Mesh di eventi: un'introduzione

RTInsights.com

Red Hat

Capitolo 1: Cosa sono gli eventi?

Un evento è un cambiamento, un'azione o un'osservazione all'interno di un sistema che produce una notifica. Gli eventi vengono attivati da modifiche quali gli aggiornamenti dell'entità di dati master, come la modifica dell'indirizzo di fatturazione di un cliente, o da un'azione come un nuovo ordine o l'autorizzazione per una transazione. Altri trigger sono più semplici, come il raggiungimento della soglia di ricarica della batteria o la lettura periodica della temperatura da un sensore. È comune fare riferimento alla notifica come un "evento". Invece di utilizzare il termine trigger o attivare, gli architetti software spesso parlano di "inviare un evento" quando si verifica una condizione specifica.

Le notifiche di eventi prodotte a livello di programmazione dal sistema di registrazione (SOR) possono contenere uno snapshot dell'entità interessata, un delta (ovvero solo gli attributi che sono stati modificati) o un riferimento (ad esempio, un numero di ordine di acquisto). Gli eventi snapshot e delta in genere sono portatori di un significato per l'attività aziendale, hanno una struttura complessa e sono di solito rappresentati in JSON o XML. Quando nella notifica di evento è contenuto un riferimento, è necessario ricorrere a un richiamo al SOR per ottenere un set di dati più significativo. Ad esempio, se un evento contiene solo il numero dell'ordine di acquisto, sarà necessario richiamare l'applicazione di gestione degli ordini per richiedere ulteriori informazioni sull'entità dell'ordine di acquisto.

Anche il registro delle modifiche del database (DB) può essere un'origine di eventi. Il vantaggio del registro delle modifiche del DB è che viene gestito automaticamente dal DBMS e non richiede che lo sviluppatore dell'applicazione esegua modifiche per produrre la notifica di evento.

In un sistema basato sugli eventi, un componente produce un evento senza necessariamente aspettarsi o verificare che tale evento venga consumato. Questo disaccoppiamento di produzione e consumo degli eventi consente agli sviluppatori di creare un evento senza necessariamente conoscere le specifiche dell'ambiente in cui verrà utilizzato, semplificando notevolmente il processo.

Indipendentemente dal tipo di evento, l'ordinamento su base temporale è una caratteristica importante. L'ordinamento temporale, anche definito conservazione delle sequenze, garantisce che il flusso di eventi mantenga l'integrità dello stato dell'attività aziendale. Ad esempio, un evento snapshot rappresenta lo stato di un'entità in un dato momento. In caso di sequenza errata, ne risulterebbe una rappresentazione non valida dell'entità. Un flusso di eventi è un insieme di eventi ordinati su base temporale che, una volta prodotto, non può essere modificato. Tale proprietà viene descritta come immutabile.

In un sistema basato sugli eventi, un componente produce un evento senza necessariamente aspettarsi o verificare che tale evento venga immediatamente consumato. Esiste un accoppiamento libero tra produttore e consumatore dell'evento. Dal punto di vista dello sviluppo, il disaccoppiamento tra produzione e consumo degli eventi semplifica notevolmente le cose. Uno sviluppatore può creare un evento senza necessariamente conoscere le specifiche dell'ambiente in cui verrà utilizzato.

Questo contrasta con i sistemi basati sulle chiamate dei servizi mediante un modello di interazione richiesta-risposta. I ruoli nelle architetture SOA (Service-Oriented Architecture) e nello stile REST più moderno si basano su un accoppiamento stretto e sincrono tra client e server. L'accoppiamento stretto dei componenti comporta alcune limitazioni, che si possono superare in un'architettura basata sugli eventi.

Tecnologie che rendono possibile l'uso di eventi

Negli ultimi 20 anni i sistemi di messaggistica hanno implementato le funzionalità principali per le comunicazioni point-to-point e pub/sub. Di recente, il cloud computing ha fatto emergere i limiti delle attuali tecnologie in aree quali l'interoperabilità e la scalabilità. Molti progetti e standard open source hanno superato le funzionalità principali e hanno consentito lo sviluppo su larga scala di applicazioni basate sugli eventi, tra cui Apache Kafka, Advanced Message Queueing Protocol e Knative.

1. Apache Kafka è un sistema di messaggistica di tipo pubblicazione-sottoscrizione che utilizza un registro di commit distribuito come registrazione durevole di tutti i messaggi. Più produttori possono aggiungere messaggi a un topic e più consumatori possono leggere i messaggi gestendo la propria posizione nella sequenza. Alcune altre caratteristiche sono la possibilità di riprodurre i messaggi, un throughput elevato e la scalabilità orizzontale. Apache Kafka viene spesso implementato quando è presente un volume di messaggi elevato, con requisiti di bassa latenza.

2. Advanced Message Queuing Protocol (AMQP) è un protocollo di messaggistica standard con funzionalità avanzate, come il controllo del flusso, la garanzia di recapito dei messaggi e la sicurezza. In qualità di protocollo wire-level, consente l'integrazione tra diverse piattaforme e prodotti di messaggistica. AMQP è una buona opzione per il protocollo della mesh di eventi in un ambiente eterogeneo cloud/on-premise. L'implementazione AMQP dell'Apache Software Foundation (QPID) offre una serie di funzionalità come coda, transazioni, gestione e clustering, oltre alle API per C++ e Java (JMS).

3. Knative è una piattaforma basata su Kubernetes per l'elaborazione senza server. Gli sviluppatori possono concentrarsi sullo sviluppo di funzioni specifiche, mentre la piattaforma si occupa dell'allocazione delle risorse di calcolo in base alle richieste. Grazie a Knative, è possibile distribuire ed eseguire le applicazioni senza server su qualsiasi piattaforma Kubernetes, evitando la dipendenza da un unico fornitore.

Knative Eventing è un componente di Knative che consente di implementare sulla piattaforma applicazioni basate sugli eventi che reagiscono alle informazioni in tempo reale tramite notifiche di eventi. È coerente con la specifica CloudEvents, che garantisce l'interoperabilità con diversi protocolli di messaggistica, tra cui Apache Kafka, MQTT e AMQP. Knative Eventing rende possibile lo sviluppo basato sugli eventi nativo del cloud e indipendente dal cloud.

Queste tre tecnologie stanno emergendo come elementi chiave per l'architettura basata sugli eventi per i microservizi.

Capitolo 2: L'approccio basato sugli eventi in azienda

Per tenere il passo con una concorrenza molto dinamica, le aziende sono costrette a migliorare velocità e reattività dei processi sia interni che visualizzabili dai clienti. In quasi tutti i settori, produttori e consumatori di eventi sono sempre più numerosi nei rispettivi ecosistemi IT. L'utilizzo corretto del volume di eventi e delle informazioni nei flussi di eventi richiede l'adozione di un'architettura basata sugli eventi (EDA).

L'architettura basata sugli eventi (EDA, Event-Driven Architecture) è un paradigma di architettura software che promuove la produzione, il rilevamento, il consumo e la reazione agli eventi[1]. Ma l'EDA non si limita alla produzione e al consumo di eventi. Prevede infatti la pianificazione del modo in cui gli eventi vengono interpretati, trasformati, pubblicati per i sottoscrittori, propagati su reti distribuite e resi persistenti. Questi modelli di progettazione illustrano l'ambito di una EDA.

Alcuni modelli di progettazione e funzionalità chiave di un'architettura EDA sono:

- Eventuale coerenza tra il sistema di registrazione e i sistemi che gestiscono le copie. Invece di eseguire transazioni in due fasi o distribuite, si utilizzano gli eventi per consentire ai sistemi di diventare coerenti dopo un certo ritardo. Il ritardo può essere molto breve (ad esempio, pochi secondi) o molto più lungo (ad esempio, alcune ore) a condizione che siano soddisfatti i requisiti aziendali.

- Middleware pub/sub e orientato ai messaggi, per garantire il trasporto necessario per spostare in modo affidabile i messaggi di evento da un punto all'altro o da server di pubblicazione a sottoscrittore.

- Routing e trasformazione dei protocolli utilizzando i modelli di integrazione aziendali. I sistemi EDA utilizzano, in genere, broker e un service bus per fornire una topologia hub e spoke e funzionalità in grado di collegare diversi formati di middleware e messaggi.

- Analisi di flusso, ovvero un tipo di elaborazione del flusso di eventi che applica machine learning o regole di business a fronte di una finestra di eventi delimitata nel tempo per prendere decisioni o intraprendere azioni.

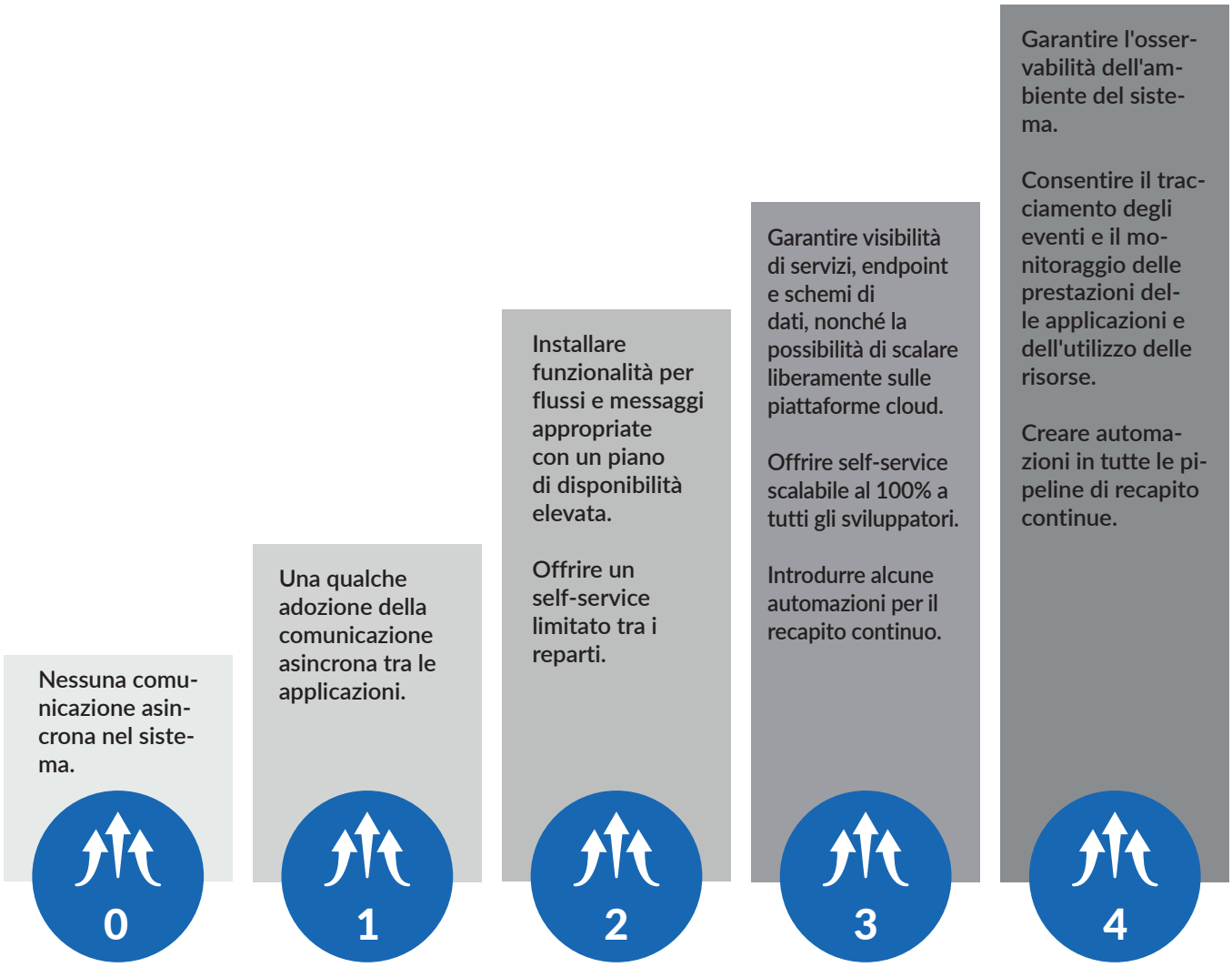
- Flusso di eventi come sistema di registrazione di prima classe, dove in un dominio di microservizi, il flusso di eventi è una fonte autorevole di dati. Questo modello viene definito Event Sourcing.

- Elaborazione dei messaggi idempotente, che garantisce che i sistemi siano progettati in modo da poter elaborare più volte lo stesso evento senza modificare il risultato oltre l'elaborazione iniziale.

- Mesh di eventi, ovvero un'infrastruttura dinamica che consente la distribuzione dei messaggi in un'azienda distribuita.

Con l'adozione più diffusa dei concetti dell'EDA, le aziende progrediscono verso livelli di maturità sempre più elevati.

[1] Fonte: Wikipedia, "Event-Driven Architecture"



Livello di maturità dell'architettura basata sugli eventi

Al livello 0, non vi è alcuna comunicazione asincrona in azienda. Tutta l'integrazione è sincrona, ad esempio i servizi Web di richiesta-risposta sincroni richiamati dai browser Web alle applicazioni back-end.

Al livello 1, si rileva una qualche adozione della comunicazione asincrona tra le applicazioni. Ad esempio, scambio di messaggi point-to-point tra sistemi correlati mediante una piattaforma di messaggistica come IBM MQ, JMS o Apache Kafka.

Al livello 2, sono installate funzionalità per flussi e messaggi con un certo livello di disponibilità elevata e self-service limitato. L'infrastruttura di messaggistica è in grado di gestire gli errori dell'infrastruttura. Può fornire garanzie sulla consegna dei messaggi (ad esempio, almeno una volta, al massimo una volta) e può partecipare a transazioni distribuite. Più applicazioni, basate su stack tecnologici diversi in altre parti dell'azienda, possono connettersi ai flussi di eventi per utilizzare al meglio le informazioni disponibili.

Al livello 3, visibilità e scalabilità sono abilitate. Le applicazioni registrano gli endpoint e lo schema dei dati in una directory a cui è possibile accedere da altre applicazioni. Le applicazioni che producono eventi diventano meno consapevoli di tutti i client connessi e gli stessi eventi si possono utilizzare per nuovi scopi, come la registrazione e il monitoraggio delle attività di business. L'infrastruttura di messaggistica è in grado di gestire carichi maggiori e variabili, grazie alla scalabilità automatica, e consente ai produttori e ai consumatori di eventi di non dover conoscere la topologia fisica della rete. Alcuni processi di automazione vengono implementati per il processo di rilascio, noto come recapito continuo, per facilitare l'implementazione delle applicazioni basate sugli eventi.

Infine, al livello 4, viene abilitata l'osservabilità dell'intero sistema e i processi completamente automatizzati per i rilasci offrono funzionalità di recapito continuo. Gli eventi sono sempre più diffusi nell'organizzazione, con nuove origini e destinazioni che vengono aggiunte con frequenza regolare. L'obiettivo diventa gestire una robusta infrastruttura di messaggistica in grado di scalare rapidamente con l'aumentare del volume dei messaggi. È disponibile un'osservabilità a livello aziendale, per consentire agli amministratori di monitorare l'utilizzo delle risorse in tempo reale e tracciare i messaggi end-to-end su più nodi della mesh di eventi. Si implementa l'automazione completa per consentire il recapito continuo, in modo che le applicazioni basate sugli eventi possano essere implementate con un semplice clic.

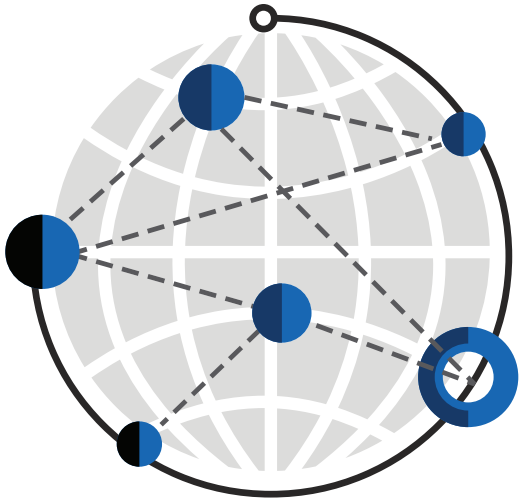
In coincidenza con il passaggio a un'architettura EDA, si sta registrando un altro cambiamento fondamentale nel campo delle architetture. In particolare, si nota un'esplosione nell'uso della tecnologia senza server, che consente agli sviluppatori di concentrarsi sul codice senza considerare i problemi di gestione e provisioning dell'infrastruttura back-end.

Data la disponibilità di queste funzionalità, le organizzazioni stanno adottando l'EDA per supportare una serie di iniziative aziendali. In coincidenza con il passaggio a un'architettura EDA, si sta registrando un altro cambiamento fondamentale nel campo delle architetture. In particolare, si nota un'esplosione nell'uso della tecnologia senza server, che consente agli sviluppatori di concentrarsi sul codice senza considerare i problemi di gestione e provisioning dell'infrastruttura back-end.

La tecnologia senza server presenta nuove sfide per l'architettura EDA. Eventi Java, cicli di gestione degli eventi e strutture EDA meno recenti non sono stati progettati per funzionare negli ambienti applicativi odierni. Queste tecnologie meno recenti erano adatte negli scenari di implementazione client-server e data center tradizionali. Non si possono adattare facilmente a scenari applicativi moderni basati su elementi accoppiati liberamente che operano insieme su un'architettura cloud distribuita.

Tecnologie come la gestione degli eventi Knative, che risponde a un'esigenza comune di sviluppo nativo del cloud e fornisce primitive componibili per consentire origini e consumatori di eventi ad associazione tardiva, contribuiscono a completare vicendevolmente i sistemi EDA e senza server.

Capitolo 3: Cos'è una mesh di eventi?



Le grandi aziende utilizzano topologie di rete e applicazioni distribuite su vasta scala. Devono quindi poter recapitare in modo affidabile un elevato volume di eventi su reti globali, attraversando cloud ibridi e applicazioni locali, evitando al contempo i colli di bottiglia. La mesh di eventi è un fattore chiave per l'azienda che utilizza architetture basate sugli eventi.

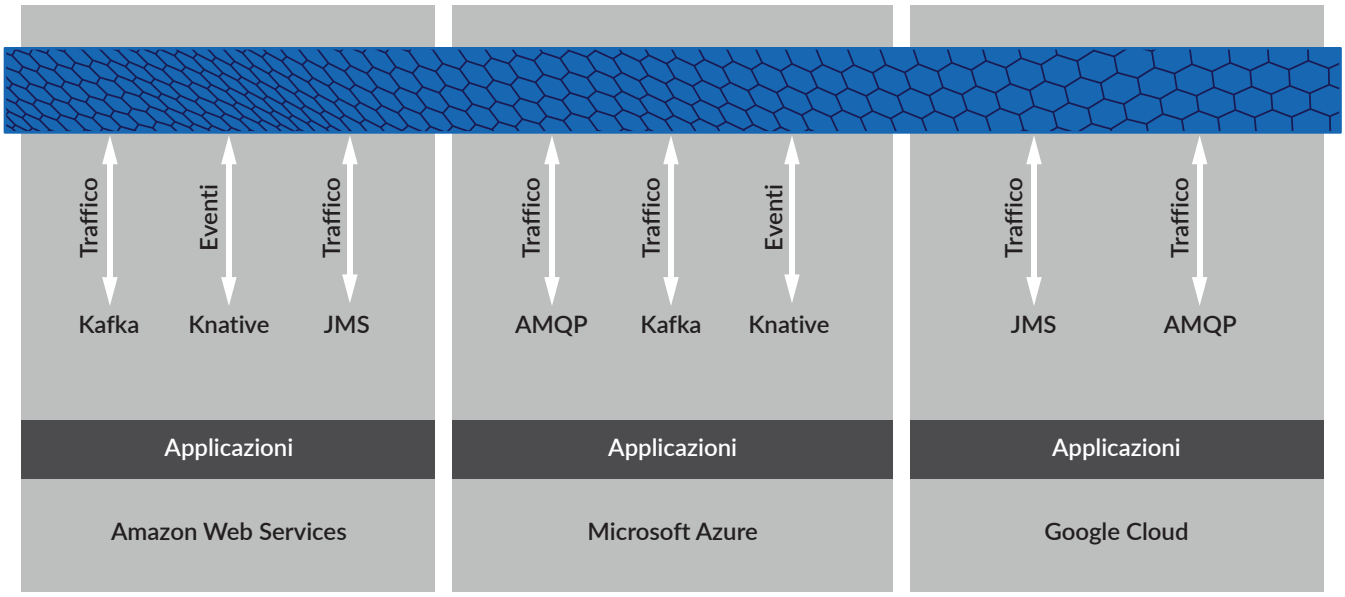
Una mesh di eventi è un'infrastruttura dinamica che propaga gli eventi su diverse piattaforme cloud ed esegue la traduzione dei protocolli. Alcune delle funzionalità principali sono:

- Supporto per l'ingresso e l'uscita di eventi con varie modalità di trasporto, come Kafka, Knative, HTTP, AMQP e altri
- Tolleranza agli errori per il recapito dei messaggi ad affidabilità elevata, con ripristino automatizzato dagli errori di rete e destinazioni di fallback per i messaggi non recapitabili
- Supporto di bridge multiprotocollo tra eventi, applicazioni e piattaforme di messaggistica diversi
- Supporto per l'implementazione in sede e multi-cloud per garantire un'infrastruttura uniforme in tutta l'azienda
- Più API client per una vasta gamma di linguaggi e ambienti di programmazione
- Supporto di Multicast (tutti i sottoscrittori ricevono una copia di ogni messaggio) o Anycast (un sottoscrittore riceve una copia di ogni messaggio prodotto)
- Console di gestione per l'amministrazione della mesh e il monitoraggio delle attività
- Trasmissione sicura dei messaggi di evento

Queste funzionalità sono necessarie per supportare lo sviluppo di applicazioni moderne. I microservizi e le architetture cloud spesso utilizzano una mesh di servizi, ovvero un'infrastruttura di rete che supera la logica della rete, consentendo al microservizio di concentrarsi sulla logica di business. Una mesh di servizi supporta l'elaborazione richiesta-risposta sincrona. Allo stesso modo, una mesh di eventi supporta gli sviluppatori di applicazioni alleviando i problemi relativi alla posizione dei consumatori sulle topologie locali, regionali e globali a supporto dei casi di utilizzo basati sugli eventi non strettamente accoppiati.

La figura riportata di seguito illustra gli elementi di una mesh di eventi. Gli eventi possono fluire in modo bidirezionale lungo la topologia multi-cloud. Con il costante aumento delle applicazioni che producono e consumano eventi, una caratteristica fondamentale della mesh di eventi è che gli eventi pubblicati da qualsiasi applicazione in qualsiasi ambiente di programmazione (ad esempio, un evento JMS Java) in un cloud possono essere utilizzati da un'applicazione su un cloud diverso utilizzando un'altra API (ad esempio, una sottoscrizione a un evento Kafka in un'applicazione Python). In questo modo, gli sviluppatori di applicazioni possono semplificare la progettazione e la gestione di reti di distribuzione di eventi complesse e i team di sviluppo delle applicazioni possono scegliere l'ambiente di sviluppo preferito, senza alcun vincolo a livello di piattaforma di messaggistica. Inoltre, il modello di distribuzione delle applicazioni senza server riduce i requisiti dell'infrastruttura per i diversi ambienti applicativi.

Mesh di eventi multi-cloud



Capitolo 4: Implementazione della mesh di eventi nel mondo reale

Un'architettura basata sugli eventi che utilizza la mesh di eventi supporta una vasta gamma di casi di utilizzo, comprese le topologie multi-cloud e distribuite su vasta scala che utilizzano stack di applicazioni diversificati.

Gli esempi riportati in questa sezione illustrano come viene utilizzata una mesh di eventi in scenari reali.

Sistema per consigli e raccomandazioni per il noleggio di auto

Una società di noleggio auto dispone di un'applicazione di prenotazione e fidelizzazione accessibile tramite app Web e mobili, con un'ampia base di utenti e molti contenuti dinamici e pagine. Le app generano un grande volume di eventi di interazione che devono essere elaborati in tempo reale al fine di creare un'esperienza personalizzata per l'utente e aggiornare il modello di comportamento del cliente. Il progetto per questo scenario prevede che i componenti dell'interfaccia utente (UI) inviino gli eventi in modalità push mediante il componente pub-sub della mesh di eventi, come Apache Kafka.

Un elaboratore di big data, come Apache Spark, sottoscrive questi flussi per l'analisi dei dati. Allo stesso tempo, l'analisi dei flussi esamina gli eventi in tempo reale mediante topic Kafka e genera nuovi eventi, che vengono inviati all'interfaccia utente come contenuti dinamici o ad altre applicazioni, come il rilevamento delle frodi. Un progetto di questo tipo si è dimostrato altamente scalabile, in grado di gestire un numero di eventi al giorno molto elevato e di offrire suggerimenti e personalizzazioni in tempo reale.

Multinazionale finanziaria

Un'organizzazione finanziaria con filiali e uffici in più paesi e città hub di contrattazioni deve ridurre latenza e tempi di risposta per garantire la sincronizzazione dei dati di trading in tempo reale. Le app distribuite si connettono agli edge della mesh di eventi e inviano messaggi ai flussi di dati identificati da nomi virtuali che rappresentano le destinazioni desiderate. I router, come Apache Qpid, garantiscono che i messaggi vengano indirizzati alla destinazione appropriata utilizzando i servizi di denominazione e le regole di routing. I messaggi vengono instradati senza che i client conoscano la topologia fisica della rete e del sistema di messaggistica.

Questo caso di utilizzo mette in evidenza un sistema di messaggistica basato su standard e multiprotocollo. Un'azienda di telecomunicazioni ha adottato un'applicazione mainframe per il servizio clienti realizzata utilizzando CICS/COBOL negli anni '80. L'azienda invia gli ordini di riparazione a un'app di distribuzione Java in esecuzione su un server Linux e, infine, invia le notifiche a un'applicazione mobile. Per supportare questo caso d'uso, il mainframe comunica con il sistema di messaggistica mediante un canale AMQP tra IBM MQ e il broker messaggi. L'applicazione Java utilizza l'API standard JMS per la ricezione/invio dei messaggi. Infine, la comunicazione con l'applicazione mobile viene gestita utilizzando un'API AMQP leggera supportata da Apache Qpid.

Messaggistica interoperativa per il controllo dei treni

La Federal Railroad Administration ha finanziato lo sviluppo di una soluzione di messaggistica che consenta alle applicazioni di scambiare messaggi, indipendentemente da posizione fisica e tipo di connettività. Le app includevano sistemi a bordo dei treni, sistemi per gli uffici e segnali lungo il tragitto. Per garantire la gestione sicura dei treni, è necessaria una disponibilità elevata di tutti gli scambi di messaggi. I treni viaggiano attraverso reti a banda più o meno larga e passano in tunnel privi di connettività di rete. La consegna di tutti i messaggi deve essere garantita e l'intera rete è altamente disponibile e tollerante agli errori. Per supportare lo scambio di messaggi, è stata scelta un'implementazione di Qpid AMQP in cluster a disponibilità elevata e multitrasporto.

Capitolo 5: Conclusioni e altre risorse

Le aziende all'avanguardia considerano un'EDA di uguale importanza e complementare a un'architettura orientata ai servizi. Il motivo: in molte aziende moderne, sono sempre più numerosi i produttori e i consumatori di eventi nei rispettivi ecosistemi IT e il volume degli eventi è in costante aumento. Con un corretto utilizzo di questi eventi, le aziende possono creare processi aziendali interni e interazioni con i clienti più agili.

In genere le aziende seguono un percorso dell'architettura EDA che può essere visto in termini di modello di maturità. La maggior parte delle imprese inizia con l'integrazione di alcune applicazioni correlate, sfruttando un unico sistema di messaggistica. Nel corso del tempo, le aziende più mature implementano una mesh di eventi come fattore abilitante per l'integrazione liberamente accoppiata in topologie distribuite su vasta scala di sistemi legacy e moderne applicazioni basate su microservizi.

Una mesh di eventi è caratterizzata da una serie di funzioni chiave che supportano tali modelli di interazione e riduce la complessità per gli sviluppatori di applicazioni moderne, che possono implementare modelli di progettazione liberamente accoppiati in tempo quasi reale. Le implementazioni senza server, i microservizi e la mesh di eventi supportano ambienti in cui le risorse di elaborazione possono scalare in modo rapido e automatico. Ma forse il vantaggio più rilevante è la possibilità offerta agli sviluppatori di applicazioni di concentrarsi sull'implementazione della logica di business, utilizzando i migliori strumenti disponibili per questo scopo. L'utilizzo di architetture EDA e mesh di eventi semplifica lo sviluppo di applicazioni. Gli sviluppatori di eventi devono preoccuparsi solo del proprio ambiente. Una volta pubblicato l'evento, qualsiasi consumatore può utilizzarlo indipendentemente dalla piattaforma di sviluppo o dalla tecnologia di streaming utilizzata o dal cloud su cui si trova l'evento.

Red Hat consente di creare una mesh di eventi dinamica, con prodotti basati su open source nella famiglia AMQ.
Per ulteriori informazioni, seguire questi link.

Webinar: Introduzione alle architetture basate sugli eventi con l'utilizzo di Apache Kafka

Articolo: Cos'è Apache Kafka?

Report degli analisti: Applicazioni basate sugli eventi con i flussi Red Hat AMQ

Scheda tecnica: Red Hat AMQ: Semplificazione di Apache Kafka su Red Hat OpenShift

In dettaglio: Architettura basata sugli eventi per un modello di cloud ibrido

Scheda tecnica: Integrazione di Red Hat: Connettività nativa del cloud per app e sistemi

RTInsights è una risorsa Web indipendente guidata da esperti per i responsabili aziendali e IT dei settori verticali. Aiutiamo i nostri lettori a capire come possono trasformare le loro aziende per raggiungere risultati di valore superiore e nuovi modelli di business con intelligenza artificiale, analisi in tempo reale e IoT. Offriamo spiegazioni e orientamenti nella gamma spesso confusa di approcci e soluzioni dei fornitori. Forniamo ai nostri partner una combinazione unica di servizi e una vasta esperienza nel settore per migliorare il marketing dei prodotti, la generazione di lead e l'attività di thought leadership.

Red Hat è il fornitore leader a livello mondiale di soluzioni software open source per l'impresa, che utilizza un approccio basato sulla comunità per fornire tecnologie affidabili e ad alte prestazioni per Linux, cloud ibrido, container, eventing e Kubernetes. Red Hat aiuta i clienti a sviluppare applicazioni native del cloud, a integrare le applicazioni IT già esistenti e nuove e ad automatizzare e gestire ambienti complessi. Red Hat, consulente di fiducia di aziende Fortune 500, offre servizi di assistenza, formazione e consulenza pluripremiati che offrono i vantaggi dell'innovazione aperta a qualsiasi settore. Red Hat è un hub connettivo all'interno di una rete globale di aziende, partner e comunità, che aiuta le organizzazioni a crescere, trasformarsi e prepararsi al futuro digitale.