AWS CLOUDFORMATION: I PRIMI 5 MOTIVI PER NON USARLO

AWS CloudFormation

beSharp | 3 Maggio 2019

Come sopravvivere alle più comuni trappole ed ottenere il meglio dal servizio, senza troppi mal di testa.

A chi non è capitato di avere problemi con AWS CloudFormation?

È tristemente noto come AWS CloudFormation sia uno strumento tanto potente quanto frustrante; in questo breve articolo illustreremo le trappole più comuni e le situazioni più odiose, insieme ovviamente a modi per evitarle.

Iniziamo il nostro viaggio dalla descrizione del servizio:

"AWS CloudFormation ti fornisce un linguaggio comune per descrivere ed effettuare il provisioning di tutte le risorse dell'infrastruttura nel tuo ambiente cloud. Con CloudFormation puoi usare un semplice file di testo per modellare ed effettuare il provisioning, in modo automatizzato e sicuro, di tutte le risorse necessarie alle tue applicazioni su tutte le regioni e tutti gli account. Questo file sarà l'unica sorgente del tuo ambiente cloud.

CloudFormation è disponibile senza alcun costo aggiuntivo; ti vengono addebitati solo i costi delle risorse AWS necessarie per l'esecuzione delle applicazioni."

Uno dei principali vantaggi di utilizzare AWS CloudFormation è appunto quello di disporre di **un template della propria infrastruttura, che può essere versionato, revisionato, e replicato** facilmente tutte le volte necessarie.

Tuttavia già dalla descrizione del servizio iniziano i primi problemi.

#1 Una modellazione ~~completa~~ incompleta

Nonostante la pagina della presentazione del servizio AWS dica esplicitamente che AWS CloudFormation offra una modellazione completa delle infrastrutture Cloud, questo di fatto non accade.

Non sempre, almeno; **esistono infatti molte configurazioni e/o risorse che non è possibile specificare all'interno di un template** AWS CloudFormation.

Ad oggi, ad esempio non è possibile:

- definire i settaggi riguardanti Cognito UserPool per federazione con un IDP esterno
- aggiungere una rotta ad una routing table che punta ad un Transit Gateway
- fare il provisioning di SSH key per EC2 (probabilmente per ragioni di sicurezza)

Il nostro consiglio è quello di assicurarsi che tutti i servizi e le configurazioni necessarie siano veramente modellizzabili con la versione attuale.

Quasi tutti i servizi vengono supportati, anche se per alcuni è necessario aspettare un notevole lasso di tempo tra il rilascio ed il supporto ufficiale di AWS CloudFormation.

Esiste poi il modo di estendere le capacità di AWS CloudFormation utilizzando le custom resources. Di fatto permettono di effettuare il deploy di una Lambda function e di eseguirla come "Custom Resource" in un determinato momento durante la creazione dello stack.

Ecco la documentazione ufficiale di questa funzionalità.

#2 Tempi di attesa

Se è vero che una volta confezionato il template l'esecuzione è più rapida di quella manuale, il processo di produzione del template è un processo lungo e basato su trial and error.

Non è infatti possibile controllare al 100% un template in locale, è possibile solo essere certi della correttezza sintattica senza prima eseguirlo.

Il che significa che prima di giungere ad una versione funzionante si deve passare per numerosi tentativi, ognuno dei quali ha tempi di esecuzione lunghi.

Per rendersi la vita più semplice **possiamo contare su alcuni alleati**:

Il primo è l'API ValidateTemplate che fa un controllo molto più approfondito di un semplice linter.

Un altro modo è l'uso estensivo dei changeset, che permettono di avere un'anteprima delle azioni che scaturiranno alle modifiche ad un template.

Inoltre si consiglia di procedere facendo variazioni piccole e di rapida esecuzione. Prima creare tutte le risorse che richiedono un basso tempo di provisioning, e solo alla fine aggiungere allo stack le risorse più onerose in termini di tempo, come ad esempio istanze EC2, RDS e simili. Così facendo

i tempi di attesa e di validazione su tutte le risorse che sono probabilmente richieste dai servizi a provisionig saranno più brevi.

#3 Errori incomprensibili

I messaggi di errore sono spesso senza senso, fuorvianti, inutili o semplicemente sbagliati.

Si può solo essere certi che lo stack abbia fallito, cosa abbia fallito e perché è una informazione tanto utile quanto difficile da estrapolare

L'unico modo per trovare rapidamente la ragione di un fallimento è **analizzare il log di esecuzione** per vedere esattamente su quale risorsa si riscontra errore, e in che momento quella risorsa è stata provisionata.

#4 Cloudformation drift detection

Il drift detection di CloudFormation è stato richiesto a gran voce da molti degli utenti, e consiste nella capacità di **rilevare automaticamente se sono state apportate modifiche alla configurazione delle risorse dello stack** al di fuori di CloudFormation tramite la console di gestione AWS, la CLI e gli SDK

Si tratta di una feature molto utile, purtroppo in pratica dà molti falsi positivi. In generale indica che è probabile che qualcosa sia stato modificato esternamente e quindi che lo stack non sia più aggiornabile in modo automatico. Purtroppo non indica che lo stack non sia effettivamente modificabile.

Meglio non considerarlo come indicazione attendibile, almeno per ora.

#5 Hard limit a 200 risorse per stack

Non è possibile creare più di 200 risorse in un unico stack.

Molti sostengono che questo non sia affatto un problema, e che sia facilmente aggirabile con buon senso. In fondo, perché mai si dovrebbero creare più di 200 cose alla volta?

Il problema risiede tutto in cosa si definisce come "risorsa"; infatti ci sono molti oggetti che sono normalmente invisibili creando le risorse dalla web console. Per realizzare una semplice infrastruttura con una decina di lambda, un api gateway con relative policy e qualche risorsa aggiuntiva **si può facilmente raggiungere l'hard limit**.

I microservizi dovrebbero essere piccoli e un'applicazione complessa dovrebbe quindi comporsi di più stack CloudFormation... tuttavia anche avere molti stack diventa oneroso dal punto di vista di organizzazione e gestione, per cui meglio dosare bene la quantità di risorse da includere in un unico template ed elaborare una strategia per partizionare un'architettura complessa.

Questi sono i primi 5 motivi per non usare AWS Cloudformation. Beh, almeno sono 5 motivi per non usarlo a prescindere pensando che possa risolvere ogni cosa senza creare grattacapi.

Concludendo, AWS CloudFormation è potente ma insidioso, meglio giocare d'anticipo quando si inizia un nuovo progetto ed evitare gli errori più comuni.

Raccontaci la tua esperienza, commenta con l'errore più insidioso che hai riscontrato. Le storie migliori diventeranno protagoniste di un prossimo articolo sulle insidie di AWS CloudFormation.

beSharp

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

Get in touch

beSharp.it
proud2becloud@besharp.it