

ALMA MATER STUDIORUM
UNIVERSITÀ DI BOLOGNA
CAMPUS DI CESENA

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea in Ingegneria e Scienze Informatiche

SERVERLESS COMPUTING IL CASO GOOGLE CLOUD FUNCTIONS

Elaborato in:
Virtualizzazione ed Integrazione di Sistemi

Relatore:
Prof.
Vittorio Ghini

Presentata da:
Yuri Bernardini

Sessione III
Anno Accademico 2021-22

*Non smettere mai di lottare
e di sognare...*

Sommario

Il serverless è un paradigma del cloud computing al giorno d'oggi sempre più diffuso; si basa sulla scrittura di funzioni stateless in quanto le attività relative alla loro manutenzione e scalabilità fanno capo al provider dei servizi cloud. Lo sviluppatore deve quindi solamente concentrarsi sulla creazione del prodotto.

Questo lavoro si apre con un'analisi del cloud computing introducendo i principali modelli di applicazione, passando dal parlare di servizi cloud, con le varie sottocategorie e i relativi utilizzi per poi arrivare a parlare di serverless. Si è scelto di focalizzarsi sulle risorse che vengono offerte dalla famosissima piattaforma Google con la suite: Google Cloud Platform; Il progetto si apre con le origini e l'analisi dei relativi servizi che l'azienda americana offre ai suoi clienti.

Il focus dell'intero progetto sono le Google Cloud Functions, una nuova offerta serverless della compagnia, di recente sviluppo e in continuo aggiornamento. Partiremo dalle prime release, analizzeremo l'ambiente di sviluppo, i casi d'uso, vantaggi, svantaggi, parleremo poi di portabilità e verranno mostrati alcuni esempi del loro utilizzo.

Indice

Sommario	i
1 Introduzione	1
1.1 Introduzione al Serverless Computing	1
1.2 Modelli Cloud	2
1.2.1 I Cloud Pubblici	2
1.2.2 I Cloud Privati	3
1.2.3 I Cloud Ibrido	4
1.2.4 I Multi Cloud	5
1.3 Servizi Cloud	6
1.3.1 IaaS - Infrastructure as a Service	7
1.3.2 CaaS - Container as a Service	8
1.3.3 PaaS - Platform as a Service	9
1.3.4 FaaS - Function as a Service	10
1.3.5 SaaS - Software as a Service	11
1.4 Un altro tipo di Cloud: Serverless	13
2 Google Cloud Platform	15
2.1 Origini di Google Cloud Platform	15
2.2 Strumenti di Google Cloud Platform	17
2.3 Perché scegliere Google	19
2.4 Orientarsi verso soluzioni Serverless	20
2.5 GCP soluzioni Serverless	21
2.5.1 Cloud Run	21

2.5.2	App Engine	22
2.5.3	Firebase	22
2.5.4	Cloud Functions	22
3	Google Cloud Functions	25
3.1	Funzionalità principali delle GCF	25
3.1.1	Esperienza di sviluppo semplice e veloce	25
3.1.2	Scalabilità automatica	26
3.1.3	Modello di pagamento a consumo	26
3.1.4	Nessun vincolo al fornitore grazie ad una tecnologia aperta	26
3.2	Casi d'uso Cloud Functions	27
3.2.1	Integrazione con servizi e API di terze parti	27
3.2.2	Backend serverless per dispositivi mobili	27
3.2.3	Backend IoT Serverless	28
3.2.4	Elaborazione file in tempo reale	29
3.2.5	Elaborazione flussi in tempo reale	29
3.2.6	Assistenti Virtuali ed esperienze di conversazione	30
3.2.7	Analisi di video e immagini	30
3.2.8	Analisi dei sentiment	31
3.3	Release ed ultime novità	32
3.3.1	Functions Frameworks	32
4	Ambiente e sviluppo Google Cloud Function	33
4.1	Requisiti	33
4.2	Home Page	34
4.3	Passi necessari per lo sviluppo di una funzione	35
4.3.1	Abilitazione API	35
4.3.2	Creazione e sviluppo	36
4.3.3	Risultato	41
4.4	Aggiunta parametri con metodo GET	42
4.4.1	Modifica codice funzione	42

5	Scrivere funzioni Cloud	45
5.1	Attivatori HTTP	45
5.2	Scrivere funzioni HTTP	46
5.2.1	Utilizzare i Function Framework	47
5.3	Accessibilità	49
5.3.1	Chiamate in locale	49
5.3.2	Chiamate dirette	49
5.3.3	Deployment	50
5.3.4	Attenzione ai CORS	51
5.4	Analisi portabilità	52
5.4.1	La scelta del linguaggio	52
5.4.2	Conclusione con vantaggi e svantaggi	52
6	Esempio di interazione con altri servizi Google	55
6.1	Architettura	55
6.2	Creare il bucket Cloud Storage	56
6.3	Creare la Cloud Function	57
6.3.1	Codifica della funzione	59
6.3.2	Test della funzione	62
6.4	Pianificazione Cloud Function	63
6.5	Resoconto e possibili implementazioni	64
	Conclusioni	67
	Bibliografia	71
	Ringraziamenti	73

Elenco delle figure

1.1	Tipologie di implementazione modelli cloud	2
1.2	Modelli di servizi cloud computing	7
2.1	Rete Google Cloud	16
2.2	Catalogo di Google Cloud Platform	17
2.3	Caratteristiche di Google Cloud Platform	19
2.4	Soluzioni Serverless di Google Cloud Platform	21
3.1	Schema CF con servizi e API terze	27
3.2	Schema CF per Firebase	28
3.3	Schema CF backend IoT serverless	28
3.4	Schema CF per elaborazione in tempo reale	29
3.5	Schema CF con flussi in tempo reale	29
3.6	Schema CF con assistenti virtuali	30
3.7	Schema CF per l'analisi di video e immagini	31
3.8	Schema CF per l'analisi di sentiment	31
4.1	Home page Google Cloud Platform	34
4.2	Abilitazione API	35
4.3	Configurazione della funzione	36
4.4	Configurazione runtime	38
4.5	Funzione di default	39
4.6	Codice della funzione	40
4.7	Status della funzione	41

4.8	Risultato della funzione	41
4.9	Codice con metodo GET	42
4.10	Output funzione GET	43
5.1	Chiamata funzione HTTP	46
5.2	Esempio funzione in Python	46
5.3	Schema deployment funzione	50
6.1	Schema architettura	55
6.2	Creazione bucket su Cloud Storage	57
6.3	Creazione Cloud Function Pub/Sub	58
6.4	Risultato deploy	62
6.5	Test funzione	62
6.6	Cloud Scheduler	63
6.7	Visualizzazione file nel bucket	64

Capitolo 1

Introduzione

In questo capitolo saranno introdotti i concetti di Serverless Computing, dicitura appartenente al mondo cloud, di cui verrà analizzata la struttura e le possibili tecnologie di implementazione che ne caratterizzano i possibili utilizzi.

1.1 Introduzione al Serverless Computing

Con la dicitura Serverless Computing si indicano una serie di modelli cloud native che consentono agli utenti di creare e gestire le proprie applicazioni senza focalizzarsi sulla gestione lato server.

Letteralmente il termine Serverless può essere tradotto in “senza server”, è importante però sottolineare che non ci troviamo in un modello in cui non è presente un server (fisico o virtuale) che gestisca e risponda alle richieste di uno o più client, come ad esempio in un modello informatico ad architettura di rete Peer To Peer; ma l'utilizzo di applicazioni serverless fa in modo che i clienti possano focalizzarsi solamente sull'implementazione e sulla logica applicativa, mentre i server, sì, vengono utilizzati, ma la loro logica e l'intera gestione avviene in automatico dai vari provider di servizi cloud che si occupano di ogni attività di routine, di manutenzione e di scalabilità, adattandosi automaticamente alle forme e ai carichi di lavoro necessari.

1.2 Modelli Cloud

Il mondo Serverless si basa sul concetto generale di cloud computing. Nel cloud computing le risorse vengono messe a disposizione in base a vari modelli. Esistono principalmente quattro modelli: Cloud pubblici, Cloud privati, Cloud Ibridi e Multicloud.

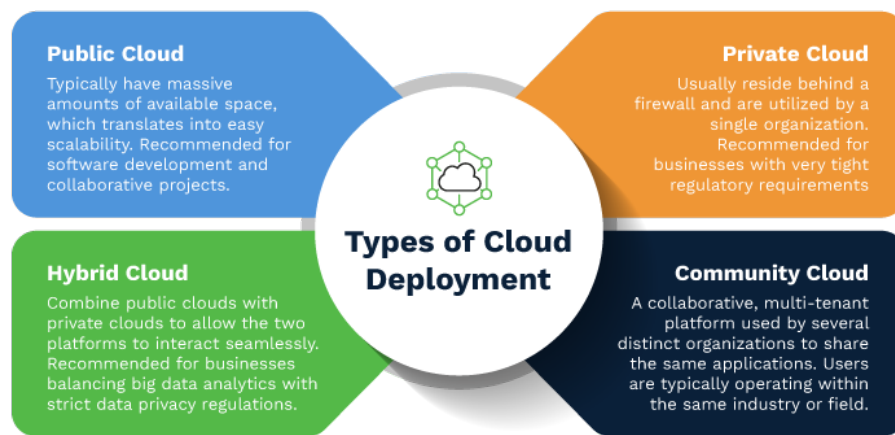


Figura 1.1: Tipologie implementazione modelli cloud

1.2.1 I Cloud Pubblici

I Cloud Pubblici sono ambienti cloud basati su una infrastruttura IT che non appartiene all'utente finale, i provider più popolari sono Amazon Web Services, Google Cloud Platform, Microsoft Azure, IBM, Alibaba Cloud. Questi IT cloud hanno in genere enormi quantità di spazio disponibile, il che comporta una facile scalabilità.

Un cloud pubblico è spesso consigliato per sviluppare software collaborativi, ottenendo anche applicazioni portabili in modo che un progetto testato in un cloud pubblico possa essere poi spostato in un cloud privato per la produzione.

Pro e Contro dei servizi cloud pubblici

Il vantaggio principale di un cloud pubblico è la sua versatilità e la formula “pay as you go” che consente di fornire più capacità on demand.

Questo non solo rende il sistema scalabile, permette anche una facile gestione dei costi in quanto questi sono legati alla crescita dell’applicativo, dell’hardware e alla sua manutenzione.

Inoltre, in casi d’emergenza permette di ottenere facilmente punti di ripristino e piani per la protezione dei dati, che risulterebbe dispendioso in ambiti di tempo e denaro nel caso in cui ognuno debba pensare di implementare i propri.

In negativo invece abbiamo che il provider cloud ha il pieno controllo del sistema operativo e dell’infrastruttura essenziale. L’utente deve accettare i termini e le condizioni stabilite e potrebbero avere difficoltà per un futuro cambio di provider. Se l’host dovesse cessare alcuni servizi, il cliente può essere costretto ad una nuova implementazione o si vede costretto a dover apportare modifiche molto costose.

1.2.2 I Cloud Privati

Solitamente i cloud privati risiedono dentro un firewall e vengono utilizzati solamente da una singola organizzazione. Un cloud completamente On-Premise può essere la soluzione preferita da aziende con requisiti normativi molto rigidi. I servizi cloud privati stanno di recente guadagnando popolarità.

Solamente gli utenti autorizzati possono accedere, utilizzare e archiviare i dati nel cloud privato, inoltre, come per i cloud pubblici, vi si può accedere da qualsiasi luogo, ma la differenza sta nel fatto che nessun altro oltre al proprietario può accedere o utilizzare tali risorse.

Pro e Contro dei servizi cloud privati

Il controllo dell'ambiente è essenziale, perciò le soluzioni cloud private offrono sicurezza e controlli aggiuntivi. Questo rende molto più semplice la limitazione agli accessi delle risorse preziose e permette all'organizzazione un facile spostamento di dati in caso di bisogno.

Inoltre, poichè si tratta di un servizio privato non è controllato da un fornitore esterno, non vi è il rischio di improvvisi cambiamenti che danneggino l'infrastruttura. Si può ottenere il supporto tecnico offerto dal partner e il loro piano di ripristino di emergenza.

Tuttavia, tutti questi vantaggi associati al cloud privato, come è facilmente deducibile hanno un elevato costo. La società proprietaria del cloud è responsabile sia del software che dell'infrastruttura.

Questi sistemi non hanno la versabilità dei cloud pubblici. Possono essere espansi solamente aggiungendo più hardware e rendono difficile veloci operazioni di scalabilità in caso di necessità.

1.2.3 I Cloud Ibrido

In breve, i cloud ibridi combinano i cloud pubblici con cloud privati. Sono progettati per garantire alle due piattaforme di interagire senza problemi. È la soluzione perfetta per quelle aziende che hanno necessità di entrambe le precedenti soluzioni, in base al settore e alle dimensioni.

Comunemente sono usati due tipi di architettura cloud ibrida:

- Il primo detto *Cloudbursting* utilizza un cloud privato come principale, in cui archivia i dati e ospita le applicazioni in un ambiente sicuro. Quando le richieste di servizi aumentano, non sempre il cloud privato è in grado di tenere il passo, quindi è qui che entra in gioco il cloud pubblico, escludendo così all'azienda la gestione in caso di aumento del traffico.
- Il secondo modello esegue anche lui le principali azioni in un cloud privato, ma esternalizza applicazioni non critiche ad un provider cloud

pubblico. Questo sistema è comune a organizzazioni che devono accedere a strumenti di sviluppo specializzati (come Adobe Creative Cloud) o software di produttività di base (come Microsoft Office 365). L'architettura multi cloud viene spesso implementata qui, incorporando più fornitori di servizi cloud, così da soddisfare una varietà di esigenze uniche.

Pro e Contro dei servizi cloud ibridi

Il vantaggio principale di un modello cloud ibrido è la sua capacità di fornire potenza di calcolo scalabile di un cloud pubblico con la sicurezza e il controllo di un cloud privato. I dati possono essere archiviati in tutta sicurezza dietro un firewall e i protocolli di crittografia di un cloud privato. Quindi spostato in modo sicuro in un ambiente cloud pubblico quando necessario, ciò è particolarmente utile nell'analisi dei big data, quando settori come l'assistenza sanitaria devono aderire a norme rigide sulla privacy utilizzando sofisticati algoritmi basati sull'intelligenza artificiale (AI) per ricevere informazioni utili da enormi masse di dati non strutturati.

A causa della combinazione dei due modelli di cloud, può essere conveniente, anche se la spesa iniziale per un cloud privato dovrebbe essere elevata. Questi costi possono essere recuperati in seguito, quando la scalabilità e la crescita possono essere gestite sul cloud pubblico al suo bisogno.

Infine, è sempre consigliabile avere una grande esperienza di questa tipologia di modello prima di deciderne l'utilizzo in quanto non è facile mettere in comunicazione un cloud privato con uno pubblico se uno è alle prime armi.

1.2.4 I Multi Cloud

I multi cloud sono conosciuti anche come community cloud, sebbene non siano utilizzati come gli altri tre, sono conosciuti come una piattaforma collaborativa multi-tenant utilizzata da diverse organizzazioni distinte per condividere le stesse applicazioni. Gli utenti in genere operano all'interno dello

stesso settore o campo e condividono preoccupazioni comuni in termini di sicurezza, conformità e prestazioni.

In sostanza un *community cloud* è un cloud privato che funziona in modo simile ad un cloud pubblico. La piattaforma è gestita in modo privato, in un data center o in locale.

Gli utenti autorizzati vengono quindi segmentati all'interno di tale ambiente. Sono implementazioni comunemente usate da agenzie governative, organizzazioni sanitarie, società di servizi finanziari e altre comunità professionali.

Pro e Contro dei servizi community cloud

Come per gli altri modelli, la scalabilità è un vantaggio e a un costo che può essere confuso tra le organizzazioni. Grazie alle esigenze comuni, in termini di conformità di leggi le organizzazioni che optano per questo servizio possono stare tranquille. Il processo decisionale relativo alle modifiche del sistema è collaborativo, quindi ogni decisione viene presa nell'interesse comune del gruppo.

Anche per lo spazio condiviso, il sistema rimane altamente flessibile: ogni proprietario può impostare i suoi controlli di sicurezza all'accesso e consente al sistema di adattarsi alle esigenze, spostando le risorse se necessario.

Sebbene tutti questi siano punti di forza, sfortunatamente hanno anche un aspetto negativo. L'archiviazione condivisa e la larghezza di banda possono creare problemi con la priorità e le prestazioni man mano che i server si adattano alle richieste. Inoltre, avere uno spazio di archiviazione condiviso, può mettere in pericolo l'incolumità e la sicurezza dei dati.

1.3 Servizi Cloud

I servizi cloud sono costituiti da infrastrutture, software e piattaforme in host presso provider esterni, naturalmente a disposizione dell'utente tramite Internet. Possono essere suddivisi in diverse categorie di soluzioni, ognuna delle quali offre flussi di dati utente dai client front end ai sistemi del provider

di servizi cloud su internet, e viceversa, ma ognuna con tipologie di servizi diverse. Di seguito le principali categorie.

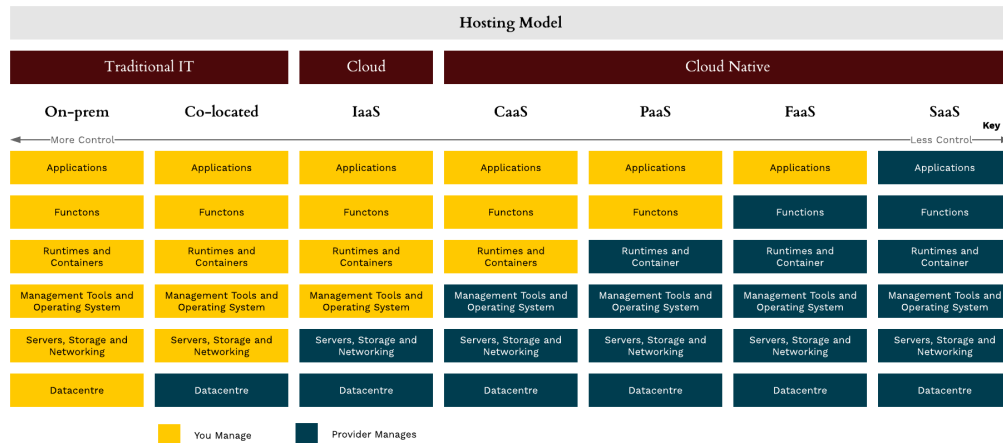


Figura 1.2: Modelli di servizi cloud computing

1.3.1 IaaS - Infrastructure as a Service

In questa tipologia di infrastruttura il provider cloud gestisce l'infrastruttura ovvero i server fisici, la rete, la virtualizzazione e lo spazio di memoria per conto del cliente, il tutto naturalmente attraverso una connessione internet. Solitamente si tratta di una tipologia a noleggio, tramite un API o una dashboard si gestiscono aspetti quali sistema operativo, app e middleware. Il provider si occupa invece di tutto l'hardware, della connettività di rete, dei dischi rigidi, dello storage dati e dei server, assumendosi la responsabilità di gestire disservizi, riparazioni e problemi hardware. Questo è il tipico modello di deployment dei provider storage su cloud.

IaaS è un modello di business che contrariamente al classico acquisto di infrastrutture, permette di affittarlo solamente quando necessario, cosiddetto "su richiesta". Un ottimo esempio di infrastruttura scalabile, a seconda delle esigenze di elaborazione e storage. Permette quindi alle aziende di risparmiare notevolmente sui costi di acquisto e manutenzione legati all'hardware.

Dal momento che i dati si trovano sul cloud, non vi è alcun singolo punto di errore. Abilita la virtualizzazione di attività amministrative, liberando il tempo per altre attività e consentendo semplici test del software su un'ampia varietà di piattaforme. Notevole anche la capacità di assorbimento dei picchi di carico.

È inoltre possibile visualizzare provider IaaS come servizi online che utilizzano determinate tecnologie di amministrazione cloud per gestire la creazione di macchine virtuali, selezionare l'hypervisor, allocare il volume di archiviazione e fornire informazioni sull'utilizzo a fini di fatturazione.

La tecnologia di orchestrazione del cloud utilizzata dalla maggior parte dei cloud pubblici è proprietaria, ma sono disponibili più soluzioni open source (come OpenStack, Apache CloudStack ecc.) anche per i cloud nativi.

1.3.2 CaaS - Container as a Service

Containers as a service è una categoria di servizi cloud in cui il fornitore offre ai clienti la possibilità di gestire e distribuire applicazioni e cluster containerizzati.

CaaS è talvolta visto come un sottotipo speciale del modello di erogazione del servizio cloud Infrastructure as a service (IaaS), ma in cui la merce principale sono i container piuttosto che l'hardware fisico e le macchine virtuali.

Il concetto di Container

Un container funziona essenzialmente come un'alternativa all'approccio di virtualizzazione tradizionale, dove invece di virtualizzare lo stack hardware utilizzando macchine virtuali, i container virtualizzano a livello del sistema operativo.

Di conseguenza, i container vengono eseguiti in modo molto più efficiente delle macchine virtuali. Offrono quindi prestazioni più elevate rispetto alla virtualizzazione / hypervisor, perché non vi è alcun sovraccarico dell'hypervisor. Usano meno risorse e una frazione di memoria rispetto alle macchine virtuali che devono avviare un intero sistema operativo ogni volta

che vengono inizializzate. Inoltre, la capacità del contenitore si ridimensiona automaticamente in base al carico di elaborazione, eliminando il problema dell'over-provisioning.

Un container è eseguito in partizioni isolate di un singolo kernel Linux in esecuzione direttamente sull'hardware fisico. I cgroup Linux (gruppi di controllo) e gli spazi dei nomi sono le tecnologie del kernel Linux sottostanti utilizzate per isolare, proteggere e gestire i contenitori.

I container esistono dalla fine degli anni '80, ma nessuna organizzazione ha fatto più di Google per sviluppare e perfezionare la pratica della gestione dei container. Spinti dalla necessità di ridurre i costi di sviluppo del software e il time-to-value, gli ingegneri di Google hanno creato un'aggiunta al kernel Linux nota come cgroups che è stata utilizzata per creare contenitori che avrebbero alimentato tutte le applicazioni di Google. Questi contenitori fungono da ambienti di esecuzione isolati per singole applicazioni utilizzando un sistema operativo semplificato.

Quindi CaaS consente agli utenti di distribuire e gestire applicazioni containerizzate offrendo prodotti specifici che forniscono un modo semplice per eseguire distribuzioni a container singolo, in particolare per eseguire microservizi semplici o fornire una piattaforma di orchestrazione container gestita come Kubernetes per eseguire distribuzioni multi container più complesse. In entrambi i casi, il runtime e altri servizi relativi ai contenitori sono forniti dal provider CaaS, così che l'utente non debba gestire la configurazione dell'hardware.

1.3.3 PaaS - Platform as a Service

PaaS consente di scrivere applicazioni utilizzando linguaggi di programmazione specifici, librerie, framework supportati dal provider PaaS. Il fornitore PaaS gestisce la rete, i server, i sistemi operativi, l'archiviazione, i runtime della lingua, le librerie, i framework. In modo che l'utente possa concentrarsi solamente sullo sviluppo e deployment della propria applicazione.

L'utilizzo della piattaforma PaaS offre innumerevoli vantaggi agli sviluppatori. Prima di tutto è da sottolineare che lo sviluppo di applicazioni è essenzialmente più facile e veloce. Inoltre, le prestazioni sono scalabili, cioè è possibile ampliare o diminuire in modo flessibile le capacità richieste a seconda delle proprie esigenze, come accade con gli altri servizi cloud. Infine, vi è anche un risparmio economico dato che non subentrano costi per la realizzazione dell'infrastruttura, di manutenzioni, di aggiornamenti o di acquisto di nuove licenze software.

Il fatto che sia il fornitore a occuparsi dell'infrastruttura non è sempre un vantaggio. Infatti, non si ha alcun controllo e non è possibile implementare in autonomia feature. Inoltre, sono utilizzabili solo i linguaggi di programmazione e i tool messi a disposizione dal fornitore.

Un altro fattore da tenere in considerazione al momento della scelta di una Platform as a Service è che con il vostro progetto siete più o meno vincolati all'ambiente di sviluppo scelto. È possibile far migrare un progetto piccolo, ma nel caso di applicazioni più grandi il codice non si può sempre riportare per intero durante il trasferimento su un'altra piattaforma: deve essere invece riscritto da capo o almeno in parte.

Non è da escludere, anche se un fatto raro, che il fornitore scelto decida di sospendere il servizio o fallisca. Per ridurre l'eventualità che si presenti un rischio simile sarebbe meglio optare per un servizio affermato già da tempo sul mercato.

1.3.4 FaaS - Function as a Service

FaaS in modo abbastanza simile a PaaS consente di concentrarsi principalmente sulla logica di business; tuttavia, ha un caso d'uso piuttosto diverso. FaaS consente di scrivere una funzione in uno dei linguaggi / framework di programmazione supportati.

In questo modello il cliente implementa solo la logica di business nelle funzioni. Non è presente un processo server sempre in esecuzione, ma vi è una

chiamata di funzione solo in caso di attivazione di un evento trigger, ad esempio una chiamata HTTP.

Per la persistenza, è necessario utilizzare un server di database esterno o un file system di rete. FaaS consente una semplice scalabilità, poiché le funzioni stateless possono essere ridimensionate orizzontalmente in modo banale. Si paga per chiamata di funzione (memoria e vCPU allocate), se la funzione non viene mai chiamata, nulla è addebitato.

Hardware, sistemi operativi, web server ed altri middleware sono gestiti completamente dal provider cloud, permettendo allo sviluppatore di focalizzarsi unicamente sullo sviluppo del codice applicativo. Più codice, meno infrastruttura: È possibile dividere i server in funzioni, le quali sono scalate automaticamente ed indipendentemente dalla piattaforma, evitando di dover gestire le risorse allocate.

Si tratta di una soluzione affidabile in quanto le funzioni vengono replicate automaticamente tra i diversi data center offerti dal provider cloud, garantendone disponibilità e tolleranza in caso di guasti.

1.3.5 SaaS - Software as a Service

SaaS, acronimo inglese per Software as a Service indica un modello piuttosto recente di distribuzione del software rispetto ai tradizionali. È un modello di business che non vende software a un utente ma lo rende semplicemente disponibile come servizio, tramite internet. Si tratta di un modello in cui il software è concesso in licenza su abbonamento ed è ospitato centralmente dal fornitore SaaS. La maggioranza delle soluzioni SaaS si basa su un'architettura multi-tenant. Con questo modello, per tutti i clienti ("tenant") viene utilizzata un'unica versione dell'applicazione, con un'unica configurazione (hardware, rete, sistema operativo).

Le applicazioni SaaS vengono spesso aggiornate più frequentemente rispetto ai software tradizionali, in molti casi su base settimanale o mensile, tutto questo poiché l'applicazione è ospitata centralmente, quindi un aggiornamen-

to viene deciso ed eseguito dal provider, non dai clienti. L'applicazione ha una sola configurazione, rendendo più veloci test di sviluppo.

Software as a Service si è fatto strada dagli inizi del 2000; il successo di molte aziende come Netflix, Dropbox, Slack, Spotify, Microsoft Office 365 e Google Apps ha reso l'industria SaaS, un modello distintivo dell'economia digitale moderna.

Per la struttura di pagamento il modello il modello di prezzo per le applicazioni SaaS è in genere un canone mensile o annuale; mentre in alcuni casi potrebbe anche essere gratuito .

Dov'è il confine tra Cloud e SaaS?

Il modello SaaS e la tecnologia cloud non sono naturalmente la stessa cosa, ma sono strettamente correlati tra loro. Con cloud si intende, in generale, la complessa tecnologia alla base dell'infrastruttura software: “un insieme di computer, server e database collegati tra loro in modo che gli utenti possano accederne e usufruire delle loro risorse”. Con SaaS ci si riferisce specificamente alle applicazioni software fornite tramite il cloud. Grazie alla diffusa crescita dell'accessibilità del cloud, per gli sviluppatori SaaS è più semplice, rapido e meno costoso per implementare applicazioni rispetto allo sviluppo dei software tradizionali on-premise.

Gli svantaggi sono principalmente legati alla sicurezza dei dati che resta ancora controversa. Nonostante venga assicurata la massima privacy e sicurezza, i dati sono memorizzati “presso il provider”. Data leak, attacchi hacker e altri “incidenti” sono fuori dal controllo dell'utente. Si necessita di una connessione Internet veloce e costante, eventuali problemi alla connessione potrebbero interrompere il tuo lavoro. Molti provider offrono anche una modalità offline. Vi possono essere anche problemi di compatibilità con sistemi operativi e browser utilizzati dagli utenti e quelli dei server.

1.4 Un altro tipo di Cloud: Serverless

Il paradigma serverless si era concretizzato inizialmente come Function-as-a-Service (FaaS), ma oggi si è arrivati ad uno step successivo.

Solitamente quando si parla di serverless ci si riferisce al modello SaaS (Software-as-a-Service) per intendere un'applicazione che è interamente serverless. In questo caso non si ha più una sola funzione, ma un insieme di funzioni, servizi e dipendenze.

Il serverless computing sta diventando sempre più quella che si pensa sarà la prossima rivoluzione del cloud computing.

Capitolo 2

Google Cloud Platform

Ad oggi le principali provider serverless in ordine di utilizzo sono:

1. AWS Amazon Web Services
2. Google Cloud Platform
3. Microsoft Azure
4. Alibaba Cloud
5. IBM

Di seguito si analizza l'ambito fornito da Google.

2.1 Origini di Google Cloud Platform

Google Cloud Platform (GCP) nasce più tardi rispetto agli altri suoi concorrenti; infatti, nell'aprile 2008 Google ha annunciato App Engine, una piattaforma per lo sviluppo e l'hosting di applicazioni web nei data center gestiti da Google, che è stato il primo servizio di cloud computing dell'azienda. Inizialmente venne fornito solamente a 10.000 sviluppatori, da utilizzare come tester; a maggio dello stesso anno quel numero era diventato 75.000 sviluppatori attivi; Infine, Google ha poi aperto a tutti l'utilizzo della piattaforma.

Negli anni successivi Google ha poi rilasciato un flusso di prodotti e nuove funzionalità: servizi come Google Cloud Storage nel 2010, Compute Engine nel 2013, Cloud SQL nel 2014 e Kubernetes Engine nel 2015; Google ha creato una suite diversificata e completa per lo sviluppo di soluzioni cloud-native.

Durante lo stesso periodo si è poi espanso anche in altre aree come la gestione delle infrastrutture, l'analisi dei dati, l'IoT e il machine learning.

Entro fine del 2017, Google aveva stabilito data center in 39 zone di 13 paesi. Queste regioni sono sin dalla prima rete in fibra globale multilivello di un importante provider di cloud pubblico. Con oltre 100 punti di presenza, Google Cloud offre ai suoi utenti una bassa latenza, indipendentemente da dove si trovino nel mondo.

Questa rete privata in fibra ottica è la spina dorsale della presenza globale di Google, messa a disposizione dei clienti GCP.

Ad oggi, nel 2022 Google Cloud ha raggiunto 34 aree geografiche, 103 zone ed è disponibile in più di 200 paesi.

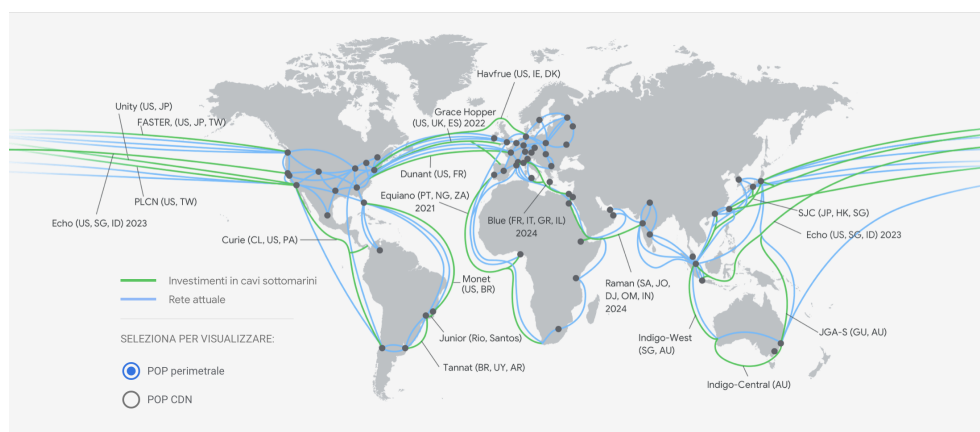


Figura 2.1: Mappa rete Google Cloud (agg.Sett 2022)

2.2 Strumenti di Google Cloud Platform

Google Cloud Platform è un prodotto derivato da decenni di esperienza nella gestione dei più grandi servizi Web e di maggior successo della storia. Google incorpora e applica queste conoscenze anche ai propri servizi. Google sta offrendo sempre più servizi cloud ed il numero di prodotti specializzati continua a crescere ad una velocità vertiginosa, come mostrato nel grafico seguente:

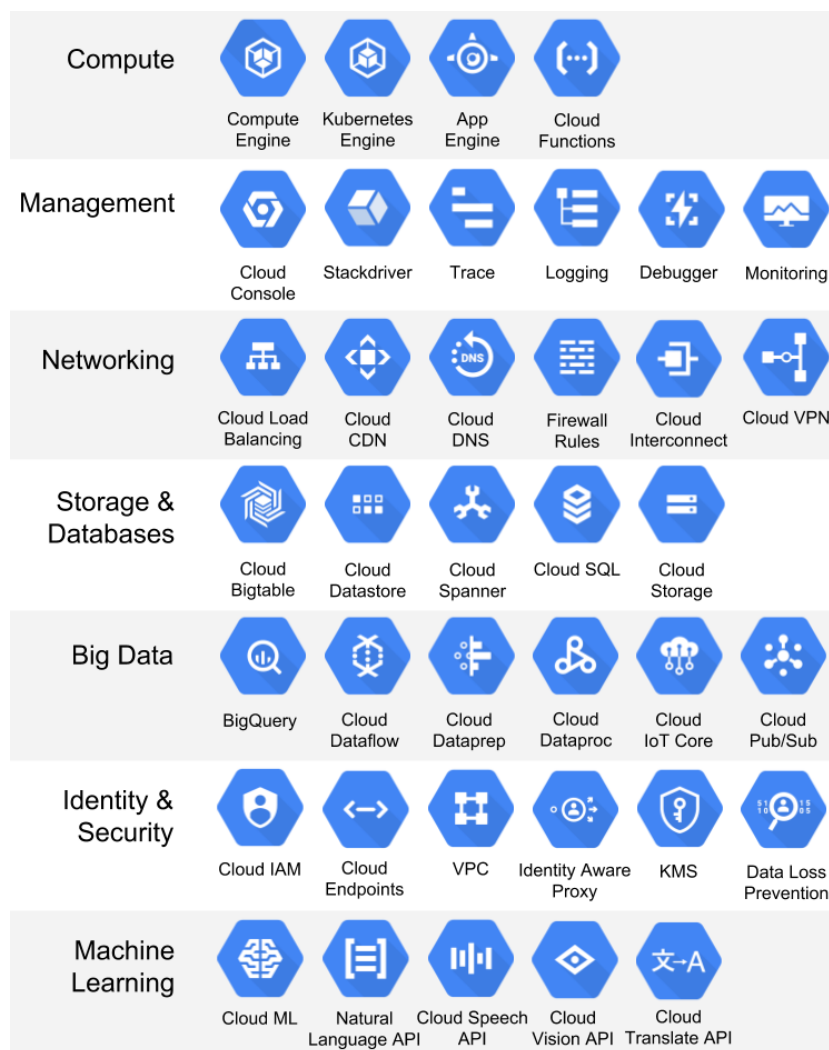


Figura 2.2: Catalogo di Google Cloud Platform

Come si nota dallo schema precedente GCP offre numerose soluzioni, con più di cento prodotti per gestire diversi ambiti, quali:

- AI e Machine learning.
- Gestione delle API
- Soluzioni Computing (App Engine, GPU Cloud, Compute Engine).
- Container (G.Kubernetes).
- Analisi dei Dati (BigQuery).
- DataBase.
- Strumenti per Sviluppatori .
- Sanità e scienze Biologiche.
- Cloud Ibrido e MultiCloud.
- Internet Of Things.
- Strumenti di Gestione.
- Media e Giochi.
- Migrazione.
- Networking.
- Suite Operative.
- Sicurezza e Identità.
- Archiviazione.
- Serverless Computing: con prodotti quali Cloud Run, CloudFunctions, App Engine e Flussi di Lavoro.

2.3 Perchè scegliere Google

Google Cloud Platform offre infrastrutture di servizi IaaS, PaaS e Serverless. Molte aziende hanno scelto di migrare verso la piattaforma cloud di Google perchè i costi sono più bassi; il servizio è scalabile e si paga solo per quello che si consuma.

Amazon, Microsoft e Google offrono tutti eccellenti piattaforme cloud pubbliche; Google punta a distinguersi come leader nei servizi gestiti scalabili e nei big data. Offre ai clienti l'accesso a molti degli stessi strumenti che utilizza internamente. Sfruttando il proprio patrimonio di conoscenze ed esperienze nell'esecuzione di importanti servizi come Ricerca e Gmail. La sua infrastruttura quindi è tra le più potenti e sicure al mondo.



Figura 2.3: Caratteristiche Google Cloud Platform

Una delle principali caratteristiche sta nel fatto della gestione Serverless Computing, il che offre una grande flessibilità con qualsiasi distribuzione di codice.

È possibile utilizzare applicazioni serverless full-stack con spazio di archiviazione, database, machine learning e molti altri metodi di Google Cloud. GCP Serverless Computing, consente di utilizzare qualsiasi linguaggio di programmazione e fornisce ogni tipo di aiuto anche con framework e librerie. Il codice

può essere distribuito in vari modi: come funzione, come codice sorgente, come contenitore, ecc.

Google Cloud si occupa di gestire tutte le attività relative a configurazione, provisioning, bilanciamento e altre mappature lato server, consentendo all'utente di avere maggior tempo con il suo prodotto.

2.4 Orientarsi verso soluzioni Serverless

In un ambiente tradizionale gli sviluppatori devono tenere conto dei server, configurandoli e gestendone l'infrastruttura.

In un ambiente serverless, gli sviluppatori non devono pensare a un server e possono concentrarsi sulla scrittura del codice. Ovviamente il codice funziona ancora su un server, ma tutto ciò che viene fornito con esso è gestito dalla piattaforma.

Microservizi

Prima di introdurre il prossimo argomento è necessario citare i microservizi. La maggior parte delle applicazioni comuni e più diffuse sono l'opposto dei microservizi, ovvero monoliti. In un'applicazione monolitica tutto il codice viene messo insieme ed un possibile errore ne influenzerebbe tutta l'applicazione.

I microservizi sono una soluzione a questo: dividono le applicazioni in piccole parti di funzionalità indipendenti. Poichè ogni funzionalità può essere vista come una piccola applicazione, queste possono essere distribuite e aggiornate separatamente, inoltre eventuali problemi non influenzerebbero le altre.

2.5 GCP soluzioni Serverless

Una delle tendenze più significative nell'informatica moderna è la crescita e l'adozione di soluzioni “highly managed”. Molte organizzazioni si stanno rendendo conto del valore di queste piattaforme, dove gli sviluppatori sono in grado di concentrarsi direttamente sull'affrontare le esigenze aziendali.

D'ora in poi, quindi, si parlerà solamente di applicazioni e logiche serverless. Analizzeremo una panoramica dei principali prodotti puri che la piattaforma di Google offre ai suoi utenti.

Vediamole qui di seguito, partendo dal più basso livello di astrazione andando verso l'alto:

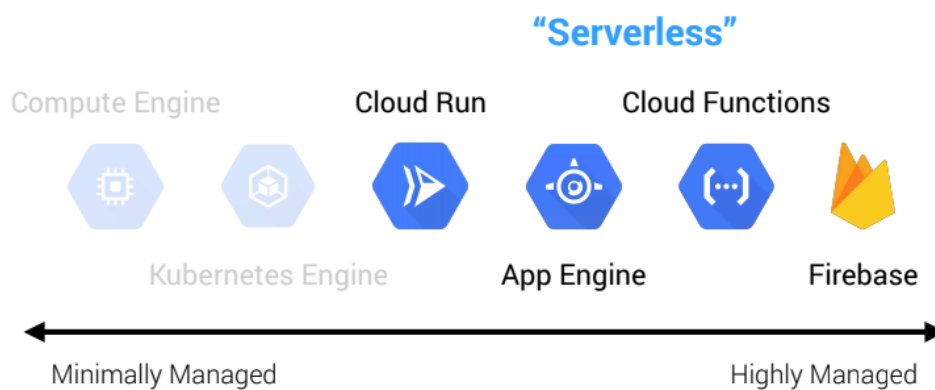


Figura 2.4: Soluzioni Serverless di Google Cloud Platform

2.5.1 Cloud Run

Cloud Run è un'offerta per fornire piattaforme di elaborazione gestite per container stateless. In sostanza, Google ne gestisce l'infrastruttura di calcolo sottostante e l'utente deve solamente fornire un container di applicazioni. Gestisce il ridimensionamento delle istanze verso l'alto e verso il basso, il bilanciamento del carico e la tolleranza agli errori. Cloud Run ha in realtà due modalità: la versione gestita da Google, che esegue i container sull'in-

infrastruttura di elaborazione interna di Google nota come Borg, e la versione GKE, che consente di eseguire carichi di lavoro sul proprio cluster GKE, questo perchè Cloud Run è basato su una piattaforma Kubernetes open source per carichi di lavoro serverless chiamata Knative .

2.5.2 App Engine

Google App Engine è una piattaforma PaaS di cloud computing per lo sviluppo e l'hosting di applicazioni web gestite dai Data Center proprietari. Le applicazioni sono incapsulate per sicurezza in sandbox ed eseguite su più server. App Engine offre una scalabilità automatica per le applicazioni a seconda del numero di richieste d'uso per quella applicazione, ed alloca in automatico risorse per gestire la domanda addizionale. Come già riportato, fu pubblicata come versione in “anteprima” nell'aprile 2008 e divenne ufficiale solo nel settembre 2011.

2.5.3 Firebase

Firebase è il più alto livello di astrazione offerto da GCP e consente di creare applicazioni mobili e Web rapidamente con un codice lato server minimo. Il compromesso è che si ha meno controllo sul sistema, ma può essere un'ottima soluzione per la prototipazione rapida di applicazioni o la creazione di un proof of concept con un investimento minimo. Il vantaggio principale per l'utente è quello di concentrare la maggior parte di sforzi nello sviluppo di codice dell'applicazione lato client e sull'esperienza utente.

2.5.4 Cloud Functions

Cloud Functions è invece un'offerta Functions-as-a-Service (FaaS) serverless di GCP. Carichi il codice funzione e Cloud Functions ne gestisce il runtime. Poichè si tratta di un ambiente sandbox, ci sono alcune restrizioni in runtime, ma è un'ottima scelta per creare servizi basati su eventi e connettere i sistemi insieme. Sebbene sia possibile sviluppare API di base rivolte

all'utente, gli strumenti operativi non sono sufficienti per i sistemi complessi. Il vantaggio è che le funzioni cloud sono altamente elastiche e hanno un sovraccarico operativo minimo poichè si tratta di una piattaforma serverless. Si tratta quindi di uno dei casi più estremi di piattaforme altamente gestite e che saranno il focus dell'intero progetto e dei prossimi capitoli.

Vedremo come le funzioni cloud consentono agli utenti di ignorare completamente la cerimonia di scrittura del codice standard dell'applicazione e dell'infrastruttura e mettersi subito al lavoro scrivendo il codice che conta davvero.

Esempio nell'uso comune

Nell'uso di tutti i giorni, le funzioni Cloud sono attivate su determinati eventi, come il caricamento di file su Cloud Storage o semplici chiamate HTTP.

Capitolo 3

Google Cloud Functions

Le Cloud Functions di Google non sono altro che dei servizi FaaS (Function as a Service) di GCP estremamente scalabili e pay-as-you-go (con pagamento a consumo) per eseguire codice senza gestirne il server.

GCF è un ambiente di esecuzione serverless per la creazione e la connessione di servizi cloud. La funzione viene attivata quando viene attivato un evento trigger ed il codice viene eseguito in un ambiente completamente gestito da Google.

Le funzioni Cloud vengono scritte in JavaScript ed eseguite in un ambiente Node.js v6.9.1 su Google Cloud Platform. Una Funzione può essere eseguita in qualsiasi runtime di Node.js standard, quindi sia la portabilità che l'esecuzione di test in locale sono estremamente semplificate.

3.1 Funzionalità principali delle GCF

Si elencano dunque le principali caratteristiche del servizio Google Cloud Functions.

3.1.1 Esperienza di sviluppo semplice e veloce

Offrono un'esperienza di sviluppo facile ed intuitiva. L'utente scrive il codice e Google Cloud gestisce la parte operativa al suo posto. Ne guadagna

la velocità di sviluppo, poichè si scrivono solo piccoli snippet di codice che rispondono ad eventi. Inoltre, Connettendosi a Google Cloud o a servizi cloud di terze parti mediante trigger si possono risolvere problemi di orchestrazione più complessi.

3.1.2 Scalabilità automatica

GCF offre scalabilità da zero a milioni di utenti su scala mondiale senza la necessità di gestire alcuna infrastruttura. Cloud Function gestisce e scala automaticamente l'infrastruttura sottostante in base ai carichi di lavoro.

3.1.3 Modello di pagamento a consumo

Google fattura solo il tempo di esecuzioni delle funzioni, per la precisione, arrotonda ai 100 millisecondi più vicini. Quando una funzione è inattiva non si paga nulla. Le Cloud Functions si avviano e si arrestano automaticamente in risposta agli eventi.

3.1.4 Nessun vincolo al fornitore grazie ad una tecnologia aperta

Le funzioni cloud possono essere eseguite in diversi ambienti di lavoro, evitando vincoli. È possibile scrivere, testare ed utilizzare le funzioni in locale oppure online su Cloud Platform; inoltre, sono supportate in ambienti quali Cloud Run ed altri ambienti serverless basati su Knative.

3.2 Casi d'uso Cloud Functions

Di seguito vado ad elencare quali sono i principali casi d'uso di utilizzo per Google Cloud Functions, così da fornire una generica conoscenza dei loro molteplici utilizzi e della loro applicazione in aree diverse.

3.2.1 Integrazione con servizi e API di terze parti

Le Cloud Functions si possono usare per esporre i microservizi tramite API HTTP o per integrarli con servizi di terze parti che offrono integrazioni webhook per estendere velocemente un'applicazione con funzionalità avanzate, quali l'invio di una mail per conferma dopo la corretta esecuzione di un pagamento Stripe o la risposta ad eventi Twilio relativi agli SMS.

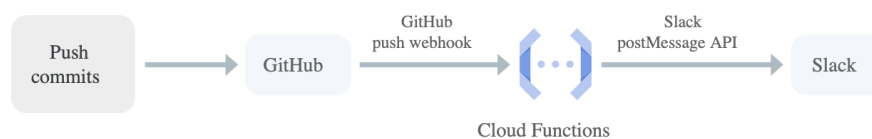


Figura 3.1: Schema CF con servizi e API di terze

3.2.2 Backend serverless per dispositivi mobili

Le CF possono essere utilizzate direttamente da Firebase per estendere funzionalità di applicazioni senza l'avvio di alcun server. È possibile eseguire il codice in risposta ad azioni di utenti, analisi ed eventi di autenticazione per garantire il coinvolgimento degli users con notifiche basate sugli eventi e scaricare da Google Cloud le attività che comportano un utilizzo estensivo di CPU e servizi networking.



Figura 3.2: Schema CF per Firebase

3.2.3 Backend IoT Serverless

Le Cloud Functions possono essere combinate con Cloud IoT Core e altri servizi completamente gestiti per creare backend per la raccolta, l'elaborazione in tempo reale e l'analisi dei dati di telemetria inviati da dispositivi Internet of Things IoT; Le CF consentono di applicare logiche personalizzate ai singoli eventi a mano a mano che si verificano.

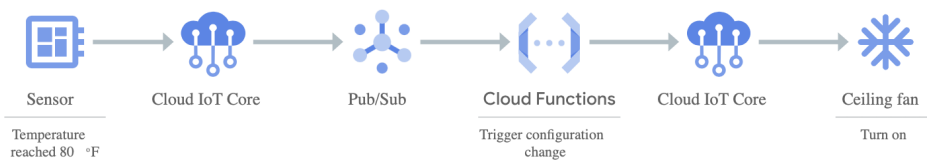


Figura 3.3: Schema CF backend IoT serverless

3.2.4 Elaborazione file in tempo reale

È possibile eseguire codice in risposta ai cambiamenti nei dati. Cloud Functions possono rispondere ad eventi generati da servizi Google Cloud, quali ad esempio, Cloud Storage, Pub/Sub e Cloud Firestore per elaborare subito file dopo il loro caricamento e creare miniature a partire da immagini caricate, elaborare log, convalidare contenuti, eseguire la transcodifica di video, nonché convalidare, aggregare e filtrare i dati in tempo reale.



Figura 3.4: Schema CF per elaborazione in tempo reale

3.2.5 Elaborazione flussi in tempo reale

Cloud Functions possono rispondere agli eventi generati da Pub/Sub per elaborare, trasformare e arricchire i flussi di dati per l'elaborazione delle transazioni, l'analisi di clickstream, il monitoraggio delle attività delle applicazioni, la telemetria dei dispositivi IoT, l'analisi dei social media e altri tipi di applicazioni.



Figura 3.5: Schema CF con flussi in tempo reale

3.2.6 Assistenti Virtuali ed esperienze di conversazione

L'intelligenza artificiale si integra facilmente nelle applicazioni CF con l'API Cloud Speech e Dialogflow, possono estendere prodotti e servizi con esperienze di conversazione naturali basate su testo e voce, che aiutano gli utenti a essere più produttivi. Grazie all'Assistente Google, ad Amazon Alexa, a Facebook Messenger e ad altri dispositivi e piattaforme comunemente utilizzati è possibile entrare in contatto con gli utenti.

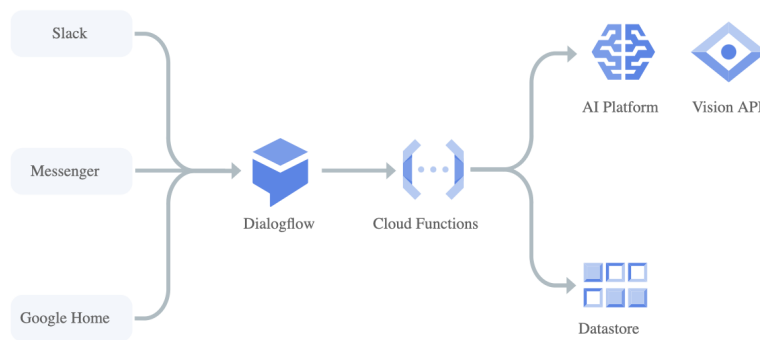


Figura 3.6: Schema CF con assistenti virtuali

3.2.7 Analisi di video e immagini

Utilizza le Cloud Functions con le API Video Intelligence e l'API Cloud Vision per recuperare informazioni pertinenti da video e immagini, consentendo, così di eseguire ricerche nei contenuti multimediali, individuarli e ricavarne insight significativi.

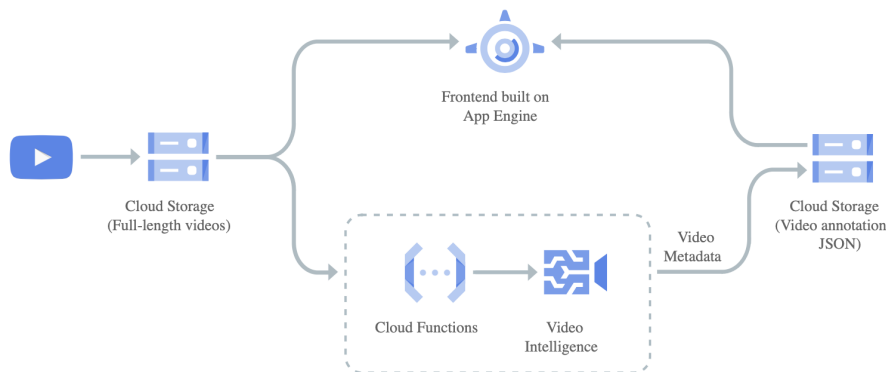


Figura 3.7: Schema CF per l'analisi di video e immagini

3.2.8 Analisi dei sentiment

Un ulteriore aspetto di utilizzo delle CF è quello che le vede, insieme all'API Cloud Natural Language per scoprire la struttura e il significato del testo e per aggiungere potenti funzionalità di analisi del sentiment e di estrazione degli intent alle tue applicazioni.

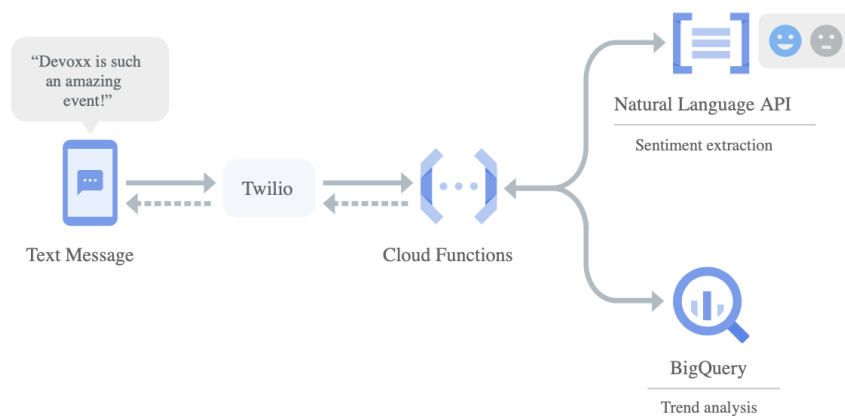


Figura 3.8: Schema CF per l'analisi di sentiment

3.3 Release ed ultime novità

Google rilasciò le Cloud Functions nel 2016 in versione Alpha, limitandola solamente a funzioni JavaScript; ma nonostante questo già prevedevano un'enorme quantità di utilità per determinate classi di problemi.

Nel 2018 fu pubblicata la prima versione beta e negli ultimi due anni subì importanti novità.

Ad oggi le GCF possono essere scritte in altri linguaggi oltre a Node.js, questi linguaggi sono: Python, Go, Java, .NET, Ruby, e PHP.

Inoltre, verso metà maggio 2022 il team Google rilasciò in beta la seconda generazione di Cloud Functions, con la possibilità di offrire un'infrastruttura sempre più avanzata, nuove funzionalità e performance migliorate.

Questa versione provvisoria subisce un importante aggiornamento verso inizio settembre, il quale lo rende disponibile a tutti gli sviluppatori.

3.3.1 Functions Frameworks

Il team di Cloud Functions ha inoltre rilasciato i Functions Frameworks: un set di librerie idiomatiche open source per ciascuno dei linguaggi supportati. I Functions Frameworks permettono di eseguire, testare e debuggare la funzione cloud nel proprio ambiente locale, offrendo lo stesso comportamento come se fosse in produzione ma senza la necessità di effettuare il deployment.

Capitolo 4

Ambiente e sviluppo Google Cloud Function

Si fornisce una prima anteprima dell'ambiente di sviluppo e una guida per lo sviluppo di funzioni cloud sulla console di Google.

4.1 Requisiti

Non è obbligatorio, ma molto consigliato, utilizzare un browser web OS Chrome, poichè ne è garantito il pieno supporto, rispetto ad altri browser quali non sempre risultano compatibili.

Il primo passo è quello di collegarsi al sito: <https://console.cloud.google.com>
Il sito chiederà di effettuare il login.

Se non si ha un account Google Cloud Platform per averne uno basta utilizzare un account mail Google (gmail.com) ed una carta di pagamento come garanzia. Al primo utilizzo si avranno a disposizione 300\$ da spendere per un periodo di 90 giorni. Dopo di che sarà richiesto un ulteriore abbonamento o versamento.

4.2 Home Page

La pagina principale di Google Cloud Platform è composta da una dashboard principale, dove è possibile vedere e tener traccia di progetti, risorse, e principali notizie delle proprie creazioni, attive e no.

Il menu a tendina a lato permette di navigare e selezionare il servizio Google dedicato; qualsiasi sia il tuo scopo o necessità dovrai passare per questo menu e selezionare il campo di interesse, è qui che si trovano le Cloud Functions. Infine, nella parte in alto a sinistra è possibile gestire e modificare il proprio profilo, pagamenti ed utilità.

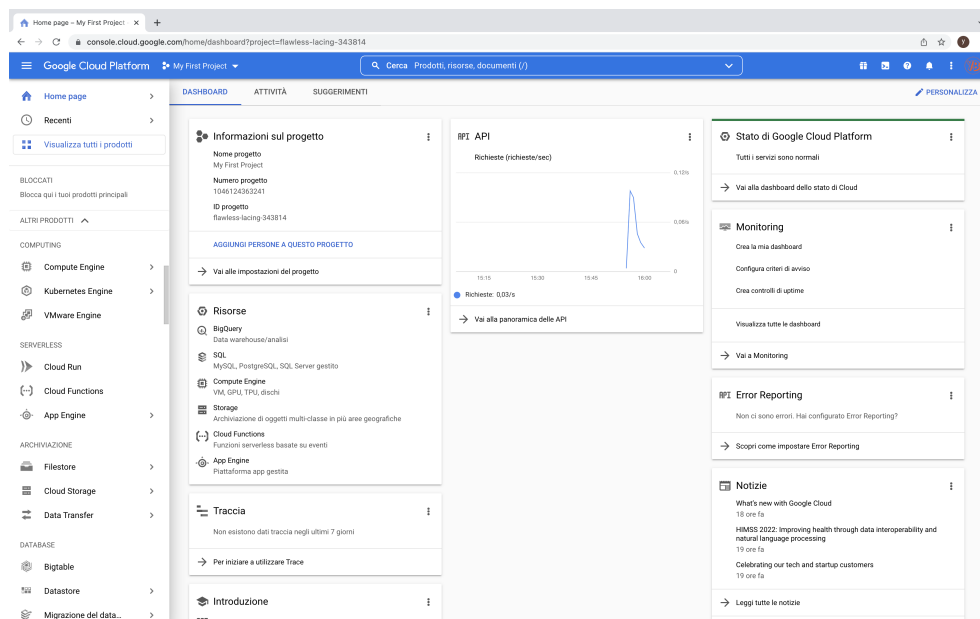


Figura 4.1: Home page Google Cloud Platform

4.3 Passi necessari per lo sviluppo di una funzione

Viene mostrato come costruire una semplice applicazione “Hello World”, soffermandosi su come utilizzare l’ambiente online e come settare tutti gli aspetti che precedono la stesura di codice per la funzione.

4.3.1 Abilitazione API

In primis è necessario abilitare le APIs di riferimento, in questo caso attiviamo quelle relative alle Functions:

- Cloud Functions API.
- Cloud Build API.

La loro abilitazione può essere fatta una tantum, in quanto resteranno attive per sempre ad ogni nostro accesso. Per chiuderle sarà necessario agire manualmente su di esse.

Una volta cliccato su abilita, siamo pronti per l’inizio di un nuovo progetto.

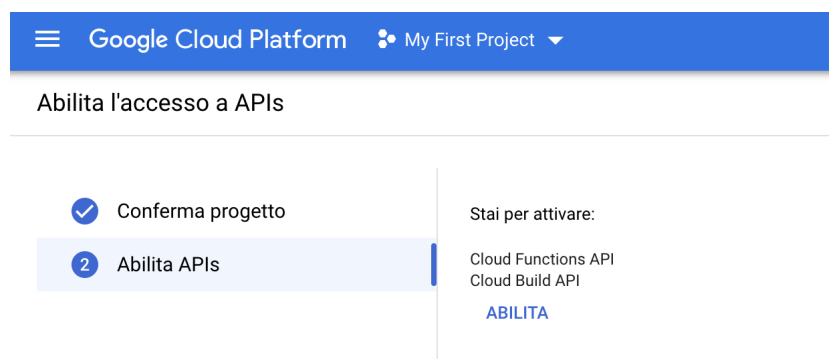


Figura 4.2: Abilitazione API

4.3.2 Creazione e sviluppo

Ci sono diversi modi per implementare una Funzione Cloud con Google. Vediamo di seguito il modo più semplice:

Andiamo sulla console di Google nel menu di sinistra e selezioniamo Cloud Functions, clicchiamo su “Crea Funzione”. Apparirà un menu del tipo:

Cloud Functions | < Crea funzione | IMPARA

1 Configurazione — 2 Codice

Nozioni di base

Ambiente
1ª gen.

Nome funzione *
hello-world

Area geografica
europe-west6

Trigger

⚙ HTTP

Tipo di trigger
HTTP

URL

https://europe-west6-flawless-lacing-343814.cloudfunctions.net/hello-world

Autenticazione

☒ Consenti chiamate non autenticate
Seleziona questa opzione se stai creando un'API o un sito web pubblici.

☐ Autenticazione necessaria
Gestisci gli utenti autorizzati con Cloud IAM.

☒ Richiede HTTPS

SALVA ANNULLA

Figura 4.3: Configurazione della funzione

- **Ambiente:** possibile selezionare la versione delle Cloud Function. (Al momento è disponibile anche la 2a versione, in beta, e lanciata sul mercato da pochissimo).
- **Nome Funzione:** si imposta il nome della funzione che si andrà ad implementare, è molto importante selezionare un nome appropriato, poichè sarà molto utile in futuro quando ne si avrà il bisogno di utilizzo.
- **Area Geografica:** la zona in cui la funzione sarà implementata. Andrà impostata la regione equivalente alla zona in cui viene utilizzata la mia applicazione.
- **Tipo di Trigger:** come verrà chiamata la mia funzione. Ci sono diversi modi di chiamata, utilizzeremo HTTP, che è la più generica.
- **Autenticazione:** scegliere se la chiamata alla funzione può avvenire solo previa autenticazione o meno.

In autonomia Google assegnerà un URL alla mia funzione, posso dunque vedere e copiarci l'indirizzo web per il suo utilizzo.

Impostazioni di runtime, build, connessioni e sicurezza

< RUNTIME BUILD CONNESSIONI REPOSITORY PE >

Memoria allocata *
128 MB

Timeout *
60 secondi ?

Account di servizio di runtime ?
Account di servizio di runtime
App Engine default service account

Scalabilità automatica ?
Numero minimo di istanze 0 Numero massimo di istanze 3000

Variabili di ambiente runtime ?
+ AGGIUNGI VARIABILE
Screenshot

Figura 4.4: Configurazione runtime

Sono presenti ulteriori campi, che presi e spiegati uno ad uno impegnerebbero interi capitoli. Bisogna fare attenzione solamente nell’allocazione della memoria, impostazione del timeout e della scalabilità automatica.

- **Memoria Allocata:** memoria RAM che sarà dedicata all’esecuzione della funzione. Il valore massimo è 8 Gb, il fatto è che le Google Cloud Functions non sono funzioni che richiedono molte risorse. Impostiamo la cella in 256 Mb, che per il nostro utilizzo sarà sufficiente.
- **Timeout:** il tempo massimo consentito per l’esecuzione della funzione. Se l’esecuzione non viene completata entro il timeout, la funzione sarà stoppata. Di default è impostato su 60 secondi, ma i valori minimi e massimi sono tra 1 secondo e 540 secondi (9 minuti).

In generale è bene selezionare valori alti poichè le GCF soffrono di Cold Start e la loro prima chiamata può essere molto lenta rispetto le successive.

- **Scalabilità automatica:** se si vogliono creare più istanze della funzione per consentire chiamate multiple allo stesso tempo o meno. Per migliorare le performance si può impostare il numero minimo di istanze con un numero superiore a 0. Puoi inoltre impostare il numero massimo di chiamate per limitare i costi e le eccedenze temporanee. Lasciare vuoto o impostare a “0” per saltare questo controllo e creare automaticamente istanze ad ogni necessità.

Infine, clicchiamo su “avanti” per la prossima schermata.

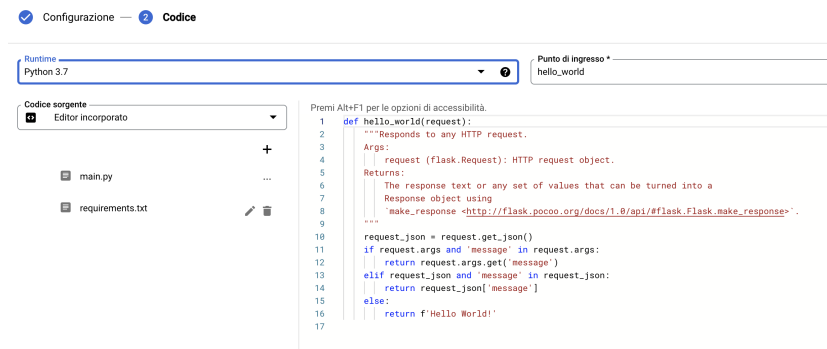


Figura 4.5: Funzione di default

Per ultimo passo si seleziona:

- **Runtime:** qui si sceglie il linguaggio di programmazione del codice sorgente. Ad oggi sono disponibili i seguenti linguaggi con le relative versioni:
 - .Net Core 3.1.
 - Go 1.13, Go 1.16.
 - Java 11, (Java 17 in anteprima).

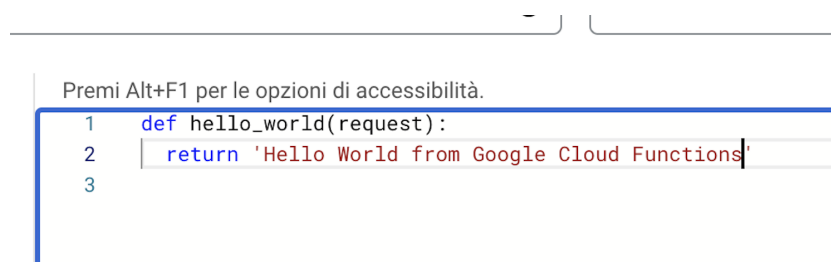
- Node.js 16, Node.js 14, Node.js 12 e Node.js 10.
 - PHP 7.4, (PHP 8.1 in anteprima).
 - Python 3.7, Python 3.8, Python 3.9 (e Python 3.10 in anteprima).
 - Ruby 2.6, Ruby 2.7 (e Ruby 3.0 in anteprima).
- **Codice Sorgente:** l'origine del codice. Noi vedremo l'utilizzo dell'editor online ma è possibile utilizzare repository in cloud.

Prima di andare ad implementare il nostro codice, analizziamo quello che appare di default:

Si notano due files: `main.py` e `requirements.txt`. Nel file `requirements` ci andranno inserite le librerie e le loro versioni necessarie per eseguire il nostro codice. Nel file `main.py`, l'esempio mostra come ricevere una chiamata HTTP e controllare il campo del messaggio se presente all'interno della richiesta effettuata. Se è presente ritorna il messaggio (ed il testo). Altrimenti ritorna "Hello World".

Ora si può scrivere il codice e dare conferma con "Esegui il deployment" ed attendere la fine del processo di deploy.

Scriveremo per esempio un semplice messaggio a schermo: "Hello World from Google Cloud Functions".



The image shows a code editor window with a title bar. Inside, there is a text area containing Python code. At the top of the text area, there is a small tooltip that says "Premi Alt+F1 per le opzioni di accessibilità." Below this, the code is as follows:

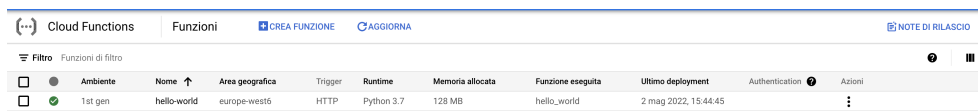
```
1 def hello_world(request):  
2     return 'Hello World from Google Cloud Functions'  
3
```

Figura 4.6: Codice della funzione

4.3.3 Risultato

La nostra prima funzione è pronta, come indicato dalla spunta verde, visibile nella figura 4.7.

Andiamo a ricercare il suo URL, clicchiamo sopra il nome della funzione, spostiamo nella sezione “Trigger” e vediamo l’indirizzo web.



The screenshot shows the Google Cloud Functions console. At the top, there's a header with 'Cloud Functions' and 'Funzioni'. Below that, there's a table listing functions. The first function is 'hello-world' with a green status icon. The table has columns for 'Ambiente', 'Nome', 'Area geografica', 'Trigger', 'Runtime', 'Memoria allocata', 'Funzione eseguita', 'Ultimo deployment', 'Authentication', and 'Azioni'.

	Ambiente	Nome	Area geografica	Trigger	Runtime	Memoria allocata	Funzione eseguita	Ultimo deployment	Authentication	Azioni
<input checked="" type="checkbox"/>	1st gen	hello-world	europe-west6	HTTP	Python 3.7	128 MB	hello_world	2 mag 2022, 15:44:45		

Figura 4.7: Status della Funzione

Copiamo l’indirizzo, incolliamolo in una nuova scheda internet ed osserviamo il risultato.

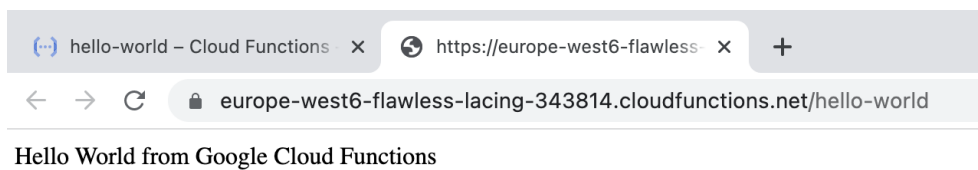


Figura 4.8: Risultato della funzione

4.4 Aggiunta parametri con metodo GET

Modifichiamo la funzione precedentemente creata per poter leggere dati in input , tramite una semplice richiesta GET.

4.4.1 Modifica codice funzione

Per modificare il codice di una funzione esistente basterà cliccare su di essa e cliccare su “modifica”.

Sarà richiesto di modificare tutte le impostazioni di configurazione e di runtime precedentemente impostate, cliccando su “avanti” si terranno valide quelle già impostate. Si passa alla seconda parte e qui sarà possibile editare il testo del codice.

Aggiungiamo, come si nota nella prossima immagine, il comando:

request.args.get(), che permette di leggere argomenti utilizzando il metodo GET.

Premi Alt+F1 per le opzioni di accessibilità.

```
1 def hello_world(request):  
2     name = request.args.get('name')  
3     return 'Hello' + name + 'from Google Cloud Functions'  
4
```

Figura 4.9: Codice con metodo GET

Ora salviamo e rieseguiamo il deploy della nostra funzione, quindi attendiamo che sia pronta.

Il metodo GET, è il metodo che comunemente utilizziamo tutti i giorni per fare una ricerca sui motori di ricerca tradizionali, non appena clicchiamo sul tasto di inizio ricerca possiamo notare come in realtà venga chiamata una

funzione seguita dal simbolo `?` più il testo digitato. Questo non è altro di come avviene il passaggio dei dati tramite il metodo GET tra diversi processi. Quindi per poter utilizzare la nostra nuova funzione basterà copiare nella nuova scheda web il relativo URL assegnato alla funzione seguito da: `? nome variabile = testo` ed ecco quindi il risultato.

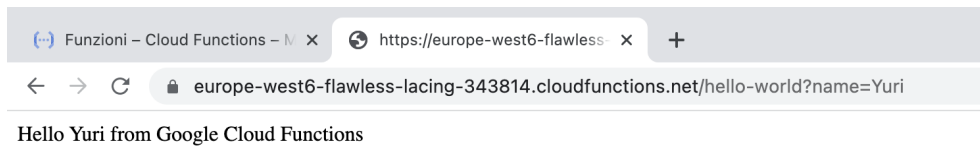


Figura 4.10: Output funzione GET

Capitolo 5

Scrivere funzioni Cloud

Dato che Cloud Functions supporta la scrittura di codice in diversi linguaggi di programmazione; il runtime scelto e il tipo di funzione determinano la struttura del codice e l'implementazione della funzione.

Sappiamo che su GCF esistono due tipi di funzioni: HTTP e Funzioni basate su eventi. Analizziamo le prime osservando la loro semantica per un corretto utilizzo.

5.1 Attivatori HTTP

Nelle Cloud Functions, un trigger HTTP consente di eseguire funzioni in risposta ad eventi HTTP/ HTTPS; questi attivatori reagiscono alle richieste dei metodi GET, POST, PUT, DELETE e OPTIONS.

Quando si specifica un trigger HTTP per una Cloud Function, questa viene associata ad un URL da cui sarà possibile ricevere tutte le richieste.

La seconda generazione di Cloud Function supporta solamente il protocollo HTTPS, mentre se si decide di utilizzare la prima generazione è possibile scegliere se il protocollo di sicurezza sarà utilizzato o meno.

Per impostazione predefinita, tutte le richieste HTTP richiedono l'autenticazione, se si vogliono invece consentire chiamate non autenticate è necessario configurare manualmente questo permesso in fase di deploy.

Questa è la struttura del deploy di una funzione HTTP eseguita a riga di comando:

```
gcloud functions deploy YOUR_FUNCTION_NAME \
--trigger-http \
[--allow-unauthenticated] \
[--security-level=SECURITY_LEVEL] \
...
```

Figura 5.1: Chiamata funzione HTTP

- Il flag `--trigger-http` specifica che la funzione utilizza chiamate HTTP.
- `--allow-unauthenticated` specifica che la funzione può essere chiamata senza autenticazione.
- `--security-level` serve se utilizzo una connessione HTTPS (le opzioni sono `secure-always` o `secure-optional`).

5.2 Scrivere funzioni HTTP

Si visualizza di seguito il file di origine della funzione HTTP di base per il runtime;

```
import functions_framework

# Register an HTTP function with the Functions Framework
@functions_framework.http
def my_http_function(request):
    # Your code here

    # Return an HTTP response
    return 'OK'
```

Figura 5.2: Esempio funzione in Python

5.2.1 Utilizzare i Function Framework

Come già accennato nel capitolo precedente, per scrivere il codice di una funzione cloud è necessario utilizzare i Function Framework. Si tratta di librerie open source, disponibili su GitHub, scritte dal team Google volte a semplificare il codice ed a rendere più leggere e performanti le funzioni.

Queste sono supportate anche in diversi ambienti Google quali Cloud Function, Knative, Cloud Run e sviluppo locale. Function Framework definisce un'interfaccia per costruire servizi modulari utilizzati tramite richieste HTTP.

Nel caso di sviluppo in Python, dopo l'utilizzo dei relativi "Function Framework per Python", la funzione deve accettare in input un oggetto "Richiesta Flask" e restituire un valore che Flask possa convertire in oggetto di risposta HTTP: quale per esempio il semplice messaggio "OK".

Oggetto di richiesta Flask

Flask è un popolare framework Python utilizzato per lo sviluppo di applicazioni web; Questo oggetto è estremamente flessibile e consente agli sviluppatori di agganciarsi a diversi servizi e API di terze parti;

La nostra funzione Cloud Function vorrà in input quindi un oggetto appartenente alla classe `Flask.flask`.

Lo stesso vale per gli altri linguaggi di programmazione supportati:

- In Node.js, la funzione si registra con il "Framework delle funzioni per Node.js". La funzione del gestore HTTP deve essere una funzione *middleware Express* che accetti gli argomenti di richiesta e risposta e invii una risposta HTTP.

- In Go, si usa il “Functions Framework for Go” nella funzione `init()`. La funzione gestore HTTP deve utilizzare l’interfaccia standard *http.HandlerFunc* per inviare una risposta HTTP.
- In Java, si usa invece l’“API Java di Framework” delle funzioni per implementare una classe HTTP con l’interfaccia *HttpFunction*. Il metodo *service()* poi deve inviare una risposta HTTP.
- Nei runtime .NET, si usa il “Framework delle funzioni per .NET” per implementare una classe di gestore HTTP con l’interfaccia *IHttpFunction*. Il metodo *HandleAsync()* accetta un oggetto *ASP.NET HttpContext* standard come argomento e deve inviare una risposta HTTP.
- In Ruby, si utilizza il “Framework delle funzioni per Ruby”. La funzione gestore del gestore HTTP deve accettare un oggetto Richiesta *Rack* come argomento e restituire un valore che può essere utilizzato come risposta HTTP.
- In PHP, infine, si registra una funzione HTTP con il “Framework delle funzioni per PHP”. La funzione gestore deve accettare un argomento che implementa l’interfaccia *PSR-7 ServerRequestInterface* e deve restituire una risposta HTTP come stringa o un oggetto che implementa l’interfaccia *PSR-7ResponseInterface*.

5.3 Accessibilità

Analizziamo come si richiama una funzione e quanto può essere accessibile da fuori, ovvero: se la mia funzione viene invocata da un programma esterno all'ambiente in cui si trova la stessa, si riesce a utilizzare conoscendo semplicemente il suo indirizzo o vi è qualche limitazione?

5.3.1 Chiamate in locale

Se ho configurato l'esecuzione della mia funzione localmente, ovvero su "localhost", si presuppone che abbia installato nella mia macchina anche lo strumento *curl*: si tratta di un comando molto importante e conosciuto nell'ambito della programmazione poichè permette di gestire diversi protocolli e simulare chiamate browser quali HTTP, HTTPS e FTP; alcuni sistemi operativi lo hanno già installato di default.

Per inviare richieste alla mia funzione cloud in locale basta assicurarsi di aver impostato il corretto URL della funzione, in questo caso di default sarà ospitata all'indirizzo *localhost:8080*.

5.3.2 Chiamate dirette

Per supportare l'integrazione ai diversi ambienti e al debug rapido, Cloud Function fornisce un comando *call* nell'interfaccia a riga di comando e la funzionalità di test direttamente nell'interfaccia utente di Cloud Console online. Questo consente di richiamare subito la funzione per testare se l'output richiesto fosse quello desiderato.

Per richiamare direttamente la funzione usando l'interfaccia a riga di comando *gcloud*, si utilizza il comando *gcloud functions call* seguito dal nome della funzione.

Altrimenti per chiamarla direttamente dalla console Google è necessario seguire i seguenti passaggi:

1. Andare su *Panoramica* in Google Functions.

2. Nell'elenco selezionare la funzione d'interesse e cliccare su *Dettagli Funzione*.
3. Fare clic sulla scheda *Test*.
4. Nel campo *Evento di Trigger* inserire i dati che la funzione prevede in input.
5. Infine, fare clic su *Testa Funzione*.

5.3.3 Deployment

```
gcloud functions deploy YOUR_FUNCTION_NAME \
[--gen2] \
--region=YOUR_REGION \
--runtime=YOUR_RUNTIME \
--source=YOUR_SOURCE_LOCATION \
--entry-point=YOUR_CODE_ENTRYPOINT \
TRIGGER_FLAGS
```

Figura 5.3: Schema deployment funzione

Gli utenti che vogliono richiamare una qualunque funzione devono avere il ruolo di Sviluppatore Cloud Function o un ruolo che includa le stesse autorizzazioni.

Nell'immagine in alto sono riportate le istruzioni per eseguire il deploy della funzione dall'interfaccia a riga di comando gcloud o da Google Cloud Console:

- Il primo argomento richiesto è il nome della funzione.
- Il flag *-gen2* specifica che si vuole eseguire il deploy in Cloud Function di seconda generazione, l'omissione di questo flag invece comporta l'utilizzo della prima generazione.
- Il flag *-region* specifica l'area geografica in cui eseguire la funzione, la loro scelta ne determina il prezzo e le prestazioni.

- Il flag *-runtime* permette di scegliere il runtime tra quelli supportati.
- Il flag *-source* specifica la posizione del codice sorgente della funzione.
- Il flag *-entry-point* indica il punto d'ingresso, ovvero il codice che verrà eseguito quando la funzione sarà chiamata; il nome di questo flag deve essere un nome di classe o di funzione esistente nel codice sorgente.

È possibile sia eseguire il deploy della mia funzione in locale, sia della funzione che si trova su Cloud Storage. Attenzione, in questo caso il codice sorgente deve essere pacchettato come file ZIP.

Per far sì che Cloud Functions possa leggere da Cloud Storage sono necessarie autorizzazioni diverse in base alla generazione utilizzata:

- Se utilizzo Cloud Functions di prima generazione l'account che esegue richiede l'autorizzazione di lettura dal bucket.
- Se invece utilizzo quelle di 2a generazione, devo distinguere in due casi: se il bucket si trova in un progetto diverso è necessario concedere manualmente l'autorizzazione; invece, se il bucket è lo stesso questa viene automaticamente concessa.

5.3.4 Attenzione ai CORS

CORS: Cross-Origin Resource Sharing è un metodo per gestire chiamate di applicazioni in gestione su un certo dominio verso un dominio esterno; Se CORS non è configurato correttamente potrebbero presentarsi errori in caso di chiamate multi origine.

Per questo motivo è necessario predisporre a priori l'applicazione a eventuali chiamate, impostando un'intestazione di autorizzazione corretta ad ogni chiamata HTTP.

L'unica attenzione va posta con il metodo OPTION in quanto le sue richieste non hanno un'intestazione di autorizzazione e quindi tutte le chiamate multi dominio verranno rifiutate; per ovviare al problema le soluzioni sono due:

ospitare la mia app web e Cloud Function sullo stesso dominio o consentire chiamate non autorizzate alla funzione.

5.4 Analisi portabilità

Ora ci soffermiamo sull'analizzare quanto sia portabile utilizzare le Cloud Function. Cosa accade se in futuro si decida di trasferire tutto su un diverso Host? o se si volesse replicare il tutto in un ambiente locale privato? Sarà sufficiente fare un semplice copia e incolla del mio codice?

5.4.1 La scelta del linguaggio

Abbiamo visto che le Cloud Function possono essere scritte in diversi linguaggi di programmazione, ma ognuno di essi utilizza set di librerie Google il che le rendono difficili da clonare in ambienti esterni o addirittura impossibile.

Può essere l'esempio di funzioni scritte utilizzando linguaggio Java: nel caso di applicazioni HTTP si utilizzano classi proprietarie di Google (HTTP. Cloud.function HTTP Function), quindi con le relative istanze, funzioni e attributi, che chiaramente non sono replicabili e utilizzabili all'infuori dell'ambiente di Google Cloud. In questo caso, come anche in tutte le altre possibilità di utilizzo, sarebbe necessario riscrivere completamente il codice della mia funzione cloud, tenendo di fatto uguale solamente la logica applicativa che ne sta dietro.

5.4.2 Conclusione con vantaggi e svantaggi

L'utilizzo di questi Google Framework rende i codici più veloci e snelli, le applicazioni risultano pienamente efficienti e tutto l'ecosistema ne trae vantaggio offrendone pieno supporto e velocità di interconnessione e scambio di risorse. Detto questo è chiaro, però, che analizzandone il grado di portabilità

otteniamo un risultato quasi pari allo zero.

Sappiamo che le Cloud Function, per loro natura, non sono mai composte da numerose righe di codice; si tratta solamente di piccole porzioni utili ad attivare e richiamare diverse APIs o servizi richiesti.

Se quindi, il mio obiettivo è quello di traslocare il mio programma altrove, la risposta è sì, è necessario riscrivere quasi completamente il mio codice.

A questo punto non sarei però coinvolto da un eccessivo carico di lavoro: Google Cloud Function supporta diversi linguaggi di programmazione, quindi un programmatore che ne intende fare uso può sin da subito scegliere quello a lui più noto, senza dover impegnare risorse nello studio di un nuovo linguaggio solamente con il fine di utilizzare questa suite.

Inoltre, in caso di riscrittura la scaletta e la logica applicativa utilizzata nel codice sarebbe esattamente la stessa.

Gli argomenti chiave nella realizzazione di una nuova applicazione tale e quale, ma in un diverso ambiente, sono in ordine di importanza: la scelta di trovare altre APIs e servizi esterni a Google che compiano gli stessi compiti precedenti, dopo di che sarebbe meglio optare per un nuovo ambiente anch'esso serverless, che offra le stesse garanzie e livelli di Google, così che l'intero tempo speso nel riscrivere il mio programma andrebbe solo ad interessare la stesura di codice e non anche nel scegliere i vari domini, spazi cloud, server, zone e livelli di esecuzione, utilizzando quindi i vantaggi che un ambiente serverless comporta.

Capitolo 6

Esempio di interazione con altri servizi Google

In questo ultimo capitolo si mostra un esempio di codice che utilizza Cloud Function insieme ad altri servizi Google per schedare dei processi automatici: osserveremo la struttura, gli agenti chiamati in causa e ne analizzeremo il risultato commentando con alcune delle possibili occupazioni che il modellino potrà avere.

6.1 Architettura

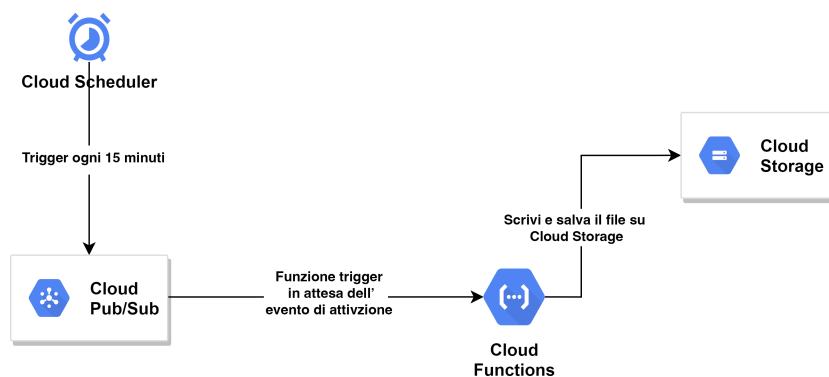


Figura 6.1: Schema architettura

Come mostra lo schema saranno coinvolti i seguenti 4 servizi:

1. Cloud Function.
2. Cloud Pub/Sub: si tratta di un servizio di messaggistica basato su eventi tra applicazioni e servizi. Ci sono due parti: un publisher di dati, colui che pubblica il messaggio ed un consumatore, ovvero colui che crea una sottoscrizione a quel messaggio. Nel nostro caso Cloud Schedule produrrà un evento Cloud Pub/Sub che andrà ad attivare la nostra Cloud Function.
3. Cloud Scheduler: si tratta di uno scheduler di lavori completamente gestito da Google Cloud Platform. Attenzione, non è un servizio di elaborazione, ma può solamente inviare messaggi HTTP o Pub/Sub, quindi di per se è piuttosto limitato. In questo caso sarà utilizzato per generare un evento Pub/Sub ogni 15 minuti.
4. Cloud Storage: si tratta di un contenitore di dati, con funzionamento simile agli stessi servizi conosciuti di archiviazione online. La sua caratteristica principale è quella di essere molto robusto poichè funge da spina dorsale per tutti servizi di Google Cloud Platform ed è un servizio business o anche enterprise, volto maggiormente ad aziende di importanti dimensioni.

6.2 Creare il bucket Cloud Storage

Una volta creato un nuovo progetto su Google Cloud Platform, il primo passo è quello di creare un bucket su Google Storage.

Quindi dalla nostra home page di GCP, utilizzando la barra di ricerca, cerchiamo il servizio “Cloud Storage” e vi clicchiamo.

Sarà necessario inserire un nome univoco per il nostro bucket e selezionare la regione interessata, assicurarsi quindi di utilizzare la stessa zona della funzione cloud che andremo ad implementare successivamente; tutti gli altri campi sono facoltativi quindi in questo momento li tralasceremo.

- Assegna un nome al bucket**
 Scegli un nome univoco globale permanente. [Linee guida per l'assegnazione dei nomi](#)

 Suggerimento: non includere informazioni sensibili
 ✓ **ETICHETTE (FACOLTATIVE)**
- Scegli dove archiviare i tuoi dati**
 Località: eu (più regioni nell'Unione europea)
 Tipo di località: Multi-region
- Scegli una classe di archiviazione predefinita per i tuoi dati**
 Classe di archiviazione predefinita: Standard
- Scegli come controllare l'accesso agli oggetti**
 Prevenzione dell'accesso pubblico: On
 Controllo dell'accesso: Uniforme
- Scegli come proteggere i dati degli oggetti**
 Strumenti di protezione: Nessuno
 Crittografia dei dati: Google-managed key

Buono a sapersi
Prezzi per ubicazione
 Le tariffe di archiviazione variano in base alla classe di archiviazione dei dati e all'ubicazione del bucket. [Dettagli dei prezzi](#)
 Configurazione attuale: Multi-region / Standard

Elemento	Costo
eu (più regioni nell'Unione europea)	\$0.026 per GB/mese

 STIMA IL COSTO MENSILE

Figura 6.2: Creazione bucket su Cloud Storage

6.3 Creare la Cloud Function

Il prossimo passo è quello di creare la Cloud Function, come visto già nei precedenti articoli ricerchiamo il servizio “Cloud Functions” e facciamo clic su “crea funzione”.

Una volta inserito il nome e la regione, dovremo selezionare il tipo di trigger: in questo caso andremo a selezionare “Cloud Pub/Sub”, come verrà mostrato nella prossima immagine.

Si aprirà quindi una nuova finestra, fare clic su “Crea argomento” per creare un nuovo argomento Pub/Sub che attiverà questa Cloud Function;

Una volta assegnato un nome appropriato anche all’argomento trigger, fare clic su “salva” per confermarne la creazione.

Concetti di base

Ambiente
1ª gen. ▼ ⓘ

Nome funzione *
function-write ⓘ

Regione
europe-west1 ▼ ⓘ

Trigger

⚡ Cloud Pub/Sub

Tipo di trigger
Cloud Pub/Sub ▼

Seleziona un argomento Cloud Pub/Sub *
projects/schedulaprocessi/topics/write_trigger ▼

☐ Riprova in caso di errore ⓘ

SALVA **ANNULLA**

Crea un argomento

ID argomento *
write_trigger ⓘ
Nome argomento: projects/schedulaprocessi/topics/write_trigger

☐ Utilizza uno schema ⓘ

☐ Imposta la durata di conservazione dei messaggi (non gratuita) ⓘ

Crittografia

☒ Chiave di crittografia gestita da Google
Nessuna configurazione richiesta

☐ Chiave di crittografia gestita dal cliente (CMEK)
Gestisci tramite Google Cloud Key Management Service

Figura 6.3: Creazione Cloud Function Pub/Sub

In questo caso tralasceremo gli altri campi, poichè si tratta di una funzione molto semplice, quindi riguardo impostazioni di runtime, build e connessioni utilizzeremo quelle che Google imposta di default. L'unico punto sui cui soffermarci sarà quello delle "Connessioni", qui potremmo decidere di impostare "consenti solo traffico interno" per evitarci chiamate esterne dannose e inaspettate.

6.3.1 Codifica della funzione

Ora si aprirà la finestra dell'editor di testo e del codice sorgente; Qui indicheremo il nostro ambiente di runtime e scriveremo il codice della nostra funzione. Utilizziamo Python come linguaggio di programmazione.

Il nostro progetto è composto da due file:

- file `main.py`: il file dove risiede tutto il codice della nostra funzione e viene eseguito con l'evento trigger. Siccome abbiamo selezionato di creare una funzione Pub/Sub, ci troviamo preimpostata la funzione **`hello pubsub(event, context)`** ovvero la funzione utilizzata come punto di ingresso.
Se si decide di modificare il nome della funzione bisogna assicurarsi di andare anche a cambiare il nome del punto di ingresso, poichè deve essere identico.
- file `requirements.txt`: dove inserire dipendenze e librerie, nel nostro caso sarà necessaria solamente una riga per agganciarci a Cloud Storage.

Codice main.py

```
.....

import base64
from google.cloud import storage
from datetime import datetime

def upload_blob(bucket_name, source_file_name,
                destination_blob_name):
    # Per Caricare il file su bucket

    storage_client = storage.Client()
    bucket = storage_client.bucket(bucket_name)
    blob = bucket.blob(destination_blob_name)

    blob.upload_from_filename(source_file_name)

def create_file_name():
    # Generare il file txt con orario corrente

    date = datetime.now().strftime("%Y_%m_%d-%H_%M_%S_%P")
    file_name = 'print_' + date + '.txt'
    return file_name

def write_to_file(file_name):
    # Creare il file txt con il testo all'interno
    # Attenzione -> in cloud function possiamo
    # scrivere solo nella cartella /tmp

    file_name = '/tmp/' + file_name

    with open(file_name, 'w') as f:
```

```
f.write('some text')

f.close()

def hello_pubsub(event, context):
    # Trigger per il messaggio Pub/Sub

    file_name = create_file_name()

    # Scrivi testo e salva file
    write_to_file(file_name)

    # Salva file sul bucket
    bucket_name = 'process_bucket_yuri'
    local_file_location = '/tmp/' + file_name
    upload_blob(bucket_name, local_file_location, file_name)

.....
```

Codice requirements.txt

```
.....

google-cloud-storage==1.30.0

.....
```

Una volta composti i due file sarà necessario effettuare il deploy della funzione, la prima volta potrebbe volerci qualche minuto. Quando sarà tutto eseguito correttamente visualizzeremo una spunta verde a fianco del nome, come mostrato nella prossima immagine.



The screenshot shows the Google Cloud Functions console. At the top, there are tabs for 'Cloud Functions' and 'Funzioni'. Below the tabs, there are buttons for 'CREA FUNZIONE' and 'AGGIORNA'. A filter section is visible with the text 'Filtro Funzioni di filtro'. Below this is a table with columns: Ambiente, Nome, Ultimo deployment, Regione, Trigger, Runtime, Memoria allocata, Funzione eseguita, Authentication, and Azioni. The table contains one row with the following data: Ambiente (1st gen), Nome (function-write), Ultimo deployment (23 ott 2022, 10:45:53), Regione (europe-west1), Trigger (Argomento: write_trigger), Runtime (Python 3.8), Memoria allocata (256 MB), Funzione eseguita (hello_pubsub), Authentication (icon), and Azioni (three dots icon).

Ambiente	Nome	Ultimo deployment	Regione	Trigger	Runtime	Memoria allocata	Funzione eseguita	Authentication	Azioni
1st gen	function-write	23 ott 2022, 10:45:53	europe-west1	Argomento: write_trigger	Python 3.8	256 MB	hello_pubsub		

Figura 6.4: Risultato deploy

6.3.2 Test della funzione

Dato che la nostra funzione non richiede dati in input possiamo fare clic direttamente online sul pulsante “Esegui test funzione”.

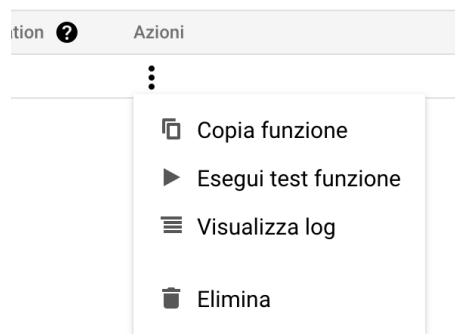


Figura 6.5: Test funzione

Se risulta tutto corretto la funzione dovrebbe stampare “OK” sull’output dei registri altrimenti sarà necessario leggere l’errore che appare ed andarlo a risolvere.

A questo punto non resta che pianificarne l’esecuzione periodica.

6.4 Pianificazione Cloud Function

Ritorniamo nella console principale di Google Cloud Platform, ricerchiamo tramite la barra di ricerca il servizio “Cloud Scheduler” e creiamo un nuovo job seguendo i passi mostrati a seguito.

Figura 6.6: Cloud Scheduler

Naturalmente inseriamo un nome, questo deve essere univoco all'interno della regione di utilizzo;

Impostiamo la frequenza, in questo caso vogliamo che la mia Cloud Function sia evocata ogni 15 minuti, quindi inseriremo la stringa: “*/15 * * * *” poichè è necessario utilizzare il formato universale unix-cron.

Assicuriamoci di impostare il corretto fuso orario.

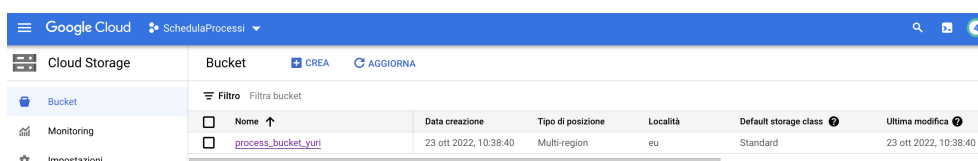
Ora per concludere la configurazione dobbiamo selezionare il tipo target Pub-Sub e come argomento quello della nostra Cloud Function che abbiamo precedentemente creato.

Infine, è necessario inserire il corpo del messaggio anche se non verrà elaborato.

Ora selezioniamo “crea” per pianificarne l’esecuzione ogni 15 minuti.

È anche possibile lanciare immediatamente l’esecuzione cliccando su “esegui ora”.

Ora e successivamente ogni 15 minuti sarà eseguito l’intero progetto e potremo visualizzare nel nostro bucket cloud un nuovo file che conterrà i nostri risultati.



The screenshot shows the Google Cloud console interface. On the left, there's a sidebar with 'Cloud Storage' selected. The main area shows a table of buckets. One bucket is listed: 'process_bucket_yuri'. The table has columns for 'Nome', 'Data creazione', 'Tipo di posizione', 'Località', 'Default storage class', and 'Ultima modifica'.

Nome	Data creazione	Tipo di posizione	Località	Default storage class	Ultima modifica
process_bucket_yuri	23 ott 2022, 10:38:40	Multi-region	eu	Standard	23 ott 2022, 10:38:40

Figura 6.7: Visualizzazione file nel bucket

Attenzione: cosa molto importante è quella di stoppare l’esecuzione di tutti i servizi attivi se non ne abbiamo necessità.

Tutti i servizi Cloud rimarranno sempre in esecuzione in background finché non la termineremo manualmente; potremmo, quindi, incorrere a costi indesiderati ed eccessivi oltre che chiamate da fonti indesiderate.

6.5 Resoconto e possibili implementazioni

Solitamente chi decide di utilizzare servizi cron pensa di implementare macchine virtuali, ma ormai vale la pena rivisitare questi compiti e pensare di implementarli utilizzando funzioni cloud.

Dal punto di vista informatico e implementativo questo breve programmino può avere diverse applicazioni e innumerevoli implementazioni, per esempio si può decidere di creare sistemi che elaborino regolarmente dei file csv, che

eseguano previsioni di modelli periodicamente, creare file che generino tabelle di riepilogo giornaliere, caricare file esterni su grandi tabelle di query ogni giorno, ecc. . . solo per citarne alcuni.

Inoltre, ricordiamoci che siamo dentro ad un'architettura serverless che consente molta flessibilità e minimo sforzo.

Conclusioni

Arrivato alla fine di questo elaborato di tesi che mi ha impegnato per quasi l'intero anno, posso reputarmi soddisfatto del lavoro svolto.

Sono contento di aver scelto questo argomento perchè ammetto che fossero concetti a me lontani e parzialmente sconosciuti con cui non avevo mai lavorato o studiato prima d'ora.

Grazie anche all'aiuto del relatore sono riuscito a intraprendere una nuova strada e interessarmi allo studio di questi argomenti.

Credo possa essere una buona guida introduttiva per coloro che vogliono conoscere i prodotti serverless disponibili su Google Cloud Platform.

Naturalmente non sono mancate le difficoltà, da un lato dovute alla poca diffusione del servizio e alla sua recente nascita, dall'altro dovuto al fatto di essere un prodotto “di nicchia”.

Il mio obiettivo con questo elaborato è quello di semplificare la strada per coloro che decidano di cimentarsi nelle Google Cloud Function partendo da zero.

Trattandosi di servizi molto recenti e molto potenti, sono anche prodotti “pay as use” disponibili, quindi, a pagamento e riservati solamente alle aziende che decidano di investirci parecchie risorse.

La mia speranza nell'immediato futuro è che Google decida di creare istanze pubbliche e gratuite di questi servizi per poterli utilizzare anche in ambienti didattici e far sì che i vari utenti possano intraprendere questo percorso gratuitamente in autonomia.

Ho utilizzato più volte l'aggettivo “di nicchia” all'interno dell'elaborato, perchè

penso che siano prodotti molto dispendiosi e ancora sconosciuti, sia dal punto di vista economico ma anche riguardo al tempo da dedicarvi.

Le difficoltà dovute alla recente nascita del servizio hanno fatto sì che le uniche guide disponibili che sono riuscito a trovare siano quelle provenienti dai canali ufficiali di Google e gli esempi quasi tutti provenienti dal medesimo host.

Questi ultimi sono molto complessi e richiedono l'agganciamento ad altri servizi a pagamento, non facilmente assimilabili da un privato che intende farne un utilizzo a scopo personale, di ricerca o test.

Durante la stesura di questo documento Google ha pubblicato diversi aggiornamenti riguardo le Cloud Function, sia piccoli che di grande importanza, come lo è stata la seconda generazione.

Spero che in futuro si possa arrivare ad una linea omogenea così che le guide e le fonti disponibili siano universali ed adatte alle esigenze dei vari utenti. Nel mio caso, infatti, mi sono trovato di fronte ad una vera rivoluzione di funzioni e proprietà, di conseguenza ho dovuto nuovamente redigere il documento e cambiarlo a causa delle innovazioni introdotte da un mese all'altro. Mi auguro che questo documento possa aprire la strada a coloro che, come me, avevano poca confidenza e/o conoscenza di questo ambiente e che questo possa essere l'inizio di innovative implementazioni ed utilizzi di GCP con Google Functions.

Bibliografia

- [1] Techmagic: Artem Arkhipov. Top 5 serverless platform in 2022.
- [2] Ted Hunter and Steven Porter. *Google Cloud Platform for developers: build highly scalable cloud solutions with the power of Google Cloud Platform*. Packt Publishing Ltd.
- [3] New Business Media. Esegui il codice senza gestire il server.
- [4] RedHat. Tipologie di cloud computing.
- [5] News.srl: Simone Catania. Che cos'è saas e perché è considerata l'industria del futuro.
- [6] Google. Functions.
- [7] Google Online Bootcamp. Deploy functions in python.
- [8] Toward Data Science. How to schedule a serverless google cloud function to run periodically.

Ringraziamenti

Vorrei dedicare qualche riga a coloro che hanno contribuito alla realizzazione della mia tesi di laurea.

Innanzitutto, ringrazio il mio relatore Professore Vittorio Ghini, per avermi dato la possibilità di lavorare con lui ed avermi seguito passo passo nella realizzazione della tesi.

Dedico queste ultime righe a coloro che per me ci sono sempre stati ovvero la mia famiglia e la mia ragazza; grazie a loro sono riuscito a superare ostacoli e difficoltà ed arrivare fin qui.

Infine, dedico questa tesi a me stesso, ai miei sacrifici e alla mia tenacia che mi hanno permesso di concludere questo percorso con felicità e soddisfazione.