Whitepaper AWS

Architetture serverless multi-livello di AWS con Amazon API Gateway e AWS Lambda

Architetture serverless multi-livello di AWS con Amazon API Gateway e AWS Lambda : Whitepaper AWS

Table of Contents

Architetture serverless multi-livello di AWS con Amazon API Gateway e AWS Lambda

Data di pubblicazione: 20 ottobre 2021 (Revisioni del documento)

Riassunto

Questo whitepaper illustra come le novità di Amazon Web Services (AWS) possono essere utilizzate per modificare il modo in cui si progettano architetture multi-livello e si implementano pattern comuni come microservizi, back-end mobili e applicazioni a pagina singola. Architetti e sviluppatori possono impiegare Amazon API Gateway, AWS Lambda e altri servizi per ridurre i cicli operativi e di sviluppo necessari per creare e gestire applicazioni multi-livello.

Introduzione

L'applicazione multi-livello (a tre livelli, a n livelli e così via) è stata un pattern di architettura fondamentale per decenni e rimane un pattern popolare per le applicazioni rivolte agli utenti. Sebbene il linguaggio utilizzato per descrivere un'architettura multi-livello vari, un'applicazione multi-livello è generalmente composta dagli elementi seguenti:

- Livello di presentazione: componente con cui l'utente interagisce direttamente (ad esempio, pagine Web e interfacce utente delle applicazioni per dispositivi mobili).
- Livello logico: codice necessario per tradurre le azioni dell'utente in funzionalità dell'applicazione (ad esempio, le operazioni di database CRUD e l'elaborazione dei dati).
- Livello dati: supporti di archiviazione (ad esempio, database, archivi oggetti, cache e file system) che contengono i dati rilevanti per l'applicazione.

Il pattern di architettura multi-livello fornisce un framework generale per garantire che i componenti applicativi disaccoppiati e scalabili in modo indipendente possano essere sviluppati e gestiti separatamente (spesso da team diversi).

Come conseguenza di questo pattern in cui la rete (un livello deve effettuare una chiamata di rete per interagire con un altro livello) funge da confine tra i livelli, lo sviluppo di un'applicazione multi-livello spesso richiede la creazione di più componenti applicativi indifferenziati. Alcuni di questi componenti sono:

- Codice che definisce una coda di messaggi per la comunicazione tra i livelli
- Codice che definisce un'interfaccia del programma dell'applicazione (API) e un modello di dati
- Codice relativo alla sicurezza che garantisce l'accesso appropriato all'applicazione

Tutti questi possono essere considerati componenti "standard" che, sebbene necessari nelle applicazioni multi-livello, non variano di molto nella loro implementazione da un'applicazione all'altra.

AWS offre una serie di servizi per la creazione di applicazioni multi-livello serverless, in grado di semplificare notevolmente il processo di implementazione di tali applicazioni nell'ambiente di produzione e di eliminare il sovraccarico associato alla gestione tradizionale dei server. Amazon API Gateway, un servizio per la creazione e la gestione delle API, e AWS Lambda, un servizio per l'esecuzione di funzioni di codice arbitrarie, possono essere utilizzati in contemporanea per semplificare la creazione di solide applicazioni multi-livello.

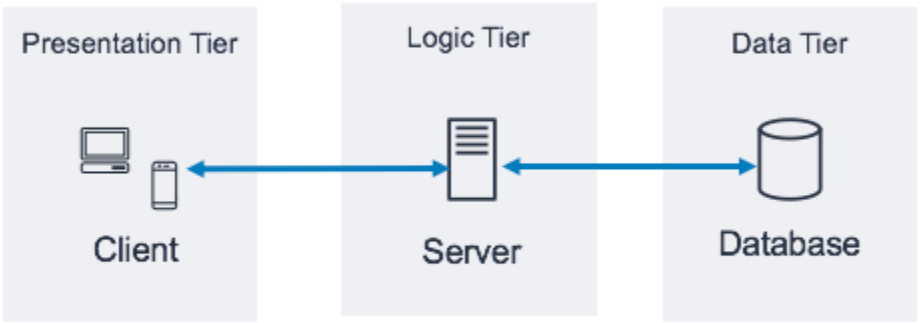
L'integrazione di Amazon API Gateway con AWS Lambda consente di avviare funzioni di codice definite dall'utente direttamente tramite richieste HTTPS. Indipendentemente dal volume delle richieste, sia API Gateway che Lambda si ridimensionano automaticamente per supportare le esigenze delle applicazioni (consulta la sezione Quote e note importanti API Gateway Amazon API Gateway per informazioni sulla scalabilità). Combinando questi due servizi, potrai creare un livello che ti consente di scrivere solo il codice pertinente alla tua applicazione, senza doverti concentrare su vari altri aspetti indifferenziati dell'implementazione di un'architettura multi-livello, come la progettazione dell'architettura per l'alta disponibilità, la scrittura di SDK client, la gestione dei server e del sistema operativo (OS), il ridimensionamento e l'implementazione di un meccanismo di autorizzazione client.

API Gateway e Lambda consentono la creazione di un livello logico serverless. A seconda dei requisiti dell'applicazione, AWS offre anche opzioni per creare un livello di presentazione serverless (ad esempio, con Amazon CloudFront e Amazon Simple Storage Service) e un livello dati (ad esempio, Amazon Aurora, Amazon DynamoDB).

Questo whitepaper è incentrato sull'esempio più popolare di architettura multi-livello, l'applicazione Web a tre livelli. Tuttavia, è possibile applicare questo pattern multi-livello ben oltre una tipica applicazione Web a tre livelli.

Panoramica sull'architettura a tre livelli

L'architettura a tre livelli è l'implementazione più diffusa dell'architettura multi-livello e consiste in un singolo livello di presentazione, un livello logico e un livello dati. La figura seguente mostra un esempio di applicazione a tre livelli semplice e generica.



Pattern di architettura per applicazioni a tre livelli

Online puoi trovare tante ottime risorse da consultare per saperne di più sul pattern di architettura a tre livelli in generale. Questo whitepaper è incentrato su un pattern di implementazione specifico di questa architettura tramite Amazon API Gateway e AWS Lambda.

Livello logico serverless

Il livello logico dell'architettura a tre livelli rappresenta il cervello dell'applicazione. È qui che si utilizza Amazon API Gateway ed è qui che AWS Lambda può avere un impatto maggiore rispetto a un'implementazione tradizionale basata su server. Le funzioni di questi due servizi consentono di creare un'applicazione serverless altamente disponibile, scalabile e sicura. In un modello tradizionale, l'applicazione potrebbe richiedere migliaia di server; tuttavia, utilizzando Amazon API Gateway e AWS Lambda non hai la responsabilità della gestione di qualsiasi numero di server. Inoltre, utilizzando tutti questi servizi gestiti in contemporanea, potrai ottenere i seguenti vantaggi:

- AWS Lambda:
  - Nessun sistema operativo da scegliere, proteggere, gestire o a cui applicare patch
  - Nessun server da ridimensionare o monitorare
  - Riduzione dei rischi per i costi derivanti da un eccessivo provisioning
  - Riduzione dei rischi per le prestazioni derivanti da un provisioning insufficiente

- Amazon API Gateway:
  - Meccanismi semplificati per implementare, monitorare e proteggere le API
  - Prestazioni dell'API migliorate grazie alla memorizzazione nella cache e alla distribuzione di contenuti

AWS Lambda

AWS Lambda è un servizio di calcolo che consente di eseguire funzioni di codice arbitrario in uno qualsiasi dei linguaggi supportati (Node.js, Python, Ruby, Java, Go, .NET. Per ulteriori informazioni, consulta le domande frequenti su Lambda) senza doversi occupare del provisioning, della gestione o del dimensionamento dei server. Le funzioni Lambda vengono eseguite in un container isolato e gestito e sono avviate in risposta a un evento che può essere uno dei numerosi trigger programmatici resi disponibili da AWS, chiamato origine eventi. Consulta le domande frequenti su Lambda per informazioni su tutte le origini eventi.

Molti casi d'uso comuni di Lambda ruotano intorno ai flussi di lavoro di elaborazione dati basati su eventi, come l'elaborazione dei file archiviati in Amazon S3 o lo streaming dei record di dati da Amazon Kinesis. Se utilizzata insieme ad Amazon API Gateway, una funzione Lambda esegue le funzionalità di un tipico servizio Web: avvia il codice in risposta a una richiesta HTTPS del

client; API Gateway funge da porta principale del livello logico e AWS Lambda richiama il codice dell'applicazione.

La logica di business va qui, non sono necessari server

Lambda richiede la scrittura di funzioni di codice, chiamate gestori, che verranno eseguite quando avviate da un evento. Per utilizzare Lambda con API Gateway, puoi configurare API Gateway sull'avvio delle funzioni del gestore quando si verifica una richiesta HTTPS verso l'API. In un'architettura multi-livello serverless, ciascuna delle API create in API Gateway si integrerà con una funzione Lambda (e il gestore al suo interno) che richiama la logica di business richiesta.

L'utilizzo delle funzioni di AWS Lambda per comporre il livello logico consente di definire il livello di granularità desiderato per l'esposizione della funzionalità dell'applicazione (una funzione Lambda per API o una funzione Lambda per metodo API). All'interno della funzione Lambda, il gestore può raggiungere qualsiasi altra dipendenza (ad esempio, altri metodi che hai caricato con il codice, librerie, binari nativi e servizi Web esterni) o persino altre funzioni Lambda.

Per creare o aggiornare una funzione Lambda è necessario caricare il codice come pacchetto di implementazione Lambda sotto forma di file compresso in un bucket Amazon S3 o creare un pacchetto di codice e dipendenze sotto forma di immagine del container. Le funzioni possono utilizzare diversi metodi di implementazione, come la Console di gestione AWS, l'esecuzione di AWS Command Line Interface (AWS CLI) o l'esecuzione di modelli Infrastructure as Code o di framework come AWS CloudFormation, AWS Serverless Application Model (AWS SAM) o AWS Cloud Development Kit (AWS CDK). Quando crei la funzione utilizzando uno di questi metodi, occorre specificare quale metodo all'interno del pacchetto di implementazione fungerà da gestore della richiesta. È possibile riutilizzare lo stesso pacchetto di implementazione per più definizioni delle funzioni Lambda, dove ogni funzione Lambda potrebbe avere un gestore univoco all'interno dello stesso pacchetto di implementazione.

Sicurezza Lambda

Per essere eseguita, una funzione Lambda deve essere richiamata da un evento o da un servizio consentito da una policy AWS Identity and Access Management (IAM). Tramite le policy IAM, è possibile creare una funzione Lambda che non può essere avviata a meno che non venga richiamata da una risorsa API Gateway definita dall'utente. Tale policy può essere definita utilizzando una policy basata sulle risorse in vari servizi AWS.

Ogni funzione Lambda assume un ruolo IAM che viene assegnato al momento dell'implementazione della funzione Lambda stessa. Questo ruolo IAM definisce gli altri servizi e risorse AWS con cui la

funzione Lambda può interagire (ad esempio, Amazon DynamoDB Amazon S3). Nel contesto della funzione Lambda, si parla di ruolo di esecuzione.

Non archiviare informazioni sensibili all'interno di una funzione Lambda. IAM gestisce l'accesso ai servizi AWS tramite il ruolo di esecuzione Lambda; se è necessario accedere ad altre credenziali (ad esempio, credenziali del database e chiavi API) dall'interno della funzione Lambda, puoi utilizzare AWS Key Management Service (AWS KMS) con variabili di ambiente o utilizzare un servizio come AWS Secrets Manager per mantenere queste informazioni al sicuro quando non sono in uso.

Prestazioni e scalabilità

Il codice estratto come immagine del container da Amazon Elastic Container Registry (Amazon ECR), o da un file compresso caricato su Amazon S3, viene eseguito in un ambiente isolato gestito da AWS. Non è necessario ridimensionare le funzioni Lambda: ogni volta che la funzione riceve una notifica sull'evento, AWS Lambda individua la capacità disponibile all'interno del relativo parco istanze di calcolo ed esegue il codice con le configurazioni di runtime, memoria, disco e timeout definite da te. Con questo pattern, AWS può avviare tutte le copie della funzione necessarie.

Un livello logico basato su Lambda ha sempre le dimensioni giuste per le esigenze dei clienti. La capacità di assorbire rapidamente i picchi di traffico attraverso la scalabilità gestita e l'avvio simultaneo del codice, unita ai prezzi a consumo di Lambda, consente di soddisfare sempre le richieste dei clienti senza pagare per la capacità di calcolo inattiva.

Implementazione e gestione serverless

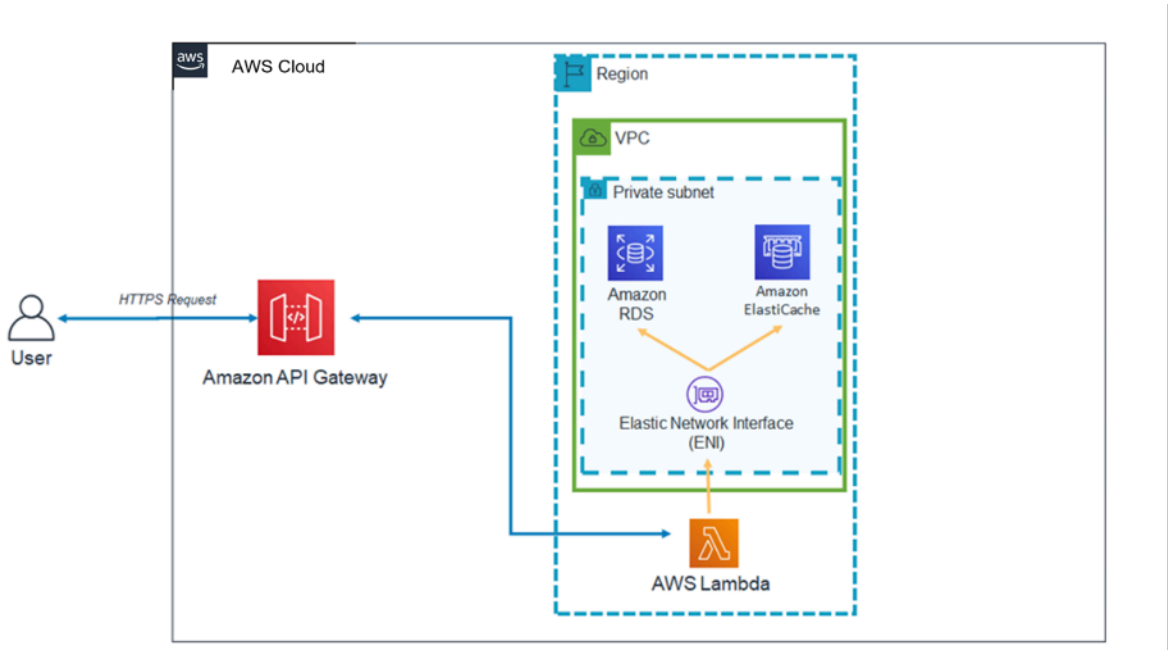
Per implementare e gestire le funzioni Lambda in modo più agevole, utilizza AWS SAM AWS Serverless Application Model (AWS SAM), un framework open source che include:

- Specifiche del modello AWS SAM: sintassi utilizzata per definire le funzioni e descriverne ambienti, autorizzazioni, configurazioni ed eventi per semplificare il caricamento e l'implementazione.
- CLI di AWS SAM: comandi che consentono di verificare la sintassi del modello SAM, richiamare funzioni localmente, eseguire il debug di funzioni Lambda e che offrono funzioni del pacchetto di implementazione.

Puoi inoltre utilizzare AWS CDK, un framework di sviluppo software per la definizione dell'infrastruttura cloud tramite linguaggi di programmazione, ed effettuarne il provisioning tramite CloudFormation. Con CDK è possibile definire le risorse di AWS in modo imperativo, mentre con AWS SAM puoi definirle in modo dichiarativo.

In genere, quando si implementa una funzione Lambda, questa viene richiamata con delle autorizzazioni definite dal ruolo IAM assegnato ed è in grado di raggiungere gli endpoint con connessione Internet. Come fulcro del livello logico, AWS Lambda è il componente che si integra direttamente con il livello dati. Se il livello dati contiene informazioni sensibili sull'azienda o sugli utenti, è importante assicurarsi che questo livello dati sia adeguatamente isolato (in una sottorete privata).

Puoi configurare una funzione Lambda per connetterti a sottoreti private in un Virtual Private Cloud (VPC) nel tuo account AWS se desideri che la funzione Lambda acceda a risorse che non puoi esporre pubblicamente, come le istanze di database private. Quando si collega una funzione a un VPC, Λ crea un'interfaccia di rete elastica per ogni sottorete nella configurazione VPC della funzione e l'interfaccia di rete elastica viene utilizzata per accedere privatamente alle risorse interne.



Pattern di architettura Lambda all'interno di un VPC

L'uso di Lambda con VPC implica che i database e gli altri supporti di archiviazione da cui dipende la logica di business possono essere resi inaccessibili su Internet. Il VPC garantisce inoltre che l'unico modo per interagire con i dati da Internet sia attraverso le API definite da te e le funzioni del codice Lambda scritte da te.

Amazon API Gateway

Amazon API Gateway è un servizio completamente gestito che consente agli sviluppatori di creare, pubblicare, gestire, monitorare e proteggere le API su qualsiasi scala.

I client (ovvero i livelli di presentazione) si integrano con le API esposte tramite API Gateway utilizzando richieste HTTPS standard. L'applicabilità delle API esposte tramite API Gateway a un'architettura multi-livello orientata ai servizi è la capacità di separare singole parti della funzionalità dell'applicazione ed esporre questa funzionalità attraverso gli endpoint REST. Amazon API Gateway dispone di funzioni e qualità specifiche in grado di aggiungere efficaci funzionalità al livello logico.

Integrazione con AWS Lambda

Amazon API Gateway supporta le API sia di tipo REST che HTTP. Un'API di API Gateway è composta da risorse e metodi. Una risorsa è un'entità logica a cui un'app può accedere tramite un percorso della risorsa (ad esempio, `/tickets`). Un metodo corrisponde a una richiesta API inviata a una risorsa API (ad esempio, `GET /tickets`). API Gateway consente di supportare ciascun metodo con una funzione Lambda ovvero, quando richiami l'API tramite l'endpoint HTTPS esposto in API Gateway, API Gateway richiama la funzione Lambda.

Puoi connettere le funzioni API Gateway e Lambda utilizzando integrazioni proxy e integrazioni non proxy.

Integrazioni proxy

In un'integrazione proxy, l'intera richiesta del client HTTPS viene inviata così com'è alla funzione Lambda. API Gateway trasmette l'intera richiesta del client come parametro evento della funzione del gestore Lambda e l'output della funzione Lambda (inclusi codice di stato, intestazioni e così via) viene restituito direttamente al client.

Integrazioni non proxy

In un'integrazione non proxy, configuri il modo in cui i parametri, le intestazioni e il corpo della richiesta del client vengono trasmessi al parametro evento della funzione del gestore Lambda. Inoltre, è possibile configurare il modo in cui l'output Lambda viene ritradotto all'utente.

> **Note**
>
> API Gateway può inoltre fungere da proxy verso risorse serverless aggiuntive esterne a AWS Lambda, come le integrazioni simulate (utili per lo sviluppo iniziale dell'applicazione) e da proxy diretto verso gli oggetti S3.

Prestazioni API stabili in tutte le regioni

Ogni implementazione di Amazon API Gateway include una distribuzione di Amazon CloudFront dietro le quinte. CloudFront è un servizio di distribuzione di contenuti che utilizza la rete globale di posizione edge di Amazon come punto di connessione per i client che utilizzano la tua API. In questo modo, si riduce la latenza di risposta dell'API. Utilizzando più posizioni edge in tutto il mondo, Amazon CloudFront offre anche funzionalità per combattere gli scenari di attacco DDoS (Distributed Denial of Service) distribuiti. Per ulteriori informazioni, consulta il whitepaper sulle best practice AWS per la resilienza DDoS.

Puoi migliorare le prestazioni delle richieste API specifiche utilizzando API Gateway per archiviare le risposte in una cache in memoria opzionale. Questo approccio non solo offre vantaggi in termini di prestazioni nel caso delle richieste API ripetute, ma riduce anche il numero di volte in cui le funzioni Lambda vengono richiamate, il che può portare a una diminuzione del costo complessivo.

Incoraggiare l'innovazione e ridurre il sovraccarico con funzioni incorporate

I costi di sviluppo per la creazione di una nuova applicazione rappresentano un investimento. API Gateway consente alle organizzazioni di sperimentare e innovare più liberamente, riducendo quindi la quantità di tempo necessaria per svolgere determinate attività di sviluppo e diminuendo i costi totali di sviluppo.

Durante le fasi iniziali di sviluppo dell'applicazione, l'implementazione della registrazione e della raccolta dei parametri viene spesso trascurata perché si vuole rilasciare la nuova applicazione più rapidamente. Ciò può comportare un debito tecnico e un rischio operativo se queste funzioni vengono implementate in un'applicazione già in esecuzione nell'ambiente di produzione. Amazon API Gateway si integra senza interruzioni con Amazon CloudWatch, che raccoglie i dati non elaborati da API Gateway e li elabora trasformandoli in parametri leggibili quasi in tempo reale per il monitoraggio dell'esecuzione dell'API. API Gateway supporta inoltre la registrazione degli accessi con report configurabili e il monitoraggio di AWS X-Ray per le operazioni di debug. Ognuna di queste funzioni non richiede la scrittura di codice e può essere adattata nelle applicazioni in esecuzione nell'ambiente di produzione senza rischi per la logica di business di base.

La durata complessiva di un'applicazione potrebbe essere sconosciuta o nota invece come breve. La creazione di un business case per la compilazione di tali applicazioni può essere più semplice se il punto di partenza include già le funzioni gestite fornite da API Gateway e se si sostengono i costi di infrastruttura solo dopo che le API hanno iniziato a ricevere le richieste. Per ulteriori informazioni, consulta Prezzi di Amazon API Gateway.

Eseguire l'iterazione rapidamente e mantenere l'agilità

L'utilizzo di Amazon API Gateway e AWS Lambda per creare il livello logico dell'API ti consente di adattarti rapidamente alle mutevoli esigenze della tua base di utenti semplificando l'implementazione delle API e la gestione delle versioni.

Implementazione in fasi

Quando si implementa un'API in API Gateway, è necessario associare l'implementazione a una fase di API Gateway: ogni fase è una snapshot dell'API e viene resa disponibile per la chiamata delle app client. Utilizzando questa convenzione, potrai implementare facilmente le app nelle fasi di sviluppo, test, gestione temporanea o produzione e spostare le implementazioni tra le diverse fasi. Ogni volta che implementi l'API in una fase, crei una versione diversa dell'API che può essere ripristinata, se necessario. Queste funzioni consentono alle funzioni e alle dipendenze client esistenti di continuare senza interruzioni mentre la nuova funzione viene rilasciata come versione API separata.

Integrazione disaccoppiata con Lambda

L'integrazione tra le API in API Gateway e nella funzione Lambda può essere disaccoppiata utilizzando le variabili di fase di API Gateway e un alias della funzione Lambda. Ciò semplifica e velocizza l'implementazione dell'API. Invece di configurare il nome o l'alias della funzione Lambda direttamente nell'API, è possibile configurare la variabile di fase nell'API che può puntare a un particolare alias nella funzione Lambda. Durante l'implementazione, se modifichi il valore della variabile di fase in modo che punti a un alias della funzione Lambda, l'API eseguirà la versione della funzione Lambda dietro l'alias Lambda per una determinata fase.

Distribuzione di una release Canary

Con rilascio canary si intende una strategia di sviluppo software in cui una nuova versione di un'API viene implementata a scopo di test e contemporaneamente la versione di base rimane implementata come rilascio di produzione per le normali operazioni nella stessa fase. Nell'implementazione di un rilascio canary, il traffico API totale viene separato in modo casuale tra rilascio di produzione e rilascio canary, in base a un rapporto preconfigurato. Le API in API Gateway possono essere configurate per l'implementazione del rilascio canary allo scopo di testare nuove funzioni con un numero limitato di utenti.

Nomi di dominio personalizzati

Puoi scegliere per l'API un nome di URL intuitivo per le aziende al posto del nome di URL fornito da API Gateway. API Gateway fornisce funzioni di configurazione dei domini personalizzati per le

API. Con i nomi di dominio personalizzati, è possibile impostare il nome host dell'API e scegliere un percorso base multi-livello (ad esempio `myservice`, `myservice/cat/v1` o `myservice/dog/v2`) per mappare l'URL alternativo all'API.

Assegnare la priorità alla sicurezza delle API

Tutte le applicazioni devono garantire che solo i client autorizzati abbiano accesso alle proprie risorse API. Quando si progetta un'applicazione multi-livello, è possibile sfruttare i diversi modi in cui Amazon API Gateway contribuisce alla protezione del livello logico:

Sicurezza del transito

Tutte le richieste alle API possono essere effettuate tramite HTTPS per abilitare la crittografia in transito.

API Gateway fornisce certificati SSL/TLS integrati: se utilizzi l'opzione del nome di dominio personalizzato per le API pubbliche, puoi fornire un tuo proprio certificato SSL/TLS utilizzando AWS Certificate Manager. API Gateway supporta anche l'autenticazione mutual TLS (mTLS). mutual TLS migliora la sicurezza dell'API e aiuta a proteggere i tuoi dati da attacchi come lo spoofing dei client o gli attacchi man-in-the middle.

Autorizzazione API

A ogni combinazione di risorsa/metodo che crei come parte della tua API, viene assegnato un Amazon Resource Name (ARN) univoco a cui si può fare riferimento nelle policy AWS Identity and Access Management (IAM).

Esistono tre metodi generali per aggiungere l'autorizzazione a un'API in API Gateway:

- Ruoli e policy IAM: i client utilizzano l'autorizzazione AWS Signature Version 4 (SigV4) e le policy IAM per l'accesso alle API. Le stesse credenziali possono limitare o consentire l'accesso ad altri servizi e risorse AWS in base alle esigenze (ad esempio, bucket Amazon S3 o tabelle di Amazon DynamoDB).
- Pool di utenti di Amazon Cognito: i clienti accedono tramite un pool di utenti di Amazon Cognito e ottengono i token, inclusi nell'intestazione di autorizzazione di una richiesta.
- Autorizzazione per Lambda: consente di definire una funzione Lambda che implementa uno schema di autorizzazione personalizzato che per identificare gli utenti utilizza una strategia del token di connessione (ad esempio OAuth e SAML) o i parametri della richiesta.

Limitazioni di accesso

API Gateway supporta la generazione di chiavi API e l'associazione di queste chiavi a un piano di utilizzo configurabile. Puoi monitorare l'utilizzo delle chiavi API con CloudWatch.

API Gateway supporta la limitazione della larghezza di banda della rete, limiti di velocità e limiti di velocità di burst per ogni metodo nell'API.

API private

Utilizzando API Gateway, puoi creare API REST private a cui è possibile accedere solo dal cloud privato virtuale in Amazon VPC utilizzando un endpoint VPC di interfaccia. Si tratta di un'interfaccia di rete dell'endpoint creata nel VPC.

Utilizzando le policy delle risorse, puoi consentire o rifiutare l'accesso all'API da VPC ed endpoint VPC selezionati, inclusi gli account AWS. Ogni endpoint può essere utilizzato per l'accesso a più API private. Puoi inoltre utilizzare AWS Direct Connect per stabilire una connessione da una rete in locale ad Amazon VPC e accedere all'API privata durante tale connessione.

In tutti i casi, il traffico alla tua API privata utilizza connessioni sicure e non lascia la rete Amazon; è isolato dall'Internet pubblico.

Protezione del firewall tramite AWS WAF

Le API con connessione Internet sono vulnerabili agli attacchi malevoli. AWS WAF è un firewall per applicazioni Web che aiuta a proteggere le API da tali attacchi. Protegge le API dagli exploit Web più comuni come SQL injection e attacchi cross-site scripting (XSS). Puoi utilizzare AWS WAF con API Gateway per proteggere le API.

Livello dati

L'utilizzo di AWS Lambda come livello logico non limita le opzioni di archiviazione dati disponibili nel livello dati. Le funzioni Lambda si connettono a qualsiasi opzione di archiviazione dati includendo il driver di database appropriato nel pacchetto di implementazione Lambda e utilizzano l'accesso basato sui ruoli IAM o le credenziali crittografate (tramite AWS KMS o AWS Secrets Manager).

La scelta di un archivio dati per l'applicazione dipende in larga misura dai requisiti dell'applicazione. AWS offre una serie di archivi dati serverless e non serverless che puoi utilizzare per comporre il livello dati della tua applicazione.

Opzioni di archiviazione dati serverless

Amazon S3 è un servizio di archiviazione di oggetti che offre scalabilità, disponibilità dei dati, sicurezza e prestazioni all'avanguardia nel settore.

Amazon Aurora è un database relazionale compatibile con MySQL e PostgreSQL creato per il cloud che unisce le prestazioni e la disponibilità dei database aziendali tradizionali con la semplicità e l'economicità dei database open source. Aurora offre modelli di utilizzo serverless e tradizionali.

Amazon DynamoDB è un database chiave-valore e di documenti in grado di garantire prestazioni in millisecondi a una cifra, indipendentemente dalle dimensioni. È un database completamente gestito, serverless, multi-regione, multi-master e durevole, con capacità integrate di sicurezza, backup e ripristino, oltre al sistema di cache in memoria per le applicazioni su scala Internet.

Amazon Timestream è un servizio di database di serie temporali rapido, scalabile e completamente gestito pensato per IoT e le applicazioni operative; questo servizio semplifica l'archiviazione e l’analisi di migliaia di miliardi di eventi al giorno a un decimo del costo dei database relazionali. Spinti dalla crescita dei dispositivi IoT, dei sistemi IT e delle macchine industriali intelligenti, i dati di serie temporali, dati che misurano come le cose cambiano nel tempo, è uno dei tipi di dati a crescita più rapida.

Amazon Quantum Ledger Database (Amazon QLDB) è un database di libro mastro completamente gestito che fornisce un registro delle transazioni trasparente, immutabile e verificabile crittograficamente di proprietà di un’autorità attendibile a livello centrale. Amazon QLDB monitora ogni cambiamento dei dati di un’applicazione e conserva uno storico completo e verificabile dei cambiamenti nel corso del tempo.

Amazon Keyspaces (per Apache Cassandra) è un servizio di database compatibile con Apache Cassandra, gestito, scalabile e altamente disponibile. Con Amazon Keyspaces, puoi eseguire i carichi di lavoro Cassandra su AWS utilizzando lo stesso codice applicativo e gli stessi strumenti di sviluppo Cassandra che utilizzi oggi. Non sei tenuto ad eseguire il provisioning, applicare patch o gestire i server né ad installare, manutenere o utilizzare il software. Amazon Keyspaces è serverless, quindi paghi soltanto in base alle risorse che utilizzi e il servizio può dimensionare automaticamente le tabelle in risposta al traffico delle applicazioni.

Amazon Elastic File System (Amazon EFS) offre un file system semplice, serverless, impostabile in maniera permanente ed elastico che consente di condividere i dati dei file senza occuparsi del provisioning o della gestione dell'archiviazione. Può essere utilizzato con servizi AWS Cloud o risorse on-premise ed è stato creato per scalare on demand fino ai petabyte senza interrompere le applicazioni. Con Amazon EFS puoi aumentare o ridurre automaticamente le dimensioni dei file system quando aggiungi o rimuovi file, eliminando la necessità di gestire la capacità ed effettuarne il provisioning per supportare la crescita. Amazon EFS può essere montato con la funzione Lambda che lo rende una valida opzione di archiviazione di file per le API.

Opzioni di archiviazione dati non serverless

Amazon Relational Database Service (Amazon RDS) è un servizio Web gestito che semplifica la configurazione, il funzionamento e la scalabilità dei database relazionali tramite uno qualsiasi dei motori disponibili (Amazon Aurora, PostgreSQL, MySQL, MariaDB, Oracle e Microsoft SQL Server) e in esecuzione su diversi tipi di istanze di database ottimizzate per la memoria, le prestazioni o l'I/O.

Amazon Redshift è un servizio di data warehouse in scala petabyte e completamente gestito nel cloud.

Amazon ElastiCache è una implementazione completamente gestita di Redis o Memcached. Distribuisci, gestisci e dimensiona con facilità i più popolari archivi dati in memoria compatibili con open source.

Amazon Neptune è un servizio di database a grafo rapido, affidabile e completamente gestito che semplifica la creazione e l'esecuzione di applicazioni che funzionano con set di dati altamente connessi. Neptune supporta i modelli di grafo più diffusi, i grafi delle proprietà e il Resource Description Framework (RDF) di W3C, con le relative sintassi di query che consentono di creare facilmente query efficaci su set di dati altamente connessi.

Amazon DocumentDB (compatibile con MongoDB) è un servizio di database di documenti veloce, scalabile, a disponibilità elevata e completamente gestito che supporta i carichi di lavoro MongoDB.

Infine, come livello dati di un'applicazione multi-livello, puoi utilizzare anche gli archivi dati in esecuzione in modo indipendente su Amazon EC2

Livello di presentazione

Il livello di presentazione è responsabile dell'interazione con il livello logico tramite gli endpoint REST di API Gateway esposti su Internet. Qualsiasi client o dispositivo compatibile con HTTPS può comunicare con questi endpoint, dando al livello di presentazione la flessibilità necessaria per assumere diverse forme (applicazioni desktop, applicazioni per dispositivi mobili, pagine Web, dispositivi IoT e così via). A seconda delle tue esigenze, il livello di presentazione può utilizzare le seguenti offerte serverless di AWS: Qualsiasi client o dispositivo compatibile con HTTPS può comunicare con questi endpoint, dando al livello di presentazione la flessibilità necessaria per assumere molte forme (applicazioni desktop, applicazioni per dispositivi mobili, pagine Web, dispositivi IoT e così via). A seconda delle tue esigenze, il livello di presentazione può utilizzare le seguenti offerte serverless di AWS:

- Amazon Cognito: un servizio di sincronizzazione di dati e identità utente serverless che consente di aggiungere in modo rapido ed efficiente funzioni di registrazione degli utenti, di accesso e di controllo degli accessi alle applicazioni Web e per dispositivi mobili. Amazon Cognito ricalibra le risorse per milioni di utenti e supporta l'accesso con provider di identità social, come Facebook, Google e Amazon, e provider di identità aziendali attraverso SAML 2.0.
- Amazon S3 con CloudFront: consente di servire siti Web statici, ad esempio applicazioni a pagina singola, direttamente da un bucket S3 senza richiedere il provisioning di un server Web. Puoi utilizzare CloudFront come rete per la distribuzione di contenuti gestita (CDN) per migliorare le prestazioni e abilitare SSL/TL tramite certificati gestiti o personalizzati.

AWS Amplify è un insieme di strumenti e servizi che possono essere utilizzati insieme o separatamente per consentire agli sviluppatori di applicazioni per dispositivi mobili e Web front-end di creare soluzioni complete e scalabili, con tecnologia AWS. Amplify offre un servizio completamente gestito per l'implementazione e l'hosting di applicazioni Web statiche a livello mondiale, servite dalla CDN affidabile di Amazon, con centinaia di punti di presenza in tutto il globo e flussi di lavoro di CI/CD integrati che permettono di accelerare il ciclo di rilascio dell'applicazione. Amplify supporta i framework Web più diffusi tra cui JavaScript, React, Angular, Vue, Next.js e le piattaforme mobili tra cui Android, iOS, React Native, Ionic e Flutter. A seconda delle configurazioni di rete e dei requisiti dell'applicazione, potrebbe essere necessario abilitare le API di API Gateway per garantire la conformità con la condivisione delle risorse tra origini (CORS). La conformità CORS consente ai browser Web di richiamare direttamente le API dalle pagine Web statiche.

Quando implementi un sito Web con CloudFront, riceverai un nome di dominio CloudFront da usare per raggiungere l'applicazione (ad esempio, `d2d47p2vcczkh2.cloudfront.net`). Puoi utilizzare Amazon Route 53 per registrare i nomi di dominio e indirizzarli alla distribuzione CloudFront oppure per indirizzare i nomi di dominio già posseduti alla distribuzione CloudFront. Ciò consente agli utenti di accedere al tuo sito utilizzando un nome di dominio conosciuto. Tieni presente che puoi utilizzare Route 53 anche per assegnare un nome di dominio personalizzato alla distribuzione API Gateway, consentendo quindi agli utenti di richiamare le API utilizzando nomi di dominio conosciuti.

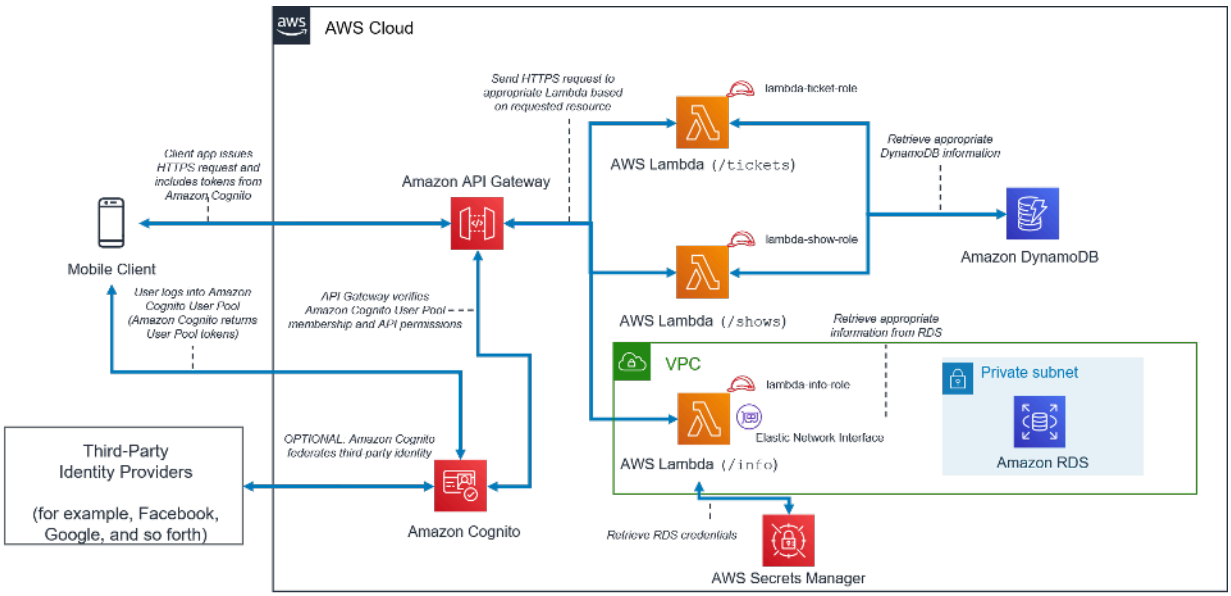
Pattern di architettura di esempio

Puoi implementare i pattern di architettura più diffusi utilizzando API Gateway e AWS Lambda come livello logico. In questo whitepaper sono descritti i pattern di architettura più diffusi che utilizzano i livelli logici basati su AWS Lambda:

- Back-end mobile: le applicazioni per dispositivi mobili comunicano con API Gateway e Lambda per accedere ai dati delle applicazioni. Questo pattern può essere esteso ai client HTTPS generici che non utilizzano risorse AWS serverless per ospitare le risorse del livello di presentazione (come client desktop, server Web in esecuzione su EC2 e così via).
- Applicazione a pagina singola: le applicazioni a pagina singola ospitate in Amazon S3 e CloudFront comunicano con API Gateway e AWS Lambda per accedere ai dati delle applicazioni.
- Applicazione Web: le applicazioni Web sono back-end per applicazioni Web generici e basati su eventi che utilizzano AWS Lambda con API Gateway per la logica di business. Utilizzano anche DynamoDB come proprio database e Amazon Cognito per la gestione degli utenti. Tutto il contenuto statico è ospitato tramite Amplify.

Oltre a questi due pattern, in questo whitepaper viene illustrata l'applicabilità di Lambda e API Gateway a un'architettura generale di microservizi. Un'architettura di microservizi è un pattern popolare che, sebbene diverso dall'architettura standard a tre livelli, prevede il disaccoppiamento dei componenti applicativi e la loro implementazione come singole unità di funzionalità senza stato che comunicano tra loro.

Back-end per dispositivi mobili



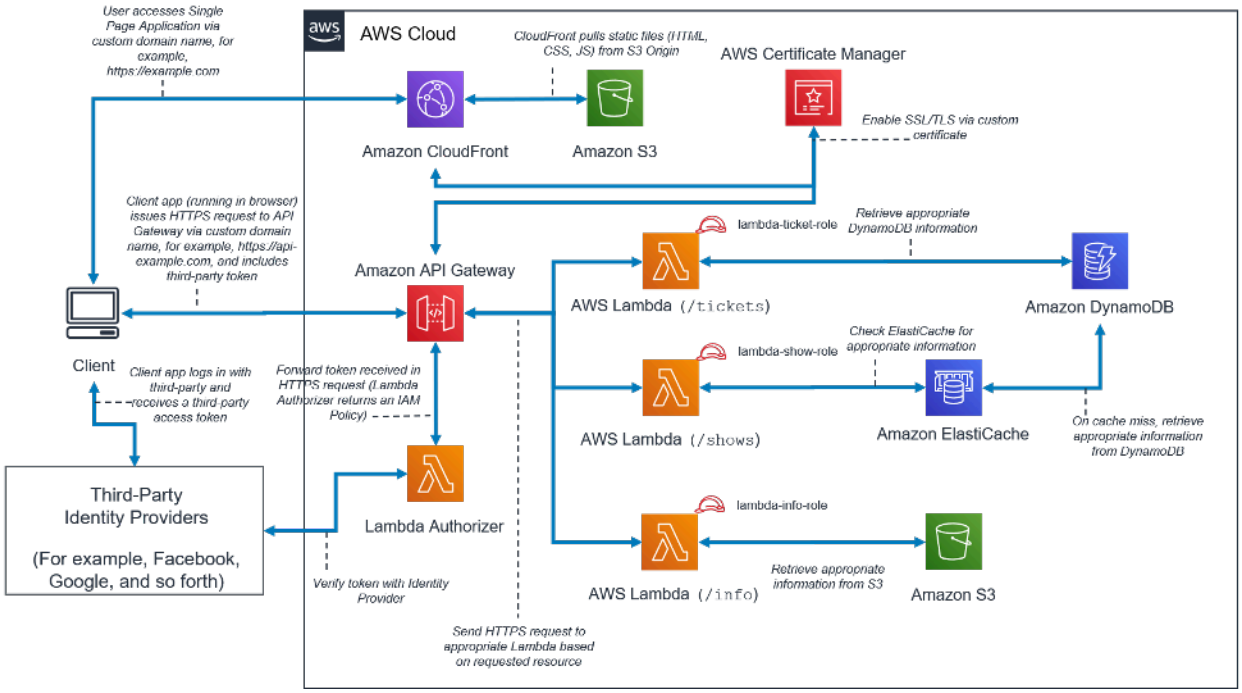
Pattern di architettura per back-end per dispositivi mobili serverless

Tabella 1 - Componenti del livello di back-end per dispositivi mobili

Livello	Componenti
Presentazione	Applicazione per dispositivi mobili in esecuzion e su un dispositivo utente.
Logic (Logica)	Amazon API Gateway con AWS Lambda. Questa architettura mostra tre servizi esposti (`/tickets`, `/shows` e `/info`). Gli endpoint di API Gateway sono protetti dai pool di utenti di Amazon Cognito. In questo metodo, gli utenti accedono ai pool di utenti di Amazon Cognito (utilizzando una terza parte federata, se necessario) e ricevono token e ID di accesso da utilizzare per autorizzare le chiamate di API Gateway. A ciascuna funzione Lambda viene assegnato un ruolo di Identity and Access Managemen

Livello	Componenti
	t (IAM) per fornire l'accesso all'origine dati appropriata.
Dati	DynamoDB viene utilizzato per i `/tickets` servizi e `/shows`. Amazon RDS viene utilizzato per il servizio `/info`. Questa funzione Lambda recupera le credenziali di Amazon RDS da AWS Secrets Manager e utilizza un'interfaccia di rete elastica per accedere alla sottorete privata.

Applicazione a pagina singola

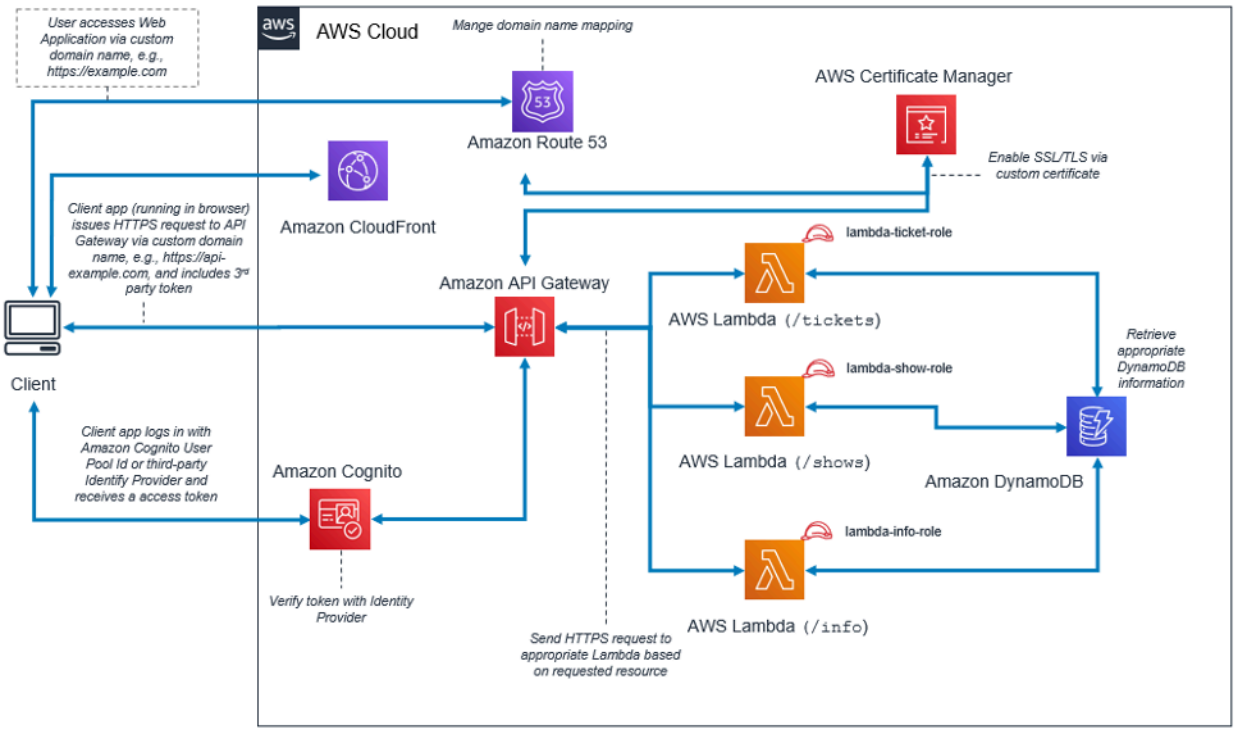


Pattern di architettura per applicazioni a pagina singola serverless

Tabella 2 - Componenti delle applicazioni a pagina singola

Livello	Componenti
Presentazione	Contenuti statici di siti Web ospitati in Amazon S3, distribuiti da CloudFront. AWS Certificate Manager consente di utilizzare un certificato SSL/TLS personalizzato.
Logic (Logica)	API Gateway con AWS Lambda. Questa architettura mostra tre servizi esposti (`/tickets`, `/shows` e `/info`). Gli endpoint di API Gateway sono protetti da un'autori zzazione per Lambda. In questo metodo, gli utenti accedono tramite un provider di identità di terze parti e ricevono ID e token di accesso. Questi token sono inclusi nelle chiamate di API Gateway e l'autorizzazione per Lambda li convalida e genera una policy IAM contenente le autorizzazioni per l'avvio dell'API. A ciascuna funzione Lambda viene assegnato un ruolo IAM per fornire l'accesso all'origine dati appropriata.
Dati	Amazon DynamoDB viene utilizzato per i servizi `/tickets` e `/shows`. Amazon ElastiCache viene utilizzato dal servizio `/shows` per migliorare le prestazioni del database. I mancati riscontri nella cache vengono inviati a DynamoDB. Amazon S3 viene utilizzato per ospitare i contenuti statici utilizzati da `/info service`.

Applicazione Web



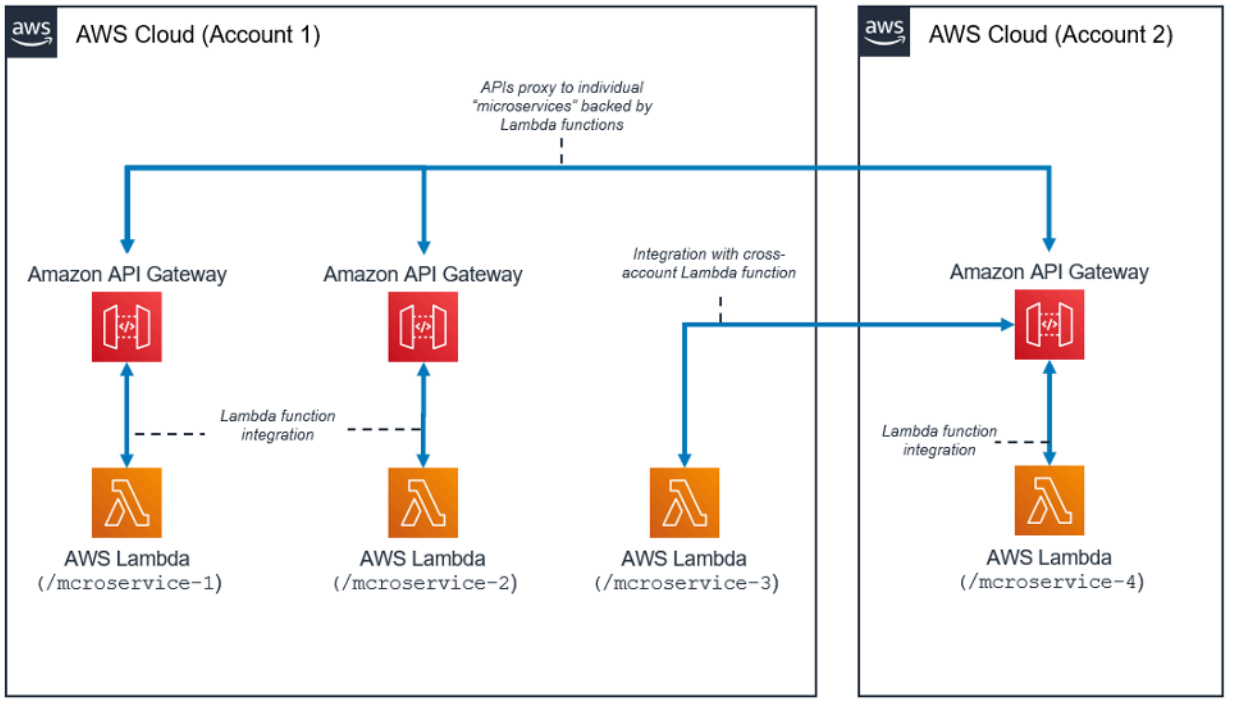
Pattern di architettura per applicazioni Web

Tabella 3 - Componenti dell'applicazione Web

Livello	Componenti
Presentazione	L'applicazione front-end è costituita da tutti i contenuti statici (HTML, CSS, JavaScript e immagini) generati dalle utility React, come create-react-app. Amazon CloudFront ospita tutti questi oggetti. Quando viene utilizzata, l'applicazione Web scarica tutte le risorse sul browser e inizia ad essere eseguita da lì. L'applicazione Web si connette al back-end che esegue la chiamata alle API.
Logic (Logica)	Il livello logico è composto dalle funzioni Lambda con le API REST di API Gateway come front-end.

Livello	Componenti
	In questa architettura sono presenti più servizi esposti. Vi sono diverse funzioni Lambda, ognuna delle quali gestisce un aspetto diverso dell'applicazione. Le funzioni Lambda si trovano dietro API Gateway e sono accessibili tramite percorsi URL dell'API. L'autenticazione degli utenti viene gestita tramite i pool di utenti di Amazon Cognito o i provider di utenti federati. API Gateway utilizza l'integrazione pronta all'uso con Amazon Cognito. Solo dopo che un utente è stato autenticato, il client riceverà un token JSON Web Token (JWT) che dovrà quindi utilizzare quando effettua le chiamate API. A ciascuna funzione Lambda viene assegnato un ruolo IAM per fornire l'accesso all'origine dati appropriata.
Dati	In questo particolare esempio, DynamoDB viene utilizzato per l'archiviazione dati, ma è possibile utilizzare altri database o servizi di archiviazione Amazon appositamente creati a seconda del caso d'uso e dello scenario di utilizzo.

Microservizi con Lambda



Pattern di architettura per microservizi con Lambda

Il pattern di architettura dei microservizi non è legato alla tipica architettura a tre livelli; tuttavia, questo popolare pattern può offrire vantaggi significativi derivanti dall'uso di risorse serverless.

In questa architettura, ciascuno dei componenti applicativi viene disaccoppiato e implementato e gestito in modo indipendente. Un'API creata con Amazon API Gateway e le funzioni successivamente avviate da AWS Lambda, sono tutto ciò che serve per creare un microservizio. Il team può utilizzare questi servizi per disaccoppiare e frammentare l'ambiente al livello di granularità desiderato.

In generale, un ambiente di microservizi può presentare le seguenti difficoltà: sovraccarico ripetuto per la creazione di ogni nuovo microservizio, problemi con l'ottimizzazione della densità e dell'utilizzo del server, complessità dell'esecuzione simultanea di più versioni di più microservizi e proliferazione dei requisiti del codice lato client per l'integrazione con molti servizi separati.

Se invece crei microservizi utilizzando risorse serverless, questi problemi diventano più facili da risolvere e, in alcuni casi, semplicemente spariscono. Il pattern con microservizi serverless riduce gli ostacoli relativi alla creazione di ciascun microservizio successivo (API Gateway consente persino la clonazione delle API esistenti e l'utilizzo delle funzioni Lambda in altri account). L'ottimizzazione dell'utilizzo del server non è più rilevante in questo pattern. Infine, Amazon API Gateway fornisce

SDK client generati a livello di programmazione in molti dei linguaggi più comuni per ridurre il sovraccarico dell'integrazione.

Conclusione

Il pattern di architettura multi-livello incoraggia la best practice di creazione di componenti applicativi semplici da gestire, disaccoppiare e dimensionare. Quando si crea un livello logico in cui l'integrazione avviene tramite API Gateway e il calcolo avviene all'interno di AWS Lambda, si realizzano questi obiettivi in modo notevolmente più agevole. Insieme, questi servizi forniscono un front-end API HTTPS per i clienti e un ambiente sicuro in cui applicare la logica di business eliminando il sovraccarico legato alla gestione di una tipica infrastruttura basata su server.

Collaboratori

I collaboratori di questo documento includono:

- Andrew Baird, AWS Solutions Architect
- Bryant Bost, AWS ProServe Consultant
- Stefano Buliani, Senior Product Manager, Tech, AWS Mobile
- Vyom Nagrani, Senior Product Manager, AWS Mobile
- Ajay Nair, Senior Product Manager, AWS Mobile
- Rahul Popat, Global Solutions Architect
- Brajendra Singh, Senior Solutions Architect

Approfondimenti

Per ulteriori informazioni, fai riferimento a:

- Whitepaper e guide AWS

Revisioni del documento

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

update-history-change	update-history-description	update-history-date
Whitepaper aggiornato	Aggiornato con nuove funzioni e pattern di servizio.	20 ottobre 2021
Whitepaper aggiornato	Aggiornato con nuove funzioni e pattern di servizio.	1 giugno 2021
Whitepaper aggiornato	Aggiornato con nuove funzioni del servizio.	25 settembre 2019
Pubblicazione iniziale	Whitepaper pubblicato.	1 novembre 2015

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.