# Best practice per Confluent Kafka

NetApp Solutions

NetApp
April 26, 2024

# Sommario

# Best practice per Confluent Kafka

## TR-4912: Linee guida sulle Best practice per lo storage a più livelli Confluent Kafka con NetApp

Karthikeyan Nagalingam, Joseph Kandatilparambil, NetApp Rankesh Kumar, Confluent

Apache Kafka è una piattaforma per lo streaming di eventi distribuita dalla community in grado di gestire migliaia di miliardi di eventi al giorno. Inizialmente concepito come coda di messaggistica, Kafka si basa su un'astrazione di un log di commit distribuito. Da quando è stata creata e open-source da LinkedIn nel 2011, Kafka si è evoluta da una coda di messaggi a una piattaforma completa per lo streaming di eventi. Confluent offre la distribuzione di Apache Kafka con la Confluent Platform. La piattaforma Confluent integra Kafka con funzionalità commerciali e di community aggiuntive progettate per migliorare l'esperienza di streaming di operatori e sviluppatori in produzione su vasta scala.

Questo documento descrive le linee guida sulle Best practice per l'utilizzo dello storage a livelli confluente su un'offerta di storage a oggetti NetApp, fornendo i seguenti contenuti:

- Verifica confluente con lo storage a oggetti NetApp: NetApp StorageGRID
- Tiered storage performance test
- Linee guida sulle Best practice per Confluent sui sistemi storage NetApp

### Perché lo storage a livelli confluente?

Confluent è diventata la piattaforma di streaming real-time predefinita per molte applicazioni, in particolare per i big data, gli analytics e i carichi di lavoro di streaming. Tiered Storage consente agli utenti di separare il calcolo dallo storage nella piattaforma Confluent. L'archiviazione dei dati risulta più conveniente, consente di memorizzare quantità virtualmente infinite di dati e di scalare i carichi di lavoro on-demand in su (o in giù) e semplifica le attività amministrative come il ribilanciamento dei dati e dei tenant. I sistemi storage compatibili con S3 possono sfruttare tutte queste funzionalità per democratizzare i dati con tutti gli eventi in un unico posto, eliminando la necessità di un'ingegneria dei dati complessa. Per ulteriori informazioni sul motivo per cui dovresti utilizzare lo storage a più livelli per Kafka, consulta "Questo articolo di Confluent".

### Perché scegliere NetApp StorageGRID per lo storage su più livelli?

StorageGRID è una piattaforma di storage a oggetti leader del settore di NetApp. StorageGRID è una soluzione di storage a oggetti, software-defined, che supporta API a oggetti standard di settore, inclusa l'API S3 (Simple Storage Service) di Amazon. StorageGRID archivia e gestisce i dati non strutturati su larga scala per fornire uno storage a oggetti sicuro e durevole. I contenuti vengono posizionati nella giusta posizione, al momento giusto e nel giusto Tier di storage, ottimizzando i flussi di lavoro e riducendo i costi per i contenuti multimediali distribuiti a livello globale.

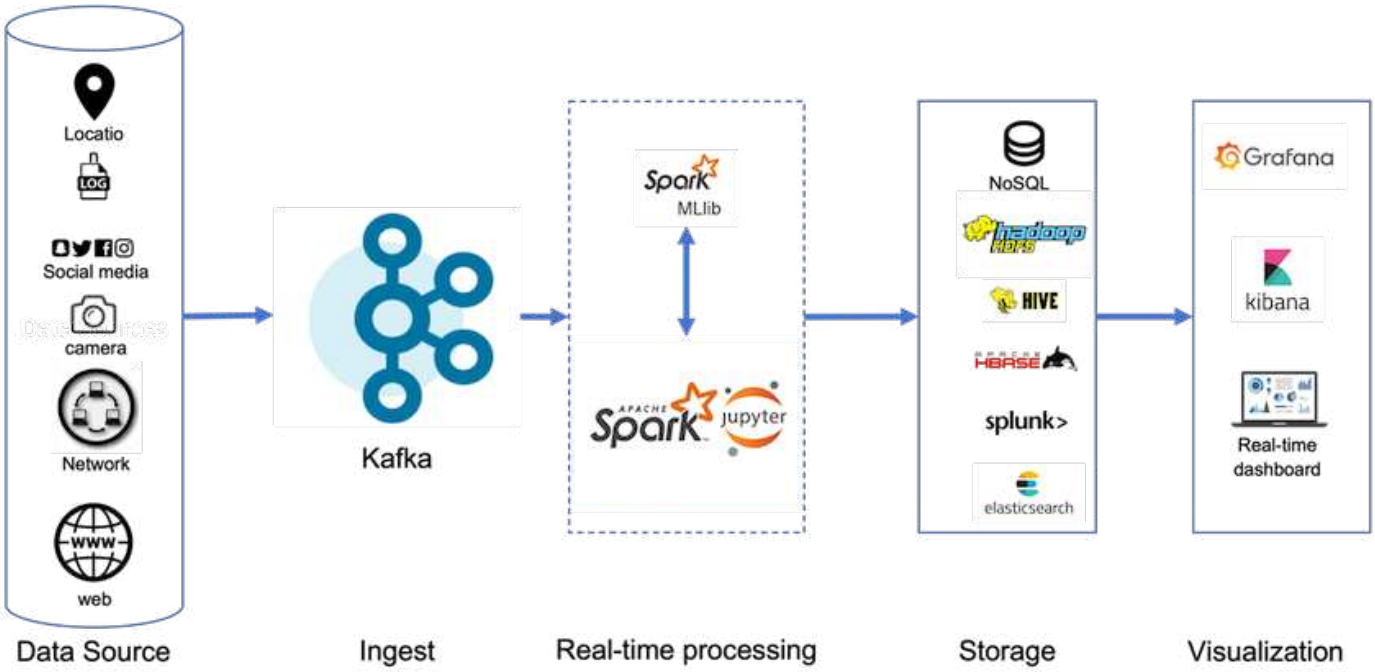
Il principale elemento di differenziazione per StorageGRID è il motore di policy per la gestione del ciclo di vita delle informazioni (ILM) che consente la gestione del ciclo di vita dei dati basata su policy. Il motore delle policy può utilizzare i metadati per gestire il modo in cui i dati vengono memorizzati per tutta la vita utile, per ottimizzare inizialmente le performance e ottimizzare automaticamente i costi e la durata con l'invecchiamento dei dati.

## Abilitare lo storage a livelli confluente

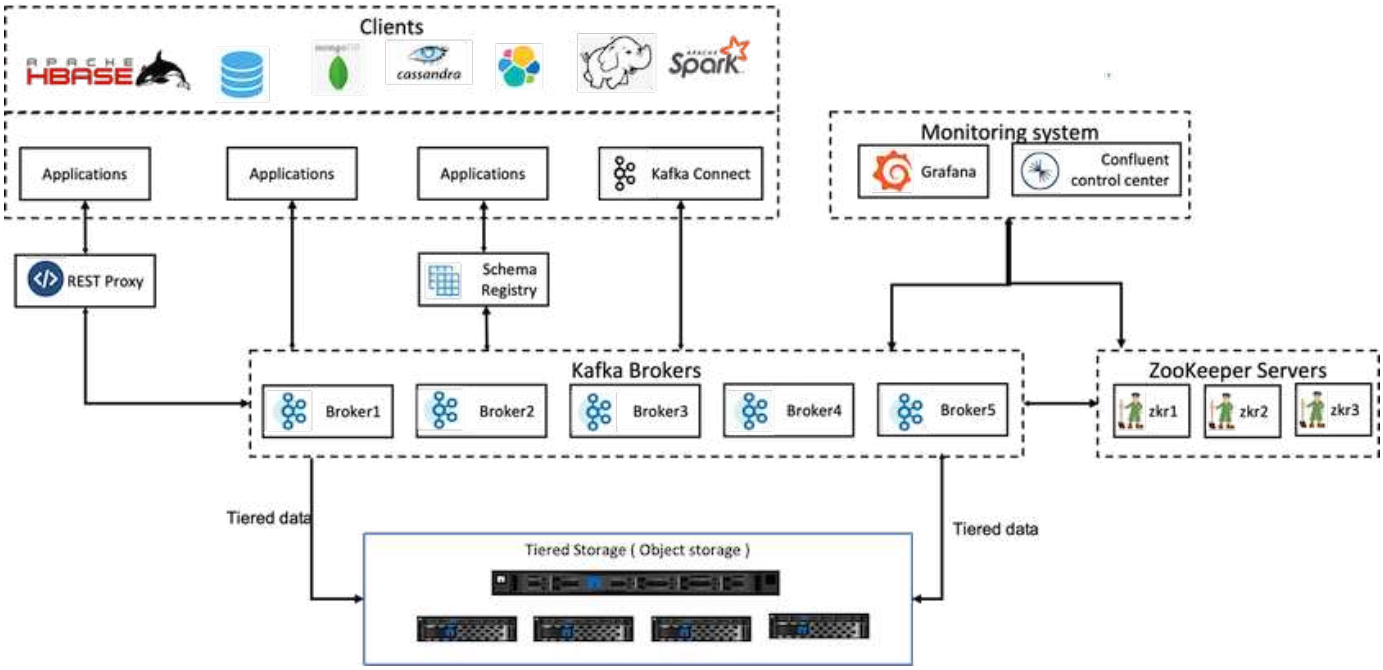
L'idea di base dello storage su più livelli è separare le attività dello storage dei dati dall'elaborazione dei dati. Grazie a questa separazione, la scalabilità indipendente del Tier di storage e del Tier di elaborazione dei dati diventa molto più semplice.

Una soluzione storage a più livelli per Confluent deve affrontare due fattori. Innanzitutto, deve aggirare o evitare proprietà comuni di coerenza e disponibilità degli archivi di oggetti, come incoerenze nelle operazioni DI ELENCO e occasionali indisponibilità degli oggetti. In secondo luogo, deve gestire correttamente l'interazione tra lo storage su più livelli e il modello di tolleranza agli errori e replica di Kafka, inclusa la possibilità che i leader zombie continuino a tierare gli intervalli di offset. Lo storage a oggetti NetApp offre la disponibilità costante degli oggetti e il modello ha che rendono lo storage stanco disponibile per gli intervalli di offset del Tier. Lo storage a oggetti NetApp offre una disponibilità degli oggetti coerente e un modello ha per rendere lo storage stanco disponibile per gli intervalli di offset del Tier.

Con lo storage a più livelli, puoi utilizzare piattaforme dalle performance elevate per letture e scritture a bassa latenza in prossimità della coda dei dati in streaming e puoi anche utilizzare archivi di oggetti scalabili e più economici come NetApp StorageGRID per letture storiche ad alto throughput. Disponiamo anche di una soluzione tecnica per Spark con controller di storage netapp e i dettagli sono qui. La figura seguente mostra come Kafka si inserisce in una pipeline di analytics in tempo reale.




La figura seguente mostra come NetApp StorageGRID si inserisce nel livello di storage a oggetti di Confluent Kafka.




# Dettagli sull'architettura della soluzione

Questa sezione descrive l'hardware e il software utilizzati per la verifica confluente. Queste informazioni sono applicabili all'implementazione della piattaforma confluente con lo storage NetApp. La seguente tabella illustra l'architettura della soluzione testata e i componenti di base.

| Componenti della soluzione | Dettagli |
|---|---|
| Confluent Kafka versione 6.2 | • Tre zookeeper<br>• Cinque server di broker<br>• Cinque server di strumenti<br>• Una Grafana<br>• Un centro di controllo |
| Linux (Ubuntu 18.04) | Tutti i server |
| NetApp StorageGRID per lo storage su più livelli | • Software StorageGRID<br>• 1 SG1000 (bilanciamento del carico)<br>• 4 x SGF6024<br>• 4 SSD 24 x 800<br>• Protocollo S3<br>• 4 x 100 GbE (connettività di rete tra istanze di broker e StorageGRID) |
| 15 server Fujitsu PRIMERGY RX2540 | Ciascuno dotato di: * 2 CPU, 16 core fisici totali * Intel Xeon * 256 GB di memoria fisica * 100 GbE a doppia porta |

# Panoramica della tecnologia

Questa sezione descrive la tecnologia utilizzata in questa soluzione.

## NetApp StorageGRID

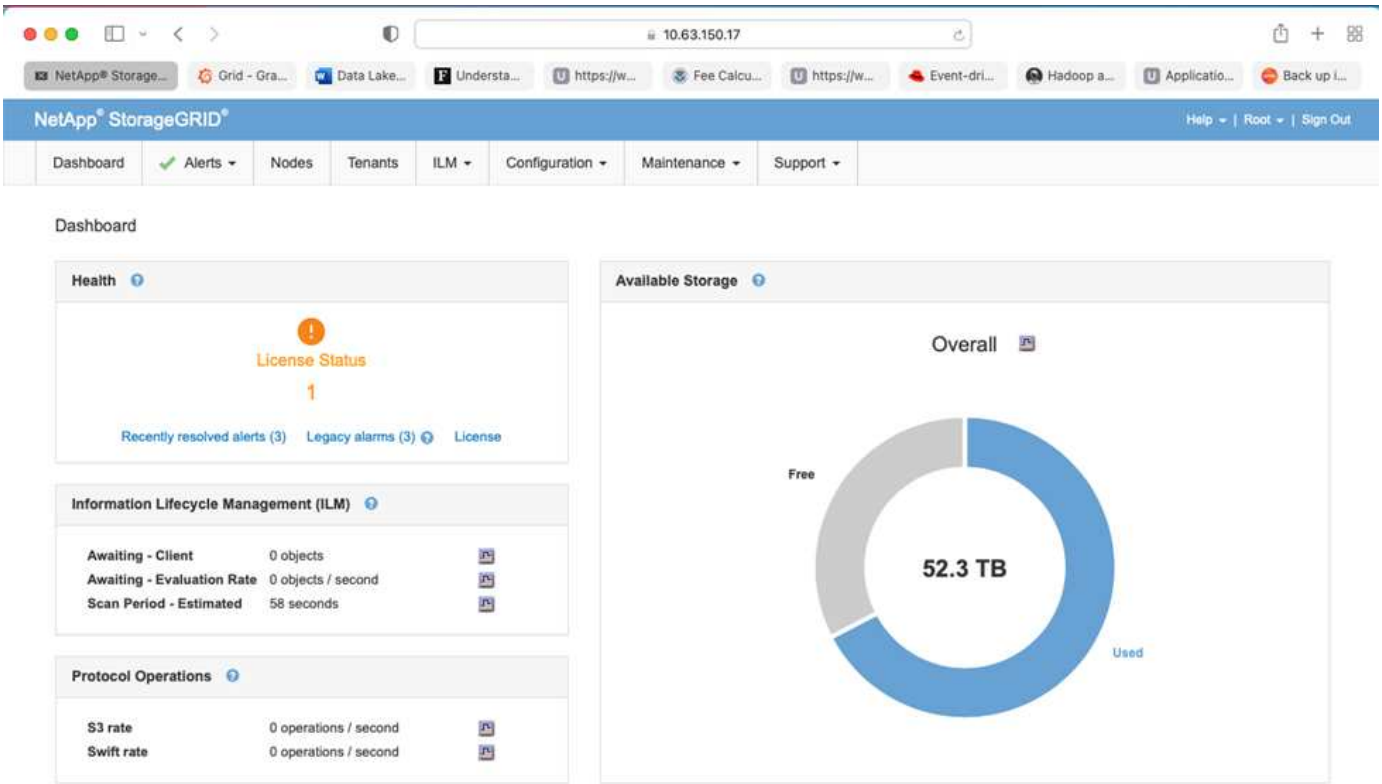
NetApp StorageGRID è una piattaforma di storage a oggetti dalle performance elevate e conveniente. Utilizzando lo storage a più livelli, la maggior parte dei dati su Confluent Kafka, che sono memorizzati nello storage locale o NELLO storage SAN del broker, viene scaricata nell'archivio a oggetti remoto. Questa configurazione comporta significativi miglioramenti operativi riducendo i tempi e i costi per ribilanciare, espandere o ridurre i cluster o sostituire un broker guasto. Lo storage a oggetti svolge un ruolo importante nella gestione dei dati che risiedono nel Tier dell'archivio di oggetti, motivo per cui è importante scegliere lo storage a oggetti giusto.

StorageGRID offre una gestione dei dati globale intelligente e basata su policy utilizzando un'architettura grid distribuita e basata su nodi. Semplifica la gestione di petabyte di dati non strutturati e miliardi di oggetti attraverso il suo onnipresente namespace globale a oggetti combinato con sofisticate funzionalità di gestione dei dati. L'accesso a oggetti a chiamata singola si estende tra i siti e semplifica le architetture ad alta disponibilità garantendo al contempo un accesso continuo agli oggetti, indipendentemente dalle interruzioni del sito o dell'infrastruttura.

La multi-tenancy consente di gestire in modo sicuro più cloud non strutturati e applicazioni dati aziendali all'interno dello stesso grid, aumentando il ROI e i casi di utilizzo per NetApp StorageGRID. Puoi creare diversi livelli di servizio con policy sul ciclo di vita degli oggetti basate sui metadati, ottimizzando durata, protezione, performance e località in più aree geografiche. Gli utenti possono regolare le policy di gestione dei dati, monitorare e applicare i limiti di traffico per riallinearsi con il panorama dei dati senza interruzioni in base al cambiamento dei requisiti in ambienti IT in continua evoluzione.

### Gestione semplice con Grid Manager

StorageGRID Grid Manager è un'interfaccia grafica basata su browser che consente di configurare, gestire e monitorare il sistema StorageGRID in ubicazioni distribuite a livello globale in un unico pannello di controllo.




Con l'interfaccia Gestore griglia di StorageGRID è possibile eseguire le seguenti attività:

- Gestisci repository distribuiti a livello globale e scalabili in petabyte di oggetti come immagini, video e record.
- Monitorare i nodi e i servizi di grid per garantire la disponibilità degli oggetti.
- Gestire il posizionamento dei dati a oggetti nel tempo utilizzando le regole ILM (Information Lifecycle Management). Queste regole regolano ciò che accade ai dati di un oggetto dopo l'acquisizione, il modo in cui sono protetti dalla perdita, dove sono memorizzati i dati dell'oggetto e per quanto tempo.
- Monitorare transazioni, performance e operazioni all'interno del sistema.

### Policy di Information Lifecycle Management

StorageGRID offre policy di gestione dei dati flessibili che includono la conservazione delle copie di replica degli oggetti e l'utilizzo di schemi EC (erasure coding) come 2+1 e 4+2 (tra gli altri) per memorizzare gli oggetti, a seconda dei requisiti specifici di performance e protezione dei dati. Con il variare dei carichi di lavoro e dei requisiti nel tempo, è comune che anche le policy ILM debbano cambiare nel tempo. La modifica delle policy ILM è una funzionalità fondamentale che consente ai clienti StorageGRID di adattarsi al loro ambiente in continua evoluzione in modo rapido e semplice. Controllare "Policy ILM" e. "Regole ILM" configurazione in StorageGRID.

### Performance

StorageGRID scala le performance aggiungendo più nodi di storage, che possono essere macchine virtuali, bare metal o appliance appositamente costruite come "SG5712, SG5760, SG6060 O SGF6024". Nei nostri test, abbiamo superato i requisiti di performance chiave di Apache Kafka con un grid a tre nodi di dimensioni minime utilizzando l'appliance SGF6024. Man mano che i clienti scalano il cluster Kafka con broker aggiuntivi, possono aggiungere più nodi di storage per aumentare performance e capacità.

### Bilanciamento del carico e configurazione degli endpoint

I nodi di amministrazione in StorageGRID forniscono l'interfaccia utente (interfaccia utente) e l'endpoint REST API per visualizzare, configurare e gestire il sistema StorageGRID, nonché registri di controllo per tenere traccia dell'attività del sistema. Per fornire un endpoint S3 altamente disponibile per lo storage a più livelli Confluent Kafka, abbiamo implementato il bilanciamento del carico StorageGRID, che viene eseguito come servizio su nodi di amministrazione e nodi gateway. Inoltre, il bilanciamento del carico gestisce anche il traffico locale e comunica con GSLB (Global Server Load Balancing) per agevolare il disaster recovery.

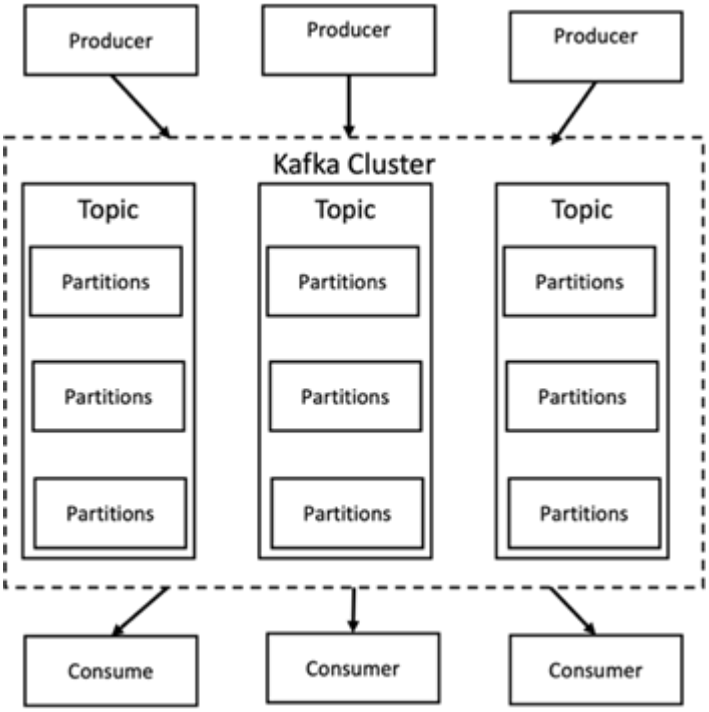
Per migliorare ulteriormente la configurazione degli endpoint, StorageGRID fornisce policy di classificazione del traffico integrate nel nodo di amministrazione, consente di monitorare il traffico dei workload e applica vari limiti di qualità del servizio (QoS) ai carichi di lavoro. I criteri di classificazione del traffico vengono applicati agli endpoint del servizio bilanciamento del carico StorageGRID per i nodi gateway e i nodi di amministrazione. Queste policy possono essere utili per la definizione e il monitoraggio del traffico.

### Classificazione del traffico in StorageGRID

StorageGRID dispone di funzionalità QoS integrate. I criteri di classificazione del traffico possono aiutare a monitorare diversi tipi di traffico S3 provenienti da un'applicazione client. È quindi possibile creare e applicare policy per limitare il traffico in base alla larghezza di banda in/out, al numero di richieste simultanee in lettura/scrittura o alla velocità di richiesta in lettura/scrittura.

## Apache Kafka

Apache Kafka è un'implementazione framework di un bus software che utilizza l'elaborazione del flusso scritta in Java e Scala. Il suo scopo è fornire una piattaforma unificata, ad alto throughput e a bassa latenza per la gestione dei feed di dati in tempo reale. Kafka può connettersi a un sistema esterno per l'esportazione e l'importazione dei dati tramite Kafka Connect e fornisce Kafka Streams, una libreria di elaborazione del flusso Java. Kafka utilizza un protocollo binario basato su TCP ottimizzato per l'efficienza e basato su un'astrazione "message set" che raggruppa naturalmente i messaggi per ridurre l'overhead del roundtrip di rete. Ciò consente operazioni su disco sequenziali più grandi, pacchetti di rete più grandi e blocchi di memoria contigui, consentendo a Kafka di trasformare un flusso bursty di scritture casuali di messaggi in scritture lineari. La figura seguente mostra il flusso di dati di base di Apache Kafka.



Kafka memorizza i messaggi chiave-valore che provengono da un numero arbitrario di processi chiamati produttori. I dati possono essere suddivisi in partizioni diverse all'interno di diversi argomenti. All'interno di una partizione, i messaggi vengono ordinati in base agli offset (la posizione di un messaggio all'interno di una partizione) e indicizzati e memorizzati insieme a un indicatore data e ora. Altri processi denominati consumer possono leggere i messaggi dalle partizioni. Per l'elaborazione dei flussi, Kafka offre l'API Streams che consente di scrivere applicazioni Java che consumano dati da Kafka e di riscrivere i risultati a Kafka. Apache Kafka funziona anche con sistemi di elaborazione del flusso esterni come Apache Apex, Apache Flink, Apache Spark, Apache Storm e Apache NiFi.

Kafka viene eseguito su un cluster di uno o più server (chiamati broker) e le partizioni di tutti gli argomenti vengono distribuite tra i nodi del cluster. Inoltre, le partizioni vengono replicate su più broker. Questa architettura consente a Kafka di inviare flussi di messaggi enormi in modo fault-tolerant e ha consentito al reparto IT di sostituire alcuni dei sistemi di messaggistica convenzionali come Java message Service (JMS), Advanced message Queuing Protocol (AMQP) e così via. A partire dalla release 0.11.0.0, Kafka offre scritture transazionali, che forniscono un'elaborazione del flusso esattamente una volta usando l'API Streams.

Kafka supporta due tipi di argomenti: Regolare e compatto. Gli argomenti regolari possono essere configurati con un tempo di conservazione o un limite di spazio. Se ci sono record più vecchi del tempo di conservazione specificato o se viene superato il limite di spazio per una partizione, Kafka può eliminare i vecchi dati per liberare spazio di storage. Per impostazione predefinita, gli argomenti sono configurati con un tempo di conservazione di 7 giorni, ma è anche possibile memorizzare i dati a tempo indeterminato. Per gli argomenti compattati, i record non scadono in base ai limiti di tempo o spazio. Kafka considera invece i messaggi successivi come aggiornamenti dei messaggi meno recenti con la stessa chiave e garantisce di non eliminare mai l'ultimo messaggio per chiave. Gli utenti possono eliminare completamente i messaggi scrivendo un cosiddetto messaggio tombstone con il valore null per una chiave specifica.

Kafka offre cinque API principali:

- **Producer API.** consente a un'applicazione di pubblicare flussi di record.
- **API Consumer.** consente a un'applicazione di iscriversi ad argomenti ed elaborare flussi di record.
- **Connector API.** esegue le API riutilizzabili di produttori e consumatori che possono collegare gli argomenti alle applicazioni esistenti.
- **Streams API.** questa API converte i flussi di input in output e produce il risultato.
- **Admin API.** utilizzato per gestire argomenti Kafka, broker e altri oggetti Kafka.

Le API consumer e Producer si basano sul protocollo di messaggistica Kafka e offrono un'implementazione di riferimento per i clienti consumer e Producer Kafka in Java. Il protocollo di messaging sottostante è un protocollo binario che gli sviluppatori possono utilizzare per scrivere i propri client consumer o Producer in qualsiasi linguaggio di programmazione. In questo modo, Kafka viene sbloccato dall'ecosistema JVM (Java Virtual Machine). Un elenco di client non Java disponibili viene mantenuto nel wiki Apache Kafka.

### Casi di utilizzo di Apache Kafka

Apache Kafka è più popolare per la messaggistica, il monitoraggio delle attività dei siti Web, le metriche, l'aggregazione dei log, l'elaborazione dei flussi, sourcing degli eventi e registrazione del commit.

- Kafka ha migliorato il throughput, il partizionamento integrato, la replica e la tolleranza agli errori, il che lo rende una buona soluzione per le applicazioni di elaborazione dei messaggi su larga scala.
- Kafka può ricostruire le attività di un utente (visualizzazioni di pagine, ricerche) in una pipeline di monitoraggio come un insieme di feed di iscrizione alla pubblicazione in tempo reale.
- Kafka viene spesso utilizzato per il monitoraggio dei dati operativi. Ciò comporta l'aggregazione di statistiche da applicazioni distribuite per produrre feed centralizzati di dati operativi.

- Molte persone utilizzano Kafka come sostituto di una soluzione di aggregazione dei log. L'aggregazione dei log generalmente raccoglie i file di log fisici dai server e li colloca in una posizione centrale (ad esempio, un file server o HDFS) per l'elaborazione. Kafka astratta i dettagli dei file e fornisce un'astrazione più pulita dei dati di log o degli eventi come flusso di messaggi. Ciò consente un'elaborazione a latenza ridotta e un supporto più semplice per più origini dati e un consumo di dati distribuito.
- Molti utenti di Kafka elaborano i dati in pipeline di elaborazione costituite da più fasi, in cui i dati di input raw vengono utilizzati da argomenti di Kafka e quindi aggregati, arricchiti o altrimenti trasformati in nuovi argomenti per un ulteriore consumo o un'elaborazione di follow-up. Ad esempio, una pipeline di elaborazione per consigliare articoli di notizie potrebbe strisciare il contenuto degli articoli dai feed RSS e pubblicarlo in un argomento "articoli". Un'ulteriore elaborazione potrebbe normalizzare o deduplicare questo contenuto e pubblicare il contenuto pulito dell'articolo su un nuovo argomento, mentre una fase finale di elaborazione potrebbe tentare di consigliare questo contenuto agli utenti. Tali pipeline di elaborazione creano grafici dei flussi di dati in tempo reale in base ai singoli argomenti.
- L'origine degli eventi è uno stile di progettazione dell'applicazione per cui le modifiche di stato vengono registrate come una sequenza di record ordinata in base al tempo. Il supporto di Kafka per i dati di log memorizzati di grandi dimensioni lo rende un eccellente backend per un'applicazione costruita in questo stile.
- Kafka può fungere da commit-log esterno per un sistema distribuito. Il log consente di replicare i dati tra i nodi e funge da meccanismo di risyncing per i nodi guasti per il ripristino dei dati. La funzione di compattazione del log di Kafka aiuta a supportare questo caso d'utilizzo.

## Confluente

Confluent Platform è una piattaforma Enterprise-ready che completa Kafka con funzionalità avanzate progettate per accelerare lo sviluppo e la connettività delle applicazioni, consentire trasformazioni attraverso l'elaborazione del flusso, semplificare le operazioni aziendali su larga scala e soddisfare rigorosi requisiti architetturali. Creato dai creatori originali di Apache Kafka, Confluent amplia i vantaggi di Kafka con funzionalità di livello Enterprise, eliminando al contempo il peso della gestione o del monitoraggio di Kafka. Oggi, oltre il 80% delle aziende Fortune 100 è basato su tecnologia di streaming dei dati, e la maggior parte di esse utilizza Confluent.

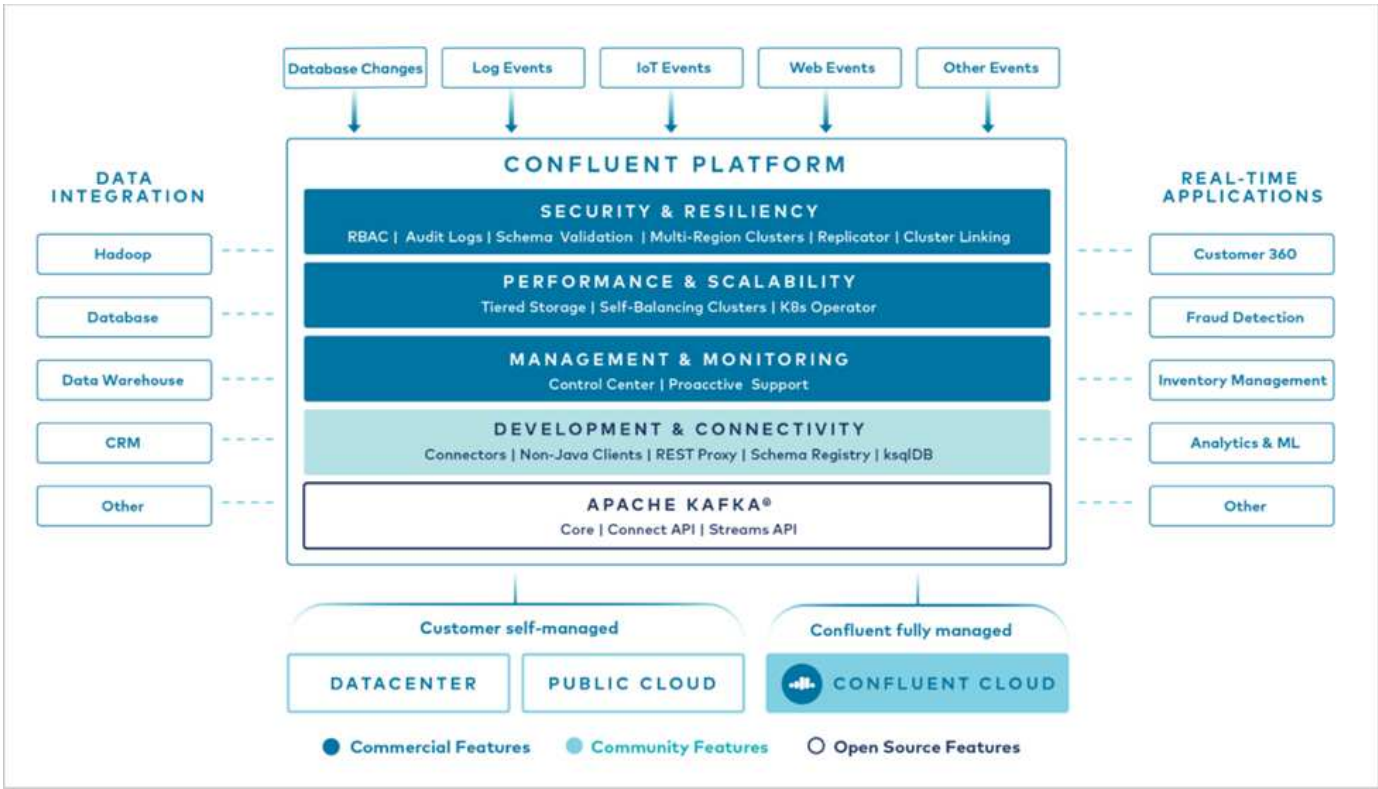
### Perché confluente?

Integrando dati storici e in tempo reale in un'unica fonte di verità centrale, Confluent semplifica la creazione di una categoria completamente nuova di applicazioni moderne e basate sugli eventi, l'acquisizione di una pipeline universale di dati e lo sblocco di nuovi casi di utilizzo potenti con scalabilità, performance e affidabilità complete.

### A cosa serve Confluent?

Confluent Platform ti consente di concentrarti su come ricavare il valore di business dai tuoi dati piuttosto che preoccuparsi delle meccaniche sottostanti, come ad esempio il modo in cui i dati vengono trasportati o integrati tra sistemi diversi. In particolare, Confluent Platform semplifica la connessione delle origini dati a Kafka, la creazione di applicazioni di streaming e la protezione, il monitoraggio e la gestione dell'infrastruttura Kafka. Attualmente, Confluent Platform viene utilizzata per un'ampia gamma di casi di utilizzo in numerosi settori, dai servizi finanziari alla vendita al dettaglio, alle auto autonome, al rilevamento delle frodi, Microservizi e IoT.

La figura seguente mostra i componenti della piattaforma Confluent Kafka.




### Panoramica della tecnologia di streaming degli eventi di Confluent

Il fulcro della piattaforma confluente è "Apache Kafka", la piattaforma di streaming distribuito open-source più diffusa. Le principali funzionalità di Kafka sono le seguenti:

- Pubblicare e sottoscrivere flussi di record.
- Memorizzare i flussi di record in modo tollerante agli errori.
- Elaborazione di flussi di record.

Confluent Platform include anche il Registro di sistema dello schema, il proxy REST, oltre 100 connettori Kafka preintegrati e ksqlDB.

### Panoramica delle funzionalità aziendali della piattaforma Confluent

- **Confluent Control Center.** sistema basato su GUI per la gestione e il monitoraggio di Kafka. Consente di gestire facilmente Kafka Connect e creare, modificare e gestire le connessioni ad altri sistemi.
- **Confluent per Kubernetes.** Confluent per Kubernetes è un operatore di Kubernetes. Gli operatori di Kubernetes estendono le funzionalità di orchestrazione di Kubernetes fornendo funzionalità e requisiti unici per una specifica applicazione della piattaforma. Per Confluent Platform, ciò include una notevole semplificazione del processo di implementazione di Kafka su Kubernetes e l'automazione delle attività tipiche del ciclo di vita dell'infrastruttura.
- **Connettori confluenti verso Kafka.** i connettori utilizzano l'API Kafka Connect per connettere Kafka ad altri sistemi come database, archivi di valori chiave, indici di ricerca e file system. Confluent Hub dispone di connettori scaricabili per le fonti di dati e i sink più diffusi, incluse le versioni completamente testate e supportate di questi connettori con Confluent Platform. Ulteriori dettagli sono disponibili "qui".
- **Cluster con bilanciamento automatico.** offre bilanciamento del carico automatico, rilevamento degli errori e riparazione automatica. Fornisce supporto per l'aggiunta o la disattivazione di broker in base alle necessità, senza tuning manuale.

- **Collegamento di cluster confluente.** collega direttamente i cluster e esegue il mirroring degli argomenti da un cluster all'altro tramite un bridge di collegamento. Il collegamento dei cluster semplifica la configurazione di implementazioni di cloud ibrido, multi-cluster e multi-data center.
- **Confluent auto data balancer.** monitora il cluster per il numero di broker, la dimensione delle partizioni, il numero di partizioni e il numero di leader all'interno del cluster. Consente di spostare i dati per creare un carico di lavoro uniforme nel cluster, riducendo al contempo il ribilanciamento del traffico per ridurre al minimo l'effetto sui carichi di lavoro di produzione durante il ribilanciamento.
- **Confluent Replicator.** semplifica la gestione di più cluster Kafka in più data center.
- **Tiered storage.** offre opzioni per l'archiviazione di grandi volumi di dati Kafka utilizzando il tuo cloud provider preferito, riducendo così il carico operativo e i costi. Con lo storage a più livelli, puoi mantenere i dati su uno storage a oggetti conveniente e scalare i broker solo quando hai bisogno di più risorse di calcolo.
- **Confluent JMS client.** Confluent Platform include un client compatibile con JMS per Kafka. Questo client Kafka implementa l'API standard JMS 1.1, utilizzando i broker Kafka come backend. Questo è utile se si utilizzano applicazioni legacy con JMS e si desidera sostituire il message broker JMS esistente con Kafka.
- **Il proxy MQTT confluente.** offre un modo per pubblicare i dati direttamente su Kafka da dispositivi e gateway MQTT senza la necessità di un broker MQTT al centro.
- **I plug-in di sicurezza confluenti.** i plug-in di sicurezza confluenti vengono utilizzati per aggiungere funzionalità di sicurezza a vari strumenti e prodotti della piattaforma confluente. Attualmente, è disponibile un plug-in per il proxy REST confluente che consente di autenticare le richieste in entrata e propagare l'identità autenticata alle richieste a Kafka. Ciò consente ai client proxy REST confluenti di utilizzare le funzionalità di sicurezza multi-tenant del broker Kafka.

# Verifica confluente

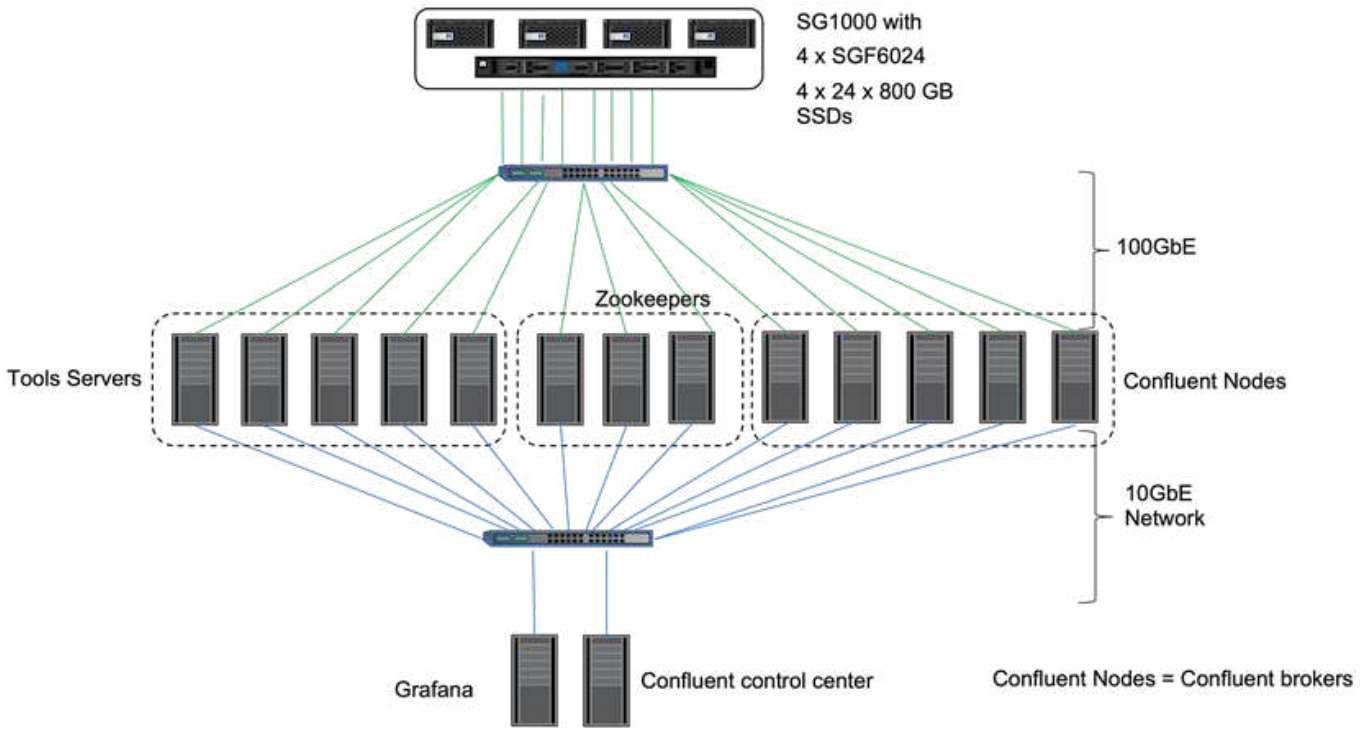
Abbiamo eseguito la verifica con la piattaforma confluente 6.2 Tiered Storage in NetApp StorageGRID. I team NetApp e Confluent hanno lavorato insieme a questa verifica ed hanno eseguito i casi di test richiesti per la verifica.

## Configurazione della piattaforma confluente

Per la verifica abbiamo utilizzato la seguente configurazione.

Per la verifica, abbiamo utilizzato tre zookeeper, cinque broker, cinque server di esecuzione script di test, server named tools con 256 GB di RAM e 16 CPU. Per lo storage NetApp, abbiamo utilizzato StorageGRID con un bilanciamento del carico SG1000 con quattro SGF6024. Lo storage e i broker erano connessi tramite connessioni 100GbE.

La figura seguente mostra la topologia di rete della configurazione utilizzata per la verifica confluente.




I server degli strumenti fungono da client applicativi che inviano richieste ai nodi confluenti.

## Configurazione dello storage a più livelli confluente

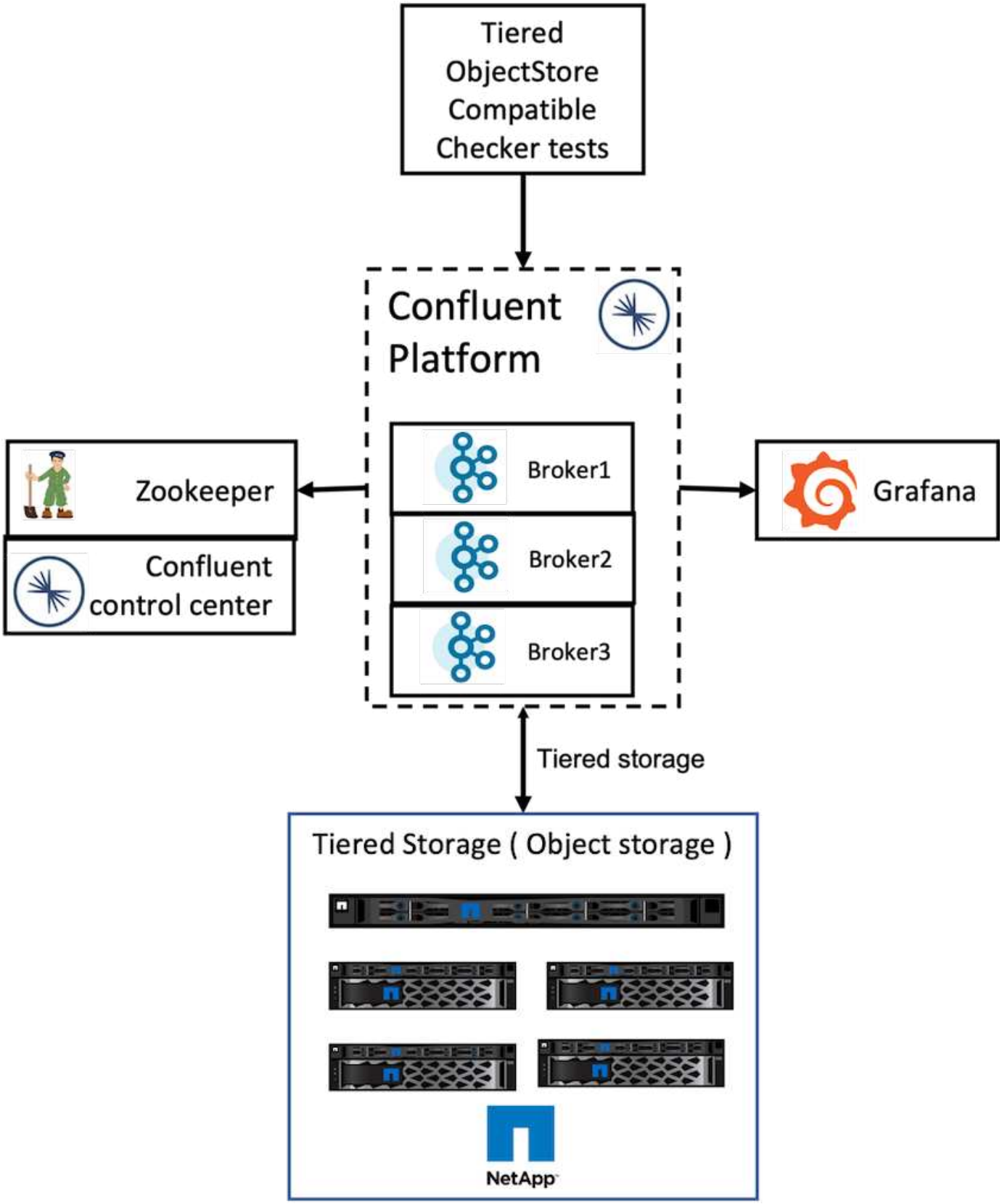
La configurazione dello storage a più livelli richiede i seguenti parametri in Kafka:

```
Confluent.tier.archiver.num.threads=16
confluent.tier.fetcher.num.threads=32
confluent.tier.enable=true
confluent.tier.feature=true
confluent.tier.backend=S3
confluent.tier.s3.bucket=kafkasgdbucket1-2
confluent.tier.s3.region=us-west-2
confluent.tier.s3.cred.file.path=/data/kafka/.ssh/credentials
confluent.tier.s3.aws.endpoint.override=http://kafkasgd.rtpppe.netapp.com:
10444/
confluent.tier.s3.force.path.style.access=true
```

Per la verifica, abbiamo utilizzato StorageGRID con il protocollo HTTP, ma funziona anche HTTPS. La chiave di accesso e la chiave segreta vengono memorizzate nel nome file fornito in `confluent.tier.s3.cred.file.path` parametro.

## Storage a oggetti NetApp - StorageGRID

Abbiamo configurato la configurazione a sito singolo in StorageGRID per la verificazione.




## Test di verifica

Abbiamo completato i seguenti cinque casi di test per la verifica. Questi test vengono eseguiti sul framework Trogdor. I primi due erano test di funzionalità e i restanti tre erano test di performance.

### Test di correttezza dell'archivio di oggetti

Questo test determina se tutte le operazioni di base (ad esempio, GET/put/delete) sull'API dell'archivio di oggetti funzionano correttamente in base alle esigenze dello storage su più livelli. Si tratta di un test di base che ogni servizio dell'archivio di oggetti dovrebbe superare prima dei seguenti test. Si tratta di un test assertivo che può essere superato o non superato.

### Test di correttezza delle funzionalità di tiering

Questo test determina se la funzionalità di storage a più livelli end-to-end funziona correttamente con un test assertivo che supera o non supera. Il test crea un argomento di test che per impostazione predefinita è configurato con il tiering attivato e una dimensione hotset altamente ridotta. Produce un flusso di eventi per l'argomento di test appena creato, attende che i broker archivino i segmenti nell'archivio di oggetti, quindi consuma il flusso di eventi e convalida che il flusso consumato corrisponda al flusso prodotto. Il numero di messaggi prodotti per il flusso di eventi è configurabile, il che consente all'utente di generare un carico di lavoro sufficientemente grande in base alle esigenze di test. La dimensione ridotta degli hotset garantisce che i fetch consumer al di fuori del segmento attivo vengano serviti solo dall'archivio di oggetti; questo aiuta a verificare la correttezza dell'archivio di oggetti per le letture. Questo test è stato eseguito con e senza un'iniezione di errori dello store di oggetti. Abbiamo simulato il guasto del nodo arrestando il servizio di gestione dei servizi in uno dei nodi in StorageGRID e convalidando che la funzionalità end-to-end funziona con lo storage a oggetti.

### Benchmark Tier fetch

Questo test ha validato le prestazioni di lettura dello storage a più livelli e verificato l'intervallo di richieste di lettura di recupero sotto carico pesante dai segmenti generati dal benchmark. In questo benchmark, Confluent ha sviluppato client personalizzati per soddisfare le richieste di recupero del Tier.

### Benchmark sui carichi di lavoro producete-consumate

Questo test ha generato indirettamente il carico di lavoro di scrittura nell'archivio di oggetti attraverso l'archiviazione dei segmenti. Il carico di lavoro di lettura (segmenti letti) è stato generato dallo storage a oggetti quando i gruppi di consumatori hanno recuperato i segmenti. Questo carico di lavoro è stato generato dallo script di test. Questo test ha verificato le prestazioni di lettura e scrittura sullo storage a oggetti in thread paralleli. Abbiamo eseguito test con e senza l'iniezione di errori del negozio di oggetti come abbiamo fatto per il test di correttezza della funzionalità di tiering.

### Benchmark del carico di lavoro di conservazione

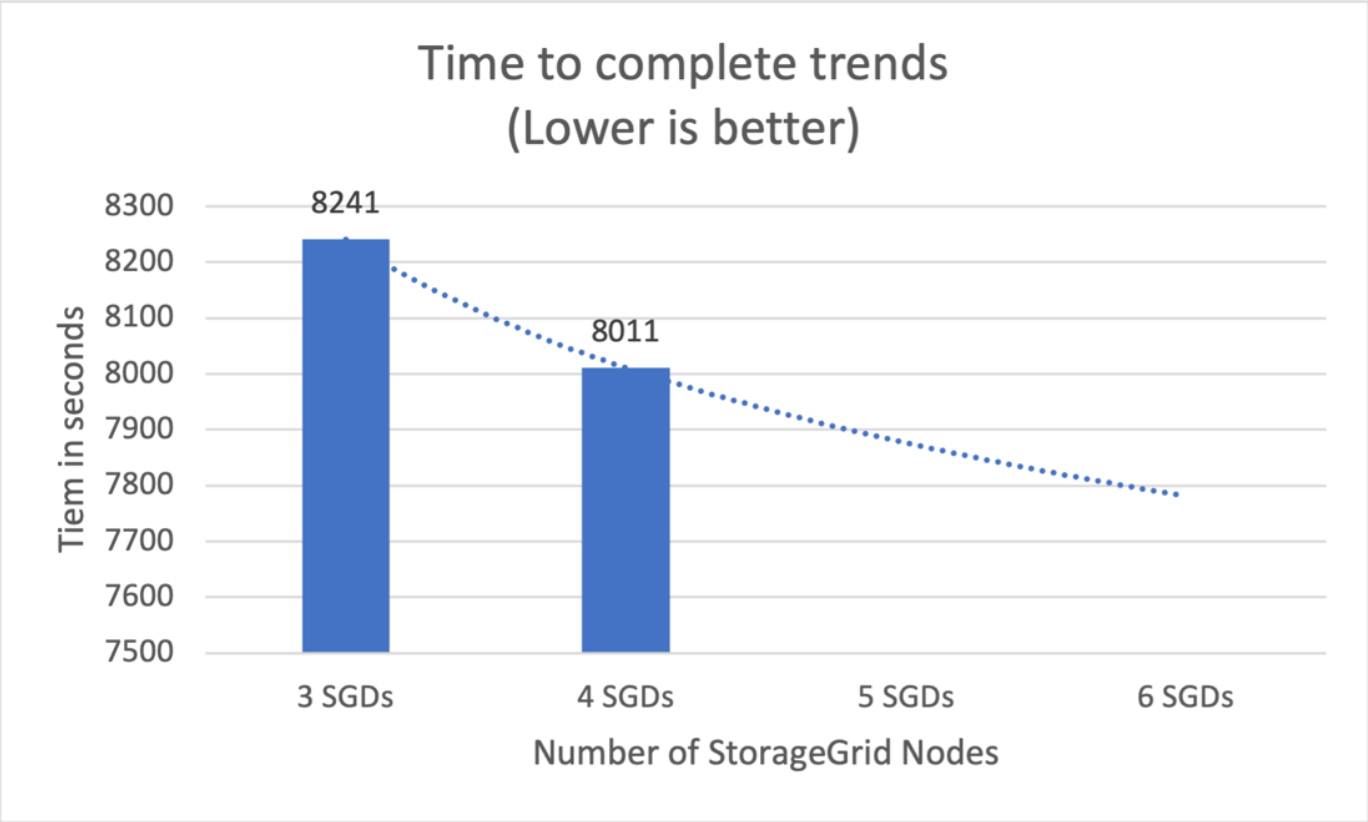
Questo test ha controllato le prestazioni di eliminazione di un archivio di oggetti con un carico di lavoro pesante di conservazione degli argomenti. Il carico di lavoro di conservazione è stato generato utilizzando uno script di test che produce molti messaggi in parallelo a un argomento di test. L'argomento del test è stato configurato con un'impostazione di conservazione aggressiva basata sulle dimensioni e sul tempo che ha causato la rimozione continua del flusso di eventi dall'archivio di oggetti. I segmenti sono stati quindi archiviati. Ciò ha portato a un gran numero di eliminazioni nello storage a oggetti da parte del broker e alla raccolta delle performance delle operazioni di eliminazione degli archivi di oggetti.

## Test delle performance con scalabilità

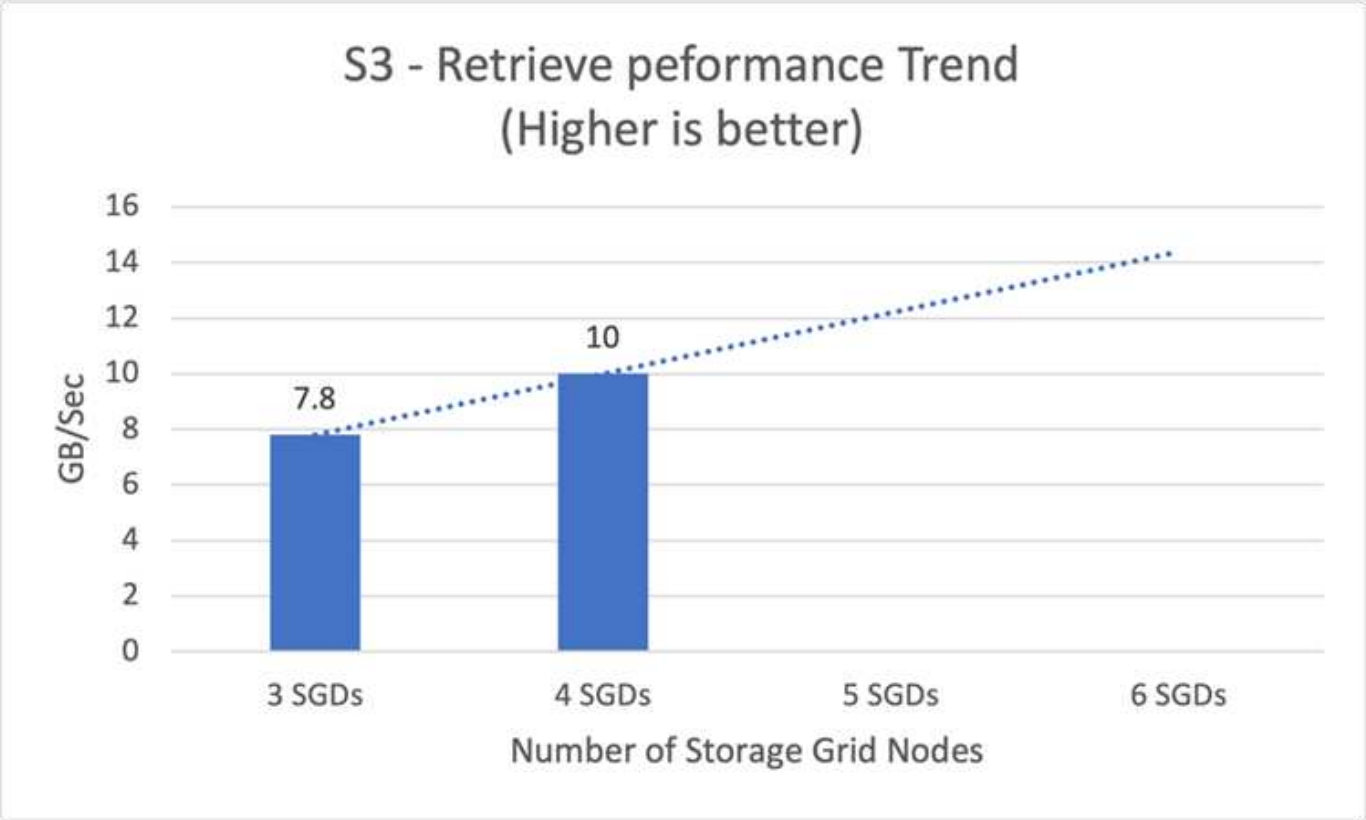
Con la configurazione di NetApp StorageGRID, abbiamo eseguito il test dello storage su più livelli con da tre a quattro nodi per carichi di lavoro consumer e di produttori. Secondo i nostri test, il tempo di completamento e i risultati delle performance erano direttamente proporzionali al numero di nodi StorageGRID. L'installazione di StorageGRID richiedeva

almeno tre nodi.

- Il tempo necessario per completare le operazioni di produzione e di consumo è diminuito in modo lineare con l'aumento del numero di nodi di storage.




- Le performance per l'operazione di recupero s3 sono aumentate linearmente in base al numero di nodi StorageGRID. StorageGRID supporta fino a 200 nodi StorgeGRID.




## Connettore s3 confluente

Amazon S3 Sink Connector esporta i dati dagli argomenti di Apache Kafka in oggetti S3 nei formati Avro, JSON o Bytes. Amazon S3 sink Connector esegue periodicamente il polling dei dati da Kafka e a sua volta li carica in S3. Un partitioner viene utilizzato per suddividere i dati di ogni partizione Kafka in blocchi. Ogni blocco di dati viene rappresentato come un oggetto S3. Il nome della chiave codifica l'argomento, la partizione Kafka e l'offset iniziale di questo blocco di dati.

In questa configurazione, viene illustrato come leggere e scrivere argomenti nello storage a oggetti da Kafka direttamente utilizzando il connettore del sink Kafka s3. Per questo test, abbiamo utilizzato un cluster Confluent autonomo, ma questa configurazione è applicabile a un cluster distribuito.

1. Scarica Confluent Kafka dal sito Web di Confluent.
2. Disimballare il pacchetto in una cartella sul server.
3. Esportare due variabili.

```
Export CONFLUENT_HOME=/data/confluent/confluent-6.2.0
export PATH=$PATH:/data/confluent/confluent-6.2.0/bin
```

4. Per un'installazione autonoma di Confluent Kafka, il cluster crea una cartella root temporanea in `/tmp`. Inoltre, crea Zoosekeeper, Kafka, un registro dello schema, Connect, un server ksql, e control center da cui copiare i rispettivi file di configurazione `$CONFLUENT_HOME`. Vedere il seguente esempio:

```
root@stlrx2540m1-108:~# ls -ltr /tmp/confluent.406980/
total 28
drwxr-xr-x 4 root root 4096 Oct 29 19:01 zookeeper
drwxr-xr-x 4 root root 4096 Oct 29 19:37 kafka
drwxr-xr-x 4 root root 4096 Oct 29 19:40 schema-registry
drwxr-xr-x 4 root root 4096 Oct 29 19:45 kafka-rest
drwxr-xr-x 4 root root 4096 Oct 29 19:47 connect
drwxr-xr-x 4 root root 4096 Oct 29 19:48 ksql-server
drwxr-xr-x 4 root root 4096 Oct 29 19:53 control-center
root@stlrx2540m1-108:~#
```

5. Configurare Zoosekeeper. Non è necessario modificare nulla se si utilizzano i parametri predefiniti.

```
root@stlrx2540m1-108:~# cat
/tmp/confluent.406980/zookeeper/zookeeper.properties  | grep -iv ^#
dataDir=/tmp/confluent.406980/zookeeper/data
clientPort=2181
maxClientCnxns=0
admin.enableServer=false
tickTime=2000
initLimit=5
syncLimit=2
server.179=controlcenter:2888:3888
root@stlrx2540m1-108:~#
```

Nella configurazione precedente, è stato aggiornato il `server. xxx` proprietà. Per impostazione predefinita, sono necessari tre Zooseeper per la selezione dei leader Kafka.

6. Abbiamo creato un file myid in `/tmp/confluent.406980/zookeeper/data` Con un ID univoco:

```
root@stlrx2540m1-108:~# cat /tmp/confluent.406980/zookeeper/data/myid
179
root@stlrx2540m1-108:~#
```

Abbiamo utilizzato l'ultimo numero di indirizzi IP per il file myid. Abbiamo utilizzato i valori predefiniti per Kafka, CONNECT, Control-Center, Kafka, Kafka-REST, configurazioni del server ksql e del registro di sistema dello schema.

7. Avviare i servizi Kafka.

```
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin# confluent
local services  start
The local commands are intended for a single-node development
environment only,
NOT for production usage.

Using CONFLUENT_CURRENT: /tmp/confluent.406980
ZooKeeper is [UP]
Kafka is [UP]
Schema Registry is [UP]
Kafka REST is [UP]
Connect is [UP]
ksqlDB Server is [UP]
Control Center is [UP]
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin#
```

Per ciascuna configurazione è disponibile una cartella di log che consente di risolvere i problemi. In alcuni casi, l'avvio dei servizi richiede più tempo. Assicurarsi che tutti i servizi siano attivi e in esecuzione.

8. Installare Kafka Connect utilizzando `confluent-hub`.

```
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin# ./confluent-
hub install confluentinc/kafka-connect-s3:latest
The component can be installed in any of the following Confluent
Platform installations:
  1. /data/confluent/confluent-6.2.0 (based on $CONFLUENT_HOME)
  2. /data/confluent/confluent-6.2.0 (where this tool is installed)
Choose one of these to continue the installation (1-2): 1
Do you want to install this into /data/confluent/confluent-
6.2.0/share/confluent-hub-components? (yN) y

Component's license:
Confluent Community License
http://www.confluent.io/confluent-community-license
I agree to the software license agreement (yN) y
Downloading component Kafka Connect S3 10.0.3, provided by Confluent,
Inc. from Confluent Hub and installing into /data/confluent/confluent-
6.2.0/share/confluent-hub-components
Do you want to uninstall existing version 10.0.3? (yN) y
Detected Worker's configs:
  1. Standard: /data/confluent/confluent-6.2.0/etc/kafka/connect-
distributed.properties
  2. Standard: /data/confluent/confluent-6.2.0/etc/kafka/connect-
standalone.properties
  3. Standard: /data/confluent/confluent-6.2.0/etc/schema-
```

```
registry/connect-avro-distributed.properties
  4. Standard: /data/confluent/confluent-6.2.0/etc/schema-
registry/connect-avro-standalone.properties
  5. Based on CONFLUENT_CURRENT:
/tmp/confluent.406980/connect/connect.properties
  6. Used by Connect process with PID 15904:
/tmp/confluent.406980/connect/connect.properties
Do you want to update all detected configs? (yN) y
Adding installation directory to plugin path in the following files:
  /data/confluent/confluent-6.2.0/etc/kafka/connect-
distributed.properties
  /data/confluent/confluent-6.2.0/etc/kafka/connect-
standalone.properties
  /data/confluent/confluent-6.2.0/etc/schema-registry/connect-avro-
distributed.properties
  /data/confluent/confluent-6.2.0/etc/schema-registry/connect-avro-
standalone.properties
  /tmp/confluent.406980/connect/connect.properties
  /tmp/confluent.406980/connect/connect.properties

Completed
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin#
```

È inoltre possibile installare una versione specifica utilizzando `confluent-hub install confluentinc/kafka-connect-s3:10.0.3`.

9. Per impostazione predefinita, `confluentinc-kafka-connect-s3` è installato in `/data/confluent/confluent-6.2.0/share/confluent-hub-components/confluentinc-kafka-connect-s3`.
10. Aggiornare il percorso del plug-in con il nuovo `confluentinc-kafka-connect-s3`.

```
root@stlrx2540m1-108:~# cat /data/confluent/confluent-
6.2.0/etc/kafka/connect-distributed.properties | grep plugin.path
#
plugin.path=/usr/local/share/java,/usr/local/share/kafka/plugins,/opt/co
nnectors,
plugin.path=/usr/share/java,/data/zookeeper/confluent/confluent-
6.2.0/share/confluent-hub-components,/data/confluent/confluent-
6.2.0/share/confluent-hub-components,/data/confluent/confluent-
6.2.0/share/confluent-hub-components/confluentinc-kafka-connect-s3
root@stlrx2540m1-108:~#
```

11. Arrestare e riavviare i servizi confluenti.

```
confluent local services  stop
confluent local services  start
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin# confluent
local services  status
The local commands are intended for a single-node development
environment only,
NOT for production usage.

Using CONFLUENT_CURRENT: /tmp/confluent.406980
Connect is [UP]
Control Center is [UP]
Kafka is [UP]
Kafka REST is [UP]
ksqlDB Server is [UP]
Schema Registry is [UP]
ZooKeeper is [UP]
root@stlrx2540m1-108:/data/confluent/confluent-6.2.0/bin#
```

12. Configurare l'ID di accesso e la chiave segreta in `/root/.aws/credentials` file.

```
root@stlrx2540m1-108:~# cat /root/.aws/credentials
[default]
aws_access_key_id = xxxxxxxxxxxx
aws_secret_access_key = xxxxxxxxxxxxxxxxxxxxxxxxx
root@stlrx2540m1-108:~#
```

13. Verificare che il bucket sia raggiungibile.

```
root@stlrx2540m4-01:~# aws s3 –endpoint-url
http://kafkasgd.rtpppe.netapp.com:10444 ls kafkasgdbucket1-2
2021-10-29 21:04:18       1388 1
2021-10-29 21:04:20       1388 2
2021-10-29 21:04:22       1388 3
root@stlrx2540m4-01:~#
```

14. Configurare il file di proprietà s3-sink per la configurazione s3 e bucket.

```
root@stlrx2540m1-108:~# cat /data/confluent/confluent-
6.2.0/share/confluent-hub-components/confluentinc-kafka-connect-
s3/etc/quickstart-s3.properties | grep -v ^#
name=s3-sink
connector.class=io.confluent.connect.s3.S3SinkConnector
tasks.max=1
topics=s3_testtopic
s3.region=us-west-2
s3.bucket.name=kafkasgdbucket1-2
store.url=http://kafkasgd.rtpppe.netapp.com:10444/
s3.part.size=5242880
flush.size=3
storage.class=io.confluent.connect.s3.storage.S3Storage
format.class=io.confluent.connect.s3.format.avro.AvroFormat
partitioner.class=io.confluent.connect.storage.partitioner.DefaultPartit
ioner
schema.compatibility=NONE
root@stlrx2540m1-108:~#
```

15. Importare alcuni record nel bucket s3.

```
kafka-avro-console-producer --broker-list localhost:9092 --topic
s3_topic \
--property
value.schema='{"type":"record","name":"myrecord","fields":[{"name":"f1",
"type":"string"}]}'
{"f1": "value1"}
{"f1": "value2"}
{"f1": "value3"}
{"f1": "value4"}
{"f1": "value5"}
{"f1": "value6"}
{"f1": "value7"}
{"f1": "value8"}
{"f1": "value9"}
```

16. Caricare il connettore s3-sink.

```
root@stlrx2540m1-108:~# confluent local services connect connector load
s3-sink  --config /data/confluent/confluent-6.2.0/share/confluent-hub-
components/confluentinc-kafka-connect-s3/etc/quickstart-s3.properties
The local commands are intended for a single-node development
environment only,
NOT for production usage.
https://docs.confluent.io/current/cli/index.html
{
  "name": "s3-sink",
  "config": {
    "connector.class": "io.confluent.connect.s3.S3SinkConnector",
    "flush.size": "3",
    "format.class": "io.confluent.connect.s3.format.avro.AvroFormat",
    "partitioner.class":
"io.confluent.connect.storage.partitioner.DefaultPartitioner",
    "s3.bucket.name": "kafkasgdbucket1-2",
    "s3.part.size": "5242880",
    "s3.region": "us-west-2",
    "schema.compatibility": "NONE",
    "storage.class": "io.confluent.connect.s3.storage.S3Storage",
    "store.url": "http://kafkasgd.rtpppe.netapp.com:10444/",
    "tasks.max": "1",
    "topics": "s3_testtopic",
    "name": "s3-sink"
  },
  "tasks": [],
  "type": "sink"
}
root@stlrx2540m1-108:~#
```

17. Controllare lo stato del sink s3.

```
root@stlrx2540m1-108:~# confluent local services connect connector
status s3-sink
The local commands are intended for a single-node development
environment only,
NOT for production usage.
https://docs.confluent.io/current/cli/index.html
{
  "name": "s3-sink",
  "connector": {
    "state": "RUNNING",
    "worker_id": "10.63.150.185:8083"
  },
  "tasks": [
    {
      "id": 0,
      "state": "RUNNING",
      "worker_id": "10.63.150.185:8083"
    }
  ],
  "type": "sink"
}
root@stlrx2540m1-108:~#
```

18. Controllare il registro per assicurarsi che s3-sink sia pronto ad accettare gli argomenti.

```
root@stlrx2540m1-108:~# confluent local services connect log
```

19. Consulta gli argomenti di Kafka.

```
kafka-topics --list --bootstrap-server localhost:9092
…
connect-configs
connect-offsets
connect-statuses
default_ksql_processing_log
s3_testtopic
s3_topic
s3_topic_new
root@stlrx2540m1-108:~#
```

20. Controllare gli oggetti nel bucket s3.

```
root@stlrx2540m1-108:~# aws s3 --endpoint-url
http://kafkasgd.rtpppe.netapp.com:10444 ls --recursive kafkasgdbucket1-
2/topics/
2021-10-29 21:24:00        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000000.avro
2021-10-29 21:24:00        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000003.avro
2021-10-29 21:24:00        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000006.avro
2021-10-29 21:24:08        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000009.avro
2021-10-29 21:24:08        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000012.avro
2021-10-29 21:24:09        213
topics/s3_testtopic/partition=0/s3_testtopic+0+0000000015.avro
root@stlrx2540m1-108:~#
```

21. Per verificare il contenuto, copiare ciascun file da S3 al file system locale eseguendo il seguente comando:

```
root@stlrx2540m1-108:~# aws s3 --endpoint-url
http://kafkasgd.rtpppe.netapp.com:10444 cp s3://kafkasgdbucket1-
2/topics/s3_testtopic/partition=0/s3_testtopic+0+0000000000.avro
tes.avro
download: s3://kafkasgdbucket1-
2/topics/s3_testtopic/partition=0/s3_testtopic+0+0000000000.avro to
./tes.avro
root@stlrx2540m1-108:~#
```

22. Per stampare i record, utilizzare avro-tools-1.11.0.1.jar (disponibile in "Archivi Apache").

```
root@stlrx2540m1-108:~# java -jar /usr/src/avro-tools-1.11.0.1.jar
tojson tes.avro
21/10/30 00:20:24 WARN util.NativeCodeLoader: Unable to load native-
hadoop library for your platform... using builtin-java classes where
applicable
{"f1":"value1"}
{"f1":"value2"}
{"f1":"value3"}
root@stlrx2540m1-108:~#
```

# Clusters a bilanciamento automatico confluente

Se hai già gestito un cluster Kafka in precedenza, probabilmente conosci le sfide legate alla riassegnazione manuale delle partizioni a diversi broker per garantire che il carico di lavoro sia bilanciato in tutto il cluster. Per le organizzazioni con implementazioni Kafka di grandi dimensioni, rimescolare grandi quantità di dati può essere scoraggiante, noioso e rischioso, soprattutto se le applicazioni mission-critical sono costruite sul cluster. Tuttavia, anche per i più piccoli casi di utilizzo di Kafka, il processo richiede tempo e può essere soggetto a errori umani.

Nel nostro laboratorio, abbiamo testato la funzionalità dei cluster di bilanciamento automatico Confluent, che automatizza il ribilanciamento in base alle modifiche della topologia dei cluster o al carico non uniforme. Il test di ribilanciamento confluente consente di misurare il tempo necessario per aggiungere un nuovo broker quando un guasto al nodo o il nodo di scalabilità richiede il ribilanciamento dei dati tra broker. Nelle configurazioni Kafka classiche, la quantità di dati da ribilanciare aumenta con la crescita del cluster, ma, nello storage a più livelli, il ribilanciamento è limitato a una piccola quantità di dati. In base alla nostra convalida, il ribilanciamento dello storage a più livelli richiede secondi o minuti in una classica architettura Kafka e cresce linearmente con la crescita del cluster.

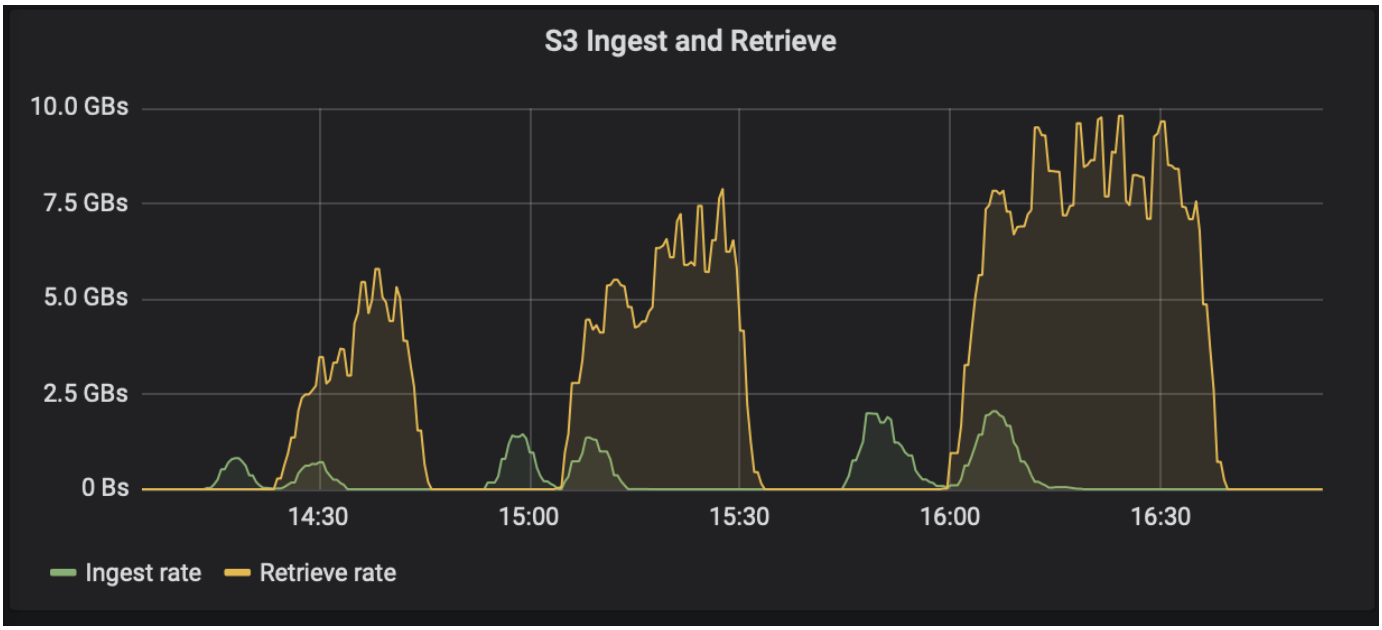
Nei cluster con bilanciamento automatico, i ribilanciamenti delle partizioni sono completamente automatizzati per ottimizzare il throughput di Kafka, accelerare la scalabilità dei broker e ridurre il carico operativo di un cluster di grandi dimensioni. A stato stazionario, i cluster con bilanciamento automatico monitorano l'inclinazione dei dati tra i broker e riassegnano continuamente le partizioni per ottimizzare le performance del cluster. Quando la piattaforma viene scalata verso l'alto o verso il basso, i cluster a bilanciamento automatico riconoscono automaticamente la presenza di nuovi broker o la rimozione di vecchi broker e attivano una successiva riassegnazione delle partizioni. In questo modo potrai aggiungere e decommissionare facilmente i broker, rendendo i cluster Kafka fondamentalmente più elastici. Questi vantaggi non richiedono interventi manuali, calcoli complessi o il rischio di errori umani che le riassegnazioni delle partizioni comportano in genere. Di conseguenza, i ribilanciamenti dei dati vengono completati in meno tempo e puoi concentrarti su progetti di streaming di eventi di valore superiore, invece di dover monitorare costantemente i tuoi cluster.

# Linee guida sulle Best practice

Questa sezione presenta le lezioni apprese da questa certificazione.

- In base alla nostra convalida, lo storage a oggetti S3 è la soluzione migliore per Confluent per conservare i dati.
- Possiamo utilizzare SAN ad alto throughput (in particolare FC) per mantenere i dati hot del broker o il disco locale, perché, nella configurazione dello storage a più livelli Confluent, la dimensione dei dati contenuti nella directory dei dati broker si basa sulle dimensioni del segmento e sul tempo di conservazione quando i dati vengono spostati nello storage a oggetti.
- Gli archivi di oggetti offrono performance migliori quando segment.bytes è più elevato; abbiamo testato 512 MB.
- In Kafka, la lunghezza della chiave o del valore (in byte) per ciascun record prodotto per l'argomento è controllata da `length.key.value` parametro. Per StorageGRID, le performance di acquisizione e recupero degli oggetti S3 sono aumentate a valori più elevati. Ad esempio, 512 byte hanno fornito un recupero da 5,8 Gbps, 1024 byte hanno fornito un recupero s3 da 7,5 Gbps e 2048 byte sono forniti quasi a 10 Gbps.

La figura seguente illustra l'acquisizione e il recupero dell'oggetto S3 in base a. `length.key.value`.




- **Tuning di Kafka.** per migliorare le performance dello storage a più livelli, è possibile aumentare TierFetcherNumThreads e TierArchiverNumThreads. Come linea guida generale, si desidera aumentare TierFetcherNumThreads in modo che corrisponda al numero di core della CPU fisica e aumentare TierArchiverNumThreads fino alla metà del numero di core della CPU. Ad esempio, nelle proprietà del server, se si dispone di un computer con otto core fisici, impostare confluent.Tier.fetcher.num.threads = 8 e confluent.Tier.archiver.num.threads = 4.
- **Intervallo di tempo per l'eliminazione dell'argomento.** quando un argomento viene cancellato, l'eliminazione dei file di segmenti di registro nella memoria a oggetti non inizia immediatamente. Piuttosto, esiste un intervallo di tempo con un valore predefinito di 3 ore prima che venga eseguita l'eliminazione di tali file. È possibile modificare la configurazione, confluent.tier.topic.delete.check.interval.ms, per modificare il valore di questo intervallo. Se si elimina un argomento o un cluster, è anche possibile eliminare manualmente gli oggetti nel rispettivo bucket.
- **ACL su argomenti interni dello storage a più livelli.** Una Best practice consigliata per le implementazioni on-premise è abilitare un autorizzatore ACL sugli argomenti interni utilizzati per lo storage a più livelli. Impostare le regole ACL per limitare l'accesso a questi dati solo all'utente del broker. In questo modo si proteggono gli argomenti interni e si impedisce l'accesso non autorizzato ai dati e ai metadati dello storage a più livelli.

```
kafka-acls --bootstrap-server localhost:9092 --command-config adminclient-
configs.conf \
--add --allow-principal User:<kafka> --operation All --topic "_confluent-
tier-state"
```

Sostituire l'utente `<kafka>` con l'effettivo broker principal nella tua implementazione.

Ad esempio, il comando `confluent-tier-state` Imposta gli ACL sull'argomento interno per lo storage a più livelli. Attualmente, esiste solo un singolo argomento interno relativo allo storage a più livelli. Nell'esempio viene creato un ACL che fornisce l'autorizzazione principale Kafka per tutte le operazioni sull'argomento interno.

# Dimensionamento

Il dimensionamento di Kafka può essere eseguito con quattro modalità di configurazione: Semplice, granulare, inversa e partizioni.

## Semplice

La modalità Simple è adatta per gli utenti Apache Kafka per la prima volta o per i primi casi di utilizzo. Per questa modalità, vengono forniti requisiti come throughput Mbps, fanout di lettura, conservazione e percentuale di utilizzo delle risorse (il valore predefinito è 60%). Si accede anche all'ambiente, ad esempio on-premise (bare-metal, VMware, Kubernetes o OpenStack) o al cloud. In base a queste informazioni, il dimensionamento di un cluster Kafka fornisce il numero di server richiesti per il broker, lo zookeeper, gli impiegati di connessione Apache Kafka, il registro dello schema, un proxy REST, ksqlDB e il centro di controllo Confluent.

Per lo storage su più livelli, prendere in considerazione la modalità di configurazione granulare per il dimensionamento di un cluster Kafka. La modalità granulare è adatta agli utenti esperti di Apache Kafka o a casi di utilizzo ben definiti. Questa sezione descrive il dimensionamento per produttori, processori di streaming e consumatori.

### Produttori

Per descrivere i produttori di Apache Kafka (ad esempio un client nativo, un proxy REST o un connettore Kafka), fornire le seguenti informazioni:

- **Nome.** Spark.
- **Tipo produttore.** applicazione o servizio, proxy (REST, MQTT, Altro) e database esistente (RDBMS, NOSQL, Altro). È anche possibile selezionare "non so".
- **Throughput medio.** in eventi al secondo (ad esempio 1,000,000).
- **Throughput massimo.** in eventi al secondo (ad esempio 4,000,000).
- **Dimensione media dei messaggi.** in byte, non compressi (massimo 1 MB; 1000 ad esempio).
- **Message format.** le opzioni includono Avro, JSON, buffer di protocollo, binario, testo, "Non lo so" e altri.
- **Fattore di replica.** le opzioni sono 1, 2, 3 (raccomandazione confluente), 4, 5, oppure 6.
- **Tempo di conservazione.** un giorno (ad esempio). Per quanto tempo vuoi che i tuoi dati siano memorizzati in Apache Kafka? Inserire -1 con qualsiasi unità per un tempo infinito. Il calcolatore presuppone un tempo di conservazione di 10 anni per una conservazione infinita.
- Selezionare la casella di controllo "Enable Tiered Storage to Decrease Broker Count and Allow for Infinite Storage?" (attiva lo storage a livelli per ridurre il numero di broker e consentire lo storage
- Quando lo storage su più livelli è attivato, i campi di conservazione controllano l'hot set di dati memorizzati localmente sul broker. I campi di conservazione dell'archivio controllano la durata della memorizzazione dei dati nello storage a oggetti di archiviazione.
- **Archival Storage Retention.** un anno (ad esempio). Per quanto tempo vuoi che i tuoi dati siano memorizzati nello storage di archiviazione? Inserire -1 con qualsiasi unità per una durata infinita. Il calcolatore presuppone una conservazione di 10 anni per una conservazione infinita.
- **Growth Multiplier.** 1 (ad esempio). Se il valore di questo parametro si basa sul throughput corrente, impostarlo su 1. Per dimensionare in base alla crescita aggiuntiva, impostare questo parametro su un moltiplicatore di crescita.
- **Numero di istanze produttore.** 10 (ad esempio). Quante istanze di produttori saranno in esecuzione?

Questo input è necessario per incorporare il carico della CPU nel calcolo del dimensionamento. Un valore vuoto indica che il carico della CPU non è incorporato nel calcolo.

Sulla base di questo esempio di input, il dimensionamento ha il seguente effetto sui produttori:

- Throughput medio in byte non compressi: 1 Gbps. Throughput massimo in byte non compressi: 4 Gbps. Throughput medio in byte compressi: 400 Mbps. Throughput massimo in byte compressi: 1,6 Gbps. Si basa su un tasso di compressione predefinito del 60% (è possibile modificare questo valore).
  - Storage hotset on-broker totale richiesto: 31,104 TB, inclusa la replica, compressa. Storage di archiviazione off-broker totale richiesto: 378,432 TB, compresso. Utilizzare "https://fusion.netapp.com" Per il dimensionamento StorageGRID.

I processori stream devono descrivere le proprie applicazioni o servizi che consumano dati da Apache Kafka e riproducono in Apache Kafka. Nella maggior parte dei casi, questi sono integrati in flussi ksqlDB o Kafka.

- **Nome.** Spark streamer.
- **Tempo di elaborazione.** quanto tempo impiega questo processore per elaborare un singolo messaggio?
  - 1 ms (semplice, stateless transformation) [esempio], 10 ms (stateful in-memory operation).
  - 100 ms (funzionamento su disco o rete stateful), 1000 ms (chiamata DI PAUSA di terze parti).
  - Ho eseguito un benchmark di questo parametro e so esattamente quanto tempo occorre.
- **Conservazione dell'output.** 1 giorno (esempio). Un stream processor produce l'output di Apache Kafka. Per quanto tempo vuoi che questi dati di output siano memorizzati in Apache Kafka? Inserire -1 con qualsiasi unità per una durata infinita.
- Selezionare la casella di controllo "Enable Tiered Storage to Decrease Broker Count and Allow for Infinite Storage?" (attiva lo storage a più livelli per ridurre il numero di broker e
- **Archival Storage Retention.** 1 anno (ad esempio). Per quanto tempo vuoi che i tuoi dati siano memorizzati nello storage di archiviazione? Inserire -1 con qualsiasi unità per una durata infinita. Il calcolatore presuppone una conservazione di 10 anni per una conservazione infinita.
- **Output Passthrough percent.** 100 (ad esempio). Un stream processor produce l'output di Apache Kafka. Quale percentuale di throughput in entrata verrà restituita ad Apache Kafka? Ad esempio, se il throughput in entrata è 20 Mbps e questo valore è 10, il throughput in uscita sarà 2 Mbps.
- Da quali applicazioni viene letto? Selezionare "Spark" (Spark), il nome utilizzato nel dimensionamento basato sul tipo di produttore. In base all'input di cui sopra, è possibile prevedere i seguenti effetti del dimensionamento sulle istanze del processo di flusso e sulle stime delle partizioni degli argomenti:
- Questa applicazione di elaborazione del flusso richiede il seguente numero di istanze. Gli argomenti in entrata probabilmente richiedono anche queste partizioni. Contattare Confluent per confermare questo parametro.
  - 1,000 per throughput medio senza moltiplicatore di crescita
  - 4,000 per throughput di picco senza moltiplicatore di crescita
  - 1,000 per un throughput medio con un moltiplicatore di crescita
  - 4,000 per throughput di picco con un moltiplicatore di crescita

**Consumatori**

Descrivi le tue applicazioni o servizi che consumano dati da Apache Kafka e non vengono prodotti in Apache Kafka, ad esempio un client nativo o un connettore Kafka.

- **Nome.** Spark consumer.

- **Tempo di elaborazione.** quanto tempo impiega questo cliente per elaborare un singolo messaggio?
  - 1 ms (ad esempio, un'attività semplice e senza stato come la registrazione)
  - 10 ms (scritture rapide in un datastore)
  - 100 ms (scritture lente in un datastore)
  - 1000 ms (chiamata DI RIPOSO di terze parti)
  - Alcuni altri processi di riferimento di durata nota.
- **Consumer type.** Application, proxy, or sink to a existing datastore (RDBMS, NoSQL, other).
- Da quali applicazioni viene letto? Collegare questo parametro al produttore e al dimensionamento del flusso determinati in precedenza.

In base all'input di cui sopra, è necessario determinare il dimensionamento per le istanze consumer e le stime delle partizioni degli argomenti. Un'applicazione consumer richiede il seguente numero di istanze.

- 2,000 per throughput medio, nessun moltiplicatore di crescita
- 8,000 per throughput di picco, nessun moltiplicatore di crescita
- 2,000 per throughput medio, incluso il moltiplicatore di crescita
- 8,000 per throughput di picco, incluso il moltiplicatore di crescita

Gli argomenti in entrata probabilmente necessitano anche di questo numero di partizioni. Contatta Confluent per confermare.

Oltre ai requisiti per i produttori, i processori di streaming e i consumatori, devi fornire i seguenti requisiti aggiuntivi:

- **Tempo di ricostruzione.** ad esempio, 4 ore. Se un host del broker Apache Kafka si guasta, i dati vengono persi e viene eseguito il provisioning di un nuovo host per sostituire l'host guasto, quanto velocemente deve essere ricostruito questo nuovo host? Lasciare vuoto questo parametro se il valore non è noto.
- **Obiettivo di utilizzo delle risorse (percentuale).** ad esempio, 60. In che modo desiderate che i vostri host siano utilizzati durante il throughput medio? Confluent consiglia un utilizzo del 60% a meno che non si utilizzino cluster di bilanciamento automatico Confluent, nel qual caso l'utilizzo può essere maggiore.

**Descrivi il tuo ambiente**

- **In quale ambiente verrà eseguito il tuo cluster?** Amazon Web Services, Microsoft Azure, piattaforma cloud Google, bare-metal on-premise, VMware on premise, OpenStack on premise o Kubernates on premise?
- **Dettagli host.** numero di core: 48 (ad esempio), tipo di scheda di rete (10GbE, 40GbE, 16GbE, 1GbE o altro tipo).
- **Storage Volumes.** host: 12 (ad esempio). Quanti dischi rigidi o SSD sono supportati per host? Confluent consiglia 12 dischi rigidi per host.
- **Capacità/volume di storage (in GB).** 1000 (ad esempio). Quanto storage può memorizzare un singolo volume in gigabyte? Confluent consiglia dischi da 1 TB.
- **Configurazione dello storage.** come vengono configurati i volumi dello storage? Confluent consiglia RAID10 per sfruttare tutte le funzionalità confluenti. JBOD, SAN, RAID 1, RAID 0, RAID 5, e altri tipi sono supportati.
- **Throughput di un volume singolo (Mbps).** 125 (ad esempio). Quanto velocemente un singolo volume di storage può leggere o scrivere in megabyte al secondo? Confluent consiglia dischi rigidi standard, che in

genere hanno un throughput di 125 MBps.

- **Capacità di memoria (GB).** 64 (ad esempio).

Dopo aver determinato le variabili ambientali, selezionare Size my Cluster (dimensione cluster). In base ai parametri di esempio sopra indicati, abbiamo determinato il seguente dimensionamento per Confluent Kafka:

- * Apache Kafka.* numero di broker: 22. Il cluster è legato allo storage. Considerare la possibilità di abilitare lo storage a più livelli per ridurre il numero di host e consentire lo storage infinito.
- **Apache Zooseeper.** Conteggio: 5; Apache Kafka Connect Workers: Conteggio: 2; Registro dello schema: Conteggio: 2; REST Proxy: Conteggio: 2; ksqlDB: Conteggio: 2; Centro di controllo confluente: Conteggio: 1.

Utilizza la modalità inversa per i team di piattaforme senza un caso d'utilizzo. Utilizzare la modalità Partitions per calcolare il numero di partizioni richiesto da un singolo argomento. Vedere https://eventsizer.io per il dimensionamento in base alle modalità di reverse e partitions.

# Conclusione

Questo documento fornisce le linee guida sulle Best practice per l'utilizzo dello storage a livelli confluente con lo storage NetApp, inclusi test di verifica, risultati delle performance dello storage a livelli, tuning, connettori Confluent S3 e funzionalità di bilanciamento automatico. Considerando le policy ILM, le performance costanti con test delle performance multipli per la verifica e le API S3 standard di settore, lo storage a oggetti NetApp StorageGRID è la scelta ottimale per lo storage a livelli confluente.

## Dove trovare ulteriori informazioni

Per ulteriori informazioni sulle informazioni descritte in questo documento, consultare i seguenti documenti e/o siti Web:

- Che cos'è Apache Kafka

  "https://www.confluent.io/what-is-apache-kafka/"

- Documentazione sui prodotti NetApp

  "https://www.netapp.com/support-and-training/documentation/"

- Dettagli del parametro S3-sink

  "https://docs.confluent.io/kafka-connect-s3-sink/current/configuration_options.html#s3-configuration-options"

- Apache Kafka

  "https://en.wikipedia.org/wiki/Apache_Kafka"

- Storage infinito nella piattaforma confluente

  "https://www.confluent.io/blog/infinite-kafka-storage-in-confluent-platform/"

- Confluent Tiered Storage - Best practice e dimensionamento

"https://docs.confluent.io/platform/current/kafka/tiered-storage.html#best-practices-and-recommendations"

- Connettore Amazon S3 sink per Confluent Platform

  "https://docs.confluent.io/kafka-connect-s3-sink/current/overview.html"

- Dimensionamento di Kafka

  "https://eventsizer.io"

- Dimensionamento StorageGRID

  "https://fusion.netapp.com/"

- Casi di utilizzo di Kafka

  "https://kafka.apache.org/uses"

- Cluster Kafka con bilanciamento automatico nella piattaforma confluente 6.0

  "https://www.confluent.io/blog/self-balancing-kafka-clusters-in-confluent-platform-6-0/"

  "https://www.confluent.io/blog/confluent-platform-6-0-delivers-the-most-powerful-event-streaming-platform-to-date/"