GO SERVERLESS! PARTE 1: REALIZZIAMO UN'APPLICAZIONE DI FILE SHARING BASATA SUI SERVIZI DI AWS

AWS Lambda | CI/CD | Continuous Delivery | Serverless

beSharp | 11 Settembre 2018

Negli ultimi anni, il termine **"Serverless"** ha preso sempre più piede nell'ambito IT.

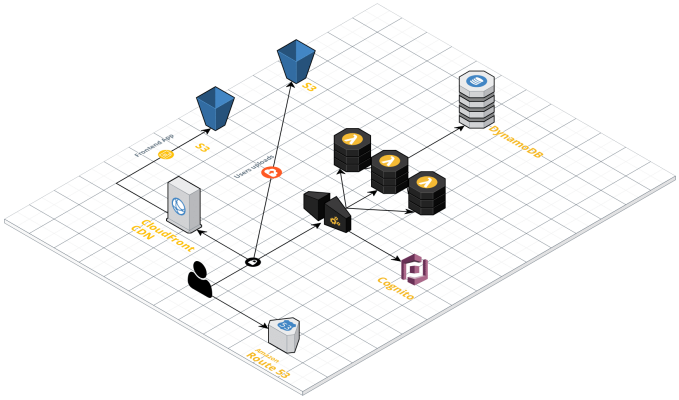
L'ennesima buzzword? Oppure è davvero possibile sviluppare un'applicazione che non utilizzi alcun server, come questa parola suggerirebbe? Cerchiamo di chiarire questo possibile equivoco...

Serverless è un **paradigma Cloud** che permette l'esecuzione di applicazioni in maniera agnostica rispetto all'infrastruttura sottostante. Grazie a servizi Cloud come (ad esempio) **AWS Lambda, API Gateway ed EKS,** si possono sviluppare applicazioni per le quali le operazioni di provisioning, scalabilità e gestione siano svolte in maniera trasparente ed automatizzata, senza l'intervento manuale di chi scrive il codice.

Questa nuova tecnologia ha numerosi vantaggi che migliorano sia l'esperienza dell'utente che quella dello sviluppatore e saperli sfruttare è fondamentale per realizzare soluzioni competitive, altamente scalabili e performanti.

In questa serie di 3 articoli illustreremo come sia possibile costruire un **sistema di file sharing** completo di login, "drag and drop" e condivisione dei file mediante link, sfruttando le potenzialità della tecnologia serverless e i servizi messi a disposizione da AWS.

Amazon mette a disposizione una vasta gamma di **servizi gestiti** che è possibile sfruttare per implementare rapidamente architetture completamente serverless: realizzeremo quindi un'infrastruttura così come rappresentata nell'illustrazione sottostante, per poi analizzare nel dettaglio come configurare i trigger necessari per rispondere agli eventi, come l'upload dei file.

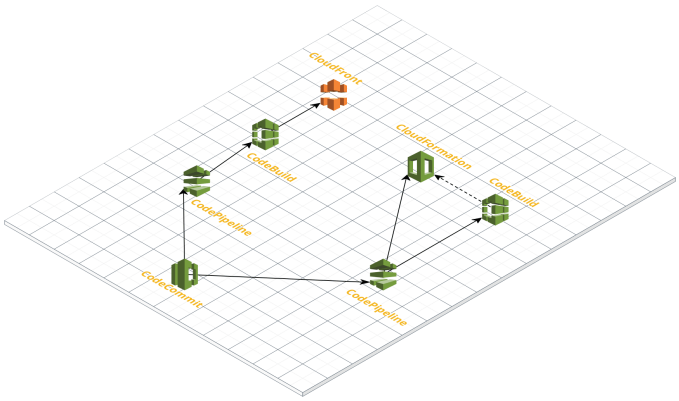


L'applicazione si comporrà di un front-end statico e di una serie di API di back-end, entrambi protetti da autenticazione. Le informazioni sui file e le condivisioni saranno salvate su un database di tipo chiave-valore, mentre tutti i file verranno salvati su object storage.

C'è un punto interessante che vale la pena sottolineare: gli upload degli utenti saranno effettuati direttamente sull'object storage mediante un apposito meccanismo di autorizzazione. Questo permetterà all'applicazione di **scalare in modo rapido** all'aumento delle richieste, minimizzando le risorse da preallocare ed i relativi costi.

Per realizzare questa soluzione andremo ad impiegare alcuni dei servizi Serverless più interessanti, come **S3** per lo storage degli oggetti, **DynamoDB** come base dati, **Lambda** per l'esecuzione delle API di back-end, **Api Gateway** per esporre le API, **CloudFront** come CDN per servire il front-end e **Cognito** per i servizi di registrazione e autenticazione degli utenti.

Affronteremo anche il tema del deploy automatico del codice realizzando una pipeline di **Continuous Deployment/Continuous Integration** che consenta di rilasciare in modo agile ed automatico le modifiche al software.



Nello specifico saranno realizzate due pipeline separate per gestire la CD/CI di back-end e front-end. Quella del back-end utilizzerà un template di CloudFormation per effettuare il provisioning delle funzioni Lambda e aggiornare il codice, mentre quella del front-end provvederà alla copia degli asset sull'apposito bucket e a gestire la CDN con le invalidazioni necessarie.

Andiamo ora ad illustrare i servizi utilizzati in funzione delle scelte progettuali.

Amazon S3

“Amazon S3 è uno storage di oggetti creato per memorizzare e ripristinare qualsiasi volume di dati da qualsiasi origine. È un servizio che offre un’infrastruttura di storage estremamente durevole, altamente disponibile e scalabile in modo illimitato a costi ridotti.”

Sfrutteremo S3 sia come sorgente per i file che compongono il Front-End dell’applicazione sia come storage per i file caricati dall’utente. L’utilizzo di questo servizio permette di avere spazio virtualmente illimitato altamente disponibile e a costi ridotti.

Per rendere l’implementazione più sicura utilizzeremo 2 Bucket separati, allo scopo di partizionare meglio i permessi e tenere fortemente isolato lo storage applicativo da quello dei dati degli utenti. Il Bucket riservato agli asset statici sarà la sorgente per la CDN utilizzata per distribuire il frontend ai client.

CloudFront

“Amazon CloudFront è una rete per la distribuzione di contenuti o CDN (Content Delivery Network) globale, che permette la distribuzione di dati, video, applicazioni e API agli utenti con latenza minima e velocità di trasferimento elevata.”

Utilizzeremo CloudFront per servire il Fornt-end della nostra applicazione. Oltre a migliorare le performance e l’esperienza utente, permette anche di ottenere protezione da **attacchi di tipo DDoS** e di abbattere i costi di accesso ad S3 per file che vengono richiesti frequentemente.

DynamoDB

“Amazon DynamoDB è un database non relazionale che fornisce prestazioni affidabili su qualsiasi scala. Si tratta di un database multi-master, multiregione e completamente gestito che fornisce latenza costante di pochi millisecondi e sicurezza integrata, backup e ripristino e cache in memoria.”

La scelta di DynamoDB su altri validi DBMS è dovuta principalmente dal fatto che l’applicazione salva metadati relativi ai file caricati e alla loro condivisione, dove tutto è fortemente “file centrico”. La struttura dei dati è calzante con il modello chiave valore di DynamoDB, condizione resa possibile tra l’altro dalla delega della gestione dell’autorizzazione e dell’anagrafica utenti a Cognito.

Inoltre, scegliendo di utilizzare DynamoDB, l’applicazione beneficia in modo quasi del tutto automatico di una base dati facilmente e automaticamente scalabile orizzontalmente, **full managed** e altamente disponibile.

Cognito

“Amazon Cognito permette di aggiungere strumenti di registrazione, accesso e controllo degli accessi alle app Web e per dispositivi mobili in modo rapido e semplice. Amazon Cognito permette di ricalibrare le risorse per milioni di utenti e supporta l’accesso con provider di identità social quali Facebook, Google e Amazon e provider di identità aziendali tramite SAML 2.0.”

Abbiamo scelto di delegare l'autenticazione e la gestione dell'anagrafica utenti ad un servizio gestito. Questo consente di limitare la superficie d'attacco del servizio che stiamo costruendo, aumentare le possibilità di login e autenticazione e disporre di una integrazione rapidissima e bullet proof con API gateway. Quest'ultima permette di ridurre i costi in quanto tutto l'effort di autenticazione viene effettuato dal gateway e non incide sul tempo di calcolo delle Lambda. Inoltre, eventuali tentativi di login errati o chiamate effettuate con token scaduti o errati non raggiungeranno mai l'applicazione ma saranno gestiti direttamente sul perimetro esterno dell'architettura.

L'anagrafica gestita consente anche di annullare il traffico sulla base dati per reperire le principali informazioni sugli utenti, sgrava dal mantenimento e dalla messa in sicurezza delle tabelle con dati sensibili e alla gestione delle credenziali.

Lambda

"AWS Lambda consente di eseguire codice senza dover effettuare il provisioning né gestire server. Le tariffe sono calcolate in base ai tempi di elaborazione, perciò non viene addebitato alcun costo quando il codice non è in esecuzione."

Il back-end sarà sviluppato interamente con AWS Lambda. Questa tecnologia, principe della filosofia serverless, permette di pagare per singola esecuzione, e solo per il tempo effettivamente utilizzato, annullando i costi di idle, lo spreco di potenza di calcolo e risparmiandoci di dover configurare e testare una adeguata politica di auto scaling.

Api Gateway

"Amazon API Gateway è un servizio completamente gestito che semplifica agli sviluppatori creazione, pubblicazione, manutenzione, monitoraggio e protezione delle API su qualsiasi scala."

In questo caso si tratta di una scelta naturale, solo AWS Api Gateway dispone di tutti gli hook e le integrazioni necessarie ad un inserimento seamless ed efficace nell'architettura applicativa. Lo scopo principale è quello di fornire un'interfaccia altamente disponibile, performante ed affidabile alle funzioni lambda del backend. Permette inoltre di integrare Cognito per autenticare le chiamate e passare al backend solo quelle correttamente autenticate e legittime, bloccando di fatto tutte le richieste non valide. Un altro vantaggio è quello di poter integrare facilmente cloudfront per conservare in cache le risposte del backend per **risparmiare in termini di tempo di calcolo** e aumentando allo stesso tempo la responsività dell'applicazione riducendo la latenza.

Nei prossimi articoli esamineremo in dettaglio la struttura dell'applicazione, l'infrastruttura serverless e i trigger da configurare per rispondere agli eventi importanti per l'applicazione.

Stay tuned!

Leggi la seconda parte | Leggi la terza parte

beSharp

Dal 2011 beSharp guida le aziende italiane sul Cloud. Dalla piccola impresa alla grande multinazionale, dal manifatturiero al terziario avanzato, aiutiamo le realtà più all'avanguardia a realizzare progetti innovativi in campo IT.

Get in touch

beSharp.it
proud2becloud@besharp.it