Whitepaper AWS

Implementazione di microservizi in AWS

Implementazione di microservizi in AWS: Whitepaper AWS

Table of Contents

Implementazione di microservizi in AWS

Data di pubblicazione: 9 novembre 2021 (Cronologia del documento e collaboratori)

Riassunto

I microservizi sono un approccio architettonico e organizzativo allo sviluppo del software per accelerare i cicli di implementazione, favorire l'innovazione e la proprietà, migliorare la manutenibilità e la scalabilità delle applicazioni software e dimensionare le organizzazioni che forniscono software e servizi utilizzando un approccio agile che aiuta i team a lavorare in modo indipendente. Con un approccio a microservizi, il software è composto di servizi di dimensioni ridotte, che comunicano tra loro tramite interfaccia del programma dell'applicazione (API) definite in modo preciso che possono essere implementate in modo indipendente. Questi servizi sono controllati da piccoli team autonomi. L'agilità di questo approccio è la chiave per ridimensionare con successo la tua organizzazione.

Si osservano tre modelli comuni quando i clienti AWS creano microservizi: basato sulle API, basato sugli eventi e flussi di dati. Questo whitepaper presenta i tre approcci e sintetizza le caratteristiche comuni dei microservizi, illustrando le principali problematiche relative alla loro creazione e mostrando in che modo i team di prodotto possono avvalersi di Amazon Web Services (AWS) per superare queste sfide.

A causa della natura piuttosto complessa di vari argomenti discussi in questo whitepaper, tra cui l'archiviazione dei dati, la comunicazione asincrona e l'individuazione dei servizi, prima di effettuare scelte di architettura si consiglia di valutare requisiti specifici e casi d'uso delle loro applicazioni.

Introduzione

Le architetture di microservizi non sono un approccio completamente nuovo al settore dell'ingegneria del software, quanto piuttosto rappresentano una combinazione di vari concetti di successo nonché comprovati come:

- Sviluppo di software agile
- Architetture orientate ai servizi
- Progettazione di API di primo livello
- Integrazione e distribuzione continua (CI/CD)

In molti casi, per i microservizi sono utilizzati anche i modelli di progettazione proposti da Twelve-Factor App.

Questo whitepaper descrive, per prima cosa, i diversi aspetti di un'architettura di microservizi altamente scalabile e con tolleranza agli errori (interfaccia utente, implementazione di microservizi e datastore) e come crearla in AWS avvalendosi della tecnologia dei container. Quindi suggerisce i servizi AWS necessari per implementare un'architettura di microservizi serverless generica in modo da ridurre la complessità operativa.

Un modello operativo serverless è definito dai seguenti principi:

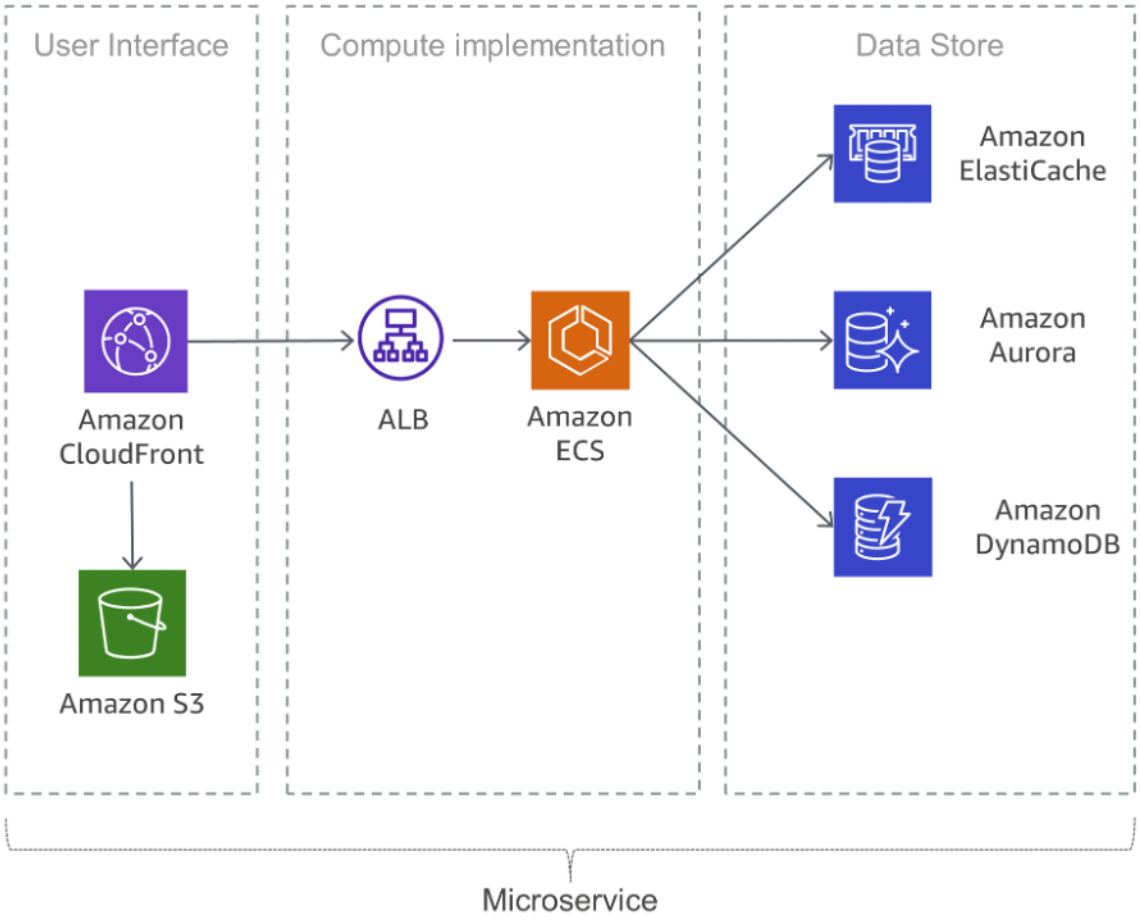
- Nessuna infrastruttura da fornire o gestire
- Ridimensionamento automatico per unità di consumo
- Modello di fatturazione Pay for value
- Disponibilità e tolleranza ai guasti integrate

Infine, questo whitepaper offre uno sguardo al sistema in generale e discute gli aspetti relativi a molteplici servizi di un'architettura di microservizi, come monitoraggio e auditing distribuiti, consistenza dei dati e comunicazione asincrona.

Questo whitepaper si concentra solo sui carichi di lavoro in esecuzione su AWS Cloud. Non copre scenari ibridi o strategie di migrazione. Per ulteriori informazioni sulla migrazione, fare riferimento al whitepaper Metodologia di migrazione dei container.

L'architettura di microservizi su AWS

Le tipiche applicazioni monolitiche sono realizzate su vari livelli: un'interfaccia utente, un livello logico di dominio e un livello di persistenza. L'idea centrale di un'architettura di microservizi è la suddivisione delle funzionalità in categorie verticali logiche, non secondo i livelli tecnologici ma implementando un dominio specifico. La seguente immagine mostra un'architettura di riferimento per una tipica applicazione di microservizi su AWS.



Tipica applicazione di microservizi su AWS

Argomenti

- Interfaccia utente
- Microservizi
- Datastore

Interfaccia utente

Le moderne applicazioni web spesso usano framework JavaScript per implementare un'applicazione su singola pagina che comunichi con un'API Representational State Transfer (REST) o RESTful. I contenuti Web statici possono essere offerti utilizzando Amazon Simple Storage Service (S3) e Amazon CloudFront.

Poiché i client di microservizi sono offerti dalla posizione edge più vicina e ricevono risposte dalla cache o da un server proxy con connessioni ottimizzate al server di origine, la latenza può essere ridotta in modo significativo. Tuttavia, i microservizi in esecuzione a breve distanza gli uni dagli altri non traggono beneficio dalle reti per la distribuzione di contenuti. In alcuni casi, questo approccio potrebbe effettivamente aggiungere ulteriore latenza. Una best practice prevede l'implementazione di altri sistemi di caching per ridurre il sovraccarico di comunicazioni e la latenza. Per ulteriori informazioni, fai riferimento all'argomento the section called “Sovraccarico di comunicazioni”.

Microservizi

Le API sono la porta che conduce ai microservizi, il che significa che le API fungono da punto di ingresso per la logica delle applicazioni dietro una serie di interfacce programmatiche, in genere un'API per servizi web RESTful. Tale API accetta ed elabora le chiamate dai client e può implementare funzionalità specifiche quali gestione del traffico, filtraggio delle richieste, routing, memorizzazione nella cache, autenticazione e autorizzazione.

Implementazione microservizi

AWS presenta componenti integrati che supportano lo sviluppo di microservizi. Due approcci molto usati prevedono l'utilizzo di AWS Lambda e container Docker con AWS Fargate.

Con AWS Lambda, è sufficiente caricare il codice e lasciare che Lambda si occupi di tutte le operazioni necessarie per avviare e dimensionare l'implementazione in modo da soddisfare la curva effettiva delle richieste con la massima disponibilità. Non è necessaria alcuna amministrazione dell'infrastruttura. Il servizio Lambda supporta diverse sintassi di programmazione e può essere attivato da altri servizi AWS oppure richiamato direttamente da qualsiasi applicazione web o per dispositivi mobili. Uno dei maggiori vantaggi di AWS Lambda è la possibilità di progredire in modo rapido: è possibile concentrarsi sulla logica di business dal momento che la sicurezza e il ridimensionamento sono gestiti da AWS. L'approccio prescrittivo di Lambda guida la piattaforma scalabile.

Un metodo utilizzato di frequente per la riduzione del carico operativo dell'implementazione è l'uso di container. Le tecnologie di container come Docker sono diventate sempre più popolari negli ultimi anni grazie a vantaggi come la portabilità, la produttività e l'efficienza. La curva di apprendimento dei container può essere ripida e richiede di prestare attenzione agli aspetti di sicurezza per le immagini Docker e il loro monitoraggio. Amazon Elastic Container Service (Amazon ECS) e Amazon Elastic Kubernetes Service (Amazon EKS) eliminano la necessità di installare, gestire e dimensionare la propria infrastruttura di gestione dei cluster. Grazie alle chiamate API, è possibile avviare e interrompere applicazioni basate su Docker, eseguire query sullo stato di un cluster e accedere a molte funzionalità comuni come gruppi di sicurezza, sistemi di bilanciamento del carico, volumi di Amazon Elastic Block Store (Amazon EBS) e ruoli AWS Identity and Access Management (IAM).

AWS Fargate è un motore di calcolo serverless per container che funziona con Amazon ECS e Amazon EKS. Con Fargate, non devi più preoccuparti di eseguire il provisioning di risorse di calcolo sufficienti per le tue applicazioni di container. Fargate è in grado di lanciare decine di migliaia di container e scalare per eseguire le applicazioni più critiche nella massima semplicità.

Amazon ECS supporta strategie e restrizioni di collocazione dei container per personalizzare le modalità di posizionamento e di arresto delle attività da parte di Amazon ECS. Una restrizione al posizionamento di un'attività è una regola presa in esame durante tale processo. È possibile associare degli attributi alle istanze di container, che sono essenzialmente delle coppie chiave-valore, e quindi utilizzare una regola di restrizione per posizionare l'attività in base ad essi. Utilizzando questo metodo, è possibile posizionare determinati microservizi in base al tipo o alla capacità delle istanze, ad esempio solo su istanze basate su GPU.

Amazon EKS esegue versioni aggiornate del software open source Kubernetes: è possibile quindi utilizzare tutti i plug-in e gli strumenti messi a disposizione della community di Kubernetes. Le applicazioni in esecuzione su Amazon EKS sono completamente compatibili con quelle in esecuzione in qualsiasi ambiente Kubernetes standard, sia che si tratti di data center in locale o di cloud pubblici. Amazon EKS integra IAM con Kubernetes, il che ti consente di registrare entità IAM con il sistema di autenticazione nativo in Kubernetes. Non è necessario impostare manualmente le credenziali per l'autenticazione con il piano di controllo Kubernetes. L'integrazione IAM consente di utilizzare questo servizio per autenticarsi direttamente sul piano di controllo stesso e fornire un accesso granulare preciso all'endpoint pubblico del piano di controllo Kubernetes.

Le immagini Docker usate in Amazon ECS e in Amazon EKS possono essere memorizzate in Amazon Elastic Container Registry (Amazon ECR.) Con Amazon ECR viene meno la necessità di operare e ricalibrare le risorse dell'infrastruttura necessarie per eseguire il registro del container.

Le pipeline di integrazione e distribuzione continua (CI/CD) rappresentano una best practice e una parte vitale di un'iniziativa DevOps che consente rapidi cambiamenti del software mantenendo stabilità e sicurezza del sistema. Tuttavia, questo tema non rientra nell'ambito di questo whitepaper. Per ulteriori informazioni, consultare il whitepaper Integrazione e distribuzione continua in AWS.

Collegamenti privati

AWS PrivateLink è una tecnologia altamente disponibile e scalabile che permette di connettere privatamente il Virtual Private Cloud (VPC) a servizi AWS supportati, ospitati da altri account AWS (servizi di endpoint VPC), e servizi supportati dai partner di AWS Marketplace. Per comunicare con il servizio, non sono necessari un gateway Internet, un dispositivo NAT, un indirizzo IP pubblico, una connessione AWS Direct Connect o una connessione VPN. Il traffico tra VPC e il servizio non esce dalla rete Amazon.

I collegamenti privati sono un ottimo modo per aumentare l'isolamento e la sicurezza dell'architettura di microservizi. Un microservizio, ad esempio, potrebbe essere implementato in un VPC completamente separato, gestito da un bilanciatore del carico ed esposto ad altri microservizi attraverso un endpoint AWS PrivateLink. Con questa configurazione, utilizzando AWS PrivateLink, il traffico di rete da e verso il microservizio non attraversa mai la rete Internet pubblica. Un caso d'uso per tale isolamento include la conformità alle normative per i servizi che gestiscono dati sensibili come PCI, HIPPA e EU/US Privacy Shield. Inoltre, AWS PrivateLink consente di collegare microservizi tra diversi account e Amazon VPC, senza la necessità di regole firewall, definizioni di percorsi o tabelle di routing, semplificando così la gestione della rete. Utilizzando PrivateLink, i fornitori di software come servizio (SaaS) e gli ISV possono offrire le loro soluzioni basate su microservizi con isolamento operativo completo e accesso sicuro.

Datastore

Un datastore permette storage persistente dei dati necessari per il funzionamento dei microservizi. I datastore più utilizzati per i dati di sessione sono sistemi di caching in memoria quali Memcached o Redis. AWS offre entrambe le tecnologie all'interno del servizio gestito Amazon ElastiCache.

L'utilizzo di una cache tra i server applicativi e il database è un meccanismo comune per ridurre il carico di lettura nel database, che a sua volta libera risorse utilizzabili per supportare un numero maggiore di scritture. I sistemi di cache sono anche in grado di migliorare la latenza.

I database relazionali sono ancora molto utilizzati per memorizzare dati strutturati e oggetti di business. AWS fornisce sei motori di database (Microsoft SQL Server, Oracle, MySQL, MariaDB,

PostgreSQL e Amazon Aurora) sotto forma di servizio gestito tramite Amazon Relational Database Service (Amazon RDS).

I database relazionali, tuttavia, non sono progettati per fornire scalabilità illimitata, perciò l'applicazione di tecniche per il supporto di volumi elevati di query può rappresentare un'operazione lunga e complessa.

I database NoSQL sono stati progettati per ottenere scalabilità, prestazioni e disponibilità con la stessa consistenza dei database relazionali. Un fattore da non sottovalutare nei database NoSQL è che in genere non applicano uno schema rigoroso. I dati sono distribuiti su partizioni dotate di scalabilità orizzontale e vengono recuperati tramite chiavi di partizione.

Poiché i singoli microservizi sono progettati per operazioni dedicate, in genere dispongono di modelli di dati semplificati adatti al livello di persistenza dei database NoSQL. È tuttavia importante ricordare che questo tipo di database hanno pattern di accesso differenti rispetto ai database relazionali. Ad esempio, non è possibile unire tabelle. Se questa operazione dovesse rendersi necessaria, è necessario implementare la logica direttamente nell'applicazione. È possibile usare Amazon DynamoDB per creare una tabella di database in grado di memorizzare e recuperare qualsiasi volume di dati e servire qualsiasi livello di traffico. DynamoDB offre prestazioni a livelli di qualche millisecondo, tuttavia alcuni casi d'uso richiedono tempi di risposta nell'ordine dei microsecondi. Amazon DynamoDB Accelerator (DAX) offre funzionalità di memorizzazione nella cache per l'accesso ai dati.

DynamoDB offre anche una funzionalità di auto scaling per regolare dinamicamente la capacità di throughput in risposta al traffico effettivo. Ciononostante, ci sono casi in cui la pianificazione della capacità è difficile o impossibile a causa di picchi di attività di breve durata nell'ambito dell’applicazione. Per tali situazioni, DynamoDB offre un'opzione on demand, che offre una semplice tariffazione per richiesta. DynamoDB on demand è in grado di soddisfare immediatamente migliaia di richieste al secondo senza dover pianificare la capacità a disposizione.

Riduzione della complessità operativa

L'architettura descritta in precedenza in questo whitepaper utilizza già servizi gestiti, ma le istanze di Amazon Elastic Compute Cloud (Amazon EC2) devono comunque ancora essere gestite. Le attività operative necessarie per eseguire, mantenere e monitorare i microservizi possono essere ulteriormente ridotte utilizzando un'architettura completamente serverless.

Argomenti

Implementazione di API

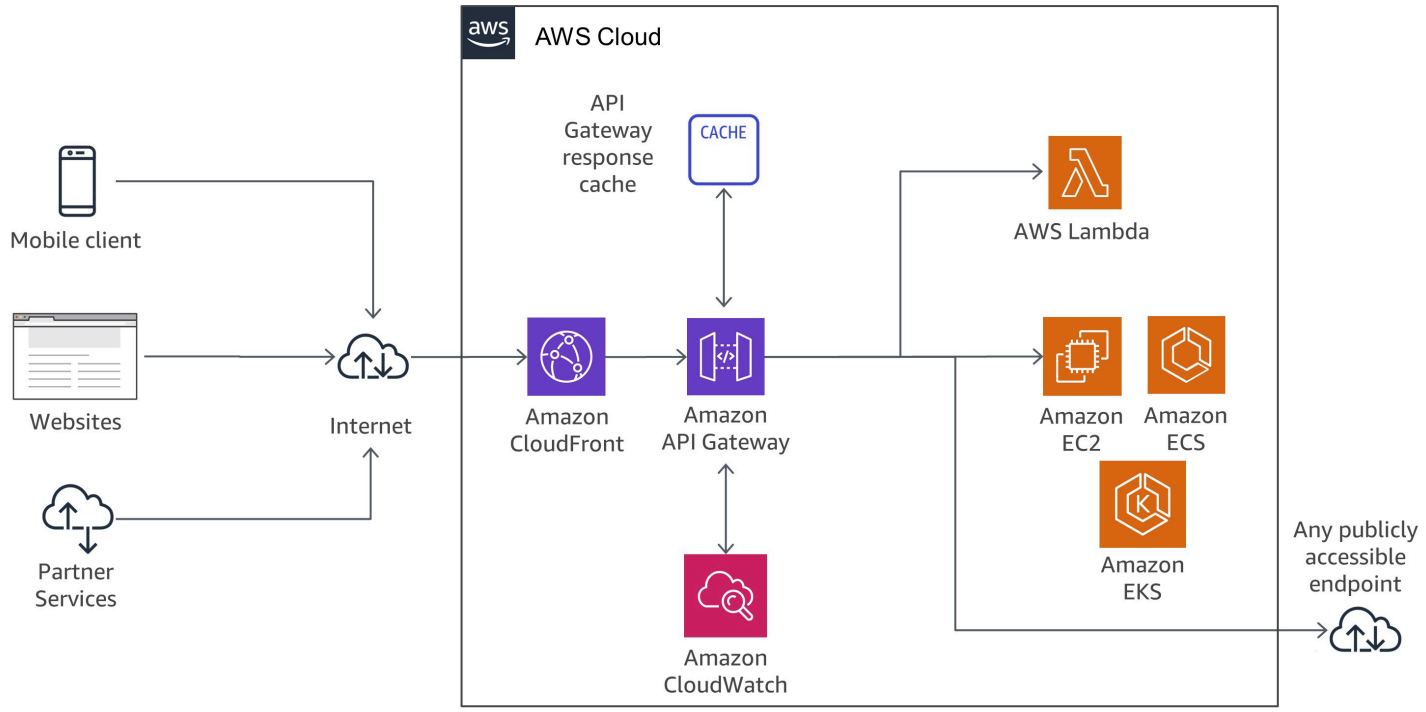
Il processo di progettazione, implementazione, monitoraggio, miglioramento continuo e manutenzione di un'API può essere molto lungo. In alcuni casi, è necessario eseguire più di una versione di un'API per garantire la retrocompatibilità di tutti i client. Le diverse fasi del ciclo di sviluppo (ad esempio sviluppo, test e produzione) rendono le attività operative ancora più complesse.

L'autorizzazione è una caratteristica di importanza fondamentale per tutte le API, ma in genere è complicata da implementare e implica un lavoro ripetitivo. Quando un'API viene pubblicata e funziona correttamente, la sfida successiva è gestire, monitorare e monetizzare l'ecosistema di sviluppatori di terze parti che la utilizzano.

Altre funzionalità e sfide importanti sono le richieste di throttling per proteggere i servizi di back-end, il caching delle risposte API, la gestione della trasformazione di richieste e risposte e la generazione di documentazione e definizioni di API con strumenti come Swagger.

Amazon API Gateway permette di risolvere questi problemi e ridurre la complessità operativa correlata a creazione e manutenzione di API RESTful. API Gateway permette di creare API in modo programmatico importando definizioni Swagger tramite l'API AWS o la Console di gestione AWS. API Gateway funge da porta di accesso per le applicazioni web in esecuzione su Amazon EC2, Amazon ECS, AWS Lambda o qualsiasi ambiente on-premise. Fondamentalmente, API Gateway consente di eseguire API senza dover gestire i server.

La seguente immagine illustra in che modo API Gateway gestisce le chiamate API e interagisce con altri componenti. Le richieste provenienti da dispositivi mobili, siti web o altri servizi di back-end vengono instradate verso il CloudFront Point of Presence (PoP) più vicino per ridurre al minimo la latenza e offrire un'esperienza utente ottimale.



Flusso delle chiamate API Gateway

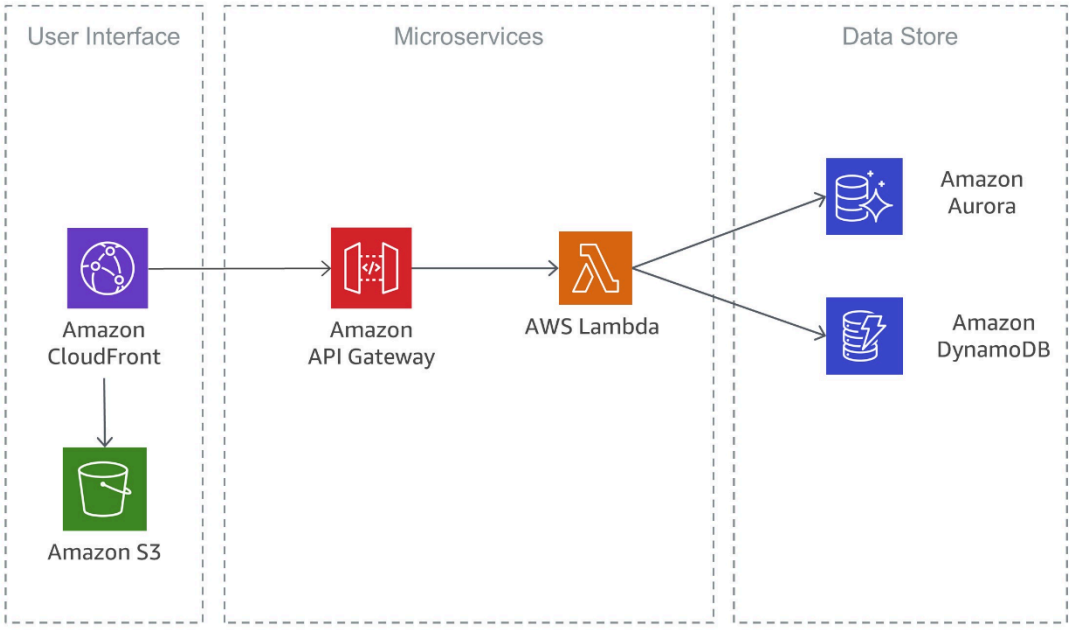
Microservizi serverless

"Nessun server è più semplice da gestire di... nessun server".

Un modo eccellente per eliminare la complessità operativa è eliminare i server.

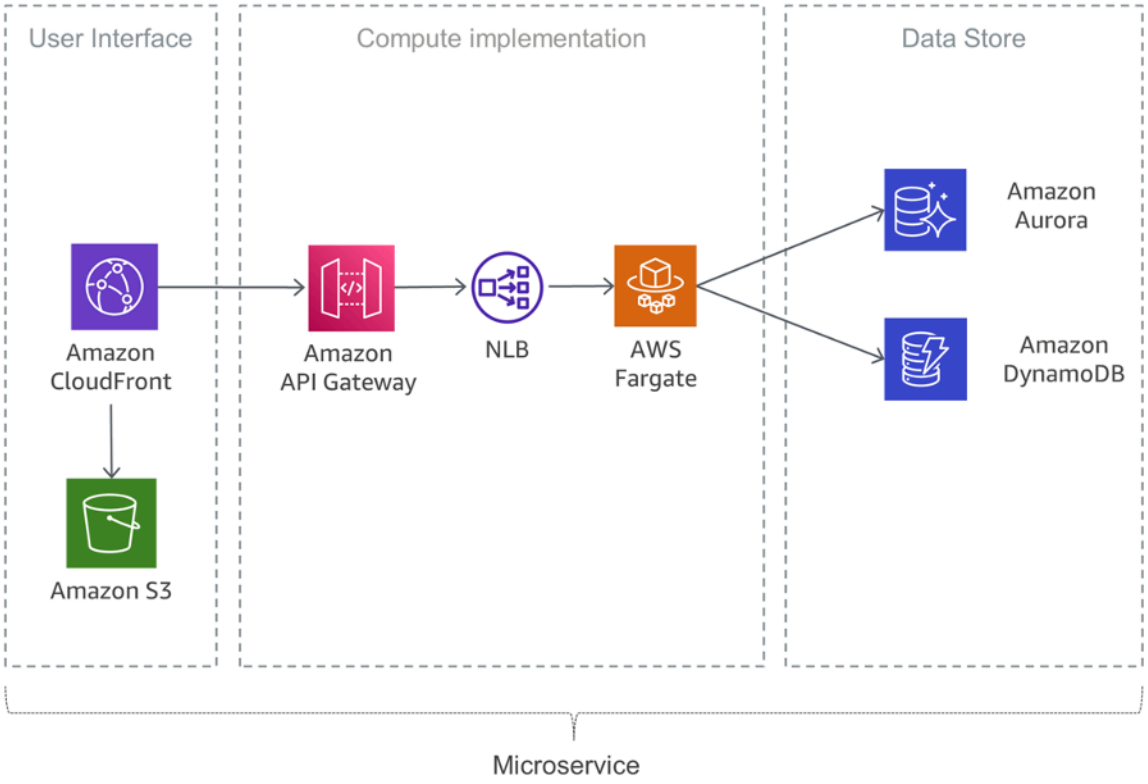
Lambda si integra strettamente con API Gateway. La possibilità di effettuare chiamate sincrone da API Gateway a Lambda consente la creazione di applicazioni completamente serverless ed è descritta in dettaglio nella Guida per gli sviluppatori di Amazon API Gateway.

La seguente immagine mostra l'architettura di un microservizio serverless con AWS Lambda in cui il servizio completo è costituito da servizi gestiti, che eliminano la complessità architetturale causata dalla progettazione per ottenere scalabilità e alta disponibilità ed elimina gli sforzi operativi di esecuzione e monitoraggio dell'infrastruttura sottostante del microservizio.



Microservizio serverless che utilizza AWS Lambda

Un'implementazione simile basata anche su servizi serverless è illustrata nell'immagine seguente. In questa architettura, i container Docker vengono utilizzati con Fargate, quindi non è necessario preoccuparsi dell'infrastruttura sottostante. Oltre ad DynamoDB, viene utilizzato Amazon Aurora Serverless, che è una configurazione di scalabilità automatica on demand per Amazon Aurora (edizione compatibile con MySQL), in cui il database si avvia, si interrompe e ridimensiona la capacità automaticamente in base alle esigenze dell'applicazione.



Microservizi serverless con Fargate

Ripristino di emergenza

Come accennato in precedenza nell'introduzione di questo whitepaper, le tipiche applicazioni di microservizi vengono implementate utilizzando i modelli di applicazione Twelve-Factor. La sezione Processi afferma che "I processi Twelve-Factor sono senza stato e non prevedono alcun tipo di condivisione. Tutti i dati che devono essere persistenti devono essere archiviati in un servizio di backup con stato, in genere un database".

Per una tipica architettura di microservizi, ciò significa che l'obiettivo principale per il ripristino di emergenza dovrebbe essere i servizi a valle che mantengono lo stato dell'applicazione. Ad esempio, possono essere file system, database o code. Quando si crea una strategia di ripristino di emergenza, le organizzazioni generalmente pianificano gli obiettivi del tempo e del punto di ripristino.

L'obiettivo del tempo di ripristino è il ritardo massimo accettabile tra l'interruzione del servizio ed il suo ripristino. Questo obiettivo determina quella che è considerata una finestra temporale accettabile durante la quale il servizio non è disponibile ed è definito dall'organizzazione.

L'obiettivo del punto di ripristino è il periodo di tempo massimo accettabile dall'ultimo punto di ripristino dei dati. Questo obiettivo determina quella che è considerata una perdita accettabile di dati tra l'ultimo punto di ripristino e l'interruzione del servizio ed è definito dall'organizzazione.

Per ulteriori informazioni, consultare il whitepaper Disaster recovery dei carichi di lavoro su AWS: disaster recovery nel cloud.

Elevata disponibilità

Questa sezione esamina più da vicino l'elevata disponibilità per diverse opzioni di calcolo.

Amazon EKS esegue istanze del piano di controllo e del piano dati di Kubernetes in molteplici zone di disponibilità per garantire un'elevata disponibilità. Amazon EKS rileva e sostituisce automaticamente le istanze del piano di controllo non integre e fornisce loro gli aggiornamenti automatici delle versioni e le patch. Questo piano di controllo è costituito da almeno due nodi server API e tre nodi etcd distribuiti su tre zone di disponibilità all'interno di una regione. Amazon EKS utilizza l'architettura di Regioni AWS per mantenere un'elevata disponibilità.

Amazon ECR ospita immagini in un'architettura a elevata disponibilità e ad alte prestazioni, che consente di implementare in modo affidabile le immagini per applicazioni container nelle zone di disponibilità. Amazon ECR funziona con Amazon EKS, Amazon ECS e AWS Lambda, semplificando il flusso di lavoro dallo sviluppo alla produzione.

Amazon ECS è un servizio regionale che semplifica l'esecuzione di container di applicazioni in modo altamente disponibile in più zone di disponibilità all'interno di una Regione AWS. Amazon ECS include numerose strategie di pianificazione che determinano il posizionamento dei container nei cluster in base alle esigenze di risorse (ad esempio CPU o RAM) e ai requisiti di disponibilità.

AWS Lambda esegue la funzione in più zone di disponibilità per assicurare che sia disponibile per elaborare eventi in caso di interruzione del servizio in una specifica zona. Se si configura la funzione affinché si connetta a un Virtual Private Cloud (VPC) nel proprio account, specificare le sottoreti in più zone di disponibilità per garantire un'elevata disponibilità.

Implementazione di applicazioni basate su Lambda

È possibile utilizzare AWS CloudFormation per specificare, implementare e configurare applicazioni serverless.

AWS Serverless Application Model (AWS SAM) è un modo comodo per definire applicazioni serverless. AWS SAM è supportato nativamente da CloudFormation e definisce una sintassi

semplificata per l'espressione di risorse serverless. Per implementare l'applicazione, basta specificare le risorse necessarie e le relative policy di autorizzazione in un modello di CloudFormation, creare un pacchetto con gli artefatti di implementazione e distribuire il modello. Basato su AWS SAM, SAM Local è uno strumento di AWS Command Line Interface (AWS CLI) che fornisce un ambiente in cui poter sviluppare, eseguire test e analizzare le applicazioni serverless in locale prima di caricarle sul runtime di Lambda. È possibile utilizzare AWS SAM Local per creare un ambiente di test locale che simuli l'ambiente di runtime di AWS.

Componenti di sistemi distribuiti

Dopo aver esaminato in che modo AWS permette di risolvere le problematiche correlate ai singoli microservizi, l'attenzione si sposta sulle problematiche relative all'uso di più servizi, come l'individuazione dei servizi, la consistenza dei dati, la comunicazione asincrona e il monitoraggio e verifica distribuiti.

Argomenti

Individuazione dei servizi

Una delle problematiche principali correlate alle architetture di microservizi è consentire ai servizi di rilevarsi l'un l'altro e interagire tra loro. Il carattere distribuito delle architetture di microservizi non solo rende più difficoltosa la comunicazione, ma presenta altri potenziali problemi, in particolare nel controllo dello stato dei sistemi e nell'annuncio della disponibilità di nuove applicazioni. Inoltre, è necessario decidere come e dove memorizzare le informazioni di metastore, ad esempio i dati di configurazione utilizzabili da parte delle applicazioni. In questa sezione approfondiamo diverse tecniche di rilevamento di servizi in AWS per le architetture di microservizi.

Rilevamento di servizi basato su DNS

Amazon ECS ora include il rilevamento di servizi integrato che semplifica l'identificazione e la connessione reciproca dei servizi containerizzati.

In precedenza, per garantire che i servizi fossero in grado di essere rilevati e di connettersi tra loro, era necessario configurare ed eseguire il proprio sistema di rilevamento dei servizi basato su Amazon Route 53, AWS Lambda ed ECS Event Stream o collegare tutti i servizi a un bilanciatore del carico.

Amazon ECS crea e gestisce un registro di nomi di servizi utilizzando l'API di auto-denominazione di Route 53. I nomi vengono automaticamente associati a un set di record DNS in modo che sia

possibile i n modo da creare nel codice riferimenti ai servizi per nome e scrivere query DNS che risolvano il nome all'endpoint corrispondente in fase di runtime. È possibile specificare le condizioni di controllo dello stato nella definizione dell'attività di un servizio. Amazon ECS garantirà che tramite una singola ricerca di servizi si restituisca solo endpoint di servizio integri.

Inoltre, è anche possibile utilizzare l'individuazione unificata dei servizi per i servizi gestiti da Kubernetes. Per consentire questa integrazione, AWS ha contribuito al progetto External DNS, un progetto di Kubernetes Incubator.

Un'altra opzione è sfruttare le capacità di AWS Cloud Map. AWS Cloud Map estende le funzionalità delle API Auto Naming fornendo un registro di servizio per risorse, come Internet Protocol (IP), Uniform Resource Locator (URL) e Amazon Resource Name (ARN), e offrendo un meccanismo di individuazione dei servizi basato su API con una propagazione delle modifiche più rapida e la possibilità di utilizzare gli attributi per restringere ulteriormente l'insieme delle risorse rilevate. Le risorse Auto Naming di Route 53 esistenti vengono aggiornate ad AWS Cloud Map in modo automatico.

Software di terze parti

Un diverso approccio per implementare il rilevamento dei servizi è l'utilizzo di software di terze parti come HashiCorp Consul, etcd e Netflix Eureka. I tre servizi citati sono archivi chiave-valore distribuiti affidabili. Per HashiCorp Consul è anche disponibile una Guida rapida di AWS che permette di impostare un ambiente AWS Cloud flessibile e scalabile e di avviare automaticamente HashiCorp Consul con una configurazione personalizzata.

Mesh di servizi

In un'architettura di microservizi avanzata, l'applicazione effettiva può essere composta da centinaia o addirittura migliaia di servizi. In mote situazioni, la parte più complessa dell'applicazione non è rappresentata dai servizi stessi, quanto piuttosto dalla comunicazione tra loro. Le mesh di servizi sono un ulteriore livello per la gestione della comunicazione tra servizi, che è responsabile del monitoraggio e del controllo del traffico nelle architetture di microservizi. Ciò consente ad attività come l'individuazione dei servizi di essere gestite completamente da tale livello.

In genere, una mesh di servizio viene suddivisa in un piano dati e un piano di controllo. Il piano dati è costituito da una serie di proxy intelligenti che vengono distribuiti con il codice dell'applicazione come proxy sidecar speciale che intercetta tutte le comunicazioni di rete tra microservizi. Il piano di controllo è responsabile della comunicazione con i proxy.

Le mesh di servizi offrono trasparenza, il che significa che gli sviluppatori di applicazioni non devono essere necessariamente a conoscenza di questo livello aggiuntivo e non viene loro richiesto di apportare modifiche al codice dell'applicazione esistente. AWS App Mesh è una mesh di servizi che offre reti a livello di applicazione per consentire la comunicazione tra servizi tra più tipi di infrastrutture di calcolo. App Mesh standardizza la modalità di comunicazione dei servizi, offrendo visibilità end-to-end e garantendo un'alta disponibilità alle applicazioni.

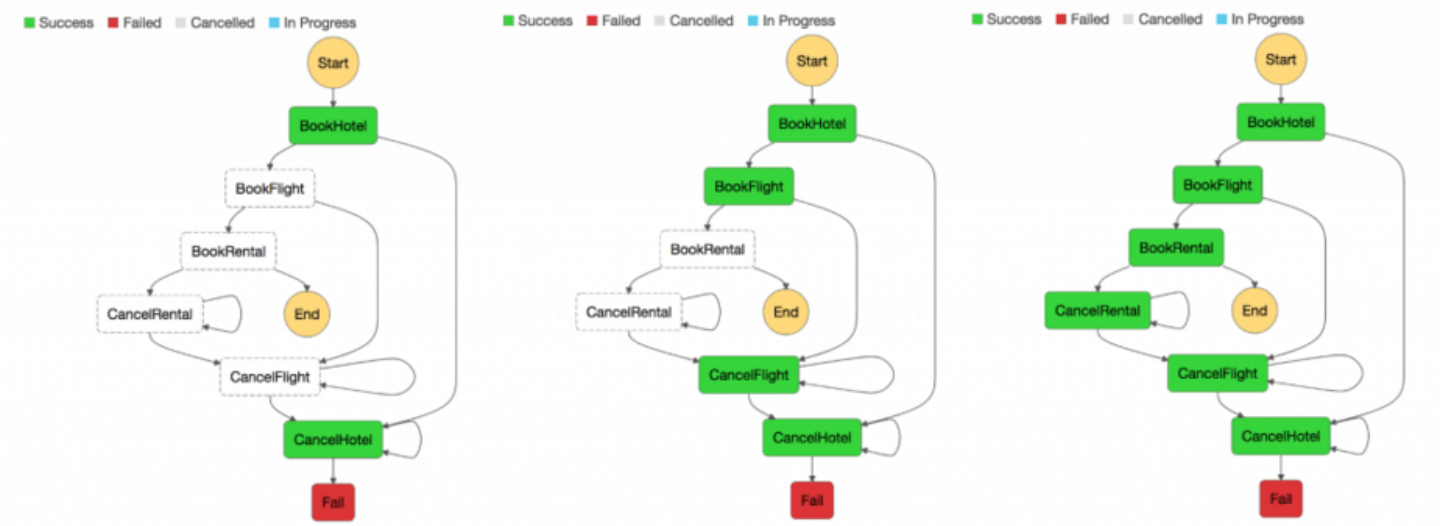
È possibile utilizzare AWS App Mesh con microservizi esistenti o nuovi in esecuzione su AWS Fargate, Amazon ECS, Amazon EKS e Kubernetes autogestiti su AWS. App Mesh può monitorare e controllare le comunicazioni di microservizi in esecuzione su cluster, sistemi di orchestrazione o VPC come una singola applicazione senza dover apportare alcuna modifica al codice.

Gestione di dati distribuiti

Le applicazioni monolitiche, in genere, si basano su un database relazionale di grandi dimensioni, che definisce un singolo modello di dati da applicare a tutti i componenti dell'applicazione. Quando si sceglie di utilizzare i microservizi, un database centralizzato impedirebbe la creazione di componenti indipendenti e decentralizzati. Ogni componente di un microservizio deve disporre del proprio livello di persistenza.

La gestione di dati distribuiti però presenta nuove problematiche. Come conseguenza del teorema CAP, le architetture di microservizi distribuite, per loro natura, sacrificano la consistenza in favore delle prestazioni e sono costrette a fare affidamento sulla consistenza finale.

In un sistema distribuito, le transazioni di business possono comprendere più microservizi. Poiché non possono creare una singola transazione ACID, è possibile che il processo termini con delle esecuzioni parziali. In questo caso, è richiesta una logica di controllo per la ripetizione di transazioni già elaborate. A tale scopo, viene comunemente utilizzato il modello Saga distribuito. Nel caso di una transazione di business non andata a buon fine, il modello Saga orchestra una serie di transazioni compensative che annullano le modifiche apportate dalle transazioni precedenti. AWS Step Functions semplifica l'implementazione di un coordinatore dell'esecuzione del modello Saga, come illustrato nella figura di seguito.



Coordinatore dell'esecuzione del modello Saga

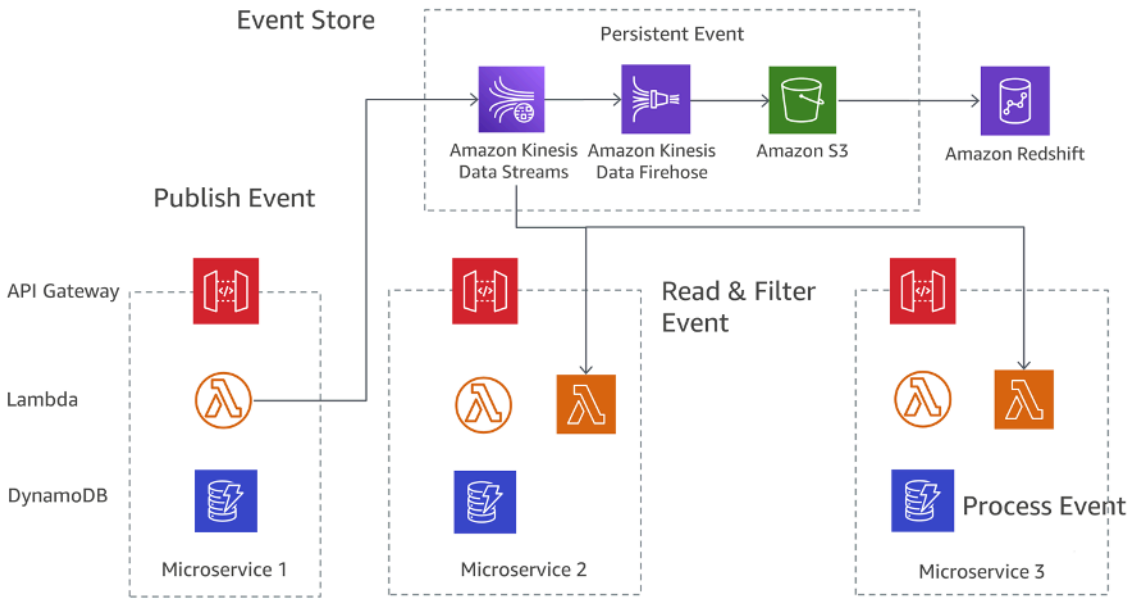
La creazione di un archivio centralizzato per i dati di riferimento critici, governato da strumenti e procedure di gestione dei dati master, offre la possibilità ai microservizi di sincronizzare i dati più importanti ed, eventualmente, di eseguire un rollback. Utilizzando Lambda con una pianificazione di Amazon CloudWatch Events, è possibile realizzare un semplice meccanismo di pulizia e deduplicazione.

Le modifiche di stato molto spesso interessano più di un microservizio. In questi casi, l'event sourcing si è rilevato uno strumento molto prezioso. L'idea di base è rappresentare e consolidare le modifiche di ciascuna applicazione come record di evento. Invece di consolidare uno stato, le informazioni sono memorizzate sotto forma di flussi di eventi. Due esempi molto frequenti di event sourcing sono la registrazione dei log di database e i sistemi di controllo delle versioni. L'event sourcing offre alcuni vantaggi evidenti: gli stati possono essere determinati e ricostruiti in uno storico preciso. Pertanto, produce intrinsecamente un audit trail persistente e semplifica le operazioni di debug.

Nell'ambito delle architetture di microservizi, questo modello permette di disaccoppiare le varie parti di un'applicazione utilizzando pattern pub/sub e inviando gli stessi dati di evento in diversi modelli di dati per diversi microservizi. L'event sourcing viene spesso utilizzato insieme ai pattern CQRS (Command, Query, Responsibility, Segregation) per separare i carichi di lavoro di lettura da quelli di scrittura e ottimizzare prestazioni, scalabilità e sicurezza di entrambi. Nei sistemi di gestione dei dati tradizionali, i comandi e le query sono eseguiti sugli stessi repository di dati.

La figura seguente mostra come implementare il modello di event sourcing degli eventi in AWS. Amazon Kinesis Data Streams funge da elemento principale dello storage di eventi centralizzato che acquisisce le modifiche apportate all'applicazione sotto forma di eventi e li archivia in Amazon

S3. La figura illustra tre diversi microservizi composti da Amazon API Gateway, AWS Lambda e Amazon DynamoDB. Le frecce blu indicano il flusso degli eventi: quando il Microservizio 1 rileva una variazione dello stato di un evento, pubblica un evento scrivendo un messaggio in Kinesis Data Streams. Tutti i microservizi eseguono in AWS Lambda una propria applicazione Kinesis Data Streams che legge una copia del messaggio, lo filtra in base alla rilevanza per il microservizio e, se necessario, lo inoltra ad altri servizi per ulteriore elaborazione. Se la funzione restituisce un errore, Lambda effettua nuovi tentativi sui batch finché l'elaborazione non va a buon fine o fino alla scadenza dei dati. Per evitare partizioni bloccate, è possibile configurare la mappatura dell'origine eventi in modo che effettui un nuovo tentativo con una dimensione di batch ridotta, limitare il numero di tentativi o eliminare record troppo vecchi. Per mantenere gli eventi scartati, è possibile configurare la mappatura dell'origine eventi per inviare i dettagli sui batch non correttamente elaborati su una coda Amazon Simple Queue Service (Amazon SQS) o verso un argomento Amazon Simple Notification Service (Amazon SNS).



Modello di sourcing degli eventi attivo su AWS

Amazon S3 memorizza in modo permanente tutti gli eventi di tutti i microservizi e costituisce il SSOT (Single Source Of Truth) per quanto riguarda debug, ripristino dello stato dell'applicazione e verifica delle modifiche. Ci sono due motivi principali per cui i record possono essere distribuiti più di una volta all'applicazione Kinesis Data Streams: nuovi tentativi da parte del produttore e nuovi tentativi da parte del consumatore. Un'applicazione deve prevedere e gestire in modo appropriato l'elaborazione multipla di singoli record.

Gestione della configurazione

In una tipica architettura di microservizi con decine di servizi diversi, ogni servizio deve accedere a diversi servizi a valle e componenti dell'infrastruttura che espongono dati al servizio. Esempi potrebbero essere code di messaggi, database e altri microservizi. Una delle sfide principali è configurare ogni servizio in modo coerente per fornire informazioni sulla connessione ai servizi e all'infrastruttura a valle. Inoltre, la configurazione dovrebbe contenere anche informazioni sull'ambiente in cui opera il servizio e non dovrebbe essere necessario riavviare l'applicazione per utilizzare nuovi dati di configurazione.

Il terzo principio dei modelli Twelve-Factor App copre questo argomento: "L'app Twelve-Factor memorizza la configurazione nelle variabili di ambiente (spesso abbreviate in env vars o env)". Per Amazon ECS, le variabili di ambiente possono essere passate al container utilizzando il parametro ambiente di definizione del container che viene mappato sull'opzione di esecuzione docker `--env`. Le variabili di ambiente possono essere passate ai container in blocco utilizzando il parametro di definizione del container `environmentFiles` per elencare uno o più file contenenti le variabili di ambiente. Il file deve essere ospitato in Amazon S3. In AWS Lambda, il runtime rende le variabili di ambiente disponibili per il codice e imposta variabili di ambiente aggiuntive che contengono informazioni sulla funzione e sulla richiesta di invocazione. Per Amazon EKS, è possibile definire le variabili di ambiente nel campo env del manifesto di configurazione del pod corrispondente. Un modo diverso di usare le variabili env è sfruttare una ConfigMap.

Comunicazione asincrona e messaggistica leggera

Nelle applicazioni monolitiche, in genere, la comunicazione è molto semplice: una sezione dell'applicazione utilizza chiamate di metodo o meccanismi interni di distribuzione di eventi per comunicare con le altre sezioni. Se una medesima applicazione viene implementata tramite microservizi separati, la comunicazione tra le diverse sezioni deve avvenire tramite rete.

Comunicazione basata su REST

Il protocollo HTTP/S è il più utilizzato per allestire comunicazione sincrona tra microservizi. Nella maggior parte dei casi, le API RESTful adottano come layer di trasporto il protocollo HTTP. Il modello di architettura REST è basato su comunicazione stateless, interfacce uniformi e metodi standard.

Con API Gateway, è possibile creare un'API che funga da porta d'ingresso delle applicazioni per l'accesso a dati, logica di business o funzionalità dei servizi di back-end. Gli sviluppatori di API

possono creare API che accedono ad AWS o ad altri servizi web, nonché ai dati archiviati in AWS Cloud. Un oggetto API definito con API Gateway è costituito da un gruppo di risorse e metodi.

Una risorsa è un oggetto con tipo che fa parte del dominio dell'API e può essere associato a un modello di dati o a relazioni con altre risorse. Ciascuna risorsa può essere configurata in modo da rispondere a uno o più metodi, ovvero verbi HTTP standard quali GET, POST e PUT. Le API REST possono essere implementate in diverse fasi, generando nuove versione come evoluzioni o cloni di versioni precedenti.

API Gateway gestisce tutte le attività di accettazione ed elaborazione relative a centinaia di migliaia di chiamate API simultanee, inclusi gestione del traffico, controllo di accessi e autorizzazioni, monitoraggio e gestione delle versioni delle API.

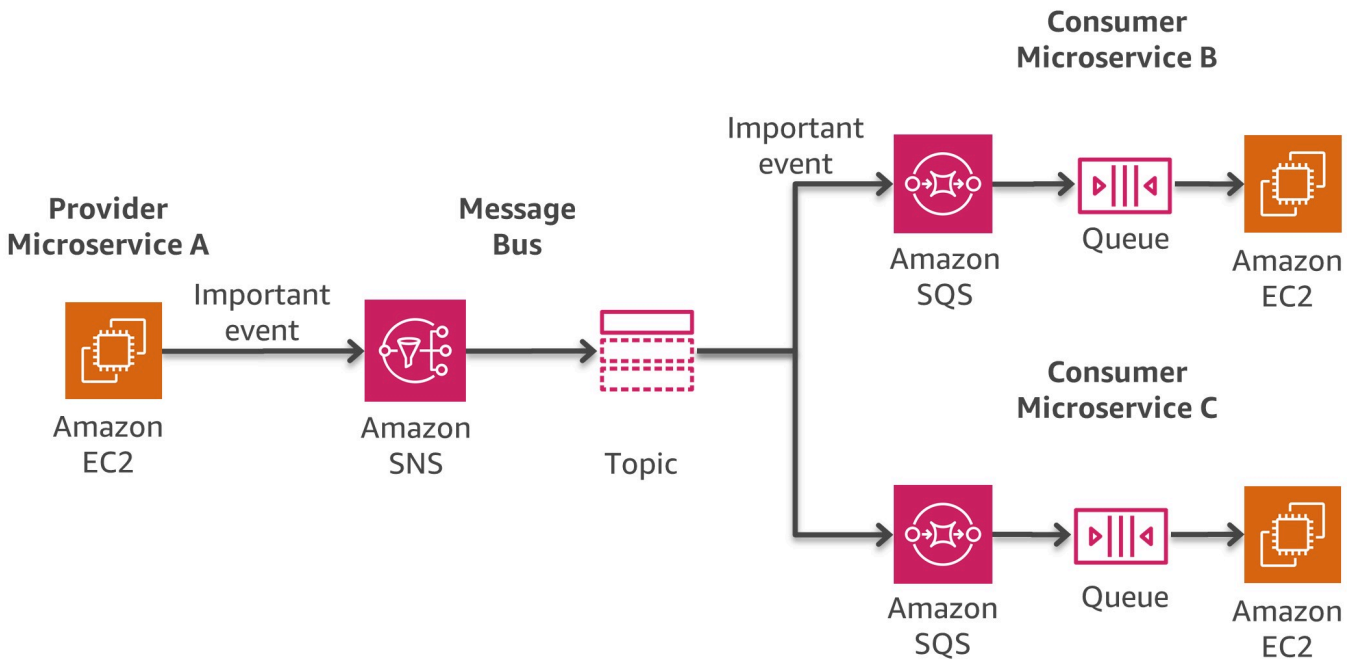
Messaggistica asincrona e trasferimento degli eventi

Un altro pattern utilizzato per implementare le comunicazioni tra microservizi è il trasferimento di messaggi. I servizi comunicano scambiandosi messaggi tramite code. Uno dei vantaggi di questo metodo è che non è necessario implementare l'individuazione di servizi e che questi ultimi sono accoppiati in modo non vincolante.

I sistemi sincroni sono accoppiati in modo stretto, il che significa che un problema in una dipendenza sincrona a valle ha un impatto immediato sul chiamante a monte. I tentativi da parte dei chiamanti a monte possono rapidamente far emergere e amplificare i problemi.

A seconda dei requisiti specifici, come i protocolli, AWS offre diversi servizi che aiutano a implementare questo modello. Un'implementazione possibile usa una combinazione di coda Amazon Simple Queue Service (Amazon SQS) e Amazon Simple Notification Service (Amazon SNS).

Entrambi i servizi funzionano in sinergia: Amazon SNS consente alle applicazioni di inviare messaggi a più sottoscrittori tramite un meccanismo push. L'utilizzo di Amazon SNS e Amazon SQS permette di inoltrare un singolo messaggio a più consumatori. La figura seguente illustra l'integrazione di Amazon SNS e Amazon SQS.



Pattern di bus di messaggio in AWS

Quando si sottoscrive una coda SQS a un argomento SNS, è possibile pubblicare un messaggio sull'argomento e Amazon SNS invia un messaggio alla coda Amazon SQS sottoscritta. Tali messaggi conterranno l'oggetto e il messaggio pubblicati nell'argomento e le informazioni sui metadati in formato JSON.

Un'altra opzione per la creazione di architetture basate su eventi con origini di eventi che abbracciano applicazioni interne, applicazioni SaaS di terze parti e servizi AWS, su larga scala, è Amazon EventBridge. Un servizio di bus eventi completamente gestito, EventBridge riceve eventi da diverse fonti, identifica una destinazione in base a una regola di routing e fornisce dati quasi in tempo reale a tale destinazione, incluse, tra le altre, AWS Lambda, Amazon SNS e Amazon Kinesis Streams. Un evento in ingresso può anche essere personalizzato, tramite la trasformazione di ingresso, prima della distribuzione.

Per sviluppare applicazioni basate su eventi in modo significativamente più veloce, i registri degli schemi di EventBridge raccolgono e organizzano gli schemi, inclusi gli schemi per tutti gli eventi generati dai servizi AWS. I clienti possono anche definire schemi personalizzati o utilizzare l'opzione dello schema di deduzione per scoprire automaticamente gli schemi. Nel complesso, tuttavia, un potenziale compromesso per tutte queste funzionalità è un valore di latenza relativamente più elevato

per la consegna da parte di EventBridge. Inoltre, il throughput e le quote di default per EventBridge potrebbero richiedere un incremento, tramite una richiesta di supporto, in base al caso d'uso.

Una diversa strategia di implementazione si basa su Amazon MQ, che può essere utilizzato se il software esistente utilizza API e protocolli standard aperti per la messaggistica, tra cui JMS, NMS, AMQP, STOMP, MQTT e WebSocket. Amazon SQS espone un'API personalizzata, il che significa che se si dispone di un'applicazione esistente da cui si desidera migrare, ad esempio da un ambiente on-premise ad AWS, sono necessarie modifiche al codice. In molti casi, con Amazon MQ questo non è necessario.

Amazon MQ gestisce l'amministrazione e la manutenzione di ActiveMQ, un noto broker di messaggi open source. L'infrastruttura sottostante viene automaticamente adattata per la disponibilità elevata e la persistenza dei messaggi in modo da garantire l'affidabilità delle applicazioni.

Gestione stato e orchestrazione

La natura distribuita dei microservizi rende difficoltosa l'orchestrazione dei flussi di lavoro che ne coinvolgono un numero significativo. Gli sviluppatori potrebbero ritenere più conveniente aggiungere il codice di orchestrazione direttamente nei servizi. Si tratta però di un errore da evitare, perché introdurrebbe un accoppiamento stretto e renderebbe complicato sostituire rapidamente singoli servizi.

Per realizzare applicazioni a partire da singoli componenti, ciascuno dei quali esegue una funzione ben definita, è possibile utilizzare AWS Step Functions. Questo servizio offre macchine a stati che si prendono carico delle complessità correlate all'orchestrazione dei servizi, ad esempio la gestione degli errori e la divisione in fasi eseguite in serie o in parallelo. Grazie a questa soluzione, sarà più semplice ricalibrare le risorse o sostituire applicazioni con la massima rapidità senza dover sincronizzare il codice all'interno dei servizi.

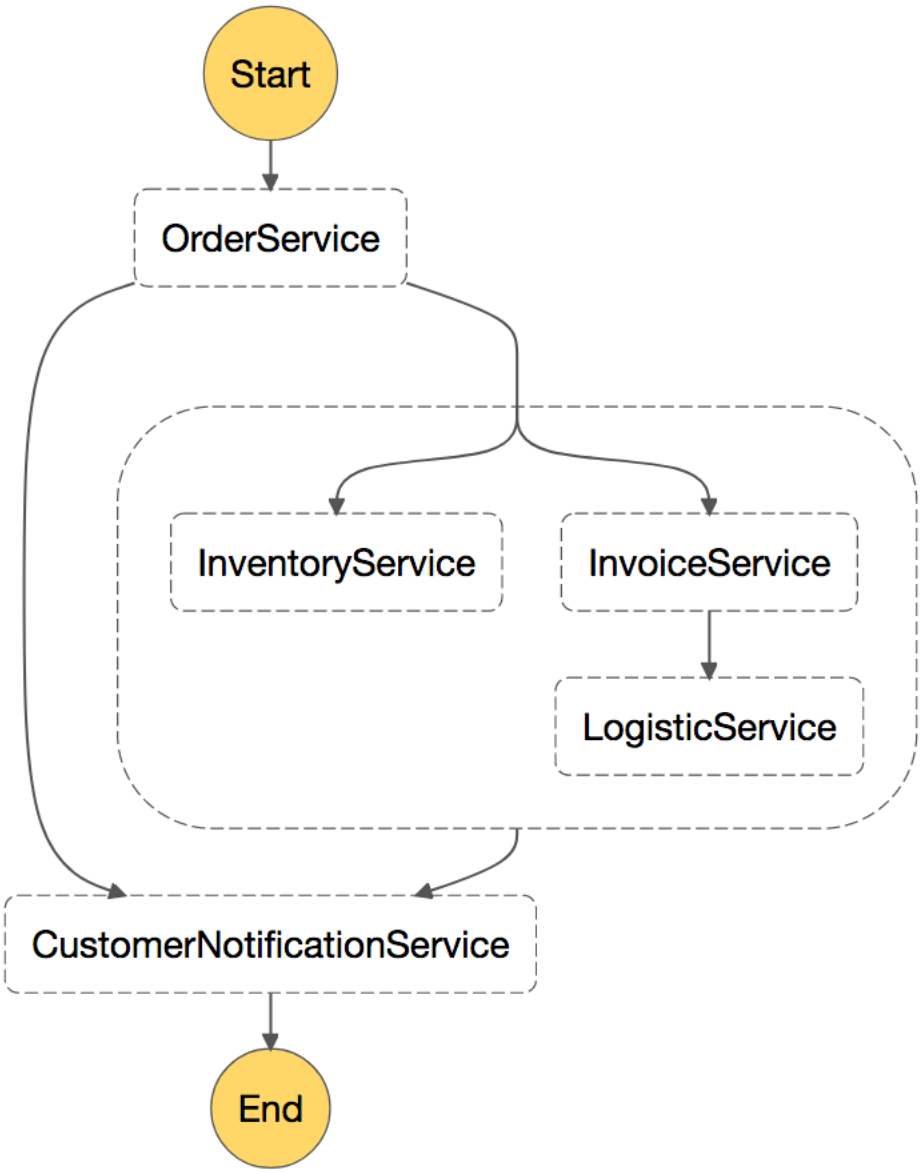
Step Functions è uno strumento affidabile per coordinare componenti e controllare in ogni fase la funzionalità di un'applicazione. Questo servizio dispone di una console grafica in cui è possibile disporre e visualizzare i componenti dell'applicazione in una serie di fasi. Di conseguenza, i servizi distribuiti sono più facili da realizzare ed eseguire.

Step Functions attiva e tiene traccia automaticamente di ogni fase e, in caso di errore, esegue nuovi tentativi per garantire che l'applicazione venga sempre eseguita nell'ordine e nelle modalità previste. Questo servizio registra lo stato di ciascuna fase, perciò quando si verificano problemi è più semplice eseguire diagnosi e debug in modo rapido. È possibile modificare o aggiungere fasi senza scrivere una riga di codice per innovare più rapidamente l'applicazione.

Step Functions è parte della AWS Serverless Platform e supporta l'orchestrazione delle funzioni Lambda e delle applicazioni basate su risorse di calcolo, come Amazon EC2, Amazon EKS, Amazon ECS e servizi aggiuntivi come Amazon SageMaker e AWS Glue. Step Functions gestisce il funzionamento e l'infrastruttura in modo automatico e garantisce la disponibilità dell'applicazione su qualsiasi scala.

Per la realizzazione dei flussi di lavoro, Step Functions impiega la sintassi Amazon State Language. I flussi di lavoro possono contenere fasi in parallelo o in sequenza e diramazioni.

La seguente figura illustra un esempio di flusso di lavoro per un'architettura di microservizi che combina fasi in sequenza e in parallelo. Per richiamare tale flusso di lavoro è possibile utilizzare l'API Step Functions o passare da API Gateway.



Un esempio di flusso di lavoro di microservizi richiamato da Step Functions

Monitoraggio distribuito

Un'architettura di microservizi è composta da diversi componenti distribuiti che devono essere monitorati. È possibile utilizzare Amazon CloudWatch per raccogliere e monitorare parametri, accentrare e monitorare file di log, impostare avvisi e rispondere automaticamente alle variazioni nell'ambiente AWS. CloudWatch consente il monitoraggio di risorse AWS quali le istanze Amazon EC2, le tabelle di DynamoDB e le istanze database di Amazon RDS, nonché i parametri personalizzati generati dalle applicazioni e dai servizi del cliente e i file di log generati dalle applicazioni.

Monitoraggio

CloudWatch può essere impiegato per ottenere visibilità a livello di sistema su utilizzo delle risorse, prestazioni delle applicazioni e stato di integrità operativa. Il servizio garantisce una soluzione di monitoraggio affidabile, scalabile e flessibile pronta all'uso in pochi minuti. Non sarà più necessario realizzare, gestire e ricalibrare le risorse di un'infrastruttura e di sistemi di monitoraggio personalizzati. Nell'ambito delle infrastrutture di microservizi, la possibilità di monitorare parametri personalizzati con CloudWatch è un ulteriore vantaggio per gli sviluppatori, perché potranno decidere quali parametri raccogliere per ciascun servizio. Inoltre, è possibile implementare la scalabilità dinamica in base a tali parametri personalizzati.

Oltre ad Amazon CloudWatch, è anche possibile utilizzare CloudWatch Container Insights per raccogliere, aggregare e riepilogare parametri e log da applicazioni e microservizi containerizzati. CloudWatch Container Insights raccoglie automaticamente i parametri per molte risorse, come CPU, memoria, disco e rete, e li aggrega come parametri CloudWatch a livello di cluster, nodo, pod, attività e servizio. Utilizzando CloudWatch Container Insights, è possibile accedere ai parametri del pannello di controllo di CloudWatch Container Insights. Fornisce inoltre informazioni diagnostiche, ad esempio errori di riavvio del container, che consentono di isolare i problemi e risolverli in modo rapido. È anche possibile impostare avvisi CloudWatch sui parametri raccolti da Container Insights.

Container Insights è disponibile per le piattaforme Amazon ECS, Amazon EKS e Kubernetes su Amazon EC2. Il supporto Amazon ECS include il supporto per Fargate.

Un'altra opzione comune, specialmente per Amazon EKS, è l'utilizzo di Prometheus. Prometheus è un kit di strumenti di monitoraggio e notifica open source che viene spesso utilizzato in combinazione

con Grafana per visualizzare i parametri raccolti. Molti componenti di Kubernetes memorizzano parametri su `/metrics` e Prometheus può prelevare tali parametri a intervalli regolari.

Amazon Managed Service for Prometheus (AMP) è un servizio di monitoraggio compatibile con Prometheus che semplifica il monitoraggio delle applicazioni containerizzate in modo sicuro e su larga scala. Con AMP, è possibile utilizzare il linguaggio open source per le query Prometheus (PromQL) per monitorare le prestazioni dei carichi di lavoro containerizzati senza dover gestire l'infrastruttura sottostante necessaria per gestire l'importazione dati, l'archiviazione e le query dai parametri operativi. È possibile raccogliere i parametri Prometheus dagli ambienti Amazon EKS e Amazon ECS, utilizzando AWS Distro per OpenTelemetry o i server Prometheus come agenti di raccolta.

AMP viene spesso utilizzato in combinazione con Amazon Managed Service for Grafana (AMG). AMG semplifica le query, la visualizzazione, gli avvisi e la comprensione dei parametri indipendentemente da dove vengono archiviati. Con AMG, è possibile analizzare i parametri, registri e tracciamenti senza dover effettuare il provisioning dei server, configurare e aggiornare il software o occuparsi del pesante lavoro di assicurare e scalare Grafana nella fase di produzione.

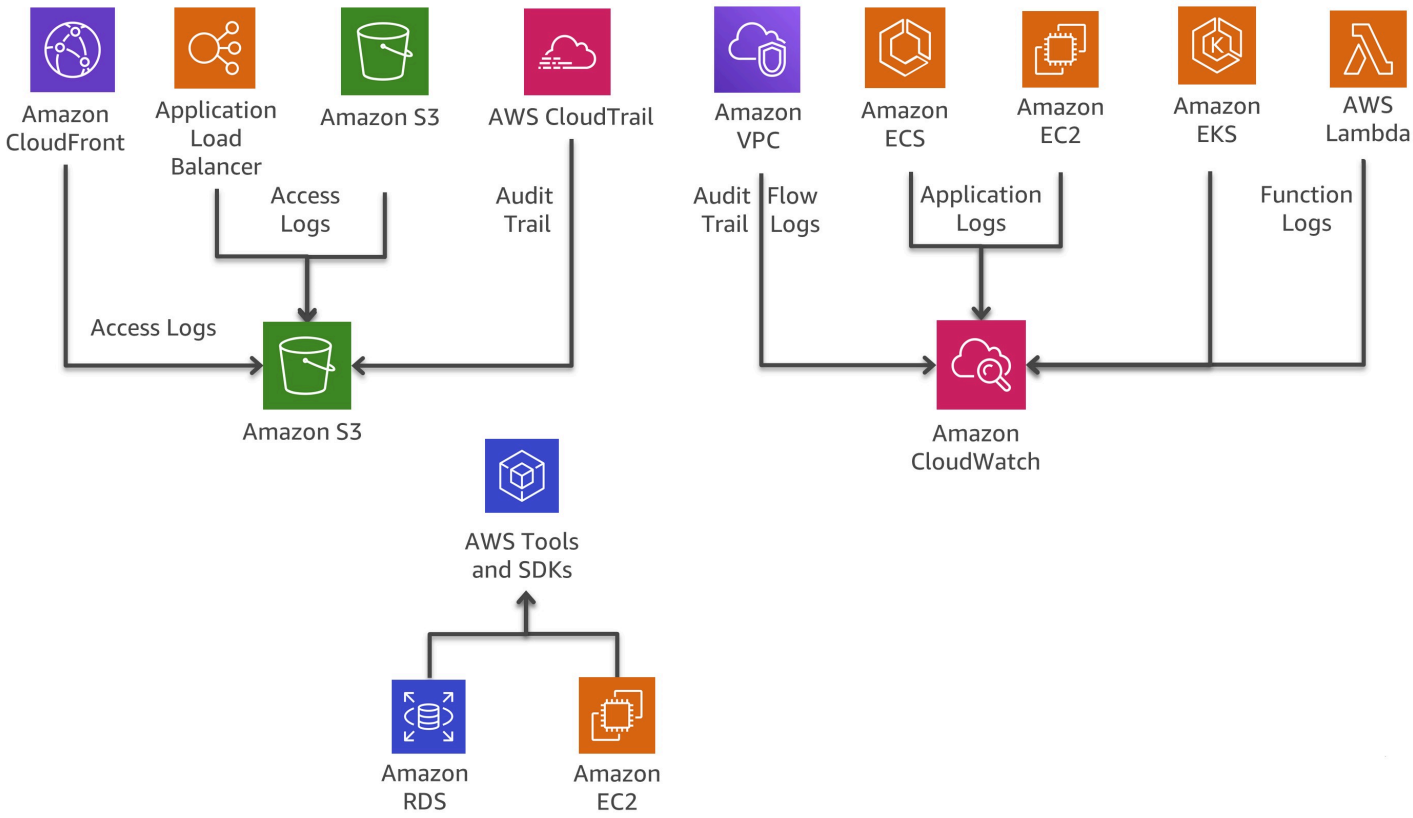
Raccolta centralizzata dei log

Raccogliere i log in modo uniforme è di importanza critica per l'identificazione e la risoluzione dei problemi. I microservizi permettono di produrre un numero maggiore di release incoraggiando i tecnici a sperimentare nuove caratteristiche in produzione. Analizzare l'impatto sui clienti è un'operazione fondamentale per migliorare in modo graduale un'applicazione.

Per impostazione di default, la maggior parte dei servizi AWS centralizza i propri file di log. La destinazione principale per questi log su AWS sono Amazon S3 e Amazon CloudWatch Logs. Per le applicazioni in esecuzione su istanze Amazon EC2, è anche disponibile un daemon che invia automaticamente i file di log in CloudWatch Logs. Le funzioni Lambda inviano in modo nativo i log in CloudWatch Logs e Amazon ECS supporta il driver log awslogs che permette il salvataggio centralizzato dei log dei container nello stesso servizio. Per Amazon EKS, Fluent Bit o FluentD possono inoltrare i log dalle singole istanze nel cluster a una registrazione centralizzata dei log di CloudWatch Logs, dove vengono combinati per creare report di livello superiore utilizzando Amazon OpenSearch Service e Kibana. Per il suo ingombro ridotto e per i vantaggi in termini di prestazioni, Fluent Bit è maggiormente consigliato rispetto a FluentD.

La figura seguente illustra le funzionalità di registrazione dei log di alcuni servizi. I diversi team potranno eseguire ricerche tra i log e analizzarli utilizzando strumenti quali Amazon OpenSearch

Service e Kibana. Amazon Athena è un servizio che può essere utilizzato per eseguire query ad hoc su file di log in Amazon S3.



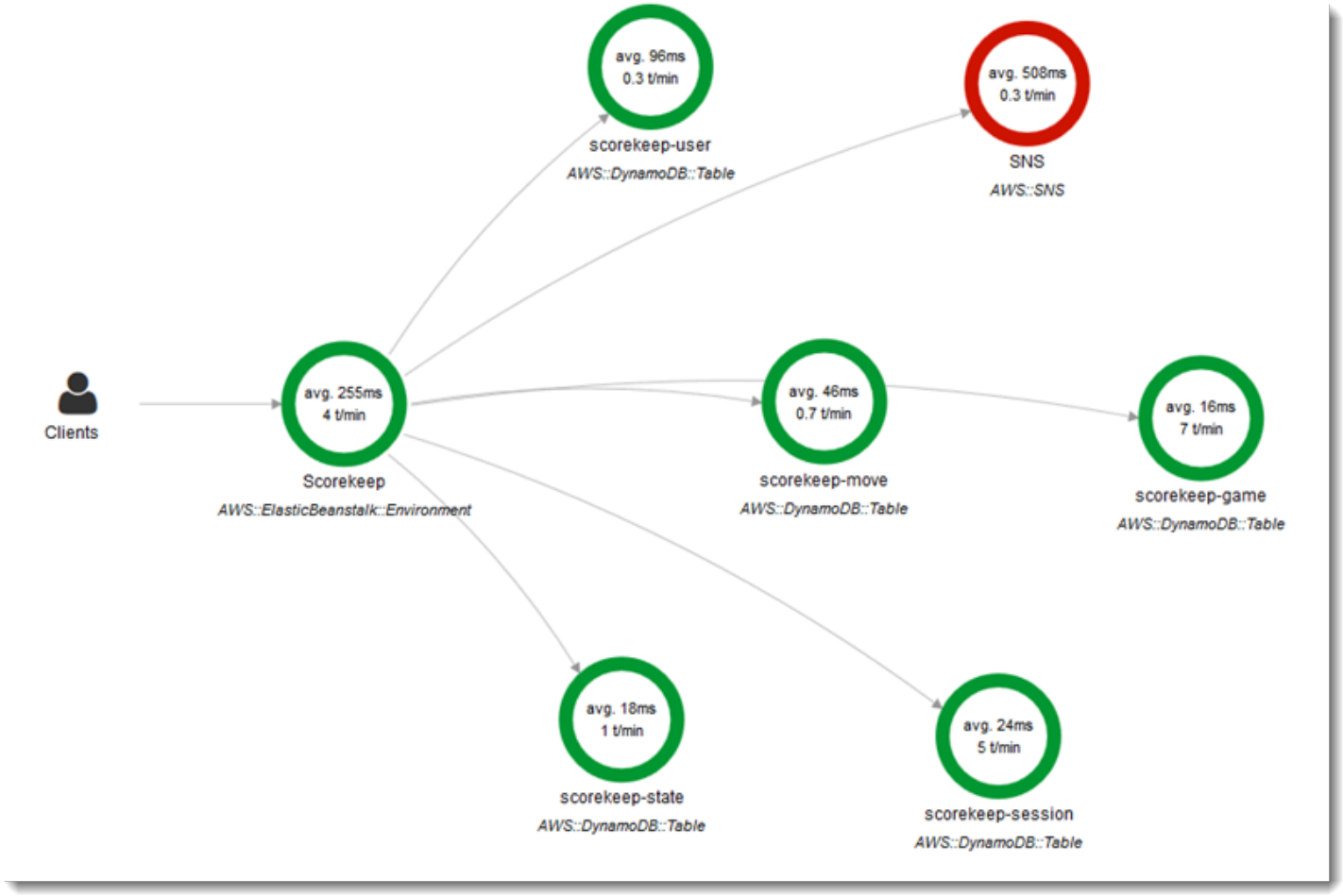
Funzionalità di registrazione di log dei servizi AWS

Tracciamento distribuito

In molti casi, sono chiamati a gestire una singola richiesta diversi microservizi. Supponiamo di avere un sistema complesso composto da decine di microservizi; in questo sistema si verifica un errore in uno dei servizi nella catena di chiamate. Anche se ciascuno dei microservizi registra in modo corretto i propri log e li consolida in un sistema centralizzato, può risultare difficoltoso trovare tutti i messaggi di log relativi all'errore.

Il concetto su cui si basa AWS X-Ray è l'utilizzo di ID di correlazione, identificatori univoci allegati a tutte le richieste e i messaggi delle singole catene di eventi. Nel momento in cui la richiesta raggiunge il primo servizio che si integra con X-Ray, (ad esempio Application Load Balancer o API Gateway), un ID di tracciamento viene aggiunto alle richieste HTTP all'interno di intestazioni apposite denominate `X-Amzn-Trace-Id` e viene incluso nella risposta. Utilizzando il SDK X-Ray, ciascun microservizio è in grado di leggere e aggiornare tale intestazione.

X-Ray funziona con Amazon EC2, Amazon ECS, Lambda e AWS Elastic Beanstalk. Può essere utilizzato con tutte le applicazioni implementate su tali servizi scritte in Java, Node.js e .NET.



Mappatura dei servizi di AWS X-Ray

Epsagon è un SaaS completamente gestito che include il tracciamento di tutti i servizi AWS, API di terze parti (tramite chiamate HTTP) e altri servizi comuni come Redis, Kafka ed Elastic. Il servizio Epsagon include funzionalità di monitoraggio, avvisi sui servizi più comuni e visibilità del payload in ogni chiamata effettuata dal codice.

AWS Distro for OpenTelemetry è una distribuzione del progetto OpenTelemetry sicura, pronta per la produzione e supportata da AWS. AWS Distro for OpenTelemetry fa parte della Cloud Native Computing Foundation e offre API open source, librerie e agenti per la raccolta di tracciamenti e parametri distribuiti per il monitoraggio delle applicazioni. Con AWS Distro for OpenTelemetry è possibile puoi dotare le applicazioni dei componenti necessari un'unica volta per inviare parametri e tracciamenti correlati a molteplici soluzioni di monitoraggio di AWS e partner. Utilizzare gli agenti di strumentazione automatica per raccogliere i tracciamenti senza modifiche al codice. AWS Distro for

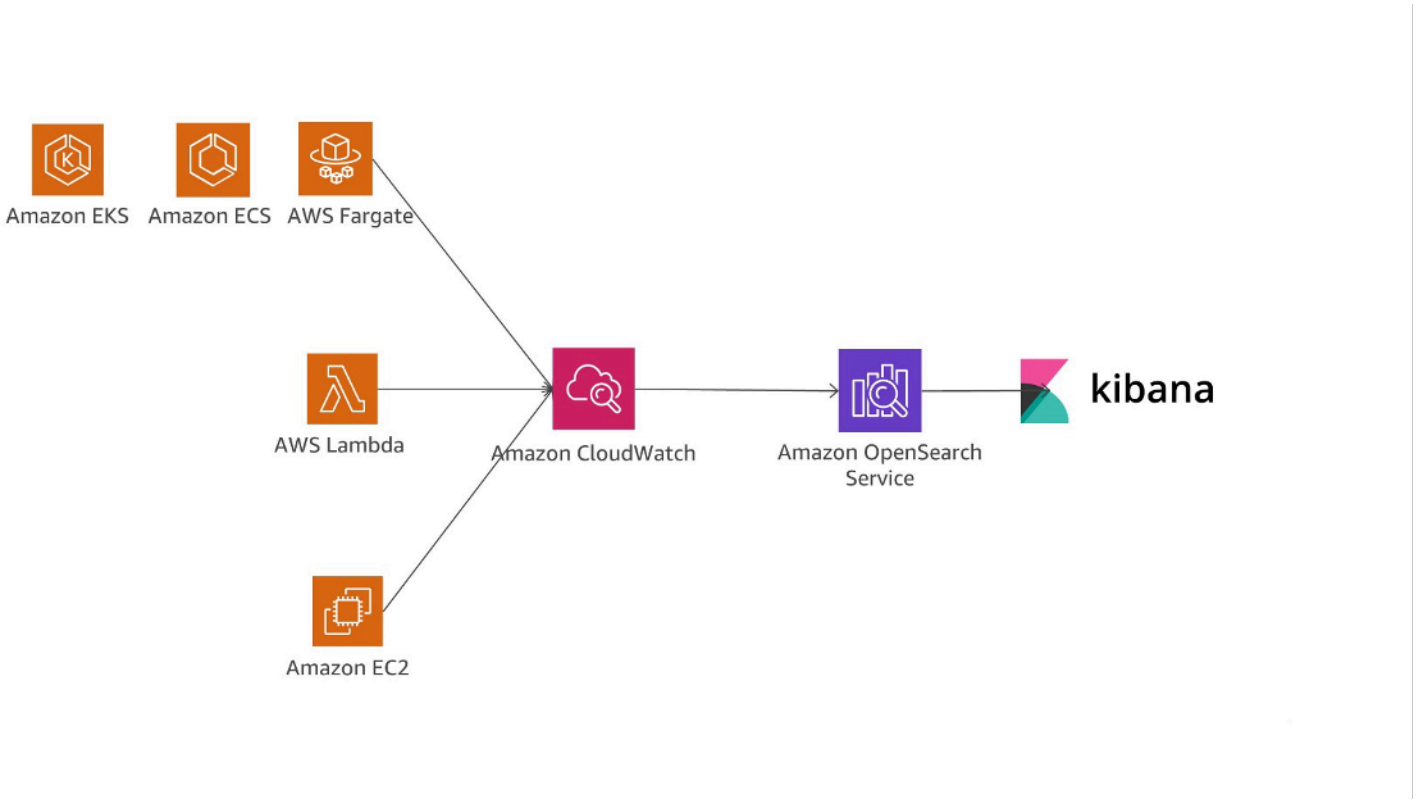
OpenTelemetry raccoglie anche i metadati dalle risorse e dai servizi gestiti di AWS. In questo modo è possibile correlare i dati delle prestazioni delle applicazioni con i dati dell'infrastruttura sottostante, riducendo il tempo di risoluzione del problema. Utilizzare AWS Distro per OpenTelemetry per dotare dei componenti necessari le applicazioni in esecuzione su Amazon EC2, Amazon ECS, Amazon EKS su Amazon EC2, Fargate e AWS Lambda, oltre che on-premise.

Alternative per l'analisi di log in AWS

Per comprendere il funzionamento di un sistema distribuito è di vitale importanza visualizzare, ricercare e analizzare i dati di log. Amazon CloudWatch Logs Insights è un ottimo servizio per esplorare, analizzare e visualizzare i log all'istante. Ciò consente di risolvere i problemi operativi. Un altro metodo adottato per l'analisi di file di log è l'impiego di Amazon OpenSearch Service assieme a Kibana.

Amazon OpenSearch Service può essere impiegato per ricerche full text, ricerche strutturate, analisi dei dati e tutte e tre le operazioni contemporaneamente. Kibana è un plug-in di visualizzazione dei dati open source che si integra perfettamente con Amazon OpenSearch Service.

La figura seguente illustra l'analisi dei log con Amazon OpenSearch Service e Kibana. CloudWatch Logs può essere configurato per inoltrare flussi di elementi di log a Amazon OpenSearch Service quasi in tempo reale tramite una sottoscrizione a CloudWatch Logs. Kibana permette di visualizzare i dati e offre una comoda interfaccia di ricerca per i datastore in Amazon OpenSearch Service. Questa soluzione può anche essere impiegata insieme ad altri prodotti software, ad esempio ElastAlert, per implementare sistemi di avvisi che inviino notifiche SNS ed e-mail, creino ticket JIRA e così via ogni volta che i dati indicano la presenza di anomalie, picchi o altri pattern degni di interesse.



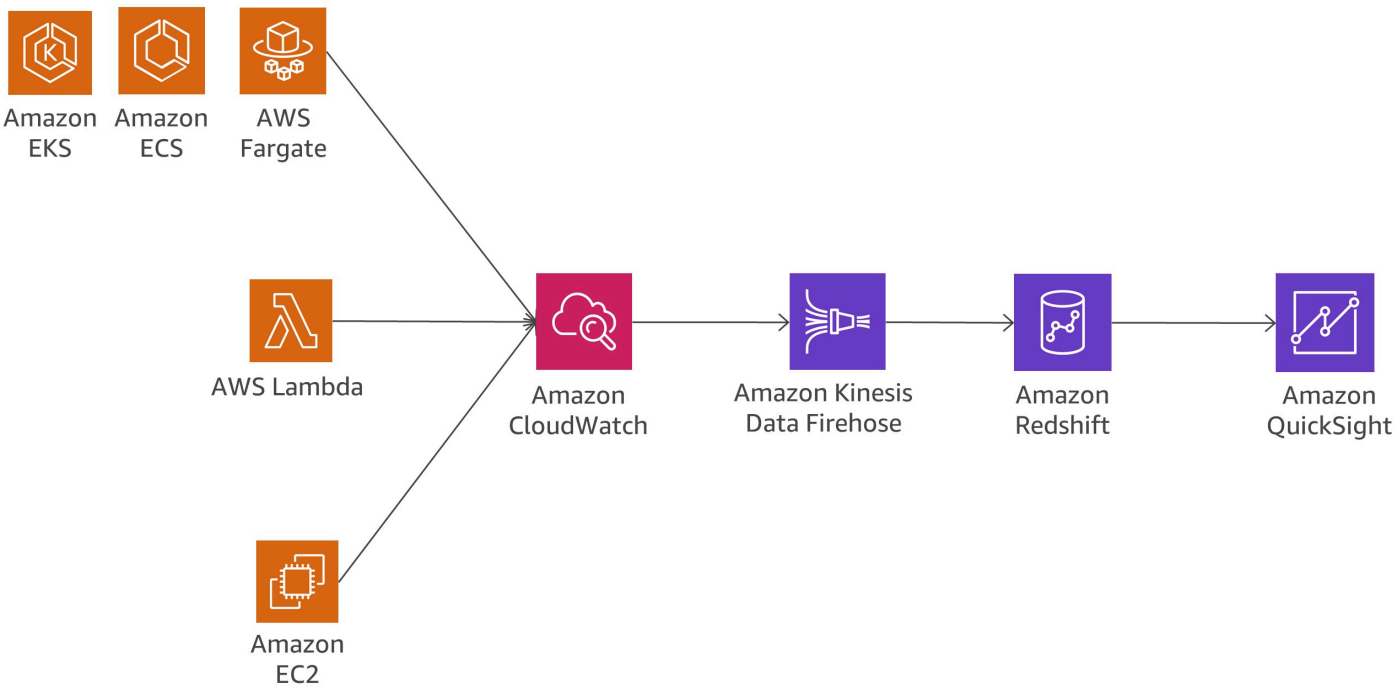
Analisi dei log con Amazon OpenSearch Service e Kibana

Un altro metodo adottato per l'analisi di file di log è l'impiego di Amazon Redshift con Amazon QuickSight.

Amazon QuickSight può essere facilmente collegato ai servizi dati AWS, tra cui Amazon Redshift, Amazon RDS, Amazon Aurora, Amazon EMR, DynamoDB, Amazon S3 e Amazon Kinesis.

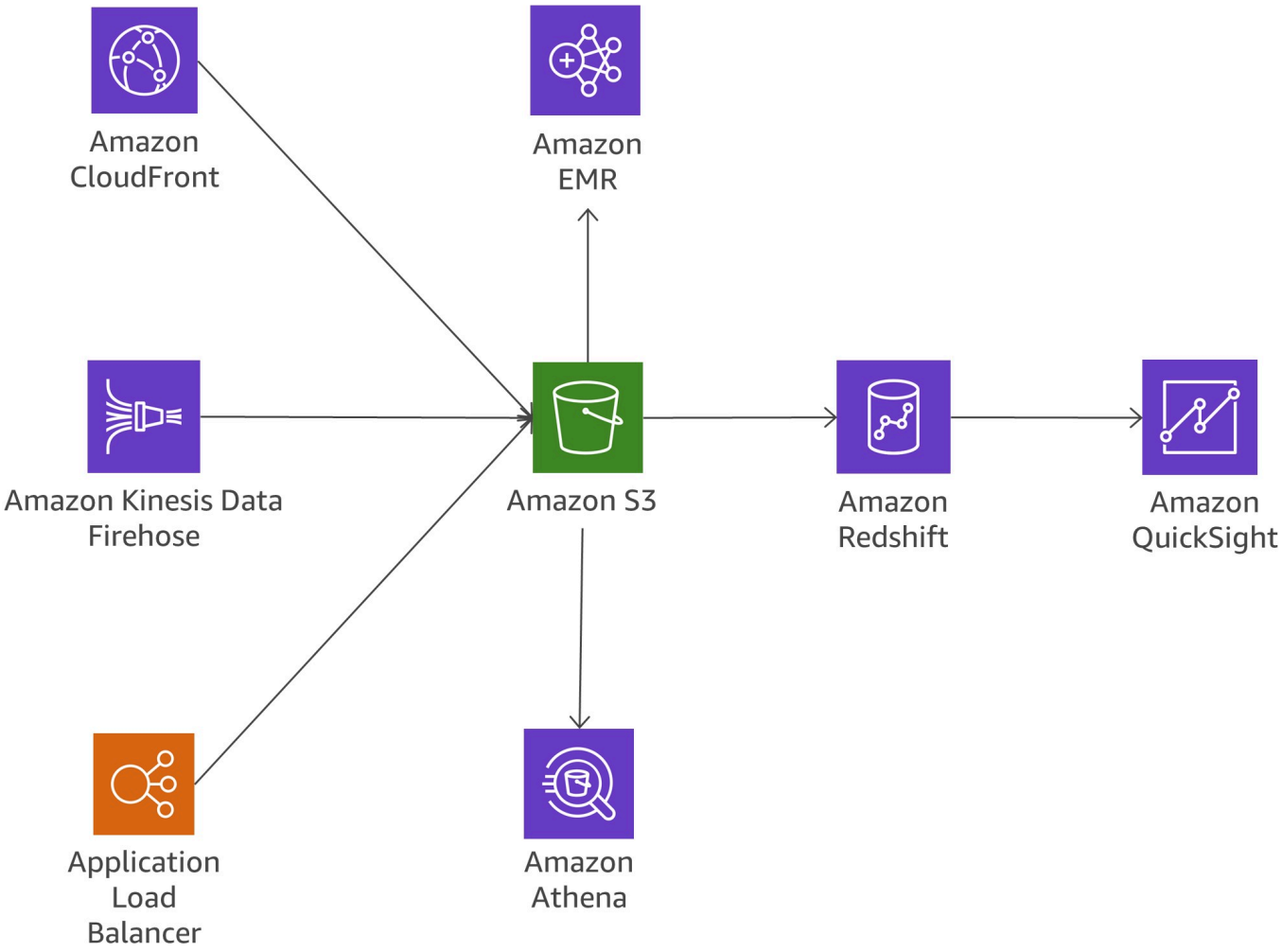
Amazon CloudWatch Logs può fungere da archivio centralizzato di log e, oltre alla semplice memorizzazione dei dati, è in grado di inoltrarli sotto forma di flussi su Amazon Kinesis Data Firehose.

La figura seguente descrive uno scenario in cui le voci di log vengono trasmesse da diverse origini in Amazon Redshift utilizzando CloudWatch Logs e Kinesis Data Firehose. Amazon QuickSight si avvale dei dati memorizzati in Amazon Redshift per analisi, reportistica e visualizzazione.



Analisi dei log con Amazon Redshift e Amazon QuickSight

La figura seguente mostra uno scenario di analisi dei log su Amazon S3. Quando i log vengono memorizzati in bucket Amazon S3, i relativi dati possono essere caricati su diversi servizi AWS, come ad esempio in Amazon Redshift e Amazon EMR, per analizzare il flusso di log e individuare eventuali anomalie.



Analisi di log in Amazon S3

Sovraccarico di comunicazioni

Scomponendo le applicazioni monolitiche in microservizi di piccole dimensioni, il carico di comunicazioni aumenta, perché ogni microservizio deve comunicare con tutti gli altri. In molte implementazioni, si opta per l'utilizzo di REST su HTTP poiché si tratta di un protocollo di comunicazione leggero. Tuttavia, volumi elevati di messaggi possono causare problemi. In alcuni casi, il consolidamento di servizi che scambiano grandi volumi di messaggi può essere un'opzione da prendere in considerazione. Se per ridurre il sovraccarico di comunicazioni il numero di servizi consolidati cresce eccessivamente, potrebbe essere il caso di rivalutare il proprio modello di dominio.

Protocolli

All'inizio di questo whitepaper, nella sezione the section called “Comunicazione asincrona e messaggistica leggera”, vengono discussi diversi protocolli possibili. In genere, per i microservizi si consiglia di utilizzare protocolli semplici come l'HTTP. I messaggi tra servizi possono tuttavia anche essere codificati, ad esempio in formati comprensibili come JSON e YAML o in formati binari più efficienti quali Avro e Protocol Buffers.

Memorizzazione nella cache

Le cache consentono di ridurre la latenza e il sovraccarico di comunicazioni delle architetture basate su microservizi. In base al caso d'uso e ai potenziali colli di bottiglia, è possibile implementare diversi livelli di cache. Molte applicazioni di microservizi eseguite in AWS adottano ElastiCache per ridurre il volume delle chiamate ad altri microservizi memorizzando i risultati in una cache locale. Anche API Gateway offre un livello di caching integrato che permette di ridurre il carico sui server di back-end. Inoltre, il caching è utile per ridurre il carico dal livello di persistenza. La sfida da affrontare per utilizzare qualsiasi meccanismo di memorizzazione in cache è individuare il giusto equilibrio tra percentuale di riscontri nella cache e tempestività/consistenza dei dati.

Audit

Un'altra problematica da affrontare per adottare le architetture di microservizi con centinaia di servizi distribuiti è garantire la visibilità sulle operazioni degli utenti in ciascun servizio e, allo stesso tempo, la visibilità generale di tutti i servizi a livello organizzativo. Per agevolare l'applicazione di policy di sicurezza, è importante eseguire verifiche sia dell'accesso alle risorse, sia delle attività che generano modifiche di sistema.

Le modifiche devono essere monitorate a livello di singolo servizio, nonché tra i servizi in esecuzione sul sistema più ampio. Di norma, le architetture di microservizi sono soggette frequentemente a modifiche, il che rende l’audit alle attività che le provocano ancora piu importante. Questa sezione prende in esame i servizi e le caratteristiche più importanti di AWS per agevolare l'audit di questo tipo di architetture.

Percorsi di audit

AWS CloudTrail è un utile strumento che permette di monitorare le modifiche nei microservizi abilitando la registrazione di log per tutte le chiamate API effettuate su Cloud AWS e inviando tali log a CloudWatch Logs in tempo reale o su Amazon S3 entro alcuni minuti.

Sarà così possibile eseguire ricerche tra le operazioni degli utenti e dei sistemi automatizzati per individuare comportamenti inattesi o violazioni delle policy aziendali e per eseguire il debug. Le informazioni memorizzate includono una marca temporale, informazioni su utente e account, il servizio richiamato, l'operazione del servizio richiesta, l'indirizzo IP dell'autore della chiamata, nonché i parametri della richiesta e gli elementi della risposta.

CloudTrail permette di definire diversi percorsi per uno stesso account. In tal modo tutte le parti interessate, tra cui amministratori della sicurezza, sviluppatori software e addetti all'auditing IT, possono realizzare e gestire i propri percorsi. Se il team di microservizi dispone di più account AWS, è possibile aggregare tali percorsi in un singolo un bucket S3.

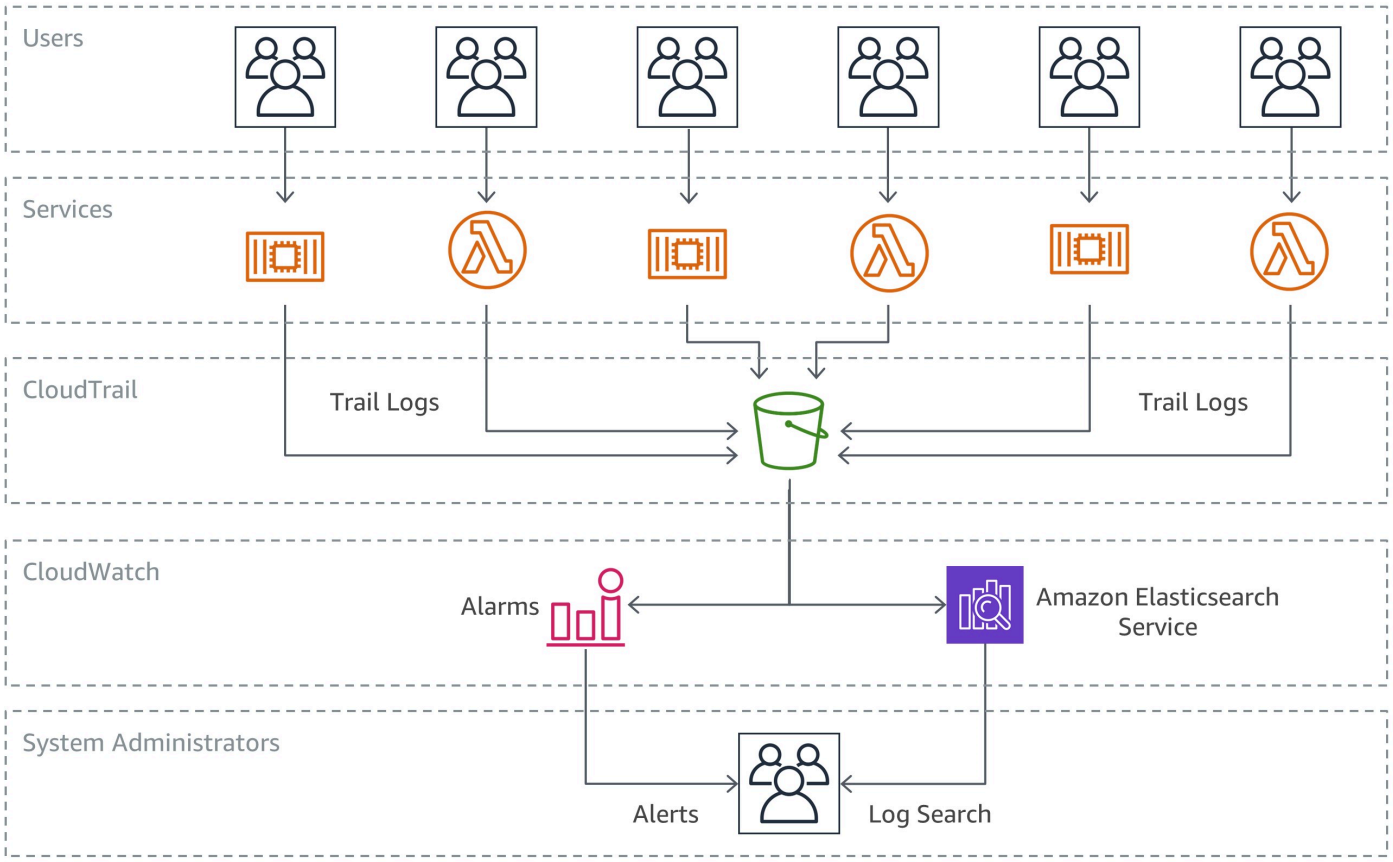
Tra i vantaggi della memorizzazione dei percorsi di audit su CloudWatch ci sono l'acquisizione dei dati in tempo reale e la facilità di reinstradamento delle informazioni verso Amazon OpenSearch Service per eseguire ricerche e visualizzazioni. È possibile configurare CloudTrail per salvare i log sia in Amazon S3, sia in CloudWatch Logs.

Eventi e operazioni in tempo reale

Ad alcune modifiche dell'architettura dei sistemi è vitale rispondere il più rapidamente possibile, mettendo anche in pratica soluzioni di remediation o specifiche procedure di governance di autorizzazione. Mediante la sua integrazione con CloudTrail, Amazon CloudWatch Events è in grado di generare eventi per tutte le chiamate API che apportano modifiche a qualunque servizio AWS. È anche possibile definire eventi personalizzati o generare eventi in base a una pianificazione fissa.

Quando un evento viene generato e corrisponde a una regola definita, un gruppo predefinito di persone nell'organizzazione può ricevere immediatamente una notifica, in modo da intraprendere l'operazione appropriata. In caso di possibile automazione dell'operazione richiesta, la regola può attivare automaticamente un flusso di lavoro integrato o richiamare una funzione Lambda per la risoluzione del problema.

La figura seguente illustra un ambiente in cui CloudTrail e CloudWatch Events operano in sinergia per soddisfare i requisiti di auditing e remediation nell'ambito di un'architettura di microservizi. Tutti i microservizi vengono monitorati da CloudTrail, e il percorso di audit viene memorizzato in un bucket Amazon S3. CloudWatch Events si accorge delle modifiche operative appena si verificano. CloudWatch Events risponde a questi cambiamenti operativi e adotta le necessarie misure correttive inviando messaggi per rispondere all'ambiente, attivando funzioni, operando modifiche e acquisendo informazioni sullo stato. CloudWatch Events opera al di sopra di CloudTrail e attiva avvisi quando vengono apportate determinate modifiche all'architettura.



Auditing e remediation

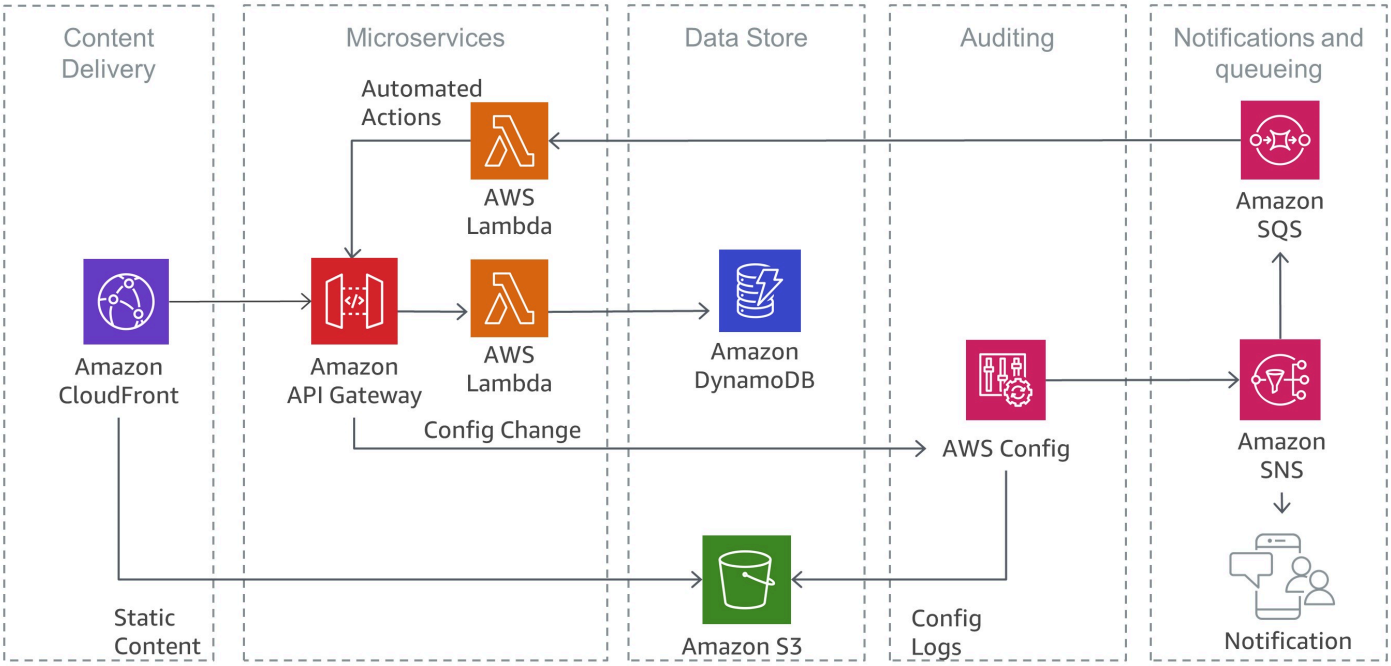
Inventario delle risorse e gestione delle modifiche

Per poter mantenere il controllo sulle mutevoli configurazioni dell'infrastruttura in un ambiente di sviluppo agile, è fondamentale adottare un approccio automatizzato e gestito a verifiche e controlli.

Mentre CloudTrail e CloudWatch Events sono elementi di base importanti per monitorare e rispondere alle modifiche dell'infrastruttura tra microservizi, le regole di AWS Config consentono a un'azienda di definire policy di sicurezza con regole specifiche che rilevano e monitorano le relative violazioni inviando avvisi.

L'esempio seguente chiarisce in che modo è possibile rilevare modifiche non conformi alla configurazione di un'architettura di microservizi, come acquisire informazioni a riguardo e come agire in modo automatico. Un membro del team di sviluppo ha apportato una modifica all’API Gateway per un microservizio, il che consente all'endpoint di accettare il traffico HTTP in entrata, anziché consentirne solo le richieste.

Poiché l'organizzazione ha già identificato in precedenza questa situazione come un problema di conformità della sicurezza, esiste già una regola di configurazione AWS che la monitora. La regola identifica la modifica come una violazione alla sicurezza ed esegue due operazioni: crea un registro della modifica rilevata in un bucket Amazon S3 per l'auditing e genera una notifica SNS. In questo scenario, Amazon SNS serve due scopi: invia un'e-mail a gruppi specifici per informarli della violazione alla sicurezza e aggiunge un messaggio alla coda SQS. Successivamente, il messaggio viene prelevato e la conformità viene ristabilita modificando la configurazione di API Gateway.



Rilevamento di violazioni di sicurezza con AWS Config

Conclusione

Le architetture di microservizi rappresentano un approccio di progettazione distribuito pensato per superare le limitazioni delle architetture monolitiche tradizionali. I microservizi permettono di ricalibrare le risorse delle applicazioni e le risorse aziendali velocizzando i cicli di rilascio. Tuttavia, presentano anche un paio di problematiche che, se non affrontate, possono addirittura accrescere la complessità e il carico operativo.

AWS offre un'ampia gamma di servizi gestiti che può aiutare gli sviluppatori a realizzare architetture di microservizi riducendo al minimo complessità operativa e architetturale. Il presente whitepaper illustra i servizi AWS più utili a riguardo e spiega come implementare soluzioni comuni, come rilevamento di servizi ed event sourcing, utilizzando i servizi AWS in modo nativo.

Risorse

- Centro di progettazione AWS
- Whitepaper AWS
- Mensile di architettura AWS
- Blog sull'architettura di AWS
- Video La mia architettura
- AWS Answers
- Documentazione AWS

Cronologia del documento e collaboratori

Cronologia dei documenti

Per ricevere una notifica sugli aggiornamenti di questo whitepaper, iscriviti al feed RSS.

update-history-change	update-history-description	update-history-date
Whitepaper aggiornato	Integrazione di Amazon EventBridge, AWS OpenTelemetry, AMP, AMG, Container Insights, modifiche minori al testo.	9 novembre 2021
Aggiornamenti minori	Layout di pagina modificato	30 aprile 2021
Aggiornamenti minori	Modifiche minori al testo.	1 agosto 2019
Whitepaper aggiornato	Integrazione di Amazon EKS, AWS Fargate, Amazon MQ, AWS PrivateLink, AWS App Mesh, AWS Cloud Map	1 giugno 2019
Whitepaper aggiornato	Integrazione con AWS Step Functions, AWS X-Ray e i flussi di eventi di ECS.	1 settembre 2017
Pubblicazione iniziale	Pubblicazione di Implement azione di microservizi in AWS.	1 dicembre 2016

> **Note**
>
> Per iscriversi e ricevere gli aggiornamenti RSS, è necessario disporre di un plug-in RSS abilitato per il browser in uso.

Collaboratori

Hanno contribuito alla stesura di questo documento:

- Sascha Möllering, Solutions Architecture, AWS
- Christian Müller, Solutions Architecture, AWS
- Matthias Jung, Solutions Architecture, AWS
- Peter Dalbhanjan, Solutions Architecture, AWS
- Peter Chapman, Solutions Architecture, AWS
- Christoph Kassen, Solutions Architecture, AWS
- Umair Ishaq, Solutions Architecture, AWS
- Rajiv Kumar, Solutions Architecture, AWS

Avvisi

I clienti sono responsabili della propria valutazione autonoma delle informazioni contenute in questo documento. Questo documento: (a) è solo a scopo informativo, (b) mostra le offerte e le pratiche attuali dei prodotti AWS soggette a modifiche senza preavviso, e (c) non crea alcun impegno o garanzia da parte di AWS e dei suoi affiliati, fornitori o licenziatari. I prodotti o servizi AWS sono forniti "così come sono" senza garanzie, dichiarazioni o condizioni di alcun tipo, sia esplicite che implicite. Le responsabilità e gli obblighi di AWS verso i propri clienti sono disciplinati dagli accordi AWS e il presente documento non fa parte né modifica alcun accordo tra AWS e i suoi clienti.