AWS Whitepaper

Implementazione di microservizi su AWS

Implementazione di microservizi su AWS: AWS Whitepaper

Table of Contents

Implementazione di microservizi su AWS

Data di pubblicazione: 31 luglio 2023 () Cronologia dei documenti

I microservizi offrono un approccio semplificato allo sviluppo del software che accelera l'implementazione, incoraggia l'innovazione, migliora la manutenibilità e aumenta la scalabilità. Questo metodo si basa su servizi di piccole dimensioni e scarsamente accoppiati che comunicano attraverso team ben definiti e gestiti da team autonomi. APIs L'adozione dei microservizi offre vantaggi, come una migliore scalabilità, resilienza, flessibilità e cicli di sviluppo più rapidi.

Questo white paper esplora tre modelli di microservizi popolari: basati su API, basati su eventi e streaming di dati. Forniamo una panoramica di ogni approccio, descriviamo le caratteristiche principali dei microservizi, affrontiamo le sfide del loro sviluppo e illustriamo come Amazon Web Services (AWS) può aiutare i team applicativi a superare questi ostacoli.

Considerando la natura complessa di argomenti come l'archiviazione dei dati, la comunicazione asincrona e l'individuazione dei servizi, ti consigliamo di valutare le esigenze e i casi d'uso specifici della tua applicazione oltre alle indicazioni fornite per prendere decisioni sull'architettura.

Introduzione

Le architetture di microservizi combinano concetti collaudati e di successo provenienti da vari campi, come:

- Sviluppo agile del software
- Architetture orientate ai servizi
- Progettazione incentrata sulle API
- Continuo) Integration/Continuous Delivery (CI/CD

Spesso, i microservizi incorporano modelli di progettazione dell'app Twelve-Factor.

Sebbene i microservizi offrano molti vantaggi, è fondamentale valutare i requisiti unici del caso d'uso e i costi associati. L'architettura monolitica o gli approcci alternativi possono essere più appropriati in alcuni casi. La decisione tra microservizi o monoliti dovrebbe essere effettuata su case-by-case base mirata, considerando fattori quali la scala, la complessità e i casi d'uso specifici.

Per prima cosa esploriamo un'architettura di microservizi altamente scalabile e tollerante ai guasti (interfaccia utente, implementazione di microservizi, archivio dati) e dimostriamo come costruirla

utilizzando tecnologie container. AWS Suggeriamo quindi AWS servizi per implementare una tipica architettura di microservizi serverless, che riduca la complessità operativa.

Serverless è caratterizzato dai seguenti principi:

- Nessuna infrastruttura da fornire o gestire
- Scalabilità automatica per unità di consumo
- Modello di fatturazione «Pay for value»
- Disponibilità e tolleranza ai guasti integrate
- Event Driven Architecture (EDA)

Infine, esaminiamo l'intero sistema e discutiamo gli aspetti trasversali di un'architettura di microservizi, come il monitoraggio distribuito, la registrazione, il tracciamento, il controllo, la coerenza dei dati e la comunicazione asincrona.

Questo documento si concentra sui carichi di lavoro in esecuzione, esclusi gli scenari ibridi e le strategie di migrazione. Cloud AWS Per informazioni sulle strategie di migrazione, consulta il white paper sulla metodologia di migrazione dei container.

Sei Well-Architected?

Il AWS Well-Architected Framework ti aiuta a comprendere i pro e i contro delle decisioni che prendi quando crei sistemi nel cloud. I sei pilastri del Framework consentono di apprendere le migliori pratiche architettoniche per progettare e gestire sistemi affidabili, sicuri, efficienti, convenienti e sostenibili. Utilizzando AWS Well-Architected Tool, disponibile gratuitamente in Console di gestione AWS, puoi esaminare i tuoi carichi di lavoro rispetto a queste best practice rispondendo a una serie di domande per ogni pilastro.

In Serverless Application Lens, ci concentriamo sulle migliori pratiche per l'architettura delle applicazioni serverless. AWS

Per ulteriori indicazioni e best practice da parte di esperti per la tua architettura cloud (implementazioni dell'architettura di riferimento, diagrammi e white paper), consulta l'Architecture Center.AWS

Modernizzazione ai microservizi

I microservizi sono essenzialmente piccole unità indipendenti che costituiscono un'applicazione. La transizione dalle strutture monolitiche tradizionali ai microservizi può seguire varie strategie.

Questa transizione influisce anche sul modo in cui opera l'organizzazione:

- Incoraggia lo sviluppo agile, in cui i team lavorano in cicli rapidi.
- I team sono in genere piccoli, a volte descritti come due squadre di pizzerie, abbastanza piccole da poter sfamare l'intero team con due pizze.
- I team si assumono la piena responsabilità dei propri servizi, dalla creazione all'implementazione e alla manutenzione.

Architettura di microservizi semplice su AWS

Le applicazioni monolitiche tipiche sono costituite da diversi livelli: un livello di presentazione, un livello di applicazione e un livello di dati. Le architetture di microservizi, d'altra parte, separano le funzionalità in verticali coesivi in base a domini specifici, anziché a livelli tecnologici. La Figura 1 illustra un'architettura di riferimento per una tipica applicazione di microservizi su. AWS

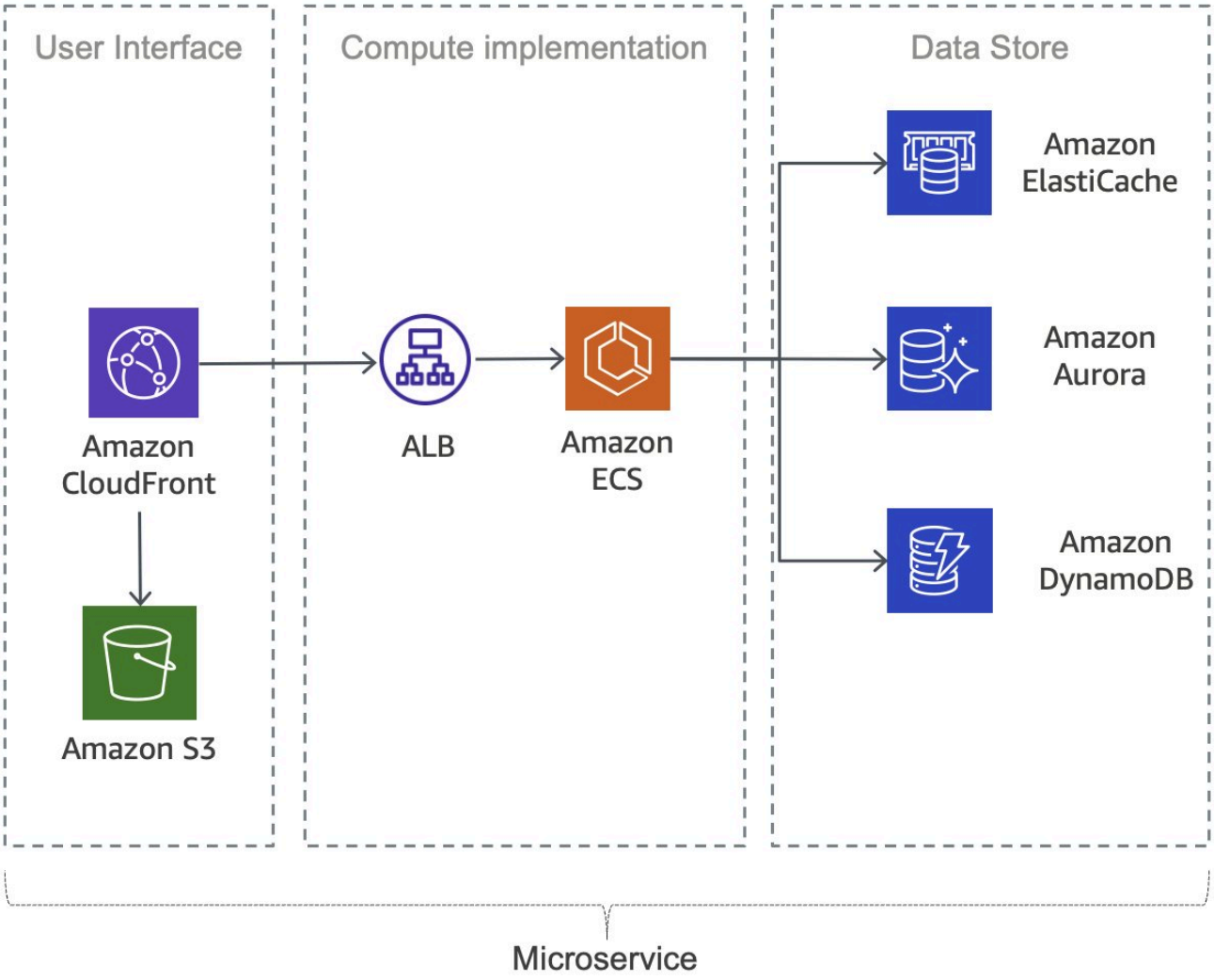

Figura 1: Applicazione di microservizi tipica su AWS

Interfaccia utente

Le applicazioni web moderne spesso utilizzano JavaScript framework per sviluppare applicazioni a pagina singola che comunicano con il backend. APIs Questi APIs sono in genere creati utilizzando

Representational State Transfer (REST) o RESTful APIs GraphQL APIs. I contenuti web statici possono essere serviti utilizzando Amazon Simple Storage Service (Amazon S3) e Amazon. CloudFront

Micro-servizi

APIs sono considerati la porta d'ingresso dei microservizi, in quanto rappresentano il punto di ingresso per la logica delle applicazioni. In genere, vengono RESTful utilizzati servizi Web API o GraphQL APIs . Questi APIs gestiscono ed elaborano le chiamate dei client, gestendo funzioni come la gestione del traffico, il filtraggio delle richieste, il routing, la memorizzazione nella cache, l'autenticazione e l'autorizzazione.

Implementazioni di microservizi

AWS offre elementi costitutivi per lo sviluppo di microservizi, tra cui Amazon ECS e Amazon EKS come scelte per i motori di orchestrazione dei container AWS Fargate ed EC2 come opzioni di hosting. AWS Lambda è un altro modo serverless su cui creare microservizi. AWS La scelta tra queste opzioni di hosting dipende dai requisiti del cliente per la gestione dell'infrastruttura sottostante.

AWS Lambda consente di caricare il codice, ridimensionandolo automaticamente e gestendone l'esecuzione con elevata disponibilità. Ciò elimina la necessità di gestire l'infrastruttura, così puoi muoverti rapidamente e concentrarti sulla logica di business. Lambda supporta più linguaggi di programmazione e può essere attivata da altri AWS servizi o richiamata direttamente da applicazioni Web o mobili.

Le applicazioni basate su container hanno guadagnato popolarità grazie alla portabilità, alla produttività e all'efficienza.AWS offre diversi servizi per creare, implementare e gestire container.

- App2Container, uno strumento da riga di comando per la migrazione e la modernizzazione delle applicazioni Web Java e.NET in formato contenitore. AWS A2C analizza e crea un inventario di applicazioni in esecuzione su macchine virtuali bare metal, istanze Amazon Elastic Compute Cloud (EC2) o nel cloud.

- Amazon Elastic Container Service (Amazon ECS) e Amazon Elastic Kubernetes Service (Amazon EKS) gestiscono l'infrastruttura dei container, semplificando l'avvio e la manutenzione di applicazioni containerizzate.
  - Amazon EKS è un servizio Kubernetes gestito per eseguire Kubernetes nel AWS cloud e nei data center locali (Amazon EKS Anywhere). Ciò estende i servizi cloud in ambienti locali per

bassa latenza, elaborazione locale dei dati, elevati costi di trasferimento dei dati o requisiti di residenza dei dati (consulta il white paper su "Running Hybrid Container Workloads With Amazon EKS Anywhere «). Puoi utilizzare tutti i plug-in e gli strumenti esistenti della community Kubernetes con EKS.

- Amazon Elastic Container Service (Amazon ECS) è un servizio di orchestrazione di container completamente gestito che semplifica la distribuzione, la gestione e la scalabilità delle applicazioni containerizzate. I clienti scelgono ECS per la semplicità e la profonda integrazione con i servizi. AWS

Per ulteriori informazioni, consulta il blog Amazon ECS vs Amazon EKS: dare un senso ai servizi di AWS container.

- AWS App Runnerè un servizio di applicazioni container completamente gestito che consente di creare, distribuire ed eseguire applicazioni Web e servizi API containerizzati senza precedenti esperienze in infrastrutture o container.
- AWS Fargate, un motore di elaborazione serverless, funziona sia con Amazon ECS che con Amazon EKS per gestire automaticamente le risorse di elaborazione per le applicazioni container.
- Amazon ECR è un registro di container completamente gestito che offre hosting ad alte prestazioni, in modo da poter distribuire in modo affidabile immagini e artefatti delle applicazioni ovunque.

Integrazione continua e distribuzione continua (CI/CD)

L'integrazione continua e la distribuzione continua (CI/CD) sono una parte cruciale di un' DevOpsiniziativa per rapide modifiche del software. AWS offre servizi da implementare CI/CD per i microservizi, ma una discussione dettagliata esula dallo scopo di questo documento. Per ulteriori informazioni, consulta il white paper Practicing Continuous Integration and Continuous Delivery on AWS.

Reti private

AWS PrivateLink è una tecnologia che migliora la sicurezza dei microservizi consentendo connessioni private tra il Virtual Private Cloud (VPC) e i servizi supportati. AWS Aiuta a isolare e proteggere il traffico di microservizi, assicurando che non attraversi mai la rete Internet pubblica. Ciò è particolarmente utile per rispettare normative come PCI o HIPAA.

Datastore

Il data store viene utilizzato per rendere persistenti i dati necessari ai microservizi. Gli archivi più diffusi per i dati delle sessioni sono le cache in memoria come Memcached o Redis. AWS offre entrambe le tecnologie come parte del ElastiCache servizio Amazon gestito.

L'inserimento di una cache tra i server delle applicazioni e un database è un meccanismo comune per ridurre il carico di lettura sul database, il che, a sua volta, può consentire l'utilizzo di risorse per supportare più scritture. Le cache possono anche migliorare la latenza.

I database relazionali sono ancora molto diffusi per archiviare dati strutturati e oggetti aziendali. AWS offre sei motori di database (Microsoft SQL Server, Oracle, MySQL, MariaDB, PostgreSQL e Amazon Aurora) come servizi gestiti tramite Amazon Relational Database Service (Amazon RDS).

I database relazionali, tuttavia, non sono progettati per una scalabilità infinita, il che può rendere difficile e dispendiosa in termini di tempo l'applicazione di tecniche per supportare un numero elevato di query.

I database NoSQL sono stati progettati per favorire la scalabilità, le prestazioni e la disponibilità rispetto alla coerenza dei database relazionali. Un elemento importante dei database NoSQL è che in genere non applicano uno schema rigoroso. I dati vengono distribuiti su partizioni che possono essere ridimensionate orizzontalmente e recuperati utilizzando le chiavi di partizione.

Poiché i singoli microservizi sono progettati per fare bene una cosa, in genere dispongono di un modello di dati semplificato che potrebbe essere adatto alla persistenza NoSQL. È importante comprendere che i database NoSQL hanno modelli di accesso diversi rispetto ai database relazionali. Ad esempio, non è possibile unire le tabelle. Se necessario, la logica deve essere implementata nell'applicazione. Puoi usare Amazon DynamoDB per creare una tabella di database in grado di archiviare e recuperare qualsiasi quantità di dati e soddisfare qualsiasi livello di traffico di richiesta. DynamoDB offre prestazioni a una cifra in millisecondi, tuttavia, ci sono alcuni casi d'uso che richiedono tempi di risposta in microsecondi. DynamoDB Accelerator (DAX) offre funzionalità di caching per l'accesso ai dati.

DynamoDB offre anche una funzionalità di scalabilità automatica per regolare dinamicamente la capacità di throughput in risposta al traffico effettivo. Tuttavia, ci sono casi in cui la pianificazione della capacità è difficile o impossibile a causa di grandi picchi di attività di breve durata nell'applicazione. Per tali situazioni, DynamoDB offre un'opzione on-demand, che offre prezzi semplici. pay-per-request DynamoDB on-demand è in grado di gestire migliaia di richieste al secondo istantaneamente senza pianificazione della capacità.

Per ulteriori informazioni, consulta How to Gestione distribuita dei dati Choose a Database.

Semplificazione delle operazioni

Per semplificare ulteriormente gli sforzi operativi necessari per eseguire, mantenere e monitorare i microservizi, possiamo utilizzare un'architettura completamente serverless.

Implementazione di applicazioni basate su Lambda

Puoi distribuire il codice Lambda caricando un archivio di file o creando e caricando `zip` un'immagine del contenitore tramite l'interfaccia utente della console utilizzando un URI di immagine Amazon ECR valido. Tuttavia, quando una funzione Lambda diventa complessa, vale a dire che ha livelli, dipendenze e autorizzazioni, il caricamento tramite l'interfaccia utente può diventare complicato per le modifiche al codice.

L'utilizzo di AWS CloudFormation and the AWS Serverless Application Model (AWS SAM) o Terraform semplifica il AWS Cloud Development Kit (AWS CDK) processo di definizione delle applicazioni serverless. AWS SAM, supportato nativamente da CloudFormation, offre una sintassi semplificata per specificare le risorse serverless.AWS Lambda I livelli aiutano a gestire le librerie condivise su più funzioni Lambda, riducendo al minimo l'ingombro delle funzioni, centralizzando le librerie compatibili con i tenant e migliorando l'esperienza degli sviluppatori. Lambda SnapStart per Java migliora le prestazioni di avvio per le applicazioni sensibili alla latenza.

Per distribuire, specifica le risorse e le politiche di autorizzazione in un CloudFormation modello, impacchetta gli elementi di distribuzione e distribuisci il modello. SAM Local, uno AWS CLI strumento, consente lo sviluppo, il test e l'analisi locali di applicazioni serverless prima del caricamento su Lambda.

L'integrazione con strumenti come AWS Cloud9 IDE e AWS CodePipeline semplifica la creazione AWS CodeBuild AWS CodeDeploy, il test, il debug e la distribuzione di applicazioni basate su SAM.

Il diagramma seguente mostra la distribuzione delle risorse utilizzando strumenti CI/CD. AWS Serverless Application Model CloudFormation AWS

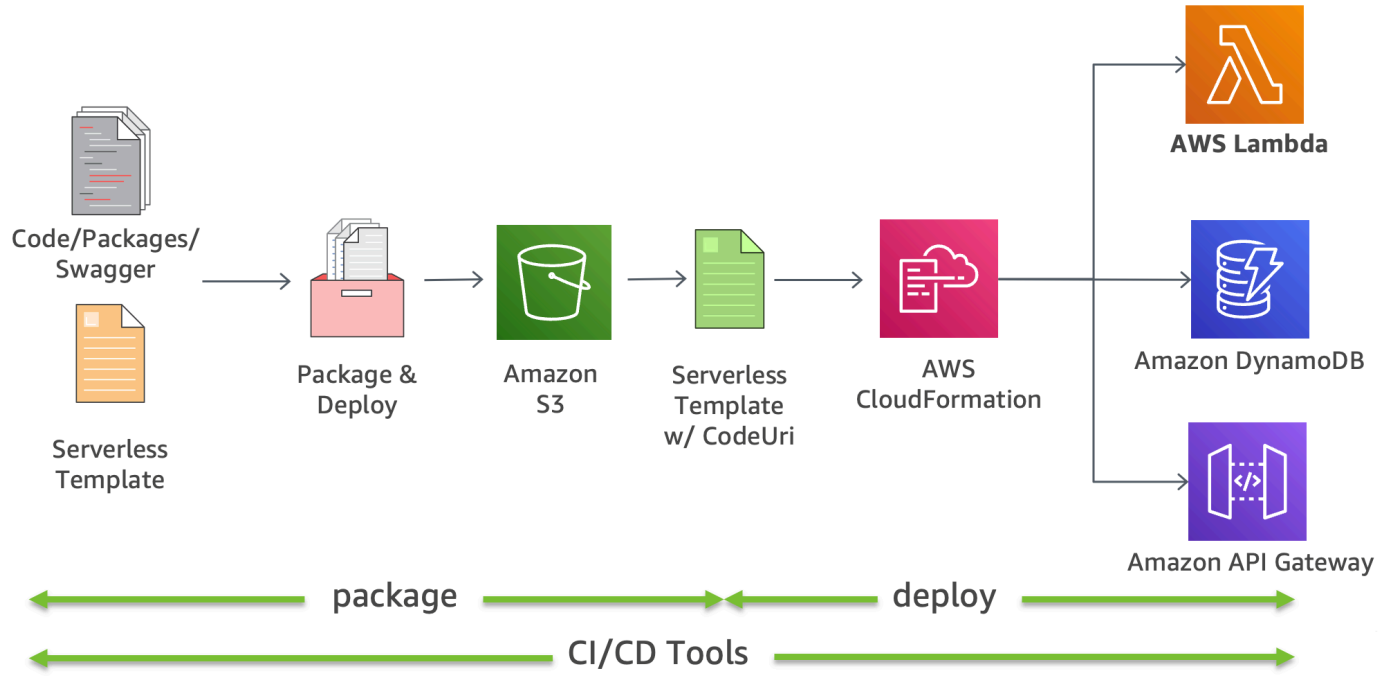

Figura 2: AWS Serverless Application Model ()AWS SAM

Astrazione delle complessità legate alla multi-tenancy

In un ambiente multi-tenant come le piattaforme SaaS, è fondamentale semplificare le complessità legate alla multi-tenancy, permettendo agli sviluppatori di concentrarsi sullo sviluppo di caratteristiche e funzionalità. Ciò può essere ottenuto utilizzando strumenti come AWS Lambda Layers, che offrono librerie condivise per affrontare problemi trasversali. La logica alla base di questo approccio è che le librerie e gli strumenti condivisi, se usati correttamente, gestiscono in modo efficiente il contesto dei tenant.

Tuttavia, non dovrebbero estendersi all'incapsulamento della logica aziendale a causa della complessità e dei rischi che possono comportare. Un problema fondamentale delle librerie condivise è la maggiore complessità degli aggiornamenti, che li rende più difficili da gestire rispetto alla duplicazione del codice standard. Pertanto, è essenziale trovare un equilibrio tra l'uso di librerie condivise e la duplicazione nella ricerca dell'astrazione più efficace.

Gestione API

La gestione APIs può richiedere molto tempo, soprattutto se si considerano più versioni, fasi del ciclo di sviluppo, autorizzazioni e altre funzionalità come la limitazione e la memorizzazione nella cache. Oltre ad API Gateway, alcuni clienti utilizzano anche ALB (Application Load Balancer) o NLB (Network Load Balancer) per la gestione delle API. Amazon API Gateway aiuta a ridurre la complessità operativa di creazione e manutenzione RESTful APIs. Ti consente di creare in modo APIs programmatico, funge da «porta d'ingresso» per accedere ai dati, alla logica di business o alle funzionalità dei tuoi servizi di backend, autorizzazione e controllo degli accessi, limitazione della velocità, memorizzazione nella cache, monitoraggio e gestione del traffico e funziona senza la gestione dei server. APIs

La Figura 3 illustra come API Gateway gestisce le chiamate API e interagisce con altri componenti. Le richieste provenienti da dispositivi mobili, siti Web o altri servizi di backend vengono indirizzate al CloudFront Point of Presence (PoP) più vicino per ridurre la latenza e fornire un'esperienza utente ottimale.

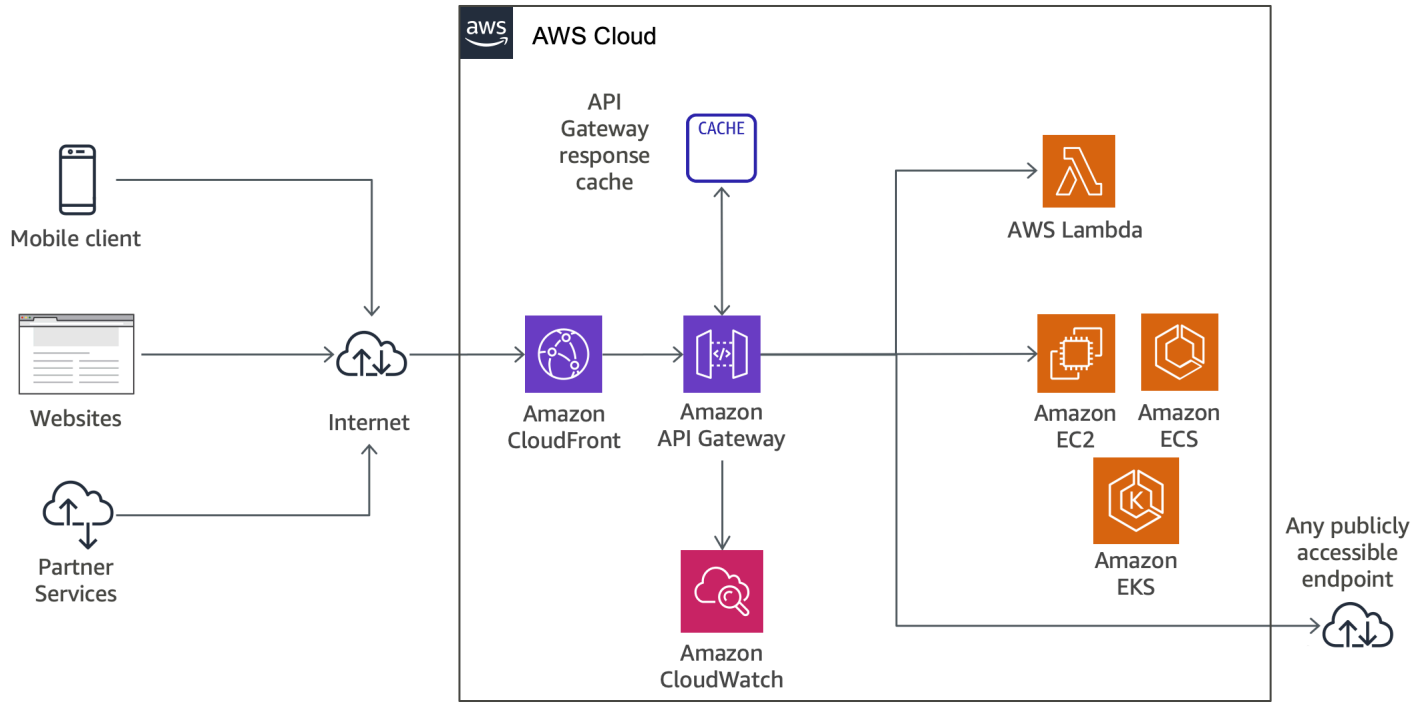

Figura 3: flusso di chiamate API Gateway

Microservizi su tecnologie serverless

L'utilizzo di microservizi con tecnologie serverless può ridurre notevolmente la complessità operativa. AWS Lambda e AWS Fargate, integrato con API Gateway, consente la creazione di applicazioni completamente serverless. A partire dal 7 aprile 2023, le funzioni Lambda possono trasmettere progressivamente i payload di risposta al client, migliorando le prestazioni per le applicazioni web e mobili. In precedenza, le applicazioni basate su Lambda che utilizzavano il tradizionale modello di invocazione richiesta-risposta dovevano generare e bufferizzare la risposta prima di restituirla al client, il che poteva ritardare il passaggio al primo byte. Con lo streaming delle risposte, le funzioni possono inviare risposte parziali al client non appena sono pronte, riducendo notevolmente il tempo necessario per arrivare al primo byte, aspetto a cui le applicazioni web e mobili sono particolarmente sensibili.

La Figura 4 illustra un'architettura di microservizi serverless che utilizza AWS Lambda e gestisce servizi. Questa architettura serverless mitiga la necessità di progettare in modo scalabile e ad alta disponibilità e riduce lo sforzo necessario per l'esecuzione e il monitoraggio dell'infrastruttura sottostante.

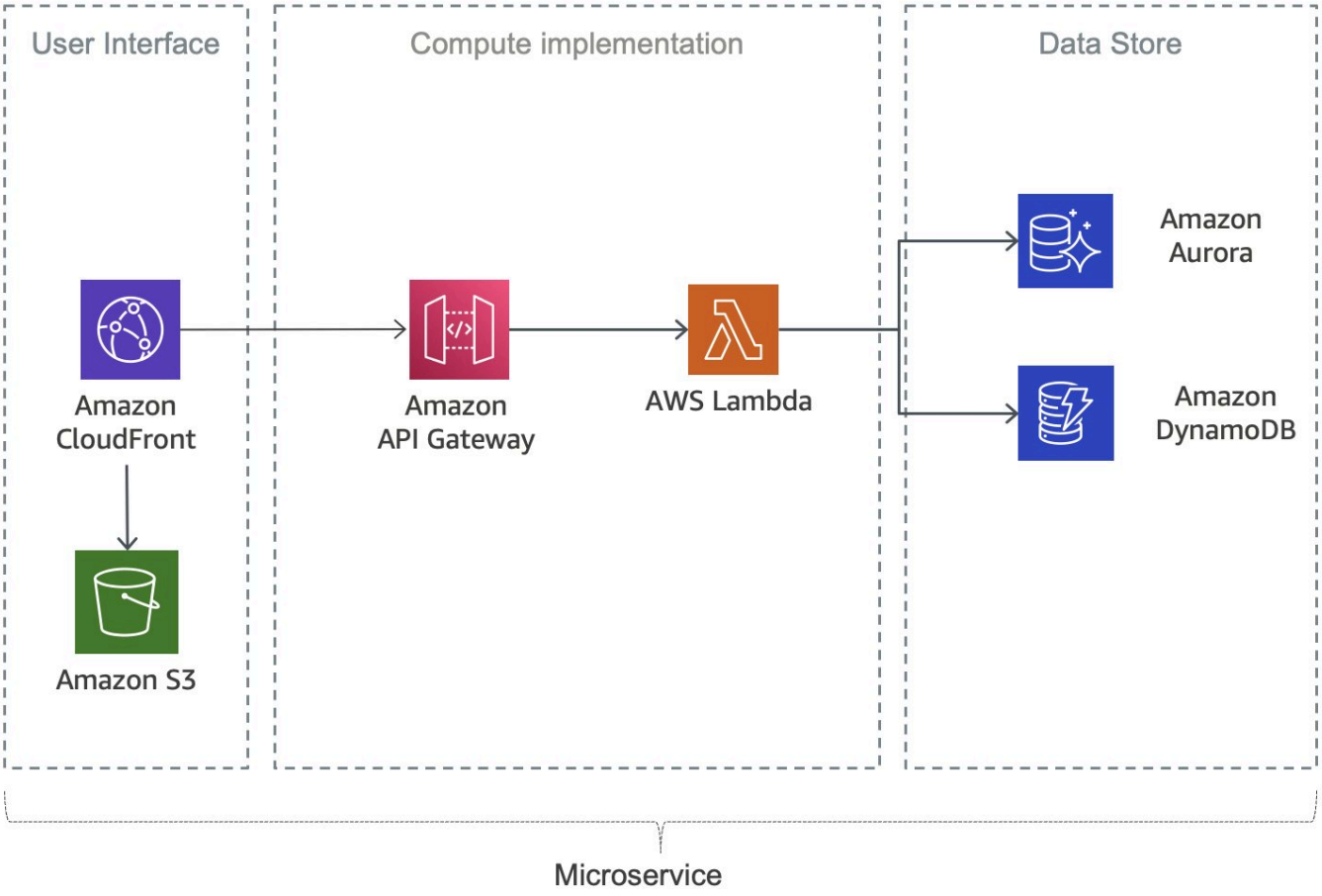

Figura 4: Utilizzo di un microservizio serverless AWS Lambda

La Figura 5 mostra un'implementazione serverless simile che utilizza contenitori con AWS Fargate, eliminando i problemi relativi all'infrastruttura sottostante. Include anche Amazon Aurora Serverless, un database on-demand con scalabilità automatica che regola automaticamente la capacità in base ai requisiti dell'applicazione.

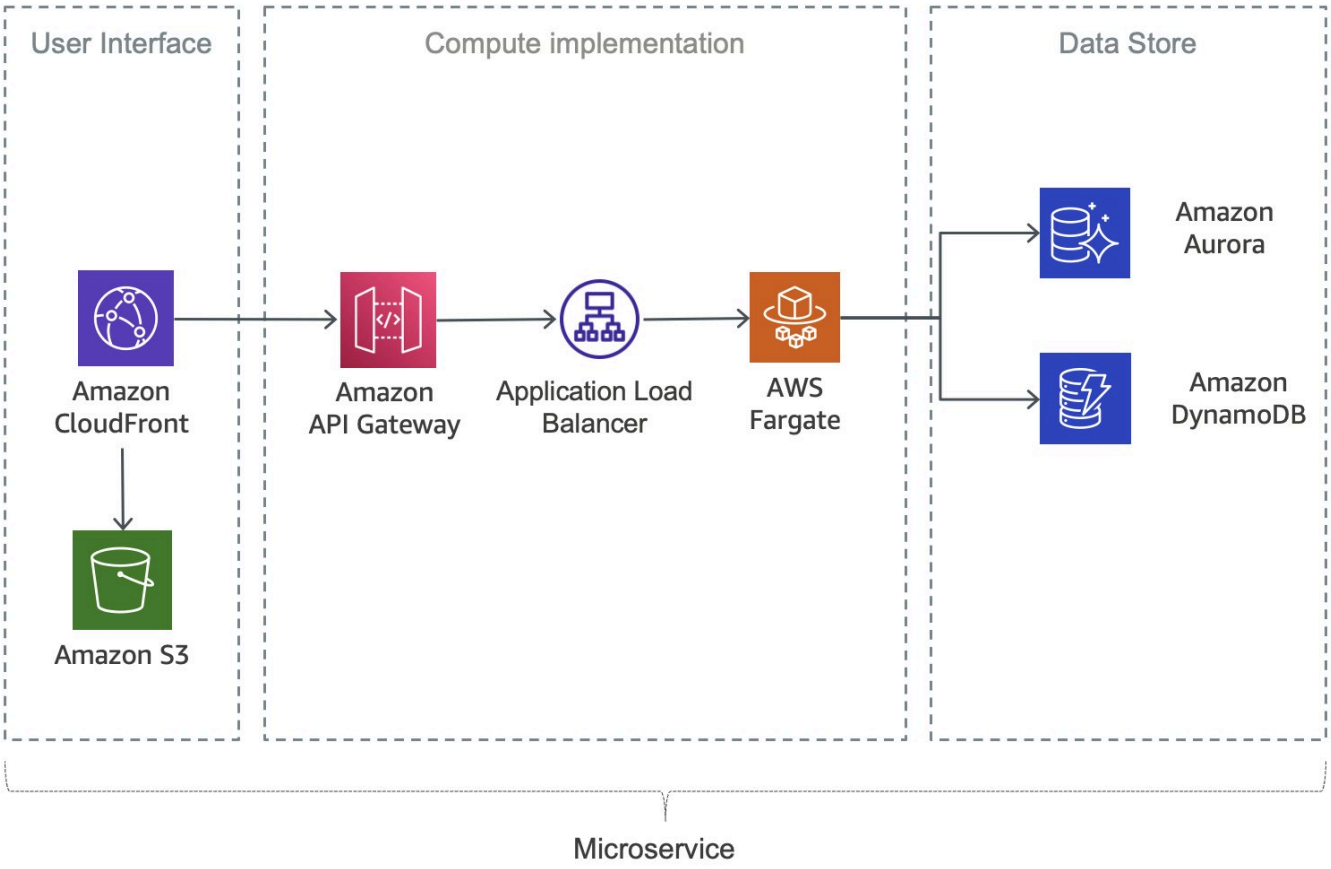

Figura 5: utilizzo di microservizi serverless AWS Fargate

Sistemi resilienti ed efficienti

Disaster recovery (DR)

Le applicazioni di microservizi seguono spesso i modelli delle applicazioni a dodici fattori, in cui i processi sono stateless e i dati persistenti vengono archiviati in servizi di supporto allo stato come i database. Ciò semplifica il disaster recovery (DR) perché in caso di guasto di un servizio, è facile avviare nuove istanze per ripristinare la funzionalità.

Le strategie di disaster recovery per i microservizi dovrebbero concentrarsi sui servizi a valle che mantengono lo stato dell'applicazione, come file system, database o code. Le organizzazioni devono pianificare il Recovery Time Objective (RTO) e il Recovery Point Objective (RPO). RTO è il ritardo massimo accettabile tra l'interruzione del servizio e il ripristino, mentre RPO è il tempo massimo trascorso dall'ultimo punto di ripristino dei dati.

Per ulteriori informazioni sulle strategie di disaster recovery, consulta il white paper Disaster Recovery of Workloads on AWS: Recovery in the Cloud.

Alta disponibilità (HA)

Esamineremo l'alta disponibilità (HA) per vari componenti di un'architettura di microservizi.

Amazon EKS offre un'elevata disponibilità eseguendo istanze di controllo e piano dati Kubernetes su più zone di disponibilità. Rileva e sostituisce automaticamente le istanze del piano di controllo non funzionanti e fornisce aggiornamenti e patch automatici delle versioni.

Amazon ECR utilizza Amazon Simple Storage Service (Amazon S3) per lo storage per rendere le immagini dei container altamente disponibili e accessibili. Funziona con Amazon EKS, Amazon ECS e semplifica il AWS Lambda flusso di lavoro dallo sviluppo alla produzione.

Amazon ECS è un servizio regionale che semplifica l'esecuzione dei container in modo altamente disponibile in più zone di disponibilità all'interno di una regione, offrendo diverse strategie di pianificazione che collocano i contenitori in base alle esigenze di risorse e ai requisiti di disponibilità.

AWS Lambda opera in più zone di disponibilità, garantendo la disponibilità durante le interruzioni del servizio in una singola zona. Se colleghi la tua funzione a un VPC, specifica le sottoreti in più zone di disponibilità per un'elevata disponibilità.

Componenti di sistemi distribuiti

In un'architettura di microservizi, la scoperta dei servizi si riferisce al processo di localizzazione e identificazione dinamica delle posizioni di rete (indirizzi IP e porte) dei singoli microservizi all'interno di un sistema distribuito.

Quando scegli un approccio su AWS, considera fattori come:

- Modifica del codice: è possibile ottenere i vantaggi senza modificare il codice?
- Traffico tra VPC o più account: se necessario, il tuo sistema necessita di una gestione efficiente delle comunicazioni tra diversi o? VPCs Account AWS
- Strategie di implementazione: il sistema utilizza o prevede di utilizzare strategie di implementazione avanzate come le implementazioni blue-green o canary?
- Considerazioni sulle prestazioni: se la vostra architettura comunica spesso con servizi esterni, quale sarà l'impatto sulle prestazioni complessive?

AWS offre diversi metodi per implementare il rilevamento dei servizi nell'architettura dei microservizi:

- Amazon ECS Service Discovery: Amazon ECS supporta il rilevamento dei servizi utilizzando il suo metodo basato su DNS o tramite l'integrazione con AWS Cloud Map (vedi ECS Service discovery). ECS Service Connect migliora ulteriormente la gestione delle connessioni, il che può essere particolarmente utile per applicazioni di grandi dimensioni con più servizi interagenti.
- Amazon Route 53: Route 53 si integra con ECS e altri AWS servizi, come EKS, per facilitare l'individuazione dei servizi. In un contesto ECS, Route 53 può utilizzare la funzionalità ECS Service Discovery, che sfrutta l'API Auto Naming per registrare e annullare automaticamente la registrazione dei servizi.
- AWS Cloud Map: Questa opzione offre un rilevamento dinamico dei servizi basato su API, che propaga le modifiche tra i servizi.

Per esigenze di comunicazione più avanzate, Amazon VPC Lattice è un servizio di rete di applicazioni che connette, monitora e protegge costantemente le comunicazioni tra i tuoi servizi, contribuendo a migliorare la produttività in modo che gli sviluppatori possano concentrarsi sulla creazione di funzionalità importanti per la tua azienda. Puoi definire politiche per la gestione, l'accesso e il monitoraggio del traffico di rete per connettere i servizi di elaborazione in modo semplificato e coerente tra istanze, contenitori e applicazioni serverless.

Se utilizzi già software di terze parti, come HashiCorp Consul o Netflix Eureka per la scoperta dei servizi, potresti preferire continuare a utilizzarli durante la migrazione, per una transizione più fluida. AWS

La scelta tra queste opzioni dovrebbe essere in linea con le tue esigenze specifiche. Per requisiti più semplici, potrebbero essere sufficienti soluzioni basate su DNS come Amazon ECS o AWS Cloud Map . Per sistemi più complessi o più grandi, le mesh di servizio come Amazon VPC Lattice potrebbero essere più adatte.

In conclusione, progettare un'architettura di microservizi significa AWS selezionare gli strumenti giusti per soddisfare le tue esigenze specifiche. Tenendo presenti le considerazioni discusse, potete assicurarvi di prendere decisioni informate per ottimizzare l'individuazione dei servizi e la comunicazione tra servizi del sistema.

Gestione distribuita dei dati

Nelle applicazioni tradizionali, tutti i componenti spesso condividono un unico database. Al contrario, ogni componente di un'applicazione basata su microservizi mantiene i propri dati, promuovendo l'indipendenza e il decentramento. Questo approccio, noto come gestione distribuita dei dati, comporta nuove sfide.

Una di queste sfide deriva dal compromesso tra coerenza e prestazioni nei sistemi distribuiti. Spesso è più pratico accettare lievi ritardi negli aggiornamenti dei dati (eventuale coerenza) piuttosto che insistere su aggiornamenti istantanei (coerenza immediata).

A volte, le operazioni aziendali richiedono la collaborazione di più microservizi. Se una parte si guasta, potrebbe essere necessario annullare alcune attività completate. Il modello Saga aiuta a gestire questo problema coordinando una serie di azioni compensative.

Per aiutare i microservizi a rimanere sincronizzati, è possibile utilizzare un archivio dati centralizzato. Questo negozio, gestito con strumenti come AWS Lambda, e Amazon AWS Step Functions EventBridge, può aiutare a ripulire e deduplicare i dati.

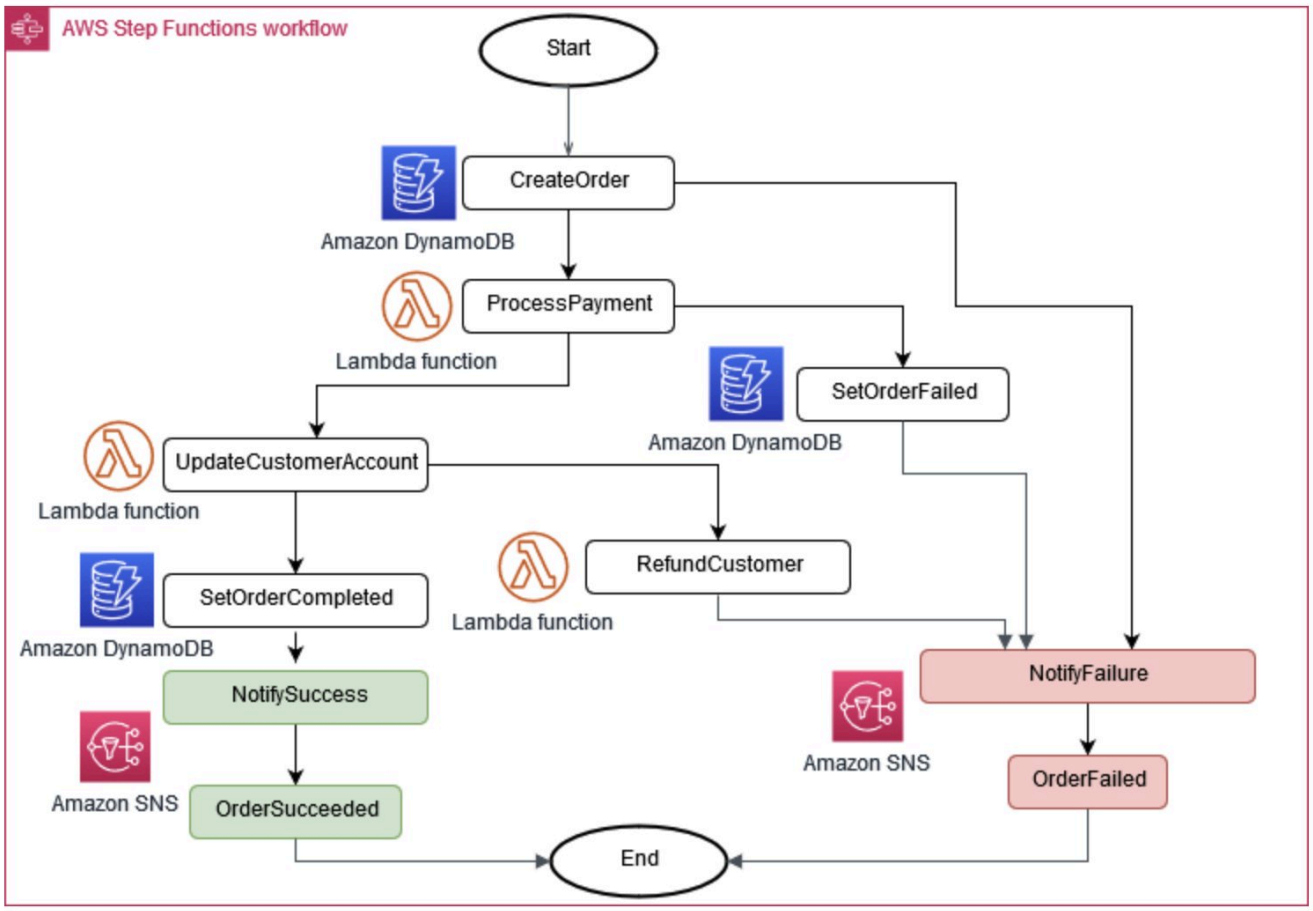

Figura 6: Coordinatore dell'esecuzione di Saga

Un approccio comune alla gestione delle modifiche tra i microservizi è l'event sourcing. Ogni modifica nell'applicazione viene registrata come evento, creando una sequenza temporale dello stato del sistema. Questo approccio non solo aiuta a eseguire il debug e l'audit, ma consente anche a diverse parti di un'applicazione di reagire agli stessi eventi.

L'event sourcing spesso funziona hand-in-hand con il pattern Command Query Responsibility Segregation (CQRS), che separa la modifica dei dati e l'interrogazione dei dati in diversi moduli per migliorare le prestazioni e la sicurezza.

È possibile implementare questi modelli utilizzando una combinazione di servizi. AWS Come illustrato nella Figura 7, Amazon Kinesis Data Streams può fungere da archivio centrale degli eventi, mentre Amazon S3 fornisce uno storage durevole per tutti i record di eventi. AWS Lambda, Amazon DynamoDB e Amazon API Gateway collaborano per gestire ed elaborare questi eventi.

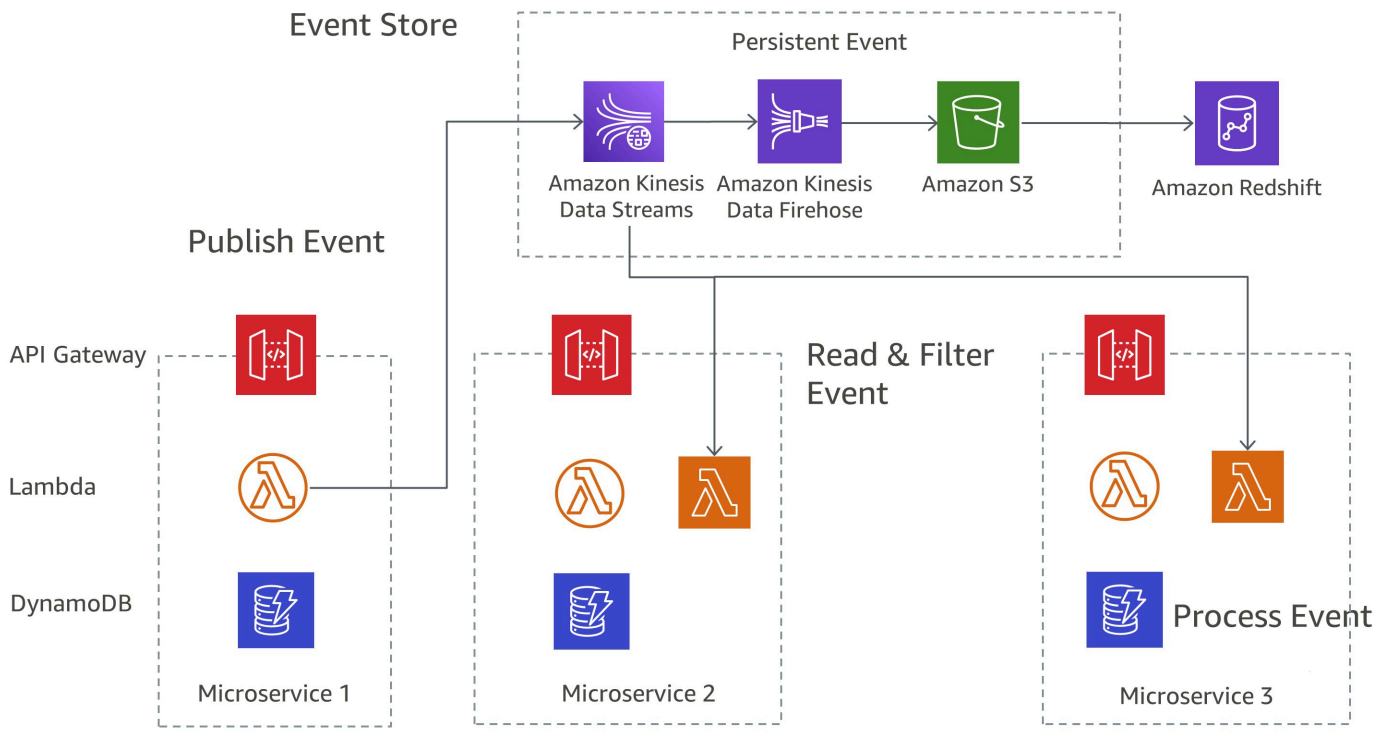

Figura 7: Schema di origine degli eventi attivo AWS

Ricorda che nei sistemi distribuiti, gli eventi potrebbero essere recapitati più volte a causa di nuovi tentativi, quindi è importante progettare le applicazioni in modo da gestire questo problema.

Gestione della configurazione

In un'architettura di microservizi, ogni servizio interagisce con varie risorse come database, code e altri servizi. Un modo coerente per configurare le connessioni e l'ambiente operativo di ciascun servizio è fondamentale. Idealmente, un'applicazione dovrebbe adattarsi alle nuove configurazioni senza dover riavviare. Questo approccio fa parte dei principi dell'app Twelve-Factor, che consigliano di memorizzare le configurazioni in variabili di ambiente.

Un approccio diverso consiste AWS nell'utilizzare App Config. È una funzionalità di AWS Systems Manager che consente ai clienti di configurare, convalidare e implementare in modo rapido e sicuro i flag delle funzionalità e la configurazione delle applicazioni. I dati relativi alle caratteristiche e alle configurazioni possono essere convalidati sintatticamente o semanticamente nella fase di pre-implementazione e possono essere monitorati e ripristinati automaticamente se viene attivato un allarme configurato. AppConfig può essere integrato con Amazon ECS e Amazon EKS utilizzando l' AWS AppConfig agente. L'agente funziona come un contenitore secondario che funziona insieme alle applicazioni container Amazon ECS e Amazon EKS. Se utilizzi i flag di AWS AppConfig funzionalità o altri dati di configurazione dinamici in una funzione Lambda, ti consigliamo di aggiungere l'estensione AWS AppConfig Lambda come livello alla tua funzione Lambda.

GitOpsè un approccio innovativo alla gestione della configurazione che utilizza Git come fonte di verità per tutte le modifiche alla configurazione. Ciò significa che qualsiasi modifica apportata ai file di configurazione viene automaticamente tracciata, versionata e verificata tramite Git.

Gestione dei segreti

La sicurezza è fondamentale, quindi le credenziali non devono essere trasmesse in testo semplice. AWS offre servizi sicuri per questo, come AWS Systems Manager Parameter Store e Gestione dei segreti AWS. Questi strumenti possono inviare segreti ai contenitori in Amazon EKS come volumi o ad Amazon ECS come variabili di ambiente. Nel AWS Lambda, le variabili di ambiente vengono rese disponibili automaticamente nel codice. Per i flussi di lavoro Kubernetes, External Secrets Operator recupera i segreti direttamente da servizi come Gestione dei segreti AWS la creazione di Kubernetes Secrets corrispondenti. Ciò consente una perfetta integrazione con le configurazioni native di Kubernetes.

Ottimizzazione dei costi e sostenibilità

L'architettura dei microservizi può migliorare l'ottimizzazione dei costi e la sostenibilità. Suddividendo un'applicazione in parti più piccole, è possibile scalare solo i servizi che richiedono più risorse, riducendo costi e sprechi. Ciò è particolarmente utile quando si ha a che fare con un traffico variabile. I microservizi sono sviluppati in modo indipendente. In questo modo gli sviluppatori possono eseguire aggiornamenti più piccoli e ridurre le risorse spese per i test end-to-end. Durante l'aggiornamento, dovranno testare solo un sottoinsieme delle funzionalità anziché i monoliti.

I componenti stateless (servizi che archiviano lo stato in un archivio dati esterno anziché in un archivio dati locale) nella tua architettura possono utilizzare le istanze Amazon EC2 Spot, che offrono EC2 capacità inutilizzata nel cloud. AWS Queste istanze sono più convenienti rispetto alle istanze on demand e sono perfette per carichi di lavoro in grado di gestire le interruzioni. Ciò può ridurre ulteriormente i costi mantenendo al contempo un'elevata disponibilità.

Con i servizi isolati, puoi utilizzare opzioni di elaborazione ottimizzate in termini di costi per ogni gruppo di auto-scaling. Ad esempio, AWS Graviton offre opzioni di elaborazione convenienti e ad alte prestazioni per carichi di lavoro adatti alle istanze basate su ARM.

L'ottimizzazione dei costi e dell'utilizzo delle risorse aiuta anche a ridurre al minimo l'impatto ambientale, in linea con il pilastro della sostenibilità del Well-Architected Framework. Puoi monitorare i tuoi progressi nella riduzione delle emissioni di carbonio utilizzando il Customer Carbon Footprint Tool. AWS Questo strumento fornisce informazioni sull'impatto ambientale del tuo AWS utilizzo.

Meccanismi di comunicazione

Nel paradigma dei microservizi, diversi componenti di un'applicazione devono comunicare tramite una rete. Gli approcci comuni per questo includono la messaggistica basata su REST, GraphQL, GRPC e asincrona.

Comunicazione basata su REST

Il HTTP/S protocollo, ampiamente utilizzato per la comunicazione sincrona tra microservizi, spesso funziona tramite. RESTful APIs API Gateway offre un modo semplificato per creare un'API che funga da punto di accesso centralizzato ai servizi di backend, gestendo attività come la gestione del traffico, l'autorizzazione, il monitoraggio e il controllo delle versioni.

Comunicazione basata su GraphQL

Allo stesso modo, GraphQL è un metodo molto diffuso per la comunicazione sincrona, che utilizza gli stessi protocolli di REST ma limita l'esposizione a un singolo endpoint. Con AWS AppSync, puoi creare e pubblicare applicazioni GraphQL che interagiscono direttamente con AWS servizi e datastore o incorporare funzioni Lambda per la logica aziendale.

Comunicazione basata su gRPC

gRPC è un protocollo di comunicazione RPC sincrono, leggero, ad alte prestazioni e open source. gRPC migliora i protocolli sottostanti utilizzando HTTP/2 e abilitando più funzionalità come la compressione e la prioritizzazione dei flussi. Utilizza il Protobuf Interface Definition Language (IDL) che è codificato in modo binario e quindi sfrutta il framing binario HTTP/2.

Messaggistica asincrona e trasmissione di eventi

La messaggistica asincrona consente ai servizi di comunicare inviando e ricevendo messaggi tramite una coda. Ciò consente ai servizi di rimanere strettamente collegati e di promuovere la scoperta dei servizi.

La messaggistica può essere definita nei seguenti tre tipi:

- Code di messaggi: una coda di messaggi funge da buffer che separa mittenti (produttori) e destinatari (consumatori) dei messaggi. I produttori inseriscono i messaggi nella coda e i

consumatori li rimuovono dalla coda e li elaborano. Questo modello è utile per la comunicazione asincrona, il livellamento del carico e la gestione di picchi di traffico.

- Publish-Subscribe: nel pattern publish-subscribe, un messaggio viene pubblicato su un argomento e più abbonati interessati ricevono il messaggio. Questo modello consente di trasmettere eventi o messaggi a più consumatori in modo asincrono.
- Messaggistica basata sugli eventi: la messaggistica basata sugli eventi implica l'acquisizione e la reazione agli eventi che si verificano nel sistema. Gli eventi vengono pubblicati su un broker di messaggi e i servizi interessati sottoscrivono tipi di eventi specifici. Questo modello consente l'accoppiamento libero e consente ai servizi di reagire agli eventi senza dipendenze dirette.

Per implementare ciascuno di questi tipi di messaggi, AWS offre diversi servizi gestiti come Amazon SQS, Amazon SNS, Amazon, EventBridge Amazon MQ e Amazon MSK. Questi servizi hanno caratteristiche uniche adattate a esigenze specifiche:

- Amazon Simple Queue Service (Amazon SQS) e Amazon Simple Notification Service (Amazon SNS): come illustrato nella Figura 8, questi due servizi si completano a vicenda, con Amazon SQS che fornisce uno spazio per l'archiviazione dei messaggi e Amazon SNS che consente la consegna dei messaggi a più abbonati. Sono efficaci quando lo stesso messaggio deve essere recapitato a più destinazioni.

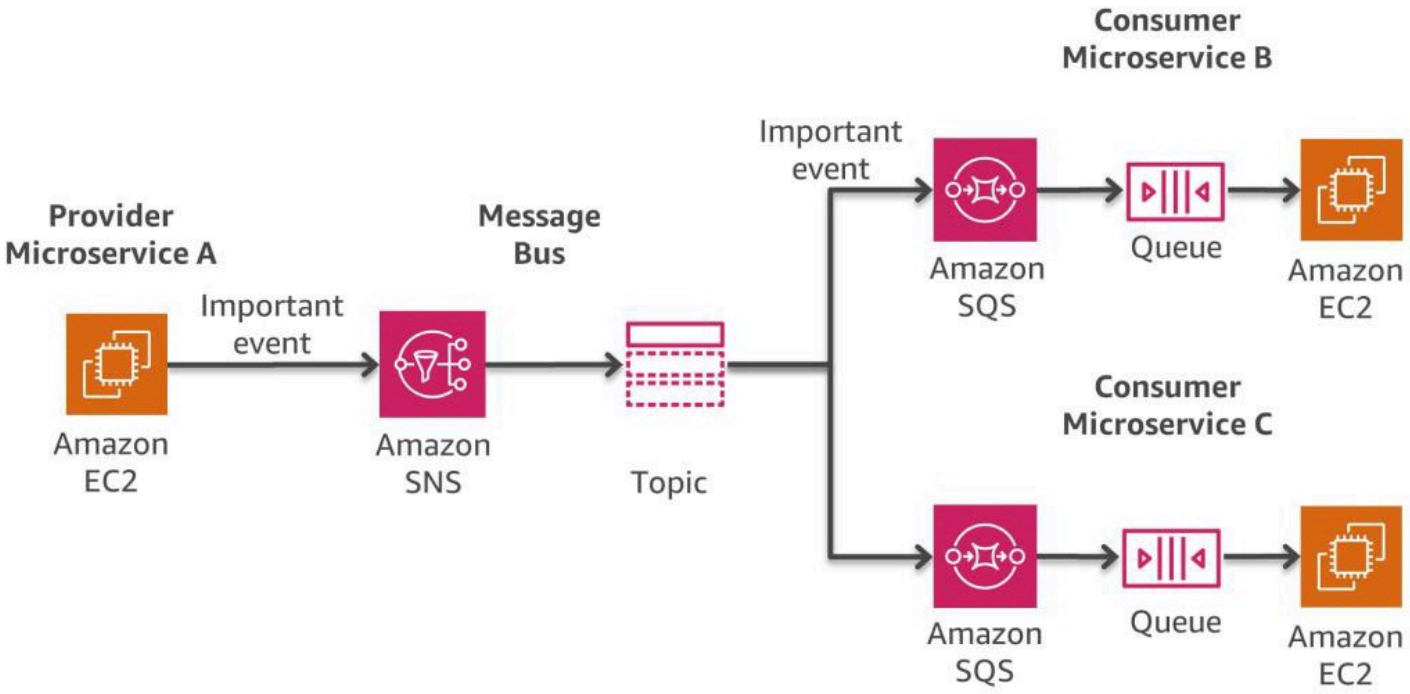

Figura 8: Schema del bus dei messaggi attivo AWS

- Amazon EventBridge: un servizio serverless che utilizza gli eventi per connettere tra loro i componenti delle applicazioni, semplificando la creazione di applicazioni scalabili basate sugli eventi. Usalo per indirizzare gli eventi da fonti come applicazioni, AWS servizi e software di terze parti sviluppati internamente alle applicazioni destinate ai consumatori in tutta l'organizzazione. EventBridge offre un modo semplice e coerente per importare, filtrare, trasformare e distribuire eventi in modo da poter creare nuove applicazioni rapidamente. EventBridge gli event bus sono ideali per il many-to-many routing degli eventi tra servizi basati sugli eventi.
- Amazon MQ: una buona scelta se disponi di un sistema di messaggistica preesistente che utilizza protocolli standard come JMS, AMQP o simili. Questo servizio gestito sostituisce il sistema senza interrompere le operazioni.
- Amazon MSK (Managed Kafka): un sistema di messaggistica per l'archiviazione e la lettura dei messaggi, utile nei casi in cui i messaggi devono essere elaborati più volte. Supporta anche lo streaming di messaggi in tempo reale.
- Amazon Kinesis: elaborazione e analisi in tempo reale di dati in streaming. Ciò consente lo sviluppo di applicazioni in tempo reale e offre una perfetta integrazione con l'ecosistema. AWS

Ricorda che il servizio migliore per te dipende dalle tue esigenze specifiche, quindi è importante capire cosa offre ognuno di essi e in che modo si allinea alle tue esigenze.

Orchestrazione e gestione dello stato

L'orchestrazione dei microservizi si riferisce a un approccio centralizzato, in cui un componente centrale, noto come orchestratore, è responsabile della gestione e del coordinamento delle interazioni tra i microservizi. L'orchestrazione dei flussi di lavoro su più microservizi può essere difficile. L'incorporazione del codice di orchestrazione direttamente nei servizi è sconsigliata, in quanto introduce un accoppiamento più stretto e ostacola la sostituzione dei singoli servizi.

Step Functions fornisce un motore di workflow per gestire le complessità di orchestrazione dei servizi, come la gestione degli errori e la serializzazione. Ciò consente di scalare e modificare rapidamente le applicazioni senza aggiungere codice di coordinamento. Step Functions fa parte della piattaforma AWS serverless e supporta funzioni Lambda, Amazon EC2, Amazon EKS, Amazon ECS SageMaker , AI e altro ancora. AWS Glue

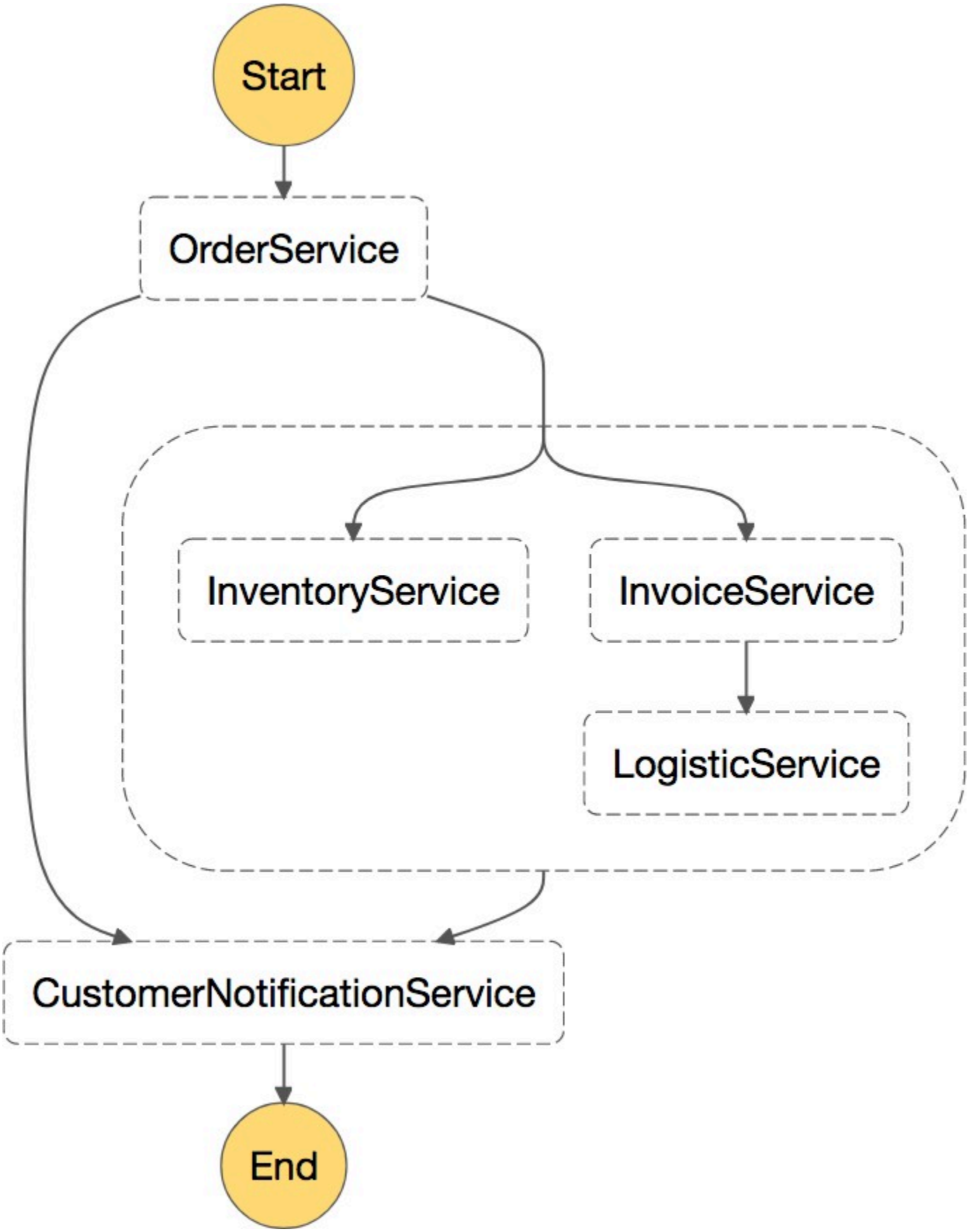
Start
OrderService
InventoryService
InvoiceService
LogisticService
CustomerNotificationService
End

Figura 9: Un esempio di flusso di lavoro di microservizi con passaggi paralleli e sequenziali richiamati da AWS Step Functions

Amazon Managed Workflows for Apache Airflow (Amazon MWAA) è un'alternativa a Step Functions. Se dai priorità all'open source e alla portabilità, dovresti usare Amazon MWAA. Airflow ha una community open source ampia e attiva che offre regolarmente nuove funzionalità e integrazioni.

Osservabilità

Poiché le architetture di microservizi sono intrinsecamente costituite da molti componenti distribuiti, l'osservabilità di tutti questi componenti diventa fondamentale. Amazon CloudWatch consente tutto ciò, raccogliendo e tracciando metriche, monitorando i file di registro e reagendo ai cambiamenti del tuo AWS ambiente. Può monitorare AWS le risorse e le metriche personalizzate generate dalle tue applicazioni e dai tuoi servizi.

Argomenti

Monitoraggio

CloudWatch offre una visibilità a livello di sistema sull'utilizzo delle risorse, sulle prestazioni delle applicazioni e sullo stato operativo. In un'architettura di microservizi, il monitoraggio delle metriche personalizzate CloudWatch è vantaggioso, in quanto gli sviluppatori possono scegliere quali metriche raccogliere. Il ridimensionamento dinamico può anche basarsi su queste metriche personalizzate.

CloudWatch Container Insights estende questa funzionalità, raccogliendo automaticamente le metriche per molte risorse come CPU, memoria, disco e rete. Aiuta a diagnosticare i problemi relativi ai container, semplificando la risoluzione.

Per Amazon EKS, una scelta spesso preferita è Prometheus, una piattaforma open source che offre funzionalità complete di monitoraggio e avviso. In genere è abbinato a Grafana per una visualizzazione intuitiva delle metriche. Amazon Managed Service for Prometheus (AMP) offre un servizio di monitoraggio completamente compatibile con Prometheus, che consente di supervisionare facilmente le applicazioni containerizzate. Inoltre, Amazon Managed Grafana (AMG) semplifica l'analisi e la visualizzazione delle metriche, eliminando la necessità di gestire l'infrastruttura sottostante.

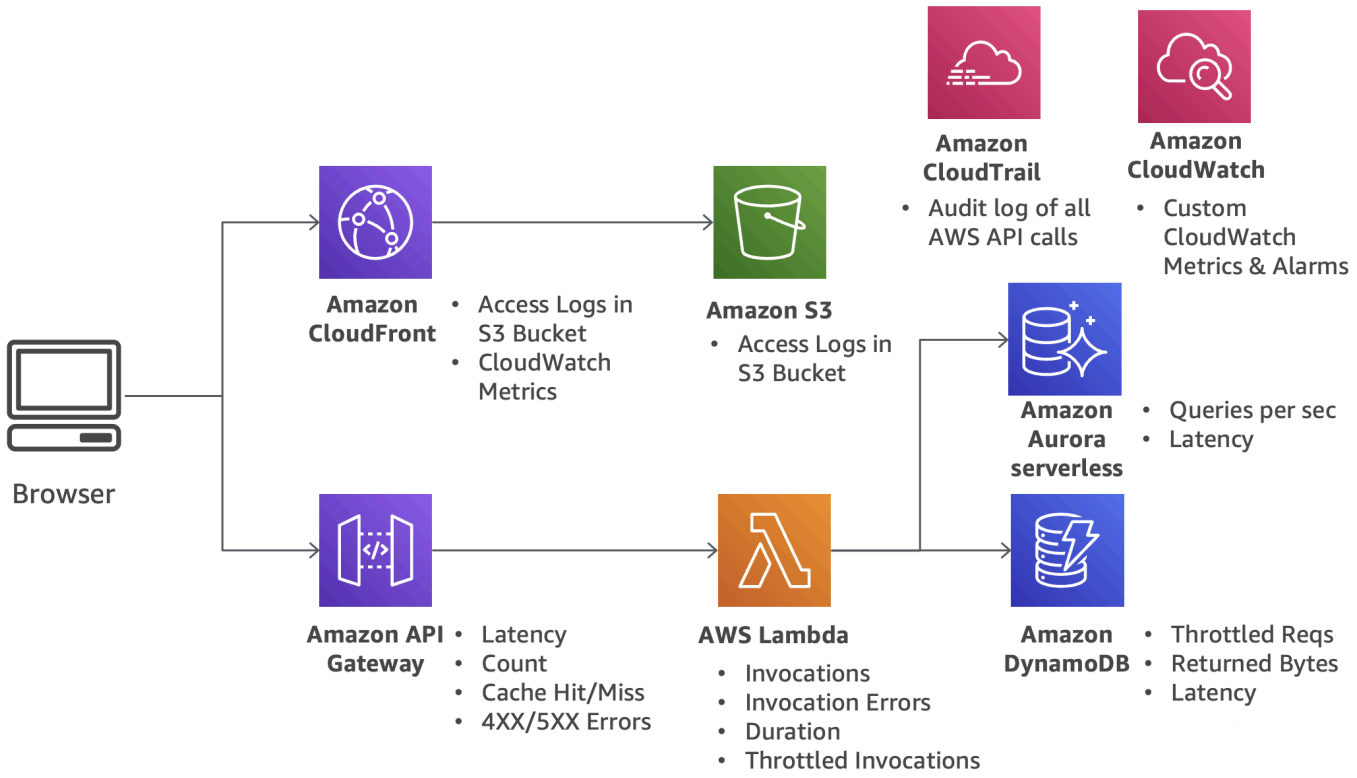

Figura 10: Un'architettura serverless con componenti di monitoraggio

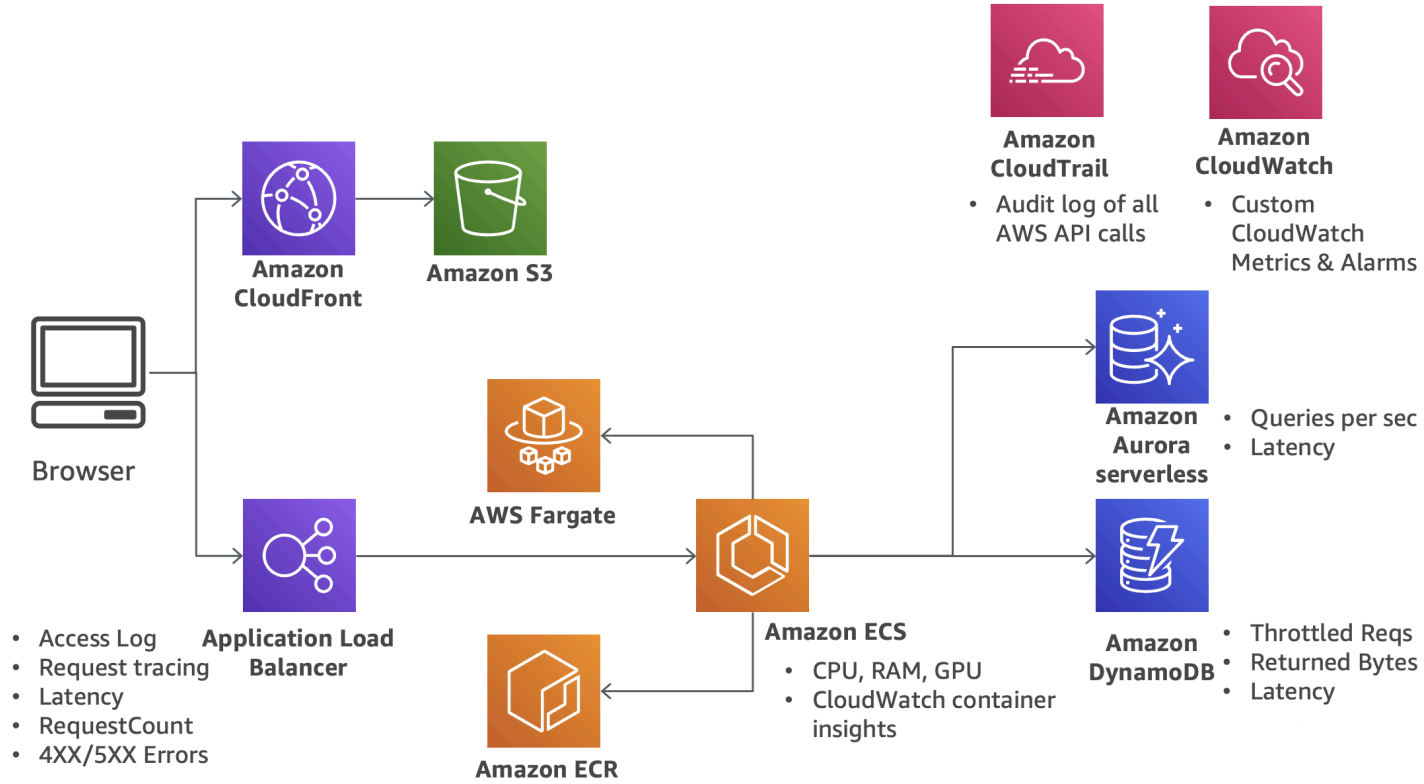

Figura 11: Un'architettura basata su contenitori con componenti di monitoraggio

Centralizzazione dei log

La registrazione è fondamentale per individuare e risolvere i problemi. Con i microservizi, puoi rilasciare più frequentemente e sperimentare nuove funzionalità. AWS fornisce servizi come Amazon S3, CloudWatch Logs e Amazon OpenSearch Service per centralizzare i file di registro. Amazon EC2 utilizza un daemon per inviare i log a, mentre CloudWatch Lambda e Amazon ECS inviano nativamente i loro output di log. Per Amazon EKS, è possibile utilizzare Fluent Bit o Fluentd per inoltrare i log a Kibana CloudWatch per la rendicontazione. OpenSearch Tuttavia, a causa del minore ingombro e dei vantaggi in termini di prestazioni, Fluent Bit è consigliato rispetto a Fluentd.

La Figura 12 illustra come i log di vari AWS servizi vengono indirizzati ad Amazon S3 e. CloudWatch Questi log centralizzati possono essere ulteriormente analizzati utilizzando Amazon OpenSearch Service, incluso Kibana per la visualizzazione dei dati. Inoltre, Amazon Athena può essere utilizzato per query ad hoc sui log archiviati in Amazon S3.

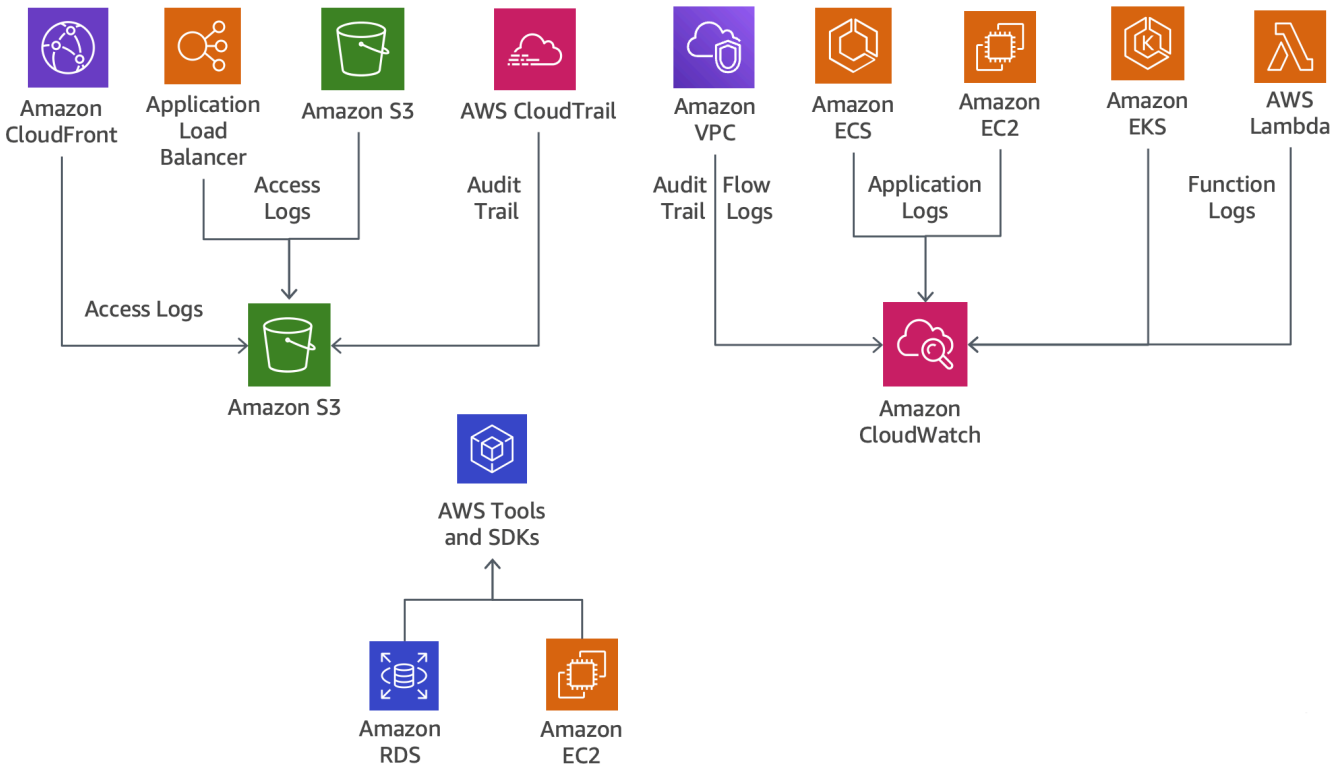

Figura 12: Funzionalità di registrazione dei servizi AWS

Tracciamento distribuito

I microservizi spesso collaborano per gestire le richieste. AWS X-Ray utilizza gli ID di correlazione per tenere traccia delle richieste tra questi servizi. X-Ray funziona con Amazon EC2, Amazon ECS, Lambda ed Elastic Beanstalk.

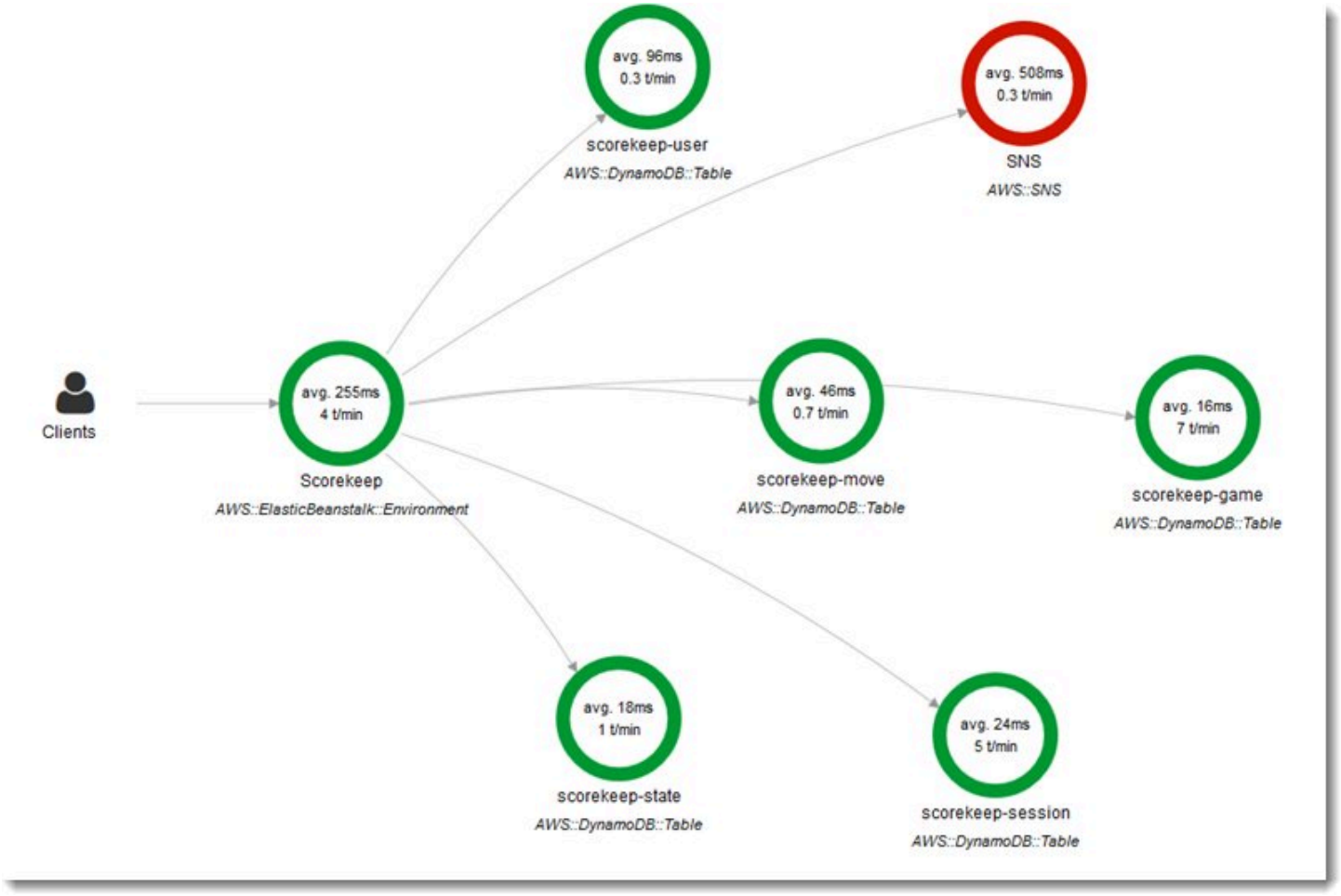

Figura 13: mappa dei AWS X-Ray servizi

AWS Distro for OpenTelemetry fa parte del OpenTelemetry progetto e fornisce strumenti open source e agenti per raccogliere tracce APIs e metriche distribuite, migliorando il monitoraggio delle applicazioni. Invia metriche e tracce a soluzioni di monitoraggio multiple AWS e partner. Raccogliendo i metadati dalle AWS risorse, allinea le prestazioni delle applicazioni ai dati dell'infrastruttura sottostante, accelerando la risoluzione dei problemi. Inoltre, è compatibile con una varietà di AWS servizi e può essere utilizzato in locale.

Analisi del registro su AWS

Amazon CloudWatch Logs Insights consente l'esplorazione, l'analisi e la visualizzazione dei log in tempo reale. Per un'ulteriore analisi dei file di registro, Amazon OpenSearch Service, che include Kibana, è uno strumento potente. CloudWatch I log possono trasmettere le voci di registro al OpenSearch Servizio in tempo reale. Kibana, perfettamente integrato con OpenSearch, visualizza questi dati e offre un'interfaccia di ricerca intuitiva.

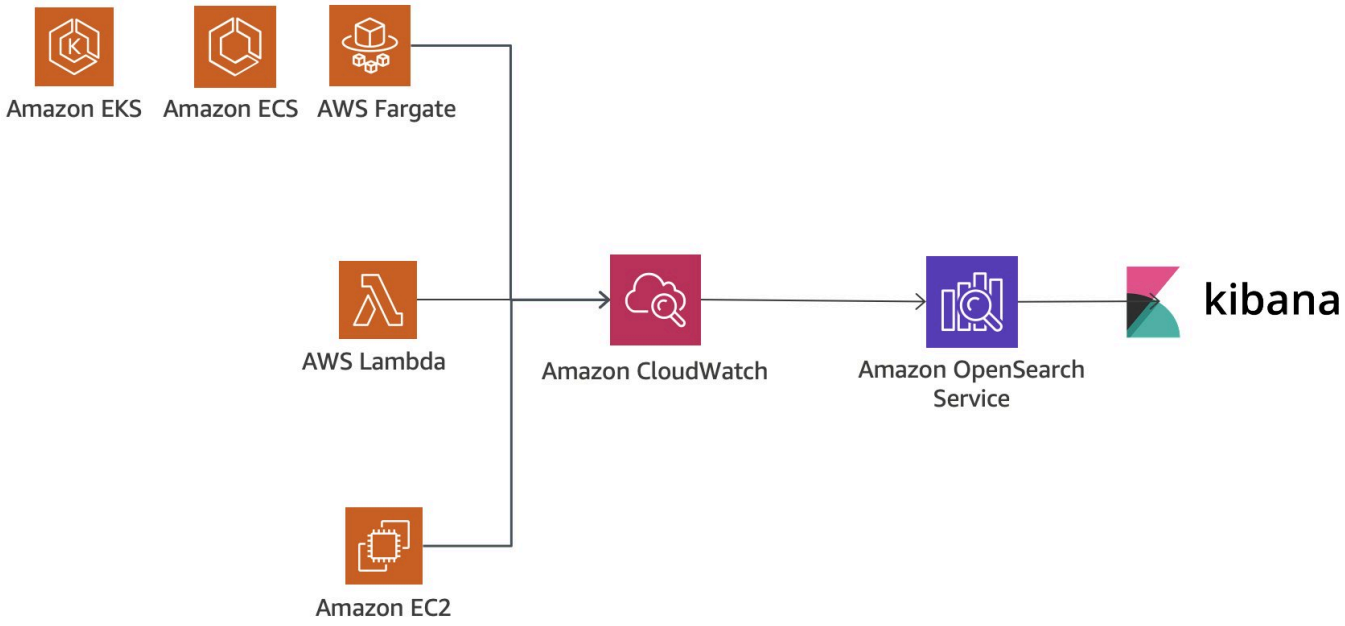

Figura 14: Analisi dei log con Amazon OpenSearch Service

Altre opzioni di analisi

Per un'ulteriore analisi dei log, Amazon Redshift, un servizio di data warehouse completamente gestito, e Quick, un servizio di business intelligence scalabile, offrono soluzioni efficaci. QuickSight fornisce una facile connettività a vari servizi di AWS dati come Redshift, RDS, Aurora, EMR, DynamoDB, Amazon S3 e Kinesis, semplificando l'accesso ai dati.

CloudWatch I log possono trasmettere le voci di log ad Amazon Data Firehose, un servizio per la distribuzione di dati di streaming in tempo reale. QuickSight utilizza quindi i dati archiviati in Redshift per analisi, reportistica e visualizzazione complete.

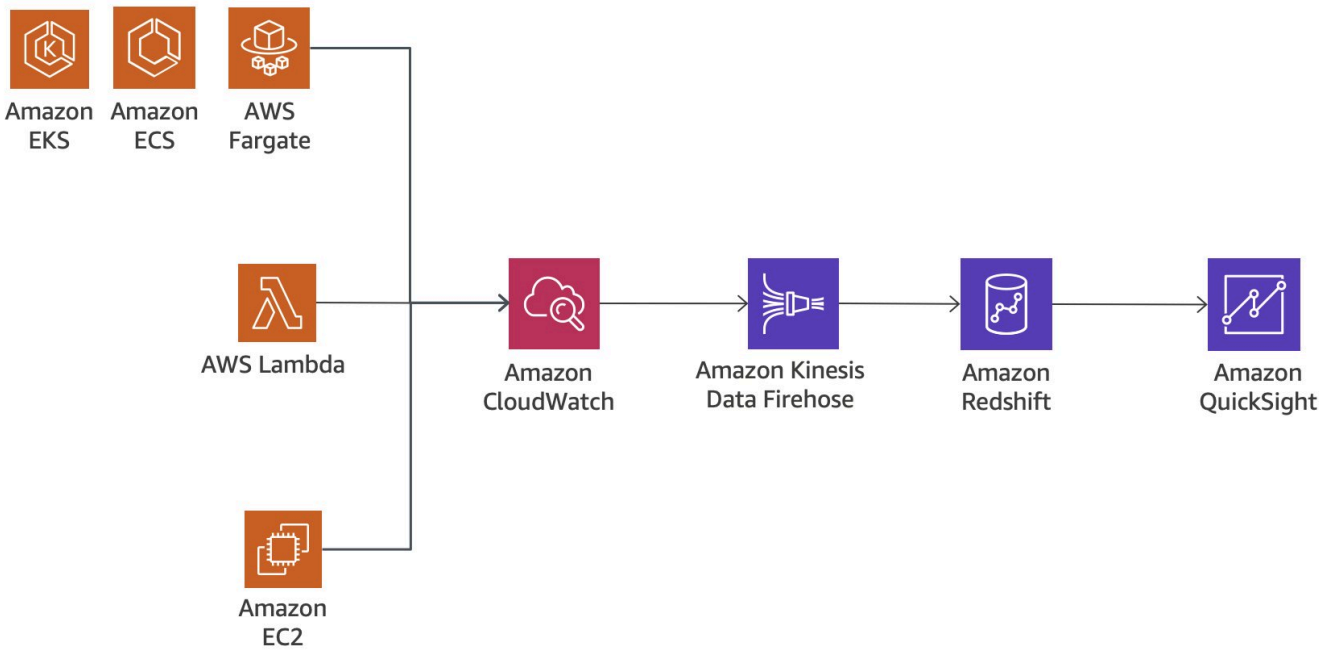

Figura 15: analisi dei log con Amazon Redshift e Quick

Inoltre, quando i log vengono archiviati in bucket S3, un servizio di object storage, i dati possono essere caricati in servizi come Redshift o EMR, una piattaforma di big data basata sul cloud, che consente un'analisi approfondita dei dati di log memorizzati.

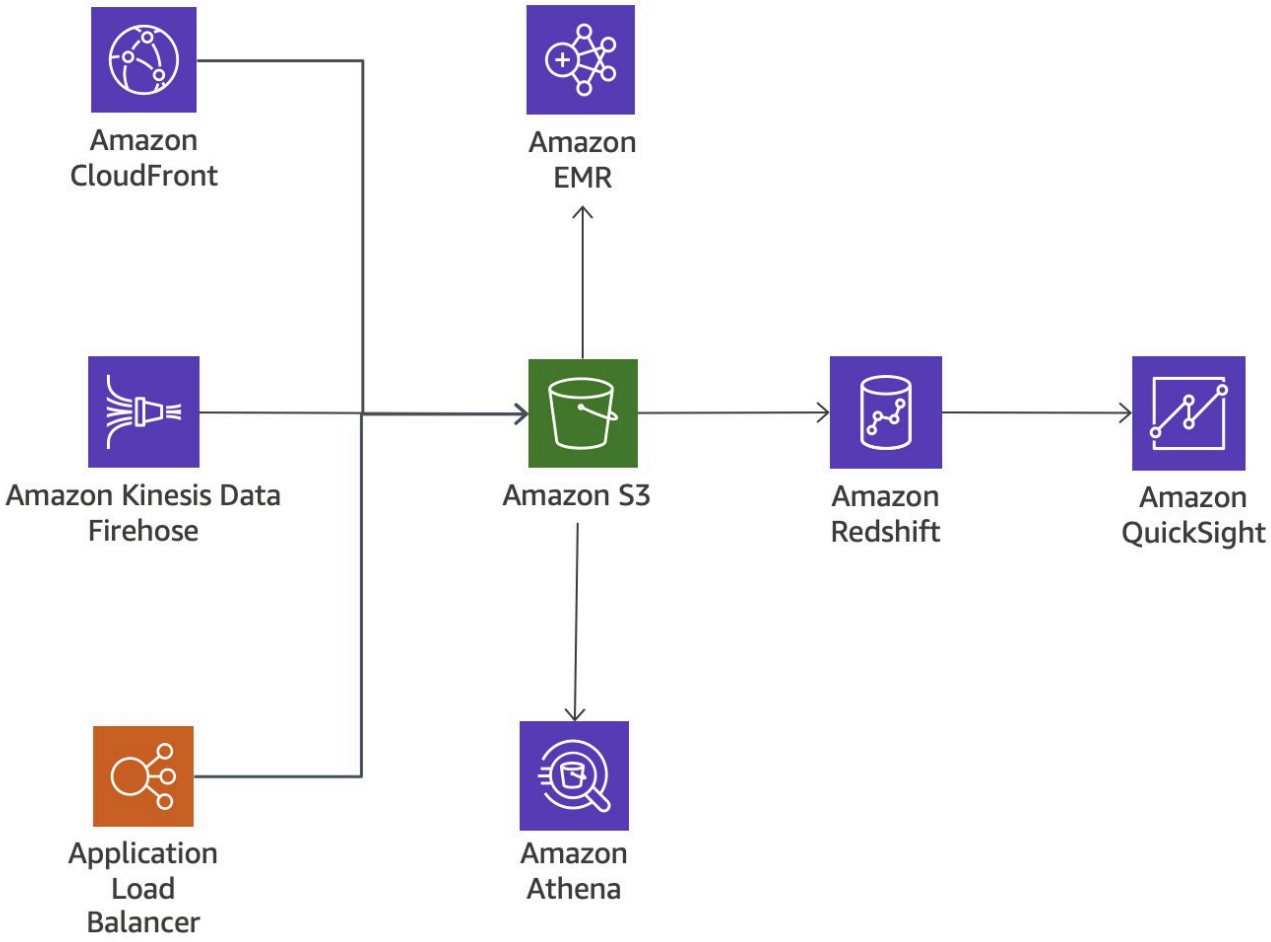

Figura 16: Semplificazione dell'analisi dei log: dai servizi a AWS QuickSight

Gestione della chiacchierata nella comunicazione tramite microservizi

La chiacchierata si riferisce all'eccessiva comunicazione tra microservizi, che può causare inefficienza a causa dell'aumento della latenza di rete. È essenziale gestire la chiacchierata in modo efficace per un sistema ben funzionante.

Alcuni strumenti chiave per la gestione delle chat sono REST APIs, HTTP APIs e gRPC. APIs REST APIs offre una gamma di funzionalità avanzate come chiavi API, limitazione per client, convalida delle richieste, integrazione o endpoint API privati. AWS WAF APIs Gli HTTP sono progettati con funzionalità minime e quindi hanno un prezzo inferiore. Per maggiori dettagli su questo argomento e un elenco delle funzionalità principali disponibili in REST APIs e HTTP APIs, vedi Scelta tra REST APIs e HTTP APIs.

Spesso, i microservizi utilizzano REST su HTTP per la comunicazione a causa del suo uso diffuso. Tuttavia, in situazioni ad alto volume, il sovraccarico di REST può causare problemi di prestazioni. È perché la comunicazione utilizza l'handshake TCP, necessario per ogni nuova richiesta. In questi casi, l'API gRPC è la scelta migliore. gRPC riduce la latenza in quanto consente più richieste su una singola connessione TCP. gRPC supporta anche lo streaming bidirezionale, che consente a client e server di inviare e ricevere messaggi contemporaneamente. Ciò porta a una comunicazione più efficiente, in particolare per trasferimenti di dati di grandi dimensioni o in tempo reale.

Se la chiacchierata persiste nonostante la scelta del tipo di API giusto, potrebbe essere necessario rivalutare l'architettura dei microservizi. Il consolidamento dei servizi o la revisione del modello di dominio potrebbero ridurre la chiacchierata e migliorare l'efficienza.

Utilizzo di protocolli e memorizzazione nella cache

I microservizi utilizzano spesso protocolli come gRPC e REST per la comunicazione (vedere la sezione precedente Meccanismi di comunicazione su). gRPC utilizza HTTP/2 per il trasporto, mentre REST utilizza tipicamente HTTP/1.1. gRPC utilizza buffer di protocollo per la serializzazione, mentre REST di solito utilizza JSON o XML. Per ridurre la latenza e il sovraccarico di comunicazione, è possibile applicare la memorizzazione nella cache. Servizi come Amazon ElastiCache o il livello di caching in API Gateway possono aiutare a ridurre il numero di chiamate tra i microservizi.

Audit

In un'architettura di microservizi, è fondamentale avere visibilità sulle azioni degli utenti su tutti i servizi. AWS fornisce strumenti come AWS CloudTrail, che registra tutte le chiamate API effettuate e AWS CloudWatch che viene utilizzato per acquisire i log delle applicazioni. AWS Ciò consente di tenere traccia delle modifiche e analizzare il comportamento nei microservizi. Amazon EventBridge può reagire rapidamente alle modifiche del sistema, avvisando le persone giuste o persino avviando automaticamente i flussi di lavoro per risolvere i problemi.

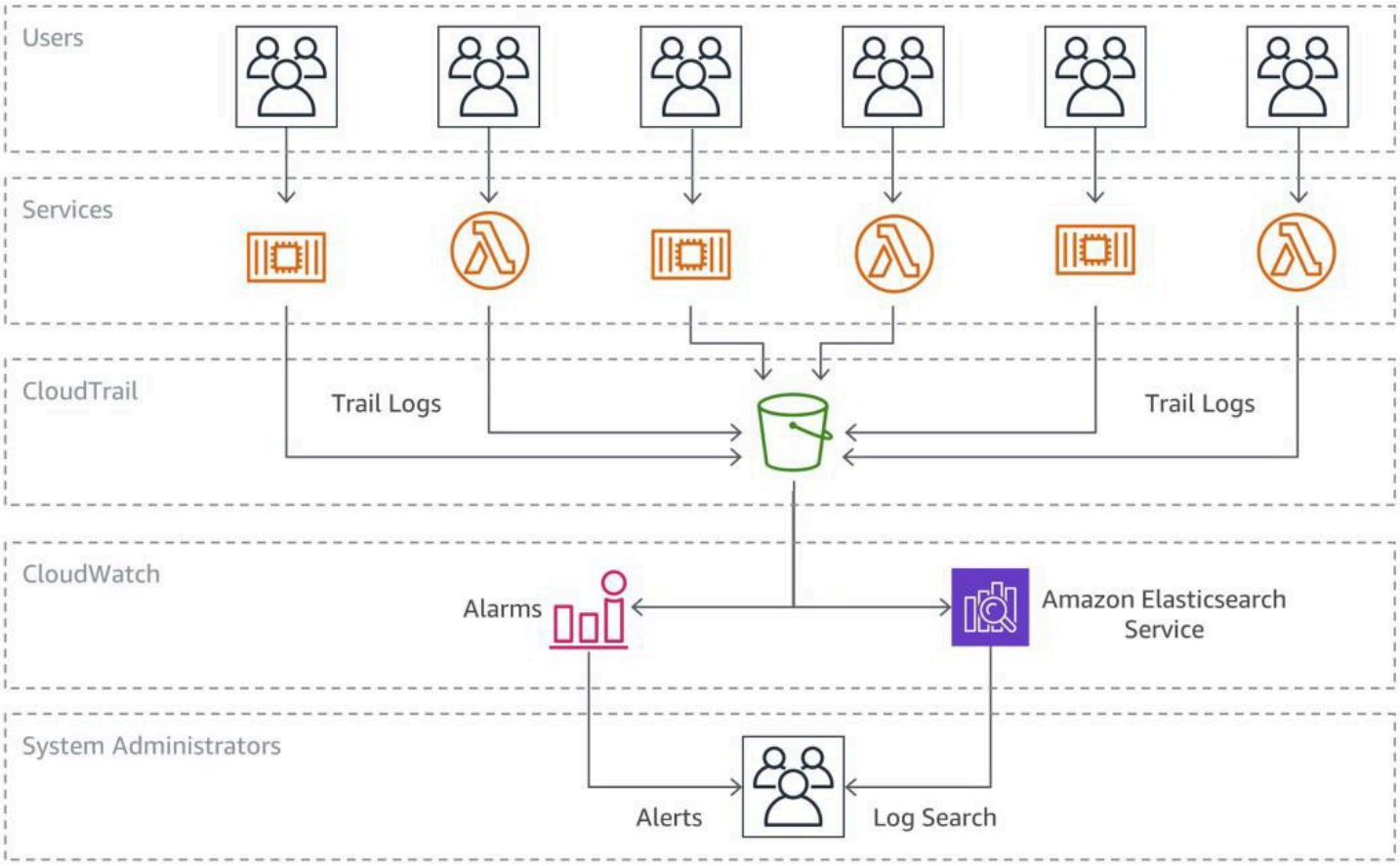

Figura 17: Controllo e correzione tra i microservizi

Inventario delle risorse e gestione delle modifiche

In un ambiente di sviluppo agile con configurazioni dell'infrastruttura in rapida evoluzione, l'auditing e il controllo automatizzati sono fondamentali. Regole di AWS Config forniscono un approccio gestito al monitoraggio di queste modifiche su tutti i microservizi. Consentono la definizione di politiche di sicurezza specifiche che rilevano, tracciano e inviano automaticamente avvisi sulle violazioni delle politiche.

Ad esempio, se una configurazione API Gateway in un microservizio viene modificata per accettare il traffico HTTP in entrata anziché solo le richieste HTTPS, una AWS Config regola predefinita può rilevare questa violazione della sicurezza. Registra la modifica per il controllo e attiva una notifica SNS, ripristinando lo stato di conformità.

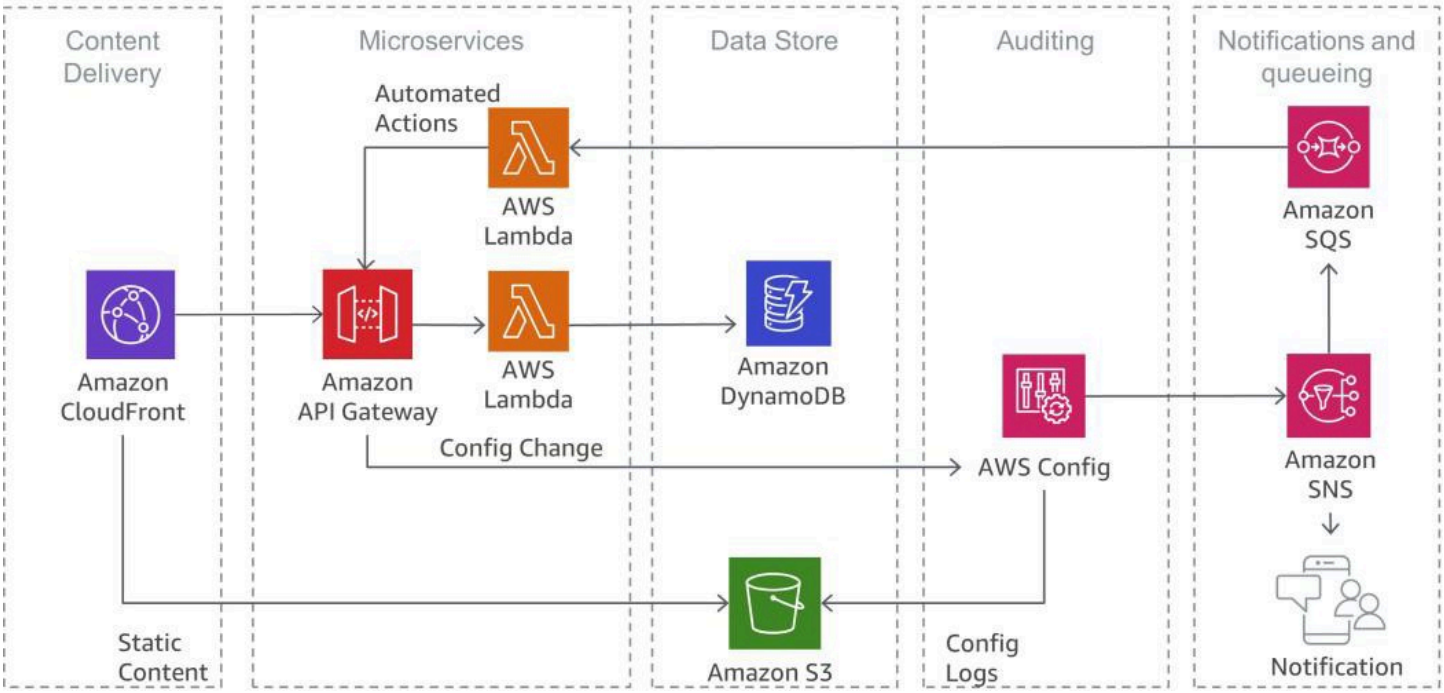

Figura 18: Rilevamento delle violazioni della sicurezza con AWS Config

Conclusioni

L'architettura dei microservizi, un approccio progettuale versatile che offre un'alternativa ai tradizionali sistemi monolitici, aiuta a scalare le applicazioni, aumentare la velocità di sviluppo e promuovere la crescita organizzativa. Grazie alla sua adattabilità, può essere implementato utilizzando contenitori, approcci serverless o una combinazione dei due, adattandosi a esigenze specifiche.

Tuttavia, non è una soluzione. one-size-fits-all Ogni caso d'uso richiede una valutazione meticolosa, dato il potenziale aumento della complessità architettonica e delle esigenze operative. Tuttavia, se affrontati in modo strategico, i vantaggi dei microservizi possono superare in modo significativo queste sfide. La chiave sta nella pianificazione proattiva, in particolare nelle aree di osservabilità, sicurezza e gestione del cambiamento.

È anche importante notare che oltre ai microservizi, esistono framework architettonici completamente diversi, come le architetture di intelligenza artificiale generativa come Retrieval Augmented Generation (RAG), che offrono una gamma di opzioni per soddisfare al meglio le vostre esigenze.

AWS, con la sua solida suite di servizi gestiti, consente ai team di creare architetture di microservizi efficienti e ridurre al minimo efficacemente la complessità. Questo white paper ha lo scopo di guidarvi attraverso i AWS servizi pertinenti e l'implementazione dei modelli chiave. L'obiettivo è fornirti le conoscenze necessarie per sfruttare la potenza dei microservizi AWS, consentendoti di capitalizzarne i vantaggi e trasformare il percorso di sviluppo delle applicazioni.

Collaboratori

Le seguenti persone e organizzazioni hanno contribuito a questo documento:

- Sascha Möllering, architettura delle soluzioni, Amazon Web Services
- Christian Müller, architettura delle soluzioni, Amazon Web Services
- Matthias Jung, architettura delle soluzioni, Amazon Web Services
- Peter Dalbhanjan, Architettura delle soluzioni, Amazon Web Services
- Peter Chapman, architettura delle soluzioni, Amazon Web Services
- Christoph Kassen, architettura delle soluzioni, Amazon Web Services
- Umair Ishaq, architettura delle soluzioni, Amazon Web Services
- Rajiv Kumar, architettura delle soluzioni, Amazon Web Services
- Ramesh Dwarakanath, architettura delle soluzioni, Amazon Web Services
- Andrew Watkins, architettura delle soluzioni, Amazon Web Services
- Yann Stoneman, architettura delle soluzioni, Amazon Web Services
- Mainak Chaudhuri, architettura delle soluzioni, Amazon Web Services
- Gaurav Acharya, Architettura delle soluzioni, Amazon Web Services

Cronologia dei documenti

Per ricevere una notifica sugli aggiornamenti del presente whitepaper, iscriviti al feed RSS.

Modifica	Descrizione	Data
Aggiornamento principale	Sono state aggiunte informazioni su AWS Customer Carbon Footprint Tool, Amazon EventBridge, AWS AppSync (GraphQL) AWS Lambda , Layers, SnapStart Lambda, Large Language Models (LLMs), Amazon Managed Streaming for Apache Kafka (MSK), Amazon Managed Workflows for Apache Airflow (MWAA), Amazon VPC Lattice,. AWS AppConfig È stata aggiunta una sezione separata sull'ottimizzazione dei costi e la sostenibilità.	31 luglio 2023
Aggiornamenti minori	Aggiunto Well-Architected all'abstract.	13 aprile 2022
Aggiornamento del whitepaper	Integrazione di Amazon EventBridge, AWS OpenTelemetry, AMP, AMG, Container Insights, modifiche di testo minori.	9 novembre 2021
Aggiornamenti minori	Layout di pagina modificato	30 aprile 2021
Aggiornamenti minori	Modifiche di testo minori.	1 agosto 2019
Aggiornamento del whitepaper	Integrazione di Amazon EKS, AWS Fargate, Amazon MQ,	1 giugno 2019

	AWS PrivateLink, AWS App Mesh, AWS Cloud Map	
Aggiornamento del whitepaper	Integrazione di flussi di eventi AWS Step Functions, AWS X-Ray ed ECS.	1 settembre 2017
Pubblicazione iniziale	Pubblicata l'implementazione dei microservizi su AWS.	1° dicembre 2016

> **Note**
>
> Per abbonarti agli aggiornamenti RSS, devi avere un plug-in RSS abilitato per il browser che stai utilizzando.

Note

I clienti sono responsabili della propria valutazione indipendente delle informazioni contenute nel presente documento. Questo documento: (a) è solo a scopo informativo, (b) rappresenta le offerte e le pratiche attuali di AWS prodotti, che sono soggette a modifiche senza preavviso, e (c) non crea alcun impegno o assicurazione da parte dei suoi affiliati, AWS fornitori o licenzianti. AWS i prodotti o i servizi sono forniti «così come sono» senza garanzie, dichiarazioni o condizioni di alcun tipo, esplicite o implicite. Le responsabilità e le responsabilità dei AWS propri clienti sono regolate da AWS accordi e il presente documento non fa parte di, né modifica, alcun accordo tra AWS e i suoi clienti.

AWS Glossario

Per la AWS terminologia più recente, consultate il AWS glossario nella sezione Reference.Glossario AWS

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