Monitoraggio dell'utilizzo delle risorse di dipendenza esterne .....	517
Avviamento dell'applicazione Apache Flink in locale .....	517
Monitoraggio delle prestazioni .....	517
Monitoraggio delle prestazioni con le metriche CloudWatch .....	517
Monitoraggio delle prestazioni con i file di log e gli allarmi CloudWatch .....	517
Quota .....	519
Maintenance (Manutenzione) .....	521
Impostare un UUID per tutti gli operatori .....	523
Preparazione per la produzione .....	524
Applicazioni di test del carico .....	524
Parallelismo massimo .....	524
Impostare un UUID per tutti gli operatori .....	525
Best practice .....	526
Tolleranza agli errori: checkpoint e savepoint .....	526
Versioni di connettori non supportate .....	527
Prestazioni e parallelismo .....	527
Impostazione del parallelismo per operatore .....	528
Registrazione .....	529
Codifica .....	529
Credenziali root. ....	530
Lettura da fonti con pochi shard/partizioni .....	530
Intervallo di aggiornamento del notebook Studio .....	531
Prestazioni ottimali del notebook Studio .....	531
Come le strategie di watermark e le partizioni inattive influiscono sulle finestre temporali .....	531
Riepilogo .....	533
Esempio .....	533
Impostare un UUID per tutti gli operatori .....	542
Aggiungi al plugin ServiceResourceTransformer Maven Shade .....	543
Apache Flink Stateful Functions .....	544
Modello di applicazione Apache Flink .....	544
Ubicazione della configurazione del modulo .....	545
Versioni precedenti .....	546
Utilizzo del connettore di flussi Apache Flink Kinesis con versioni precedenti di Apache Flink ...	547

Creazione di applicazioni con Apache Flink 1.8.2 .....	548
Creazione di applicazioni con Apache Flink 1.6.2 .....	549
Aggiornamento di applicazioni .....	550
Connettori disponibili in Apache Flink 1.6.2 e 1.8.2 .....	550
Nozioni di base: Flink 1.13.2 .....	550
Componenti dell'applicazione .....	551
Prerequisiti .....	552
Fase 1: impostazione di un account .....	552
Fase successiva .....	555
Fase 2: configurazione della AWS CLI .....	555
Fase 3: creazione di un'applicazione .....	557
Fase 4: eliminazione .....	574
Fase 5: passaggi successivi .....	575
Nozioni di base: Flink 1.11.1 .....	576
Componenti dell'applicazione .....	577
Prerequisiti .....	578
Fase 1: impostazione di un account .....	578
Fase 2: configurazione della AWS CLI .....	581
Fase 3: creazione di un'applicazione .....	583
Fase 4: eliminazione .....	600
Fase 5: passaggi successivi .....	602
Nozioni di base: Flink 1.8.2 .....	603
Componenti dell'applicazione .....	141
Prerequisiti .....	604
Fase 1: impostazione di un account .....	604
Fase 2: configurazione della AWS CLI .....	607
Fase 3: creazione di un'applicazione .....	609
Fase 4: eliminazione .....	626
Nozioni di base: Flink 1.6.2 .....	628
Componenti dell'applicazione .....	628
Prerequisiti .....	629
Fase 1: impostazione di un account .....	629
Fase 2: configurazione della AWS CLI .....	632
Fase 3: creazione di un'applicazione .....	634
Fase 4: eliminazione .....	651
Impostazioni di Flink .....	653

Configurazione di Apache Flink .....	653
Backend di stato .....	654
Checkpoint .....	654
Savepoint .....	656
Dimensioni heap .....	656
Debloating del buffer .....	656
Proprietà di configurazione modificabili di Flink .....	657
Tolleranza agli errori .....	657
Checkpoint e backend di stato .....	657
Checkpoint .....	657
Metriche native di RocksDB .....	657
Opzioni avanzate di backend di stato .....	659
Opzioni complete di TaskManager .....	659
Configurazione della memoria .....	659
RPC/Akka .....	660
Client .....	660
Opzioni avanzate per il cluster .....	660
Configurazioni del filesystem .....	660
Opzioni avanzate di tolleranza agli errori .....	661
Configurazione della memoria .....	659
Parametri .....	661
Opzioni avanzate per client ed endpoint REST .....	661
Opzioni di sicurezza SSL avanzate .....	661
Opzioni di pianificazione avanzate .....	661
Opzioni avanzate per l'interfaccia utente web di Flink .....	661
Visualizzazione delle proprietà Flink configurate .....	661
Utilizzare un Amazon VPC .....	663
Nozioni di Amazon VPC .....	663
Autorizzazioni per applicazioni VPC .....	664
Politica di autorizzazione per accedere a un Amazon VPC .....	664
Accesso a Internet e ai servizi .....	666
Informazioni correlate .....	667
API VPC .....	667
Crea applicazione .....	667
Aggiungi configurazione VPC all'applicazione .....	668
Elimina configurazione VPC applicazione .....	669

Aggiorna applicazione .....	669
Esempio: utilizzare un VPC .....	670
Risoluzione dei problemi .....	671
Risoluzione dei problemi di sviluppo .....	671
Grafici a fiamma di Apache Flink .....	671
Problema del fornitore di credenziali con il connettore EFO 1.15.2 .....	672
Applicazioni con connettori Kinesis non supportati .....	672
Errore di compilazione: "Impossibile risolvere le dipendenze per il progetto" .....	675
Scelta non valida: "kinesisanalyticsv2" .....	675
UpdateApplication L'azione non è ricaricare il codice dell'applicazione .....	675
S3 StreamingFileSink FileNotFoundException .....	676
FlinkKafkaConsumer problema con stop with savepoint .....	678
Blocco di Async sink per Flink 1.15 .....	678
L'elaborazione della fonte del flusso di dati Amazon Kinesis non funziona correttamente durante la ripartizione .....	688
Risoluzione dei problemi di runtime .....	689
Strumenti per la risoluzione dei problemi .....	689
Problematiche delle applicazioni .....	689
L'applicazione si riavvia .....	694
La velocità di trasmissione effettiva è troppo lenta .....	697
Crescita illimitata dello stato .....	698
Operatori vincolati all'I/O .....	699
Limitazione in upstream o all'origine da un flusso di dati Kinesis .....	700
Checkpoint .....	701
Checkpoint in fase di interruzione .....	707
Errori del checkpoint (Beam) .....	709
Contropressione .....	711
Disallineamento dei dati .....	712
Disallineamento dello stato .....	713
Integrazione con risorse in regioni differenti .....	714
Cronologia dei documenti .....	715
Codice di esempio dell'API .....	721
AddApplicationCloudWatchLoggingOption .....	722
AddApplicationInput .....	722
AddApplicationInputProcessingConfiguration .....	723
AddApplicationOutput .....	724

---

AddApplicationReferenceDataSource .....	724
AddApplicationVpcConfiguration .....	725
CreateApplication .....	726
CreateApplicationSnapshot .....	727
DeleteApplication .....	727
DeleteApplicationCloudWatchLoggingOption .....	727
DeleteApplicationInputProcessingConfiguration .....	728
DeleteApplicationOutput .....	728
DeleteApplicationReferenceDataSource .....	728
DeleteApplicationSnapshot .....	729
DeleteApplicationVpcConfiguration .....	729
DescribeApplication .....	729
DescribeApplicationSnapshot .....	729
DiscoverInputSchema .....	730
ListApplications .....	730
ListApplicationSnapshots .....	731
StartApplication .....	731
StopApplication .....	731
UpdateApplication .....	732
Documentazione di riferimento delle API .....	733

Il servizio gestito 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 il servizio gestito da Amazon 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.

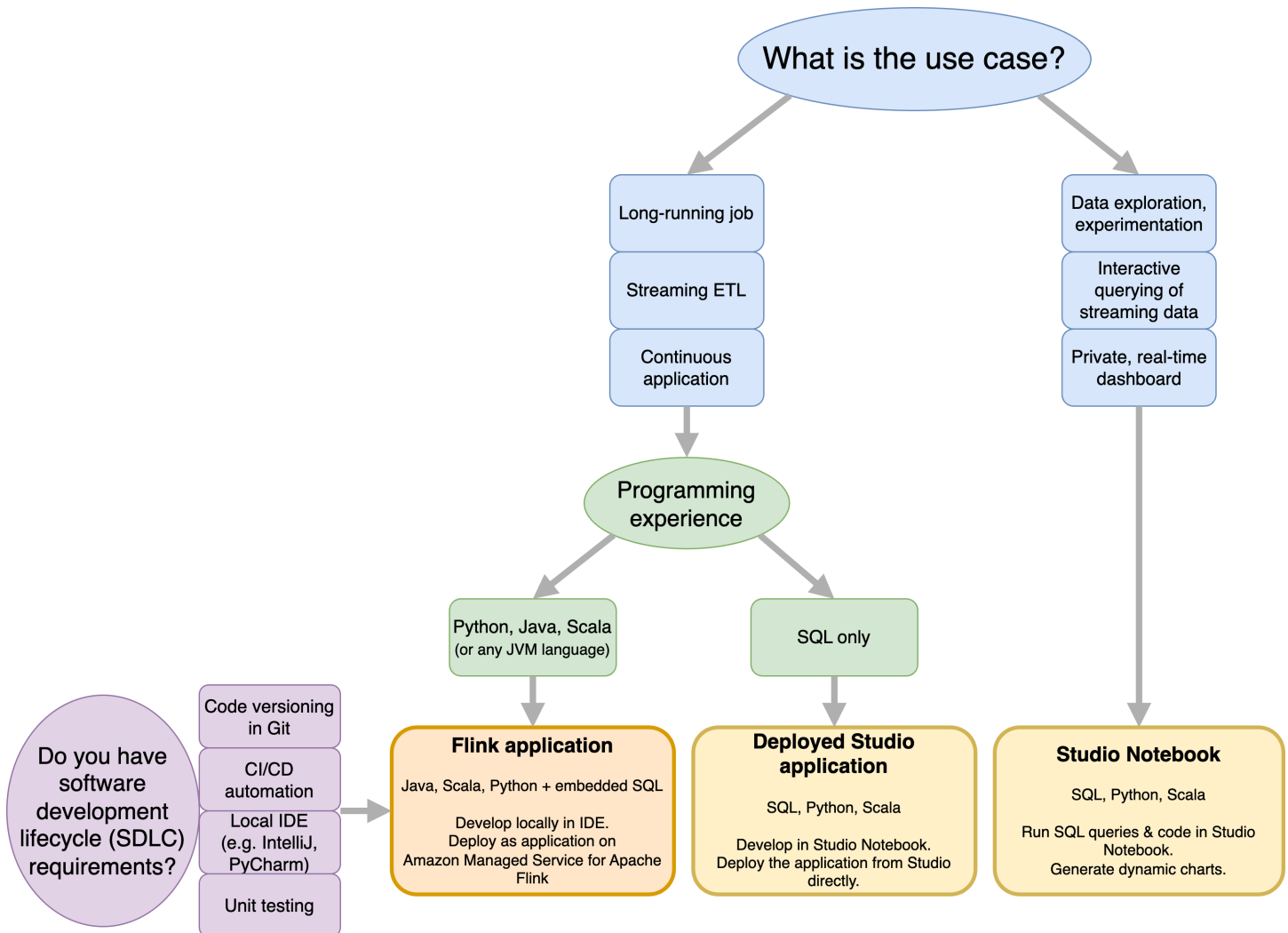
[Puoi creare applicazioni con il linguaggio che preferisci in Managed Service for Apache Flink utilizzando librerie open source basate su Apache Flink.](#) Apache Flink è un celebre framework e motore per l'elaborazione di flussi di dati.

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.

## Scelta del servizio gestito per Apache Flink o del servizio gestito 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 le API Apache Flink Datastream o Table. 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 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 per Apache Flink.](#) Questo perché potete creare la vostra applicazione Flink utilizzando le API Flink 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.

## Scelta delle API Apache Flink da utilizzare in Managed Service for Apache Flink

Puoi creare applicazioni utilizzando Java, Python e Scala in Managed Service for Apache Flink utilizzando le API Apache Flink in un IDE di tua scelta. [Puoi trovare indicazioni su come creare applicazioni utilizzando le API Flink Datastream e Table nella documentazione.](#) È possibile selezionare la lingua in cui creare l'applicazione Flink e le API da 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.

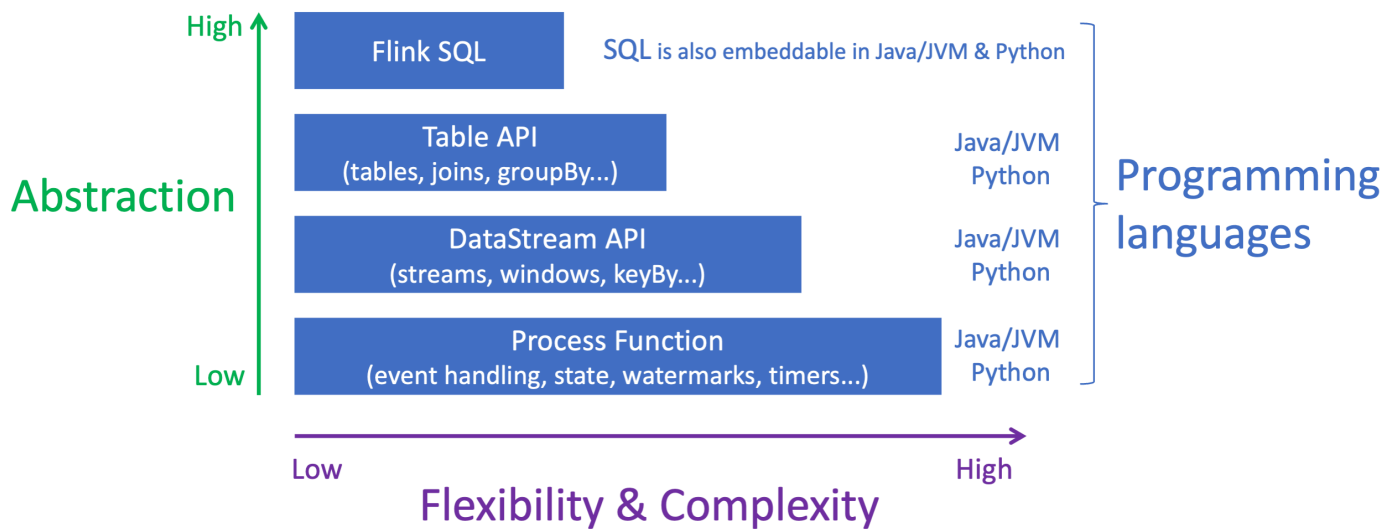
### Scelta di un'API Flink

Le API Apache Flink hanno diversi livelli di astrazione che possono influire sul modo in cui decidi di creare l'applicazione. Sono espressive e flessibili e possono essere utilizzate insieme per creare un'applicazione. Non è necessario utilizzare una sola API Flink. [Puoi saperne di più sulle API Flink 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)
- Vuoi usare timer personalizzati

## Apache Flink APIs



### Note

Scelta di una lingua con l'API Datastream:

- 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 bassa latenza/alta velocità di trasmissione, dovresti prendere in considerazione Java/Scala indipendentemente dall'API.
- Se prevedi di utilizzare Async IO nell'API Process Functions, dovrai usare Java.

## Nozioni di base

Innanzitutto, è consigliabile creare un'applicazione di 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:

- [Il servizio gestito da Amazon per Apache Flink: come funziona](#)
- [Guida introduttiva ad 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 nella 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:

- [Utilizzo di un notebook Studio con il servizio gestito per Apache Flink](#)
- [Creazione di un notebook Studio](#)

# Il servizio gestito da Amazon per Apache Flink: come funziona

Il servizio gestito per Apache Flink è un servizio completamente gestito da Amazon che consente di creare e gestire un'applicazione Apache Flink per elaborare i flussi di dati.

## Programmazione dell'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'[API DataStream](#) o l'[API Table](#). Sono disponibili anche le altre API Apache Flink, ma sono utilizzate meno comunemente nella creazione di applicazioni di streaming.

Le funzionalità delle due API sono descritte di seguito:

### API DataStream

Il modello di programmazione dell'API DataStream di 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'API DataStream 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'API DataStream possono essere scritte in Java o Scala e possono essere lette da un flusso di dati Kinesis, un argomento 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'uso dell'API DataStream, consulta [API DataStream](#).

## 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 [API Table](#).

# Creazione dell'applicazione del servizio gestito per Apache Flink

Il servizio gestito per Apache Flink è un servizio AWS che crea l'ambiente per ospitare la tua applicazione Apache Flink e fornisce le seguenti impostazioni:

- [Proprietà di runtime](#): parametri che puoi fornire alla tua applicazione. È possibile modificare questi parametri senza ricompilare il codice dell'applicazione.
- [Tolleranza agli errori](#): in che modo l'applicazione viene ripristinata dopo interruzioni e riavvii.
- [Registrazione e monitoraggio](#): in che modo l'applicazione registra eventi sui file di log di CloudWatch.
- [Dimensionamento](#): 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 [Guida introduttiva \(API\) DataStream](#).

## Creazione di un'applicazione del servizio gestito per Apache Flink

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

Questo argomento contiene le sezioni seguenti:

- [Creazione del codice dell'applicazione del servizio gestito per Apache Flink](#)
- [Creazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Avvio dell'applicazione del servizio gestito per Apache Flink](#)
- [Verifica dell'applicazione del servizio gestito per Apache Flink](#)

## Creazione del codice dell'applicazione del servizio gestito per Apache Flink

Questa sezione descrive i componenti utilizzati per creare il codice dell'applicazione per un'applicazione del servizio gestito per Apache Flink.

Per il codice dell'applicazione, ti consigliamo di utilizzare la versione di Apache Flink supportata più recente. La versione di Apache Flink più recente supportata dal servizio gestito per Apache Flink è la 1.15.2. Per informazioni sull'aggiornamento delle applicazioni del servizio gestito per Apache Flink, consulta [Aggiornamento di applicazioni](#).



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.

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

Componente	Versione
Java	11 (consigliata)
Scala	Vedi la nota sul disaccoppiamento di Scala riportata di seguito
Servizio gestito per Apache Flink Runtime () <code>aws-kinesisanalytics-runtime</code>	1.2.0
<a href="#">AWSConnettore Kinesis () flink-connector-kinesis</a>	1.15.2
Apache Beam (solo applicazioni Beam)	2.33.0, con la versione Jackson 2.12.2

A partire dalla versione 1.15, Flink non supporta più Scala. Le applicazioni possono ora utilizzare l'API Java da qualsiasi versione di Scala. Dovrai raggruppare la libreria standard Scala che hai scelto nelle tue applicazioni Scala.

Per un esempio di file `pom.xml` per un'applicazione del servizio gestito per Apache Flink che utilizza la versione 1.15.2 di Apache Flink, consulta l'[applicazione introduttiva del servizio gestito per Apache Flink](#).

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

## Specificare la versione di 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. Dovrai fornire il parametro `-Dflink.version` alla versione di Apache Flink nel modo seguente:

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

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

## Creazione dell'applicazione del servizio gestito per Apache Flink

Dopo aver creato il codice dell'applicazione, procedi come segue per creare l'applicazione del servizio gestito per 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 istruzioni su come caricare il codice dell'applicazione, consulta [the section called "Caricamento del codice Java di streaming di Apache Flink"](#) nel tutorial [Guida introduttiva \(API\) DataStream](#).
- Creazione dell'applicazione del servizio gestito per Apache Flink: utilizza uno dei seguenti metodi per creare l'applicazione del servizio gestito per Apache Flink:
  - Creazione dell'applicazione del servizio gestito per Apache Flink utilizzando la console AWS: puoi creare e configurare l'applicazione utilizzando la console AWS.

Quando crei un'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 istruzioni sull'utilizzo della console per creare applicazioni, consulta [the section called "Creazione ed esecuzione dell'applicazione \(console\)"](#) nel tutorial [Guida introduttiva \(API\) DataStream](#).

- Creazione dell'applicazione del servizio gestito per Apache Flink utilizzando l'AWS CLI: puoi creare e configurare l'applicazione utilizzando l'AWS CLI.

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`.

Per istruzioni sull'utilizzo della CLI per creare applicazioni, consulta [the section called “Creazione ed esecuzione di un'applicazione utilizzando la CLI”](#) nel tutorial [Guida introduttiva \(API\) DataStream](#).

#### Note

Non è possibile modificare il `RuntimeEnvironment` di un'applicazione esistente. Per modificare il `RuntimeEnvironment` di un'applicazione esistente, è necessario eliminare l'applicazione e crearla nuovamente.

## Avvio dell'applicazione del servizio gestito per 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:

- Avvio dell'applicazione del servizio gestito per Apache Flink utilizzando la console AWS: è possibile eseguire l'applicazione scegliendo **Esegui** nella pagina dell'applicazione nella console AWS.
- Avvia l'applicazione Managed Service for Apache Flink utilizzando l'AWSAPI: puoi eseguire l'applicazione utilizzando l'azione. [StartApplication](#)

## Verifica dell'applicazione del servizio gestito per 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](#)

- 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 Kinesis Data 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.

## Esecuzione di un'applicazione di servizio gestito per 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 processi di Apache Flink, consulta [Processi e pianificazione](#) nella [documentazione di Apache Flink](#).

## Status dell'applicazione e del processo

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 un'applicazione rimane in uno stato transitorio per

un periodo di tempo prolungato, è possibile interromperla utilizzando l'azione [Interrompi applicazione](#) con il parametro `Force` impostato su `true`. Questi status includono le seguenti opzioni:

- **STARTING**: si verifica dopo l'azione [Avvia applicazione](#). L'applicazione passa dallo status `READY` a `RUNNING`.
- **STOPPING**: si verifica dopo l'azione [Arresta applicazione](#). L'applicazione passa dallo status `RUNNING` a `READY`.
- **DELETING**: si verifica dopo l'azione [Cancella applicazione](#). L'applicazione è in fase di eliminazione.
- **UPDATING**: si verifica dopo l'azione [Aggiorna applicazione](#). L'applicazione si sta aggiornando e tornerà allo status `RUNNING` o `READY`.
- **AUTOSCALING**: la proprietà `AutoScalingEnabled` della [Configurazione del parallelismo](#) dell'applicazione è 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'azione [Interrompi applicazione](#) con il parametro `Force` impostato su `true`. Per ulteriori informazioni sul dimensionamento automatico, consulta [Dimensionamento automatico](#).
- **FORCE\_STOPPING**: si verifica dopo l'esecuzione dell'azione [Arresta applicazione](#) con il parametro `Force` 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 l'esecuzione dell'azione [Ripristina applicazione](#). È in corso il ripristino dell'applicazione a una versione precedente. L'applicazione passa dallo status `UPDATING` o `AUTOSCALING` a `RUNNING`.
- In seguito al corretto ripristino di un'applicazione, **ROLLED\_BACK**: questo diventa lo status della versione da cui è stato eseguito il ripristino. Per informazioni sul ripristino di un'applicazione, consulta [Ripristina applicazione](#).
- **MAINTENANCE**: si verifica quando il servizio gestito per Apache Flink applica le patch all'applicazione. Per ulteriori informazioni, consulta [Maintenance \(Manutenzione\)](#).

È possibile controllare lo status dell'applicazione tramite la console o l'azione [Descrivi applicazione](#).

- **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.

È possibile controllare lo stato del lavoro esaminando il log CloudWatch dell'applicazione per verificare eventuali modifiche allo stato.

## 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](#).

## Risorse dell'applicazione

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.

## Risorse dell'applicazione del servizio gestito per Apache Flink

Servizio gestito per Apache Flink è un servizio AWS che crea un ambiente per l'hosting dell'applicazione Apache Flink. Il servizio gestito per Apache Flink fornisce risorse utilizzando unità denominate Kinesis Processing Units (KPU), ossia unità di elaborazione Kinesis.

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.

Le KPU eseguono le applicazioni in unità di esecuzione distinte denominate attività e attività secondarie. Un'attività secondaria rappresenta l'equivalente di un thread.

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

Per ulteriori informazioni sul parallelismo delle applicazioni, consulta [Dimensionamento](#).

## Risorse applicative 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, consulta [Pianificazione dei processi](#) 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 sulla concatenazione degli operatori, consulta [Concatenazione delle attività e gruppi di risorse](#) nella [documentazione di Apache Flink](#).

### Concatenazione 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.

## API DataStream

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

Questa sezione contiene gli argomenti seguenti:

- [Utilizzo dei connettori per spostare i dati nel servizio gestito per Apache Flink con l'API DataStream](#) : questi componenti spostano i dati tra l'applicazione e le origini e destinazioni di dati esterne.
- [Trasformazione dei dati utilizzando gli operatori nel servizio gestito per Apache Flink con l'API DataStream](#): questi componenti trasformano o raggruppano gli elementi di dati all'interno dell'applicazione.
- [Tracciamento degli eventi nel servizio gestito per Apache Flink utilizzando l'API DataStream](#): questo argomento descrive il modo in cui il servizio gestito per Apache Flink tiene traccia degli eventi quando viene utilizzata l'API DataStream.

## Utilizzo dei connettori per spostare i dati nel servizio gestito per Apache Flink con l'API DataStream

Nell' API DataStream 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 da 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:

- [Origini](#): invio di dati all'applicazione da un flusso di dati, un file o un'altra origine dati Kinesis.



- [Sink](#): invio di dati dall'applicazione a un flusso di dati Kinesis, a un flusso Kinesis Data Firehose o ad altra destinazione dati.
- [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 di Medio Oriente (Emirati Arabi Uniti), Asia Pacifico (Hyderabad), Israele (Tel Aviv), Europa (Zurigo), Asia Pacifico (Melbourne) o Asia Pacifico (Giacarta), potrebbe essere necessario ricostruire l'archivio delle applicazioni con un connettore aggiornato o passare a Flink 1.15. Di seguito sono riportate le linee guida consigliate:

#### Aggiornamenti dei connettori

Versione di Flink	Connettore usato	Risoluzione
1, - 1,	Firehose	L'applicazione dipende da una versione obsoleta del connettore e Firehose che non

V di Fl	Connettore usato	Risoluzio ne
		riconosce le regioni AWS più recenti. Ricostruisce l'archivio o delle applicazioni con il connettore e Firehose versione 2.1.0. <a href="#">v2.1.0</a>

V di Fl	Connettore usato	Risoluzione
1.	Kinesis	L'applicazione dipende da una versione obsoleta del connettore e Flink Kinesis che non riconosce le regioni AWS più recenti. Ricostruisce l'archivio dell'applicazione con il connettore e Flink Kinesis

V di Fl	Connettore usato	Risoluzio ne
		versione 1.6.1. <a href="https://github.com/aws-labs/flink-connector-aws-kinesis-conector-flink">https:// g ithub.com / awslabs/ tree/1.6 .1 amazon- ki nesis- con nector- fl ink</a>

V di Fl	Connettore usato	Risoluzione
1.	Kinesis	L'applicazione dipende da una versione obsoleta del connettore Flink Kinesis che non riconosce le regioni AWS più recenti. Ricostruisce l'archivio dell'applicazione con il connettore Flink Kinesis

V di Fl	Connettore usato	Risoluzio ne
		versione 2.4.1. <a href="https://github.com/aws-labs/amazon-ki-nesis-connector-flink/tree/2.4.1">https:// g ithub.com / awslabs/ amazon- ki nesis- con nector- fl ink / tree/2.4 .1</a>

V di Fl	Connettore usato	Risoluzio ne
1. e 1.	Kinesis	L'applica zione dipende da una versione obsoleta del connettor e Flink Kinesis che non riconosce le regioni AWS più recenti. Sfortunat amente, Flink non rilascia più patch o correzion i di bug

V di Fl	Connettore usato	Risoluzione
		per i connettori i 1.6/1.13. Ti consigliamo di eseguire l'aggiornamento a Flink 1.15 ricostruendo l'archivio o dell'applicazione con Flink 1.15.

## Aggiunta di origini dati di streaming al servizio gestito per 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.



## Flussi di dati Kinesis

L'origine `FlinkKinesisConsumer` fornisce dati di streaming all'applicazione da un flusso di dati Amazon Kinesis.

### Creazione di un `FlinkKinesisConsumer`

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

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

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

Per ulteriori informazioni sull'utilizzo di una `FlinkKinesisConsumer`, consulta [Download ed esame del codice Java di streaming di Apache Flink](#).

### Creazione di un `FlinkKinesisConsumer` che utilizza un consumatore EFO

`FlinkKinesisConsumer` Ora 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: fan-out avanzato per consumatori AWS Kinesis](#).

È possibile 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 consumatore nello stesso flusso di dati Kinesis causerà l'interruzione del precedente consumatore che utilizzava quel nome.

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

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

Per un esempio di un'applicazione del servizio gestito per Apache Flink che utilizza un consumatore EFO, consulta [Consumatore EFO](#).

## MSK Amazon

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](#).

## Scrittura di dati mediante i sink nel servizio gestito per Apache Flink

Nel codice dell'applicazione, utilizzi un [sink Apache Flink](#) per scrivere dati da un flusso Apache Flink a un servizio AWS come il flusso di dati Kinesis.

Apache Flink fornisce sink per file, socket e sink personalizzati. In AWS sono disponibili i seguenti sink:

### 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 [Guida introduttiva \(API\) `DataStream`](#).

## 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”](#).

## Kinesis Data Firehose

Il `FlinkKinesisFirehoseProducer` è un sink Apache Flink affidabile e scalabile per l'archiviazione dell'output delle applicazioni utilizzando il servizio [Kinesis Data 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 `FlinkKinesisFirehoseProducer` e inviarlo da un flusso di dati Apache Flink al servizio Kinesis Data Firehose.

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

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

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

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

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

public class StreamingJob {

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

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

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

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

    private static FlinkKinesisFirehoseProducer<String>
    createFirehoseSinkFromStaticConfig() {
        /*
         * com.amazonaws.services.kinesisanalytics.flink.connectors.config.
```

```
* ProducerConfigConstants
* lists of all of the properties that firehose sink can be configured with.
*/

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

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

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

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

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

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

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

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

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

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

Per un tutorial completo su come utilizzare il sink Kinesis Data Firehose, consulta [the section called “Sink Kinesis Data Firehose”](#).

## Utilizzo dell'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](#).

## Trasformazione dei dati utilizzando gli operatori nel servizio gestito per 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 l'argomento relativo agli [operatori](#) nella [documentazione di Apache Flink](#).

Questo argomento contiene le sezioni seguenti:

- [Operatori di trasformazione](#)
- [Operatori di aggregazione](#)

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

## 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 un esempio di codice completo che utilizza gli operatori, consulta [Guida introduttiva \(API\) DataStream](#). Il codice di origine per l'applicazione introduttiva è disponibile in [Nozioni di base](#) nel repository GitHub degli [esempi Java del servizio gestito per Apache Flink](#).

## Tracciamento degli eventi nel servizio gestito per 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.

È possibile impostare 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 [Ora evento](#) nella [documentazione di Apache Flink](#).

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

## Connettori dell'API Table

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

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

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



## Origini dell'API Table

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 origini delle tabelle, consulta [Tabella e connettori](#) nella [documentazione di Apache Flink](#).

## Sink dell'API Table

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 [Formati](#) nella [documentazione di Apache Flink](#).

Per ulteriori informazioni sui sink di tabella, consulta [Tabella e connettori](#) nella [documentazione di Apache Flink](#).

## Origini 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 di dati, puoi definire fonti e sink personalizzati. Per ulteriori informazioni, consulta [Origini e sink definiti dall'utente](#) nella [documentazione di Apache Flink](#).

## Attributi temporali dell'API Table

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, sei tu a definire quale di questi timestamp utilizza attraverso il metodo [setStreamTimeCharacteristic](#).

Per ulteriori informazioni sull'utilizzo dei timestamp con l'API Table, consulta [Attributi temporali](#) nella [documentazione di Apache Flink](#).

## Uso di Python con il servizio gestito 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 di PyFlink 1.15. In questo

caso consigliamo di eseguire l'interprete Python in Docker. Per istruzioni dettagliate, consulta [Sviluppo di PyFlink 1.15 su Apple Silicon per Mac](#).

La versione 1.15.2 di Apache Flink include il supporto per la creazione di applicazioni utilizzando Python versione 3.8, attraverso la libreria [PyFlink](#). 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 [Introduzione all'API Python Table](#) nella [documentazione di Apache Flink](#).

## Programmazione di un'applicazione del servizio gestito per Apache Flink per 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 Python Table 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](#)

## Letture e scrittura di dati in streaming

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

### Creazione di 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)
```

### Letture di dati in 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))
```

### Scrittura di 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))
```

## Lettura delle 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:

- Utilizzando l'azione [CreateApplication](#).
- Utilizzando l'azione [CreateApplication](#).
- 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]
```

## Creazione del pacchetto di codice dell'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.

## Creazione dell'applicazione del servizio gestito per Apache Flink Python

### Specificazione dei file di codice

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

Quando crei l'applicazione utilizzando l'azione [CreateApplication](#), devi specificare 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 Apache Flink Python, consulta [Utilizzo della riga di comando](#) 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"
        }
      }
    ],
  },
},
```

## Creazione dell'applicazione del servizio gestito per Apache Flink Python

Usa il log CloudWatch della tua applicazione per monitorare l'applicazione del servizio gestito per 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.

## Esecuzione di query sui log tramite Insights di CloudWatch

La seguente query di CloudWatch Insights cerca i log creati da punto di ingresso di Python durante l'esecuzione della funzione principale dell'applicazione:



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

## Proprietà di runtime nel servizio gestito per 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:

- [Utilizzo delle proprietà di runtime nella console](#)
- [Utilizzo delle proprietà di runtime nella CLI](#)
- [Accesso alle proprietà di runtime in un'applicazione del servizio gestito per Apache Flink](#)

### Utilizzo delle proprietà di runtime nella console

È possibile aggiungere, aggiornare o rimuovere le proprietà di runtime dall'applicazione del servizio gestito per Apache Flink utilizzando la console.

#### Note

Non puoi aggiungere proprietà di runtime quando crei un'applicazione nella console del servizio gestito per Apache Flink.

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

1. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/flink>
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.

### Utilizzo delle proprietà di runtime nella 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 del servizio gestito per Apache Flink](#).

#### Note

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

### Aggiunta di 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_15",
  "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::ka-app-code-username",
          "FileKey": "java-getting-started-1.0.jar"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ProducerConfigProperties",
```



```
}  
}  
}
```

### 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.

## Rimozione delle 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.

## Accesso alle proprietà di runtime in un'applicazione del servizio gestito per 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 un esempio di codice completo che utilizza le proprietà di runtime, consulta [Guida introduttiva \(API\) DataStream](#). Il codice di origine per l'applicazione introduttiva è disponibile in [Nozioni di base](#) nel repository GitHub degli [esempi Java del servizio gestito per Apache Flink](#).

## Implementazione della tolleranza agli errori nel servizio gestito per 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 backup aggiornato di un'applicazione in esecuzione che viene utilizzato per il ripristino immediato in caso di interruzione o failover imprevisti dell'applicazione.

Per i dettagli sulla creazione di checkpoint nelle applicazioni Apache Flink, consulta [Checkpoint](#) 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 mediante snapshot](#).

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 gli snapshot sono abilitati per l'applicazione e configurati tramite l'[ApplicationRestoreConfiguration](#) dell'applicazione, il servizio fornisce la semantica di elaborazione esatta durante gli aggiornamenti dell'applicazione o durante il dimensionamento o la manutenzione relativi al servizio.

## Configurazione della creazione di checkpoint nel servizio gestito per 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`: `true`

- `CheckpointInterval`: 60000
- `MinPauseBetweenCheckpoints`: 5000

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

### 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`.

## Esempi di API per la creazione di checkpoint

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 del servizio gestito per Apache Flink](#).

### Configurazione della creazione di 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_15",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::mybucket",
```

```
    "FileKey": "myflink.jar",
    "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
  }
},
"FlinkApplicationConfiguration": {
  "CheckpointConfiguration": {
    "CheckpointingEnabled": "true",
    "CheckpointInterval": 20000,
    "ConfigurationType": "CUSTOM",
    "MinPauseBetweenCheckpoints": 10000
  }
}
}
```

## Disabilitare la creazione di 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_15",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::mybucket",
          "FileKey": "myflink.jar",
          "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "FlinkApplicationConfiguration": {
        "CheckpointConfiguration": {
          "CheckpointingEnabled": "false"
        }
      }
    }
  }
}
```



## Configurazione della creazione di 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
      }
    }
  }
}
```

## Disabilitare la creazione di 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 mediante snapshot

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 sui savepoint di Apache Flink, consulta [Savepoint](#) nella [documentazione di Apache Flink](#). Utilizzando gli snapshot, è possibile riavviare un'applicazione da un particolare snapshot dello stato dell'applicazione.

#### Note

È consigliabile che l'applicazione crei snapshot più volte al giorno per riavviarsi con i dati di stato corretti. La frequenza corretta per gli 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.

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 [Quota](#). 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:

- [Creazione automatica di snapshot](#)
- [Ripristino da uno snapshot che contiene dati di stato incompatibili](#)
- [Esempi di API per l'utilizzo di snapshot](#)

## Creazione automatica di snapshot

Se `SnapshotsEnabled` è impostato su `true` nell'[ApplicationSnapshotConfiguration](#) per l'applicazione, il servizio gestito per Apache Flink crea e utilizza automaticamente gli snapshot quando l'applicazione viene aggiornata, dimensionata 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. Consulta [Attivazione dei savepoint](#)

Gli snapshot creati in modo automatico hanno le seguenti qualità:

- Lo snapshot è gestito dal servizio, ma è possibile visualizzarlo utilizzando l'operazione [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 gli snapshot utilizzando l'operazione [DeleteApplicationSnapshot](#) prima di creare manualmente altri snapshot.

## Ripristino da uno snapshot 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 il ripristino di un'applicazione da uno snapshot che contiene dati di stato incompatibili, imposta il parametro `AllowNonRestoredState` della [FlinkRunConfiguration](#) su `true` utilizzando l'operazione [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 snapshot, consulta [Savepoint](#) 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 i sink con commit a due fasi, si consiglia di utilizzare uno snapshot di sistema (SW) anziché uno 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 dei savepoint esistenti, si consiglia di saltare il ripristino dallo snapshot impostando il parametro `ApplicationRestoreType` dell'operazione [StartApplication](#) su `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.

## Esempi di API per l'utilizzo di 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 del servizio gestito per Apache Flink](#).

## Abilitare gli snapshot 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 uno snapshot

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

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

## Elenco degli snapshot per 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
}
```

## Elenco dei dettagli di uno snapshot per l'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,
}
```

## Riavvio di un'applicazione utilizzando uno snapshot denominato

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"
    }
  }
}
```

## Riavvio di un'applicazione utilizzando lo snapshot 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"
    }
  }
}
```

```
}  
}
```

Riavvio di un'applicazione senza alcuno snapshot

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"  
    }  
  }  
}
```

## Dimensionamento delle applicazioni nel servizio gestito per 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 ulteriori informazioni su come Apache Flink pianifica le istanze parallele delle attività, consulta [Esecuzione parallela](#) nella [documentazione di Apache Flink](#).

Argomenti

- [Configurazione del parallelismo delle applicazioni e della KPU ParallelismPer](#)
- [Allocazione delle unità di elaborazione Kinesis](#)
- [Aggiornamento del parallelismo dell'applicazione](#)
- [Dimensionamento automatico](#)

## Configurazione del parallelismo delle applicazioni e della 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 pari a **ParallelismPerKPU** volte il limite per le KPU (il cui valore predefinito è 64). Il limite delle KPU può essere aumentato richiedendo un aumento del limite. Per istruzioni su come richiedere un aumento del limite, consultare «Richiedere un aumento del limite» in [Quote di servizio](#).

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

Il servizio gestito per Apache Flink fornisce la capacità sotto forma di KPU. 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.

Il servizio gestito per Apache Flink calcola le KPU necessarie per eseguire l'applicazione utilizzando le proprietà **Parallelism** e **ParallelismPerKPU**, nel modo seguente:

```
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 [Aggiornamento del parallelismo dell'applicazione](#).



Il limite predefinito per le KPU per l'applicazione è 64. Per istruzioni su come richiedere un aumento di tale limite, consulta «Richiedere un aumento del limite» in [Quote di servizio](#).

### 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](#).

## Aggiornamento del 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 del servizio gestito 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_15",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::mybucket",
          "FileKey": "myflink.jar",
          "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "FlinkApplicationConfiguration": {
      "ParallelismConfiguration": {
        "AutoScalingEnabled": "true",
        "ConfigurationType": "CUSTOM",
        "Parallelism": 4,
        "ParallelismPerKPU": 4
      }
    }
  }
}
```

```
}  
}
```

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"  
      }  
    }  
  }  
}
```

## Dimensionamento automatico

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 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 la CloudWatch metrica `containerCPUUtilization` è superiore al 75% per 15 minuti. Ciò significa che l'azione `ScaleUp` viene eseguita quando ci sono 15 punti dati consecutivi con un periodo di un minuto pari o superiore al 75%.
- L'applicazione si ridimensiona (riduce il parallelismo) quando l'utilizzo della CPU rimane inferiore al 10% per sei ore. Ciò significa che l'azione `ScaleDown` viene eseguita quando ci sono 360 punti dati consecutivi con un periodo di un minuto pari o superiore al 10%.

#### Note

È possibile fare riferimento a un massimo di `containerCPUUtilization` periodi di oltre un minuto per individuare la correlazione con un punto dati utilizzato per il dimensionamento, ma non è necessario indicare il momento esatto in cui l'azione viene azionata.

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 ridimensiona l'applicazione, questa comparirà nello status `AUTOSCALING`. Puoi controllare lo stato attuale della tua candidatura 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. All'account AWS vengono addebitati i KPU allocati dal servizio gestito per Apache Flink, in base alle impostazioni `parallelism` e `parallelismPerKPU` dell'applicazione. 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 notare quanto segue in merito al ridimensionamento 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 [Quota](#).
- 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 quanto segue [the section called "Aggiornamento del 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 nel servizio gestito per Apache Flink"](#).

## Considerazioni sul parallelismo massimo

- La logica di dimensionamento automatico impedirà il dimensionamento di un processo Flink in un parallelismo che causerà interferenze con il processo e l'operatore `maxParallelism`. Ad esempio, se si tratta di un processo semplice con solo una sorgente e un sink in cui la sorgente ha `16 maxParallelism` e sink ha 8, il processo non dimensionerà automaticamente oltre l'8.
- Se non `maxParallelism` è impostato per un processo, il valore predefinito di Flink è 128. Pertanto, nel caso in cui si ritenga che un processo debba essere eseguito con un parallelismo superiore a 128, sarà necessario impostare tale numero per l'applicazione.
- Se il processo dovrebbe dimensionarsi in maniera automatica ma ciò non avviene, assicurarsi di verificare che i valori `maxParallelism` lo consentano.

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

Per un esempio, vedi [kda-flink-app-autoscaling](#).

## Utilizzo del tagging

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:

- [Aggiunta di tag quando viene creata un'applicazione](#)
- [Aggiunta o aggiornamento di tag per un'applicazione esistente](#)
- [Elenco dei tag per un'applicazione](#)
- [Rimozione di tag da un'applicazione](#)

## Aggiunta di tag quando viene creata un'applicazione

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

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

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

```
    },
    {
      "Key": "Key2",
      "Value": "Value2"
    }
  ]
```

## Aggiunta o aggiornamento di tag per un'applicazione esistente

È possibile aggiungere i tag a un'applicazione utilizzando l'operazione [TagResource](#). Non è possibile aggiungere tag a un'applicazione utilizzando l'operazione [UpdateApplication](#).

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"
    }
  ]
}
```

## Elenco dei tag per un'applicazione

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

L'esempio seguente di richiesta per l'`ListTagsForResource` elenca i tag per un'applicazione:

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

## Rimozione di tag da un'applicazione

Per rimuovere i tag da un'applicazione, utilizzare l'operazione [UntagResource](#).

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 di CloudFormation con il servizio gestito per Apache Flink

Gli esercizi seguenti mostrano come avviare un'applicazione Flink creata tramite AWS CloudFormation utilizzando una funzione Lambda nello stesso stack.

### Prima di iniziare

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

### Scrittura di 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 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 elaborerà gli eventi AWS CloudFormation in arrivo, filtrerà tutto tranne Create e Update, otterrà lo stato dell'applicazione e, se lo stato è READY, la avvierà. Per ottenere lo stato dell'applicazione, è necessario creare il ruolo Lambda, come illustrato di seguito:

## Creazione di un ruolo Lambda

Crei un ruolo per Lambda per "parlare" con successo con l'applicazione e scrivere i log. Questo ruolo utilizzerà policy gestite predefinite, ma potresti voler restringere il campo utilizzando policy 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: /

```

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

## Richiamo della funzione Lambda

Ora non resta che invocare la funzione Lambda. Questa operazione viene eseguita utilizzando 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.

## Esempio completo

L'esempio seguente è una versione leggermente estesa dei passaggi precedenti con un'ulteriore modifica al `RunConfiguration` 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:
                - kinesisanalytics.amazonaws.com
            Action: sts:AssumeRole
      ManagedPolicyArns:
        - arn:aws:iam::aws:policy/AmazonKinesisFullAccess
        - arn:aws:iam::aws:policy/AmazonS3FullAccess
      Path: /
  InputKinesisStream:
    Type: AWS::Kinesis::Stream
    Properties:
      ShardCount: 1
  OutputKinesisStream:
    Type: AWS::Kinesis::Stream
```

```
Properties:
  ShardCount: 1
TestFlinkApplication:
  Type: 'AWS::kinesisanalyticsv2::Application'
  Properties:
    ApplicationName: 'CFNTestFlinkApplication'
    ApplicationDescription: 'Test Flink Application'
    RuntimeEnvironment: 'FLINK-1_15'
    ServiceExecutionRole: !GetAtt TestServiceExecutionRole.Arn
  ApplicationConfiguration:
    EnvironmentProperties:
      PropertyGroups:
        - PropertyGroupId: 'KinesisStreams'
          PropertyMap:
            INPUT_STREAM_NAME: !Ref InputKinesisStream
            OUTPUT_STREAM_NAME: !Ref OutputKinesisStream
            AWS_REGION: !Ref AWS::Region
    FlinkApplicationConfiguration:
      CheckpointConfiguration:
        ConfigurationType: 'CUSTOM'
        CheckpointingEnabled: True
        CheckpointInterval: 1500
        MinPauseBetweenCheckpoints: 500
      MonitoringConfiguration:
        ConfigurationType: 'CUSTOM'
        MetricsLevel: 'APPLICATION'
        LogLevel: 'INFO'
      ParallelismConfiguration:
        ConfigurationType: 'CUSTOM'
        Parallelism: 1
        ParallelismPerKPU: 1
        AutoScalingEnabled: True
    ApplicationSnapshotConfiguration:
      SnapshotsEnabled: True
    ApplicationCodeConfiguration:
      CodeContent:
        S3ContentLocation:
          BucketARN: !Ref CodeContentBucketArn
          FileKey: !Ref CodeContentFileKey
        CodeContentType: 'ZIPFILE'
  StartApplicationLambdaRole:
    Type: AWS::IAM::Role
    DependsOn: TestFlinkApplication
  Properties:
```

```

Description: A role for lambda to use while interacting with an application.
AssumeRolePolicyDocument:
  Version: '2012-10-17'
  Statement:
    - Effect: Allow
      Principal:
        Service:
          - lambda.amazonaws.com
      Action:
        - sts:AssumeRole
ManagedPolicyArns:
  - arn:aws:iam::aws:policy/Amazonmanaged-flinkFullAccess
  - arn:aws:iam::aws:policy/CloudWatchLogsFullAccess
Path: /
StartApplicationLambda:
  Type: AWS::Lambda::Function
  DependsOn: StartApplicationLambdaRole
  Properties:
    Description: Starts an application when invoked.
    Runtime: python3.8
    Role: !GetAtt StartApplicationLambdaRole.Arn
    Handler: index.lambda_handler
    Timeout: 30
    Code:
      ZipFile: |
        import logging
        import cfnresponse
        import boto3

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

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

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

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

```

```
        cfnresponse.send(event, context, cfnresponse.SUCCESS, {})

    return

    # use kinesisanalyticv2 API to start an application.
    client_kda = boto3.client('kinesisanalyticv2',
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 lo stato di avanzamento. 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 servizio AWS utilizzando AWS CloudFormation \(parte 1 di 3\)](#).
- [Spiegazione passo per passo: ricerca di ID di Amazon Machine Image](#).

## Utilizzo del pannello di controllo di Apache Flink con il servizio gestito 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:

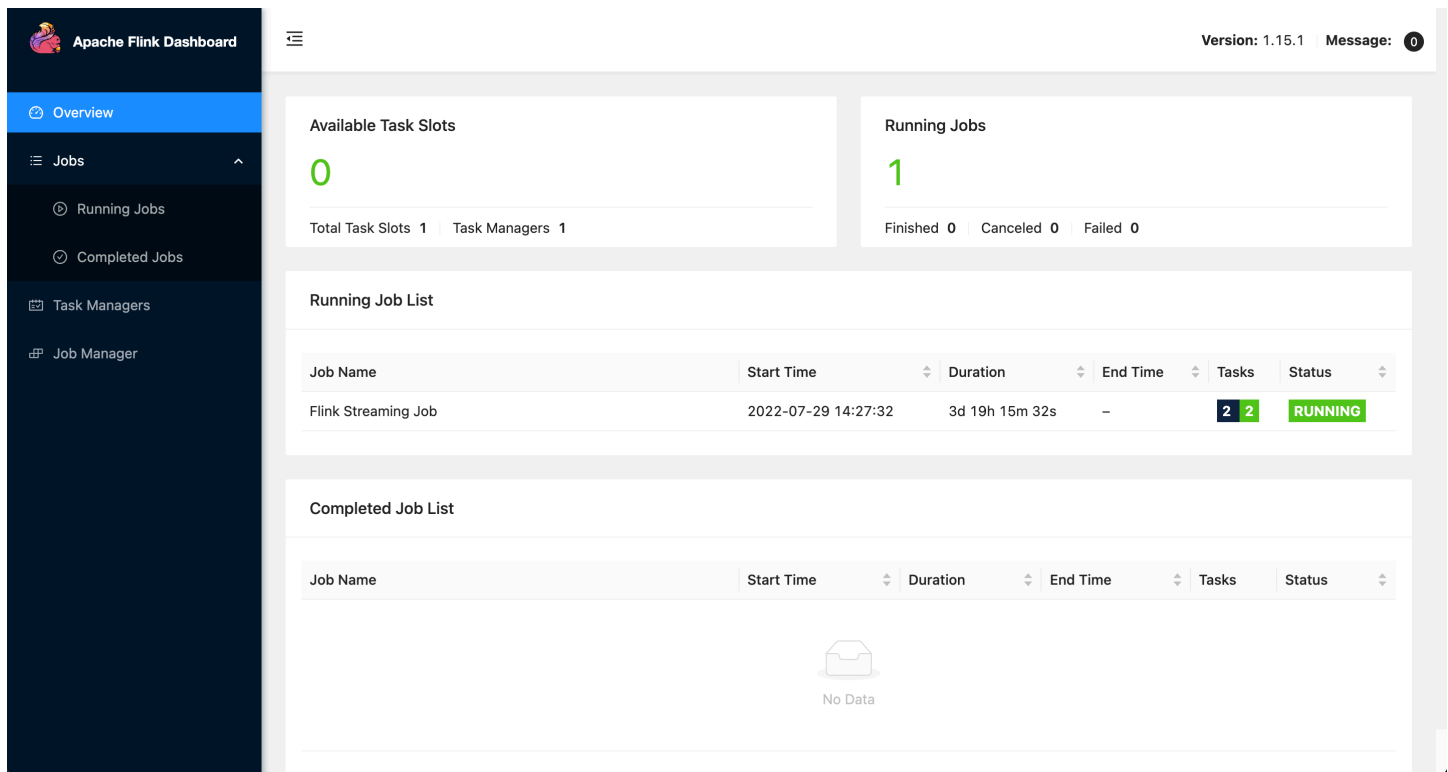
- Risorse in uso, inclusi task manager e task slot.
- Informazioni sui processi, inclusi quelli in esecuzione, completati, annullati e non riusciti.

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.

Nota 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.



The screenshot displays the Apache Flink Dashboard interface. On the left is a dark sidebar with navigation options: Overview (selected), Jobs, Running Jobs, Completed Jobs, Task Managers, and Job Manager. The main content area shows a top navigation bar with 'Version: 1.15.1' and 'Message: 0'. Below this, there are two summary cards: 'Available Task Slots' showing 0 and 'Running Jobs' showing 1. The 'Running Jobs' card also includes sub-metrics: 'Finished 0', 'Canceled 0', and 'Failed 0'. Below these is a 'Running Job List' table with one entry: 'Flink Streaming Job' with a start time of '2022-07-29 14:27:32', a duration of '3d 19h 15m 32s', and a status of 'RUNNING' with 2 tasks. At the bottom, there is a 'Completed Job List' section which is currently empty, displaying 'No Data'.

## Accesso al pannello di controllo Apache Flink dell'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.

## Accesso al pannello di controllo Apache Flink dell'applicazione utilizzando la console del servizio gestito per 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.

## Accesso al pannello di controllo Apache Flink dell'applicazione tramite la CLI del servizio gestito per Apache Flink

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'operazione [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.

### Servizio gestito da Amazon per Apache Flink versione 1.15.2

Il servizio gestito per Apache Flink supporta le seguenti nuove funzionalità in Apache 1.15.2

Funzionalità	Descrizione	Riferimento Apache FLIP
Async Sink	Un framework di AWS con contribuzioni per la creazione di destinazioni asincrone che consente agli sviluppatori di creare connettori AWS personalizzati con meno della metà dello sforzo richiesto in precedenza. Per ulteriori informazioni, consulta <a href="#">Generic Asynchronous Base Sink</a> .	<a href="#">FLIP-171: Async Sink</a> .
Kinesis Data Firehose Sink	AWS ha contribuito con un nuovo Amazon Kinesis Firehose Sink utilizzando il framework Async.	<a href="#">Amazon Kinesis Data Firehose Sink</a> .
Stop with Savepoint	Stop with Savepoint garantisce un funzionamento pulito ininterrotto e, soprattutto, supporta la semantica exactly-once per i clienti che decidono di usarlo.	<a href="#">FLIP-34: Terminate/Suspend Job with Savepoint</a> .
Scala Decoupling	Gli utenti ora possono sfruttare l'API Java di qualsiasi versione di Scala, inclusa Scala 3. I	<a href="#">FLIP-28: Obiettivo a lungo termine: rendere flink-table privo di Scala</a> .

Funzionalità	Descrizione	Riferimento Apache FLIP
	clienti dovranno raggruppare la libreria standard Scala che hanno scelto nelle loro applicazioni Scala.	
Scala	Cfr. Scala Decoupling qui sopra	<a href="#">FLIP-28: Obiettivo a lungo termine: rendere flink-table privo di Scala.</a>
Metriche unificate per i connettori.	Flink ha <a href="#">definito metriche standard</a> 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.	<a href="#">FLIP-33: Standardize Connector Metrics</a> e <a href="#">FLIP-179: Expose Standardized Operator Metrics.</a>
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.	<a href="#">FLIP-147: Support Checkpoints After Tasks Finished.</a>

## 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: [Utilizzo di un notebook Studio con il servizio gestito 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 del servizio gestito da Amazon per Apache Flink versione 1.15, assicurati di utilizzare le API per connettori Kafka più recenti. [Apache Flink ha reso obsoleti FlinkKafkaConsumer e FlinkKafkaProducer](#). Queste API per il sink Kafka non possono eseguire il commit con Kafka per Flink 1.15. Assicurati di utilizzare [KafkaSource](#) e [KafkaSink](#).

## Componenti

Componente	Versione
Java	11 (consigliata)
Scala	2.12
Servizio gestito per Apache Flink per il runtime di Flink (aws-kinesisanalytics-runtime)	1.2.0

Componente	Versione
<a href="#">AWSConnettore Kinesis (flink-connector-kinesis)</a>	1.15.4
<a href="#">Apache Beam (solo applicazioni Beam)</a>	2.33.0, con la versione Jackson 2.12.2

# Utilizzo di un notebook Studio con il servizio gestito 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 gestione AWS, 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 sulla creazione di un notebook Studio](#). 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:

- [Creazione di un notebook Studio](#)
- [Analisi interattiva dei dati di streaming](#)
- [Implementazione come applicazione con stato durevole](#)
- [Autorizzazioni IAM per notebook Studio](#)
- [Connettori e dipendenze](#)

- [Funzioni definite dall'utente](#)
- [Abilitare la creazione di checkpoint](#)
- [Utilizzo di AWS Glue](#)
- [Esempi e tutorial](#)
- [Risoluzione dei problemi](#)
- [Appendice: creazione di policy IAM personalizzate](#)

## Creazione di un notebook 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 database AWS Glue. Questo database contiene tabelle in cui sono archiviati gli schemi e gli endpoint di origine e destinazione dei dati. Per ulteriori informazioni, consulta [Utilizzo di 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 scalabilità 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 sulla scalabilità dell'applicazione, consulta [Scalabilità dell'applicazione 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 un notebook Studio utilizzando la AWS Management Console o la 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](#).

## Analisi interattiva dei dati di 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 Streaming SQL in Flink, consulta [Esempi e tutorial](#) di seguito 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 (API Table) e Scala (API Table e DataStream) per scrivere programmi

per interrogare i 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>	Creates ExecutionEnvironment/StreamExecutionEnvironment/BatchTableEnvironment/StreamTableEnvironment and provides a Scala environment
<code>%flink.pyflink</code>	<code>PyFlinkInterpreter</code>	Provides a python environment
<code>%flink.ipynk</code>	<code>IPyFlinkInterpreter</code>	Provides an ipython environment
<code>%flink.ssql</code>	<code>FlinkStreamSqlInterpreter</code>	Provides a stream sql environment
<code>%flink.bsql</code>	<code>FlinkBatchSqlInterpreter</code>	Provides a batch sql environment

Per ulteriori informazioni sugli interpreti Flink, consulta [Interprete Flink per Apache Zeppelin](#).

Se si utilizzano `%flink.pyflink` o `%flink.ipynk` 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>sekv</code>	<code>StreamExecutionEnvironment</code>
<code>stenv</code>	<code>StreamTableEnvironment</code> per <code>blink planner</code>


Puoi accedere alle risorse dell'ambiente tabellare Python con le seguenti variabili:

Variabile	Risorsa
<code>s_kv</code>	<code>StreamExecutionEnvironment</code>
<code>st_kv</code>	<code>StreamTableEnvironment</code> per <code>blink planner</code>

Per ulteriori informazioni sull'utilizzo degli ambienti tabellari, consulta [Create a TableEnvironment nella documentazione di Apache Flink](#).

## Implementazione come applicazione con 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 alla velocità effettiva 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 distribuisce 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.ssql`, `%flink.pyflink`, `%flink.ipyflink`, `%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.ipyflink` in una nota con `%flink` and `%flink.ssql` 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.ssql(parallelism=32)`) verrà ignorata nelle applicazioni implementate da una nota. È invece possibile aggiornare l'applicazione distribuita tramite l'AWSAPI 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 [Implementazione come applicazione con stato durevole in un VPC senza accesso a Internet](#).

## Criteria Scala/Python

- Nel tuo codice Scala o Python, usa il [pianificatore Blink](#) (`stenv`, `stenv` per Scala; `s_env`, per `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 magici IPython](#) come `%timeit` o `%conda` nelle note pensate per essere implementate 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.

## Criteria 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](#).

## Autorizzazioni IAM per 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

## 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 la AWS Management Console per creare il notebook Studio, il servizio gestito per Apache Flink include i seguenti connettori personalizzati per impostazione predefinita: `flink-sql-connector-flink`, `flink-connector-kafka_2.12` e `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, deseleziona 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 i connettori non sono inclusi per

impostazione predefinita. Per aggiungerli, specificarli 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 [UpdateApplicationAPI](#) 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.

## Dipendenze e connettori personalizzati

Per utilizzare la AWS Management Console per aggiungere una dipendenza o un connettore personalizzato al notebook Studio, segui questi passaggi:

1. Carica il file del connettore personalizzato in Amazon S3.
2. Nella AWS Management Console, scegli l'opzione Creazione personalizzata per creare il notebook 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. Seleziona Salvataggio delle modifiche.

Per aggiungere un JAR di dipendenza o un connettore personalizzato quando crei un nuovo notebook Studio utilizzando l'[CreateApplicationAPI](#), 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 [UpdateApplicationAPI](#) 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.

## Funzioni definite dall'utente

Le funzioni definite dall'utente (UDF) sono punti di estensione che consentono di richiamare la logica utilizzata di frequente o la logica personalizzata non esprimibili diversamente nelle query. È possibile utilizzare Python o un linguaggio JVM come Java o Scala per implementare le tue UDF in paragrafi all'interno del notebook Studio. Puoi anche aggiungere al notebook Studio dei file JAR esterni che contengono UDF implementate in un linguaggio JVM.

Quando implementi JAR che registrano le classi astratte che costituiscono una sottoclasse `UserDefinedFunction` (o le tue classi astratte), utilizza l'ambito fornito in Apache Maven, le dichiarazioni di dipendenza `compileOnly` in Gradle, l'ambito fornito in SBT o una direttiva equivalente nella configurazione di build del progetto UDF. Ciò consente la compilazione del codice di origine UDF con le API Flink, ma le classi delle API Flink non sono di per sé incluse negli elementi 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. Nella AWS Management Console, scegli l'opzione Creazione personalizzata per creare il notebook 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. Seleziona Salvataggio delle modifiche.

Per aggiungere un JAR UDF quando crei un nuovo notebook Studio utilizzando l'[CreateApplication](#) API, specifica la posizione JAR nel tipo di dati.

CustomArtifactConfiguration Per aggiungere un file JAR UDF a un notebook Studio esistente, richiama l'operazione [UpdateApplication](#) API e specifica la posizione JAR nel CustomArtifactsConfigurationUpdate tipo di dati. In alternativa, è possibile utilizzare il comando 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. sql(type=update, parallelism=1)
INSERT INTO
    table2
SELECT
    myNewFuncNameForClass(column_name)
FROM
    table1
;
```

- Se il JAR UDF include gli SDK Flink, configura il progetto Java in modo che il codice di origine UDF possa essere compilato con gli SDK Flink, ma le classi SDK Flink non sono 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](#).

## Abilitare la creazione di checkpoint

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](#).

### Impostazione dell'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"
)
```

### Impostazione del 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"
)
```

## Utilizzo di AWS Glue

Il notebook Studio archivia e ottiene informazioni sulle sue origini di dati e sui suoi sink da AWS Glue. Quando crei il notebook Studio, specifichi il database AWS Glue che contiene le informazioni di connessione. Quando accedi alle origini dati e ai sink, specifichi le tabelle AWS Glue contenute nel

database. Le tabelle AWS Glue forniscono l'accesso alle connessioni AWS Glue che definiscono le posizioni, gli schemi e i parametri delle origini e delle destinazioni 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 connessione AWS Glue, un database e una tabella da utilizzare con i notebook Studio, consulta [Creazione di un database AWS Glue](#) nel tutorial [Tutorial sulla creazione di un notebook Studio](#).

## Proprietà tabella

Oltre ai campi dati, le tabelle AWS Glue forniscono altre informazioni al notebook di Studio utilizzando le proprietà delle tabelle. Il servizio gestito per Apache Flink utilizza le seguenti proprietà della tabella AWS Glue:

- [Utilizzo dei 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.
- [Utilizzo del connettore Flink e delle proprietà del formato](#): queste proprietà forniscono informazioni sui flussi di dati.

Per aggiungere una proprietà a una tabella AWS Glue, esegui le seguenti operazioni:

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue 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.

## Utilizzo dei 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 nella tabella AWS Glue delle proprietà che indicano al runtime del servizio gestito per Apache Flink di generare questi valori nei campi specificati.

Le chiavi e i valori utilizzati nelle proprietà della tabella sono i seguenti:

Tipo timestamp	Chiave	Valore
<a href="#">Tempo di elaborazione</a>	<code>managed-flink.proctime</code>	The column name that AWS Glue will use to expose the value. This column name does not correspond to an existing table column.
<a href="#">Ora evento</a>	<code>managed-flink.rowtime</code>	The column name that AWS Glue will use to expose the value. This column name corresponds to an existing table column.
	<code>managed-flink.watermark.column_name</code> .milliseconds	The watermark interval in milliseconds

## Utilizzo del connettore Flink e delle proprietà del 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
<a href="#">Kafka</a>	<code>format</code>	The format used to deserialize and serialize Kafka messages, e.g. <code>json</code> or <code>csv</code> .
	<code>scan.startup.mode</code>	The startup mode for the Kafka consumer, e.g. <code>earliest-offset</code> or <code>timestamp</code> .
<a href="#">Kinesis</a>	<code>format</code>	The format used to deserialize and serialize Kinesis data

Tipo di connettore	Chiave	Valore
		stream records, e.g. json or csv.
	<code>aws.region</code>	The AWS region where the stream is defined.
<a href="#">S3 (file system)</a>	<code>format</code>	The format used to deserialize and serialize files, e.g. json or csv.
	<code>path</code>	The Amazon S3 path, e.g. <code>s3://mybucket/</code> .

Per ulteriori informazioni su altri connettori oltre a Kinesis e Apache Kafka, consulta la documentazione relativa al tuo connettore.

## Esempi e tutorial

### Argomenti

- [Tutorial: creazione di un notebook Studio nel servizio gestito per Apache Flink](#)
- [Tutorial: implementazione come applicazione con stato durevole](#)
- [Esempi](#)

## Tutorial: creazione di un notebook Studio nel servizio gestito per Apache Flink

Il seguente tutorial dimostra 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:

- [Installazione](#)
- [Creazione di un database AWS Glue](#)
- [Fasi successive](#)

- [Creazione di un notebook Studio con il flusso di dati Kinesis](#)
- [Creazione di un notebook Studio con Amazon MSK](#)
- [Eliminazione dell'applicazione e delle risorse dipendenti](#)

## Installazione

Verifica che sia disponibile la AWS CLI versione 2 o successiva. Per installare la AWS CLI più recente, consulta [Installazione, aggiornamento e disinstallazione della AWS CLI versione 2](#).

## Creazione di un database AWS Glue

Il notebook Studio utilizza un database [AWS Glue](#) per i metadati sull'origine dati Amazon MSK.

### Creazione di un database AWS Glue

1. Apri la console AWS Glue all'indirizzo <https://console.aws.amazon.com/glue/>.
2. Scegli Aggiungi database. Nella finestra Aggiungi database, inserisci **default** per Nome database. Seleziona Crea.

## Fasi successive

Con questo tutorial, puoi creare un notebook Studio che utilizza il flusso di dati Kinesis o Amazon MSK:

- [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.
- [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 client Amazon EC2 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:



- [Installazione](#)
- [Creazione di 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](#)

## Installazione

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 shard aperti su **1**.

Per creare il flusso (ExampleInputStream) tramite la AWS CLI, utilizza il seguente comando `create-stream` della AWS CLI di Amazon Kinesis.

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

## Creazione di 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. È quindi possibile effettuare il check-in AWS Glue per verificare che la tabella sia stata creata correttamente.

## Creazione di una tabella

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo <https://console.aws.amazon.com/glue/>.
2. Se non disponi già di un database AWS Glue, scegli Database dalla barra di navigazione a sinistra. Scegli Aggiungi database. Nella finestra Aggiungi database, inserisci **default** per Nome database. Scegli 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. Seleziona Avanti.
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. Seleziona Avanti.
6. Nella pagina Classificazione, scegli JSON. Seleziona Avanti.
7. Nella pagina Definisci uno schema, scegli Aggiungi colonna per aggiungere una colonna. Aggiungi colonne con le seguenti proprietà:

Nome colonna	Tipo di dati
<b>ticker</b>	<b>string</b>
<b>price</b>	<b>double</b>

Seleziona Avanti.

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 l'applicazione, puoi utilizzare la AWS Management Console o la AWS CLI.

- [Creazione di un notebook Studio utilizzando la AWS Management Console](#)
- [Creazione di un notebook Studio utilizzando la AWS CLI](#)

### Creazione di un notebook Studio utilizzando la AWS Management Console

1. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/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 MyNotebookpagina, scegli Esegui. Attendi che lo stato mostri In esecuzione. Si applicano costi quando il notebook è in funzione.

### Creazione di un notebook Studio utilizzando la AWS CLI

Per creare un notebook Studio utilizzando la 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/ZeppelinRole` 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 kinesisanalyticstv2 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:kinesisanalyticstv2:us-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 kinesisanalyticstv2 start-application --application-arn
arn:aws:kinesisanalyticstv2-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 console di AWS CloudFormation si apre con il modello Kinesis Data Generator. Seleziona Avanti.
4. Nella pagina Specifica i dettagli dello stack, inserisci il nome utente e la password per l'utente Cognito. Seleziona Avanti.
5. Nella pagina Configura opzioni dello stack, scegli Successivo.
6. Nella pagina Review Kinesis-Data-Generator-Cognito-User, scegli le risorse Riconosco che potrebbero creare IAM. AWS CloudFormation casella di controllo. Scegli Crea stack.
7. Attendi il termine della creazione dello stack AWS CloudFormation. Una volta completato lo stack, apri lo stack Kinesis-Data-Generator-Cognito-User nella console AWS CloudFormation e scegli la scheda Output. Apri l'URL elencato per il valore KinesisDataGeneratorUrldi output.
8. Nella pagina Amazon Kinesis Data Generator, accedi con le credenziali create nel passaggio 4.
9. Nella pagina successiva, specifica i seguenti valori:

Regione	<b>us-east-1</b>
Flusso/flusso Kinesis Data Firehose	<b>ExampleInputStream</b>
Record al secondo	<b>1</b>

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. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard>.
2. Nella pagina Applicazioni del servizio gestito per Apache Flink, scegli la scheda Notebook Studio. Scegli MyNotebook.
3. Nella MyNotebookpagina, scegli Apri in Apache Zeppelin.

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.ssql(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](#).

## Creazione di 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:

- [Installazione](#)
- [Aggiungi un gateway NAT al tuo VPC](#)
- [Creazione di una connessione e una tabella AWS Glue](#)
- [Crea un notebook Studio con Amazon MSK](#)
- [Invio di dati al cluster Amazon MSK](#)
- [Test del notebook Studio](#)

### Installazione

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 [Nozioni di base per l'uso di Amazon MSK](#) per creare un Amazon VPC, un cluster Amazon MSK, un argomento e un'istanza client Amazon EC2.

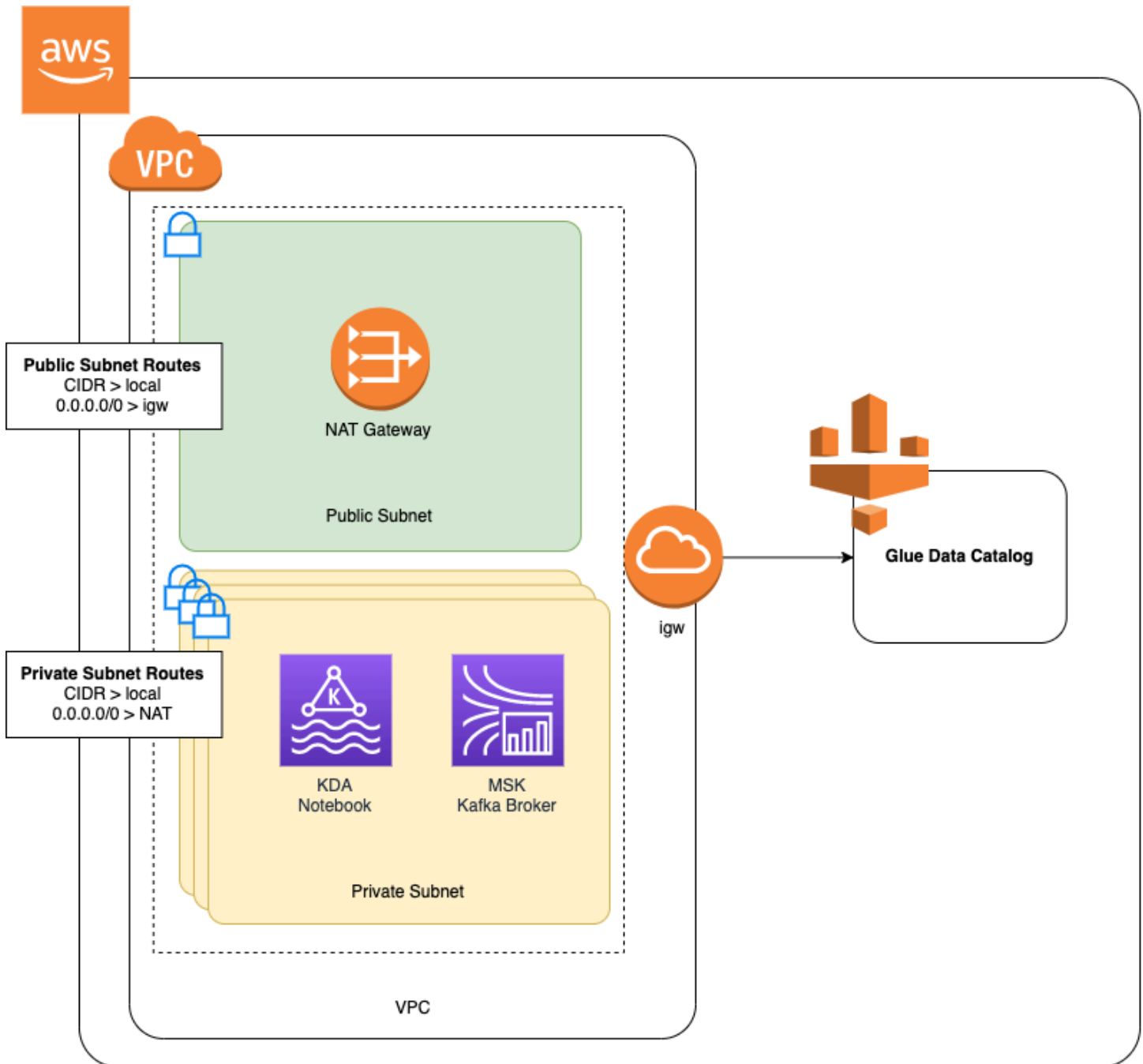
Seguendo il tutorial, completa le seguenti operazioni:

- 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 l'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	<b>ZeppelinGateway</b>
Sottorete	AWSKafkaTutorialSubnet1
ID allocazione IP elastico	Choose an available Elastic IP. If there are no Elastic IPs available, choose Allocazione di un IP elastico, and then choose the Elastic IP that the console creates.

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 VPC (ad esempio AWSKafkaTutorialVPC).

Seleziona Crea.

8. Nell'elenco delle tabelle di routing, 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 Gateway NAT, ZeppelinGateway. Seleziona Salva route. Scegli Chiudi.
11. Nella pagina Tabelle di routing, con ZeppelinRouteTable selezionato, scegli la scheda Associazioni sottorete. Scegli Modifica associazioni sottorete.
12. Nella pagina Modifica associazioni sottorete, scegli AWSKafkaTutorialSubnet2 e AWSKafkaTutorialSubnet3. Seleziona Salva.

Creazione di una connessione e una tabella AWS Glue

Il notebook Studio utilizza un database [AWS Glue](#) per i metadati sull'origine dati Amazon MSK. In questa sezione, viene creata una connessione AWS Glue che descrive come accedere al cluster

Amazon MSK e una tabella AWS Glue che descrive come presentare i dati dell'origine dati a client come il notebook Studio.

### Creazione di una connessione

1. Accedi alla AWS Management Console, quindi apri la console AWS Glue all'indirizzo <https://console.aws.amazon.com/glue/>.
2. Se non disponi già di un database AWS Glue, scegli Database dalla barra di navigazione a sinistra. Scegli Aggiungi database. Nella finestra Aggiungi database, inserisci **default** per Nome database. Seleziona 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 **ZepplinConnection**.
  - Per Tipo di connessione, scegli Kafka.
  - Per URL del server di bootstrap Kafka, 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.

Seleziona Successivo.

5. Nella pagina VPC, fornisci i seguenti valori:
  - Per VPC, scegli il nome del VPC (ad esempio AWSKafkaTutorialVPC).
  - Per Sottorete, scegli AWSKafkaTutorialSubnet2.
  - Per Gruppi di sicurezza, scegli tutti i gruppi disponibili.

Seleziona 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 verificare 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 il Nome tabella. Assicurati di selezionare il database creato in precedenza. Seleziona Successivo.
3. Nella pagina Aggiungi un datastore, scegli Kafka. Per Nome argomento, inserisci il nome dell'argomento (ad esempio AWSKafkaTutorialTopic). Per Connessione, scegli ZeppelinConnection.
4. Nella pagina Classificazione, scegli JSON. Seleziona 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
<b>ticker</b>	<b>string</b>
<b>price</b>	<b>double</b>

Seleziona Successivo.

6. Nella pagina successiva, verifica le impostazioni e scegli Fine.
7. Scegli la tabella appena creata dall'elenco delle tabelle.
8. Scegli Modifica tabella e aggiungi una proprietà con la chiave `managed-flink.proctime` e il valore `proctime`.
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 la AWS Management Console o la AWS CLI.

- [Creazione di un notebook Studio utilizzando la AWS Management Console](#)
- [Creazione di un notebook Studio utilizzando la 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**.

### Creazione di un notebook Studio utilizzando la AWS Management Console

1. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/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 **MyNotebook**, scegli la scheda **Configurazione**. Nella sezione **Reti**, scegli **Modifica**.
5. Nella pagina **Modifica reti per MyNotebook**, scegli la configurazione VPC basata sul cluster Amazon MSK. Scegli il cluster Amazon MSK per Cluster Amazon MSK. Seleziona **Salva modifiche**.

6. Nella pagina MyNotebook, scegli Esegui. Attendi che lo stato mostri In esecuzione.

## Creazione di un notebook Studio utilizzando la AWS CLI

Per creare un notebook Studio utilizzando la 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.
  - Gli ID di sottorete e l'ID del gruppo di sicurezza per 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"
        }
      }
    }
  }
}
```

```

    }
  }
}

```

- Per creare l'applicazione, esegui il comando riportato di seguito:

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

- 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/ZeppelinRole",
    ...
  }
}

```

- 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 client Amazon EC2 per inviare dati all'origine dati Amazon MSK.

- Esegui la connessione al client Amazon EC2.
- 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. Configura la AWS CLI sul 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 dei broker di bootstrap del tuo cluster Amazon MSK e aggiorna il nome dell'argomento se non è `AWSKafkaTutorialTopic`:

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

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

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

while True:
    data =getStock()
    # print(data)
    try:
        future = producer.send("AWSKafkaTutorialTopic", value=data)
        producer.flush()
```

```
record_metadata = future.get(timeout=10)
print("sent event to Kafka! topic {} partition {} offset
{}".format(record_metadata.topic, record_metadata.partition,
record_metadata.offset))
except Exception as e:
    print(e.with_traceback())
```

5. 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. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/managed-flink/home?region=us-east-1#/applications/dashboard>.
2. Nella pagina Applicazioni del servizio gestito per Apache Flink, scegli la scheda Notebook Studio. Scegli MyNotebook.
3. Nella pagina MyNotebook, scegli Apri in Apache Zeppelin.

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](#).

## Eliminazione dell'applicazione e delle risorse dipendenti

### Eliminazione del notebook Studio

1. Apri la console del servizio gestito per Apache Flink.
2. Scegli MyNotebook.
3. Scegli Operazioni, quindi Elimina.

### Eliminazione del database e della connessione AWS Glue

1. Apri la console AWS Glue 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. Scegli Operazione, Elimina connessione. Conferma la selezione.

### Eliminazione del ruolo e della policy IAM

1. Apri la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
2. Dalla barra di navigazione sinistra, scegli Ruoli.
3. Utilizza la barra di ricerca per cercare il ruolo ZeppelinRole.
4. Scegli il ruolo ZeppelinRole. Scegli Elimina ruolo. Conferma l'eliminazione.

### Eliminazione del gruppo di log CloudWatch

La console crea automaticamente un gruppo di log e un flusso di log CloudWatch quando crei l'applicazione utilizzando la console. Non ottieni un gruppo e un flusso di log se l'applicazione viene creata utilizzando la AWS CLI.

1. Apri la console CloudWatch all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.

2. Scegli Gruppi di log dalla barra di navigazione a sinistra.
3. Scegli il gruppo di log `/aws/KinesisAnalytics/MyNotebook`.
4. Scegli Operazioni > Elimina gruppo/i di log. Conferma l'eliminazione.

### Eliminazione delle risorse del flusso di dati Kinesis

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 eliminare l'istanza client Amazon EC2, l'Amazon VPC e il cluster Amazon MSK.

### Eliminazione del cluster Amazon MSK

Segui questi passaggi se hai creato un cluster Amazon MSK per questo tutorial.

1. Apri la console Amazon MSK all'indirizzo <https://console.aws.amazon.com/msk/home?region=us-east-1#/home/>.
2. Scegli AWSKafkaTutorialCluster. 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 client Amazon EC2 per questo tutorial.

1. Apri la console Amazon EC2 all'indirizzo <https://console.aws.amazon.com/ec2/>.
2. Scegli Istanze nella barra di navigazione a sinistra.
3. Seleziona la casella di controllo accanto a ZeppelinClient.
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 console Amazon EC2 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 AWSKafkaTutorialVPC. Scegli Operazioni > Elimina VPC. Inserisci **delete** e conferma l'eliminazione.

## Tutorial: implementazione come applicazione 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:

- [Installazione](#)
- [Implementazione di un'applicazione con stato durevole utilizzando la AWS Management Console](#)
- [Implementazione di un'applicazione con stato durevole utilizzando la AWS CLI](#)

### Installazione

Crea un nuovo notebook Studio seguendo le istruzioni in [Tutorial sulla creazione di un notebook Studio](#) 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:
  - AmazonS3FullAccess
  - Amazonmanaged-flinkFullAccess
  - Accedi a origini, destinazioni e VPC, a seconda dei casi. Per ulteriori informazioni, consulta [Autorizzazioni IAM per notebook Studio](#).
3. Utilizza il seguente codice di esempio:

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

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

4. 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 Implementa tramite console AWS.

- Verrà visualizzata la pagina AWS Management Console per la creazione di un'applicazione del servizio gestito per 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 implementare un'applicazione utilizzando la AWS CLI, è necessario aggiornare la AWS CLI per l'utilizzo del modello di servizio fornito con le informazioni sulla Beta 2. Per informazioni sull'utilizzo del modello di servizio aggiornato, consulta [Installazione](#).

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 kinesisanalyticstv2 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 kinesisanalyticstv2 create-application-presigned-url \
  --application-name <app-name> \
  --url-type ZEPPELIN_UI_URL \

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

```

## Esempi

Le seguenti query di esempio mostrano come analizzare i dati utilizzando le query a finestra in un notebook di Studio.

- [Creazione di tabelle con Amazon MSK/Apache Kafka](#)
- [Creazione di tabelle con Kinesis](#)

- [Finestra a cascata](#)
- [Finestra scorrevole](#)
- [SQL interattivo](#)
- [BlackHole Connettore SQL](#)
- [Generatore di dati](#)
- [Scala interattivo](#)
- [Python interattivo](#)
- [Python, SQL e Scala interattivi](#)
- [Flusso di dati Kinesis tra account diversi](#)

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](#).

## Creazione di 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](#).



## Creazione di 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](#).

## Finestra a cascata

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)
```

## 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.ssql(type=update)
```

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

## 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 “Generatore di dati”](#) 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.ssql(type=single, parallelism=4, refreshInterval=1000, template=<h1>{2}</h1>
records seen until <h1>Processing Time: {1}</h1> and <h1>Event Time: {0}</h1>)
```

```
-- An interactive query prints how many records from the `key-value-stream` we have
seen so far, along with the current processing and event time.
```

```
SELECT
  MAX(`et`) as `et`,
  MAX(`pt`) as `pt`,
  SUM(`value`) as `sum`
FROM
  `key-values`
```

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

```
-- An interactive tumbling window query that displays the number of records observed
per (event time) second.
```

```
-- Browse through the chart views to see different visualizations of the streaming
result.
```

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

## BlackHole Connettore 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
```

## Generatore di dati

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`
```

## Scala interattivo

Questa è la traduzione in Scala di [the section called “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`
```

## Python interattivo

Questa è la traduzione in Python di [the section called “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`
```

## 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!")

```

## Flusso di dati Kinesis tra account diversi

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'
)
```

## Risoluzione dei problemi

Questa sezione contiene informazioni sulla risoluzione dei problemi per i notebook Studio.

### Arresto di un'applicazione bloccata

Per interrompere un'applicazione bloccata in uno stato transitorio, richiamate 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](#).

### Implementazione come applicazione con 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):
3. `kinesis.analytics.flink.run.options`
  - a. `python: source/note.py`
  - b. `file jar: lib/ .jar PythonApplicationDependencies`
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 Dimensione D e riduzione del tempo 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 processi

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)
```

## Riavvio dell'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.

## Appendice: creazione di policy IAM personalizzate

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 policy dell'esempio seguente concede l'autorizzazione per l'accesso al 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](#).

# Guida introduttiva ad Amazon Managed Service per Apache Flink (DataStream API)

Questa sezione presenta i concetti fondamentali di Managed Service for 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 di Apache Flink](#)
- [Prerequisiti per il completamento degli esercizi](#)
- [Fase 1: impostazione di un account AWS 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](#)
- [Fase 4: eliminazione delle risorse AWS](#)
- [Fase 5: passaggi successivi](#)

## Componenti dell'applicazione del servizio gestito di 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 [Origini](#).
- **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 dell'API DataStream](#).
- **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 Kinesis Data Firehose, un bucket Amazon S3, ecc. Per ulteriori informazioni, consulta [Sink](#).

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 [Fase 1: impostazione di un account AWS e creazione di un utente amministratore](#).

## Fase 1: impostazione di un account AWS e creazione di un utente amministratore

Prima di utilizzare il servizio gestito per Apache Flink per la prima volta, completa le seguenti attività:

### Registrarsi per creare un Account AWS

Se non disponi di un Account AWS, completa la procedura seguente per crearne uno.

Per registrarsi a un Account AWS

1. Apri la pagina all'indirizzo <https://portal.aws.amazon.com/billing/signup>.
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.

Durante la registrazione di un Account AWS, viene creato un Utente root dell'account AWS. 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 amministrativo](#) e utilizza solo l'utente root per eseguire [attività che richiedono l'accesso di un utente root](#).

Al termine del processo di registrazione, riceverai un'e-mail di conferma da AWS. È possibile visualizzare l'attività corrente dell'account e gestire l'account in qualsiasi momento accedendo all'indirizzo <https://aws.amazon.com/> e selezionando Il mio account.

## Creazione di un utente amministratore

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

### Protezione dell'Utente root dell'account AWS

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

Per informazioni sull'accesso utilizzando un utente root, consulta la pagina [Accesso come utente root](#) 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 [Abilitazione di un dispositivo MFA virtuale per l'utente root dell'Account AWS \(console\)](#) nella Guida per l'utente IAM.

### Creazione di un utente amministratore

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 Centro identità IAM, assegna l'accesso amministrativo a un utente amministratore.

Per un tutorial sull'utilizzo di IAM Identity Center directory come origine di identità, consulta [Configure user access with the default IAM Identity Center directory](#) nella Guida per l'utente di AWS IAM Identity Center.

## 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 [Accedere al portale di accesso AWS](#) nella Guida per l'utente Accedi ad AWS.

## Concessione dell'accesso programmatico

Gli utenti hanno bisogno di un accesso programmatico se desiderano interagire con AWS esternamente a 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 temporanee e per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	<p>Segui le istruzioni per l'interfaccia che desideri utilizzare.</p> <ul style="list-style-type: none"> <li>• Per la AWS CLI, consulta la pagina <a href="#">Configurazione della AWS CLI per l'uso di AWS IAM Identity Center</a> nella Guida per l'utente dell'AWS Command Line Interface.</li> <li>• Per gli SDK AWS, gli strumenti e le API AWS, consulta la pagina <a href="#">Autenticazione Centro identità IAM</a></li> </ul>



Quale utente necessita dell'accesso programmatico?	Per	Come
		nella Guida di riferimento per SDK e strumenti AWS.
IAM	Utilizza credenziali temporanee e per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	Segui le istruzioni in <a href="#">Utilizzo di credenziali temporanee con le risorse AWS</a> nella Guida per l'utente IAM.
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	<p>Segui le istruzioni per l'interfaccia che desideri utilizzare.</p> <ul style="list-style-type: none"> <li>• Per la AWS CLI, consulta la pagina <a href="#">Autenticazione tramite credenziali utente IAM</a> nella Guida per l'utente dell'AWS Command Line Interface.</li> <li>• Per gli SDK e gli strumenti AWS, consulta la pagina <a href="#">Autenticazione con credenziali a lungo termine</a> nella Guida di riferimento per SDK e strumenti AWS.</li> <li>• Per le API AWS, consulta la pagina <a href="#">Gestione delle chiavi di accesso per utenti IAM</a> nella Guida per l'utente IAM.</li> </ul>

## Fase successiva

### [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 AWS CLI è già 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 di AWS Command Line Interface. Per verificare la versione della AWS CLI, esegui il comando seguente:

```
aws --version
```

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

```
aws-cli/1.16.63
```

Per configurare la AWS CLI

1. Scarica e configura la AWS CLI. Per 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 file AWS CLI della `config`. Puoi usare questo profilo quando esegui i comandi della AWS CLI. Per ulteriori informazioni sui profili denominati, consulta [Profili denominati](#) nella Guida per l'utente di 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 regioni AWS disponibili, consulta [Regioni ed endpoint](#) nel documento 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 accountAWS CLI, puoi provare l'esercizio successivo, in cui configurerai un'applicazione di esempio e verificherai la configurazione. end-to-end

## Fase successiva

### [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:

- [Creazione di due flussi di dati Amazon Kinesis](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice Java di streaming di Apache Flink](#)
- [Compilazione del codice dell'applicazione](#)

- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)
- [Fase successiva](#)

## Creazione di 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 di 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 create-stream della AWS CLI di Amazon Kinesis.

```
$ 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
```

## Scrittura di 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
```

## Download ed esame del codice Java di streaming di Apache Flink

Il codice dell'applicazione Java per questo esempio è disponibile presso 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`.

Nota 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 [Proprietà di runtime](#).

## Compilazione del 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](#).

## Compilazione del 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=1.15.3
```

- 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.

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 tuo nome utente, per renderlo globalmente univoco. Seleziona Avanti.

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 bucket ka-app-code - e scegli Carica. <username>
8. Nella fase Seleziona file, scegli Aggiungi file. Individua il file aws-kinesis-analytics-java-apps-1.0.jar creato nella fase precedente. Seleziona Avanti.
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 del servizio gestito per Apache 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, le tue risorse AWS Identity and Access Management (IAM) e Amazon CloudWatch Logs vengono create automaticamente. Quando crei l'applicazione utilizzando la AWS CLI, devi creare queste risorse separatamente.

### Argomenti

- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Creazione ed esecuzione dell'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 <https://console.aws.amazon.com/flink>



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 Descrizione, inserisci **My java test app**.
  - 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 ottenere 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 della policy IAM

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

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (`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 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
<b>ProducerConfigProperties</b>	<b>flink.inputstream.initpos</b>	<b>LATEST</b>

ID gruppo	Chiave	Valore
<b>ProducerConfigProperties</b>	<b>aws.region</b>	<b>us-west-2</b>
<b>ProducerConfigProperties</b>	<b>AggregationEnabled</b>	<b>false</b>

- 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 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

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 MyApplication pagina, 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.

## Creazione ed esecuzione dell'applicazione (AWS CLI)

In questa sezione, viene utilizzata la AWS CLI per creare ed eseguire l'applicazione del servizio gestito per Apache Flink. Il servizio gestito per Apache Flink utilizza il comando `kinesisanalyticsv2` della AWS CLI 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

`AKReadStreamWriteSinkStream`. 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 (ARN) (*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 for 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 > Crea 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: Rivedi. 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. Scegli Collega policy.
- c. Nella casella di ricerca, inserisci **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- d. Scegli la ReadSourceStreamWriteSinkStream policy AK e scegli Allega 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, utilizzerai 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 kinesisanalyticstv2 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, utilizzerai 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 richiesta seguente per interrompere l'applicazione:

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

L'applicazione è ora interrotta.

Aggiungi un'opzione di registrazione CloudWatch

Puoi usare il AWS CLI per aggiungere un flusso di CloudWatch log Amazon alla tua applicazione. Per informazioni sull'utilizzo di CloudWatch Logs con la tua applicazione, consulta [the section called "Configurazione della registrazione"](#).

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 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](#) della AWS 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 disabilitazione del controllo delle versioni](#).

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione e chiama `UpdateApplication`, specificando lo stesso bucket Amazon S3 e lo stesso nome 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 [the section called “Creazione di 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": "SAMPLEUehYngP87ex1nzYIGYgfhyvpDU"
        }
      }
    }
  }
}
```

## Fase successiva

### [Fase 4: eliminazione delle risorse AWS](#)

## Fase 4: eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 bucket e degli oggetti Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)
- [Fase successiva](#)

## Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello Managed Service for Apache Flink, scegliete. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

## Eliminazione del flusso di dati Kinesis

1. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/flink>
2. Nel pannello 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 del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.
2. <username>Scegli il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

## 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. 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/](https://console.aws.amazon.com/cloudwatch/).
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log MyApplication/aws/kinesis-analytics/.
4. Quindi scegli Elimina gruppo di log e conferma l'eliminazione.

## Fase successiva

### [Fase 5: passaggi successivi](#)

## Fase 5: passaggi successivi

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 streaming di dati AWS per Amazon Kinesis](#): la soluzione di streaming di dati AWS per Amazon Kinesis configura automaticamente i servizi AWS necessari per acquisire, archiviare, elaborare e distribuire dati di streaming con facilità. 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 di taxi di New York. La soluzione configura tutte le AWS risorse necessarie, come i ruoli e le politiche IAM, una dashboard e gli allarmi. CloudWatch CloudWatch
- [Soluzione di streaming di dati AWS per Amazon MSK](#): la soluzione di streaming di dati AWS per Amazon MSK fornisce modelli AWS CloudFormation 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 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 società 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.

- [Servizio gestito per Apache Flink: esempi](#): 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.

#### 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 in Flink Managed Service per Apache Flink.

# 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'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

- [Componenti dell'applicazione del servizio gestito di Apache Flink](#)
- [Prerequisiti](#)
- [Creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione delle risorse AWS](#)
- [Fasi successive](#)

## Componenti dell'applicazione del servizio gestito di 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 della tabella:** l'applicazione consuma dati utilizzando un'origine. Un connettore di origine legge i dati da un flusso di dati Kinesis, da un argomento Amazon MSK o simili. Per ulteriori informazioni, consulta [Origini dell'API Table](#).
- **Funzioni:** l'applicazione elabora i dati utilizzando una o più funzioni. Una funzione può trasformare, arricchire o aggregare dati.
- **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 Kinesis Data Firehose, un argomento Amazon MSK, un bucket Amazon S3 e così via. Per ulteriori informazioni, consulta [Sink dell'API Table](#).



Dopo aver creato, compilato e impacchettato il codice dell'applicazione, caricherai il pacchetto di codice in un bucket Amazon S3. Creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink. Inserisci la posizione del pacchetto di codice, un argomento di Amazon MSK come origine dati di streaming e in genere una posizione di streaming o di file che riceve i dati elaborati dall'applicazione.

## Prerequisiti

Prima di iniziare questo tutorial, completa la procedura in [Guida introduttiva ad Amazon Managed Service per Apache Flink \(DataStream API\)](#):

- [Fase 1: impostazione di un account AWS e creazione di un utente amministratore](#)
- [Fase 2: configurazione dell'AWS Command Line Interface \(AWS CLI\)](#)

Per iniziare, consulta [Crea 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 un argomento Amazon MSK come origine e un bucket Amazon S3 come sink.

Questa sezione contiene le fasi seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice Java di streaming di Apache Flink](#)
- [Compilazione del codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)
- [Fase successiva](#)

### Creazione delle risorse dipendenti

Prima di creare un servizio gestito per Apache Flink per questo esercizio, è necessario creare le seguenti risorse dipendenti:

- Un cloud privato virtuale (VPC) basato su Amazon VPC e un cluster Amazon MSK
- Un bucket Amazon S3 per archiviare il codice e l'output dell'applicazione (`ka-app-code-<username>`)

## Creazione di un VPC e un cluster Amazon MSK

Per creare un cluster VPC e Amazon MSK a cui accedere dall'applicazione del servizio gestito per Apache Flink, segui il tutorial [Nozioni di base sull'utilizzo di Amazon MSK](#).

Quando completi il tutorial, nota quanto segue:

- Registrare l'elenco dei server di bootstrap per il tuo cluster. Puoi ottenere l'elenco dei server di bootstrap con il seguente comando, sostituendo il `ClusterArn` con il nome della risorsa Amazon (ARN) del cluster MSK:

```
aws kafka get-bootstrap-brokers --region us-west-2 --cluster-arn ClusterArn
{...
  "BootstrapBrokerStringTls": "b-2.aws-kafka-tutorial-cluster.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-1.aws-kafka-tutorial-cluster.t79r6y.c4.kafka.us-
west-2.amazonaws.com:9094,b-3.aws-kafka-tutorial-cluster.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 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 a livello globale aggiungendo il tuo nome di accesso, ad esempio `ka-app-code-<username>`.

## Altre risorse

Quando crei la tua applicazione, Managed Service for Apache Flink crea le seguenti CloudWatch risorse Amazon se non esistono già:

- Un gruppo di log chiamato `/AWS/KinesisAnalytics-java/MyApplication`.
- Un flusso di log chiamato `kinesis-analytics-log-stream`.

## Scrittura di record di esempio nel flusso di input

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

1. Connettiti all'istanza client che hai creato nel [Fase 4: creazione di una macchina client](#) del tutorial [Nozioni di base sull'utilizzo di Amazon MSK](#).
2. Installa Python3, Pip e la libreria Python di Kafka:

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

3. Crea un file denominato `stock.py` con i seguenti contenuti. Sostituisci il valore `BROKERS` con l'elenco dei broker bootstrap che hai registrato in precedenza.

```
import datetime
import json
import random
import boto3

STREAM_NAME = "ExampleInputStream"

def get_data():
    return {
        'event_time': datetime.datetime.now().isoformat(),
        'ticker': random.choice(['AAPL', 'AMZN', 'MSFT', 'INTC', 'TBV']),
        'price': round(random.random() * 100, 2)}

def generate(stream_name, kinesis_client):
    while True:
        data = get_data()
        print(data)
        kinesis_client.put_record(
            StreamName=stream_name,
```

```
Data=json.dumps(data),
PartitionKey="partitionkey")

if __name__ == '__main__':
    generate(STREAM_NAME, boto3.client('kinesis', region_name='us-west-2'))
```

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

```
$ python3 stock.py
```

## Download ed esame del codice Java di streaming di Apache Flink

Il codice dell'applicazione Java 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-kinesis-data-analytics-examples.git
```

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

Nota 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 `StreamingJob.java` contiene il metodo `main` che definisce la funzionalità dell'applicazione.
- L'applicazione utilizza un `FlinkKafkaConsumer` per leggere dall'argomento Amazon MSK. Il seguente snippet crea un oggetto `FlinkKafkaConsumer`:

```
final FlinkKafkaConsumer<StockRecord> consumer = new
    FlinkKafkaConsumer<StockRecord>(kafkaTopic, new KafkaEventDeserializationSchema(),
    kafkaProps);
```

- L'applicazione crea connettori di origine e sink per accedere alle risorse esterne utilizzando oggetti `StreamExecutionEnvironment` e `TableEnvironment`.

- L'applicazione crea connettori di origine e sink utilizzando le proprietà dinamiche dell'applicazione, in modo da poter specificare i parametri dell'applicazione (come il bucket S3) senza ricompilare il codice.

```
//read the parameters from the Managed Service for Apache Flink environment
Map<String, Properties> applicationProperties =
    KinesisAnalyticsRuntime.getApplicationProperties();
Properties flinkProperties = null;

String kafkaTopic = parameter.get("kafka-topic", "AWSKafkaTutorialTopic");
String brokers = parameter.get("brokers", "");
String s3Path = parameter.get("s3Path", "");

if (applicationProperties != null) {
    flinkProperties = applicationProperties.get("FlinkApplicationProperties");
}

if (flinkProperties != null) {
    kafkaTopic = flinkProperties.get("kafka-topic").toString();
    brokers = flinkProperties.get("brokers").toString();
    s3Path = flinkProperties.get("s3Path").toString();
}
```

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

#### Note

Durante la creazione dell'applicazione, consigliamo vivamente di creare ed eseguire l'applicazione del servizio gestito per Apache Flink nella stessa regione del cluster Amazon MSK. Questo perché il connettore Flink Kafka è ottimizzato per impostazione predefinita per ambienti a bassa latenza. Se devi consumare dati di un cluster Kafka interregionale, valuta la possibilità di aumentare il valore di configurazione per `receive.buffer.byte`, ad esempio 2097152.

Per ulteriori informazioni, consulta [Configurazioni MSK personalizzate](#).

## Compilazione del 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](#).

### Compilazione del 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=1.15.3
```

- 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.

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 () 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 tuo nome utente, per renderlo globalmente univoco. Seleziona Avanti.
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 bucket ka-app-code - e scegli Carica. *<username>*
8. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `aws-kinesis-analytics-java-apps-1.0.jar` creato nella fase precedente. Seleziona Avanti.
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 del servizio gestito per Apache 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 <https://console.aws.amazon.com/flink>
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 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 ottenere 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 della policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere al bucket Amazon S3.

Per modificare la policy IAM per aggiungere le autorizzazioni del bucket S3

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "S3",
      "Effect": "Allow",
      "Action": [
        "s3:Abort*",
        "s3:DeleteObject*",
        "s3:GetObject*",
```



```

        "s3:GetBucket*",
        "s3:List*",
        "s3:ListBucket",
        "s3:PutObject"
    ],
    "Resource": [
        "arn:aws:s3:::ka-app-code-<username>",
        "arn:aws:s3:::ka-app-code-<username>/*"
    ]
},
{
    "Sid": "DescribeLogGroups",
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogGroups"
    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
    ]
},
{
    "Sid": "DescribeLogStreams",
    "Effect": "Allow",
    "Action": [
        "logs:DescribeLogStreams"
    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
    ]
},
{
    "Sid": "PutLogEvents",
    "Effect": "Allow",
    "Action": [
        "logs:PutLogEvents"
    ],
    "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
}
]

```

}

## Configurazione dell'applicazione

Per configurare l'applicazione, utilizza la procedura seguente.

Per configurare l'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. Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
<b>FlinkApplicationProperties</b>	<b>kafka-topic</b>	<b>AWSKafkaTutorialTopic</b>
<b>FlinkApplicationProperties</b>	<b>brokers</b>	<b><i>Your Amazon MSK cluster's Bootstrap Brokers list</i></b>
<b>FlinkApplicationProperties</b>	<b>s3Path</b>	<b>ka-app-code- <i>&lt;username&gt;</i></b>
<b>FlinkApplicationProperties</b>	<b>security.protocol</b>	<b>SSL</b>
<b>FlinkApplicationProperties</b>	<b>ssl.truststore.location</b>	<b>/usr/lib/jvm/java-11-amazon-corretto</b>

ID gruppo	Chiave	Valore
		<b>/lib/security/cacerts</b>
<b>FlinkApplicationProperties</b>	<b>ssl.truststore.password</b>	<b>changeit</b>

- In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
- Per la CloudWatch registrazione, seleziona la casella di controllo Abilita.
- Nella sezione Cloud privato virtuale (VPC), scegli la configurazione VPC basata sul cluster Amazon MSK. Scegli AWSKafkaTutorialCluster.
- Scegli Aggiorna.

#### 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

Per eliminare l'applicazione, utilizza la procedura seguente.

Per eseguire l'applicazione

- Nella MyApplication pagina, scegli Esegui. Conferma l'operazione.
- Quando l'applicazione è in esecuzione, aggiorna la pagina. La console mostra il Grafico dell'applicazione.
- Dal tuo client Amazon EC2, esegui lo script Python creato in precedenza per scrivere record nel cluster Amazon MSK affinché l'applicazione possa elaborare:

```
$ python3 stock.py
```

## Interruzione dell'applicazione

Per interrompere l'applicazione, nella MyApplication pagina, scegli Stop. Conferma l'operazione.

## Fase successiva

### [Eliminazione delle risorse AWS](#)

## Eliminazione delle risorse AWS

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

Questo argomento contiene le sezioni seguenti:

- [Eliminazione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione del cluster Amazon MSK](#)
- [Eliminazione del VPC](#)
- [Eliminazione del bucket e degli oggetti Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)
- [Fase successiva](#)

## 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/kinesis>.
2. Nel pannello Managed Service for Apache Flink, scegliete. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e conferma l'eliminazione.

## Eliminazione del cluster Amazon MSK

Per eliminare il cluster Amazon MSK, segui la [Fase 8: eliminazione del cluster Amazon MSK](#) nella Guida per gli sviluppatori di Streaming gestito da Amazon per Apache Kafka.

## Eliminazione del VPC

Per eliminare l'Amazon VPC, procedi come indicato di seguito:

- Apri la console Amazon VPC.
- Scegli il tuo VPC.
- In Operazioni, scegli Elimina VPC.

## Eliminazione del bucket e degli oggetti Amazon S3

Per eliminare gli oggetti e il bucket S3, utilizza la procedura seguente.

Per eliminare gli oggetti e il bucket S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.
2. <username>Scegliete il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare 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-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. Scegli Elimina ruolo e conferma l'eliminazione.

## CloudWatch Elimina le tue risorse

Usa la seguente procedura per eliminare CloudWatch le tue risorse.

Per eliminare le tue CloudWatch risorse

1. Apri la CloudWatch console all'[indirizzo https://console.aws.amazon.com/cloudwatch/](https://console.aws.amazon.com/cloudwatch/).
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log MyApplication/aws/kinesis-analytics/.
4. Quindi scegli Elimina gruppo di log e conferma l'eliminazione.

## Fase successiva

[Fasi successive](#)

## Fasi successive

Ora che hai creato ed eseguito un'applicazione del servizio gestito per Apache Flink che utilizza l'API Table, consulta [Fase 5: passaggi successivi](#) in [Guida introduttiva ad 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 prima applicazione.

## Argomenti

- [Nozioni di base su Pyflink: l'interprete Python per Apache | Amazon Web Services](#)
- [Componenti dell'applicazione servizio gestito per Apache Flink](#)
- [Prerequisiti](#)
- [Creazione ed esecuzione di un servizio gestito per Apache Flink per applicazioni Python](#)
- [Eliminazione delle risorse AWS](#)

### 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.

## Nozioni di base su Pyflink: l'interprete Python per Apache | Amazon Web Services

Prima di iniziare, ti invitiamo a guardare il video seguente:

[Nozioni di base su Pyflink: l'interprete Python per Apache | Amazon Web Services](#)

## Componenti dell'applicazione servizio gestito per Apache Flink

Per elaborare i dati, l'applicazione del servizio per Apache Flink utilizza un'applicazione Python che elabora l'input e produce l'output utilizzando il runtime di Apache Flink.

L'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 della tabella:** l'applicazione consuma dati utilizzando un'origine. Un connettore di origine legge i dati da un flusso di dati Kinesis, da un argomento Amazon MSK o simili. Per ulteriori informazioni, consulta [Origini dell'API Table](#).
- **Funzioni:** l'applicazione elabora i dati utilizzando una o più funzioni. Una funzione può trasformare, arricchire o aggregare dati.
- **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 Kinesis Data Firehose, un argomento Amazon MSK, un bucket Amazon S3 e così via. Per ulteriori informazioni, consulta [Sink dell'API Table](#).

Dopo aver creato e incluso in un pacchetto il codice dell'applicazione, carica il pacchetto di codice in un bucket Amazon S3. Puoi quindi creare un'applicazione del servizio gestito per Apache Flink. Inserisci la posizione del pacchetto di codice, un'origine dati di streaming e in genere una posizione di streaming o di file che riceve i dati elaborati dall'applicazione.

## Prerequisiti

Prima di iniziare questo tutorial, completa le prime due fasi di [Guida introduttiva ad Amazon Managed Service per Apache Flink \(DataStream API\)](#):

- [Fase 1: impostazione di un account AWS e creazione di un utente amministratore](#)
- [Fase 2: configurazione dell'AWS Command Line Interface \(AWS CLI\)](#)

Per iniziare, consulta [Crea un'applicazione](#).

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

In questo esercizio, 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](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice Java di streaming Apache Flink](#)
- [Aggiunta di dipendenze di terze parti alle app Python](#)
- [Caricamento del codice Python di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)
- [Fase successiva](#)

## 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 archiviare il codice e l'output dell'applicazione (ka-app-code-*<username>*)

### 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). 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 di 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 create-stream della AWS CLI di Amazon Kinesis.

```
$ 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
```

## 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 a livello globale aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

## Altre risorse

Quando crei la tua applicazione, Managed Service for Apache Flink crea le seguenti CloudWatch risorse Amazon se non esistono già:

- Un gruppo di log chiamato `/AWS/KinesisAnalytics-java/MyApplication`.
- Un flusso di log chiamato `kinesis-analytics-log-stream`.

## Scrittura di 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, inserisci il comando seguente:

```
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.

## Download ed esame del codice Java di streaming 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. Installa il client Git se non lo hai già fatto. Per ulteriori informazioni, consulta la sezione [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/GettingStarted`.

Il codice dell'applicazione si trova nei file `getting_started.py`. Nota quanto segue riguardo al codice dell'applicazione:

- L'applicazione utilizza un'origine 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_table(output_table_name, output_stream, output_region)
```

La `create_table` nzione utilizza un comando SQL per creare una tabella supportata dall'origine di streaming:

```
def create_table(table_name, stream_name, region, stream_initpos = None):  
    init_pos = "\n'scan.stream.initpos' = '{0}',".format(stream_initpos) if  
    stream_initpos is not None else ''  
  
    return """ CREATE TABLE {0} (  
        ticker VARCHAR(6),  
        price DOUBLE,
```

```

        event_time TIMESTAMP(3),
        WATERMARK FOR event_time AS event_time - INTERVAL '5' SECOND
    )
    PARTITIONED BY (ticker)
    WITH (
        'connector' = 'kinesis',
        'stream' = '{1}',
        'aws.region' = '{2}',{3}
        'format' = 'json',
        'json.timestamp-format.standard' = 'ISO-8601'
    ) """.format(table_name, stream_name, region, init_pos)
}

```

- L'applicazione crea due tabelle, quindi scrive il contenuto di una tabella nell'altra.

```

# 2. Creates a source table from a Kinesis Data Stream
table_env.execute_sql(
    create_table(input_table_name, input_stream, input_region)
)

# 3. Creates a sink table writing to a Kinesis Data Stream
table_env.execute_sql(
    create_table(output_table_name, output_stream, output_region, stream_initpos)
)

# 4. Inserts the source table data into the sink table
table_result = table_env.execute_sql("INSERT INTO {0} SELECT * FROM {1}"
    .format(output_table_name, input_table_name))

```

- L'applicazione utilizza il connettore Flink, dal file [flink-sql-connector-kinesis](#) \_2.12/1.15.2.

## Aggiunta di dipendenze di terze parti alle app Python

Quando si utilizzano pacchetti python di terze parti (come [boto3](#)), è necessario aggiungere le relative dipendenze transitive e le proprietà necessarie per indirizzare tali dipendenze. Ad alto livello, per quanto riguarda PyPi le dipendenze, puoi copiare i file e le cartelle che si trovano all'interno della `site-packages` cartella degli ambienti python per creare una struttura di directory come la seguente:

```
PythonPackages
```

```
# README.md
# python-packages.py
#
####my_deps
    ####boto3
    # # session.py
    # # utils.py
    # # ...
    #
    ####botocore
    # # args.py
    # # auth.py
    # ...
    ####mynonpypimodule
    # # mymodulefile1.py
    # # mymodulefile2.py
    ...
####lib
# # flink-sql-connector-kinesis-1.15.2.jar
# # ...
...
```

Per aggiungere boto3 come dipendenza di terze parti:

1. Crea un ambiente Python autonomo (conda o simile) sulla tua macchina locale con le dipendenze richieste.
2. Nota l'elenco iniziale dei pacchetti nella cartella `site_packages` di quell'ambiente.
3. `pip-install` tutte le dipendenze richieste per la tua app.
4. Annota i pacchetti che sono stati aggiunti alla cartella `site_packages` dopo il passaggio 3 precedente. Queste sono le cartelle da includere nel pacchetto (sotto la cartella `my_deps`), organizzate come mostrato sopra. Ciò ti consentirà di acquisire una diff dei pacchetti tra i passaggi 2 e 3 per individuare le dipendenze dei pacchetti corrette per la tua applicazione.
5. Fornisci `my_deps/` come argomento per la proprietà `pyFiles` nel gruppo di proprietà `kinesis.analytics.flink.run.options` come descritto di seguito per la proprietà `jarfiles`. Flink consente anche di specificare le dipendenze di Python utilizzando la funzione [add\\_python\\_file](#), ma è importante tenere presente che è sufficiente specificare l'una o l'altra, non entrambe.

**Note**

Non è necessario assegnare un nome alla cartella `my_deps`. La parte importante è registrare le dipendenze utilizzando `pyFiles` o `add_python_file`. Un esempio può essere trovato in [Come usare boto3 all'interno di PyFlink](#).


## Caricamento del codice Python di streaming di Apache Flink

In questa sezione, viene creato un bucket Amazon S3 e si procede al caricamento del codice dell'applicazione.

Per caricare il codice dell'applicazione utilizzando la console:


1. Usa la tua applicazione di compressione preferita per comprimere i file [https://mvnrepository.com/artifact/org.apache.flink/flink-sql-connector-kinesis\\_2.12/1.15.2](https://mvnrepository.com/artifact/org.apache.flink/flink-sql-connector-kinesis_2.12/1.15.2). Assegna un nome all'archivio `myapp.zip`. Se includi la cartella esterna nel tuo archivio, devi includerla nel percorso con il codice nei tuoi file di configurazione: `GetStarted/getting-started.py`.
2. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.
3. Seleziona Crea bucket.
4. Inserisci **ka-app-code-*<username>*** nel campo Nome bucket. Aggiungi un suffisso al nome del bucket, ad esempio il tuo nome utente, per renderlo globalmente univoco. Seleziona Avanti.
5. Nella fase Configura opzioni, non modificare le impostazioni e scegli Successivo.
6. Nella fase Imposta autorizzazioni, non modificare le impostazioni e scegli Successivo.
7. Seleziona Crea bucket.
8. Nella console Amazon S3, scegli il bucket `ka-app-code-<username>` e scegli Carica. `<username>`
9. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `myapp.zip` creato nella fase precedente. Seleziona Avanti.
10. Non è necessario modificare alcuna delle impostazioni dell'oggetto, quindi scegli Carica.

Per caricare il codice dell'applicazione utilizzando la AWS CLI:

 Note

Non utilizzare le funzionalità di compressione in Finder (macOS) o Windows Explorer (Windows) per creare l'archivio `myapp.zip`. Il codice dell'applicazione potrebbe risultare non valido.

1. Usa la tua applicazione di compressione preferita per comprimere i file <https://mvnrepository.com/artifact/org.apache.flink/flink-streaming-file-sink.py> `sql-connector-kinesis_2.12/1.15.2`.

 Note

Non utilizzare le funzionalità di compressione in Finder (macOS) o Windows Explorer (Windows) per creare l'archivio `myapp.zip`. Il codice dell'applicazione potrebbe risultare non valido.

2. [Utilizzate l'applicazione di compressione preferita per comprimere i file e https://mvnrepository.com/artifact/org.apache.flink/1.15.2.getting-started.py](https://mvnrepository.com/artifact/org.apache.flink/1.15.2.getting-started.py) `flink-sql-connector-kinesis` Assegna un nome all'archivio `myapp.zip`. Se includi la cartella esterna nel tuo archivio, devi includerla nel percorso con il codice nei tuoi file di configurazione: `GettingStarted/getting-started.py`.
3. Esegui il comando seguente:

```
$ aws s3 --region aws region cp myapp.zip s3://ka-app-code-<username>
```

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. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/flink>
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 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 ottenere 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

Per configurare l'applicazione, utilizza la procedura seguente.

Per configurare l'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 **myapp.zip**.
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. In Proprietà, scegli Aggiungi gruppo.
  5. Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
<b>consumer.config.0</b>	<b>input.stream.name</b>	<b>ExampleInputStream</b>
<b>consumer.config.0</b>	<b>aws.region</b>	<b>us-west-2</b>
<b>consumer.config.0</b>	<b>scan.stream.initpos</b>	<b>LATEST</b>

Selezionare Salva.

6. In Proprietà, scegli nuovamente Aggiungi gruppo.
7. Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
<b>producer.config.0</b>	<b>output.stream.name</b>	<b>ExampleOutputStream</b>
<b>producer.config.0</b>	<b>aws.region</b>	<b>us-west-2</b>
<b>producer.config.0</b>	<b>shard.count</b>	<b>1</b>

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 [Specificazione dei file di codice](#).
9. Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
<b>kinesis.analytics.flink.run.options</b>	<b>python</b>	<b>getting-started.py</b>
<b>kinesis.analytics.flink.run.options</b>	<b>jarfile</b>	<b>flink-sql-connector-kinesis-1.15.2.jar</b>

- 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 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

## Modifica della policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere al bucket Amazon S3.

Per modificare la policy IAM per aggiungere le autorizzazioni del bucket S3

- Apri la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
- Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
- Nella pagina Riepilogo, scegli Modifica policy. Seleziona la scheda JSON.
- Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (**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/myapp.zip"
      ]
    },
    {
      "Sid": "DescribeLogGroups",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogGroups"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:*"
      ]
    },
    {
      "Sid": "DescribeLogStreams",
      "Effect": "Allow",
      "Action": [
        "logs:DescribeLogStreams"
      ],
      "Resource": [
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:*"
      ]
    },
    {
      "Sid": "PutLogEvents",
      "Effect": "Allow",
      "Action": [
        "logs:PutLogEvents"
      ],
      "Resource": [
```

```
        "arn:aws:logs:us-west-2:012345678901:log-group:/aws/kinesis-
analytics/MyApplication:log-stream:kinesis-analytics-log-stream"
    ]
  },
  {
    "Sid": "ReadInputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleInputStream"
  },
  {
    "Sid": "WriteOutputStream",
    "Effect": "Allow",
    "Action": "kinesis:*",
    "Resource": "arn:aws:kinesis:us-west-2:012345678901:stream/
ExampleOutputStream"
  }
]
}
```

## 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.

## Fase successiva

### [Eliminazione delle risorse AWS](#)

## Eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 bucket e degli oggetti 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/kinesis>.
2. Nel pannello Managed Service for Apache Flink, scegliete. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e conferma l'eliminazione.

## Eliminazione del flusso di dati Kinesis

1. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/flink>
2. Nel pannello 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 del bucket e degli oggetti Amazon S3

Per eliminare gli oggetti e il bucket S3, utilizza la procedura seguente.

Per eliminare gli oggetti e il bucket S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.
2. <username>Scegli il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare 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-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. Scegli Elimina ruolo e conferma l'eliminazione.

## CloudWatch Elimina le tue risorse

Usa la seguente procedura per eliminare CloudWatch le tue risorse.

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 di log MyApplication/aws/kinesis-analytics/.
4. 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, 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 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](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice dell'applicazione](#)
- [Per creare e compilare il codice dell'applicazione](#)
- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Creazione ed esecuzione dell'applicazione \(CLI\)](#)
- [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 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 AWS CLI di Amazon Kinesis per la creazione di un 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 risorse Amazon CloudWatch, se non esistono già:

- Un gruppo di log denominato `/AWS/KinesisAnalytics-java/MyApplication`
- Un flusso di log denominato `kinesis-analytics-log-stream`

## Scrittura di 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.

## Download ed esame del 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`.

Nota 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 oggetto `StreamExecutionEnvironment`.
- 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 su <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. Seleziona 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. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/flink>
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, immetti **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 pagina MyApplication, 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 Accedi 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
<b>ConsumerConfigProperties</b>	<b>input.stream.name</b>	<b>ExampleInputStream</b>
<b>ConsumerConfigProperties</b>	<b>aws.region</b>	<b>us-west-2</b>
<b>ConsumerConfigProperties</b>	<b>flink.stream.initpos</b>	<b>LATEST</b>

Seleziona Salva.

- In Proprietà, scegli di nuovo Aggiungi gruppo.
- Immetti i seguenti dati:

ID gruppo	Chiave	Valore
<b>ProducerConfigProperties</b>	<b>output.stream.name</b>	<b>ExampleOutputStream</b>
<b>ProducerConfigProperties</b>	<b>aws.region</b>	<b>us-west-2</b>

- In Monitoraggio, accertati che il Monitoraggio del livello dei parametri sia impostato su Applicazione.
- In Registrazione CloudWatch, seleziona la casella di controllo Abilita.
- Scegli Aggiorna.

#### Note

Quando scegli di abilitare la registrazione Amazon CloudWatch, 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 della 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. Apri 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 gli ID account di esempio (`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 pagina MyApplication, scegli Stop. Confermare 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 comando della AWS CLI `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 `AKReadStreamWriteSinkStream`. 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 (ARN) (**(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 istruzioni dettagliate per la creazione di una policy di autorizzazione, consulta [Tutorial: Creare e collegare la prima policy gestita dal cliente](#) nella Guida per l'utente IAM.

## 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. Apri 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.

Ora hai creato il ruolo di esecuzione del servizio utilizzato dall'applicazione per accedere alle risorse. Prendi nota dell'ARN del nuovo ruolo.

Per le istruzioni dettagliate di creazione di un ruolo, consulta [Creazione di un ruolo IAM \(console\)](#) nella Guida per l'utente IAM.

## Creazione dell'applicazione

Salva 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_15",
  "ServiceExecutionRole": "arn:aws:iam::012345678901:role/MF-stream-rw-role",
  "ApplicationConfiguration": {
    "ApplicationCodeConfiguration": {
      "CodeContent": {
        "S3ContentLocation": {
          "BucketARN": "arn:aws:s3:::ka-app-code-username",
```

```

        "FileKey": "getting-started-scala-1.0.jar"
      }
    },
    "CodeContentType": "ZIPFILE"
  },
  "EnvironmentProperties": {
    "PropertyGroups": [
      {
        "PropertyGroupId": "ConsumerConfigProperties",
        "PropertyMap" : {
          "aws.region" : "us-west-2",
          "stream.name" : "ExampleInputStream",
          "flink.stream.initpos" : "LATEST"
        }
      },
      {
        "PropertyGroupId": "ProducerConfigProperties",
        "PropertyMap" : {
          "aws.region" : "us-west-2",
          "stream.name" : "ExampleOutputStream"
        }
      }
    ]
  }
},
"CloudWatchLoggingOptions": [
  {
    "LogStreamARN": "arn:aws:logs:us-west-2:012345678901:log-
group:MyApplication:log-stream:kinesis-analytics-log-stream"
  }
]
}

```

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.

## Avvio dell'applicazione

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

## Per avviare l'applicazione

1. Salva 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 console Amazon CloudWatch per verificare che l'applicazione funzioni.

## Interruzione dell'applicazione

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

Per interrompere l'applicazione

1. Salva 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 di registrazione CloudWatch

Puoi utilizzare la AWS CLI per aggiungere un flusso di log Amazon CloudWatch alla tua applicazione. Per informazioni sull'utilizzo dei log CloudWatch con l'applicazione, consulta [Configurazione della registrazione dell'applicazione](#).

## 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. Salva 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 della CLI [UpdateApplication](#).

### 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 `UpdateApplication`, specificando lo stesso bucket Amazon S3 e lo stesso nome 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": "SAMPLEUehYngP87ex1nzYIGYgfhyvDU"
    }
}
}
```

## Pulizia delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 del bucket e degli oggetti Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [Eliminazione delle risorse CloudWatch](#)

## Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console del servizio gestito per Apache Flink all'indirizzo <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. Apri la console Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello del flusso di dati Kinesis, scegli ExampleInputStream.
3. Nella pagina ExampleInputStream, scegli Elimina flusso Kinesis e conferma l'eliminazione.
4. Nella pagina Flussi di Kinesis, scegli ExampleOutputStream, scegli Operazioni, scegli Elimina e 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 bucket ka-app-code-**<username>**.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

## Eliminazione delle risorse IAM

1. Apri 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 policy 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-MyApplication-us-west-2.
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

## Eliminazione delle risorse CloudWatch

1. Apri la console CloudWatch all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log /aws/Kinesis-Analytics/MyApplication.
4. Quindi scegli Elimina gruppo di log e conferma l'eliminazione.

# Creazione di applicazioni del servizio gestito per Apache Flink con 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 [Utilizzo di CloudFormation con il servizio gestito per Apache Flink](#).

Questo argomento contiene le sezioni seguenti:

- [Utilizzo di Apache Beam con il servizio gestito per Apache Flink](#)
- [Funzionalità Beam](#)
- [Creazione di un'applicazione utilizzando Apache Beam](#)

## Utilizzo di Apache Beam con il servizio gestito per Apache Flink

Nota 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 Beam

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](#).

## Note

Per impostare i prerequisiti richiesti per questo esercizio, completa innanzitutto l'esercizio [Guida introduttiva \(API\) DataStream](#).

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice dell'applicazione](#)
- [Compilazione del codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione delle risorse AWS](#)
- [Fasi successive](#)

## 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 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 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 univoco a livello globale aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

## Scrittura di 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.

## Download ed esame del 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 nel file `BasicBeamStreamingJob.java`. Nota quanto segue riguardo al codice dell'applicazione:

- L'applicazione utilizza [ParDo](#) Apache Beam 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());
        }
    }
}
```

## Compilazione del 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 [Prerequisiti](#) nel tutorial [Guida introduttiva \(API\) DataStream](#).
2. Compila l'applicazione con il seguente comando:

```
mvn package -Dflink.version=1.15.3 -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 bucket `ka-app-code-<username>` e seleziona 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 di un'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. Apri la console del servizio gestito per Apache Flink all'indirizzo `https://console.aws.amazon.com/flink`
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 ottenere 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: `kinesis-analytics-MyApplication-us-west-2`

## Modifica della policy IAM

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

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (`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>/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 pagina MyApplication, 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
<b>BeamApplicationProperties</b>	<b>InputStreamName</b>	<b>ExampleInputStream</b>
<b>BeamApplicationProperties</b>	<b>OutputStreamName</b>	<b>ExampleOutputStream</b>
<b>BeamApplicationProperties</b>	<b>AwsRegion</b>	<b>us-west-2</b>

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

### Note

Quando scegli di abilitare la registrazione CloudWatch, 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 console CloudWatch per verificare che l'applicazione funzioni.

## Eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 del bucket e degli oggetti Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [Eliminazione delle risorse CloudWatch](#)

## Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console del servizio gestito per Apache Flink all'indirizzo <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. Apri la console Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello del flusso di dati Kinesis, scegli ExampleInputStream.

3. Nella pagina ExampleInputStream, scegli Elimina flusso Kinesis e conferma l'eliminazione.
4. Nella pagina Flussi di Kinesis, scegli ExampleOutputStream, scegli Operazioni, scegli Elimina e conferma l'eliminazione.

## Eliminazione del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 su <https://console.aws.amazon.com/s3/>.
2. Scegli il bucket ka-app-code-**<username>**.
3. Per confermare l'eliminazione, scegli Elimina, quindi inserisci il nome del bucket.

## Eliminazione delle risorse IAM

1. Apri 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 policy 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-MyApplication-us-west-2.
8. Quindi scegli Elimina ruolo e conferma l'eliminazione.

## Eliminazione delle risorse CloudWatch

1. Apri la console CloudWatch all'indirizzo <https://console.aws.amazon.com/cloudwatch/>.
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log /aws/Kinesis-Analytics/MyApplication.
4. Quindi scegli Elimina gruppo di log e conferma l'eliminazione.

## Fasi successive

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

- [Sviluppo a livello locale di applicazioni Apache Flink prima dell'implementazione sul servizio gestito per Apache Flink per Apache Flink](#)
- [Rilevamento di eventi con il servizio gestito per Apache Flink Studio](#)
- [AWS Soluzione per lo streaming di dati per Amazon Kinesis](#)
- [Laboratorio clickstream con Apache Flink e Apache Kafka](#)
- [Dimensionamento personalizzato utilizzando il dimensionamento automatico delle applicazioni](#)
- [CloudWatch Pannello di controllo Amazon](#)
- [AWS Soluzione per lo streaming di dati per Amazon Kinesis](#)
- [Altro servizio gestito per soluzioni Apache Flink su GitHub](#)

## Sviluppo a livello locale di applicazioni Apache Flink prima dell'implementazione sul servizio gestito per Apache Flink per 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 sul servizio gestito per Apache Flink per Apache Flink.

La soluzione è disponibile a questo indirizzo: [Guida introduttiva allo sviluppo locale con Apache Flink](#)

## Rilevamento di eventi con il servizio gestito 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.

La soluzione è disponibile a questo indirizzo: [Rilevamento di eventi con il servizio gestito per Apache Flink per Apache Flink](#)



## AWS Soluzione per lo streaming di dati per Amazon Kinesis

La soluzione AWS per lo streaming di dati per Amazon Kinesis configura automaticamente i servizi AWS necessari per acquisire, archiviare, elaborare e distribuire dati in streaming con facilità. La soluzione offre diverse opzioni per risolvere i casi d'uso relativi allo streaming di dati. 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 di taxi di New York.

Ogni soluzione include i seguenti componenti:

- un pacchetto AWS CloudFormation per implementare l'esempio completo;
- Una dashboard per la visualizzazione delle metriche delle applicazioni. CloudWatch
- CloudWatch allarmi sulle metriche applicative più rilevanti.
- tutti i ruoli e le politiche IAM necessari.

La soluzione è disponibile a questo indirizzo: [Soluzione per lo streaming di dati per Amazon Kinesis](#)

## 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.

La soluzione è disponibile a questo indirizzo: [Laboratorio clickstream](#)

## Dimensionamento personalizzato utilizzando il dimensionamento automatico delle applicazioni

Un esempio che aiuta gli utenti a scalare automaticamente le proprie applicazioni Managed Service for Apache Flink utilizzando Application Auto Scaling. Ciò consente agli utenti di configurare politiche di dimensionamento e attributi di dimensionamento personalizzati.

Le soluzioni sono disponibili a questo indirizzo:

- [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](#).

## CloudWatch Pannello di controllo 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.

La soluzione è disponibile a questo indirizzo: [Servizio gestito per la dashboard delle metriche di Apache Flink](#)

## AWS Soluzione per lo streaming di dati per Amazon Kinesis

La AWS soluzione per lo streaming di dati per Amazon MSK fornisce AWS CloudFormation modelli in cui i dati fluiscono attraverso produttori, archiviazione in streaming, consumatori e destinazioni.

La soluzione è disponibile a questo indirizzo: [AWSSoluzione per lo streaming di dati per Amazon MSK](#)

## Altro servizio gestito per soluzioni Apache Flink su GitHub

end-to-end Gli 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](#)

# Utilità

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 azionare regolarmente i savepoint e gli snapshot, per garantire un ripristino ottimale in seguito a eventuali errori. 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, vedere [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](#)

# Servizio gestito per Apache Flink: esempi

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 un servizio gestito per le applicazioni Apache Flink e testare i risultati.

Prima di esplorare questi esempi, ti consigliamo di esaminare quanto segue:

- [Come funziona](#)
- [Guida introduttiva \(API\) DataStream](#)

## 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

Gli esempi seguenti mostrano come creare applicazioni utilizzando l'API Apache 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 in Kinesis Data Firehose](#)
- [Esempio: lettura da un flusso Kinesis in un account diverso.](#)
- [Tutorial: utilizzo di un truststore personalizzato con Amazon MSK](#)

## Esempio: finestra a cascata

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 [Guida introduttiva \(API\) DataStream](#) .

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice dell'applicazione](#)
- [Compilazione del codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione 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 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** 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 univoco a livello globale aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

## Scrittura di 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.

## Download ed esame del 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 la sezione [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`. Nota 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());
```

## Compilazione del 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 [Prerequisiti](#) nel tutorial [Guida introduttiva \(API\) DataStream](#).
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 bucket ka-app-code - e scegli Carica. <username>
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 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. Apri la console del servizio gestito per Apache Flink all'indirizzo `https://console.aws.amazon.com/flink`
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 ottenere 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 della policy IAM**

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

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (`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 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 Accedi 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, 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`

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 le metriche del servizio gestito per Apache Flink sulla CloudWatch console per verificare che l'applicazione funzioni.

## Eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 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 <https://console.aws.amazon.com/flink>
2. nel pannello Managed Service for 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 Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello 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 del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.
2. <username>Scegli il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

## 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. 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 di log MyApplication/aws/kinesis-analytics/.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

## Esempio: finestra scorrevole

### Note

Per impostare i prerequisiti richiesti per questo esercizio, completa prima l'esercizio [Guida introduttiva \(API\) DataStream](#) .

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice dell'applicazione](#)
- [Compilazione del codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)

- [Eliminazione 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 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 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 univoco a livello globale aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

## Scrittura di 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.

## Download ed esame del 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 la sezione [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`. Nota 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());
```

## Compilazione del 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 [Prerequisiti](#) nel tutorial [Guida introduttiva \(API\) DataStream](#).
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 bucket `ka-app-code-`, quindi scegli Carica. `<username>`
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 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. Apri la console del servizio gestito per Apache Flink all'indirizzo `https://console.aws.amazon.com/flink`
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 ottenere 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 della policy IAM

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

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (`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 Accedi 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 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

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 kinesisanalyticstv2 update-application
  --application-name MyApplication
  --current-application-version-id <VersionId>
  --application-configuration-update "{\"FlinkApplicationConfigurationUpdate
\": { \"ParallelismConfigurationUpdate\": {\"ParallelismUpdate\": 5,
  \"ConfigurationTypeUpdate\": \"CUSTOM\" }}}"
```

Puoi recuperare l'ID della versione corrente dell'applicazione utilizzando le azioni o.

[DescribeApplicationListApplications](#)

## 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 le metriche del servizio gestito per Apache Flink sulla CloudWatch console per verificare che l'applicazione funzioni.

## Eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 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 <https://console.aws.amazon.com/flink>
2. Nel pannello Managed Service for Apache Flink, scegliete. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

### Eliminazione dei flussi di dati Kinesis

1. Apri la console Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello 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 del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.
2. <username>Scegli il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

### 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. 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/](https://console.aws.amazon.com/cloudwatch/).
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log MyApplication/aws/kinesis-analytics/.
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 [Guida introduttiva \(API\) DataStream](#) .

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice dell'applicazione](#)
- [Modifica del codice dell'applicazione](#)
- [Compilazione del codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)
- [Verifica dell'output dell'applicazione](#)
- [Facoltativo: personalizzazione dell'origine e del sink](#)
- [Eliminazione 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 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>***. Crea due cartelle (**code** e **data**) nel bucket Amazon S3.

L'applicazione crea le seguenti risorse se non esistono già: CloudWatch

- Un gruppo di log chiamato `/AWS/KinesisAnalytics-java/MyApplication`.
- Un flusso di log chiamato `kinesis-analytics-log-stream`.

## Scrittura di 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.

## Download ed esame del 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 la sezione [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`. Nota 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";
```

## Compilazione del 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 [Prerequisiti](#) nel tutorial [Guida introduttiva \(API\) DataStream](#).
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 bucket ka-app-code -, accedi alla cartella del codice e scegli Carica. <username>
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 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. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/flink>
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 ottenere 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:

- 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 della policy IAM

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

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (`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 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 **code/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. 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, 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`

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 “Compilazione del 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.DefaultRollingPol
import
  org.apache.flink.streaming.api.functions.sink.filesystem.bucketassigners.DateTimeBucketAss
```

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](#) azione 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();
    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](#).

## Eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 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 <https://console.aws.amazon.com/flink>
2. Nel pannello Managed Service for Apache Flink, scegliete. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e conferma l'eliminazione.

### Eliminazione del flusso di dati Kinesis

1. Apri la console Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello Kinesis Data Streams, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e 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 ka-app-codebucket -<username>.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

## 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. 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/](https://console.aws.amazon.com/cloudwatch/).
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log MyApplication/aws/kinesis-analytics/.
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

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 [Guida introduttiva \(API\) DataStream](#).

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](#)
- [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, nota 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 AWSKafkaTutorialTopicDestination
```

- Registra l'elenco dei server di bootstrap per il cluster. È possibile ottenere l'elenco dei server di bootstrap con il seguente comando (sostituirlo `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 la sezione [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 [Guida introduttiva \(API\) DataStream](#).

### Note

Se il bucket Amazon S3 è stato eliminato dal tutorial Nozioni di base, segui nuovamente il passaggio [the section called “Caricamento del codice Java di streaming di Apache Flink”](#).

1. Nella console Amazon S3, scegli il bucket ka-app-code - e scegli Carica. <username>
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 all'indirizzo `https://console.aws.amazon.com/flink`
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 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 ottenere 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 `KafkaGettingStartedJob-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.


### 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.

4. In Proprietà, scegli Aggiungi gruppo. Inserisci le proprietà seguenti:

ID gruppo	Chiave	Valore
<b>KafkaSource</b>	topic	AWSKafkaTutorialTopic
<b>KafkaSource</b>	bootstrap.servers	<i>L'elenco dei server di bootstrap salvato in precedenza</i>
<b>KafkaSource</b>	security.protocol	SSL

ID gruppo	Chiave	Valore
<b>KafkaSource</b>	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
<b>KafkaSource</b>	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
<b>KafkaSink</b>	topic	AWSKafkaTutorialTopicDestination
<b>KafkaSink</b>	bootstrap.servers	<i>L'elenco dei server di bootstrap salvato in precedenza</i>
<b>KafkaSink</b>	security.protocol	SSL
<b>KafkaSink</b>	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
<b>KafkaSink</b>	ssl.truststore.password	changeit
<b>KafkaSink</b>	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 [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, 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`

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.

## 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 AWSKafkaTutorialTopicDestination --from-  
beginning
```

Se non viene visualizzato alcun record nell'argomento di destinazione, consulta la sezione [Impossibile accedere alle risorse in un VPC](#) dell'argomento [Risoluzione dei problemi](#).

## Esempio: utilizzo di un consumatore EFO con un flusso di dati Kinesis

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 AWS Kinesis Connector (flink-connector-kinesis) 1.15.3.

### Note

Per impostare i prerequisiti richiesti per questo esercizio, completa innanzitutto l'esercizio [Guida introduttiva \(API\) DataStream](#).

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice dell'applicazione](#)
- [Compilazione del codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione 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 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** 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 univoco a livello globale aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

## Scrittura di 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.

## Download ed esame del 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 la sezione [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`. Nota 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");
```

## Compilazione del 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 [Prerequisiti](#) nel tutorial [Guida introduttiva \(API\) DataStream](#).
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 bucket `ka-app-code` - e scegli **Carica**. `<username>`
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 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. Apri la console del servizio gestito per Apache Flink all'indirizzo `https://console.aws.amazon.com/flink`
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 ottenere 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 della policy IAM

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

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (`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
<b>ConsumerConfigProperties</b>	<b>flink.stream.recorderpublisher</b>	<b>EFO</b>
<b>ConsumerConfigProperties</b>	<b>flink.stream.efo.consumername</b>	<b>basic-efo-flink-app</b>
<b>ConsumerConfigProperties</b>	<b>INPUT_STREAM</b>	<b>ExampleInputStream</b>
<b>ConsumerConfigProperties</b>	<b>flink.inputstream.initpos</b>	<b>LATEST</b>
<b>ConsumerConfigProperties</b>	<b>AWS_REGION</b>	<b>us-west-2</b>

- In Proprietà, scegli Crea gruppo.
- Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
<b>ProducerConfigProperties</b>	<b>OUTPUT_STREAM</b>	<b>ExampleOutputStream</b>
<b>ProducerConfigProperties</b>	<b>AWS_REGION</b>	<b>us-west-2</b>
<b>ProducerConfigProperties</b>	<b>AggregationEnabled</b>	<b>false</b>

- 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, 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`

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 le metriche del servizio gestito per Apache Flink sulla CloudWatch console per verificare che l'applicazione funzioni.

Puoi anche controllare nella console Kinesis Data Streams, nella scheda fan-out Enhanced del flusso di dati, il nome del tuo consumatore (`basic-efo-flink-app`).

## Eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS create nel tutorial Finestra efo.

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 <https://console.aws.amazon.com/flink>
2. nel pannello Managed Service for 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 Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello 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 del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.
2. <username>Scegli il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

## 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. 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/](https://console.aws.amazon.com/cloudwatch/).
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log MyApplication/aws/kinesis-analytics/.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

## Esempio: scrittura in Kinesis Data Firehose

In questo esercizio, viene creata un'applicazione del servizio gestito per Apache Flink con un flusso di dati Kinesis come origine e un flusso di distribuzione Kinesis Data 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 [Guida introduttiva \(API\) DataStream](#) .

Questa sezione contiene le fasi seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice Java di streaming di Apache Flink](#)
- [Compilazione del codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione 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 Kinesis Data 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 Kinesis Data 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 Kinesis Data Firehose. Assegna un nome al flusso Kinesis Data Firehose **ExampleDeliveryStream**. Quando crei il flusso Kinesis Data Firehose, crei anche la destinazione S3 del flusso Kinesis Data 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 univoco a livello globale aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

## Scrittura di 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.

## Download ed esame del codice Java di 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`. Nota 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 Kinesis Data Firehose per scrivere dati su un flusso Kinesis Data Firehose. Il seguente snippet crea il sink Kinesis Data Firehose:

```
private static KinesisFirehoseSink<String> createFirehoseSinkFromStaticConfig() {
    Properties sinkProperties = new Properties();
    sinkProperties.setProperty(AWS_REGION, region);

    return KinesisFirehoseSink.<String>builder()
        .setFirehoseClientProperties(sinkProperties)
        .setSerializationSchema(new SimpleStringSchema())
        .setDeliveryStreamName(outputDeliveryStreamName)
        .build();
}
```

## Compilazione del 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 [Prerequisiti](#) nel tutorial [Guida introduttiva \(API\) DataStream](#).
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 dell'applicazione del servizio gestito per Apache 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, le tue risorse AWS Identity and Access Management (IAM) e Amazon CloudWatch Logs vengono create automaticamente. Quando crei l'applicazione utilizzando la AWS CLI, devi creare queste risorse separatamente.

### Argomenti

- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Creazione ed esecuzione dell'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 <https://console.aws.amazon.com/flink>
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 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 della policy IAM

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

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (**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 Accedi 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, 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

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 MyApplication pagina, 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 MyApplication pagina, 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.



## Creazione ed esecuzione dell'applicazione (AWS CLI)

In questa sezione, viene utilizzata la AWS CLI per creare ed eseguire l'applicazione 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

`AKReadStreamWriteSinkStream`. 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 (ARN) (`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": "ReadStream",
      "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 for 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 > Crea 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: Rivedi. 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. Scegli Collega policy.
- c. Nella casella di ricerca, inserisci **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- d. Scegli la ReadSourceStreamWriteSinkStream policy AK e scegli Allega 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.

### Avvio dell'applicazione

In questa sezione, utilizzerai 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, utilizzerai 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 richiesta seguente per interrompere l'applicazione:

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

L'applicazione è ora interrotta.

## Aggiungi un'opzione di registrazione CloudWatch

Puoi usare il AWS CLI per aggiungere un flusso di CloudWatch log Amazon alla tua applicazione. Per informazioni sull'utilizzo di CloudWatch Logs con la tua applicazione, consulta [the section called "Configurazione della registrazione"](#).

## Aggiornamento del codice dell'applicazione

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

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione e chiama `UpdateApplication`, specificando lo stesso bucket 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 `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"
        }
      }
    }
  }
}
```

## Eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 Kinesis Data Firehose](#)
- [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 <https://console.aws.amazon.com/flink>
2. Nel pannello Managed Service for Apache Flink, scegliete. 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. Apri la console Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello Kinesis Data Streams, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e conferma l'eliminazione.

### Eliminazione del flusso Kinesis Data Firehose

1. Apri la console Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello Kinesis Data Firehose, scegli. ExampleDeliveryStream
3. Nella ExampleDeliveryStreampagina, scegli Elimina lo stream Kinesis Data Firehose e 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. <username>Scegli il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.
4. Se hai creato un bucket Amazon S3 per la destinazione del flusso Kinesis Data 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 Kinesis Data Firehose, eliminala.
7. Nella barra di navigazione, scegli Ruoli.

8. Scegli il ruolo `kinesis-analytics - -us-west-2. MyApplication`
9. Scegli Elimina ruolo e conferma l'eliminazione.
10. Se hai creato un nuovo ruolo per il flusso Kinesis Data Firehose, eliminalo.

### CloudWatch Elimina le tue risorse

1. Apri la CloudWatch console all'[indirizzo https://console.aws.amazon.com/cloudwatch/](https://console.aws.amazon.com/cloudwatch/).
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log `MyApplication/aws/kinesis-analytics/`.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

## Esempio: lettura da un flusso Kinesis in un account diverso.

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](#)
- [Installazione](#)
- [Creazione del flusso Kinesis di origine](#)
- [Creazione e aggiornamento dei ruoli e delle policy IAM](#)
- [Aggiorna lo script Python](#)
- [Aggiornamento dell'applicazione Java.](#)
- [Creazione, caricamento ed esecuzione dell'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 [Guida introduttiva \(API\) DataStream](#) prima di procedere.
- Sono necessari due account AWS 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'account



AWS che hai usato per il tutorial Nozioni di base. Utilizza un account AWS diverso per il flusso di origine.

## Installazione

Accedi ai tuoi due account AWS utilizzando profili denominati. Modifica le credenziali AWS 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 account AWS diversi utilizzando i profili.

Per ulteriori informazioni sull'utilizzo di profili denominati con la AWS CLI, consulta [Profili denominati](#) nella documentazione dell'AWS Command Line Interface.

## Creazione del 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. Nota 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
```

## Creazione e aggiornamento dei ruoli e delle policy IAM

Per consentire l'accesso agli oggetti su più account AWS, 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 policy dell'account di 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 annotane il 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 policy 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 gli ID account di esempio (`012345678901`) con l'ID account del 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"
    }
  ]
}
```

## Aggiorna lo 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");
}
}
```

## Creazione, caricamento ed esecuzione dell'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

### API di origine dati correnti

Se utilizzi le API di origine dati correnti, l'applicazione può sfruttare l'utilità `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](#).

## API legacy SourceFunction

Se si utilizzano le SourceFunction API legacy, l'applicazione utilizzerà schemi di serializzazione e deserializzazione personalizzati che sostituiscono il metodo di caricamento del open 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 Kafka 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 [Guida introduttiva \(API\) DataStream](#).

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 certificato AWS (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 EC2](#) o [Amazon EKS](#))
- Un cluster [Confluent Kafka](#) ospitato in AWS
- Un cluster Kafka locale accessibile tramite [AWS Direct Connect](#) o VPN

Questo tutorial contiene le sezioni seguenti:

- [Creazione di un VPC con un cluster Amazon MSK](#)

- [Creazione e applicazione di un truststore personalizzato al cluster](#)
- [Creazione del codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione dell'applicazione](#)
- [Configurazione dell'applicazione](#)
- [Esecuzione dell'applicazione](#)
- [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 `AWSKafkaTutorialTopicDestination`:

```
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. È possibile ottenere l'elenco dei server di bootstrap con il seguente comando (sostituirlo `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 regione AWS selezionata nel codice, nei comandi e nelle voci della console.

## Creazione e applicazione di un truststore personalizzato al 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 la sezione [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 [Guida introduttiva \(API\) DataStream](#).

**Note**

Se il bucket Amazon S3 è stato eliminato dal tutorial Nozioni di base, segui nuovamente il passaggio [the section called “Caricamento del codice Java di streaming di Apache Flink”](#).

1. Nella console Amazon S3, scegli il bucket ka-app-code - e scegli Carica. <username>
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 `https://console.aws.amazon.com/flink`
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 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 Accedi 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
<b>KafkaSource</b>	topic	AWSKafkaTutorialTopic
<b>KafkaSource</b>	bootstrap.servers	<i>L'elenco dei server di bootstrap salvato in precedenza</i>

ID gruppo	Chiave	Valore
<b>KafkaSource</b>	security.protocol	SSL
<b>KafkaSource</b>	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
<b>KafkaSource</b>	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
<b>KafkaSink</b>	topic	AWSKafkaTutorialTopicDestination
<b>KafkaSink</b>	bootstrap.servers	<i>L'elenco dei server di bootstrap salvato in precedenza</i>
<b>KafkaSink</b>	security.protocol	SSL
<b>KafkaSink</b>	ssl.truststore.location	/usr/lib/jvm/java-11-amazon-corretto/lib/security/cacerts
<b>KafkaSink</b>	ssl.truststore.password	changeit
<b>KafkaSink</b>	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 [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, 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`

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.

## 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 AWSKafkaTutorialTopicDestination --from-  
beginning
```

Se non viene visualizzato alcun record nell'argomento di destinazione, consulta la sezione [Impossibile accedere alle risorse in un VPC](#) dell'argomento [Risoluzione dei problemi](#).

## 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

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 [Nozioni di base \(Python\)](#).

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)

- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice dell'applicazione](#)
- [Compressione e caricamento del codice Python di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione 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 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 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 univoco a livello globale aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

## Scrittura di 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, inserisci il comando seguente:

```
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.

## Download ed esame del 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 la sezione [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`. Nota quanto segue riguardo al codice dell'applicazione:

- L'applicazione utilizza un'origine 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](#)

## Compressione e caricamento del codice Python 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. 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 bucket `ka-app-code` - e scegli Carica. `<username>`
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 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. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/flink>
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 ottenere 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 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 Accedi 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. Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
<b>consumer.config.0</b>	<b>input.stream.name</b>	<b>ExampleInputStream</b>
<b>consumer.config.0</b>	<b>aws.region</b>	<b>us-west-2</b>
<b>consumer.config.0</b>	<b>scan.stream.initpos</b>	<b>LATEST</b>

Selezionare Salva.

6. In Proprietà, scegli nuovamente Aggiungi gruppo.
7. Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
<b>producer.config.0</b>	<b>output.stream.name</b>	<b>ExampleOutputStream</b>
<b>producer.config.0</b>	<b>aws.region</b>	<b>us-west-2</b>
<b>producer.config.0</b>	<b>shard.count</b>	<b>1</b>


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 [Specificazione dei file di codice](#).



## 9. Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
<b>kinesis.analytics.flink.run.options</b>	<b>python</b>	<b>tumbling-windows.py</b>
<b>kinesis.analytics.flink.run.options</b>	<b>jarfile</b>	<b>flink-sql-connector-kinesis-1.15.2.jar</b>

- 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, 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

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 della policy IAM

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

- Apri la console IAM all'indirizzo <https://console.aws.amazon.com/iam/>.
- Seleziona Policy. Scegli la policy **kinesis-analytics-service-MyApplication-us-west-2** creata dalla console nella sezione precedente.
- Nella pagina Riepilogo, scegli Modifica policy. Seleziona la scheda JSON.

4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (**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 le metriche del servizio gestito per Apache Flink sulla CloudWatch console per verificare che l'applicazione funzioni.

## Eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 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 <https://console.aws.amazon.com/flink>
2. nel pannello Managed Service for 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 Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello 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 del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.
2. <username>Scegli il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

## 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. 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/](https://console.aws.amazon.com/cloudwatch/).
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log MyApplication/aws/kinesis-analytics/.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

## Esempio: creazione di una finestra scorrevole in Python

### Note

Per impostare i prerequisiti richiesti per questo esercizio, completa prima l'esercizio [Nozioni di base \(Python\)](#).

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice dell'applicazione](#)
- [Compressione e caricamento del codice Python di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione 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 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 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 univoco a livello globale aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

## Scrittura di 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, inserisci il comando seguente:

```
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.

## Download ed esame del 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 la sezione [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`. Nota quanto segue riguardo al codice dell'applicazione:

- L'applicazione utilizza un'origine 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](#)

## Compressione e caricamento del codice Python 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](#).

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 bucket `ka-app-code` - e scegli Carica. <username>
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 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. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/flink>
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 ottenere 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 **myapp.zip**.
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. In Proprietà, scegli Aggiungi gruppo.
5. Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
<b>consumer.config.0</b>	<b>input.stream.name</b>	<b>ExampleInputStream</b>
<b>consumer.config.0</b>	<b>aws.region</b>	<b>us-west-2</b>
<b>consumer.config.0</b>	<b>scan.stream.initpos</b>	<b>LATEST</b>

Selezionare Salva.

- In Proprietà, scegli nuovamente Aggiungi gruppo.
- Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
<b>producer.config.0</b>	<b>output.stream.name</b>	<b>ExampleOutputStream</b>
<b>producer.config.0</b>	<b>aws.region</b>	<b>us-west-2</b>
<b>producer.config.0</b>	<b>shard.count</b>	<b>1</b>

- 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 [Specificazione dei file di codice](#).
- Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
<b>kinesis.analytics.flink.run.options</b>	<b>python</b>	<b>sliding-windows.py</b>
<b>kinesis.analytics.flink.run.options</b>	<b>jarfile</b>	<b>flink-sql-connector-kinesis_1.15.2.jar</b>

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, 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

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 della policy IAM

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

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (**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 le metriche del servizio gestito per Apache Flink sulla CloudWatch console per verificare che l'applicazione funzioni.

## Eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 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 <https://console.aws.amazon.com/flink>
2. nel pannello Managed Service for 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 Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello 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 del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.
2. <username>Scegli il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

### 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. 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/](https://console.aws.amazon.com/cloudwatch/).
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log MyApplication/aws/kinesis-analytics/.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

## Esempio: invio di dati di streaming ad Amazon S3 in Python

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 [Nozioni di base \(Python\)](#).

Questo argomento contiene le sezioni seguenti:

- [Creazione delle risorse dipendenti](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice dell'applicazione](#)
- [Compressione e caricamento del codice Python di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)
- [Eliminazione 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:

- 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 a livello globale aggiungendo il tuo nome di accesso, ad esempio **ka-app-code-*<username>***.

## Scrittura di 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, inserisci il comando seguente:

```
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.

## Download ed esame del 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 la sezione [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`. Nota quanto segue riguardo al codice dell'applicazione:

- L'applicazione utilizza un'origine 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](#)

## Compressione e caricamento del codice Python 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. [Utilizzate l'applicazione di compressione preferita per comprimere i file e -1.15.2.jar.streaming-file-sink.py flink-sql-connector-kinesis](#) Assegna un nome all'archivio `myapp.zip`.
2. Nella console Amazon S3, scegli il bucket `ka-app-code` - e scegli Carica. `<username>`
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 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. Apri la console del servizio gestito per Apache Flink all'indirizzo `https://console.aws.amazon.com/flink`
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 ottenere 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 **myapp.zip**.
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. In Proprietà, scegli Aggiungi gruppo.
5. Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
<b>consumer.config.0</b>	<b>input.stream.name</b>	<b>ExampleInputStream</b>
<b>consumer.config.0</b>	<b>aws.region</b>	<b>us-west-2</b>
<b>consumer.config.0</b>	<b>scan.stream.initpos</b>	<b>LATEST</b>

Selezionare 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 [Specificazione dei file di codice](#).
7. Immetti i valori e le proprietà dell'applicazione seguenti:

ID gruppo	Chiave	Valore
<b>kinesis.analytics.flink.run.options</b>	<b>python</b>	<b>streaming-file-sink.py</b>
<b>kinesis.analytics.flink.run.options</b>	<b>jarfile</b>	<b>S3Sink/lib/flink-sql-connector-kinesis-1.15.2.jar</b>

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 [Specificazione dei 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
<b>sink.config.0</b>	<b>output.bucket.name</b>	<b><i>bucket-name</i></b>

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, 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

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 della policy IAM

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

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (**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 le metriche del servizio gestito per Apache Flink sulla CloudWatch console per verificare che l'applicazione funzioni.

## Eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 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 <https://console.aws.amazon.com/flink>

2. nel pannello Managed Service for Apache Flink, scegli. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e conferma l'eliminazione.

### Eliminazione del flusso di dati Kinesis

1. Apri la console Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello Kinesis Data Streams, scegli. ExampleInputStream
3. Nella ExampleInputStreampagina, scegli Elimina Kinesis Stream e 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 ka-app-codebucket -<username>.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

### 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. 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 di log MyApplication/aws/kinesis-analytics/.
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

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, creerete una semplice applicazione di streaming che utilizza Scala 3.2.0 e l'API Java di Flink. DataStream 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:

- [Download ed esame del 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](#)
- [Eliminazione delle risorse AWS](#)

## Download ed esame del 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 la sezione [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`.

Nota quanto segue riguardo al codice dell'applicazione:

- Questo file 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 che definisce le 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 source e sink per accedere a risorse esterne utilizzando un `StreamExecutionEnvironment` oggetto.
- L'applicazione crea connettori di origine e sink utilizzando proprietà di applicazioni dinamiche. Le proprietà dell'applicazione di runtime vengono lette 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 si procede al caricamento del 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 tuo nome utente, per renderlo globalmente univoco. Seleziona Avanti.
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 `https://console.aws.amazon.com/flink`
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 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 ottenere 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

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 Accedi 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. Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
<b>ConsumerConfigProperties</b>	<b>input.stream.name</b>	<b>ExampleInputStream</b>
<b>ConsumerConfigProperties</b>	<b>aws.region</b>	<b>us-west-2</b>
<b>ConsumerConfigProperties</b>	<b>flink.stream.initpos</b>	<b>LATEST</b>

Selezionare Salva.

6. In Proprietà, scegli nuovamente Aggiungi gruppo.
7. Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
<b>ProducerConfigProperties</b>	<b>output.stream.name</b>	<b>ExampleOutputStream</b>
<b>ProducerConfigProperties</b>	<b>aws.region</b>	<b>us-west-2</b>



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 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

## Modifica della policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere al bucket Amazon S3.

Per modificare la policy IAM per aggiungere le autorizzazioni del bucket S3

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (**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, viene utilizzata la AWS Command Line Interface per creare ed eseguire l'applicazione del servizio gestito per Apache Flink. Utilizza il comando `kinesisanalyticsv2` della AWS CLI 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 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 (ARN) (**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 identità 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: Rivedi. 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 attendibilità 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. Scegli Collega policy.
- c. Nella casella di ricerca, inserisci **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.

### Avvio dell'applicazione

In questa sezione, utilizzerai 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, utilizzerai 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 avviare l'applicazione:

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

L'applicazione è ora interrotta.



## Aggiungi un'opzione di registrazione CloudWatch

Puoi usare il AWS CLI per aggiungere un flusso di CloudWatch log Amazon alla tua applicazione. Per informazioni sull'utilizzo di CloudWatch Logs con la tua applicazione, consulta [Configurazione della registrazione delle applicazioni](#).

## 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": "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 è necessario aggiornare il codice dell'applicazione con una nuova versione del pacchetto di codice, si utilizza l'azione [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 disabilitazione del controllo delle versioni](#).

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione e chiama `UpdateApplication`, specificando lo stesso bucket Amazon S3 e lo stesso nome 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": "tumbling_window",  
  "CurrentApplicationVersionId": 1,  
  "ApplicationConfigurationUpdate": {  
    "ApplicationCodeConfigurationUpdate": {  
      "CodeContentUpdate": {  
        "S3ContentLocationUpdate": {  
          "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",  
          "FileKeyUpdate": "tumbling-window-scala-1.0.jar",  
          "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhypvDU"  
        }  
      }  
    }  
  }  
}
```

```
    }  
  }  
}
```

## Eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 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 <https://console.aws.amazon.com/flink>
2. nel pannello Managed Service for 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 Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello 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 del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.

2. <username>Scegli il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

### 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. 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/](https://console.aws.amazon.com/cloudwatch/).
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log MyApplication/aws/kinesis-analytics/.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

## Esempio: creazione di una finestra scorrevole in 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, 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, creerete una semplice applicazione di streaming che utilizza Scala 3.2.0 e l'API Java di Flink. DataStream 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:

- [Download ed esame del 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](#)
- [Eliminazione delle risorse AWS](#)

## Download ed esame del 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 la sezione [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`.

Nota quanto segue riguardo al codice dell'applicazione:

- Questo file 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 che definisce le 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 source e sink per accedere a risorse esterne utilizzando un `StreamExecutionEnvironment` oggetto.
- L'applicazione crea connettori di origine e sink utilizzando proprietà di applicazioni dinamiche. Le proprietà dell'applicazione di runtime vengono lette 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 si procede al caricamento del 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 tuo nome utente, per renderlo globalmente univoco. Seleziona Avanti.
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 <https://console.aws.amazon.com/flink>
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 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 ottenere 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

Per configurare l'applicazione, utilizza la procedura seguente.

Per configurare l'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 `sliding-window-scala-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. In Proprietà, scegli Aggiungi gruppo.
5. Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
<code>ConsumerConfigProperties</code>	<code>input.stream.name</code>	<code>ExampleInputStream</code>
<code>ConsumerConfigProperties</code>	<code>aws.region</code>	<code>us-west-2</code>

ID gruppo	Chiave	Valore
<b>ConsumerConfigProperties</b>	<b>flink.stream.initpos</b>	<b>LATEST</b>

Selezionare Salva.

- In Proprietà, scegli nuovamente Aggiungi gruppo.
- Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
<b>ProducerConfigProperties</b>	<b>output.stream.name</b>	<b>ExampleOutputStream</b>
<b>ProducerConfigProperties</b>	<b>aws.region</b>	<b>us-west-2</b>

- 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 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

## Modifica della policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere al bucket Amazon S3.

## Per modificare la policy IAM per aggiungere le autorizzazioni del bucket S3

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (**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, viene utilizzata la AWS Command Line Interface per creare ed eseguire l'applicazione del servizio gestito per Apache Flink. Utilizza il comando `kinesisanalyticsv2` della AWS CLI 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 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

`AKReadStreamWriteSinkStream`. 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 (ARN) (**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 identità 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: Rivedi. 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 attendibilità 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. Scegli Collega policy.
- c. Nella casella di ricerca, inserisci **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.

```
{
  "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.

### Avvio dell'applicazione

In questa sezione, utilizzerai 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, utilizzerai 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 avviare l'applicazione:

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

L'applicazione è ora interrotta.

## Aggiungi un'opzione di registrazione CloudWatch

Puoi usare il AWS CLI per aggiungere un flusso di CloudWatch log Amazon alla tua applicazione. Per informazioni sull'utilizzo di CloudWatch Logs con la tua applicazione, consulta [Configurazione della registrazione delle applicazioni](#).

## 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": "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 è necessario aggiornare il codice dell'applicazione con una nuova versione del pacchetto di codice, si utilizza l'azione [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 disabilitazione del controllo delle versioni](#).

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione e chiama `UpdateApplication`, specificando lo stesso bucket Amazon S3 e lo stesso nome 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": "sliding_window",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationCodeConfigurationUpdate": {
      "CodeContentUpdate": {
        "S3ContentLocationUpdate": {
          "BucketARNUpdate": "arn:aws:s3:::ka-app-code-username",
          "FileKeyUpdate": "-1.0.jar",
          "ObjectVersionUpdate": "SAMPLEUehYngP87ex1nzYIGYgfhyvpDU"
        }
      }
    }
  }
}
```

## Eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 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 `https://console.aws.amazon.com/flink`
2. nel pannello Managed Service for 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 Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello 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 del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.
2. <username>Scegli il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

## 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. 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 di log MyApplication/aws/kinesis-analytics/.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

## Esempio: invio di dati di streaming ad Amazon S3 in 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, 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, creerete una semplice applicazione di streaming che utilizza Scala 3.2.0 e l'API Java di Flink. DataStream 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 \(Scala\)](#). Devi solo creare una cartella aggiuntiva **data/** nel ka-app-code bucket Amazon S3 -. <username>

Questo argomento contiene le sezioni seguenti:

- [Download ed esame del 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](#)
- [Eliminazione delle risorse AWS](#)

## Download ed esame del 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 la sezione [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`.

Nota quanto segue riguardo al codice dell'applicazione:

- Questo file 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 che definisce le 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 `StreamingFileSink` per scrivere su 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 source e sink per accedere a risorse esterne utilizzando un `StreamExecutionEnvironment` oggetto.
- L'applicazione crea connettori di origine e sink utilizzando proprietà di applicazioni dinamiche. Le proprietà dell'applicazione di runtime vengono lette 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 si procede al caricamento del 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 tuo nome utente, per renderlo globalmente univoco. Seleziona Avanti.
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 <https://console.aws.amazon.com/flink>
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 **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 ottenere 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

Per configurare l'applicazione, utilizza la procedura seguente.

Per configurare l'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 `s3-sink-scala-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. In Proprietà, scegli Aggiungi gruppo.
5. Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
<code>ConsumerConfigProperties</code>	<code>input.stream.name</code>	<code>ExampleInputStream</code>
<code>ConsumerConfigProperties</code>	<code>aws.region</code>	<code>us-west-2</code>
<code>ConsumerConfigProperties</code>	<code>flink.stream.initpos</code>	<code>LATEST</code>

Selezionare Salva.

6. In Proprietà, scegli Aggiungi gruppo.
7. Inserisci i seguenti dati:

ID gruppo	Chiave	Valore
<b>ProducerConfigProperties</b>	<b>s3.sink.path</b>	<b>s3a://ka-app-code- &lt;user-name&gt; /data</b>

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 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

## Modifica della policy IAM

Modifica la policy IAM per aggiungere le autorizzazioni per accedere al bucket Amazon S3.

Per modificare la policy IAM per aggiungere le autorizzazioni del bucket S3

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (**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, viene utilizzata la AWS Command Line Interface per creare ed eseguire l'applicazione del servizio gestito per Apache Flink. Utilizza il comando `kinesisanalyticsv2` della AWS CLI 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 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

AKReadStreamWriteSinkStream. 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 (ARN) (**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 identità 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: Rivedi. 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 attendibilità 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. Scegli Collega policy.
- c. Nella casella di ricerca, inserisci **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.

```
{
  "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.

### Avvio dell'applicazione

In questa sezione, utilizzerai 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, utilizzerai 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 avviare l'applicazione:

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

L'applicazione è ora interrotta.

## Aggiungi un'opzione di registrazione CloudWatch

Puoi usare il AWS CLI per aggiungere un flusso di CloudWatch log Amazon alla tua applicazione. Per informazioni sull'utilizzo di CloudWatch Logs con la tua applicazione, consulta [Configurazione della registrazione delle applicazioni](#).

## 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": "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 è necessario aggiornare il codice dell'applicazione con una nuova versione del pacchetto di codice, si utilizza l'azione [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 disabilitazione del controllo delle versioni](#).

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione e chiama `UpdateApplication`, specificando lo stesso bucket Amazon S3 e lo stesso nome 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 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": "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"
        }
      }
    }
  }
}
```

## Eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 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 `https://console.aws.amazon.com/flink`
2. nel pannello Managed Service for 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 Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello 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 del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.
2. <username>Scegli il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

## 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. 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/](https://console.aws.amazon.com/cloudwatch/).
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log MyApplication/aws/kinesis-analytics/.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

# Utilizzo del servizio gestito da Amazon per Apache Flink

La sicurezza del cloud in AWS ha la massima priorità. In quanto cliente AWS, potrai 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:

- La sicurezza del cloud: AWS è responsabile della protezione dell'infrastruttura che gestisce i servizi AWS nel cloud AWS. AWS fornisce inoltre servizi che puoi utilizzare in sicurezza. L'efficacia della nostra sicurezza è regolarmente testata e verificata da revisori di terze parti come parte dei [programmi di conformità AWS](#). 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 responsabilità è determinata dal servizio AWS che viene utilizzato. 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](#)
- [Monitoraggio del servizio gestito 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 Kinesis Data Firehose e Amazon S3.

## Crittografia dei dati nel servizio gestito per Apache Flink

### Crittografia dei dati inattivi

Tieni presente quanto segue sulla crittografia dei dati inattivi 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 Kinesis Data 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 inattivi.
- 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 Kinesis Data 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 consente agli amministratori di controllare in modo sicuro l'accesso alle risorse AWS. 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 il cui uso non comporta costi aggiuntivi.

### 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 dell'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 con le credenziali di identità. Devi essere autenticato (connesso a AWS) come utente root Utente root dell'account AWS, come utente IAM o assumere un ruolo IAM.

Puoi accedere ad AWS come identità federata utilizzando le credenziali fornite attraverso un'origine di identità. Gli utenti AWS IAM Identity Center (Centro identità IAM), l'autenticazione Single Sign-On (SSO) dell'azienda e le credenziali di Google o Facebook sono esempi di identità federate. 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, puoi accedere alla AWS Management Console o al portale di accesso AWS. Per ulteriori informazioni sull'accesso ad AWS, consulta la sezione [Come accedere al tuo Account AWS](#) nella Guida per l'utente di Accedi ad AWS.

Se accedi ad AWS in modo programmatico, AWS fornisce un Software Development Kit (SDK) e un'interfaccia a riga di comando (CLI) per firmare crittograficamente le richieste utilizzando le tue credenziali. Se non utilizzi gli strumenti AWS, devi firmare le richieste personalmente. Per ulteriori informazioni sulla firma delle richieste, consulta [Firma delle richieste AWS](#) nella Guida per l'utente IAM.

A prescindere dal metodo di autenticazione utilizzato, potrebbe essere necessario specificare ulteriori informazioni sulla sicurezza. AWS consiglia ad esempio di utilizzare l'autenticazione a più fattori (MFA) per aumentare la sicurezza dell'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 \(MFA\) in AWS](#) nella Guida per l'utente di IAM.

## Utente root di un Account AWS

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 root Account AWS 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 utilizzarle 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 di IAM.

## Identità federata

Come best practice, richiedi 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 ai Servizi AWS utilizzando le credenziali fornite tramite un'origine di identità. Quando le identità federate accedono agli 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 e gli Account AWS. Per ulteriori informazioni sul Centro identità IAM, consulta [Cos'è Centro identità IAM?](#) nella Guida per l'utente di AWS IAM Identity Center.

## Utenti e gruppi IAM

Un [utente IAM](#) è una identità all'interno del tuo Account AWS che dispone di autorizzazioni specifiche per una singola persona 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, per 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 di 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 Amministratori IAM e 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 [Quando creare un utente IAM \(invece di un ruolo\)](#) nella Guida per l'utente di IAM.

## Ruoli IAM

Un [ruolo IAM](#) è un'identità all'interno di un Account AWS che dispone di autorizzazioni specifiche. È simile a un utente IAM, ma non è associato a una persona specifica. È possibile assumere temporaneamente un ruolo IAM nella AWS Management Console mediante lo [scambio di ruoli](#). È possibile assumere un ruolo chiamando un'azione AWS CLI o API AWS 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 di 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 [Creazione di un ruolo per un provider di identità di terza parte](#) nella Guida per l'utente di 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 ulteriori 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 proxy). Per

informazioni sulle differenze tra ruoli e policy basate su risorse per l'accesso multi-account, consulta [Differenza tra i ruoli IAM e le policy basate su risorse](#) nella Guida per l'utente di IAM.

- **Accesso multi-servizio:** 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 archivi 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.
- **Inoltro delle sessioni di accesso (FAS):** quando si utilizza un utente o un ruolo IAM per eseguire operazioni in AWS, tale utente o ruolo viene considerato un principale. Quando si utilizzano alcuni servizi, è possibile eseguire un'operazione che attiva un'altra azione in un servizio diverso. FAS utilizza le autorizzazioni del principale che effettua la chiamata a un Servizio AWS, combinate con il Servizio AWS richiedente, per effettuare richieste a servizi a valle. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che necessita di interazioni con altri Servizi AWS o risorse per essere portata a termine. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le operazioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina [Forward access sessions](#).
- **Ruolo di servizio:** un ruolo di servizio è un [ruolo IAM](#) assunto da un servizio per eseguire operazioni per conto dell'utente. Un amministratore IAM può creare, modificare ed eliminare un ruolo di servizio dall'interno di IAM. Per ulteriori informazioni, consulta la sezione [Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS](#) nella Guida per l'utente di 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'istanza EC2 che eseguono richieste di AWS CLI o dell'API AWS. Ciò è preferibile all'archiviazione delle chiavi di accesso nell'istanza EC2. Per assegnare un ruolo AWS a un'istanza EC2, affinché sia disponibile per tutte le relative applicazioni, puoi creare un profilo dell'istanza collegato all'istanza. Un profilo dell'istanza contiene il ruolo e consente ai programmi in esecuzione sull'istanza EC2 di ottenere le credenziali temporanee. Per ulteriori informazioni, consulta [Utilizzo di un ruolo IAM per concedere autorizzazioni ad applicazioni in esecuzione su istanze di Amazon EC2](#) nella Guida per l'utente di IAM.

Per informazioni sull'utilizzo dei ruoli IAM, consulta [Quando creare un ruolo IAM \(invece di un utente\)](#) nella Guida per l'utente di IAM.

## Gestione dell'accesso con policy

Per controllare l'accesso a AWS è possibile creare policy e collegarle a identità o risorse AWS. Una policy è un oggetto in AWS che, quando associato a un'identità o a una risorsa, ne definisce le autorizzazioni. AWS valuta queste policy quando un principale IAM (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 archiviata 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 di IAM.

Gli amministratori possono utilizzare le policy AWSJSON per specificare l'accesso ai diversi elementi. In altre parole, quale principale può eseguire azioni 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 azioni sulle risorse di cui hanno bisogno, un amministratore IAM può creare policy IAM. Successivamente l'amministratore può 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'azione `iam:GetRole`. Un utente con tale policy può ottenere informazioni sul ruolo dalla AWS Management Console, la AWS CLI o l'API AWS.

### Policy basate su 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 azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta [Creazione di policy IAM](#) nella Guida per l'utente di IAM.

Le policy basate su identità possono essere ulteriormente classificate come policy inline o policy gestite. Le policy inline sono incorporate direttamente in un singolo utente, gruppo o ruolo. Le policy gestite sono policy autonome che possono essere collegate a più utenti, gruppi e ruoli in Account AWS. Le policy gestite includono le policy gestite da AWS 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 di IAM.

## Policy basate su risorse

Le policy basate su risorse sono documenti di policy JSON che è possibile allegare a una risorsa. Gli esempi più comuni di policy basate su 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 utilizzarle per controllare l'accesso a una risorsa specifica. Quando è allegata a una risorsa, una policy definisce le azioni 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, ruoli, 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 gestite da AWS da IAM in una policy basata su risorse.

## Liste di controllo degli accessi (ACL)

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni per accedere a una risorsa. Le ACL sono simili alle policy basate sulle risorse, sebbene non utilizzino il formato del documento di policy JSON.

Amazon S3, AWS WAF e Amazon VPC sono esempi di servizi che supportano le ACL. Per maggiori informazioni sulle ACL, consulta [Panoramica delle liste di controllo degli accessi \(ACL\)](#) nella Guida per gli sviluppatori di Amazon Simple Storage Service.

## Altri tipi di policy

AWS supporta altri tipi di policy meno comuni. 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 funzione 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 di IAM.
- **Policy di controllo dei servizi (SCP):** le SCP 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 degli Account



AWSmultipli di proprietà dell'azienda. Se abiliti tutte le funzionalità in un'organizzazione, puoi applicare le policy di controllo dei servizi (SCP) a uno o tutti i tuoi account. La SCP limita le autorizzazioni per le entità negli account membri, compreso ogni Utente root dell'account AWS. Per ulteriori informazioni su organizzazioni e policy SCP, consulta la pagina sulle [Policy di controllo dei servizi](#) nella Guida per l'utente di AWS Organizations.

- **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 di 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, consulta [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
<a href="#">Policy basate su identità</a>	Sì
<a href="#">Policy basate su risorse</a>	No
<a href="#">Azioni di policy</a>	Sì
<a href="#">Risorse relative alle policy</a>	Sì
<a href="#">Chiavi di condizione delle policy</a>	No



Funzionalità IAM	Supporto del servizio gestito per Apache Flink
<a href="#">Liste di controllo degli accessi (ACL)</a>	No
<a href="#">ABAC (tag nelle policy)</a>	Sì
<a href="#">Credenziali temporanee</a>	Sì
<a href="#">Autorizzazioni del principale</a>	Sì
● <a href="#">Ruoli di servizio</a>	No
<a href="#">Ruoli collegati al servizio</a>	No

Per ottenere un quadro generale del funzionamento del servizio gestito per Apache Flink e di altri servizi AWS con la maggior parte delle funzionalità di IAM, consulta [Servizi AWS 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 azioni che utenti e ruoli possono eseguire, su quali risorse e in quali condizioni. Per informazioni su come creare una policy basata su identità, consulta [Creazione di policy IAM](#) nella Guida per l'utente di 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

Supporta le policy basate su risorse Sì

Le policy basate su risorse sono documenti di policy JSON che è possibile allegare a una risorsa. Gli esempi più comuni di policy basate su 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 utilizzarle per controllare l'accesso a una risorsa specifica. Quando è allegata a una risorsa, una policy definisce le azioni 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, ruoli, utenti federati o Servizi AWS.

Per consentire l'accesso multi-account, puoi specificare un intero account o entità IAM in un altro account come principale in una policy basata sulle risorse. L'aggiunta di un principale multi-account a una policy basata sulle risorse rappresenta solo una parte della relazione di trust. Quando l'entità principale e la risorsa si trovano in diversi Account AWS, un amministratore IAM nell'account attendibile deve concedere all'entità principale (utente o ruolo) anche l'autorizzazione per accedere alla risorsa. L'autorizzazione viene concessa collegando all'entità una policy basata sull'identità. Tuttavia, se una policy basata su risorse concede l'accesso a un principale nello stesso account, non sono richieste ulteriori policy basate su identità. Per ulteriori informazioni, consulta [Differenza tra i ruoli IAM e le policy basate su risorse](#) nella Guida per l'utente di IAM.

## Operazioni di policy per il servizio gestito per Apache Flink

Supporta le azioni di policy Sì

Gli amministratori possono utilizzare le policy JSON AWS per specificare gli accessi ai diversi elementi. Cioè, quale principale può eseguire azioni 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 azioni di policy hanno spesso lo stesso nome dell'operazione API AWS. Ci sono alcune eccezioni, ad esempio le azioni 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 operazioni tramite caratteri jolly (\*). Ad esempio, per specificare tutte le operazioni che iniziano con la parola Describe, includi la seguente operazione:

```
"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	Si
-------------------------------	----

Gli amministratori possono utilizzare le policy JSON AWS per specificare gli accessi ai diversi elementi. Cioè, 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'azione. Le istruzioni devono includere un elemento Resource o un elemento NotResource. Come best practice, specifica una risorsa utilizzando il suo [nome della risorsa Amazon \(ARN\)](#). Puoi eseguire questa operazione per azioni 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 ARN, consulta [Risorse definite dal servizio gestito da Amazon per 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 JSON AWS per specificare gli accessi ai diversi elementi. Cioè, quale principale può eseguire azioni 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. Puoi 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'operazione OR logica. Tutte le condizioni devono essere soddisfatte prima che le autorizzazioni dell'istruzione vengano concesse.

Puoi anche utilizzare variabili segnaposto quando specifichi le condizioni. Ad esempio, puoi 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 globali di AWS, consulta [Chiavi di contesto delle condizioni globali di AWS](#) 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 (ACL) nel servizio gestito da Amazon per Apache Flink

Supporta le ACL

No

Le liste di controllo degli accessi (ACL) controllano quali principali (membri, utenti o ruoli dell'account) hanno le autorizzazioni ad accedere a una risorsa. Le ACL sono simili alle policy basate su risorse, sebbene non utilizzino 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, tali attributi sono denominati tag. È possibile collegare dei tag alle entità IAM (utenti o ruoli) e a numerose risorse AWS. 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 [Che cos'è ABAC?](#) nella Guida per l'utente di 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 Servizi AWS che funzionano con le credenziali temporanee, consulta [Servizi AWS supportati da IAM](#) nella Guida per l'utente 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 ad 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 [Cambio di un ruolo \(console\)](#) nella Guida per l'utente di IAM.

È possibile creare manualmente credenziali temporanee utilizzando la AWS CLI o l'API AWS. È quindi possibile utilizzare tali credenziali temporanee per accedere ad AWS. 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 sessioni di accesso diretto (FAS)	Sì
--	----

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'azione che attiva un'altra azione in un servizio diverso. FAS utilizza le autorizzazioni del principale che effettua la chiamata a un Servizio AWS, combinate con il Servizio AWS richiedente, per effettuare richieste a servizi a valle. Le richieste FAS vengono effettuate solo quando un servizio riceve una richiesta che necessita di interazioni con altri Servizi AWS o risorse per essere portata a termine. In questo caso è necessario disporre delle autorizzazioni per eseguire entrambe le operazioni. Per i dettagli delle policy relative alle richieste FAS, consulta la pagina [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 [Creazione di un ruolo per delegare le autorizzazioni a un Servizio AWS](#) nella Guida per l'utente di 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 i ruoli collegati ai servizi	Sì
---------------------------------------	----

Un ruolo collegato ai servizi è un tipo di ruolo di servizio che è collegato a un Servizio AWS. Il servizio può assumere il ruolo per eseguire un'operazione 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 sono in grado di eseguire attività utilizzando la AWS Management Console, l'AWS Command Line Interface (AWS CLI) o l'API AWS. Per concedere agli utenti l'autorizzazione per 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](#) nella Guida per l'utente di IAM.

Per informazioni dettagliate sulle operazioni e sui tipi di risorse definiti dal servizio gestito per Apache Flink incluso il formato degli ARN per ogni tipo di risorsa, consulta [Operazioni, risorse e chiavi di condizione per il servizio gestito 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 operazioni 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 gestite da AWS 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 gestite da AWS 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 cliente di AWS 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 privilegi minimi: quando imposti le autorizzazioni con le policy IAM, concedi solo le autorizzazioni richieste per eseguire un'attività. Puoi 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 di IAM.
- Condizioni d'uso nelle policy IAM per limitare ulteriormente l'accesso: per limitare l'accesso a operazioni e risorse puoi 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. Puoi inoltre utilizzare le condizioni per concedere l'accesso alle operazioni di servizio, ma solo 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 di 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 IAM Access Analyzer](#) nella Guida per l'utente di IAM.
- Richiesta dell'autenticazione a più fattori (MFA): se hai uno scenario che richiede utenti IAM o utenti root nel tuo Account AWS, 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 [Configurazione dell'accesso alle API protetto con MFA](#) nella Guida per l'utente di 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 è necessario concedere le autorizzazioni minime della console agli utenti che effettuano chiamate solo alla AWS CLI o all'API AWS. 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 la policy `ConsoleAccess` o `ReadOnly` gestita da AWS del servizio gestito per Apache Flink. 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 allegate alla relativa identità utente. La policy include le autorizzazioni per completare questa azione sulla console o a livello di programmazione utilizzando la AWS CLI o l'API AWS.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ViewOwnUserInfo",
      "Effect": "Allow",
      "Action": [
        "iam:GetUserPolicy",
        "iam:ListGroupsWithUser",
        "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: PassRole](#)
- [Voglio consentire alle persone esterne al mio account AWS 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 hai l'autorizzazione a eseguire un'operazione, devi contattare l'amministratore per ricevere 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: 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 trasmettere 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 con l'accesso, contatta l'amministratore AWS. L'amministratore è la persona che ti ha fornito le credenziali di accesso.

Voglio consentire alle persone esterne al mio account AWS 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 (ACL), utilizza tali policy per concedere alle persone l'accesso alle tue risorse.

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, consulta [Fornire l'accesso a un utente IAM in un altro Account AWS in tuo possesso](#) nella Guida per l'utente IAM.
- Per informazioni su come fornire l'accesso alle risorse ad Account AWS di terze parti, consulta [Fornire l'accesso agli Account AWS di proprietà di terze parti](#) nella Guida per l'utente 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 di IAM.
- Per informazioni sulle differenze tra l'utilizzo di ruoli e policy basate su risorse per l'accesso multi-account, consulta [Differenza tra i ruoli IAM e le policy basate su risorse](#) 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 chiamante) 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 deputy", AWS fornisce strumenti per poter proteggere i dati per tutti i servizi con principali di 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 di Managed Service for Apache Flink, consigliamo di utilizzare le chiavi [aws: SourceArn](#) e [aws: SourceAccount](#) global condition context nella policy di trust dei ruoli 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. Registrati alla Console di gestione di AWS e apri 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"
        }
      }
    }
  ]
}
```

## Monitoraggio del servizio gestito per Apache Flink

Il servizio gestito per Apache Flink fornisce funzionalità di monitoraggio per le applicazioni. Per ulteriori informazioni, consulta [Registrazione e monitoraggio](#).

## 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 come parte di più programmi per la conformità di AWS. Sono inclusi SOC, PCI e HIPAA.

Per un elenco di servizi AWS 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 all'implementazione illustrano considerazioni relative all'architettura e forniscono fasi per l'implementazione 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.
- [Risorse per la conformità di AWS](#): questa raccolta di workbook e guide potrebbe essere utile al settore e alla posizione.
- [AWS Config](#): questo servizio AWS valuta il livello di conformità delle configurazioni delle risorse con pratiche interne, linee guida e regolamenti di settore.
- [AWS Security Hub](#): questo servizio AWS fornisce una visione completa dello stato di sicurezza all'interno di AWS che consente di verificare la conformità con gli standard e le best practice di sicurezza del settore.

## FedRAMP

Il programma di conformità agli standard FedRAMP di AWS 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.

È possibile richiedere l'accesso ai pacchetti di sicurezza FedRAMP di AWS tramite il FedRAMP PMO o tramite il Sales Account Manager AWS, oppure possono essere scaricati tramite AWS Artifact all'indirizzo [AWS Artifact](#).

Per ulteriori informazioni, consulta [FedRAMP](#).

## Resilienza nel servizio gestito da Amazon per Apache Flink

L'infrastruttura globale di AWS è basata su Regioni e zone di disponibilità AWS. Le Regioni AWS forniscono più zone di disponibilità fisicamente separate e isolate che sono connesse tramite reti altamente ridondanti, a bassa latenza e velocità effettiva elevata. 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 Regioni e zone di disponibilità AWS, consulta [Infrastruttura globale di AWS](#).

Oltre all'infrastruttura globale di AWS, 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 regione AWS 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 [Tolleranza agli errori](#).

## 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 globale di AWS descritte nel whitepaper [Amazon Web Services: panoramica dei processi di sicurezza](#).

Utilizza le chiamate API pubblicate di AWS 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 per Apache Flink fornisce una serie di caratteristiche 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 Kinesis Data Firehose o i bucket Amazon S3. Non è consigliabile archiviare le credenziali AWS 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 in risorse dipendenti

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 Kinesis Data Firehose e i bucket Amazon S3. Per ulteriori informazioni su come implementare la crittografia lato server nelle risorse dipendenti, consulta [Protezione dei dati](#)

## Utilizzato per monitorare CloudTrail le chiamate API

Il servizio gestito per Apache Flink è integrato con l'AWS CloudTrail, un servizio che offre una registrazione delle operazioni eseguite da un utente, un ruolo o un servizio Amazon nel servizio gestito per Apache Flink.

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, consulta [the section called “Uso di AWS CloudTrail”](#).

# Registrazione e monitoraggio nel servizio gestito da Amazon 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 sul monitoraggio da tutte le parti della soluzione AWS per consentire un debug più facile di eventuali guasti in più punti.

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

- [Registrazione](#)
- [Monitoraggio](#)
- [Configurazione della registrazione dell'applicazione](#)
- [Analisi dei log con Logs Insights CloudWatch](#)
- [Visualizzazione di metriche e dimensioni nel servizio gestito per Apache Flink](#)
- [Scrittura di messaggi personalizzati nei CloudWatch registri](#)
- [Registrazione delle chiamate API del servizio gestito per Apache Flink con AWS CloudTrail](#)

## Registrazione

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 il ritardo dell'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.

## Interrogazione dei 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

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 [Utilizzo degli 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 ridimensionata 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 la scalabilità reattiva, ovvero quando l'applicazione è già in ritardo.
- `CPUUtilization` e `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 falliscono 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.

## Configurazione della registrazione dell'applicazione

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.

Questo argomento contiene le sezioni seguenti:

- [Configurazione della CloudWatch registrazione tramite la console](#)
- [Configurazione della CloudWatch registrazione tramite la CLI](#)
- [Livelli di monitoraggio dell'applicazione](#)
- [Registrazione di best practice](#)
- [Risoluzione dei problemi relativi alla registrazione](#)
- [Fase successiva](#)

## Configurazione della CloudWatch registrazione tramite 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/KinesisAnalytics/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 “Livelli di monitoraggio dell'applicazione”](#).

## Configurazione della CloudWatch registrazione tramite la CLI

Per aggiungere un'opzione di CloudWatch registrazione utilizzando ilAWS CLI, procedi come segue:

- Create 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.

Questa sezione contiene i seguenti argomenti:

- [Creazione di un gruppo di CloudWatch log e di un flusso di log](#)
- [Utilizzo delle opzioni di CloudWatch registrazione delle applicazioni](#)
- [Aggiungere le autorizzazioni di scrittura nel flusso di log CloudWatch](#)



## Creazione di un gruppo di CloudWatch log e di un flusso di log

Puoi creare un gruppo di CloudWatch log e uno streaming 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](#).

### Utilizzo delle opzioni di CloudWatch registrazione delle applicazioni

Utilizzate 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 del servizio gestito per Apache Flink](#).

Aggiungere un'opzione di CloudWatch registro durante la creazione di un'applicazione

L'esempio seguente mostra come utilizzare l'CreateApplicationazione per aggiungere un'opzione di CloudWatch registro quando si crea un'applicazione. Nell'esempio, sostituisci *Amazon Resource Name (ARN) del flusso di CloudWatch log per aggiungerlo alla nuova applicazione* 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:::mybucket",
          "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'`AddApplicationCloudWatchLoggingOption` per aggiungere un'opzione di CloudWatch registro a un'applicazione esistente. Nell'esempio, sostituisci ogni *segnaposto dell'input utente* 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>
}
```

## Aggiornamento di un'opzione di CloudWatch registro esistente

L'esempio seguente mostra come utilizzare l'`UpdateApplication` per modificare un'opzione di CloudWatch registro esistente. Nell'esempio, sostituisci ogni *segnaposto dell'input utente* 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>
}
```

## Eliminazione di un'opzione di CloudWatch registro da un'applicazione

L'esempio seguente mostra come utilizzare

l'`DeleteApplicationCloudWatchLoggingOption` per eliminare un'opzione di

CloudWatch registro esistente. Nell'esempio, sostituisci ogni *segnaposto dell'input utente* 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>
}
```

### Impostazione del 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 "Livelli di monitoraggio dell'applicazione"](#).

### Impostazione del 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:::mybucket",
          "FileKey": "myflink.jar",
          "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
        }
      },
      "CodeContentType": "ZIPFILE"
    },
    "FlinkApplicationConfiguration":
```

```

    "MonitoringConfiguration": {
      "ConfigurationType": "CUSTOM",
      "LogLevel": "INFO"
    }
  },
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}

```

## Aggiornamento del 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"
      }
    }
  }
}

```

## Aggiungere le autorizzazioni di scrittura nel flusso di log CloudWatch

Il servizio gestito per Apache Flink richiede le autorizzazioni per scrivere errori di configurazione errata. CloudWatch Puoi aggiungere queste autorizzazioni al ruolo AWS Identity and Access Management (IAM) assunto dal servizio gestito per 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 di autorizzazione

Per concedere a un'applicazione le autorizzazioni per la scrittura di eventi di registro CloudWatch da una risorsa Managed Service for Apache Flink, puoi utilizzare la seguente politica di autorizzazioni IAM. Fornisci i nomi della risorsa Amazon (ARN) corretti per il gruppo di log e il flusso di log.

```

{
  "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:*"
      ]
    }
  ]
}

```

## Livelli di monitoraggio dell'applicazione

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 “Dimensionamento”](#).
- **Operatore:** i parametri si riferiscono a ciascun operatore. Per ulteriori informazioni sugli operatori, consulta [the section called “Operatori dell'API DataStream”](#).
- **Parallelismo:** i parametri si riferiscono al parallelismo delle applicazioni. Puoi impostare questo livello di metrica solo utilizzando il parametro dell' [MonitoringConfigurationUpdateAPI](#). [UpdateApplication](#) Non è possibile impostare questo livello dei parametri utilizzando la console. Per ulteriori informazioni sul parallelismo, consulta [the section called “Dimensionamento”](#).

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.

## Registrazione di best practice

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.

## Risoluzione dei problemi relativi alla 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 “Aggiungere le autorizzazioni di 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](#) e [DescribeApplication](#).
- Monitora CloudWatch le metriche, ad esempio `downtime` per diagnosticare altri problemi relativi alle applicazioni. Per informazioni sulla lettura delle CloudWatch metriche, consulta [Metriche e dimensioni nel servizio gestito per Apache Flink](#)

## Fase successiva

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 “Analisi dei log”](#).

## Analisi dei log con Logs Insights CloudWatch

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 Scoperti nel riquadro di destra. 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.

## Query di esempio

Questa sezione contiene le 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:

- [Analisi delle operazioni: distribuzione delle attività](#)
- [Analisi delle operazioni: modifica del parallelismo](#)
- [Analisi degli errori: accesso negato](#)
- [Analisi degli errori: origine o sink non trovati](#)
- [Analisi degli errori: errori relativi alle attività applicative](#)

### Analisi delle 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 [Dimensionamento](#).

```
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 "Dimensionamento automatico"](#).

```
fields @timestamp, @parallelism
| filter message like /property: parallelism.default, /
| parse message "default, *" as @parallelism
| sort @timestamp asc
```

## Analisi degli errori: accesso negato

La seguente query di Logs Insights CloudWatch restituisce i log. Access Denied

```
fields @timestamp, @message, @messageType
| filter applicationARN like /arn:aws:kinesisanalyticsus-
west-2:012345678901:application\/YourApplication/
| filter @message like /AccessDenied/
| sort @timestamp desc
```

## Analisi degli errori: origine o sink non trovati

La seguente query di CloudWatch Logs Insights restituisce i log. ResourceNotFound ResourceNotFoundi 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
```

## Analisi degli errori: errori relativi alle attività applicative

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
```

## Visualizzazione di metriche e dimensioni nel servizio gestito per Apache Flink

Questo argomento contiene le sezioni seguenti:

- [Parametri di applicazioni](#)
- [Parametri del connettore del flusso di dati Kinesis](#)
- [Parametri del connettore Amazon MSK](#)
- [Parametri di Apache Zeppelin](#)
- [Visualizzazione delle metriche CloudWatch](#)
- [Impostazione dei livelli di reporting delle metriche CloudWatch](#)
- [Utilizzo delle metriche personalizzate con il servizio gestito da Amazon per Apache Flink](#)
- [Utilizzo degli CloudWatch allarmi con Amazon Managed Service per 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 applicazioni

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
backPressureTimeMsPerSecond*	Millisecondi	Il tempo (in millisecondi) in cui questa attività o questo operatore vengono sottoposti a contropressione al secondo.	Attività, operatore, parallelismo	<p>*Disponibile solo per le applicazioni del servizio gestito per Apache Flink che eseguono la versione 1.13 di Flink.</p> <p>Questi parametri possono essere utili per identificare i rallentamenti in un'applicazione.</p>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
<code>busyTimeMsPerSecond*</code>	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	<p>*Disponibile solo per le applicazioni del servizio gestito per Apache Flink che eseguono la versione 1.13 di Flink.</p> <p>Questi parametri possono essere utili per identificare i rallentamenti in un'applicazione.</p>

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
container CPUUtilization	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 contenitori e Managed Service for Apache Flink pubblica 2* cinque esempi di questa metrica per intervallo di report di 1 minuto.	Applicazione	<p>Il calcolo per container viene svolto come segue:</p> <p>Tempo totale della CPU (in secondi) consumato dal container * 100/limite CPU del container (in CPU/secondi)</p> <p>La CPUUtilization metrica tiene conto solo dell'utilizzo della CPU del processo JVM in esecuzione all'interno del contenitore. TaskManager Esistono altri componenti in esecuzione e all'esterno della JVM nello stesso container. Il parametro</p>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
				<code>containerCPUUtilization</code> 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	<p>Il calcolo per container viene svolto come segue:</p> <p>Utilizzo della memoria del container (byte) * 100/ limite di memoria del container in base alle specifiche di distribuzione del pod (in byte)</p> <p><a href="#">Le ManagedMemoryUtilizations metriche HeapMemoryUtilization and tengono conto solo di parametri di memoria specifici come Heap Memory Usage of TaskManager o JVM o</a></p>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
				<p><a href="#">Managed Memory</a> (utilizzo della memoria al di fuori di JVM per processi nativi come <a href="#">RockSDB State Backend</a>). Il parametro <code>containerMemoryUtilization</code> 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. <code>OutOfMemoryError</code> TaskManager</p>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
containerDiskUtilization	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 contenuti in Managed Service for Apache Flink pubblica. 2*cinque esempi di questa metrica per intervallo di report di 1 minuto.	Applicazione	<p>Il calcolo per container viene svolto come segue:</p> <p>Utilizzo del disco in byte * 100/limite del disco per il container in byte</p> <p>Per i container, rappresenta l'utilizzo del filesystem su cui è impostato il volume root del container.</p>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
currentInputWatermark	Millisecondi	L'ultimo watermark ricevuto da questa applicazione/operatore/attività/thread	Applicazione, operatore, attività, parallelo	Questo record viene generato solo per dimensioni con due input. Questo è il valore minimo degli ultimi watermark ricevuti.
currentOutputWatermark	Millisecondi	L'ultimo watermark generato da questa applicazione/operatore/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 il fallimento 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. Gli HeapMemoryUtilization unici tengono conto 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	<p>*Disponibile solo per le applicazioni del servizio gestito per Apache Flink che eseguono la versione 1.13 di Flink.</p> <p>Questi parametri possono essere utili per identificare i rallentamenti in un'applicazione.</p>



Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
lastCheckpointSize	Byte	La dimensione totale dell'ultimo checkpoint	Applicazione	<p>È possibile utilizzare questo parametro per determinare l'utilizzo dello storage delle applicazioni in esecuzione.</p> <p>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.</p>	

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	<p>*Disponibile solo per le applicazioni del servizio gestito per Apache Flink che eseguono la versione 1.13 di Flink.</p> <p>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.</p>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
<code>managedMemoryTotal</code> *	Byte	La quantità totale di memoria gestita.	Applicazione, operatore, attività, parallelo	<p>*Disponibile solo per le applicazioni del servizio gestito per Apache Flink che eseguono la versione 1.13 di Flink.</p> <p>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</p>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
				di JVM per processi nativi come <a href="#">RocksDB State Backend</a> )
managedMemoryUtilization*	Percentuale	managedMemoryUsedDerivato da/managedMemoryTotal	Applicazione, operatore, attività, parallelo	<p>*Disponibile solo per le applicazioni del servizio gestito per Apache Flink che eseguono la versione 1.13 di Flink.</p> <p>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.</p>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
<code>numberOfFailedCheckpoints</code>	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 velocità effettiva 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	<p>*Per applicare la statistica Somma su un periodo di tempo (secondi/minuto):</p> <ul style="list-style-type: none"> <li>• Seleziona il parametro al livello corretto. Se stai monitorando il parametro per un operatore, devi selezionare i parametri dell'operatore corrispondenti.</li> <li>• Poiché il servizio gestito per Apache Flink esegue 4 snapshot metrici al minuto, è necessari o utilizzare</li> </ul>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
				<p>il seguente parametro matematico: <math>m1/4</math> dove <math>m1</math> è la statistica Somma su un periodo (secondi/minuto)</p> <p>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.</p>



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	<p>*Per applicare la statistica Somma su un periodo di tempo (secondi/minuto):</p> <ul style="list-style-type: none"> <li>• Seleziona il parametro al livello corretto. Se stai monitorando il parametro per un operatore, devi selezionare i parametri dell'operatore corrispondenti.</li> <li>• Poiché il servizio gestito per Apache Flink esegue 4 snapshot metrici al minuto, è necessario utilizzare</li> </ul>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
				<p>il seguente parametro matematico: <math>m1/4</math> dove <math>m1</math> è la statistica Somma su un periodo (secondi/minuto)</p> <p>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.</p>	

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	<p>*Per applicare la statistica Somma su un periodo di tempo (secondi/minuto):</p> <ul style="list-style-type: none"> <li>• Seleziona il parametro al livello corretto. Se stai monitorando il parametro per un operatore, devi selezionare i parametri dell'operatore corrispondenti.</li> <li>• Poiché il servizio gestito per Apache Flink esegue 4 snapshot metrici al minuto, è necessari o utilizzare</li> </ul>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
				<p>il seguente parametro matematico: <math>m1/4</math> dove <math>m1</math> è la statistica Somma su un periodo (secondi/minuto)</p> <p>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.</p>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
numLateRecordsDropped*	Conteggio	Applicazione, operatore, attività, paralleli smo		<p>*Per applicare la statistica Somma su un periodo di tempo (secondi/minuto):</p> <ul style="list-style-type: none"> <li>• Seleziona il parametro al livello corretto. Se stai monitorando il parametro per un operatore , devi selezionare i parametri dell'operatore corrispondenti.</li> <li>• Poiché il servizio gestito per Apache Flink esegue 4 snapshot metrici al minuto, è necessari o utilizzare</li> </ul>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo	
				<p>il seguente parametro matematico: <math>m1/4</math> dove <math>m1</math> è la statistica Somma su un periodo (secondi/minuto)</p> <p>Il numero di record di questo operatore o attività è diminuito a causa dell'arrivo in ritardo.</p>	

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	<p>*Per applicare la statistica Somma su un periodo di tempo (secondi/minuto):</p> <ul style="list-style-type: none"> <li>• Seleziona il parametro al livello corretto. Se stai monitorando il parametro per un operatore, devi selezionare i parametri dell'operatore corrispondenti.</li> <li>• Poiché il servizio gestito per Apache Flink esegue 4 snapshot metrici al minuto, è necessario utilizzare</li> </ul>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
				<p>il seguente parametro matematico: <math>m1/4</math> dove <math>m1</math> è la statistica Somma su un periodo (secondi/minuto)</p> <p>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.</p>



Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
<code>oldGenerationGCCount</code>	Conteggio	Il numero totale di vecchie operazioni di rimozione di oggetti inutili 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.	Applicazione	È possibile utilizzare questo parametro per monitorare la somma, la media e il tempo massimo di rimozione di oggetti inutili.
<code>threadCount</code>	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.

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
uptime	Millisecondi	Il tempo in cui il processo è stato eseguito senza interruzioni.	Applicazione	È possibile utilizzare e questo parametro per determinare se un processo viene eseguito correttamente. Questo parametro restituisce -1 per i processi completati.

## Parametri del connettore del flusso di dati Kinesis

AWS genera tutti i record per il flusso di dati Kinesis oltre ai seguenti:

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
millisbehindLatest	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) ShardId	<ul style="list-style-type: none"> <li>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 particola</li> </ul>

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
				<p>re partition e può essere specificato in base al nome del flusso e all'ID della partizione.</p> <ul style="list-style-type: none"> <li>Il valore -1 indica che il servizio non ha ancora riportato un valore per il parametro.</li> </ul>
bytesRequestedPerFetch	Byte	I byte richiesti in una singola chiamata a <code>getRecords</code> .	Applicazione (per Stream), Parallelismo (per) ShardId	

## Parametri del connettore Amazon MSK

AWS genera 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 specifica	Applicazione (per argomento), parallelismo (per) PartitionId	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
		to in base al nome dell'argomento e all'ID della partizione.		
<code>commitsFailed</code>	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, parallelo	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.
<code>commitsSuccessful</code>	N/D	Il numero totale di commit di offset riusciti verso Kafka, se il commit di offset e la creazione di checkpoint sono abilitati.	Applicazione, operatore, attività, parallelo	

Parametro	Unità	Descrizione	Livello	Note per l'utilizzo
<code>committed_offsets</code>	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	
<code>records_lag_max</code>	Conteggio	Il ritardo massimo in termini di numero di record per ogni partizione e in questa finestra	Applicazione, operatore, attività, parallelismo	
<code>bytes_consumed_rate</code>	Byte	Il numero medio di byte consumati al secondo per un argomento	Applicazione, operatore, attività, parallelismo	

## Parametri di Apache Zeppelin

Per i notebook Studio, AWS genera i seguenti parametri a livello di applicazione: `KPUs`, `cpuUtilization`, `heapMemoryUtilization`, `oldGenerationGCTime`, `oldGenerationGCCount` e `threadCount`. Inoltre, genera i parametri mostrati nella tabella seguente, anche a livello di applicazione.

Parametro	Unità	Descrizione	Nome Prometheus
zeppelinCpuUtilization	Percentuale	Percentuale complessiva di utilizzo della CPU nel server Apache Zeppelin.	process_cpu_usage
zeppelinHeapMemoryUtilization	Percentuale	Percentuale complessiva di utilizzo della memoria heap per il server Apache Zeppelin.	jvm_memory_used_bytes
zeppelinThreadCount	Conteggio	Il numero totale di thread live utilizzati dal server Apache Zeppelin.	jvm_threads_live_threads
zeppelinWaitingJobs	Conteggio	Il numero di processi di Apache Zeppelin in coda in attesa di un thread.	jetty_threads_jobs
zeppelinServerUptime	Secondi	Il tempo totale in cui il server è stato attivo e in funzione.	process_uptime_seconds

## Visualizzazione delle metriche CloudWatch

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 i parametri usando AWS CLI

- Al prompt dei comandi utilizza il comando seguente.

```
aws cloudwatch list-metrics --namespace "AWS/KinesisAnalytics" --region region
```

## Impostazione dei livelli di reporting delle metriche CloudWatch

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 console. 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.

Il livello di reporting CloudWatch delle metriche viene impostato 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](#).



# Utilizzo delle metriche personalizzate con il servizio gestito da Amazon 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](#)
- [Esempi](#)
- [Visualizzazione dei parametri personalizzati](#)

## 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.
- 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.

## Esempi

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`.

### Parametro personalizzato del numero di record

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](#).

### Parametro personalizzato del conteggio di 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](#).

## Visualizzazione dei parametri personalizzati

Le metriche personalizzate per l'applicazione vengono visualizzate nella console CloudWatch Metrics della AWS/KinesisAnalyticsdashboard, nel gruppo di metriche dell'applicazione.

## Utilizzo degli CloudWatch allarmi con Amazon Managed Service per Apache Flink

Utilizzando gli allarmi Amazon CloudWatch Metric, controlli 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](#).

### 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
tempo di inattività > 0	Average	0	A downtime greater than zero indicates that the application has failed. If the value is larger than 0, the application is not processing any data. Recommended for all applications. The Tempo di

---

Espressioni di parametro	Statistic	Threshold	Descrizione
			<p>inattività metric measures the duration of an outage. A downtime greater than zero indicates that the application has failed. For troubleshooting, see <a href="#">L'applicazione si riavvia</a>.</p>

Espressioni di parametro	Statistic	Threshold	Descrizione
TARIFFA (punti di controllo) <code>numberOfFailed &gt; 0</code>	Average	0	<p>This metric counts the number of failed checkpoints since the application started. Depending on the application, it can be tolerable if checkpoints fail occasionally. But if checkpoints are regularly failing, the application is likely unhealthy and needs further attention. We recommend monitoring <code>RATE(numberOfFailedCheckpoints)</code> to alarm on the gradient and not on absolute values. Recommended for all applications. Use this metric to monitor application health and checkpointing progress. The application saves state data to checkpoints when it's healthy. Checkpointing can fail due to timeouts if the application isn't making progress</p>

Espressioni di parametro	Statistic	Threshold	Descrizione
			in processing the input data. For troubleshooting, see <a href="#">Il checkpoint è in fase di interruzione.</a>
Operatore . numRecordsOutPerSecond < threshold	Average	The minimum number of records emitted from the application during normal conditions.	Recommended for all applications. Falling below this threshold can indicate that the application isn't making expected progress on the input data. For troubleshooting, see <a href="#">La velocità di trasmissione effettiva è troppo lenta.</a>



Espressioni di parametro	Statistic	Threshold	Descrizione
<code>records_lag_max   millisbehindLatest &gt; threshold</code>	Maximum	The maximum expected latency during normal conditions.	If the application is consuming from Kinesis or Kafka, these metrics indicate if the application is falling behind and needs to be scaled in order to keep up with the current load. This is a good generic metric that is easy to track for all kinds of applications. But it can only be used for reactive scaling, i.e., when the application has already fallen behind. Recommended for all applications. Use the <code>records_lag_max</code> metric for a Kafka source, or the <code>millisbehindLatest</code> for a Kinesis stream source. Rising above this threshold can indicate that the application isn't making expected progress on the input data. For troubleshooting, see <a href="#">La velocità</a>

---

Espressioni di parametro	Statistic	Threshold	Descrizione
			<a href="#"><u>di trasmissione effettiva è troppo lenta.</u></a>

Espressioni di parametro	Statistic	Threshold	Descrizione
<code>lastCheckpointDurationThreshold</code>	Maximum	The maximum expected checkpoint duration during normal conditions.	Monitors how much data is stored in state and how long it takes to take a checkpoint. If checkpoints grow or take long, the application is continuously spending time on checkpointing and has less cycles for actual processing. At some points, checkpoints may grow too large or take so long that they fail. In addition to monitoring absolute values, customers should also consider monitoring the change rate with <code>TARIFFA(lastCheckpointSize)</code> and <code>TARIFFA(lastCheckpointDuration)</code> . If the <code>lastCheckpointDuration</code> continuously increases, rising above this threshold can indicate that

---

Espressioni di parametro	Statistic	Threshold	Descrizione
			the application isn't making expected progress on the input data, or that there are problems with application health such as backpressure. For troubleshooting, see <a href="#">Crescita illimitata dello stato</a> .

Espressioni di parametro	Statistic	Threshold	Descrizione
<code>lastCheckpointSize &gt; threshold</code>	Maximum	The maximum expected checkpoint size during normal conditions.	Monitors how much data is stored in state and how long it takes to take a checkpoint. If checkpoints grow or take long, the application is continuously spending time on checkpointing and has less cycles for actual processing. At some points, checkpoints may grow too large or take so long that they fail. In addition to monitoring absolute values, customers should also consider monitoring the change rate with <code>TARIFFA(lastCheckpointSize)</code> and <code>TARIFFA(lastCheckpointDuration)</code> . If the <code>lastCheckpointSize</code> continuously increases, rising above this threshold can indicate that

Espressioni di parametro	Statistic	Threshold	Descrizione
			the application is accumulating state data. If the state data becomes too large, the application can run out of memory when recovering from a checkpoint, or recovering from a checkpoint might take too long. For troubleshooting, see <a href="#">Crescita illimitata dello stato</a> .
heapMemoryUtilization > threshold	Maximum	This gives a good indication of the overall resource utilization of the application and can be used for proactive scaling unless the application is I/O bound. The maximum expected heapMemoryUtilization size during normal conditions, with a recommended value of 90 percent.	You can use this metric to monitor the maximum memory utilization of task managers across the application. If the application reaches this threshold, you need to provision more resources . You do this by enabling automatic scaling or increasing the application parallelism. For more information about increasing resources , see <a href="#">Dimensionamento</a> .

Espressioni di parametro	Statistic	Threshold	Descrizione
<code>cpuUtilization &gt; threshold</code>	Maximum	This gives a good indication of the overall resource utilization of the application and can be used for proactive scaling unless the application is I/O bound. The maximum expected <code>cpuUtilization</code> size during normal conditions, with a recommended value of 80 percent.	You can use this metric to monitor the maximum CPU utilization of task managers across the application. If the application reaches this threshold, you need to provision more resources. You do this by enabling automatic scaling or increasing the application parallelism. For more information about increasing resources, see <a href="#">Dimensionamento</a> .
<code>threadsCount &gt; threshold</code>	Maximum	The maximum expected <code>threadsCount</code> size during normal conditions.	You can use this metric to watch for thread leaks in task managers across the application. If this metric reaches this threshold, check your application code for threads being created without being closed.

Espressioni di parametro	Statistic	Threshold	Descrizione
<code>(oldGarbageCollectionTime* 100) / 60_000</code> in un periodo di 1 minuto') > threshold	Maximum	The maximum expected oldGarbageCollectionTime duration. We recommend setting a threshold such that typical garbage collection time is 60 percent of the specified threshold , but the correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that there is a memory leak in task managers across the application.
TARIFFA <code>(oldGarbageCollectionConteggio) &gt; threshold</code>	Maximum	The maximum expected oldGarbageCollectionConteggio under normal conditions. The correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that there is a memory leak in task managers across the application.
Operatore <code>. currentOutputWatermark</code> - Operatore <code>. currentInputWatermark</code> > threshold	Minimum	The minimum expected watermark increment under normal conditions. The correct threshold for your application will vary.	If this metric is continually increasing, this can indicate that either the application is processing increasingly older events, or that an upstream subtask has not sent a watermark in an increasingly long time.



# Scrittura di messaggi personalizzati nei CloudWatch registri

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 SLF4J](#)

## 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 SLF4J

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"
}
```

## Registrazione delle chiamate API del servizio gestito per 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 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](#).

### Servizio gestito per Apache Flink Information 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. È possibile visualizzare, cercare e scaricare gli eventi recenti nell'account AWS. Per ulteriori informazioni, consulta [Visualizzazione degli eventi con CloudTrail la cronologia degli eventi](#).

Per una registrazione continua degli eventi nell'account AWS, inclusi gli eventi per il servizio gestito per Apache Flink, crea un trail. Un trail consente di CloudTrail inviare file di log a un bucket Amazon S3. Per impostazione predefinita, quando si crea un trail nella console, il trail sarà valido in tutte le regioni AWS. Il trail registra gli eventi di tutte le Regioni nella partizione AWS e distribuisce i file di log nel bucket Amazon S3 specificato. Inoltre, puoi configurare altri AWS servizi per analizzare ulteriormente e agire in base ai dati sugli eventi raccolti nei CloudTrail log. 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 servizio AWS.

Per ulteriori informazioni, consulta [Elemento CloudTrail userIdentity](#).

## Informazioni sulle voci del file di log del servizio gestito da per 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

[AddApplicationCloudWatchLoggingOptione](#) [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"
  }
]
}
```

# Ottimizzazione delle performance nel servizio gestito da Amazon per Apache Flink

Questo argomento descrive le tecniche per monitorare e migliorare le prestazioni dell'applicazione di servizio gestito per Apache Flink.

## Argomenti

- [Risoluzione dei problemi relativi alle prestazioni](#)
- [Best practice per ottimizzare le prestazioni](#)
- [Monitoraggio delle prestazioni](#)

## Risoluzione dei 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.

## 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:

- Origine e destinazione dei dati: è fondamentale assicurarsi che le risorse esterne con cui interagisce l'applicazione siano adeguatamente predisposte per la velocità di trasmissione 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

- [Livelli di monitoraggio CloudWatch](#)
- [Parametri di applicazione della CPU](#)
- [Parallelismo dell'applicazione](#)
- [Log dell'applicazione](#)
- [Attività, operatore, parallelismo](#)
- [Logica dell'applicazione](#)
- [Memoria dell'applicazione](#)

### Livelli di monitoraggio CloudWatch

È importante verificare che la configurazione dei livelli di monitoraggio di CloudWatch non sia eccessivamente 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 [Livelli di monitoraggio dell'applicazione](#).



## Parametri di applicazione della CPU

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 [Dimensionamento](#).

### 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 dell'applicazione

Consente di aumentare il parallelismo dell'applicazione. Il parametro `ParallelismConfigurationUpdate` dell'azione [UpdateApplication](#) permette di aggiornare il parallelismo dell'applicazione.

Per impostazione predefinita, il numero massimo di KPU per un'applicazione è 64, tuttavia è possibile ampliarlo 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 [Attività, operatore, parallelismo](#) quanto segue.

## Log 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 "Analisi dei log"](#).

## Attività, operatore, parallelismo

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.

## Best practice per ottimizzare le 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 gli argomenti seguenti:

- [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. I KPU dell'applicazione aumenteranno automaticamente se i seguenti criteri sono soddisfatti:

- il dimensionamento automatico è abilitato per l'applicazione;
- l'utilizzo della CPU rimane superiore al 75% per 15 minuti.

Se il dimensionamento automatico è abilitato, ma l'utilizzo della CPU non rimane entro questa soglia, l'applicazione non aumenterà le 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 [Dimensionamento](#).

## 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 [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 [Risorse dell'applicazione](#).

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](#)
- [Trasformazioni DataStream](#)

## Monitoraggio dell'utilizzo delle risorse di dipendenza esterne

Se si verifica un rallentamento delle prestazioni in una destinazione (come Kinesis Streams, Kinesis Data Firehose, DynamoDB o OpenSearch Service), l'applicazione subirà una congestione. 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. Nel caso in cui si verificano malfunzionamenti nell'applicazione, verifica che non siano presenti errori nei log di CloudWatch relativi ai servizi di destinazione.

## 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 il [Tutorial per la configurazione locale](#) nella [documentazione di Apache Flink](#).

## Monitoraggio delle prestazioni

Questa sezione descrive gli strumenti per il monitoraggio delle prestazioni di un'applicazione.

### Monitoraggio delle prestazioni con le metriche CloudWatch

L'utilizzo dei parametri di CloudWatch consente di monitorare l'utilizzo delle risorse, la velocità di trasmissione effettiva, i checkpoint e i tempi di inattività dell'applicazione. Per ulteriori informazioni sull'utilizzo dei parametri CloudWatch con l'applicazione di servizio gestito per Apache Flink, consulta [Metriche e dimensioni nel servizio gestito per Apache Flink](#).

### Monitoraggio delle prestazioni con i file di log e gli allarmi CloudWatch

È possibile monitorare le condizioni di errore che potrebbero causare problemi di prestazioni utilizzando i file di log di CloudWatch.

Le condizioni di errore vengono visualizzate nei log quando lo status del processo di Apache Flink cambia da RUNNING a FAILED.

Gli allarmi CloudWatch possono essere impiegati per creare notifiche per problemi relativi alle prestazioni, come l'uso delle risorse, i parametri dei checkpoint superiori a una soglia di sicurezza o i cambiamenti imprevisti dello stato delle applicazioni.

Per informazioni sulla creazione di allarmi CloudWatch per un'applicazione di servizio gestito per Apache Flink, consulta [Allarmi](#).

# Quota per il servizio gestito per Apache Flink e Studio Notebook

Quando lavori con un servizio gestito da Amazon per Apache Flink, tieni presente la quota seguente:

- Puoi creare fino a 50 servizi gestiti per Apache Flink per regione all'interno del 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 AWS Support](#).

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 il servizio gestito per Apache Flink, al tuo account AWS vengono addebitate le risorse allocate, anziché le risorse utilizzate dall'applicazione. Viene addebitata una tariffa oraria calcolata in base al numero massimo di KPU utilizzati per eseguire l'applicazione di elaborazione del flusso. 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.

- Puoi creare fino a 1.000 servizi gestiti per Apache Flink [Snapshot](#) per applicazione.
- 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 MB
- noteSize = 5 MiB
- noteCount = 1.000
- Max cumulative UDF size = 100 MB
- Max cumulative dependency jar size = 300 MB



# Manutenzione del servizio gestito per Apache Flink

Il servizio gestito per Apache Flink corregge periodicamente le applicazioni con aggiornamenti di sicurezza del sistema operativo e dell'immagine del container per mantenere la conformità e raggiungere gli obiettivi di sicurezza di AWS. La tabella seguente elenca la finestra temporale predefinita durante la quale il servizio gestito per Apache Flink esegue questo tipo di manutenzione. La manutenzione dell'applicazione può avvenire in qualsiasi momento durante la finestra temporale corrispondente alla regione. Durante il processo di manutenzione, l'applicazione potrebbe avere un periodo di inattività compreso tra 10 e 30 secondi. Tuttavia, la durata effettiva del periodo di inattività dipende dallo stato 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"](#).

[Per modificare la finestra temporale durante la quale il servizio gestito per Apache Flink esegue la manutenzione dell'applicazione, utilizza l'API UpdateApplicationMaintenanceConfiguration.](#)

Regione	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
Asia Pacifico (Hyderabad)	16:30 - 00:30 UTC
Asia Pacifico (Seul)	13:00 - 21:00 UTC
Asia Pacifico (Singapore)	14:00 - 22:00 UTC

Regione	Finestra temporale di manutenzione
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
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

Regione	Finestra temporale di manutenzione
Sud America (San Paolo)	19:00 - 03:00 UTC
Israele (Tel Aviv)	20:00 - 04:00 UTC

## 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 ID operatore espliciti e coerenti per gli operatori grafici del processo Flink. Se non è impostato in modo esplicito, Flink genera automaticamente un ID per gli operatori. Questo perché Flink utilizza questi ID operatore 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 gli ID degli operatori di un grafico del processo e quelli definiti in un savepoint. Ciò accade quando non sono impostati ID operatore coerenti espliciti e Flink genera automaticamente ID operatore 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 di Flink. [Per ulteriori informazioni, consulta l'argomento Impostare un UUID per tutti gli operatori operatori in Preparazione per la produzione](#).

# Preparazione per la produzione

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.

## Applicazioni di test del carico

Alcuni problemi con le applicazioni si manifestano solo in condizioni di carico elevato. Abbiamo riscontrato casi in cui le applicazioni sembravano funzionanti e 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 sink di dati 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 avviare l'applicazione e farle consumare dati dall'inizio del tempo, ad esempio, usando una posizione iniziale di `TRIM_HORIZON` nel caso di un flusso di dati Kinesis.

## 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  $\leq 128$
- $\text{MIN}(\text{nextPowerOfTwo}(\text{parallelism} + (\text{parallelism} / 2)), 2^{15})$ : se il parallelismo è  $> 128$

Se si ha intenzione di scalare l'applicazione con un parallelismo superiore a 128, occorre definire esplicitamente il parallelismo massimo.

Il parallelismo massimo può essere definito a livello di applicazione, con `env.setMaxParallelism(x)` o con un solo operatore. Salvo diversa indicazione, tutti gli operatori ereditano il parallelismo massimo dell'applicazione.

Per ulteriori informazioni, consulta l'argomento relativo all'[impostazione di un parallelismo massimo esplicito](#) nella documentazione di 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](#).

# Best practice per il servizio gestito per Apache Flink

Questa sezione contiene informazioni e consigli per lo sviluppo di servizi gestiti stabili e performanti per applicazioni Apache Flink.

## Argomenti

- [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 al plugin ServiceResourceTransformer Maven Shade](#)

## 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 lasciare abilitato il checkpoint per la tua applicazione. La creazione di checkpoint offre tolleranza agli errori per l'applicazione durante la manutenzione programmata, nonché in caso di guasti imprevisti dovuti a problemi di servizio, errori di dipendenza dall'applicazione e altri problemi. Per ulteriori informazioni sulla manutenzione, consulta [Maintenance \(Manutenzione\)](#).
- Imposta [ApplicationSnapshotConfiguration::](#) su `false` durante `SnapshotsEnabled` lo sviluppo o 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 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 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 [Metriche e dimensioni nel servizio gestito per Apache Flink](#).

Per ulteriori informazioni sull'implementazione della tolleranza di errore, consulta [Tolleranza agli errori](#).

## Versioni di connettori non supportate

Il servizio gestito per Apache Flink versione 1.15 impedirà automaticamente l'avvio o l'aggiornamento delle applicazioni che utilizzano versioni dei connettori Kinesis non supportate (includere nei JAR delle applicazioni). Quando esegui l'aggiornamento del servizio gestito per Apache Flink versione 1.15, 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 saranno supportate dal servizio gestito per Apache Flink in quanto potrebbero causare problemi di coerenza o errori grazie alla funzione Arresto con savepoint che impedisce le operazioni di interruzione/aggiornamento.

## Prestazioni e parallelismo

L'applicazione può essere dimensionata per soddisfare qualsiasi velocità di trasmissione effettiva 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 vengano limitati. Se le sorgenti 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 [Dimensionamento](#).

## 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'API utilizzando `DataStream.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 la scalabilità automatica, 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, dovresti considerare di passare il parallelismo dell'operatore come parametro di configurazione dell'applicazione, in modo da modificarlo senza modificare il codice, se hai bisogno, ad esempio, di eseguire il resharding del flusso di origine.



## Registrazione

È possibile monitorare le prestazioni e le condizioni di errore dell'applicazione utilizzando Logs. CloudWatch 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, consultare [Allarmi](#)

Per ulteriori informazioni sull'implementazione della registrazione, consulta [Registrazione e monitoraggio](#).

## 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 i privilegi a tale ruolo. L'applicazione Flink in esecuzione può quindi raccogliere credenziali temporanee con i rispettivi privilegi 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 segreti [AWS Secrets Manager](#).

Molti servizi AWS nativi supportano l'autenticazione:

- [Kinesis Data ProcessTaxiStream Streams — .java](#)
- Amazon MSK — <https://github.com/aws/aws-msk-iam-auth-using-the-amazon-msk/#-library-for-iam-authentication>
- [Amazon Elasticsearch Service — .java AmazonElasticsearchSink](#)
- Amazon S3: pronto all'uso nel servizio gestito per Apache Flink

## Lettura da fonti con pochi shard/partizioni

Durante la lettura da Apache Kafka o da Kinesis Data Stream, potrebbe esserci una discrepanza tra il parallelismo del flusso (ad esempio, il numero di partizioni per Kafka e il numero di shard 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 soli 2 shard e un'applicazione con un parallelismo di 8, solo due sottoattività consumano effettivamente il flusso e 6 sottoattività rimangono inattive. Ciò può limitare notevolmente la velocità effettiva 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 migliori prestazioni con `events-per-second` moltiplicati per `number-of-keys` davano un risultato inferiore 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.

Funziona, ma ci sono un paio di avvertenze da tenere a mente. I watermark vengono generati a livello di sottoattività e scorrono nel grafico dell'operatore.

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. Le sottoattività inattive quindi non bloccano più l'avanzamento dell'orario degli eventi negli operatori a valle.

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 mai.

## Riepilogo

- l'impostazione `withIdleness` non genererà nuovi watermark nel caso in cui una partizione sia inattiva, ma escluderà solo 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 mai (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 `TRIM_HORIZON` in cui tutte le partizioni esistenti vengono consumate in parallelo, ignorando la relazione padre/figlio)
- le impostazioni `withIdleness` della strategia watermark sembrano deprecare le impostazioni specifiche dell'origine Kinesis per le partizioni inattive (`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("XXXXXXXXXXXXXXXX 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-000000000015,HashKeyRange: {StartingHashKey:
319014718988379809496913694467282698240,EndingHashKey:
340282366920938463463374607431768211455},SequenceNumberRange: {StartingSequenceNumber:
49627894338681557796096402897798370703746460841949528306,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 2 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000014,HashKeyRange: {StartingHashKey:
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-000000000004,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-000000000003,HashKeyRange: {StartingHashKey:
63802943797675961899382738893456539648,EndingHashKey:
85070591730234615865843651857942052863},SequenceNumberRange: {StartingSequenceNumber:
49627894338413948853714035420099942084474680503877763122,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,532 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000001,HashKeyRange: {StartingHashKey:
21267647932558653966460912964485513216,EndingHashKey:
42535295865117307932921825928971026431},SequenceNumberRange: {StartingSequenceNumber:
49627894338369347363316974173816870647929383780865802258,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,532 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 0 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000009,HashKeyRange: {StartingHashKey:
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.internals.KinesisDataFetcher [] -
Subtask 7 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000000,HashKeyRange: {StartingHashKey: 0,EndingHashKey:
21267647932558653966460912964485513215},SequenceNumberRange: {StartingSequenceNumber:
49627894338347046618118443550675334929656735419359821826,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:21,568 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 1 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000002,HashKeyRange: {StartingHashKey:
42535295865117307932921825928971026432,EndingHashKey:
63802943797675961899382738893456539647},SequenceNumberRange: {StartingSequenceNumber:
49627894338391648108515504796958406366202032142371782690,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
11:59:21,568 INFO org.apache.flink.streaming.connectors.kinesis.FlinkKinesisConsumer
[] - Subtask 1 will be seeded with initial shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000013,HashKeyRange: {StartingHashKey:
276479423123262501563991868538311671808,EndingHashKey:
297747071055821155530452781502797185023},SequenceNumberRange: {StartingSequenceNumber:
49627894338636956305699341651515299267201164118937567442,}}'}, starting state set as
sequence number EARLIEST_SEQUENCE_NUM
```

```
11:59:21,568 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 1 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000002,HashKeyRange: {StartingHashKey:
42535295865117307932921825928971026432,EndingHashKey:
63802943797675961899382738893456539647},SequenceNumberRange: {StartingSequenceNumber:
49627894338391648108515504796958406366202032142371782690,}}'} from sequence number
EARLIEST_SEQUENCE_NUM with ShardConsumer 0
11:59:23,209 INFO
org.apache.flink.streaming.connectors.kinesis.internals.KinesisDataFetcher [] -
Subtask 0 will start consuming seeded shard StreamShardHandle{streamName='hp-16',
shard='{ShardId: shardId-000000000009,HashKeyRange: {StartingHashKey:
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.internals.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
XXXXXXXXXXXXXXXXX Window with 3 events; Watermark: 1648030809281, 2022-03-23T10:20:09.281Z
3
1
2
XXXXXXXXXXXXXXXXX Window with 2 events; Watermark: 1648030834104, 2022-03-23T10:20:34.104Z
4
5
XXXXXXXXXXXXXXXXX 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, con le strategie di watermark integrate, 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 2022-03-23T10:21:27.170Z ma per chiudere la finestra della sessione è necessario un watermark di 10 secondi e 1 millisecondo dopo.

Se l'opzione `withIdleness` viene rimossa dalla strategia di watermark, nessuna finestra di sessione si chiuderà mai, perché il “watermark globale” dell'operatore finestra non può avanzare.

Nota che 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 la situazione sono le diverse strategie di watermarking, che aggiungono un buffer di sicurezza (`forBoundedOutOfOrderness(Duration.ofSeconds(30))`) o consentono esplicitamente l'arrivo di eventi tardivi (`allowedLateness(Time.minutes(5))`).

## Impostare un UUID per tutti gli operatori

Quando il servizio gestito per 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 operatore. Flink si aspetta ID operatore espliciti e coerenti per gli operatori dei grafici di processo Flink. Se non è impostato in modo esplicito, Flink genera automaticamente un ID per gli operatori. Questo perché Flink utilizza questi ID operatore per identificare in modo univoco gli operatori in un grafico di processo e li utilizza per memorizzare lo stato di ciascun operatore in un savepoint.

Il problema della mancata corrispondenza degli ID operatore si verifica quando Flink non trova una mappatura 1:1 tra gli ID operatore di un grafico di processo e quelli definiti in un savepoint. Ciò accade quando non sono impostati ID operatore coerenti espliciti e Flink genera automaticamente ID operatore che potrebbero non essere coerenti con ogni creazione di grafici di processo. La probabilità che le applicazioni riscontrino questo problema è elevata durante le operazioni di manutenzione. Per evitare ciò, 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 al plugin ServiceResourceTransformer 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. Si consiglia di aggiungere il [ServiceResourceTransformer](#) plugin maven shade, definito nel file 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>
  </plugins>
</build>
```

# Apache Flink Stateful Functions

[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 forti 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 contenitore 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 il servizio gestito per 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>
```



E 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);
}
```

Nota 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.

# Informazioni sulle versioni precedenti del servizio gestito per Apache Flink

Questo argomento contiene informazioni sull'utilizzo del servizio gestito per Apache Flink con versioni precedenti di Apache Flink. Le versioni di Apache Flink supportate dal servizio gestito per Apache Flink sono 1.15.2 (consigliata), 1.13.2, 1.11.1, 1.8.2 e 1.6.2.

Ti consigliamo di utilizzare l'ultima versione supportata di Apache Flink con la tua applicazione del servizio gestito per Apache Flink. Apache Flink 1.15.2 presenta le seguenti funzionalità:

- Supporto per [API ed SQL delle tabelle di Apache Flink](#)
- Supporto per applicazioni Python.
- Supporto per Java versione 11 e qualsiasi versione di Scala
- Un modello di memoria migliorato
- Ottimizzazioni RockSDB per una maggiore stabilità delle applicazioni
- Supporto per lo strumento di gestione attività e le tracce di stack nel pannello di controllo di 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 di 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](#)
- [Nozioni di base: Flink 1.8.2](#)
- [Nozioni di base: Flink 1.6.2](#)

# 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 (aws-kinesisanalytics-runtime)	1.0.1
Connettori Flink del servizio gestito per Apache Flink (aws-kinesisanalytics-flink)	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 applicazioni](#).

## 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 (aws-kinesisanalytics-runtime)	1.0.1
Connettori Flink del servizio gestito per Apache Flink (aws-kinesisanalytics-flink)	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 applicazioni](#).

## Aggiornamento di applicazioni

Per aggiornare la versione di un'applicazione del servizio gestito per Apache Flink, è necessario aggiornare il codice dell'applicazione, eliminare l'applicazione precedente e creare una nuova applicazione con il codice aggiornato. Per farlo, esegui le operazioni seguenti:

- Cambia le versioni del runtime e dei connettori Flink del servizio gestito per Apache Flink (`aws-kinesisanalytics-flink`) nel file `pom.xml` dell'applicazione a 1.1.0.
- Rimuovi la proprietà `flink.version` dal file `pom.xml` dell'applicazione. Fornirai questo parametro quando compilerai il codice dell'applicazione nel passaggio successivo.
- Ricompila il codice dell'applicazione tramite il comando seguente:

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

- Elimina l'applicazione esistente. Crea nuovamente l'applicazione e scegli Apache Flink 1.15.2 (versione consigliata) per il runtime dell'applicazione.

### Note

Non è possibile utilizzare snapshot delle versioni precedenti dell'applicazione.

## 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 di Managed Service for 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 account AWS e creazione di un utente amministratore](#)
- [Fase successiva](#)
- [Fase 2: configurazione dell'AWS Command Line Interface \(AWS CLI\)](#)
- [Fase 3: creazione ed esecuzione di un'applicazione del servizio gestito per Apache Flink](#)
- [Fase 4: eliminazione delle risorse AWS](#)
- [Fase 5: passaggi successivi](#)

## 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 [Origini](#).
- **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 dell'API DataStream](#).
- **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 Kinesis Data Firehose, un bucket Amazon S3, ecc. Per ulteriori informazioni, consulta [Sink](#).

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 [Fase 1: impostazione di un account AWS e creazione di un utente amministratore](#).

## Fase 1: impostazione di un account AWS e creazione di un utente amministratore

### Registrarsi per creare un Account AWS

Se non disponi di un Account AWS, completa la procedura seguente per crearne uno.

Per registrarsi a un Account AWS

1. Apri la pagina all'indirizzo <https://portal.aws.amazon.com/billing/signup>.
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.

Durante la registrazione di un Account AWS, viene creato un Utente root dell'account AWS. L'utente root dispone dell'accesso a tutte le risorse e tutti i Servizi AWS nell'account. Come best



pratiche di sicurezza, [assegna l'accesso amministrativo a un utente amministrativo](#) e utilizza solo l'utente root per eseguire [attività che richiedono l'accesso di un utente root](#).

Al termine del processo di registrazione, riceverai un'e-mail di conferma da AWS. È possibile visualizzare l'attività corrente dell'account e gestire l'account in qualsiasi momento accedendo all'indirizzo <https://aws.amazon.com/> e selezionando Il mio account.

## Creazione di un utente amministratore

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

### Protezione dell'Utente root dell'account AWS

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

Per informazioni sull'accesso utilizzando un utente root, consulta la pagina [Accesso come utente root](#) 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 [Abilitazione di un dispositivo MFA virtuale per l'utente root dell'Account AWS \(console\)](#) nella Guida per l'utente IAM.

## Creazione di un utente amministratore

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 Centro identità IAM, assegna l'accesso amministrativo a un utente amministratore.

Per un tutorial sull'utilizzo di IAM Identity Center directory come origine di identità, consulta [Configure user access with the default IAM Identity Center directory](#) nella Guida per l'utente di AWS IAM Identity Center.

## 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 [Accedere al portale di accesso AWS](#) nella Guida per l'utente Accedi ad AWS.

## Concessione dell'accesso programmatico

Gli utenti hanno bisogno di un accesso programmatico se desiderano interagire con AWS esternamente a 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 richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	<p>Segui le istruzioni per l'interfaccia che desideri utilizzare.</p> <ul style="list-style-type: none"> <li>• Per la AWS CLI, consulta la pagina <a href="#">Configurazione della AWS CLI per l'uso di AWS IAM Identity Center</a> nella Guida per l'utente dell'AWS Command Line Interface.</li> <li>• Per gli SDK AWS, gli strumenti e le API AWS, consulta la pagina <a href="#">Autenticazione Centro identità IAM</a> nella Guida di riferimento per SDK e strumenti AWS.</li> </ul>
IAM	Utilizza credenziali temporane e per firmare richieste programmatiche alla AWS	Segui le istruzioni in <a href="#">Utilizzo di credenziali temporanee con le</a>

Quale utente necessita dell'accesso programmatico?	Per	Come
	CLI, agli SDK AWS o alle API AWS.	<a href="#">risorse AWS</a> nella Guida per l'utente IAM.
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	<p>Segui le istruzioni per l'interfaccia che desideri utilizzare.</p> <ul style="list-style-type: none"> <li>• Per la AWS CLI, consulta la pagina <a href="#">Autenticazione tramite credenziali utente IAM</a> nella Guida per l'utente dell'AWS Command Line Interface.</li> <li>• Per gli SDK e gli strumenti AWS, consulta la pagina <a href="#">Autenticazione con credenziali a lungo termine</a> nella Guida di riferimento per SDK e strumenti AWS.</li> <li>• Per le API AWS, consulta la pagina <a href="#">Gestione delle chiavi di accesso per utenti IAM</a> nella Guida per l'utente IAM.</li> </ul>

## Fase successiva

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

## Fase successiva

[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 AWS CLI è già 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 di AWS Command Line Interface. Per verificare la versione della AWS CLI, esegui il comando seguente:

```
aws --version
```

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

```
aws-cli/1.16.63
```

## Per configurare la AWS CLI

1. Scarica e configura la AWS CLI. Per 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 file AWS CLI della `config`. Puoi usare questo profilo quando esegui i comandi della AWS CLI. Per ulteriori informazioni sui profili denominati, consulta [Profili denominati](#) nella Guida per l'utente di 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 regioni AWS disponibili, consulta [Regioni ed endpoint](#) nel documento 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 accountAWS CLI, puoi provare l'esercizio successivo, in cui configurerai un'applicazione di esempio e verificherai la configurazione. end-to-end

## Fase successiva

### [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:

- [Creazione di due flussi di dati Amazon Kinesis](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice Java di streaming di Apache Flink](#)
- [Compilazione del codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)
- [Fase successiva](#)

## Creazione di 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 di 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 `create-stream` della AWS CLI di Amazon Kinesis.

```
$ 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
```

## Scrittura di 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
```

## Download ed esame del codice Java di streaming di Apache Flink

Il codice dell'applicazione Java per questo esempio è disponibile presso 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`.

Nota 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 [Proprietà di runtime](#).

## Compilazione del 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](#).


### Compilazione del 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 tuo nome utente, per renderlo globalmente univoco. Seleziona Avanti.
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 bucket ka-app-code - e scegli Carica. *<username>*
8. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `aws-kinesis-analytics-java-apps-1.0.jar` creato nella fase precedente. Seleziona Avanti.
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 del servizio gestito per Apache 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, le tue risorse AWS Identity and Access Management (IAM) e Amazon CloudWatch Logs vengono create automaticamente. Quando crei l'applicazione utilizzando la AWS CLI, devi creare queste risorse separatamente.

### Argomenti

- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Creazione ed esecuzione dell'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 <https://console.aws.amazon.com/flink>
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 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 ottenere 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 della policy IAM

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

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (`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
<b>ProducerConfigProperties</b>	<b>flink.inputstream.initpos</b>	<b>LATEST</b>
<b>ProducerConfigProperties</b>	<b>aws.region</b>	<b>us-west-2</b>
<b>ProducerConfigProperties</b>	<b>AggregationEnabled</b>	<b>false</b>

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 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

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.

## Creazione ed esecuzione dell'applicazione (AWS CLI)

In questa sezione, viene utilizzata la AWS CLI per creare ed eseguire l'applicazione del servizio gestito per Apache Flink. Il servizio gestito per Apache Flink utilizza il comando `kinesisanalyticsv2` della AWS CLI 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 (ARN) (*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 for 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 > Crea 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: Rivedi. 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. Scegli Collega policy.
- c. Nella casella di ricerca, inserisci **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- d. Scegli la ReadSourceStreamWriteSinkStream policy AK e scegli Allega 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, utilizzerai 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, utilizzerai 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 richiesta seguente per interrompere l'applicazione:

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

L'applicazione è ora interrotta.

### Aggiungi un'opzione di registrazione CloudWatch

Puoi usare il AWS CLI per aggiungere un flusso di CloudWatch log Amazon alla tua applicazione. Per informazioni sull'utilizzo di CloudWatch Logs con la tua applicazione, consulta [the section called "Configurazione della registrazione"](#).

## 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 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](#) della AWS 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 disabilitazione del controllo delle versioni](#).

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione e chiama `UpdateApplication`, specificando lo stesso bucket Amazon S3 e lo stesso nome 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 [the section called "Creazione di 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"
        }
      }
    }
  }
}
```

## Fase successiva

### [Fase 4: eliminazione delle risorse AWS](#)

## Fase 4: eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 bucket e degli oggetti Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)
- [Fase successiva](#)

### Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello Managed Service for Apache Flink, scegliete. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

### Eliminazione del flusso di dati Kinesis

1. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/flink>
2. Nel pannello 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 del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.

2. <username>Scegli il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

## 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. 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/](https://console.aws.amazon.com/cloudwatch/).
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log MyApplication/aws/kinesis-analytics/.
4. Quindi scegli Elimina gruppo di log e conferma l'eliminazione.

## Fase successiva

### [Fase 5: passaggi successivi](#)

## Fase 5: passaggi successivi

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 streaming di dati AWS per Amazon Kinesis](#): la soluzione di streaming di dati AWS per Amazon Kinesis configura automaticamente i servizi AWS necessari per acquisire, archiviare, elaborare e distribuire dati di streaming con facilità. 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 di taxi di New York. La soluzione configura tutte le AWS risorse necessarie, come i ruoli e le politiche IAM, una dashboard e gli allarmi. CloudWatch CloudWatch

- [Soluzione di streaming di dati AWS per Amazon MSK](#): la soluzione di streaming di dati AWS per Amazon MSK fornisce modelli AWS CloudFormation 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 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 società 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.
- [Servizio gestito per Apache Flink: esempi](#): 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.

#### 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 in Flink Managed Service per Apache Flink.

## Nozioni di base: Flink 1.11.1

Questo argomento contiene una versione del tutorial [Guida introduttiva \(API\) DataStream](#) 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 account AWS 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](#)
- [Fase 4: eliminazione delle risorse AWS](#)
- [Fase 5: passaggi successivi](#)

## 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 [Origini](#).
- **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 dell'API DataStream](#).
- **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 Kinesis Data Firehose, un bucket Amazon S3, ecc. Per ulteriori informazioni, consulta [Sink](#).

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 [Fase 1: impostazione di un account AWS e creazione di un utente amministratore](#).

## Fase 1: impostazione di un account AWS e creazione di un utente amministratore

### Registrarsi per creare un Account AWS

Se non disponi di un Account AWS, completa la procedura seguente per crearne uno.

Per registrarsi a un Account AWS

1. Apri la pagina all'indirizzo <https://portal.aws.amazon.com/billing/signup>.
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.

Durante la registrazione di un Account AWS, viene creato un Utente root dell'account AWS. 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 amministrativo](#) e utilizza solo l'utente root per eseguire [attività che richiedono l'accesso di un utente root](#).

Al termine del processo di registrazione, riceverai un'e-mail di conferma da AWS. È possibile visualizzare l'attività corrente dell'account e gestire l'account in qualsiasi momento accedendo all'indirizzo <https://aws.amazon.com/> e selezionando Il mio account.

## Creazione di un utente amministratore

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

### Protezione dell'Utente root dell'account AWS

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

Per informazioni sull'accesso utilizzando un utente root, consulta la pagina [Accesso come utente root](#) 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 [Abilitazione di un dispositivo MFA virtuale per l'utente root dell'Account AWS \(console\)](#) nella Guida per l'utente IAM.

### Creazione di un utente amministratore

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 Centro identità IAM, assegna l'accesso amministrativo a un utente amministratore.

Per un tutorial sull'utilizzo di IAM Identity Center directory come origine di identità, consulta [Configure user access with the default IAM Identity Center directory](#) nella Guida per l'utente di AWS IAM Identity Center.

### 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 [Accedere al portale di accesso AWS](#) nella Guida per l'utente Accedi ad AWS.

## Concessione dell'accesso programmatico

Gli utenti hanno bisogno di un accesso programmatico se desiderano interagire con AWS esternamente a 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 richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	<p>Segui le istruzioni per l'interfaccia che desideri utilizzare.</p> <ul style="list-style-type: none"> <li>• Per la AWS CLI, consulta la pagina <a href="#">Configurazione della AWS CLI per l'uso di AWS IAM Identity Center</a> nella Guida per l'utente dell'AWS Command Line Interface.</li> <li>• Per gli SDK AWS, gli strumenti e le API AWS, consulta la pagina <a href="#">Autenticazione Centro identità IAM</a> nella Guida di riferimento per SDK e strumenti AWS.</li> </ul>
IAM	Utilizza credenziali temporane e per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	Segui le istruzioni in <a href="#">Utilizzo di credenziali temporanee con le risorse AWS</a> 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 alla AWS CLI, agli SDK AWS o alle API AWS.	Segui le istruzioni per l'interfaccia che desideri utilizzare. <ul style="list-style-type: none"><li>• Per la AWS CLI, consulta la pagina <a href="#">Autenticazione tramite credenziali utente IAM</a> nella Guida per l'utente dell'AWS Command Line Interface.</li><li>• Per gli SDK e gli strumenti AWS, consulta la pagina <a href="#">Autenticazione con credenziali a lungo termine</a> nella Guida di riferimento per SDK e strumenti AWS.</li><li>• Per le API AWS, consulta la pagina <a href="#">Gestione delle chiavi di accesso per utenti IAM</a> nella Guida per l'utente IAM.</li></ul>

## Fase successiva

### [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 AWS CLI è già 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 di AWS Command Line Interface. Per verificare la versione della AWS CLI, esegui il comando seguente:

```
aws --version
```

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

```
aws-cli/1.16.63
```

Per configurare la AWS CLI

1. Scarica e configura la AWS CLI. Per 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 file AWS CLI della config. Puoi usare questo profilo quando esegui i comandi della AWS CLI. Per ulteriori informazioni sui profili denominati, consulta [Profili denominati](#) nella Guida per l'utente di 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 regioni AWS disponibili, consulta [Regioni ed endpoint](#) nel documento 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 accountAWS CLI, puoi provare l'esercizio successivo, in cui configurerai un'applicazione di esempio e verificherai la configurazione. end-to-end

## Fase successiva

### [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:

- [Creazione di due flussi di dati Amazon Kinesis](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice Java di streaming di Apache Flink](#)
- [Compilazione del codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)
- [Fase successiva](#)

## Creazione di 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 di 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 `create-stream` della AWS CLI di Amazon Kinesis.

```
$ 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
```

## Scrittura di 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
```

## Download ed esame del codice Java di streaming di Apache Flink

Il codice dell'applicazione Java per questo esempio è disponibile presso 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`.

Nota 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 [Proprietà di runtime](#).

## Compilazione del 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](#).

### Compilazione del 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 tuo nome utente, per renderlo globalmente univoco. Seleziona Avanti.
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 bucket ka-app-code - e scegli Carica. <username>
8. Nella fase Seleziona file, scegli Aggiungi file. Individua il file aws-kinesis-analytics-java-apps-1.0.jar creato nella fase precedente. Seleziona Avanti.
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 del servizio gestito per Apache 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, le tue risorse AWS Identity and Access Management (IAM) e Amazon CloudWatch Logs vengono create automaticamente. Quando crei l'applicazione utilizzando la AWS CLI, devi creare queste risorse separatamente.

### Argomenti

- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Creazione ed esecuzione dell'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 <https://console.aws.amazon.com/flink>
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 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 ottenere 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 della policy IAM

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

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (`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 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
<b>ProducerConfigProperties</b>	<b>flink.inputstream.initpos</b>	<b>LATEST</b>
<b>ProducerConfigProperties</b>	<b>aws.region</b>	<b>us-west-2</b>
<b>ProducerConfigProperties</b>	<b>AggregationEnabled</b>	<b>false</b>

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 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

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.

### Creazione ed esecuzione dell'applicazione (AWS CLI)

In questa sezione, viene utilizzata la AWS CLI per creare ed eseguire l'applicazione del servizio gestito per Apache Flink. Un servizio gestito per Apache Flink utilizza il comando `kinesisanalyticsv2` della AWS CLI 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

`AKReadStreamWriteSinkStream`. 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 (ARN) (`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": "ReadStream",
      "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 for 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 > Crea 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: Rivedi**. 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. Scegli **Collega policy**.
- c. Nella casella di ricerca, inserisci **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- d. Scegli la **ReadSourceStreamWriteSinkStream** policy AK e scegli **Allega 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 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, utilizzerai 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, utilizzerai 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 richiesta seguente per interrompere l'applicazione:

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

L'applicazione è ora interrotta.

## Aggiungi un'opzione di registrazione CloudWatch

Puoi usare il AWS CLI per aggiungere un flusso di CloudWatch log Amazon alla tua applicazione. Per informazioni sull'utilizzo di CloudWatch Logs con la tua applicazione, consulta [the section called "Configurazione della registrazione"](#).

## 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 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](#) della AWS 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 disabilitazione del controllo delle versioni](#).

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione e chiama `UpdateApplication`, specificando lo stesso bucket Amazon S3 e lo stesso nome 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 [the section called "Creazione di 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": "SAMPLEUehYngP87ex1nzYIGYgfhyvDU"
    }
}
}
```

## Fase successiva

### [Fase 4: eliminazione delle risorse AWS](#)

## Fase 4: eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 bucket e degli oggetti Amazon S3](#)
- [Eliminazione delle risorse IAM](#)
- [CloudWatch Elimina le tue risorse](#)
- [Fase successiva](#)

## Eliminazione dell'applicazione del servizio gestito per Apache Flink

1. Apri la console Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello Managed Service for Apache Flink, scegliete. MyApplication
3. Nella pagina dell'applicazione, scegli Elimina e quindi conferma l'eliminazione.

## Eliminazione del flusso di dati Kinesis

1. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/flink>



2. Nel pannello 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 del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.
2. <username>Scegli il bucket `ka-app-code-`.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

## 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. 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/](https://console.aws.amazon.com/cloudwatch/).
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log `MyApplication/aws/kinesis-analytics/`.
4. Quindi scegli Elimina gruppo di log e conferma l'eliminazione.

## Fase successiva

### [Fase 5: passaggi successivi](#)

## Fase 5: passaggi successivi

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 streaming di dati AWS per Amazon Kinesis](#): la soluzione di streaming di dati AWS per Amazon Kinesis configura automaticamente i servizi AWS necessari per acquisire, archiviare, elaborare e distribuire dati di streaming con facilità. 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 di taxi di New York. La soluzione configura tutte le AWS risorse necessarie, come i ruoli e le politiche IAM, una dashboard e gli allarmi. CloudWatch CloudWatch
- [Soluzione di streaming di dati AWS per Amazon MSK](#): la soluzione di streaming di dati AWS per Amazon MSK fornisce modelli AWS CloudFormation 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 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 società 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.
- [Servizio gestito per Apache Flink: esempi](#): 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.

### 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 in Flink Managed Service per Apache Flink.

- Esempi di [codice Apache Flink: un GitHub archivio di un'ampia varietà di esempi](#) di applicazioni Apache Flink.

## Nozioni di base: Flink 1.8.2

Questo argomento contiene una versione del tutorial [Guida introduttiva \(API\) DataStream](#) che utilizza Apache Flink 1.8.2.

### Argomenti

- [Componenti dell'applicazione del servizio gestito di Apache Flink](#)
- [Prerequisiti per il completamento degli esercizi](#)
- [Fase 1: impostazione di un account AWS 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](#)
- [Fase 4: eliminazione delle risorse AWS](#)

## Componenti dell'applicazione del servizio gestito di 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 [Origini](#).
- 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 dell'API DataStream](#).
- 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 Kinesis Data Firehose, un bucket Amazon S3, ecc. Per ulteriori informazioni, consulta [Sink](#).

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, vedi [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 account AWS e creazione di un utente amministratore](#).

## Fase 1: impostazione di un account AWS e creazione di un utente amministratore

### Registrarsi per creare un Account AWS

Se non disponi di un Account AWS, completa la procedura seguente per crearne uno.

Per registrarsi a un Account AWS

1. Apri la pagina all'indirizzo <https://portal.aws.amazon.com/billing/signup>.

## 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.

Durante la registrazione di un Account AWS, viene creato un Utente root dell'account AWS. 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 amministrativo](#) e utilizza solo l'utente root per eseguire [attività che richiedono l'accesso di un utente root](#).

Al termine del processo di registrazione, riceverai un'e-mail di conferma da AWS. È possibile visualizzare l'attività corrente dell'account e gestire l'account in qualsiasi momento accedendo all'indirizzo <https://aws.amazon.com/> e selezionando Il mio account.

## Creazione di un utente amministratore

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

### Protezione dell'Utente root dell'account AWS

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

Per informazioni sull'accesso utilizzando un utente root, consulta la pagina [Accesso come utente root](#) 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 [Abilitazione di un dispositivo MFA virtuale per l'utente root dell'Account AWS \(console\)](#) nella Guida per l'utente IAM.

## Creazione di un utente amministratore

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 Centro identità IAM, assegna l'accesso amministrativo a un utente amministratore.

Per un tutorial sull'utilizzo di IAM Identity Center directory come origine di identità, consulta [Configure user access with the default IAM Identity Center directory](#) nella Guida per l'utente di AWS IAM Identity Center.

### 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 [Accedere al portale di accesso AWS](#) nella Guida per l'utente Accedi ad AWS.

### Concessione dell'accesso programmatico

Gli utenti hanno bisogno di un accesso programmatico se desiderano interagire con AWS esternamente a 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 richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	<p>Segui le istruzioni per l'interfaccia che desideri utilizzare.</p> <ul style="list-style-type: none"> <li>• Per la AWS CLI, consulta la pagina <a href="#">Configurazione della AWS CLI per l'uso di AWS IAM Identity Center</a> nella Guida per l'utente dell'AWS Command Line Interface.</li> <li>• Per gli SDK AWS, gli strumenti e le API AWS, consulta la pagina <a href="#">Autenticazione Centro identità IAM</a></li> </ul>

Quale utente necessita dell'accesso programmatico?	Per	Come
		nella Guida di riferimento per SDK e strumenti AWS.
IAM	Utilizza credenziali temporanee e per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	Segui le istruzioni in <a href="#">Utilizzo di credenziali temporanee con le risorse AWS</a> nella Guida per l'utente IAM.
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	<p>Segui le istruzioni per l'interfaccia che desideri utilizzare.</p> <ul style="list-style-type: none"> <li>• Per la AWS CLI, consulta la pagina <a href="#">Autenticazione tramite credenziali utente IAM</a> nella Guida per l'utente dell'AWS Command Line Interface.</li> <li>• Per gli SDK e gli strumenti AWS, consulta la pagina <a href="#">Autenticazione con credenziali a lungo termine</a> nella Guida di riferimento per SDK e strumenti AWS.</li> <li>• Per le API AWS, consulta la pagina <a href="#">Gestione delle chiavi di accesso per utenti IAM</a> nella Guida per l'utente IAM.</li> </ul>

## 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 AWS CLI è già 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 di AWS Command Line Interface. Per verificare la versione della AWS CLI, esegui il comando seguente:

```
aws --version
```

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

```
aws-cli/1.16.63
```

## Per configurare la AWS CLI

1. Scarica e configura la AWS CLI. Per 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 file AWS CLI della `config`. Puoi usare questo profilo quando esegui i comandi della AWS CLI. Per ulteriori informazioni sui profili denominati, consulta [Profili denominati](#) nella Guida per l'utente di 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 regione AWS 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 accountAWS CLI, è possibile provare l'esercizio successivo, in cui si configura un'applicazione di esempio e si verifica la end-to-end configurazione.

## Fase successiva

### [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:

- [Creazione di due flussi di dati Amazon Kinesis](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice Java di streaming di Apache Flink](#)
- [Compilazione del codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)
- [Fase successiva](#)

## Creazione di 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 di 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 `create-stream` della AWS CLI di Amazon Kinesis.

```
$ 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
```

## Scrittura di 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
```

## Download ed esame del codice Java di streaming di Apache Flink

Il codice dell'applicazione Java per questo esempio è disponibile presso 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`.

Nota 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 [Proprietà di runtime](#).

## Compilazione del 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 tuo nome utente, per renderlo globalmente univoco. Seleziona Avanti.
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 bucket ka-app-code - e scegli Carica. *<username>*
8. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `aws-kinesis-analytics-java-apps-1.0.jar` creato nella fase precedente. Seleziona Avanti.
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 del servizio gestito per Apache 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, le tue risorse AWS Identity and Access Management (IAM) e Amazon CloudWatch Logs vengono create automaticamente. Quando crei l'applicazione utilizzando la AWS CLI, devi creare queste risorse separatamente.

### Argomenti

- [Creazione ed esecuzione dell'applicazione \(console\)](#)
- [Creazione ed esecuzione dell'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 <https://console.aws.amazon.com/flink>
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 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 ottenere 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 della policy IAM

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

1. Apri 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. Seleziona la scheda JSON.

4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (**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. Immetti i valori e le proprietà dell'applicazione seguenti:

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

- 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 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

- Nella MyApplicationpagina, scegli Esegui. Conferma l'operazione.
- 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 MyApplication pagina, scegli Configura. Aggiorna le impostazioni dell'applicazione e scegli Aggiorna.

## Creazione ed esecuzione dell'applicazione (AWS CLI)

In questa sezione, viene utilizzata la AWS CLI per creare ed eseguire l'applicazione del servizio gestito per Apache Flink. Il servizio gestito per Apache Flink utilizza il comando `kinesisanalyticsv2` della AWS CLI 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

`AKReadStreamWriteSinkStream`. 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 (ARN) (*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 for 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 > Crea 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: Rivedi. 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. Scegli Collega policy.
- c. Nella casella di ricerca, inserisci **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- d. Scegli la ReadSourceStreamWriteSinkStream policy AK e scegli Allega 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.

### Avvio dell'applicazione

In questa sezione, utilizzerai 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, utilizzerai 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 richiesta seguente per interrompere l'applicazione:

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

L'applicazione è ora interrotta.

## Aggiungi un'opzione di registrazione CloudWatch

Puoi usare il AWS CLI per aggiungere un flusso di CloudWatch log Amazon alla tua applicazione. Per informazioni sull'utilizzo di CloudWatch Logs con la tua applicazione, consulta [the section called "Configurazione della registrazione"](#).

## 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 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](#) della AWS 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 disabilitazione del controllo delle versioni](#).

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione e chiama `UpdateApplication`, specificando lo stesso bucket Amazon S3 e lo stesso nome 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 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 di 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": "SAMPLEUehYngP87ex1nzYIGYgfhyvDU"
        }
      }
    }
  }
}
```

## Fase successiva

### [Fase 4: eliminazione delle risorse AWS](#)

## Fase 4: eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 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 Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello Managed Service for Apache Flink, scegliete. 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. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/flink>
2. Nel pannello 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 del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.
2. <username>Scegli il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

## 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. 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/](https://console.aws.amazon.com/cloudwatch/).
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log MyApplication/aws/kinesis-analytics/.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

## Nozioni di base: Flink 1.6.2

Questo argomento contiene una versione del tutorial [Guida introduttiva \(API\) DataStream](#) che utilizza Apache Flink 1.6.2.

### Argomenti

- [Componenti di un servizio gestito per Apache Flink](#)
- [Prerequisiti per il completamento degli esercizi](#)
- [Fase 1: impostazione di un account AWS 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](#)
- [Fase 4: eliminazione delle risorse AWS](#)

## Componenti di un 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 [Origini](#).
- **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 dell'API DataStream](#).

- 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 Kinesis Data Firehose, un bucket Amazon S3, ecc. Per ulteriori informazioni, consulta [Sink](#).

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 account AWS e creazione di un utente amministratore](#).

## Fase 1: impostazione di un account AWS e creazione di un utente amministratore

### Registrarsi per creare un Account AWS

Se non disponi di un Account AWS, completa la procedura seguente per crearne uno.

Per registrarsi a un Account AWS

1. Apri la pagina all'indirizzo <https://portal.aws.amazon.com/billing/signup>.

## 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.

Durante la registrazione di un Account AWS, viene creato un Utente root dell'account AWS. 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 amministrativo](#) e utilizza solo l'utente root per eseguire [attività che richiedono l'accesso di un utente root](#).

Al termine del processo di registrazione, riceverai un'e-mail di conferma da AWS. È possibile visualizzare l'attività corrente dell'account e gestire l'account in qualsiasi momento accedendo all'indirizzo <https://aws.amazon.com/> e selezionando Il mio account.

## Creazione di un utente amministratore

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

### Protezione dell'Utente root dell'account AWS

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

Per informazioni sull'accesso utilizzando un utente root, consulta la pagina [Accesso come utente root](#) 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 [Abilitazione di un dispositivo MFA virtuale per l'utente root dell'Account AWS \(console\)](#) nella Guida per l'utente IAM.

## Creazione di un utente amministratore

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 Centro identità IAM, assegna l'accesso amministrativo a un utente amministratore.

Per un tutorial sull'utilizzo di IAM Identity Center directory come origine di identità, consulta [Configure user access with the default IAM Identity Center directory](#) nella Guida per l'utente di AWS IAM Identity Center.

### 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 [Accedere al portale di accesso AWS](#) nella Guida per l'utente Accedi ad AWS.

### Concessione dell'accesso programmatico

Gli utenti hanno bisogno di un accesso programmatico se desiderano interagire con AWS esternamente a 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 richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	<p>Segui le istruzioni per l'interfaccia che desideri utilizzare.</p> <ul style="list-style-type: none"> <li>• Per la AWS CLI, consulta la pagina <a href="#">Configurazione della AWS CLI per l'uso di AWS IAM Identity Center</a> nella Guida per l'utente dell'AWS Command Line Interface.</li> <li>• Per gli SDK AWS, gli strumenti e le API AWS, consulta la pagina <a href="#">Autenticazione Centro identità IAM</a></li> </ul>

Quale utente necessita dell'accesso programmatico?	Per	Come
		nella Guida di riferimento per SDK e strumenti AWS.
IAM	Utilizza credenziali temporanee e per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	Segui le istruzioni in <a href="#">Utilizzo di credenziali temporanee con le risorse AWS</a> nella Guida per l'utente IAM.
IAM	(Non consigliato) Utilizza credenziali a lungo termine per firmare richieste programmatiche alla AWS CLI, agli SDK AWS o alle API AWS.	<p>Segui le istruzioni per l'interfaccia che desideri utilizzare.</p> <ul style="list-style-type: none"> <li>• Per la AWS CLI, consulta la pagina <a href="#">Autenticazione tramite credenziali utente IAM</a> nella Guida per l'utente dell'AWS Command Line Interface.</li> <li>• Per gli SDK e gli strumenti AWS, consulta la pagina <a href="#">Autenticazione con credenziali a lungo termine</a> nella Guida di riferimento per SDK e strumenti AWS.</li> <li>• Per le API AWS, consulta la pagina <a href="#">Gestione delle chiavi di accesso per utenti IAM</a> nella Guida per l'utente IAM.</li> </ul>

## 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 AWS CLI è già 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 di AWS Command Line Interface. Per verificare la versione della AWS CLI, esegui il comando seguente:

```
aws --version
```

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

```
aws-cli/1.16.63
```

## Per configurare la AWS CLI

1. Scarica e configura la AWS CLI. Per 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 file AWS CLI della `config`. Puoi usare questo profilo quando esegui i comandi della AWS CLI. Per ulteriori informazioni sui profili denominati, consulta [Profili denominati](#) nella Guida per l'utente di 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 regioni AWS disponibili, consulta [Regioni ed endpoint](#) nel documento 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 accountAWS CLI, è possibile provare l'esercizio successivo, in cui si configura un'applicazione di esempio e si verifica la end-to-end configurazione.

## Fase successiva

### [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:

- [Creazione di due flussi di dati Amazon Kinesis](#)
- [Scrittura di record di esempio nel flusso di input](#)
- [Download ed esame del codice Java di streaming di Apache Flink](#)
- [Compilazione del codice dell'applicazione](#)
- [Caricamento del codice Java di streaming di Apache Flink](#)
- [Creazione ed esecuzione dell'applicazione del servizio gestito per Apache Flink](#)

## Creazione di 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 di 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 `create-stream` della AWS CLI di Amazon Kinesis.

```
$ 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
```

## Scrittura di 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
```

## Download ed esame del codice Java di streaming di Apache Flink

Il codice dell'applicazione Java per questo esempio è disponibile presso 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`.

Nota 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 [Proprietà di runtime](#).

## Compilazione del 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 "Specificare la versione di Apache Flink dell'applicazione"](#).

- 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 tuo nome utente, per renderlo globalmente univoco. Seleziona Avanti.
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 bucket ka-app-code - e scegli Carica. *<username>*
8. Nella fase Seleziona file, scegli Aggiungi file. Individua il file `aws-kinesis-analytics-java-apps-1.0.jar` creato nella fase precedente. Seleziona Avanti.
9. Nella fase Imposta autorizzazioni, non modificare le impostazioni. Seleziona Avanti.
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 dell'applicazione del servizio gestito per Apache 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, le tue risorse AWS Identity and Access Management (IAM) e Amazon CloudWatch Logs vengono create automaticamente. Quando crei l'applicazione utilizzando la AWS CLI, devi creare queste risorse separatamente.

### Argomenti

- [Creazione ed esecuzione dell'applicazione \(console\)](#)

- [Creazione ed esecuzione dell'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 <https://console.aws.amazon.com/flink>
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 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 ottenere 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 della policy IAM

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

1. Apri 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. Seleziona la scheda JSON.
4. Aggiungi alla policy la sezione evidenziata del seguente esempio di policy. Sostituisci gli ID account di esempio (`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 Accedi 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
<b>ProducerConfigProperties</b>	<b>flink.inputstream.initpos</b>	<b>LATEST</b>
<b>ProducerConfigProperties</b>	<b>aws.region</b>	<b>us-west-2</b>
<b>ProducerConfigProperties</b>	<b>AggregationEnabled</b>	<b>false</b>

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 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

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.

## Creazione ed esecuzione dell'applicazione (AWS CLI)

In questa sezione, viene utilizzata la AWS CLI per creare ed eseguire l'applicazione del servizio gestito per Apache Flink. Il servizio gestito per Apache Flink utilizza il comando `kinesisanalyticsv2` della AWS CLI 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

`AKReadStreamWriteSinkStream`. 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 (ARN) (*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 for 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 > Crea 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: Rivedi. 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. Scegli Collega policy.
- c. Nella casella di ricerca, inserisci **AKReadSourceStreamWriteSinkStream** (la policy creata nella sezione precedente).
- d. Scegli la ReadSourceStreamWriteSinkStream policy AK e scegli Allega 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 kinesisanalyticsv2 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, utilizzerai 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, utilizzerai 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 richiesta seguente per interrompere l'applicazione:

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

L'applicazione è ora interrotta.

## Aggiungi un'opzione di registrazione CloudWatch

Puoi usare il AWS CLI per aggiungere un flusso di CloudWatch log Amazon alla tua applicazione. Per informazioni sull'utilizzo di CloudWatch Logs con la tua applicazione, consulta [the section called "Configurazione della registrazione"](#).

## 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'operazione [UpdateApplication](#) AWS CLI.

Per utilizzare la AWS CLI, elimina il pacchetto di codice precedente dal bucket Amazon S3, carica la nuova versione e chiama `UpdateApplication`, specificando lo stesso bucket 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 "Creazione di 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"
      }
    }
  }
}
```

## Fase 4: eliminazione delle risorse AWS

Questa sezione include le procedure per l'eliminazione delle risorse AWS 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 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 Kinesis all'indirizzo <https://console.aws.amazon.com/kinesis>.
2. Nel pannello Managed Service for Apache Flink, scegliete. 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. Apri la console del servizio gestito per Apache Flink all'indirizzo <https://console.aws.amazon.com/flink>
2. Nel pannello 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 del bucket e degli oggetti Amazon S3

1. Apri la console Amazon S3 all'indirizzo <https://console.aws.amazon.com/s3/>.
2. <username>Scegli il bucket ka-app-code-.
3. Scegli Elimina, quindi inserisci il nome del bucket per confermare l'eliminazione.

## 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. 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/](https://console.aws.amazon.com/cloudwatch/).
2. Nella barra di navigazione, scegli Log.
3. Scegli il gruppo di log MyApplication/aws/kinesis-analytics/.
4. Scegli Elimina gruppo di log e conferma l'eliminazione.

# Impostazioni di 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 dalle applicazioni del servizio gestito per Apache Flink nel codice, mentre altri non possono essere modificati.

Questo argomento contiene le sezioni seguenti:

- [Configurazione di Apache Flink](#)
- [Backend di stato](#)
- [Checkpoint](#)
- [Savepoint](#)
- [Dimensioni heap](#)
- [Debloating del buffer](#)
- [Proprietà di configurazione modificabili di Flink](#)
- [Visualizzazione delle 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, se è necessario modificare le proprietà di configurazione di Flink per migliorare le prestazioni di determinate applicazioni con elevato parallelismo, elevato utilizzo della memoria e dello stato o abilitare nuove funzionalità di debug in Apache Flink, è possibile 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 di stato

Il servizio gestito per Apache Flink archivia i dati transitori in un backend di stato. Il servizio gestito per Apache Flink utilizza il backend di RocksDBState. 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

Nel servizio gestito per Apache Flink, la configurazione di `state.backend.rocksdb.ttl.compaction.filter.enabled` è abilitata per impostazione predefinita. Utilizzando questo filtro, è possibile aggiornare il codice dell'applicazione per abilitare la strategia di pulizia della compattazione. [Consentire lo stato non ripristinato](#) nella [documentazione di Apache Flink](#).

Per maggiori informazioni sui backend di stato, consultare [Backend di stato](#) 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. Per usare i valori modificati per i checkpoint, occorre impostare [CheckpointConfiguration.ConfigurationType](#) su CUSTOM servizio gestito per Apache Flink.

Impostazione	Può essere modificato?	In che modo	Valore predefinito
CheckpointingEnabled	Modificabile	<a href="#">Crea applicazione</a> <a href="#">Aggiorna applicazione</a> <a href="#">AWS CloudFormation</a>	True
CheckpointInterval	Modificabile	<a href="#">Crea applicazione</a>	60000

Impostazione	Può essere modificato?	In che modo	Valore predefinito
		<a href="#">Aggiorna applicazione</a> <a href="#">AWS CloudFormation</a>	
MinPauseBetweenCheckpoints	Modificabili	<a href="#">Crea applicazione</a> <a href="#">Aggiorna applicazione</a> <a href="#">AWS CloudFormation</a>	5000
Checkpoint non allineati	Modificabile	<a href="#">Caso di supporto</a>	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
Strategia di riavvio	Non modificabile	N/D	Ritardo fisso, con tentativi infiniti ogni 10 secondi.
Ubicazione di checkpoint e savepoint	Non modificabile	N/D	Archiviamo dati durevoli su checkpoint e savepoint in un bucket S3 di proprietà del servizio.

Impostazione	Può essere modificato?	In che modo	Valore predefinito
Soglia di memoria del backend di stato	Non modificabile	N/D	1048576

## 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 la riuscita dell'operazione impostando il parametro `AllownOnRestoredState` di [FlinkRunConfiguration](#) dell'applicazione 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 heap

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 "Dimensionamento"](#).

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 attraverso un [caso 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.

### Tolleranza agli errori

```
restart-strategy:
```

```
restart-strategy.fixed-delay.delay:
```

### Checkpoint e backend di stato

```
state.backend:
```

```
state.backend.fs.memory-threshold:
```

```
state.backend.incremental:
```

### Checkpoint

```
execution.checkpointing.unaligned:
```

### Metriche native di RocksDB

I parametri nativi di RocksDB non vengono spediti a CloudWatch. Una volta abilitate, è possibile accedere a queste metriche dalla dashboard di Flink o dalla REST API di Flink con strumenti personalizzati.

[Il servizio gestito per Apache Flink consente ai clienti di accedere alla REST API più recente 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 impiegata anche da strumenti di monitoraggio personalizzati.

```
state.backend.rocksdb.compaction.style:
```

```
state.backend.rocksdb.memory.partitioned-index-filters:
```

`state.backend.rocksdb.metrics.actual-delayed-write-rate:`  
`state.backend.rocksdb.metrics.background-errors:`  
`state.backend.rocksdb.metrics.block-cache-capacity:`  
`state.backend.rocksdb.metrics.block-cache-pinned-usage:`  
`state.backend.rocksdb.metrics.block-cache-usage:`  
`state.backend.rocksdb.metrics.column-family-as-variable:`  
`state.backend.rocksdb.metrics.compaction-pending:`  
`state.backend.rocksdb.metrics.cur-size-active-mem-table:`  
`state.backend.rocksdb.metrics.cur-size-all-mem-tables:`  
`state.backend.rocksdb.metrics.estimate-live-data-size:`  
`state.backend.rocksdb.metrics.estimate-num-keys:`  
`state.backend.rocksdb.metrics.estimate-pending-compaction-bytes:`  
`state.backend.rocksdb.metrics.estimate-table-readers-mem:`  
`state.backend.rocksdb.metrics.is-write-stopped:`  
`state.backend.rocksdb.metrics.mem-table-flush-pending:`  
`state.backend.rocksdb.metrics.num-deletes-active-mem-table:`  
`state.backend.rocksdb.metrics.num-deletes-imm-mem-tables:`  
`state.backend.rocksdb.metrics.num-entries-active-mem-table:`  
`state.backend.rocksdb.metrics.num-entries-imm-mem-tables:`  
`state.backend.rocksdb.metrics.num-immutable-mem-table:`  
`state.backend.rocksdb.metrics.num-live-versions:`  
`state.backend.rocksdb.metrics.num-running-compactions:`  
`state.backend.rocksdb.metrics.num-running-flushes:`

`state.backend.rocksdb.metrics.num-snapshots:`

`state.backend.rocksdb.metrics.size-all-mem-tables:`

`state.backend.rocksdb.thread.num:`

## Opzioni avanzate di backend di stato

`state.storage.fs.memory-threshold:`

## Opzioni complete di 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 avanzate per il cluster

`cluster.intercept-user-system-exit:`

`cluster.processes.halt-on-fatal-error:`

## Configurazioni del filesystem

`fs.s3.connection.maximum:`

`fs.s3a.connection.maximum:`

`fs.s3a.threads.max:`

`s3.upload.max.concurrent.uploads:`

## Opzioni avanzate di tolleranza agli errori

`heartbeat.timeout:`

`jobmanager.execution.failover-strategy:`

## Configurazione della memoria

`jobmanager.memory.heap.size:`

## Parametri

`metrics.latency.interval:`

## Opzioni avanzate per client ed endpoint 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:`

## Visualizzazione delle 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.

# Configurazione di un servizio gestito per Apache Flink per accedere alle risorse in un cloud privato virtuale di Amazon

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. Il cloud privato virtuale (VPC) di Amazon 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:

- [Nozioni di Amazon VPC](#)
- [Autorizzazioni per applicazioni VPC](#)
- [Accesso a Internet e ai servizi per un'applicazione di servizio gestito per Apache Flink connessa a VPC](#)
- [API VPC per il servizio gestito per Apache Flink](#)
- [Esempio: utilizzare un VPC per accedere ai dati in un cluster Amazon MSK](#)

## Nozioni di Amazon VPC

Il cloud privato virtuale (VPC) di Amazon è il livello di rete per Amazon EC2. Se è la prima volta che utilizzi Amazon EC2, puoi ottenere una breve panoramica consultando [Che cos'è Amazon EC2?](#) nella Guida per l'utente di Amazon EC2 per le istanze Linux.

Di seguito sono elencati i concetti fondamentali relativi ai VPC:

- un cloud privato virtuale (VPC) è una rete virtuale dedicata all'account AWS;
- 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 consente di connettere privatamente il VPC ai servizi AWS supportati, nonché ai servizi endpoint VPC supportati da PrivateLink senza il bisogno di 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 [Dimensionamento](#).

#### 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.

### Politica di autorizzazione per accedere 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

Una volta specificate le risorse dell'applicazione utilizzando la console (come i log di CloudWatch o un Amazon VPC), quest'ultima modifica il ruolo di esecuzione dell'applicazione e concede l'autorizzazione ad accedere 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 di servizio gestito 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 del servizio gestito per Apache Flink all'indirizzo: <https://console.aws.amazon.com/flink>;
- 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](#)

## API VPC per il servizio gestito per Apache Flink

Utilizza le seguenti operazioni API per il servizio gestito per Apache Flink per gestire i VPC per l'applicazione. Per informazioni sull'utilizzo delle API per il servizio gestito per Apache Flink consulta [Codice di esempio dell'API](#).

## Crea applicazione

L'azione [Crea applicazione](#) consente di aggiungere una configurazione VPC all'applicazione nel corso della 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:::mybucket",
        "FileKey":"myflink.jar",
        "ObjectVersion":"AbCdEfGhIjKlMnOpQrStUvWxYz12345"
      }
    },
    "CodeContentType":"ZIPFILE"
  },
  "FlinkApplicationConfiguration":{
    "ParallelismConfiguration":{
      "ConfigurationType":"CUSTOM",
      "Parallelism":2,
      "ParallelismPerKPU":1,
      "AutoScalingEnabled":true
    }
  },
  "VpcConfigurations": [
    {
      "SecurityGroupIds": [ "sg-0123456789abcdef0" ],
      "SubnetIds": [ "subnet-0123456789abcdef0" ]
    }
  ]
}
}

```

## Aggiungi configurazione VPC all'applicazione

L'azione [Aggiungi configurazione VPC all'applicazione](#) consente di aggiungere una configurazione VPC all'applicazione dopo averla 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" ]
  }
}

```

```
}
```

## Elimina configurazione VPC applicazione

Utilizza l'azione [Cancella configurazione VPC applicazione](#) per rimuovere una configurazione VPC dall'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 applicazione

Utilizza l'azione [Aggiorna applicazione](#) 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 del servizio gestito per Apache Flink

Le informazioni riportate di seguito possono aiutarti a risolvere i problemi che si possono verificare con il servizio gestito da Amazon per Apache Flink.

## Argomenti

- [Risoluzione dei problemi di sviluppo](#)
- [Risoluzione dei problemi di runtime](#)

## Risoluzione dei problemi di sviluppo

### Argomenti

- [Grafici a fiamma di Apache Flink](#)
- [Problema del fornitore 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 è ricaricare il codice dell'applicazione](#)
- [S3 StreamingFileSink FileNotFoundExceptions](#)
- [FlinkKafkaConsumer problema con stop with savepoint](#)
- [Blocco di Async sink per Flink 1.15](#)
- [L'elaborazione della fonte del flusso di dati Amazon Kinesis non funziona correttamente durante la ripartizione](#)

## Grafici a fiamma di Apache Flink

I grafici a fiamma sono abilitati per impostazione predefinita nelle applicazioni nelle versioni di servizio gestito per Apache Flink che li supportano. I grafici a fiamma possono influire sulle prestazioni dell'applicazione se rimangono aperti, come indicato nella [documentazione di Flink](#).

Per disabilitare i grafici a fiamma per l'applicazione è sufficiente creare un caso per richiederne la disabilitazione per l'ARN dell'applicazione. Per ulteriori informazioni, consulta il [AWS Centro di supporto](#).

## Problema del fornitore 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

Il servizio gestito per Apache Flink per la versione 1.15 [rifiuterà automaticamente l'avvio o l'aggiornamento delle applicazioni](#) se utilizzano versioni del connettore Kinesis non supportate (precedenti alla versione 1.15.2) raggruppate in file JAR o archivi (ZIP).

### Errore: 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](#).
- Riconfeziona l'applicazione.
- Carica oggetti in un bucket Amazon S3



- Invia nuovamente la richiesta di creazione/aggiornamento dell'applicazione 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 `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. Procedi all'aggiornamento `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>

...

</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. Procedi all'aggiornamento `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 (versione 1.8.2. da <https://flink.apache.org/downloads.html>) per il connettore. 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 la versione 2 del servizio gestito API per Apache Flink è necessaria la versione più recente di AWS Command Line Interface (AWS CLI).

Per informazioni sull'installazione di AWS CLI, consulta [Installazione di AWS Command Line Interface](#) nella AWS Command Line Interface Guida per l'utente.

## UpdateApplication L'azione non è 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 di servizio gestito per Apache Flink possono essere eseguite in un file delle parti denominato 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://your-s3-bucket/pathj/
INSERT/2023/4/19/7/_part-2-1234_tmp_12345678-1234-1234-1234-123456789012
    at
    org.apache.hadoop.fs.s3a.S3AFileSystem.s3GetFileStatus(S3AFileSystem.java:2231)
    at
    org.apache.hadoop.fs.s3a.S3AFileSystem.innerGetFileStatus(S3AFileSystem.java:2149)
    at
    org.apache.hadoop.fs.s3a.S3AFileSystem.getFileStatus(S3AFileSystem.java:2088)
    at org.apache.hadoop.fs.s3a.S3AFileSystem.open(S3AFileSystem.java:699)
    at org.apache.hadoop.fs.FileSystem.open(FileSystem.java:950)
    at
    org.apache.flink.fs.s3hadoop.HadoopS3AccessHelper.getObject(HadoopS3AccessHelper.java:98)
    at
    org.apache.flink.fs.s3.common.writer.S3RecoverableMultipartUploadFactory.recoverInProgressPart
    ...
```

### [Flink StreamingFileSink scrive record su file system supportati dall'astrazione Flink. FileSystem](#)

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 rolover determinano le tempistica, la dimensione e la denominazione dei file delle parti.

#### Note

Per ulteriori informazioni, consulta [Ciclo di vita 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 servizio gestito per Apache Flink; migrare l'applicazione alla

versione 1.15 di servizio gestito per Apache Flink per evitare che si ripeta. Ti suggeriamo anche di effettuare la migrazione 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](#) nel 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 l'ombreggiatura per impacchettare alcune dipendenze, incluso il AWS SDK, nel connettore jar. Per aggiornare la versione AWS SDK e sostituire le classi ombreggiate è sufficiente seguire la procedura seguente:

Maven

1. Aggiungere il connettore Kinesis e i moduli AWS SDK richiesti come dipendenze del progetto.
2. Configurare `maven-shade-plugin`:
  - a. Aggiungere un filtro per escludere le classi AWS SDK ombreggiate durante la copia del contenuto del connettore jar di Kinesis.

- b. Aggiungere la regola di riposizionamento per spostare le classi AWS SDK aggiornate nel pacchetto previsto 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>
          ...
          <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. Aggiungere il connettore Kinesis e i moduli AWS SDK richiesti come dipendenze del progetto.
2. Modificare la configurazione di ShadowJar:
  - a. Escludere le classi AWS SDK ombreggiate nel corso della copia del contenuto del connettore jar di Kinesis.
  - b. Trasferire le classi AWS SDK aggiornate in un pacchetto atteso 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("org/apache/flink/kinesis/shaded/software/amazon/awssdk/**/*.*.class")
    exclude("org/apache/flink/kinesis/shaded/org/reactivestreams/**/*.*.class")
    exclude("org/apache/flink/kinesis/shaded/io/netty/**/*.*.class")
    exclude("org/apache/flink/kinesis/shaded/com/typesafe/netty/**/*.*.class")

    relocate("software.amazon.awssdk",
"org.apache.flink.kinesis.shaded.software.amazon.awssdk")
    relocate("org.reactivestreams",
"org.apache.flink.kinesis.shaded.org.reactivestreams")
    relocate("io.netty", "org.apache.flink.kinesis.shaded.io.netty")
    relocate("com.typesafe.netty",
"org.apache.flink.kinesis.shaded.com.typesafe.netty")
}
...
```

## Altri connettori interessati

Se l'applicazione utilizza un altro connettore interessato:

Per aggiornare la versione AWS SDK, quest'ultima deve essere applicata nella configurazione di compilazione del progetto.

## Maven

Aggiungere la distinta base AWS SDK (BOM) alla sezione di gestione delle dipendenze del file `pom.xml` 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>
      </dependency>
    </dependencies>
  </dependencyManagement>
</project>
```

```
        <scope>import</scope>
        <type>pom</type>
    </dependency>
    ...
</dependencies>
</dependencyManagement>
...
</project>
```

## Gradle

Aggiungere la dipendenza della piattaforma alla distinta base (BOM) di 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. Compilare e installare il connettore jar, specificando la versione AWS SDK richiesta. 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. Compilare e installare il connettore jar, specificando la versione AWS SDK richiesta. 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- tubo kinesis-fire

### 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 AWS SDK richiesta. 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 AWS SDK richiesta. Per velocizzare la compilazione, usa `-DskipTests` per saltare l'esecuzione dei test e `-Dfast` per saltare i controlli aggiuntivi del codice sorgente:

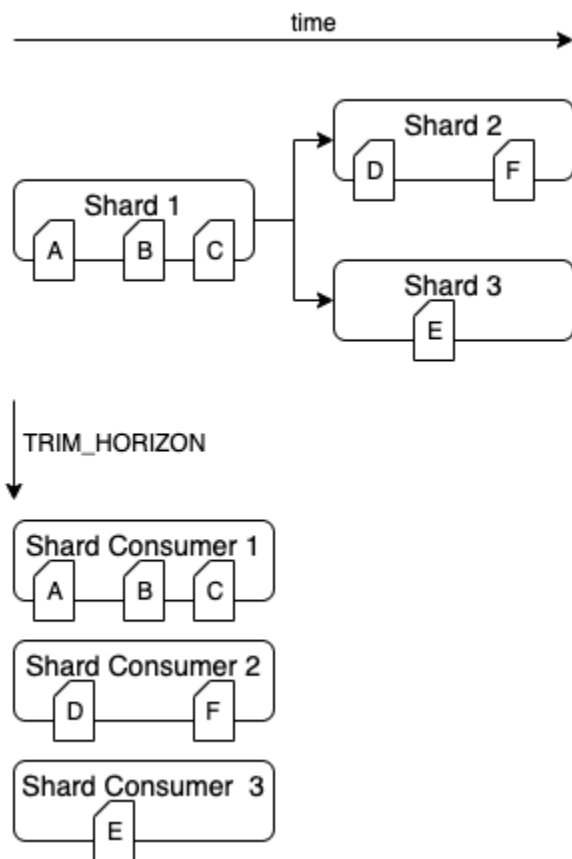
```
mvn clean install -DskipTests -Dfast -Dflink.version=1.15.4 -  
Daws.sdk.version=2.20.144
```

5. Il jar risultante sarà disponibile all'indirizzo:

```
flink-sql-connector-dynamodb/target/flink-sql-connector-dynamodb-3.0.0.jar
```

## L'elaborazione della fonte del flusso di dati Amazon Kinesis non funziona correttamente durante la ripartizione

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.



# 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](#)
- [Problematiche delle applicazioni](#)
- [L'applicazione si riavvia](#)
- [La velocità di trasmissione effettiva è troppo lenta](#)
- [Crescita illimitata dello stato](#)
- [Operatori vincolati all'I/O](#)
- [Limitazione 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 regioni differenti](#)

## Strumenti per la risoluzione dei problemi

Gli allarmi CloudWatch rappresentano lo strumento principale per rilevare le problematiche delle applicazioni. Consentono infatti di impostare soglie per i parametri CloudWatch, che indicano errori o problematiche relative all'applicazione. Per informazioni relative agli allarmi CloudWatch consigliati, consulta [Utilizzo degli CloudWatch allarmi con Amazon Managed Service per Apache Flink](#).

## Problematiche delle applicazioni

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 rimasta bloccata in uno stato di transizione](#)

- [Generazione di snapshot non riuscita](#)
- [Impossibile accedere alle risorse in un VPC](#)
- [I dati sono stati persi nel corso della scrittura su un bucket Amazon S3](#)
- [L'applicazione è nello stato IN ESECUZIONE ma non elabora i dati](#)
- [Errori legati agli snapshot, all'aggiornamento delle applicazioni o al loro arresto: `InvalidApplicationConfigurationException`](#)
- [`java.nio.file.NoSuchFileException: /usr/local/openjdk-8/lib/security/cacerts`](#)

## L'applicazione è rimasta bloccata in uno stato di transizione

Se l'applicazione rimane in uno stato transitorio (STARTING, UPDATING, STOPPING o AUTOSCALING) è possibile forzarne l'interruzione utilizzando [StopApplication](#) e impostando su `Force` il parametro `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, è consigliabile consultare l'azione [RollbackApplication](#).

### 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 fallisce, consulta quanto [Generazione di snapshot non riuscita](#) 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'istantanea contiene dati sullo stato dell'operatore mancante. Inoltre, l'applicazione rimarrà bloccata nello status STOPPED o UPDATING. [Per modificare questo comportamento e consentire il completamento del ripristino, modifica il parametro AllowOnRestoredState della configurazione FlinkRunConfiguration](#), trasformandolo 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 log di CloudWatch 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. Consigliamo di controllare i log del JobManager di Flink e il codice dell'applicazione, per verificare se il ritardo nel metodo `main` è normale. 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](#).

## Generazione di snapshot non riuscita

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 [Snapshot](#).
- 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:

- Verifica che i log di CloudWatch non contengano il 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 log di CloudWatch, consulta [Registrazione e monitoraggio](#).

## I dati sono stati persi nel corso della 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. L'utilizzo di Kinesis Data Firehose è raccomandato per scrivere su un bucket Amazon S3 utilizzando Apache Flink 1.6.2. Per ulteriori

informazioni sull'utilizzo di Kinesis Data Firehose con servizio gestito per Apache Flink, consulta [Sink Kinesis Data Firehose](#).

L'applicazione è nello stato IN ESECUZIONE ma non elabora i dati

È possibile controllare lo stato dell'applicazione utilizzando le azioni [ListApplications](#) o [DescribeApplication](#). Se l'applicazione entra nello stato RUNNING ma non scrive dati sul sink, per risolvere il problema è sufficiente aggiungere all'applicazione un flusso di log di Amazon CloudWatch. Per ulteriori informazioni, consulta [Utilizzo delle opzioni di CloudWatch registrazione delle applicazioni](#). Il flusso di log contiene messaggi che consentono di risolvere eventuali problemi relativi alle applicazioni.

Errori legati agli snapshot, all'aggiornamento delle applicazioni o al loro arresto:  
`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 del servizio gestito per Apache Flink o utilizzando il parametro `SnapshotsEnabledUpdate` dell'azione [UpdateApplication](#).
- prova a comprendere perché è impossibile creare gli snapshot; Per ulteriori informazioni, consulta [L'applicazione è rimasta bloccata in uno stato di transizione](#).
- 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 riavvia

Se l'applicazione non è integra, il processo Apache Flink fallisce e si riavvia continuamente. Questa sezione descrive i sintomi e le procedure di risoluzione dei problemi relativi a questa condizione.

### Sintomi

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 registro dell'applicazione contiene le modifiche allo stato `RESTARTING` o `FAILED`. Puoi interrogare il registro dell'applicazione per queste modifiche di stato utilizzando la seguente query di CloudWatch Logs Insights: [Analisi degli errori: errori relativi alle attività applicative](#).

### 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 (perché interpreta 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 "Analisi degli errori: errori relativi alle attività applicative"](#).

- I flussi di dati Kinesis vengono forniti correttamente: se un'origine o un sink della tua applicazione è un flusso di dati Kinesis, controlla le [metriche](#) del flusso alla ricerca di errori `ReadProvisionedThroughputExceeded` o `WriteProvisionedThroughputExceeded`.

Se visualizzi questi errori, puoi aumentare la velocità effettiva 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 origini o i sink utilizzati nell'applicazione (ad esempio altri servizi AWS o origini o destinazioni esterne) siano forniti correttamente, non presentino limitazioni di lettura o scrittura o siano periodicamente non disponibili.

Se riscontri problemi relativi alla velocità di trasmissione effettiva 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 log 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 log 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 i parametri di CloudWatch per rilevare picchi insoliti nell'utilizzo della CPU o della memoria.
- Controlla se l'applicazione presenta problemi di velocità di trasmissione effettiva. Per ulteriori informazioni, consulta [Risoluzione dei 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 non funziona 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 esecuzione.

Per maggiori informazioni sull'utilizzo di Apache Beam con il servizio gestito per Apache Flink, consulta [Utilizzo di CloudFormation con il servizio gestito per Apache Flink](#).

- 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 la 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 richiedere una configurazione personalizzata dei buffer di rete.

## La velocità di trasmissione 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 riavvia](#).
- 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 [Risoluzione dei 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 [Dimensionamento](#).
- 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 [Risoluzione dei problemi relativi alle prestazioni](#).

## Crescita illimitata dello stato

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.

### Sintomi

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 conserva 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 [Time-To-Live \(TTL\) dello stato](#) 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 BeamApplicationProperties](#) con gli esperimenti chiave e il valore `use_deprecated_read`. Per ulteriori informazioni, consulta la [documentazione di Apache Sqoop](#).

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 vengano utilizzate [AggregateFunctions](#), 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, è possibile utilizzare il [Time-To-Live dello stato](#) 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 RockSDB). 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 vincolati all'I/O può essere utile aumentare l'impostazione [ParallelismPerKPU](#) dell'applicazione del servizio gestito per 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 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`. Per impostazione predefinita, Apache Flink è configurato per recuperare 10.000 record per chiamata `GetRecords` (questa chiamata viene effettuata per impostazione predefinita ogni 200 ms), sebbene il limite per partizione sia di soli 1.000 record.

Questo comportamento predefinito può portare a una limitazione quanto si tenta di utilizzare dati dal flusso di dati Kinesis, il che influisce sulle prestazioni e sulla stabilità dell'applicazione.

Ciò può essere confermato controllando il parametro `ReadProvisionedThroughputExceeded` di `CloudWatch` e rilevando periodi prolungati o duraturi in cui tale metrica è maggiore di zero.

Il cliente potrà inoltre visualizzarlo nei log di `CloudWatch` per l'applicazione del servizio gestito per Apache Flink visualizzando gli errori di `LimitExceededException` continui.

Risoluzione: il cliente può compiere una delle due seguenti azioni per risolvere questo scenario:

- Ridurre il limite predefinito per il numero di record recuperati per chiamata `GetRecords`.
- Il cliente può abilitare `Adaptive Reads` nella propria applicazione del servizio gestito per 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: solitamente si tratta di una fase breve; l'attività secondaria invia una conferma al JobManager ed esegue eventuali messaggi di commit (ad esempio con i sink di Kafka).

Ciascuna di queste fasi (esclusa Riconoscimento) corrisponde a un parametro di durata per il 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			
ID	Acknowledged	End to End Duration	Checkpointed Data Size	Sync Duration	Async Duration	Processed (persisted) Data	Alignment Duration	Start Delay	Unaligned Checkpoint	
0	2022-03-02 14:16:49	566ms	11.1 KB	8ms	429ms	0 B (0 B)	0ms	126ms	false	
1	2022-03-02 14:16:50	1s	1.70 MB	69ms	1s	0 B (0 B)	0ms	128ms	false	
2	2022-03-02 14:16:49	495ms	11.1 KB	8ms	357ms	0 B (0 B)	1ms	126ms	false	

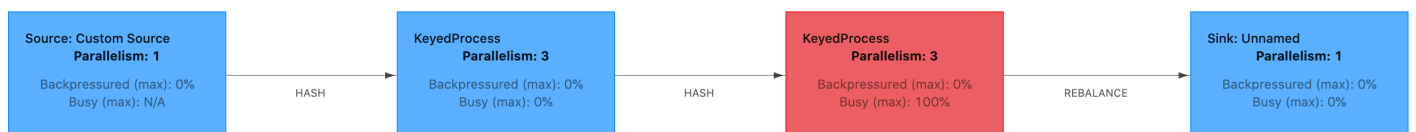
  

-	Sink: Unnamed	1/1 (100%)	2022-03-02 14:16:49	131ms	0 B	0 B (0 B)
---	---------------	------------	---------------------	-------	-----	-----------

SubTasks:										
-----------	--	--	--	--	--	--	--	--	--	--

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 grafico del processo in cui il secondo operatore KeyedProcess è occupato.



I thread dump dei grafici a fiamma e dei task manager di Flink possono essere impiegati per analizzare le cause dei rallentamenti. 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(AbstractStr
  at app//
org.apache.flink.streaming.runtime.io.AbstractStreamTaskNetworkInput.emitNext(AbstractStreamTas
  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 del thread `KeyedProcess (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 attraverso il nome, poiché riportano lo stesso nome dell'operatore unito a informazioni relative all'attività secondaria a cui sono correlati: per esempio il thread `KeyedProcess (1/3) #0` proviene dall'operatore `KeyedProcess` e dalla prima attività secondaria di 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.

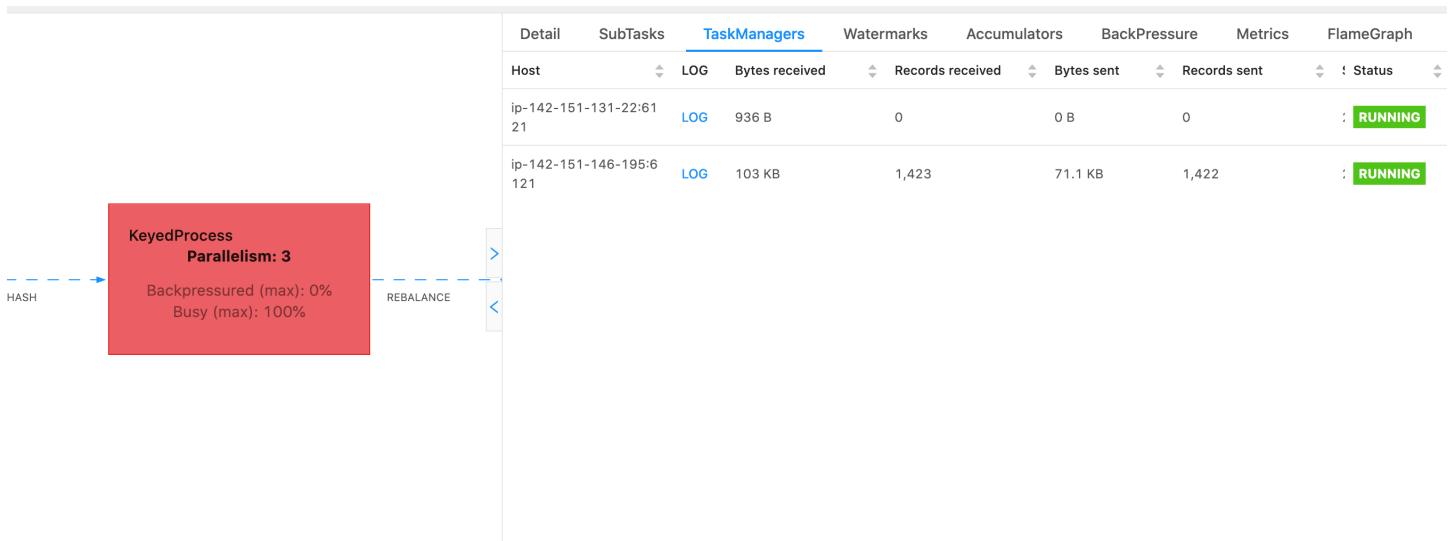
#### Note

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 scheda TaskManager quando si sceglie un determinato operatore. Ciò dimostra che l'operatore è in esecuzione su diverse attività secondarie di un operatore e può essere eseguito su diversi Task Manager.



The screenshot shows a Flink dashboard with a 'TaskManagers' tab. On the left, a red box represents a 'KeyedProcess' operator with 'Parallelism: 3'. It shows 'Backpressured (max): 0%' and 'Busy (max): 100%'. A 'HASH' arrow points to it from the left, and a 'REBALANCE' arrow points to it from the right. On the right, a table lists TaskManager details:

Host	LOG	Bytes received	Records received	Bytes sent	Records sent	Status
ip-142-151-131-22:61 21	LOG	936 B	0	0 B	0	RUNNING
ip-142-151-146-195:6 121	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'operatore KeyedProcess ed è la prima attività secondaria.

```
"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(AbstractStreamTaskNetwo
  at app//
org.apache.flink.streaming.runtime.io.StreamOneInputProcessor.processInput(StreamOneInputProcess
  ...
```

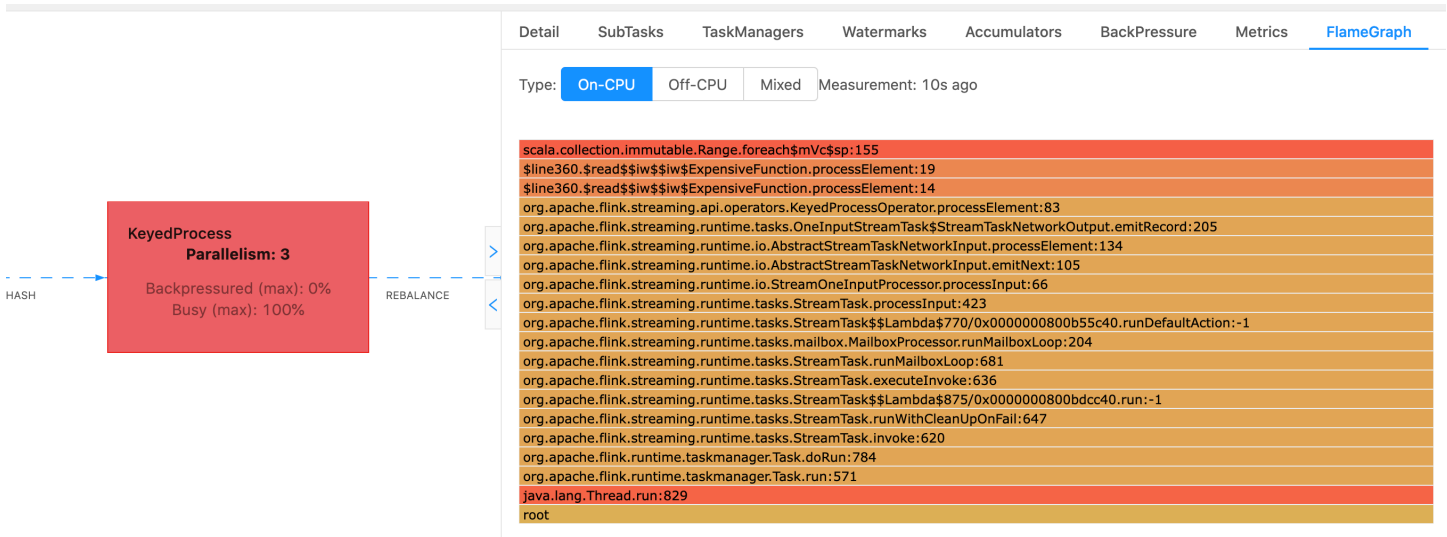
Questo può creare confusione se ci sono più operatori con lo stesso nome, tuttavia per aggirare il problema è possibile rinominare gli operatori. Per esempio:

```
....
.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](#).

Su Flink è possibile accedere al grafico a fiamma di un operatore tramite l'interfaccia utente Web, selezionando un operatore e scegliendo la scheda Grafici a fiamma. Una volta raccolto un numero sufficiente di campioni, verrà visualizzato il grafico a fiamma. Di seguito è riportato il grafico a fiamma per la ProcessFunction che ha richiesto tempistiche prolungate per effettuare il checkpoint.



Questo è un grafico a fiamma estremamente semplice; mostra che il tempo della CPU viene impiegato in un unico `foreach` all'interno del `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, consigliamo di consultare la scheda [Serializzatori personalizzati](#) 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 consulta l'argomento relativo al [monitoraggio dei checkpoint](#) 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 [Dimensionamento](#).
- 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 [ShutdownSourcesAfterIdleMS](#) impostato su 0ms, i checkpoint potrebbero non essere attivati, perché le attività compaiono nello stato "FINITO". La presente sezione descrive le caratteristiche e le soluzioni relative a questa situazione.

### Caratteristiche

Vai ai log di CloudWatch dell'applicazione servizio gestito per Apache Flink e controlla se il seguente messaggio di log 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-agg-r3-pc-1	2022-06-06 11:16:03	13m 57s	RUNNING	...
1	0 B	0	0 B	0	1	sea3-ws-agg-r3-pc-1	2022-06-06 11:16:03	3s	FINISHED	...
2	0 B	0	0 B	0	1	sea3-ws-agg-r3-pc-1	2022-06-06 11:16:03	3s	FINISHED	...
3	0 B	0	0 B	0	1	sea3-ws-agg-r3-pc-1	2022-06-06 11:16:03	3s	FINISHED	...
4	0 B	0	0 B	0	1	sea3-ws-agg-r3-pc-1	2022-06-06 11:16:03	3s	FINISHED	...

## Causa

`shutdownSourcesAfterIdleMS` è una variabile di configurazione di Beam che spegne le sorgenti rimaste inattive per il numero di millisecondi configurato. Il checkpoint non è più possibile quando una fonte viene chiusa. La conseguenza potrebbe essere un [checkpoint non riuscito](#).

Quando `ShutdownSourcesAfterIdlems` è impostato su `0ms`, le attività inattive verranno chiuse immediatamente. Si tratta di una delle cause per cui le attività passano allo stato "FINITO".

## Soluzione

Per evitare che le attività passino immediatamente allo stato "FINITO", imposta `ShutdownSourcesAfterIdlems` su `Long.MAX_VALUE`. Esistono due maniere per assicurarsene:

- Opzione 1: se la configurazione di beam è impostata nella pagina di configurazione dell'applicazione servizio gestito per Apache Flink, è possibile aggiungere una nuova coppia chiave-valore per impostare `shutdpwnSourcesAfterIdleMS` come segue:

Group	Key	Value
BeamApplicationProperties	ShutdownSourcesAfterIdleMs	9223372036854775807

- Opzione 2: se la configurazione di beam è impostata nel file JAR, imposta `ShutdownSourcesAfterIdleMS` come segue:

```
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 impiegare la scalabilità orizzontale per 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 velocità di trasmissione effettiva con 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 practices 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 le semantiche `exactly-once` abilitate per impostazione predefinita su servizio gestito per Apache Flink), l'elaborazione si arresta da una prospettiva esterna fino al completamento del checkpoint/savepoint. In caso di disallineamento dei dati, il tempo necessario per completare l'operazione può essere vincolato da una singola attività secondaria che ha accumulato una quantità di stato particolarmente elevata. In casi estremi, quando una singola attività secondaria non è in grado di conservare lo stato, l'esecuzione dei checkpoint/savepoint può non avere successo.

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 regioni differenti

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 [AWS SupportCentro](#).

# Cronologia dei documenti per il servizio gestito da Amazon per 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, alle API, all'interfaccia a riga di comando, alle politiche di accesso IAM, alle CloudWatch metriche o ai dashboard di fatturazione. AWS Le applicazioni esistenti continueranno a funzionare e come prima. Per ulteriori informazioni, consulta <a href="#">Cos'è il servizio gestito per Apache Flink?</a>	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 <a href="#">Creazione di applicazioni.</a>	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 <a href="#">Nozioni di base: Flink 1.13.2</a> .	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 <a href="#">Uso di Python</a> .	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 <a href="#">Creazione di applicazioni</a> .	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 <a href="#">Pannello di controllo di Apache Flink</a> .	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 <a href="#">Consumatore EFO</a> .	6 ottobre 2020
Apache Beam	Crea applicazioni che utilizzano o Apache Beam per elaborare i dati di streaming. Per ulteriori informazioni, consulta <a href="#">Utilizzo di CloudFormation con il servizio gestito per Apache Flink</a> .	15 settembre 2020
Prestazioni	Come risolvere i problemi di prestazioni delle applicazioni e come creare un'applicazione performante. Per ulteriori informazioni, consulta <a href="#">Prestazioni</a> .	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 <a href="#">Truststore personalizzato</a> .	10 giugno 2020
CloudWatch Allarmi	Consigli per la creazione di CloudWatch allarmi con Managed Service for Apache Flink. Per ulteriori informazioni, consulta <a href="#">Allarmi</a> .	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 <a href="#">Metriche e dimensioni nel servizio gestito per Apache Flink</a> .	12 maggio 2020
Metriche personalizzate CloudWatch	Definisci parametri specifici dell'applicazione e inviali ad Amazon Metrics. CloudWatch Per ulteriori informazioni, consulta <a href="#">Parametri personalizzati</a> .	12 maggio 2020
Esempio: lettura da un flusso Kinesis in un account diverso	Scopri come accedere a un flusso Kinesis in un altro account AWS nella tua applicazione del servizio gestito per Apache Flink. Per ulteriori informazioni, consulta <a href="#">Multi-account</a> .	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 <a href="#">Creazione di applicazioni</a> .	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 <a href="#">Utilizzare un Amazon VPC</a> .	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 <a href="#">Best practice</a> .	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 <a href="#">Analisi dei log</a> .	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 <a href="#">Proprietà di runtime</a> .	24 giugno 2019
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 <a href="#">Utilizzo del tagging</a> .	8 maggio 2019

Modifica	Descrizione	Data
Applicazioni di esempio del servizio gestito per Apache Flink	Esempi di applicazioni per Managed Service for Apache Flink che illustrano gli operatori di finestre e scrivono l'output in Logs. CloudWatch Per ulteriori informazioni, consulta <a href="#">Esempi</a> .	1 maggio 2019
Registrazione delle chiamate API del servizio gestito di Apache Flink con AWS 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 AWS nel servizio gestito per Apache Flink. Per ulteriori informazioni, consulta <a href="#">Uso di AWS CloudTrail</a> .	22 marzo 2019
Creazione di un'applicazione (sink Kinesis Data Firehose)	Esercitati a creare un servizio gestito per Apache Flink con un flusso di dati Amazon Kinesis come origine e un flusso Amazon Kinesis Data Firehose come sink. Per ulteriori informazioni, consulta <a href="#">Sink Kinesis Data Firehose</a> .	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 del servizio gestito 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'operazione con AWS Command Line Interface (AWS CLI), salva 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 la AWS CLI, consulta [Generazione dello skeleton e dei parametri di input JSON dell'interfaccia a riga di comando](#) nella Guida per l'utente dell'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'operazione [AddApplicationCloudWatchLoggingOption](#) aggiunge un'opzione di registrazione Amazon CloudWatch a un'applicazione del servizio gestito per 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'operazione [AddApplicationInput](#) aggiunge un input dell'applicazione a un'applicazione del servizio gestito per 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'operazione

[AddApplicationInputProcessingConfiguration](#) aggiunge una configurazione di elaborazione dell'input dell'applicazione a un'applicazione del servizio gestito per 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'operazione [AddApplicationOutput](#) aggiunge un flusso di dati Kinesis come output dell'applicazione a un'applicazione del servizio gestito per 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'operazione [AddApplicationReferenceDataSource](#) aggiunge un'origine dati di riferimento dell'applicazione CSV a un'applicazione del servizio gestito per Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 5,
  "ReferenceDataSource": {
    "ReferenceSchema": {
      "RecordColumns": [
        {
```

```

        "Mapping": "$.TICKER",
        "Name": "TICKER",
        "SqlType": "VARCHAR(4)"
    },
    {
        "Mapping": "$.COMPANYNAME",
        "Name": "COMPANY_NAME",
        "SqlType": "VARCHAR(40)"
    },
],
"RecordEncoding": "UTF-8",
"RecordFormat": {
    "MappingParameters": {
        "CSVMappingParameters": {
            "RecordColumnDelimiter": " ",
            "RecordRowDelimiter": "\r\n"
        }
    },
    "RecordFormatType": "CSV"
}
},
"S3ReferenceDataSource": {
    "BucketARN": "arn:aws:s3:::MyS3Bucket",
    "FileKey": "TickerReference.csv"
},
"TableName": "string"
}
}

```

## AddApplicationVpcConfiguration

Il seguente codice di richiesta di esempio per l'operazione [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'operazione [CreateApplication](#) crea un'applicazione del servizio gestito per Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "ApplicationDescription": "My-Application-Description",
  "RuntimeEnvironment": "FLINK-1_15",
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole",
  "CloudWatchLoggingOptions": [
    {
      "LogStreamARN": "arn:aws:logs:us-east-1:123456789123:log-group:my-log-group:log-stream:My-LogStream"
    }
  ],
  "ApplicationConfiguration": {
    "EnvironmentProperties": {
      "PropertyGroups": [
        {
          "PropertyGroupId": "ConsumerConfigProperties",
          "PropertyMap": {
            "aws.region": "us-east-1",
            "flink.stream.initpos": "LATEST"
          }
        },
        {
          "PropertyGroupId": "ProducerConfigProperties",
          "PropertyMap": {
            "aws.region": "us-east-1"
          }
        }
      ]
    }
  },
  "ApplicationCodeConfiguration": {
    "CodeContent": {
      "S3ContentLocation": {
        "BucketARN": "arn:aws:s3:::mybucket",
        "FileKey": "myflink.jar",
        "ObjectVersion": "AbCdEfGhIjKlMnOpQrStUvWxYz12345"
      }
    }
  },
  "CodeContentType": "ZIPFILE"
}
```

```
    },
    "FlinkApplicationConfiguration":{
      "ParallelismConfiguration":{
        "ConfigurationType":"CUSTOM",
        "Parallelism":2,
        "ParallelismPerKPU":1,
        "AutoScalingEnabled":true
      }
    }
  }
}
```

## CreateApplicationSnapshot

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

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

## DeleteApplication

Il seguente codice di richiesta di esempio per l'operazione [DeleteApplication](#) elimina un'applicazione del servizio gestito per Apache Flink:

```
{"ApplicationName": "MyApplication",
 "CreateTimestamp": 12345678912}
```

## DeleteApplicationCloudWatchLoggingOption

Il seguente codice di richiesta di esempio per l'operazione [DeleteApplicationCloudWatchLoggingOption](#) elimina un'opzione di registrazione Amazon CloudWatch da un'applicazione del servizio gestito per Apache Flink:

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

```
"CloudWatchLoggingOptionId": "3.1"
"CurrentApplicationVersionId": 3
}
```

## DeleteApplicationInputProcessingConfiguration

Il seguente codice di richiesta di esempio per l'operazione

[DeleteApplicationInputProcessingConfiguration](#) rimuove una configurazione di elaborazione dell'input da un'applicazione del servizio gestito per Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 4,
  "InputId": "2.1"
}
```

## DeleteApplicationOutput

Il seguente codice di richiesta di esempio per l'operazione [DeleteApplicationOutput](#) rimuove l'output di un'applicazione da un'applicazione del servizio gestito per Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 4,
  "OutputId": "4.1"
}
```

## DeleteApplicationReferenceDataSource

Il seguente codice di richiesta di esempio per l'operazione [DeleteApplicationReferenceDataSource](#) rimuove un'origine dati di riferimento dell'applicazione da un'applicazione del servizio gestito per Apache Flink:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 5,
  "ReferenceId": "5.1"
}
```



## DeleteApplicationSnapshot

Il seguente codice di richiesta di esempio per l'operazione [DeleteApplicationSnapshot](#) elimina uno snapshot dello stato dell'applicazione:

```
{
  "ApplicationName": "MyApplication",
  "SnapshotCreationTimestamp": 12345678912,
  "SnapshotName": "MySnapshot"
}
```

## DeleteApplicationVpcConfiguration

Il seguente codice di richiesta di esempio per l'operazione [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'operazione [DescribeApplication](#) restituisce dettagli su un'applicazione del servizio gestito per Apache Flink:

```
{"ApplicationName": "MyApplication"}
```

## DescribeApplicationSnapshot

Il seguente codice di richiesta di esempio per l'operazione [DescribeApplicationSnapshot](#) restituisce dettagli su uno snapshot dello stato dell'applicazione:

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

```
}
```

## DiscoverInputSchema

Il seguente codice di richiesta di esempio per l'operazione [DiscoverInputSchema](#) genera uno schema da un'origine 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'operazione [DiscoverInputSchema](#) genera uno schema da una fonte di riferimento:

```
{
  "S3Configuration": {
    "BucketARN": "arn:aws:s3:::mybucket",
    "FileKey": "TickerReference.csv"
  },
  "ServiceExecutionRole": "arn:aws:iam::123456789123:role/myrole"
}
```

## ListApplications

Il seguente codice di richiesta di esempio per l'operazione [ListApplications](#) restituisce un elenco di applicazioni del servizio gestito per Apache Flink nel tuo account:

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

## ListApplicationSnapshots

Il seguente codice di richiesta di esempio per l'operazione [ListApplicationSnapshots](#) restituisce un elenco di snapshot dello stato dell'applicazione:

```
{"ApplicationName": "MyApplication",
  "Limit": 50,
  "NextToken": "aBcDeFgHiJkLmNoPqRsTuVwXyZ0123"
}
```

## StartApplication

Il seguente codice di richiesta di esempio per l'operazione [StartApplication](#) avvia un'applicazione del servizio gestito per Apache Flink e carica lo stato dell'applicazione dall'ultimo snapshot (se presente):

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

## StopApplication

Il seguente codice di richiesta di esempio per l'operazione [API\\_StopApplication](#) arresta un'applicazione del servizio gestito per Apache Flink:

```
{"ApplicationName": "MyApplication"}
```

## UpdateApplication

Il seguente codice di richiesta di esempio per l'operazione [UpdateApplication](#) aggiorna un'applicazione del servizio gestito per Apache Flink per modificare la posizione del codice dell'applicazione:

```
{
  "ApplicationName": "MyApplication",
  "CurrentApplicationVersionId": 1,
  "ApplicationConfigurationUpdate": {
    "ApplicationCodeConfigurationUpdate": {
      "CodeContentTypeUpdate": "ZIPFILE",
      "CodeContentUpdate": {
        "S3ContentLocationUpdate": {
          "BucketARNUpdate": "arn:aws:s3:::my_new_bucket",
          "FileKeyUpdate": "my_new_code.zip",
          "ObjectVersionUpdate": "2"
        }
      }
    }
  }
}
```

# Documentazione di riferimento delle API del servizio gestito per Apache Flink

Per informazioni sulle API fornite dal servizio gestito per Apache Flink, consulta la [Documentazione di riferimento delle API del servizio gestito per Apache Flink](#).