Blog

<small>di W. Dal Mut, Lead of Infrastructures @ Corley - whitepaper 2020</small> <hr>

Creare microservizi ed API in genere è un processo delicato in quanto ha come obiettivo la costruzione delle fondamenta delle nostre soluzioni applicative e sempre più spesso coinvolge diverse sorgenti di dato, applicativi terzi e API esterne. Per questo motivo **è di strategica importanza l'utilizzo di un gateway** che permetta di fruire in modo flessibile e centralizzato dei nostri servizi ed aggiungere le tante funzionalità in modo centrale e trasparente, tra cui:

- Rate Limits
- Caching
- Authentication
- Authorization
- Documentation
- etc.

Grazie al servizio AWS Api Gateway possiamo integrare le nostre API in un concentratore riuscendo in questo modo ad avere un singolo punto di accesso a controllo di tutti i nostri microservizi (o servizi in generale).

Come si usa Api Gateway

Il modo migliore per fruire del servizio Api Gateway è nel contesto Serverless, cioè in uno scenario che comprenda alcuni servizi di punta di AWS come Lambda e DynamoDB, e nello specifico tramite il tool SAM, ovvero **Serverless Application Model**.

Questo componente permette di integrare le tecniche di *infrastructure as a code* al contesto di sviluppo applicativo delle nostre API (per approfondire il capitolo IaaC è possibile consultare il whitepaper [stack di CloudFormation (https://corley.it/aws-cloudformation.html)](https://corley.it/aws-cloudformation.html).

I template di AWS SAM sono veramente semplici ed intuitivi. Contengono in poche righe di codice tutte le informazioni necessarie al comparto AWS per integrare e modellare il nostro layer serverless:

```
AWSTemplateFormatVersion:
'2010-09-09'
Transform:
AWS::Serverless-2016-10-31
Description: AWS SAM template
with a simple API definition
Resources:
  Api GatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: prod
  ApiFunction:
    Type:
AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: Api
          Properties:
            Path: /
            Method: get
            RestApiId:
              Ref: Api
GatewayApi
      Runtime: python3.7
      Handler: index.handler
      InlineCode: |
        def handler(event,
context):
```

```
                return {'body':
'Hello World!', 'statusCode':
200}
```

Con un semplice comando: `sam deploy --guided` possiamo quindi delegare a AWS il deploy della nostra infrastruttura che risulterà completamente funzionante in pochissimo tempo!

AWS SAM è ovviamente un tool più complesso e completo di così. Permette tra l'altro di gestire il codice sorgente anche tramite sistemi di pacchettizzazione, senza doverlo includere all'interno del template (che è utilizzato a solo scopo illustrativo).

In questo caso è sufficiente sostituire le regole *"InlineCode"* con il posizionamento del progetto software locale tramite la direttiva *CodeUri*. In questo modo l'esempio di prima può essere rivisto in questa modalità:

```
AWSTemplateFormatVersion:
'2010-09-09'
Transform:
AWS::Serverless-2016-10-31
Description: AWS SAM template
with a simple API definition
Resources:
  Api GatewayApi:
    Type: AWS::Serverless::Api
    Properties:
      StageName: prod
  ApiFunction:
    Type:
AWS::Serverless::Function
    Properties:
      Events:
        ApiEvent:
          Type: Api
          Properties:
```

```
                    Path: /
                    Method: get
                    RestApiId:
                      Ref: Api
        GatewayApi
              Runtime: python3.7
              Handler: index.handler
            CodeUri: my-app/
```

Anche in questo caso AWS SAM permette la gestione della pacchettizzazione applicativa con le varie dipendenze di progetto grazie al comando `sam build` che si occuperà di costruire l'artefatto completo per la messa in opera definitiva.

Api Gateway come proxy e come mock

Il servizio AWS Api Gateway permette di **proxare** non solo integrazioni AWS Lambda serverless ma addirittura **altri servizi classici**, per esempio ospitati su EC2 (server virtuali) o su un qualunque punto di accesso tramite il normale protocollo HTTPs.

In aggiunta, grazie al meccanismo *mock*, è possibile integrare risposte e controlli predefiniti senza la necessità di aggiungere alcun codice custom. Questa funzionalità si rivela utilissima in diversi contesti, come nel caso in cui si voglia abilitare il supporto al *CORS (Cross Origin Resource Sharing)* senza dover integrare un layer applicativo ad hoc.

Api Gateway gestioni degli Stage

Api Gateway permette inoltre di gestire diversi ambienti (*stage*), in modo da poter suddividere in modo semplice e rapido i diversi sotto-sistemi che vogliamo interrogare, tra cui solitamente ritroviamo:

- Produzione
- Testing
- QA
- ...

Per ogni stage abbiamo a disposizione un **ambiente dedicato**. Ciò permette di operare in modo perfetto sulle singole configurazioni, impostazioni e comportamenti che rimarranno legate al singolo ambiente, tra cui:

- Caching
- Rate Limits
- WAF (Web Application Firewall)
- Gestione dei nomi di dominio custom
- Logging (vedi anche il capitolo su Observability (https://corley.it/devops-observability.html))
- Tracing
- Variabili di Ambiente
- Modello di deployment (Rolling Upgrades, Recreation, Canary, etc)
- etc.

Api Gateway e le sue features di monitoring

Api Gateway è un componente estremamente flessibile e performante. Grazie alla sua struttura possiamo definire offline la **documentazione delle nostra API** (in formato Swagger 2.0, OpenAPI 3.0, etc.) e caricarla direttamente sul servizio abilitando così la costruzione completa dell'infrastruttura API pronta per essere integrate nei nostri software.

Ecco come si presenta la dashboard iniziale di un ambiente API:

Attraverso le dashboard di controllo è possibile accedere a diverse funzionalità di monitoring e gestione e ispezionare le singole features con un elevato livello di dettaglio.

Allo stesso modo Api Gateway permette di accedere a layer di **monitoring** di tutte le Api deployate, permettendoci in questo modo di controllare e verificare che il sistema sia in linea, operativo e perfettamente funzionante.

Come visto in altri whitepaper, le tecniche di monitoring di estendono ad altri capitoli e Api Gateway non è da meno, permettendoci integrare logiche di **tracing** e consentendoci così di tracciare ogni singola richiesta e il suo ciclo di vita all'interno dell'infrastruttura.

Api Gateway Authorizers e gestione degli accessi

Grazie all'estrema flessibilità del servizio è possibile integrare delle funzioni di autenticazione e autorizzazione. L'obiettivo di questa funzionalità è aggiungere un livello di sicurezza perfettamente distribuito sulle Api del sistema. Solitamente si procede attraverso l'integrazione di due funzionalità:

- AWS Lambda
- AWS Cognito

La prima permette una più flessibile gestione dei processi autenticativi ed autorizzativi, grazie alla completa customizzazione delle funzioni Lambda.

L'integrazione con Cognito da parte sua permette invece l'utilizzo di tutto lo stack di identificazione utente integrato in AWS. Il servizio è estremamente flessibile e garantisce la completa gestione utente sfruttando modelli serverless in modo semplice, veloce e sicuro.

Un terzo metodo permette di gestire in maniera ancora diversi il capitolo autenticativo, attraverso l'integrazione di specifiche **Api Key**. Questo meccanismo permette di generare delle chiavi di accesso da consegnare ai nostri utilizzatori, vincolando così la fruizione delle API alla presenza ditali chiavi durante il processamento di una richiesta.

Api Gateway Usage Plan

Un altra funzionalità molto interessante è l'integrazione dei **piani di utilizzo**, ovvero la gestione di limiti dedicati e delle quote di esercizio. Questa particolare feature è fondamentale per chi realizza servizi e vuole gestire in modo semplice quello che sono i piani di utilizzo del nostro sistema.

Api Gateway modelli, risposte e risorse

Il servizio Api Gateway è nato per offrire una completa delle nostre risorse API. Per questo motivo possiamo sfruttare un sistema *full managed* che rappresenta, di fatto, il più grande vantaggio di questa soluzione. Ogni componente, dalle risposte alla gestione degli *header* di ingresso ed uscita, dai flussi autorizzativi alle funzioni di trasformazione degli input possono essere configurati, mappati e gestiti

Il risultato è un'importante e incisiva semplificazione del processo di realizzazione di una Api e la velocizzazione dello sviluppo della nostra applicazione.