Architetture AWS multi-tier serverless

Utilizzo di Amazon API Gateway e AWS Lambda

Novembre 2015

Note

Il presente documento è fornito a solo scopo informativo. In esso sono illustrate le attuali offerte di prodotti e le prassi di AWS alla data di pubblicazione del documento, offerte che sono soggette a modifica senza preavviso.
È responsabilità dei clienti effettuare una propria valutazione indipendente delle informazioni contenute nel presente documento e dell'uso dei prodotti o dei servizi di AWS, ciascuno dei quali viene fornito "così com'è", senza garanzie di alcun tipo, né esplicite né implicite. Il presente documento non dà origine a garanzie, rappresentazioni, impegni contrattuali, condizioni o assicurazioni da parte di AWS, delle sue società affiliate, dei suoi fornitori o dei licenzianti. Le responsabilità di AWS nei confronti dei propri clienti sono definite dai contratti AWS e il presente documento non costituisce parte né modifica qualsivoglia contratto tra AWS e i suoi clienti.

Contenuti

Sintesi

Il whitepaper illustra come le innovazioni di Amazon Web Services (AWS) possano cambiare le modalità di progettazione delle architetture multi-tier per modelli diffusi come microservizi, back end per dispositivi mobili e siti Web pubblici. Architetti e sviluppatori possono ora utilizzare un modello di implementazione che comprende Amazon API Gateway e AWS Lambda per abbreviare i cicli di sviluppo e operativi richiesti per la creazione e la gestione operativa delle applicazioni multi-tier.

Introduzione

L'applicazione multi-tier (a tre livelli, a n livelli, ecc.) è da decenni uno dei cardini dei modelli architetturali. Il modello multi-tier fornisce ottime linee guida da seguire per ottenere componenti di applicazioni disassociati e scalabili, gestibili separatamente (spesso da parte di team distinti). Le applicazioni multi-tier sono spesso realizzate secondo un approccio SOA (Service-Oriented Architecture, architettura orientata ai servizi) per l'utilizzo dei servizi Web. Secondo tale approccio, la rete rappresenta il confine tra i livelli. Esistono, tuttavia, molti aspetti indifferenziati relativi alla creazione di un nuovo livello di servizio Web nel quadro dell'applicazione. La maggior parte del codice scritto in un'applicazione Web multi-tier è il risultato diretto del modello stesso. Tra gli esempi si possono menzionare il codice che integra un livello ad un altro, il codice che definisce un'API e un modello di dati che i livelli utilizzano per capirsi e il codice correlato alla sicurezza che garantisce che i punti d'integrazione dei livelli non siano esposti in modo indesiderato.

Amazon API Gateway[1], un servizio per la creazione e la gestione delle API, e AWS Lambda[2], un servizio per l'esecuzione di funzioni del codice arbitrario, possono essere utilizzati insieme per semplificare la creazione di applicazioni multi-tier affidabili.

L'integrazione tra Amazon API Gateway e AWS Lambda consente l'attivazione delle funzioni definite dall'utente direttamente tramite una richiesta HTTPS definita dall'utente. Indipendentemente dal volume di richieste necessario, sia API Gateway sia Lambda si dimensionano automaticamente per supportare esattamente le esigenze dell'applicazione. Grazie alla loro sinergia, si può creare un livello per l'applicazione che permette di scrivere il codice pertinente per l'applicazione e di *non* disperdere l'attenzione su vari altri aspetti indifferenziati dell'attuazione di un'architettura multi-tier, come la definizione di un'architettura ad alta disponibilità, la scrittura di SDK client, la gestione del server/sistema operativo (OS), il dimensionamento e l'implementazione di un meccanismo di autorizzazione client.

Recentemente, AWS ha reso nota la capacità di creare funzioni Lambda eseguibili all'interno di Amazon Virtual Private Cloud (Amazon VPC)[3]. Tale caratteristica estende i vantaggi dell'integrazione tra API Gateway e Lambda a una serie di casi d'uso in cui è necessaria la privacy della rete, ad esempio, quando occorre integrare il servizio Web con un database relazionale che contiene informazioni sensibili. L'integrazione di Lambda e Amazon VPC ha ampliato indirettamente le funzionalità di Amazon API Gateway, perché consente agli sviluppatori di definire la propria gamma di API HTTPS accessibili da Internet, mentre il backend resta privato e sicuro nell'ambito di Amazon VPC. I vantaggi di questo potente modello sono osservabili in ogni livello dell'architettura multi-tier. Il whitepaper si sofferma in particolare sull'esempio più conosciuto di architettura multi-tier, ossia un'applicazione Web **a tre livelli**. Tale modello multi-tier, tuttavia, può trovare applicazione ben al di là di una tipica applicazione Web a tre livelli.

Panoramica dell'architettura a tre livelli

L'architettura a tre livelli è un modello diffuso per le applicazioni rivolte agli utenti. I livelli che compongono l'architettura includono il **livello di presentazione**, il **livello logico** e il **livello di dati**. Il livello di presentazione è la componente con cui gli utenti interagiscono direttamente (ad esempio una pagina Web, l'UI di un'app mobile, ecc.). Il livello logico contiene il codice necessario a tradurre le azioni dell'utente al livello di presentazione nella funzionalità che determina il comportamento dell'applicazione. Il livello di dati è costituito dai supporti di storage (database, object store, cache, file system, ecc.) che archiviano i dati pertinenti per l'applicazione. La Figura 1 mostra un esempio di una semplice applicazione a tre livelli.

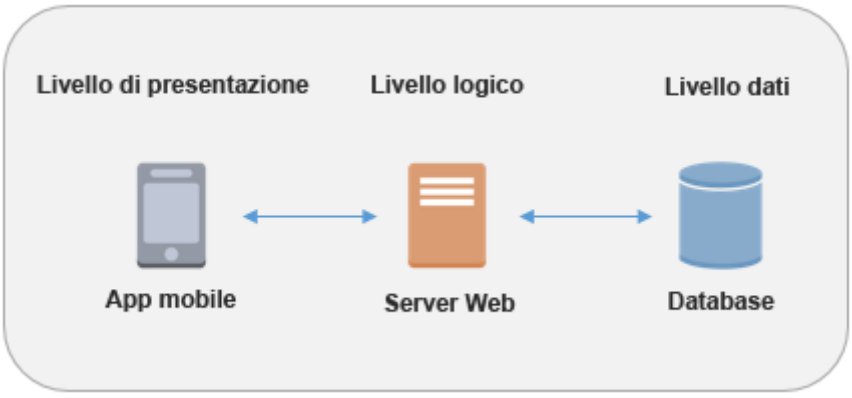

Figura 1. Modello architetturale per una semplice applicazione a tre livelli

Sono disponibili molte risorse online interessanti che forniscono ulteriori informazioni sul modello *generale* di un'architettura a tre livelli. Il whitepaper illustra un modello di implementazione specifico per tale architettura con l'ausilio di Amazon API Gateway e AWS Lambda.

Il livello logico serverless

Il livello logico dell'architettura a tre livelli rappresenta il cervello dell'applicazione. Ecco perché l'integrazione tra Amazon API Gateway e AWS Lambda per formare il livello logico può essere rivoluzionaria. Le caratteristiche dei due servizi permettono di realizzare un' applicazione di produzione serverless altamente disponibile, scalabile e sicura. L'applicazione potrebbe utilizzare migliaia di server, ma con questo modello non occorre gestirne neppure uno. L'utilizzo combinato di questi servizi gestiti, inoltre, consente di ottenere I seguenti vantaggi:

- Nessun sistema operativo da scegliere, mettere in sicurezza, correggere o gestire.
- Nessun server da dimensionare correttamente, monitorare o scalare orizzontalmente.
- Nessun rischio di provisioning eccessivo che faccia aumentare i costi.
- Nessun rischio per le prestazioni a causa di un provisioning insufficiente.

All'interno di ogni servizio sono inoltre presenti caratteristiche specifiche che vanno a vantaggio del modello architetturale multi-tier.

Amazon API Gateway

Amazon API Gateway è un servizio completamente gestito per la definizione, distribuzione e manutenzione delle API. I client si integrano alle API tramite richieste HTTPS standard. La sua applicabilità a un'architettura multi-tier orientata ai servizi è evidente. Tuttavia, dispone di caratteristiche e qualità specifiche che ne fanno uno strumento potente per il livello logico.

Integrazione con AWS Lambda

Amazon API Gateway offre all'applicazione un modo semplice (richieste HTTPS) per sfruttare direttamente l'innovazione di AWS Lambda. API Gateway costituisce il ponte di collegamento tra il livello di presentazione e le funzioni che vengono scritte in AWS Lambda. Dopo avere definito il rapporto client/server con l'API, i contenuti della richiesta HTTPS del client sono passati alla funzione Lambda per l'esecuzione. Tali contenuti includono i metadata della richiesta, le intestazioni della richiesta e il corpo della richiesta.

Prestazioni stabili delle API in tutto il mondo

Ogni distribuzione di Amazon API Gateway include anche una distribuzione di Amazon CloudFront[4]. Amazon CloudFront è un servizio Web di distribuzione di contenuti che utilizza la rete globale Amazon di edge location come punti di connessione per i client che si integrano con le API. In questo modo si contribuisce a ridurre la latenza del tempo di risposta totale dell'API. Grazie alle varie edge location in tutto il mondo, Amazon CloudFront offre anche le funzionalità per contrastare scenari di attacchi DDoS (Distributed Denial of Service). Per maggiori informazioni, consultare il whitepaper AWS Best Practices for Combatting DDoS Attacks[5] (Best practice AWS per contrastare gli attacchi DDoS).

Si possono migliorare le performance di richieste API specifiche utilizzando Amazon API Gateway per eseguire lo storage delle risposte in una cache in memoria facoltativa. In questo modo non solo si ottengono vantaggi in termini di prestazioni per richieste API ripetute, ma si riducono anche le esecuzioni backend, con conseguente possibilità di ridurre il costo totale.

Incoraggiare l'innovazione

L'attività di sviluppo richiesta per realizzare una nuova applicazione rappresenta un investimento. Occorre giustificarlo per poter avviare il progetto. Con la riduzione dell'investimento richiesto per le attività e il tempo di sviluppo, si è liberi di sperimentare maggiormente e di innovare con maggiore libertà.

Per molte applicazioni multi-tier basate sui servizi Web, il livello di presentazione viene facilmente frammentato tra vari utenti (dispositivi mobili separati, browser Web, ecc). Inoltre, questi utenti spesso non sono vincolati a livello geografico. Un livello logico disassociato, tuttavia, non è fisicamente frammentato tra gli utenti. Tutti gli utenti dipendono dalla stessa infrastruttura che esegue il livello logico, il che aumenta l'importanza dell'infrastruttura. Quando si inizia a implementare il livello logico, la tentazione di andare al risparmio ("non serve adottare parametri al momento del lancio iniziale;" "l'utilizzo iniziale sarà basso, penseremo al dimensionamento in un secondo momento;" ecc.) viene spesso giustificata come un meccanismo per distribuire più velocemente una nuova applicazione. Ciò può portare al debito tecnico e al rischio operativo quando si devono distribuire tali modifiche a un'applicazione già in produzione. Amazon API Gateway *permette* di ridurre i costi ed effettuare la distribuzione più rapidamente perché il servizio stesso ha già implementato le modifiche.

La vita complessiva di un'applicazione potrebbe essere sconosciuta oppure si potrebbe sapere già che sarà breve. La creazione di un business case per una nuova applicazione multi-tier può risultare difficoltosa per questi motivi. La situazione si può semplificare quando il punto di partenza include già le caratteristiche gestite offerte da Amazon API Gateway e i costi infrastrutturali vengono sostenuti solo dopo che le API iniziano a ricevere richieste. Per ulteriori informazioni, consultare Amazon API Gateway Pricing (Prezzi Amazon API Gateway)[6].

Creazione rapida di iterazioni e mantenimento dell'agilità

Con le nuove applicazioni, la base utenti può comunque essere scarsamente definita (dimensioni, modelli di utilizzo, ecc.). Il livello logico deve restare agile mentre la base utenti prende forma. L'applicazione e l'attività dovrebbero essere in grado di cambiare e adattarsi al mutamento delle aspettative degli early adopter. Amazon API Gateway riduce il numero di cicli di sviluppo richiesti per portare un'API dalla creazione allo sviluppo. Amazon API Gateway offre la possibilità di creare integrazioni fittizie[7] che permettono di generare direttamente da API Gateway risposte API sulla base delle quali le applicazioni client si possono sviluppare mentre, in parallelo, viene sviluppata la logica backend completa. Questo vantaggio si può sfruttare non solo alla prima distribuzione di un'API, ma anche dopo che l'azienda ha deciso che l'applicazione (e l'API esistente) deve evolversi rapidamente in risposta alle esigenze degli utenti. API Gateway e AWS Lambda consente la funzione Versioni multiple, cosicché la funzionalità esistente e le dipendenze client possono continuare indisturbate mentre la nuova funzionalità viene rilasciata come versione separata dell'API/funzione.

Sicurezza

L'implementazione del livello logico di un'applicazione Web pubblica a tre livelli come servizio Web solleva immediatamente la questione della sicurezza. L'applicazione deve garantire che solo i client autorizzati abbiano accesso al livello logico (esposto in rete). Amazon API Gateway affronta la questione della sicurezza in modo tale da garantire la protezione del backend. Per il controllo accessi, non si può fare affidamento sul fatto di fornire alle applicazioni client stringhe chiave statiche di API; tali stringhe possono infatti essere estratte dai client e utilizzate altrove. Si possono utilizzare molti dei diversi modi in cui Amazon API Gateway contribuisce a garantire il livello logico:

- Tutte le richieste alle API possono essere effettuate tramite HTTPS per abilitare la crittografia in transito.
- Le funzioni di AWS Lambda posso limitare l'accesso, in modo che esista un rapporto di fiducia solo tra una particolare API di Amazon API Gateway e una particolare funzione AWS Lambda. L'unico modo per invocare tale funzione Lambda sarà mediante l'API attraverso la quale si è scelto di esporla.
- Amazon API Gateway permette di generare SDK client che si integrano con le API. L'SDK gestisce, inoltre, la firma delle richieste quando le API richiedono l'autenticazione. Le credenziali API utilizzate dal lato client per l'autenticazione sono passate direttamente alla funzione AWS Lambda, dove vi può essere, all'occorrenza, l'ulteriore autenticazione nel codice di cui si è titolari e autori.
- A ogni combinazione di risorse/metodi creata nell'ambito dell'API viene concesso un proprio Amazon Resource Name (ARN) specifico a cui è possibile fare riferimento nelle policy AWS Identity and Access Management (IAM)[8].
  - Ciò significa che le API sono trattate come "cittadini di prima classe" insieme alle altre API di cui AWS è titolare. Le policy IAM possono essere a granularità fine e possono fare riferimento a risorse/metodi specifici di un'API creata con Amazon API Gateway.
  - L'accesso API è applicato dalle policy IAM create al di fuori del contesto del codice dell'applicazione. Ciò significa che non occorre scrivere un codice per essere consapevoli di tali livelli di accesso o per applicarli. Il codice non può contenere bug né essere utilizzato se non esiste.

— I client che forniscono l'autorizzazione con AWS Signature version 4 (SigV4)[9] e le policy IAM per l'accesso API permettono a quelle stesse credenziali di limitare o consentire l'accesso ad altri servizi e risorse AWS, in base alle esigenze (ad esempio, ai bucket di Amazon S3 o alle tabelle di Amazon DynamoDB).

AWS Lambda

AWS Lambda permette, in sostanza, l'attivazione del codice arbitrario scritto in qualsiasi linguaggio supportato (Node, basato su JVM e Python da novembre 2015) in risposta a un evento. Tale evento può essere uno di vari trigger programmatici che AWS mette a disposizione ed è definito un'**origine evento** (consultare le origini evento attualmente supportate qui[10]). Molti casi d'uso conosciuti per AWS Lambda ruotano intorno a flussi di lavoro di elaborazione dati orientati agli eventi, come l'elaborazione dei file archiviati in Amazon Simple Storage Service (Amazon S3)[11] o a record di dati di streaming di Amazon Kinesis[12].

Se utilizzata in combinazione con Amazon API Gateway, una funzione AWS Lambda può esistere nel contesto di un tipico servizio Web ed è attivabile direttamente con una richiesta HTTPS. Amazon API Gateway funge da porta d'ingresso del livello logico, ma a questo punto è necessario eseguire la logica alla base di tali API. Ed è qui che entra in gioco AWS Lambda

La logica di business confluisce qui

AWS Lambda permette di scrivere funzioni del codice, denominate **gestori**, che vengono eseguite quando sono attivate da un evento. È possibile, ad esempio, scrivere un gestore che si attivi quando l'API riceve una richiesta HTTPS. Lambda permette di creare gestori modulari al livello di granularità scelto (uno per API o uno per metodo API) che possono essere aggiornati, invocati e modificati in modo indipendente. Il gestore è quindi libero di accedere a tutte le altre dipendenze disponibili (come altre funzioni caricate insieme al codice, librerie, codici binari nativi o persino servizi Web esterni). Lambda permette di riunire in un pacchetto tutte le dipendenze richieste nella definizione della funzione durante la creazione. Quando si crea una funzione, si specifica il metodo all'interno del pacchetto di distribuzione che dovrà fungere da gestore delle richieste. È possibile utilizzare lo stesso pacchetto di distribuzione per le definizioni di più funzioni Lambda, dove ogni funzione Lambda può avere un gestore univoco all'interno dello stesso pacchetto di distribuzione. Nel modello di architettura multi-tier serverless, ciascuna API creata in Amazon API Gateway si integrerà con una funzione Lambda (e con il gestore al suo interno) che esegue la logica di business richiesta.

Integrazione con Amazon VPC

AWS Lambda, il nucleo del livello logico, sarà il componente che si integra direttamente con il livello dati. Dato che il livello dati contiene spesso informazioni sensibili dell'azienda o degli utenti, dovrebbe avere un grado di sicurezza molto elevato. Per i servizi AWS con i quali è possibile integrarsi da una funzione Lambda, si può gestire il controllo accessi mediante policy IAM. Tali servizi includono Amazon S3, Amazon DynamoDB, Amazon Kinesis, Amazon Simple Queue Service (Amazon SQS), Amazon Simple Notification Service (Amazon SNS), ulteriori funzioni AWS Lambda e molto altro. Tuttavia, potrebbe essere presente un componente che gestisce il proprio controllo accessi, come un database relazionale. Con componenti come questi si ottiene una sicurezza maggiore distribuendoli all'interno di un ambiente di rete privato, come Amazon Virtual Private Cloud (Amazon VPC)[13].

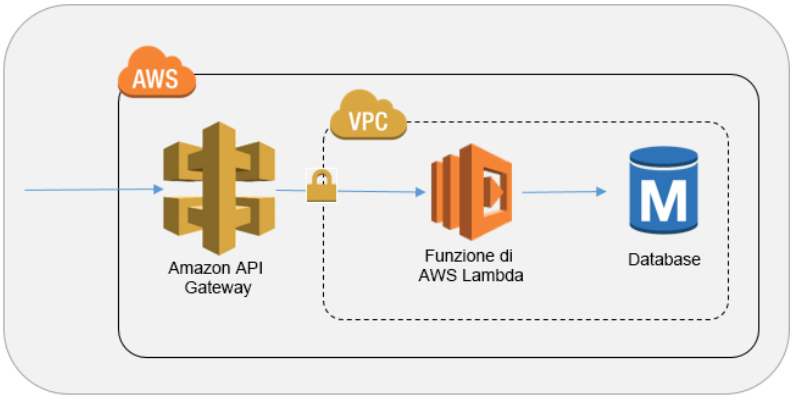

Figura 2. Modello architetturale con l'utilizzo di un VPC

L'impiego di un VPC fa sì che i database e gli altri supporti di storage da cui la logica di business dipende possono essere resi inaccessibili in Internet. Il VPC garantisce inoltre che l'*unico* modo per interagire con i dati da Internet sia attraverso le API che sono state definite e le funzioni del codice Lambda che sono state scritte.

Sicurezza

L'esecuzione di una funzione Lambda deve essere attivata da un evento o servizio che è stato autorizzato in tal senso tramite una policy IAM. È possibile creare una funzione Lambda che *non possa* essere eseguita a meno che non sia invocata da una richiesta API Gateway precedentemente definita. Il codice effettuerà l'elaborazione solo nel quadro del caso d'uso valido, definito dall'API che è stata creata.

Ogni funzione Lambda assume un ruolo IAM, una capacità che deve essere concessa tramite un rapporto di fiducia IAM. Il ruolo IAM definisce gli altri servizi/risorse AWS con cui la funzione Lambda sarà in grado di interagire (come una tabella di Amazon DynamoDB o un bucket di Amazon S3). I servizi a cui la funzione ha accesso saranno definiti e controllati dall'esterno della funzione stessa. Si tratta di una differenza sottile, ma incisiva. Permette al codice che è stato scritto di archiviare o recuperare liberamente le credenziali AWS: in questo modo non occorre effettuare l'hard coding delle chiavi API né scrivere un codice per recuperarle e archiviarle in memoria. L'attivazione della funzione Lambda che richiama i servizi a cui ha accesso, come definiti dal suo ruolo IAM, è gestita dal servizio stesso.

Il livello dati

Utilizzando AWS Lambda come livello logico, si dispone di una vasta gamma di opzioni di storage dei dati per il livello dati. Tali opzioni rientrano in due categorie generali: data store Amazon VPC in hosting e data store attivati da IAM. AWS Lambda ha la capacità di integrarsi in sicurezza con entrambe queste opzioni.

Data store Amazon VPC in hosting

L'integrazione di AWS Lambda con Amazon VPC consente alle funzioni di integrarsi con una varietà di tecnologie di storage dei dati in modo privato e sicuro.

- Amazon RDS[14]

  Si può utilizzare uno qualunque dei motori che Amazon Relational Database Service (Amazon RDS) mette a disposizione. È possibile connettersi ad Amazon RDS direttamente dal codice scritto in Lambda esattamente come si farebbe al di fuori di Lambda, ma con il vantaggio di un'integrazione semplice con AWS Key Management Service (AWS KMS) per la crittografia delle credenziali del database.

- Amazon ElastiCache[15]

  Le funzioni Lambda sono integrate a una cache in memoria gestita per potenziare le prestazioni dell'applicazione.

- Amazon RedShift[16]

  Si possono realizzare funzioni in grado di elaborare in sicurezza query per un data warehouse aziendale allo scopo sviluppare report e pannelli di controllo o di recuperare i risultati di query ad hoc.

- Servizio Web privato ospitato da Amazon Elastic Compute Cloud (Amazon EC2)[17]

  Si potrebbero eseguire privatamente le applicazioni esistenti come un servizio Web in un VPC ed effettuare le richieste HTTP nella rete VPC privata dal punto di vista logico da una funzione Lambda.

Store data attivati da IAM

Dato che AWS Lambda è integrato con IAM, può utilizzare quest'ultimo per garantire l'integrazione con qualsiasi servizio AWS utilizzabile direttamente tramite le API AWS.

- Amazon DynamoDB[18]

  Amazon DynamoDB è il database NoSQL AWS con scalabilità infinita. Amazon DynamoDB è un'opzione da tenere in considerazione quando si vogliono recuperare record di dati (400 KB o inferiori al momento della pubblicazione di questo whitepaper) con prestazioni a una cifra in termini di millisecondi, indipendentemente dalle dimensioni. Con il controllo degli accessi a granularità fine Amazon DynamoDB, le funzioni Lambda possono seguire la best practice del privilegio minimo quando eseguono query relative a dati specifici in DynamoDB

- Amazon S3[19]

  Amazon Simple Storage Service (Amazon S3) offre lo storage degli oggetti a livello di Internet. Amazon S3 è pensato per la durabilità del 99,999999999% degli oggetti, pertanto è opportuno prenderlo in considerazione quando all'applicazione occorre uno storage conveniente ed estremamente durevole. Amazon S3, inoltre, è pensato per una durabilità degli oggetti fino al 99,99% in un dato anno, pertanto si può prendere in considerazione il suo utilizzo quando l'applicazione richiede uno storage estremamente disponibile. Gli oggetti archiviati in Amazon S3 (file, immagini, log, dati binari) sono direttamente accessibili tramite HTTP. Le funzioni Lambda sono in grado di comunicare in modo sicuro con Amazon S3 tramite endpoint privati virtuali e i dati all'interno di S3 possono essere limitati alla sola policy IAM associata alla funzione Lambda.

- Amazon Elasticsearch Service[20]

  Amazon Elasticsearch Service (Amazon ES) è una versione gestita del popolare motore di ricerca e analisi Elasticsearch. Amazon ES offre il provisioning gestito di cluster, rilevamento degli errori e sostituzione dei nodi. È inoltre possibile limitare l'accesso all'API Amazon ES utilizzando le policy IAM.

Il livello di presentazione

Amazon API Gateway mette a disposizione una gamma di possibilità per il livello di presentazione. Un'API HTTPS accessibile da Internet può essere utilizzata da qualunque client in grado di effettuare comunicazioni HTTPS. L'elenco seguente contiene alcuni esempi comuni di come sia utilizzabile per il livello di presentazione di un'applicazione:

- App mobile: oltre all'integrazione con la logica di business personalizzata mediante Amazon API Gateway e AWS Lambda, si potrebbe utilizzare Amazon Cognito[21] come meccanismo di creazione e gestione delle identità degli utenti.

- Contenuti statici dei siti Web (come i file in hosting in Amazon S3): è possibile abilitare le API di Amazon API Gateway affinché siano conformi a CORS (Cross-Origin Resource Sharing, condivisione delle risorse multiorigine). In questo modo i browser Web possono invocare direttamente le API dall'interno delle pagine Web statiche.
- Qualsiasi altro dispositivo client abilitato per HTTPS: molti dispositivi connessi sono in grado di comunicare tramite HTTPS. Non vi è nulla di univoco o proprietario nel modo in cui i client comunicano con le API create con Amazon API Gateway: si tratta di semplice HTTPS. Non occorrono software o licenze speciali per il client.

Campioni di modelli architetturali

È possibile implementare i seguenti modelli architetturali diffusi utilizzando Amazon API Gateway e AWS Lambda come il collante che tiene insieme il livello logico. In ogni esempio, utilizzeremo solo i servizi AWS che non richiedono la gestione, da parte dei clienti, della propria infrastruttura.

Back end per dispositivi mobili

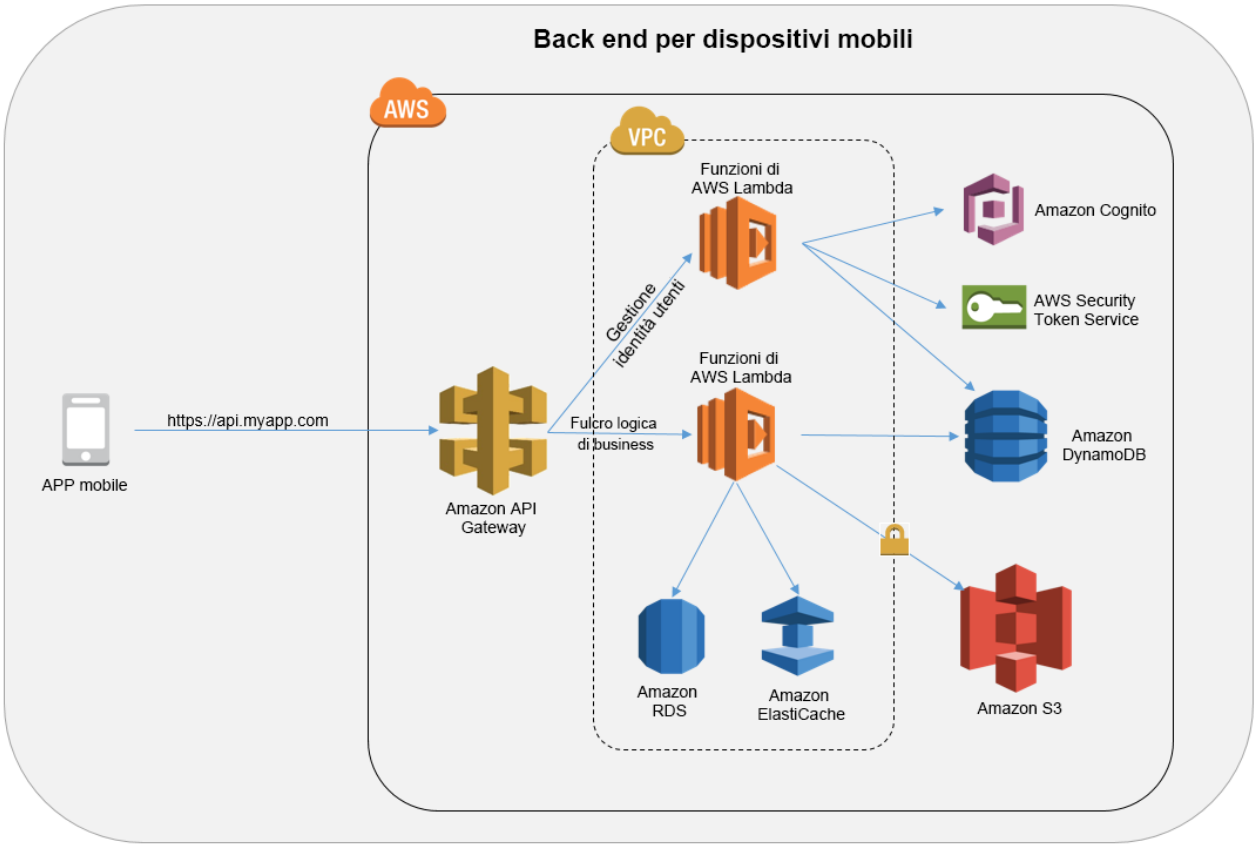

Figura 3. Modello architetturale di back end per i dispositivi mobili

- **Livello di presentazione:** un'applicazione mobile in esecuzione sullo smartphone di ogni utente.
- **Livello logico:** Amazon API Gateway e AWS Lambda. Il livello logico viene distribuito a livello globale dalla distribuzione Amazon CloudFront creata nell'ambito di ciascuna API di Amazon API Gateway. Una gamma di funzioni Lambda può essere specifica per la gestione delle identità di utenti/dispositivi e dell'autenticazione e gestita da Amazon Cognito, che permette l'integrazione con IAM per credenziali temporanee di accesso degli utenti e come diffuso provider di identità di terze parti. Altre funzioni Lambda possono definire il fulcro della logica di business per il back end per dispositivi mobili.

- **Livello dati:** i vari servizi di storage dei dati possono essere utilizzati in base alle esigenze (le relative opzioni sono trattate nelle precedenti sezioni di questo documento).

Sito Web in hosting su Amazon S3

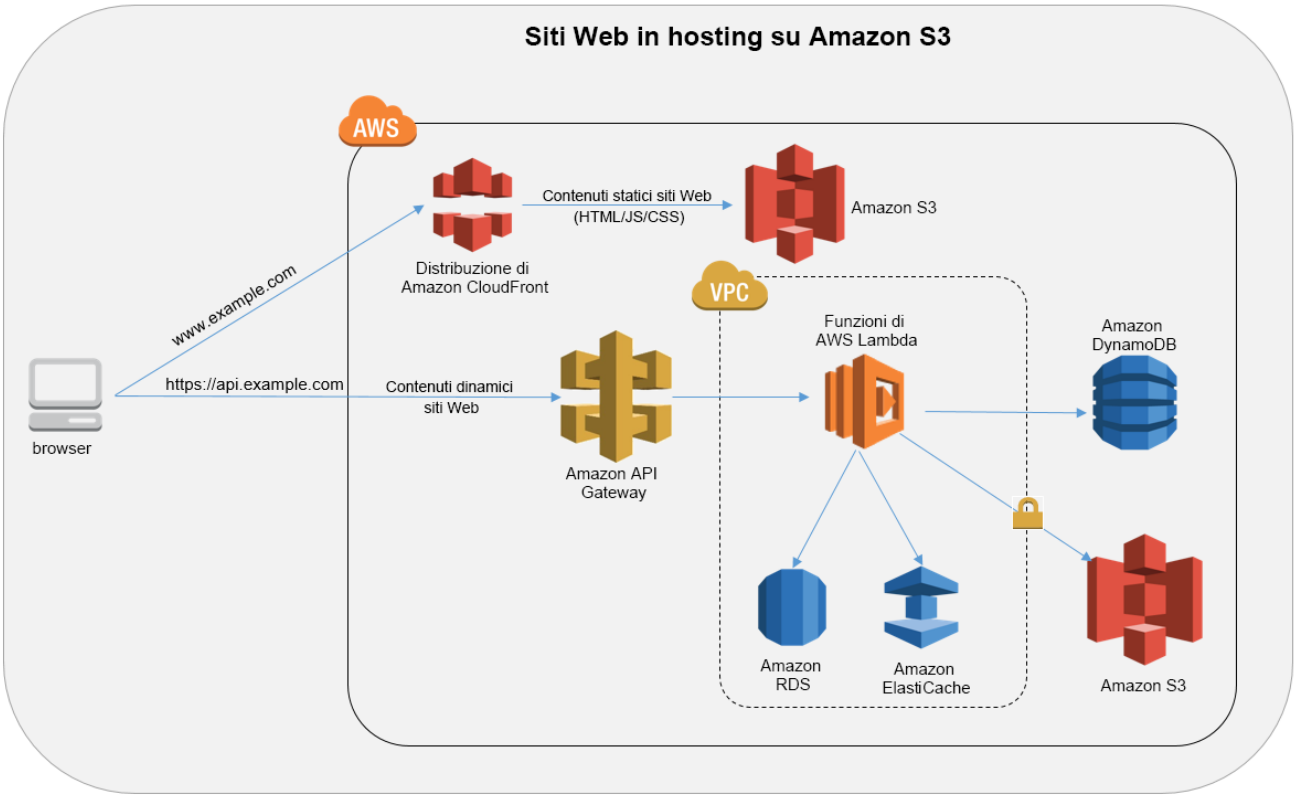

Figura 4. Modello architetturale per un sito Web statico in hosting su Amazon S3

- **Livello di presentazione:** i contenuti del sito Web statico in hosting su Amazon S3, distribuiti da Amazon CloudFront. I contenuti del sito Web statico in hosting su Amazon S3 costituiscono un'alternativa economica all'hosting dei contenuti su un'infrastruttura basata su server. Tuttavia, se il sito Web è dotato di numerose caratteristiche, il contenuto statico deve spesso integrarsi con un back end dinamico.
- **Livello logico:** Amazon API Gateway e AWS Lambda. I contenuti Web statici in hosting su Amazon S3 si possono integrare direttamente con Amazon API Gateway, che può essere conforme a CORS.

- **Livello dati:** i vari servizi di storage dei dati sono utilizzabili in base alle esigenze. Queste opzioni sono trattate nelle sezioni precedenti del documento.

Ambiente di microservizi

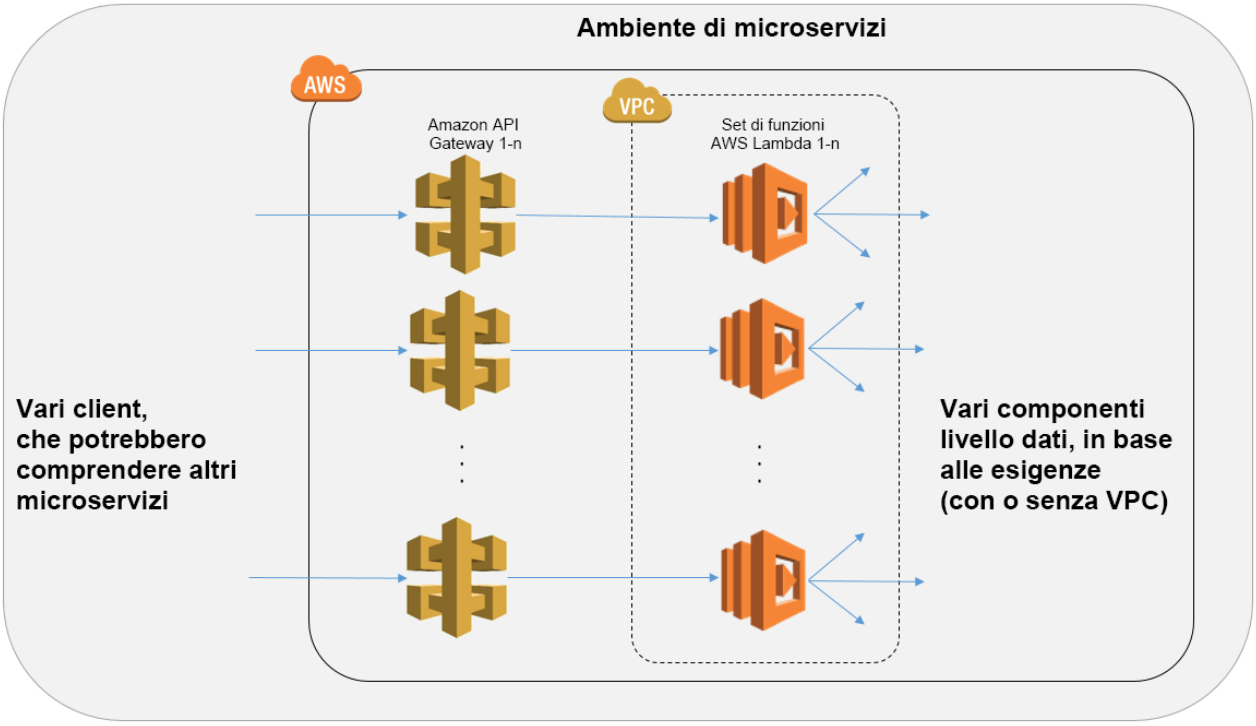

Figura 5. Modello architetturale per un ambiente di microservizi

Il modello architetturale per i **microservizi** non è legato alla tipica architettura a tre livelli che abbiamo trattato in questo whitepaper. In un'architettura per I microservizi, è presente una dissociazione massiccia dei componenti software, pertanto i vantaggi dell'architettura multi-tier sono massimizzati ovunque. Un'API creata con Amazon API Gateway e le funzioni eseguite da AWS Lambda sono tutto quello che occorre per realizzare un microservizio. Il team addetto può utilizzare liberamente questi servizi per disassociare e frammentare l'ambiente al livello di granularità desiderato.

Un ambiente di microservizi può introdurre, in genere, le seguenti difficoltà: costi sostenuti ripetutamente per la creazione di ogni nuovo microservizio, problemi con l'ottimizzazione della densità/utilizzo del server, complessità di esecuzione simultanea di più versioni di vari microservizi e proliferazione dei requisiti dal codice lato client per integrare molti servizi separati.

Tuttavia, quando si creano microservizi con l'ausilio del modello AWS serverless, questi problemi diventano più semplici da risolvere e, in alcuni casi, scompaiono semplicemente. Il modello dei microservizi AWS riduce la barriera per la creazione di ogni successivo microservizio (Amazon API Gateway permette anche di clonare le API esistenti). L'ottimizzazione dell'utilizzo del server non è più un elemento pertinente con questo modello. Sia API Gateway sia Lambda attivano funzioni semplici per versioni multiple. Amazon API Gateway, infine, offre SDK client generati sistematicamente in numerosi linguaggi diffusi per ridurre i costi di integrazione.

Conclusioni

Il modello di architettura multi-tier favorisce la best practice relativa alla creazione di componenti di applicazioni semplici da gestire, disassociati e scalabili. Quando si crea un livello logico dove l'integrazione avviene mediante Amazon API Gateway e il calcolo è effettuato in AWS Lambda, si è sulla strada giusta verso la realizzazione di tali obiettivi e la simultanea riduzione del lavoro richiesto per raggiungerli. Insieme, questi servizi offrono un front end di API HTTPS per i client e un ambiente sicuro nel VPC per l'esecuzione della logica di business. In questo modo è possibile utilizzare molti scenari diffusi in cui utilizzare i servizi gestiti invece di gestire da soli la tipica infrastruttura basata su server.

Collaboratori

Le persone e le organizzazioni indicate di seguito hanno collaborato alla stesura di questo documento:

Andrew Baird, AWS Solutions Architect

Stefano Buliani, Senior Product Manager, Tech, AWS Mobile

Vyom Nagrani, Senior Product Manager, AWS Mobile

Ajay Nair, Senior Product Manager, AWS Mobile

Note

[1] http://aws.amazon.com/api-gateway/

[2] http://aws.amazon.com/lambda/

[3] https://aws.amazon.com/vpc/

[4] https://aws.amazon.com/cloudfront/

[5] https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf

[6] https://aws.amazon.com/api-gateway/pricing/

[7] http://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-mock-integration.html

[8] http://aws.amazon.com/iam/

[9] http://docs.aws.amazon.com/general/latest/gr/signature-version-4.html

[10] http://docs.aws.amazon.com/lambda/latest/dg/intro-core-components.html#intro-core-components-event-sources

[11] https://aws.amazon.com/s3/

[12] https://aws.amazon.com/kinesis/

[13] https://aws.amazon.com/vpc/

[14] https://aws.amazon.com/rds/

[15] https://aws.amazon.com/elasticache/

[16] https://aws.amazon.com/redshift/

[17] https://aws.amazon.com/ec2/

[18] https://aws.amazon.com/dynamodb/

[19] https://aws.amazon.com/s3/storage-classes/

[20] https://aws.amazon.com/elasticsearch-service/

[21] https://aws.amazon.com/cognito/