Come integrare l'API legacy con il proxy AWS API Gateway

3 Settembre 2021 - 7 min. read

L'emergere di moderne applicazioni Web e mobile, basate su microservizi che espongono le API HTTP, ha evidenziato la necessità di integrare, distribuire, decommissionare, limitare e proteggere efficacemente una pletora di API Web eterogenee. Il motivo principale che ha causato l'affermarsi di questa necessità è da ricercarsi nella disomogeneità intrinseca delle risorse di backend permesse e promosse dal modello a microservizi: ogni micro servizio che costituisce un'applicazione complessa può essere sviluppato e distribuito con criteri esclusivi (linguaggio di programmazione, piattaforma di distribuzione, ecc.). Alcuni dei sopra citati criteri possono essere delle SaaS API, ovvero del software completamente fuori dal controllo diretto dell'azienda che sviluppa l'applicazione nel suo complesso.

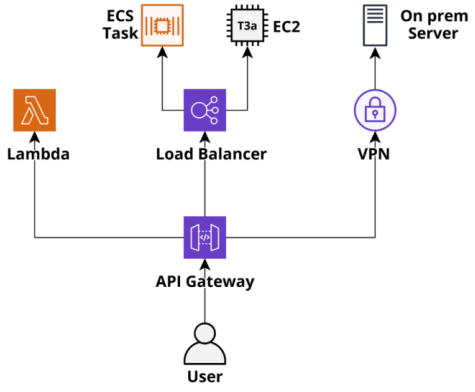
Diverse aziende e iniziative open source hanno sviluppato delle soluzioni **API gateway** al fine di soddisfare i requisiti di cui sopra, per aiutare gli sviluppatori e SysOps a gestire e proteggere le API in modo integrato ed infine per esporre un formato API consistente per tutti i microservizi che compongono l'applicazione.

Il servizio di Amazon che si fa carico delle necessità appena citate è **AWS API Gateway**, le cui integrazioni con servizi esistenti di AWS e di terze parti sono descritte nell'immagine sopra riportata. Essenzialmente questo servizio agisce da proxy (con la possibilità di modificare e arricchire le richieste in entrata), caching, load balancer e un generale layer di sicurezza generale per tutte le API.

A differenza di molte altre soluzioni, AWS API Gateway è completamente gestito e serverless; ciò si traduce nell'assenza di manutenzione e ridimensionamento in quanto le API vengono gestite da AWS su pagamento di una tariffa fissa calcolata sul tasso di richiesta (per le API REST ~ 1,5 $ per milione di richieste).

Di seguito verrà proposto un approfondimento dell'argomento principale: perché e come utilizzare questo servizio per l'integrazione di API legacy, preceduto da un breve riepilogo sulle integrazioni AWS API Gateway.

Integrazioni API Gateway



Nel contesto di API Gateway, un'integrazione API è il tipo di azione eseguita dal gateway per rispondere a una data richiesta API. L'integrazione viene invocata dopo la validazione e l'autorizzazione della richiesta (se configurata/necessaria). AWS API Gateway (API GW da qui in poi) supporta diversi tipi di integrazioni API:

1. Integrazione HTTP (HTTP/HTTP_PROXY): è il tipo di integrazione più semplice, API GW inoltra le richieste a un server HTTP. Sono disponibili due modalità: proxy (HTTP_PROXY) e non proxy (HTTP). Essendo un'integrazione HTTP generale, può essere utilizzata per integrare le API esistenti sia su AWS EC2/ECS che su server locali esistenti.
2. Integrazione Lambda (AWS_PROXY): viene richiamata una funzione lambda per gestire la richiesta. L'intera richiesta viene inoltrata a Lambda come oggetto JSON.
3. Altro servizio AWS (AWS): la richiesta viene inoltrata a un altro servizio AWS (es. kinesis, SQS, Lambda) dopo essere stata trasformata per aderire alle specifiche API del servizio AWS (se è necessaria la modifica). È possibile specificare il ruolo che eseguirà la richiesta alle API del servizio. La maggior parte dei servizi AWS è supportata.

4. Integrazione MOCK: restituisce risposte statiche o risposte a cui è stata implementata una logica minima utilizzando modelli di velocità (vedi https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-mock-integration.html)

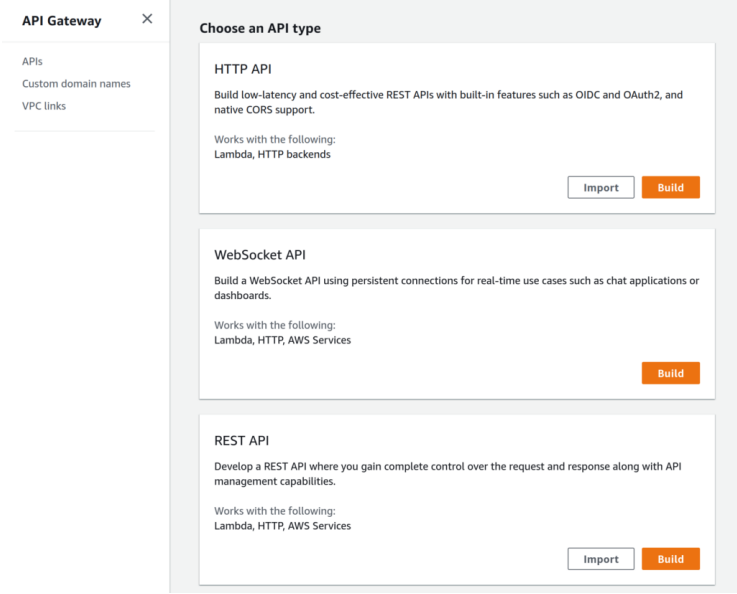
Un caso d'uso molto comune di API gateway nel mondo reale, in cui molte delle integrazioni descritte in precedenza sono effettivamente utilizzate, consiste nella migrazione di un'applicazione legacy da un'infrastruttura monolitica a una moderna applicazione modularizzata basata su microservizi. In questo caso d'uso si avranno tipicamente molte API ancora ospitate sull'infrastruttura monolitica e altre API già migrate su AWS Lambda (con l'integrazione AWS_PROXY) e/o AWS ECS (Fargate) esposte tramite un bilanciatore di carico.

API gateway esporrà un singolo endpoint che può essere mappato su un Fully Qualified Domain Name (FQDN) a scelta utilizzando la funzionalità Custom Domain API Gateway.

Proxy di un'API esistente

Concentriamoci ora sull'integrazione con la parte monolitica legacy dell'applicazione in fase di migrazione e supponiamo, per semplicità, che le API esistenti siano pubblicamente raggiungibili, anche se con qualche sforzo in più le API esistenti esposte privatamente sia on-prem che su AWS possono anche essere proxate (https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-develop-integrations-private.html). Nell'esempio fornito qui utilizzeremo la regione AWS Irlanda (eu-west-1).

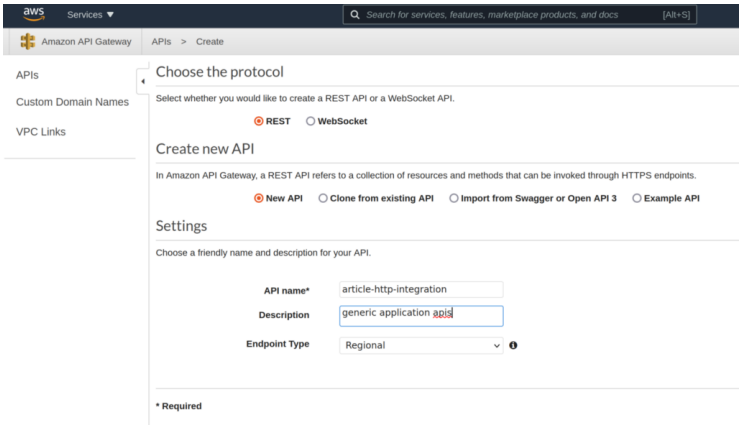
Iniziamo creando una nuova API nella console web AWS API GW: apri il tuo account AWS, vai al servizio API Gateway (https://eu-west-1.console.aws.amazon.com/apigateway/main/apis?region=eu-west-1) e creiamo una nuova API, ci viene presentata la seguente procedura guidata:



I vari tipi di API che puoi selezionare hanno funzionalità diverse: le Websocket API sono utilizzate per applicazioni full-duplex in tempo reale, sicuramente non il nostro caso, mentre le API HTTP e REST sono entrambe scelte legittime. Le API HTTP, tuttavia, implementano solo un sottoinsieme delle funzionalità disponibili con le API REST. Ad esempio non sono implementate l'autenticazione tramite il servizio Cognito, l'autenticazione Iam, le integrazioni generiche di servizi AWS e le trasformazioni delle richieste tramite velocity template.

In questo esempio, per essere il più generici possibile, useremo le API REST ma si tenga presente che le richieste API HTTP costano meno del 30% delle richieste API REST, quindi sono spesso la scelta più conveniente.

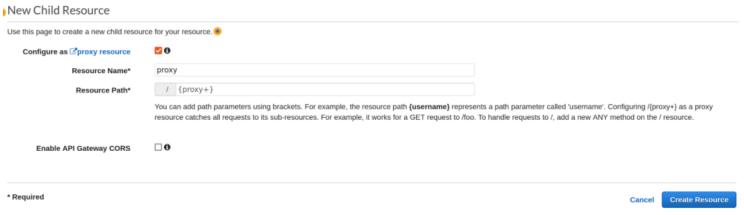
Quindi facciamo clic su Crea su API REST, quindi scegli Nuova API e imposta nome e descrizione.



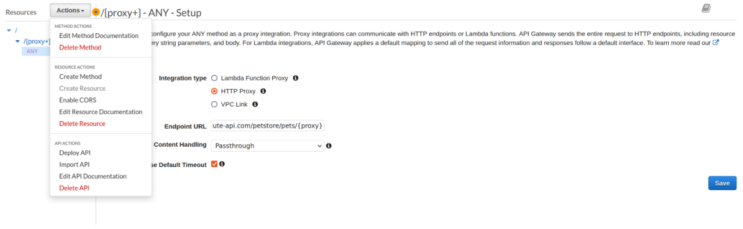
Il tipo di endpoint definisce dove sono ubicati gli endpoint: un endpoint edge crea una CDN (distribuzione Cloudfront) completamente gestita per distribuire gli endpoint in tutte le regioni AWS, quindi più vicini ai clienti. Il rovescio della medaglia è che diverse operazioni (es. associazioni di domini personalizzati, implementazioni) sono più lente, mentre il prezzo è lo stesso. In questo caso scegliamo l'endpoint Regionale.

Fare clic su Crea API per finalizzare la procedura guidata di creazione.

Ora è finalmente il momento di integrare la nostra API legacy. La cosa più semplice che puoi fare è configurare un semplice proxy utilizzando l'integrazione HTTP_PROXY. In questo esempio, l'integrazione HTTP_PROXY ha come target un'API simulata di Petstore e distribuita da AWS su http://petstore-demo-endpoint.execute-api.com.

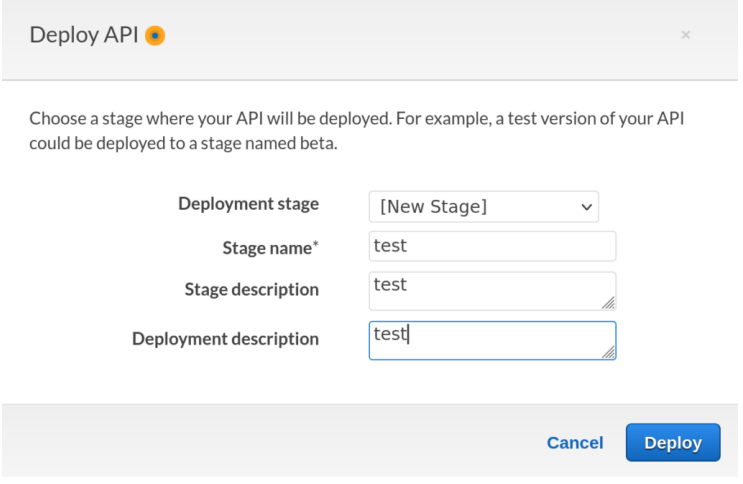


Una volta creata la risorsa, questa può essere visualizzata nella barra di sinistra come /{proxy+}. Puoi selezionarla e in Azioni Risorsa si può creare prima un Metodo di tipo ANY con le seguenti impostazioni:

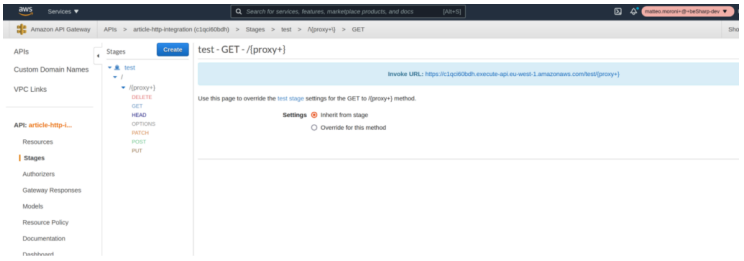


dove l'URL dell'endpoint è http://petstore-demo-endpoint.execute-api.com.

Ora è possibile Deployare l'endpoint in modo che sia raggiungibile da internet: vai su Actions, Deploy API, nel form crea un nuovo stage e imposta la descrizione come desideri. La denominazione non è importante in questa sezione.



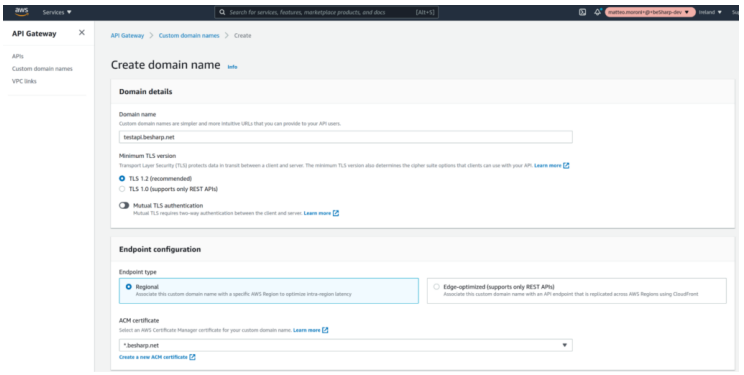
Al termine del Deployment, puoi trovare il tuo nuovo URL API nella sezione Stages.



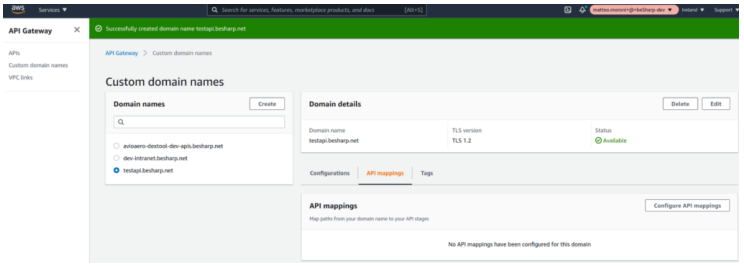
e se lo si interroga con curl (curl -vv https://c1qci60bdh.execute-api.eu-west-1.amazonaws.com/test/petstore/pets):

```
[
  {
        "id": 1,
        "type": "cane",
        "prezzo": 249,99
  },
  {
        "id": 2,
        "tipo": "gatto",
        "prezzo": 124,99
  },
  {
        "id": 3,
        "tipo": "pesce",
        "prezzo ": 0.99
  }
]
```

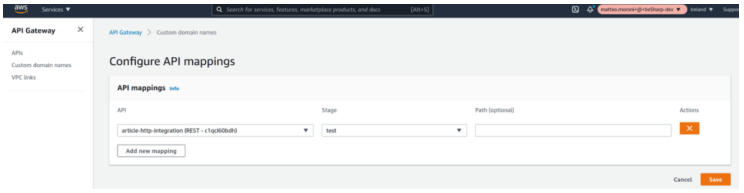
Fantastico, abbiamo configurato un proxy completamente serverless per un'API! Ora possiamo semplicemente impostare il nostro nome di dominio utilizzando Custom Domain Names. Vai a Custom Domain Names nella scheda a sinistra e crea un nuovo Custom Domain. Se selezioni Edge-optimized come tipo di endpoint, la latenza delle API sarà inferiore perché l'endpoint verrà distribuito nella edge location di una distribuzione Cloudfront gestita, tuttavia dovrai attendere diversi minuti per la creazione.



Dopo aver creato il nome di dominio (istantaneo se hai selezionato un endpoint regionale) ora devi impostare l'API Mapping in modo che sia possibile raggiungere l'API appena creata:



Per creare la mappatura fai clic su Configure API mapping nella scheda API mapping del Dominio e configurarlo come segue e Salva:

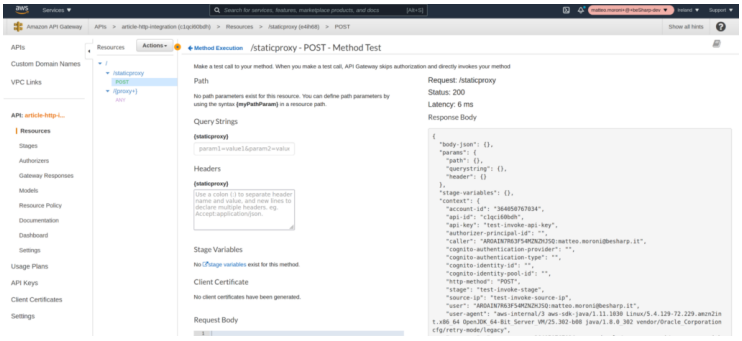


Ora è necessario creare un record DNS CNAME nel tuo provider DNS (se sei tutto su AWS, probabilmente è AWS Route 53) usando come valore del record testapi.besharp.net CNAME d-b42ojs42rb.execute-api.eu-west -1.amazonaws.com

Una volta che il DNS si è propagato, si sarà in grado di raggiungere l'API proxy tramite il tuo dominio personalizzato.

Ora si potrebbe voler configurare altri PATH in modo da poter integrare altre API con il metodo HTTP desiderato.

In questo esempio abbiamo creato un integrazione di tipo mock (https://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-mock-integration.html)



Dopo il deployment se si esegue una richiesta POST su questo nuovo metodo API simulato, il nuovo endpoint mocked risponderà. Va notato che per un determinato percorso uri l'integrazione {proxy+} viene eseguita solo se non viene trovata alcuna altra risorsa e METODO HTTP corrispondente alla richiesta effettuata.

Il passaggio successivo è modificare una richiesta in ingresso utilizzando il gateway API per integrare varie API legacy con una definizione API eterogenea in un endpoint API omogeneo esposto tramite API Gateway. A tale scopo è possibile i Mapping Templates nelle integrazioni Api Gateway sia per modificare la richiesta che la risposta. Questo argomento molto ampio ed interessante verrà spiegato in dettaglio in un articolo successivo.

Siamo giunti alla conclusione, restate sintonizzati sul nostro blog e se avete domande o trovate questo argomento interessante e desiderate interagire con noi non esitate a scrivere nei commenti.

Matteo Moroni

DevOps e Solution Architect di beSharp, mi occupo di sviluppare soluzioni Saas, Data Analysis, HPC e di progettare architetture non convenzionali a complessità divergente. Appassionato di informatica e fisica, da sempre lavoro nella prima e ho un PhD nella seconda. Parlare di tutto ciò che è tecnico e nerd mi rende felice!
