Ottimizzazione dell'economia delle aziende con architetture serverless

Settembre 2017

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.

Sommario

Sintesi

Scopo di questo whitepaper è aiutare i Chief Information Officer (CIO), i Chief Technology Officer (CTO) e i senior architect a comprendere le architetture serverless e il loro impatto su time to market, agilità del team ed economia IT. Eliminando in fase di progettazione i server inattivi e sottoutilizzati e semplificando enormemente i progetti software basati su cloud, gli approcci serverless stanno rapidamente cambiando il panorama IT.

Questo whitepaper prende in esame la base degli approcci serverless e il portafoglio serverless AWS. I numerosi casi di studio che presenta illustrano come le società che hanno adottato approcci serverless stiano già guadagnando in agilità e ottenendo benefici economici significativi. Il documento spiega come organizzazioni di qualsiasi dimensione possano usare le architetture serverless per realizzare sistemi reattivi, basati sugli eventi, e fornire rapidamente microservizi nativi per il cloud a costi molto inferiori a quelli tradizionali.

Introduzione

Molte società stanno già ottenendo benefici dall'esecuzione di applicazioni nel cloud pubblico, fra cui un risparmio sui costi grazie alla fatturazione in base al consumo e una migliore agilità in virtù dell'uso di risorse IT on-demand. Da numerosi studi condotti su diversi tipi di applicazioni e industrie è emerso che la migrazione delle architetture delle applicazioni esistenti sul cloud riduce il costo totale di proprietà (TCO) e migliora il time to market.[1]

Rispetto alle soluzioni on-premise e al cloud privato, il cloud pubblico semplifica notevolmente la creazione, la distribuzione e la gestione di flotte di server e delle applicazioni eseguite su di essi. Tuttavia, per sfruttare il cloud pubblico oggi le società dispongono di altre opzioni, a parte le classiche architetture basate su server o VM. Sebbene le società che scelgono il cloud non debbano più acquistare e mantenere il proprio hardware, *qualsiasi* architettura basata su server richiede che prevedano scalabilità e affidabilità. Inoltre, devono affrontare le sfide della creazione di patch e della loro distribuzione su queste flotte di server man mano che le loro applicazioni evolvono. Devono anche scalare le proprie flotte di server per tener conto dei carichi di picco e tentare poi di ridurle quando e se possibile per abbassare i costi. Il tutto, continuando a proteggere l'esperienza degli utenti finali e l'integrità dei sistemi interni. I server inattivi e sottoutilizzati si rivelano costosi e fonte di sprechi. Gli analisti stimano che ben l'85% dei server di fatto sia sottoutilizzato.[2]

Servizi di elaborazione serverless come AWS Lambda sono studiati per affrontare queste sfide offrendo alle società un modo diverso di avvicinarsi alla progettazione delle applicazioni, con costi intrinsecamente inferiori e time to market più rapido. *AWS Lambda elimina la complessità delle interazioni con i server a tutti i livelli dello stack tecnologico e introduce un modello di fatturazione in base alla richiesta in cui scompaiono i costi derivanti dal non utilizzo della capacità di elaborazione*. Inoltre, le funzioni Lambda consentono alle organizzazioni di adottare facilmente le architetture dei microservizi. L'eliminazione dell'infrastruttura e il passaggio a un modello Lambda offrono un duplice vantaggio economico:

- Problemi come quello dei server inattivi cessano semplicemente di esistere, così come le loro conseguenze economiche. Un servizio di elaborazione serverless come AWS Lambda non è mai "freddo" perché i costi si accumulano solo quando vengono eseguiti lavori utili, con una granularità di fatturazione a livello di millisecondi.

- La gestione della flotta (inclusi implementazione di patch di sicurezza, distribuzioni e monitoraggio dei server) non è più necessaria. Ciò significa che non occorre mantenere i relativi tool e processi e le cosiddette rotazioni su chiamata richieste per supportare un'operatività 24x7 della flotta di server. L'uso di Lambda per creare microservizi aiuta le organizzazioni a essere più agili. Senza il fardello della gestione dei server, le società possono indirizzare le scarne risorse IT di cui dispongono a ciò che è per loro importante: il loro business.

Grazie alla forte riduzione dei costi delle infrastrutture, a team più agili e mirati e a un time to market più rapido, le società che hanno già adottato gli approcci serverless stanno acquisendo un vantaggio chiave sui concorrenti.

Cosa sono le applicazioni serverless

Il vantaggio dell'approccio serverless menzionato sopra è allettante, ma quali sono le considerazioni per metterlo in pratica? Cosa distingue un'app serverless dal suo corrispettivo tradizionale basato su server?

Le app serverless sono architettate in modo che gli sviluppatori possano concentrare l'attenzione sulla loro competenza principale: scrivere la logica aziendale vera e propria. Molte delle componenti del boilerplate di un'app, come i server Web, e tutto l'undifferentiated heavy lifting (UHL), come il software per gestire l'affidabilità e la scalabilità, vengono totalmente sottratte allo sviluppatore. Ciò che rimane è un approccio funzionale nitido in cui la logica aziendale si attiva solo quando è necessario: invio di un messaggio col cellulare da parte dell'utente, caricamento di un'immagine nel cloud, arrivo in streaming delle registrazioni, e così via. Un approccio asincrono, basato su eventi, alla progettazione delle applicazioni non è richiesto ma è molto comune in applicazioni serverless perché combacia perfettamente con il concetto di codice che viene eseguito (con i costi che ne conseguono) solo quando c'è del lavoro da fare.

Un'applicazione serverless viene eseguita nel cloud pubblico, in un servizio come AWS Lambda, che si occupa di ricevere gli eventi o le invocazioni del client e poi crea un'istanza ed esegue il codice. Questo modello offre diversi vantaggi rispetto al progetto di applicazioni tradizionali basate su server:

- Non vi è alcuna necessità di provvedere a, distribuire, aggiornare, monitorare o gestire in altro modo i server. Tutto l'hardware e il software effettivi del server sono gestiti dal fornitore di servizi cloud.
- L'applicazione si ridimensiona automaticamente in base all'uso effettivo. Questo è intrinsecamente diverso dalle applicazioni tradizionali, che richiedono una flotta di ricevitori e una gestione esplicita della capacità per ridimensionarsi in base ai picchi di carico.
- Oltre alla scalabilità, sono integrate anche la disponibilità e la tolleranza ai guasti. Per beneficiare di queste funzionalità, non è necessaria alcuna codifica, configurazione o gestione.
- Non sono previsti costi per la capacità inutilizzata. Non è necessario (e di fatto non è possibile) predisporre in anticipo la capacità o fornirne in eccedenza. La fatturazione avviene in base alla richiesta e dipende dalla durata necessaria per l'esecuzione del codice.

Casi d'uso di applicazioni serverless

Il modello di applicazione serverless è generico e si applica quasi a ogni tipo di applicazione, da quella Web di una startup alla piattaforma di analisi di trading di una società Fortune 100. Ecco alcuni esempi:

- **App Web e siti Web** – L'eliminazione dei server permette di creare app Web che non costano quasi nulla quando non c'è traffico, mentre scalano simultaneamente per gestire picchi di carico anche imprevisti.
- **Backend mobili** – I backend mobili serverless offrono agli sviluppatori la possibilità di dedicarsi allo sviluppo dei client per creare facilmente backend sicuri, ad alta disponibilità e perfettamente scalati, senza dover diventare degli esperti in progettazione di sistemi distribuiti.
- **Elaborazione di log e media** – Gli approcci serverless offrono un parallelismo naturale, semplificando l'elaborazione di carichi di lavoro pesanti dal punto di vista dei calcoli, senza la complessità di dover costruire sistemi multithread o scalare manualmente flotte di elaborazione.
- **Automazione IT** – Le funzioni serverless possono essere associate ad allarmi e monitor per assicurare la personalizzazione, quando richiesta. L'implementazione di cron job e di altri requisiti di infrastruttura IT viene notevolmente semplificata rimuovendo la necessità di detenere e mantenere dei server appositi, specie quando questi job e requisiti sono sporadici o di natura variabile.

- **Backend IoT** – La capacità di portare qualsiasi codice, incluse le librerie native, semplifica il processo di creazione di sistemi basati su cloud in grado di implementare algoritmi specifici per il dispositivo.
- **Chatbot (inclusi assistenti vocali) e altri sistemi basati su webhook** – Gli approcci serverless sono ideali per qualsiasi sistema basato su webhook, come i chatbot. La loro capacità di eseguire azioni (ad esempio un codice) solo quando richiesto (ad esempio quando un utente richiede informazioni da un chatbot) ne fa un approccio semplice e generalmente più economico per queste architetture. Ad esempio, la maggior parte delle Alexa Skills per Amazon Echo sono implementate utilizzando AWS Lambda.
- **Clickstream e altri processi di dati in streaming quasi in tempo reale** – Le soluzioni serverless offrono la flessibilità di scalare verso l'alto e verso il basso in base al flusso di dati, in funzione dei requisiti di throughput, senza la complessità di dover costruire un sistema di elaborazione scalabile per ogni applicazione. Abbinata a una tecnologia come Amazon Kinesis, AWS Lambda può offrire un'elaborazione ad alta velocità dei record per l'analisi di clickstream, trigger di dati NoSQL, informazioni di trading, e altro ancora.

Oltre ai casi d'uso più diffusi presi in esame in precedenza, le società stanno applicando approcci serverless anche alle problematiche dei seguenti settori:

- big data, computazione MapReduce, transcodifica video ad alta velocità, analisi di trading e simulazioni Monte Carlo che richiedono calcoli massivi per domande di prestiti. Gli sviluppatori hanno scoperto che è molto più facile parallelizzare con un approccio serverless,[3] specie se l'attivazione è determinata dagli eventi. Questo li sta portando ad applicare in misura crescente le tecniche serverless a un'ampia gamma di problemi di big data senza dover gestire l'infrastruttura.
- Bassa latenza, elaborazione personalizzata per applicazioni Web e asset forniti tramite reti di erogazione di contenuti. Spostando la gestione degli eventi serverless nell'edge di Internet, gli sviluppatori possono sfruttare la minore latenza e la capacità di personalizzare facilmente recuperi e acquisizioni di contenuti. Ciò permette un nuovo spettro di casi d'uso con una latenza ottimizzata in base alla località in cui si trova il client.

- Dispositivi connessi, che offrono capacità serverless come le funzioni AWS Lambda per l'esecuzione all'interno di dispositivi Internet of Things (IoT) commerciali, residenziali e portatili. Soluzioni serverless come le unzioni Lambda offrono un'astrazione naturale dall'hardware fisico sottostante (e anche virtuale), permettendo una loro transizione più facile dal data center all'edge, e da un'architettura hardware a un'altra, senza sconvolgere il modello di programmazione.
- Logica personalizzata e gestione dei dati in appliance on-premise come AWS Snowball Edge. Disaccoppiando la logica aziendale dai dettagli dell'ambiente di esecuzione, le applicazioni serverless possono facilmente funzionare in una molteplicità di ambienti diversi, anche in un'appliance.

In genere, le applicazioni serverless sono costruite usando un'*architettura di microservizi* in cui un'applicazione viene separata in componenti indipendenti che eseguono lavori discreti. Questi componenti, costituiti da funzioni Lambda individuali e API, code messaggi, database e altro ancora, possono essere distribuiti, testati e scalati in modo indipendente. Di fatto, dato il loro modello basato su funzioni, le applicazioni serverless sono un abbinamento naturale per i microservizi. Evitando progetti e architetture monolitiche, le organizzazioni possono diventare più agili perché gli sviluppatori possono distribuire incrementalmente e sostituire o aggiornare singoli componenti come il livello di database, se necessario.

In molti casi, basta semplicemente isolare la logica aziendale di un'applicazione per convertirla in un'app serverless. Servizi come AWS Lambda supportano linguaggi di programmazione diffusi e consentono di usare librerie personalizzate. Le attività di lunga durata sono espresse come flussi di lavoro composti da singole funzioni che operano in lassi di tempo ragionevoli. Questo permette al sistema di riavviare o parallelizzare, come richiesto, singole unità di calcolo.

L'approccio serverless è sempre appropriato?

Quasi tutte le applicazioni moderne possono essere modificate per poter essere eseguite con successo, e nella maggior parte dei casi in modo più economico e scalabile, su una piattaforma serverless. Vi sono tuttavia dei casi in cui quella serverless non è la scelta migliore:

- Quando l'obiettivo è esplicitamente quello di evitare di apportare qualsiasi modifica a un'applicazione.
- Quando è richiesto un controllo estremamente preciso dell'ambiente, ad esempio quando si specificano patch per un particolare sistema operativo o si accede a operazioni di networking di basso livello, per consentire una corretta esecuzione del codice.
- Quando un'applicazione on-premise non è stata migrata al cloud pubblico.

Valutazione della piattaforma serverless di un provider di cloud

Al momento di architettare un'applicazione serverless, le società e le organizzazioni devono considerare altri fattori oltre alla funzionalità di elaborazione serverless che esegue il codice dell'app. Le app serverless complete richiedono un'ampia gamma di servizi, tool e capacità che abbracciano storage, messaggistica, diagnostica e altro ancora. Un portafoglio serverless incompleto o frammentato di un provider di cloud può essere problematico per gli sviluppatori serverless. Se non riescono a creare con successo un codice a un livello omogeneo di astrazione, potrebbero infatti essere costretti a tornare ad architetture basate su server.

Una piattaforma serverless è costituita dalla serie di servizi che comprendono sia l'app serverless, come i componenti di elaborazione e di storage, che i tool richiesti per scrivere, creare, distribuire e diagnosticare app serverless. L'esecuzione di un'applicazione serverless in produzione richiede una piattaforma affidabile, flessibile e attendibile in grado di gestire le richieste sia di piccole startup che di multinazionali. La piattaforma deve scalare *tutti* gli elementi di un'applicazione e fornire un'affidabilità completa. Come avviene per le app tradizionali, aiutare gli sviluppatori a creare e fornire soluzioni di successo è una sfida dalle molteplici sfaccettature. Per soddisfare le esigenze di grandi aziende operanti in più settori, una piattaforma serverless dovrebbe offrire quanto segue:

Figura 1: Capacità di una piattaforma serverless

- Un *livello di logica cloud* scalabile, affidabile e dalle elevate prestazioni.
- *Fonti di dati ed eventi* di prime parti reattive e *connettività* semplice *a sistemi di terze parti*.
- *Librerie di integrazione* che consentono agli sviluppatori di iniziare facilmente a lavorare e aggiungere nuovi pattern alle soluzioni esistenti, in modo rapido e sicuro.
- Un dinamico *ecosistema di sviluppatori* che li aiuta a scoprire e ad applicare soluzioni in molteplici campi e per un'ampia serie di sistemi e casi d'uso di terze parti.
- Una raccolta di *framework per la modellazione di applicazioni* idonee allo scopo.
- Un'*orchestrazione* che offre la gestione di flusso di lavoro e stato.
- *Scala globale* e un ampio raggio d'azione che include la certificazione di programmi di garanzia.

- *Affidabilità integrata e prestazioni su scala*, senza dover effettuare il provisioning di capacità a qualsiasi livello di scala.
- *Sicurezza integrata* assieme a un *controllo* flessibile dell'accesso per risorse e servizi sia delle prime che delle terze parti.

Al cuore di qualsiasi piattaforma serverless vi è il *livello della logica cloud* responsabile dell'esecuzione di funzioni che rappresentano la logica aziendale. Dato che queste funzioni spesso vengono eseguite in risposta a degli eventi, un'*integrazione* semplice *con le fonti di eventi sia delle prime che delle terze parti* è fondamentale per poter esprimere in modo semplice le soluzioni e permettere che si adeguino automaticamente in risposta al variare dei carichi di lavoro. Ad esempio, potrebbe essere necessario eseguire funzioni serverless ogni volta che si crea un oggetto in uno store di oggetti o per ogni aggiornamento apportato a un database NoSQL serverless. Le architetture serverless eliminano tutto il codice di scalabilità e gestione normalmente richiesto per integrare questi sistemi, spostando sul provider di cloud quel fardello operativo.

Per sviluppare con successo su una piattaforma serverless, una società deve poter iniziare a lavorare senza problemi e trovare modelli già pronti per i casi d'uso comuni, che riguardino servizi di prime o di terze parti. Queste *librerie di integrazione* sono essenziali per trasmettere pattern di successo - come elaborazione di stream di record o implementazione di webhook - specie nel periodo in cui gli sviluppatori stanno migrando da architetture basate su server a quelle serverless. Un'esigenza strettamente correlata è quella di un *ecosistema ampio e differenziato* attorno alla piattaforma centrale. Un grande ecosistema dinamico aiuta gli sviluppatori a scoprire facilmente e usare soluzioni della comunità e favorisce l'apporto di nuove idee e approcci. Data la varietà delle toolchain in uso per la gestione del ciclo di vita delle applicazioni, occorre anche un ecosistema sano per assicurare che ogni linguaggio, IDE e tecnologia di costruzione aziendale abbia i runtime, i plugin e le soluzioni open source necessarie per integrare la costruzione e la distribuzione di app serverless negli approcci esistenti. È inoltre cruciale che le app serverless sfruttino gli investimenti esistenti, inclusa la conoscenza che gli sviluppatori hanno di framework come Express e Flask e di linguaggi di programmazione diffusi. Un ecosistema ampio fornisce un'accelerazione importante fra i domini e permette agli sviluppatori di ridefinire più velocemente il codice esistente in un'architettura serverless.

Framework per la modellazione di applicazioni, come la specifica aperta AWS Serverless Application Model (AWS SAM) permettono a uno sviluppatore di esprimere i componenti che costituiscono un'app serverless e rendono possibili gli strumenti e i flussi di lavoro richiesti per costruire, distribuire e monitorare quelle applicazioni. Un altro framework cruciale per il successo di una piattaforma serverless è *la gestione di orchestrazione e stato*. La natura largamente senza stato del serverless computing richiede un meccanismo complementare per abilitare flussi di lavoro di lunga durata. Le soluzioni di orchestrazione consentono agli sviluppatori di coordinare i molteplici componenti applicativi correlati tipici di un'applicazione serverless, pur permettendo che tali applicazioni siano composte da funzioni piccole e di breve durata. I servizi di orchestrazione semplificano inoltre la gestione degli errori e forniscono l'integrazione con i sistemi e i flussi di lavoro legacy, compresi quelli che durano più a lungo di quanto generalmente consentito dalle funzioni serverless.

Per supportare i clienti in tutto il mondo, comprese le multinazionali con una presenza globale, una piattaforma serverless deve offrire una *scala globale*, inclusi data center ed edge location situati in tutto il mondo. Le edge location sono cruciali per avvicinare il serverless computing a bassa latenza agli utenti inali. Dato che la responsabilità di fornire scalabilità e alta disponibilità delle app serverless ricade sulla piattaforma, più che sullo sviluppatore dell'applicazione, la sua *affidabilità* intrinseca gioca un ruolo determinante. Funzioni come numero di tentativi incorporati e code di messaggi non recapitabili per eventi non elaborati aiutano gli sviluppatori a costruire sistemi robusti con un'affidabilità totale utilizzando approcci serverless. Anche le prestazioni sono fondamentali, specie la bassa latenza (overhead), dato che di runtime del linguaggio e codice del cliente viene creata un'istanza on demand, in un'app serverless.

Infine, la piattaforma deve disporre di un'ampia gamma di *controlli di sicurezza e accesso*, tra cui il supporto alle reti private virtuali, autorizzazioni basate sui ruoli e sugli accessi, una solida integrazione con meccanismi di autenticazione e controllo degli accessi basati su API (compresi i sistemi legacy e di terze parti) e il supporto per la crittografia di elementi applicativi, come le impostazioni delle variabili di ambiente. Progettualmente, i sistemi serverless offrono un livello intrinsecamente superiore di sicurezza e controllo per le seguenti ragioni:

- **Gestione eccellente della flotta, comprese le patch di sicurezza** – In un sistema come AWS Lambda, i server che eseguono le richieste sono costantemente monitorati, attivati e disattivati e sottoposti a scansione di sicurezza. Possono contare su patch entro poche ore dalla disponibilità degli aggiornamenti di sicurezza chiave, a differenza di molte flotte di elaborazione aziendali che possono avere SLA molto più flessibili per le patch e gli aggiornamenti.
- **Durata limitata del server** – Ogni macchina che esegue il codice cliente in AWS Lambda viene attivata e disattivata più volte al giorno, limitando la sua esposizione agli attacchi e garantendo un sistema operativo e patch di sicurezza aggiornati costantemente.
- **Autenticazione su richiesta, controllo dell'accesso e auditing** – Ogni richiesta di elaborazione eseguita in AWS Lambda, a prescindere dalla sua fonte, viene autenticata individualmente, autorizzata ad accedere a risorse specifiche e sottoposta a audit completo. Le richieste provenienti dall'esterno dei data center AWS tramite Amazon API Gateway forniscono ulteriori sistemi di difesa nei confronti di Internet, compresi quelli contro attacchi DoS. Le società che migrano ad architetture serverless possono usare AWS CloudTrail per ottenere informazioni dettagliate su utenti e sistemi a cui accedono e con quali privilegi; inoltre possono usare AWS Lambda per elaborare programmaticamente i record di audit.

La piattaforma serverless AWS

Dall'introduzione di Lambda nel 2014, AWS ha creato una piattaforma serverless completa. Dispone di un'ampia raccolta di servizi interamente gestiti che permettono alle organizzazioni di realizzare app serverless in grado di integrarsi fluidamente con altri servizi AWS e di terze parti. Figura illustra un sottoinsieme di componenti nella piattaforma serverless AWS e le loro relazioni.

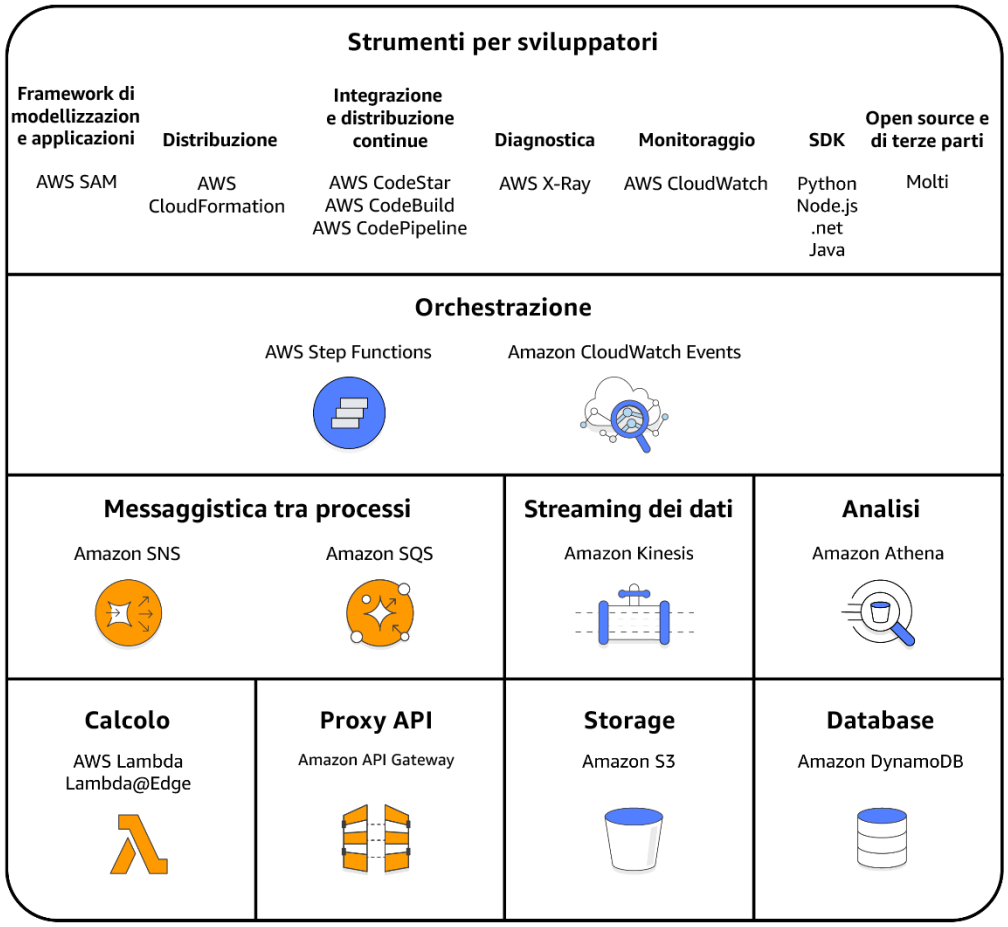

Figura 2: Componenti della piattaforma serverless AWS

Capacità della piattaforma serverless AWS

AWS offre tutte le funzionalità di base identificate nella sezione precedente come requisiti per una piattaforma serverless completa. Il livello della logica cloud viene fornito da AWS Lambda, una piattaforma di elaborazione serverless di fascia alta che non richiede provisioning, basata sulle funzioni. AWS Lambda è completata da AWS Lambda@Edge, che offre un supporto simile per eseguire funzioni Lambda a latenza estremamente bassa utilizzando routing ottimizzato per l'edge e AWS Greengrass, che permette l'esecuzione di funzioni Lambda in dispositivi connessi, comprese appliance come AWS Snowball.

Le funzioni Lambda possono essere facilmente attivate da diversi eventi di prime e di terze parti, consentendo agli sviluppatori di realizzare sistemi reattivi e determinati dagli eventi (vedere la Figura 3) senza la solita seccatura di dover configurare e gestire l'infrastruttura. In presenza di molti eventi simultanei, Lambda esegue semplicemente più copie della funzione in parallelo, rispondendo a ogni singolo trigger. Le funzioni Lambda si adeguano in modo preciso alle dimensioni del carico di lavoro, fino alla singola richiesta. Di conseguenza, non vi è alcuna possibilità di server o contenitori inattivi. Il problema degli sprechi in infrastruttura è eliminato *a livello progettuale* nelle architetture che usano le funzioni Lambda.

FaaS, o *Function as a Service*, è un approccio alla costruzione di sistemi informatici basati su eventi che poggia su funzioni quali l'unità di distribuzione e di esecuzione. *Serverless FaaS* è un tipo di FaaS in cui nel modello di programmazione non sono presenti contenitori o macchine virtuali e dove il fornitore offre scalabilità senza provisioning e affidabilità integrata.

Figura 3: Relazione tra elaborazione guidata dagli eventi, FaaS e serverless.

Le capacità di elaborazione serverless AWS messe a disposizione da Lambda sono un elemento chiave dei seguenti servizi gestiti offerti da AWS, che si integrano tutti perfettamente l'uno con l'altro:

- Amazon API Gateway – Endpoint HTTP per le funzioni Lambda, inclusa una gamma completa di proxy API e funzionalità di gestione API.
- Amazon S3 – Le funzioni Lambda possono essere usate come trigger di eventi automatici quando si crea, copia o elimina un oggetto.
- Amazon DynamoDB – Le funzioni Lambda possono essere usate per elaborare tutte le modifiche apportate a una tabella di database, o una qualsiasi di esse.
- Amazon SNS – I messaggi possono essere indirizzati alle funzioni Lambda per essere elaborati, aggiungendo la capacità di risposta dinamica al contenuto pubblicato.
- Amazon SQS – I messaggi nelle code possono essere facilmente elaborati tramite le funzioni Lambda.
- Amazon Kinesis Streams – Le funzioni Lamba forniscono l'elaborazione nell'ordine delle registrazioni dei dati in streaming, semplificando la creazione di motori di analisi quasi in tempo reale.
- Amazon Kinesis Firehose – Le funzioni Lambda possono essere applicate automaticamente alle registrazioni acquisite da un Firehose, agevolando l'aggiunta di capacità di trasformazione, filtraggio e analisi a un flusso di dati.
- Amazon Athena – Le funzioni Lambda possono essere attivate automaticamente per ogni oggetto nel set di risultati di una query.
- AWS Step Functions – È possibile orchestrare più funzioni Lambda per creare flussi di lavoro di lunga durata per processi sia manuali che automatizzati.
- Amazon CloudWatch Events – Le funzioni Lambda possono essere usate per rispondere automaticamente agli eventi, compresi quelli di terze parti.
- Amazon Aurora – I trigger di database possono essere scritti come funzioni Lambda.

Lambda offre una libreria di integrazione con blueprint per un'ampia varietà di servizi di terze parti, compresi tra gli altri Slack, Algorithmia, Twilio, Loggly, Splunk, SumoLogic e Box. Gli sviluppatori possono così creare applicazioni reattive che includono funzioni analitiche, algoritmi avanzati, comunicazioni e altro ancora, con solo poche righe di codice. Numerosi framework di applicazioni Web, tra cui Express (per applicazioni NodeJS) e Flask (per applicazioni Python) sono stati migliorati per lavorare senza problemi con le funzioni Lambda. I progetti di applicazioni Web open source includono Serverless Framework, Sparta, Chalice, e molto altro.

Le app serverless sono in genere composte di diversi pezzi: una o più funzioni, un database serverless come Amazon DynamoDB e un'API per i client da chiamare o una fonte evento che attiva l'app. Per gestire questi pezzi, AWS usa SAM, il Serverless Application Model con specifica aperta. Grazie a SAM, gli sviluppatori possono descrivere facilmente le funzioni, le API, le fonti degli eventi, le tabelle dei database e altre parti di un'app serverless. SAM li aiuta inoltre a gestire tutte le fasi del ciclo di vita dello sviluppo software. AWS offre una serie di strumenti e servizi utili, fra cui supporto nativo per il testing e il debugging locale nell'IDE di loro scelta (o tramite riga di comando) tramite SAM Local, distribuzione di app SAM con AWS CloudFormation, supporto per la costruzione di app SAM in AWS CodeBuild e supporto per CI/CD basati su GitHub per le app SAM incorporate in AWS CodePipeline. Oltre al supporto delle prime parti, numerosi framework open source, provider di CI/CD e performance management vendor offrono supporto per SAM e funzioni Lambda, incluse Serverless Framework, Claudia, CloudBees, Datadog, e molte altre ancora. Per altri esempi, vedere Strumenti per sviluppatori per applicazioni senza server.[4]

Dopo aver creato una funzione Lambda (o, nel caso di un'app SAM, potenzialmente diverse funzioni Lambda che operano assieme), gli sviluppatori possono monitorarla facilmente utilizzando la metrica creata automaticamente e i log disponibili in Amazon CloudWatch e CloudWatch Logs. AWS offre anche AWS X-Ray, una soluzione di analisi delle prestazioni e di tracciamento delle richieste fra i servizi che permette agli sviluppatori di tener traccia del funzionamento e del comportamento delle singole funzioni e degli eventi che elaborano.

Le offerte della piattaforma AWS serverless hanno una portata globale, con supporto per AWS Lambda e Amazon API Gateway praticamente in tutte le regioni AWS nel mondo. Lambda@Edge è disponibile in tutte le edge location.[5] Lambda offre una serie di funzionalità per aiutare i clienti a migliorare

l’affidabilità delle loro applicazioni, inclusi i tentativi automatici per eventi asincroni e ordinati e le code di messaggi non recapitabili per acquisire eventi che non sono stati elaborati con successo dall'applicazione. La profonda integrazione con Amazon Virtual Private Cloud (Amazon VPC) e la gamma flessibile di funzionalità di controllo dell’accesso e di autenticazione offerte da AWS Lambda consentono alle organizzazioni di creare applicazioni sicure che rispettano le best practice, come il principio del privilegio minimo. La sicurezza e la gestione dell’utente finale sono altrettanto facili – Amazon Cognito offre autorizzazione e autenticazione che possono essere facilmente combinate con Amazon API Gateway e AWS Lambda, mettendo a disposizione capacità di iscrizione e registrazione di utenti serverless, inclusa l’integrazione con social provider come Facebook e directory aziendali.

Casi di studio

Le società hanno applicato architetture serverless per utilizzare casi che vanno dalla validazione di operazioni sui titoli alla costruzione di siti di e-commerce, fino all'elaborazione del linguaggio naturale. AWS Lambda e il resto del portafoglio serverless AWS offrono la flessibilità di creare un'ampia gamma di applicazioni, comprese quelle che richiedono misure di sicurezza a privacy come la conformità PCI o HIPAA. Le sezioni seguenti illustrano alcuni dei casi d'uso più comuni, ma non costituiscono un elenco completo. Per un elenco completo dei riferimenti dei clienti e per la documentazione dei casi d'uso, vedere Applicazioni ed elaborazione serverless.[6]

Siti Web, app Web e backend mobili serverless

Gli approcci serverless sono ideali per applicazioni in cui il carico può variare dinamicamente. Utilizzare un approccio serverless significa non dover sostenere costi di elaborazione quando non c'è traffico di utenti finali, pur offrendo un dimensionamento istantaneo per soddisfare la domanda elevata, come una vendita flash su un sito di e-commerce o una segnalazione sui social media che scatena un'ondata improvvisa di traffico. Rispetto agli approcci infrastrutturali tradizionali, spesso è anche molto meno costoso sviluppare, fornire e gestire un backend Web o mobile quando è stato architettato in modo serverless.

AWS fornisce i servizi di cui gli sviluppatori handno bisogno per costruire rapidamente queste applicazioni:

- Amazon S3 offre una soluzione di hosting semplice per contenuti statici.
- AWS Lambda, assieme ad Amazon API Gateway, fornisce supporto per le richieste API dinamiche che utilizzano le funzioni.
- Amazon DynamoDB offre una soluzione di storage semplice per la sessione e in funzione dello stato del singolo utente.
- Amazon Cognito fornisce un modo semplice per gestire la registrazione, l'autenticazione e il controllo dell'accesso alle risorse da parte dell'utente finale.
- AWS SAM può essere usato dagli sviluppatori per descrivere i vari elementi di un'applicazione.
- AWS CodeStar può impostare una toolchain CI/CD con soli pochi clic.

Per maggiori informazioni, vedere il whitepaper AWS Serverless Multi-Tier Architectures, che contiene un esame dettagliato di pattern per costruire applicazioni Web serverless.[7] Per architetture di riferimento complete, vedere Serverless Reference Architecture for creating a Web Application[8] e Serverless Reference Architecture for creating a Mobile Backend[9] su GitHub.

Esempio di un cliente – Bustle.com

Bustle.com è un sito Web di notizie, intrattenimento, stile di vita e moda per donne. Il passaggio a un'architettura serverless basata su AWS Lambda e Amazon API Gateway gli ha fruttato un risparmio sui costi di circa l'84%. Gli ingegneri di Bustle hanno acquisito maggiore agilità e hanno potuto concentrarsi sulla realizzazione di nuove funzionalità del prodotto invece di occuparsi della gestione dell'infrastruttura e della scalabilità. Ora il team di Bustle è più efficiente e impiega la metà delle persone normalmente richieste per realizzare e gestire siti della scala di Bustle. Il backend serverless di Bustle supporta anche le app iOS per due delle sue proprietà Web (Bustle e Romper). Per maggiori informazioni, vedere Bustle case study.[10]

Backend IoT

Fra i vantaggi che un'architettura serverless porta al Web e alle app mobili vi è anche la semplificazione della costruzione di backend IoT e sistemi di elaborazione analitica basati su dispositivi che si adattano perfettamente al numero di dispositivi. Per un'architettura di riferimento di esempio, vedere Serverless Reference Architecture for creating an IoT Backend su GitHub.[11]

Esempio di un cliente – iRobot

iRobot, che fabbrica robot come l'aspirapolvere Roomba, usa AWS Lambda assieme al servizio AWS IoT per creare un backend serverless per la sua piattaforma IoT. Usando un'architettura serverless, il team di tecnici e ingegneri di iRobot non deve preoccuparsi di gestire l'infrastruttura o scrivere manualmente i codice per gestire disponibilità e scalabilità. Possono così innovare più rapidamente e rimanere concentrati sui clienti. Per maggiori informazioni, vedere le diapositive per la loro presentazione AWS re:Invent 2016 Serverless IoT Back Ends (IOT401)[12] o guardare il video.[13]

Elaborazione dei dati

Le più grandi applicazioni serverless elaborano enormi volumi di dati, in gran parte in tempo reale. Le tipiche architetture serverless di elaborazione dei dati usano una combinazione di Amazon Kinesis e AWS Lambda per elaborare i dati in streaming, oppure combinano Amazon S3 e AWS Lambda per attivare il calcolo in risposta alla creazione di oggetti o a eventi di aggiornamento. Quando i carichi di lavoro richiedono un'orchestrazione più complessa di un semplice trigger, gli sviluppatori possono usare AWS Step Functions per creare flussi di lavoro stateful o di lunga durata che richiamano una o più funzioni Lambda man mano che vanno avanti. Per maggiori informazioni sulle architetture di elaborazione dati serverless, vedere quanto segue su GitHub:

- Serverless Reference Architecture for Real-time Stream Processing[14]
- Serverless Reference Architecture for Real-time File Processing[15]
- Image Recognition and Processing Backend reference architecture[16]

Esempio di un cliente – FINRA

La Financial Industry Regulatory Authority (FINRA) ha usato AWS Lambda per costruire una soluzione di elaborazione dati serverless che le consenta di convalidare mezzo trilione di dati su 37 miliardi di eventi borsistici al giorno. Nel suo intervento ad AWS re:Invent 2016 intitolato The State of Serverless Computing (SVR311),[17] Tim Griesbach, Senior Director di FINRA, ha detto "Ci siamo resi conto che Lambda ci avrebbe fornito lo strumento migliore per questa soluzione cloud serverless. Con Lambda, il sistema era più rapido, meno costoso e più scalabile. Alla fine, abbiamo ridotto i nostri costi di oltre il 50% … e possiamo avere un tracciamento giornaliero, addirittura ora per ora."

Esempio di un cliente – Thomson Reuters

Thomson Reuters, un'azienda di media e informazione, ha creato una soluzione di analisi aziendale serverless che consente ai suoi team dei prodotti di analizzare facilmente i dati sull'utilizzo che ne viene fatto. La soluzione combina AWS Lambda, Amazon Kinesis Streams e Amazon Kinesis Firehose per raccogliere edelaborare dati di eventi in streaming a fini di analisi. Il risultato, chiamato Product Insight, è stato lanciato con due mesi di anticipo rispetto al programma e ha superato le aspettative di carattere tecnico.

Anders Fritz, senior manager dell'innovazione dei prodotti per Thomson Reuters, ha dichiarato "Il nostro obiettivo iniziale era far fronte a 2.000 eventi al secondo. I nostri test dimostrano che Product Insight su AWS è in grado di elaborare fino a 4.000 eventi al secondo e, entro un anno, ci aspettiamo di superare i 10.000 eventi al secondo." Questo numero equivale a oltre 25 miliardi di eventi al mese. Nonostante questo throughput elevato, il sistema non ha perso nemmeno un dato da quando è entrato in funzione. "Grazie alla robusta architettura di failover e alle capacità tecniche di AWS, non abbiamo perso un singolo evento da quando abbiamo iniziato a raccogliere i dati", ha affermato Fritz. Per maggiori informazioni, vedere il Thomson Reuters Case Study[18] oppure guardare la presentazione AWS re:Invent 2016 Real-time Data Processing Using AWS Lambda (SVR301).[19]

Big Data

AWS Lambda è la soluzione perfetta per molti carichi di lavoro di elaborazione parallela di grandi volumi di dati. Per un esempio di architettura di riferimento che utilizza MapReduce, vedere Reference architecture for running serverless MapReduce jobs.[20]

Esempio di un cliente – Fannie Mae

Fannie Mae, una delle principali fonti di finanziamento per gli erogatori di mutui ipotecari, utilizza AWS Lambda per eseguire un carico di lavoro "estremamente parallelo" per la sua modellazione finanziaria. Per gestire meglio il rischio ipotecario, Fannie Mae usa i processi di simulazione Monte Carlo per prevedere i flussi di cassa futuri dei mutui ipotecari. La società ha scoperto che le sue reti HPC esistenti non erano più in grado di soddisfare le sue crescenti esigenze aziendali. Fannie Mae ha costruito la sua nuova piattaforma su Lambda e il sistema è riuscito ad arrivare a 15.000 esecuzioni simultanee di funzioni durante

il test. Il nuovo sistema ha eseguito una simulazione su 20 milioni di mutui ipotecari che si è conclusa in 2 ore, ovvero tre volte più velocemente rispetto al vecchio sistema. Usando un'architettura serverless, Fannie Mae ora può eseguire le simulazioni Monte Carlo su larga scala e in modo economicamente vantaggioso perché non paga le risorse di calcolo inutilizzate. Può anche velocizzare i calcoli eseguendo più funzioni Lambda contemporaneamente. Fannie Mae ha ottenuto anche un time to market più rapido di quello consueto perché è riuscita a fare a meno della gestione e del monitoraggio dei server. Inoltre, ha potuto eliminare gran parte del complesso codice precedentemente richiesto per gestire la scalabilità e l'affidabilità delle applicazioni. Per maggiori informazioni, vedere la presentazione di Fannie Mae al Summit AWS 2017 SMC303: Real-time Data Processing Using AWS Lambda.[21]

Automazione IT

Gli approcci serverless eliminano l'overhead della gestione dei server, rendendo molto più facile creare e gestire la maggior parte delle attività infrastrutturali, tra cui provisioning, configurazione, gestione, allarmi/monitoraggi e cron job.

Esempio di un cliente – Autodesk

Autodesk, che produce software di progettazione 3D e ingegneria, usa AWS Lambda per automatizzare i suoi processi di creazione e gestione di account AWS in tutta la sua organizzazione ingegneristica. Autodesk stima di aver ottenuto risparmi sui costi pari al 98% (tenendo conto dei risparmi stimati in ore di manodopera dedicate agli account di provisioning). Ora riesce a effettuare il provisioning degli account in soli 10 minuti invece delle 10 ore richieste con il precedente processo basato sull'infrastruttura. La soluzione serverless permette ad Autodesk di effettuare automaticamente il provisioning degli account, configurare e applicare gli standard ed eseguire audit con maggiore automazione e meno punti di contatto manuali. Per maggiori informazioni, vedere la presentazione di Autodesk al Summit AWS 2017 SMC301: The State of Serverless Computing.[22] Visitare GitHub per vedere il servizio Autodesk Tailor.

Altri casi d'uso

I casi d'uso descritti nella sezione precedente graffiano solo la superficie di ciò che è ossibile fare con Lambda e con le altre offerte serverless AWS. Altri casi d'uso includono la comprensione estesa del linguaggio umano attraverso chatbot costruiti utilizzando Amazon Lex e AWS Lambda, Edge Computing globale a bassa latenza utilizzando Lambda@Edge con Amazon CloudFront e na potente elaborazione file on-premise con le funzioni Lambda all'interno di un AWS Snowball. Queste sono solo alcune delle straordinarie capacità di questo approccio versatile. Per maggiori informazioni, vedere AWS Lambda.[23]

Conclusioni

Gli approcci serverless sono progettati per affrontare due classici problemi di gestione IT: i server inattivi che prosciugano il bilancio di un'azienda senza offrire valore e il costo di costruzione e gestione di flotte di server e software per server, che istrae e distoglie dall'attività di creazione di valore differenziato per il cliente. AWS Lambda e le altre offerte serverless AWS risolvono questi annosi problemi eliminando server, contenitori, dischi e altre risorse a livello di infrastruttura dal modello di programmazione e fatturazione. Di conseguenza, gli sviluppatori possono lavorare con un modello di applicazione pulito che li aiuta a essere più veloci e le organizzazioni pagano solo per il lavoro utile. Il modo più semplice e veloce per architettare sistemi reattivi basati su eventi e fornire microservizi nativi per il cloud è ttraverso l'uso di architetture serverless. Per maggiori informazioni e whitepaper su rgomenti correlati, vedere Applicazioni ed elaborazione serverless.[24]

Collaboratori

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

- Tim Wagner, General Manager di AWS Serverless Applications, Amazon eb Services

Approfondimenti

Per ulteriori informazioni, consultare le seguenti fonti:

- Serverless Reference Architectures with AWS Lambda di Werner Vogels, CTO di Amazon.com
- AWS re:Invent 2016: The State of Serverless Computing [presentazione] di Tim Wagner, General Manager di AWS Serverless Applications
- The economics of serverless cloud computing di Owen Rogers, Research Director presso 451 Research

Architetture di riferimento

- Applicazioni Web
- Backend mobili
- Backend IoT
- Elaborazione di file
- Elaborazione di flussi
- Elaborazione del riconoscimento delle immagini
- MapReduce

Revisioni del documento

Data	Descrizione
Settembre 2017	Prima pubblicazione

Notes

[1] https://www.forbes.com/sites/moorinsights/2016/04/11/tco-analysis-

demonstrates-how-moving-to-the-cloud-can-save-your-company-money/#537e2bd07c4e
http://www.cloudstrategymag.com/articles/86033-understanding-tco-cloud-economics

[2] Nel 2012, l'utilizzo dei data center stimato da Gartner è passato dal 7 al 12% (http://www.nytimes.com/2012/09/23/technology/data-centers-waste-vast-amounts-of-energy-belying-industry-image.html). Secondo uno studio McKinsey del 2008, era al 6% (https://www.sallan.org/pdf-docs/McKinsey_Data_Center_Efficiency.pdf). Un documento Accenture che ha analizzato una serie di applicazioni basate su EC2, ha riscontrato un utilizzo di circa il 7% (http://ieeexplore.ieee.org/document/6118751/). Uno studio del 2014 condotto da NRDC e Anthesis ha rilevato che, nel 2013, oltre il 30% dei server era totalmente "comatoso" (collegato, ma senza far nulla di importante) (http://anthesisgroup.com/wp-content/uploads/2015/06/Case-Study_DataSupports30PercentComatoseEstimate-FINAL_06032015.pdf).

[3] Occupy the Cloud: Eric Jonas et al., *Distributed Computing for the 99%*, https://arxiv.org/abs/1702.04024.

[4] https://aws.amazon.com/serverless/developer-tools

[5] https://aws.amazon.com/lambda/edge/

[6] https://aws.amazon.com/serverless/

[7] https://d0.awsstatic.com/whitepapers/AWS_Serverless_Multi-Tier_Architectures.pdf

[8] https://github.com/awslabs/lambda-refarch-webapp

[9] https://github.com/awslabs/lambda-refarch-mobilebackend

[10] https://aws.amazon.com/solutions/case-studies/bustle/

[11] https://github.com/awslabs/lambda-refarch-iotbackend

[12] https://www.slideshare.net/AmazonWebServices/aws-reinvent-2016-serverless-iot-back-ends-iot401

[13] https://www.youtube.com/watch?v=gKMaf5E-z7Q

[14] https://github.com/awslabs/lambda-refarch-streamprocessing

[15] https://github.com/awslabs/lambda-refarch-fileprocessing

[16] https://github.com/awslabs/lambda-refarch-imagerecognition

[17] https://www.youtube.com/watch?v=AcGv3qUrRC4&feature=youtu.be&t=1153

[18] https://aws.amazon.com/solutions/case-studies/thomson-reuters/

[19] https://www.youtube.com/watch?v=VFLKOy4GKXQ&feature=youtu.be&t=1449

[20] https://github.com/awslabs/lambda-refarch-mapreduce

[21] https://www.slideshare.net/AmazonWebServices/smc303-realtime-data-processing-using-aws-lambda/28

[22] https://www.slideshare.net/AmazonWebServices/smc301-the-state-of-serverless-computing-75290821/22

[23] https://aws.amazon.com/lambda/

[24] https://aws.amazon.com/serverless/