Creare App iOS con AWS Codebuild: Pro, Contro e la nostra soluzione alternativa

23 Ottobre 2024 - 7 min. read

AWS CodeBuild | *AWS CodeBuild MacOS Builds* | *CI/CD*

Introduzione

Avete mai dovuto creare un'applicazione iOS su Amazon Web Services?

Se la risposta è no, allora questo articolo fa per voi. Se la risposta invece è sì questo articolo fa comunque al caso vostro!
Recentemente siamo stati chiamati a realizzare per uno dei nostri clienti proprio un'applicazione iOS su AWS. Non avendo alternative al momento dell'inizio del progetto, abbiamo progettato una soluzione custom di cui avremmo voluto parlarvi proprio in questo articolo.

Il mondo AWS tuttavia è continua evoluzione e, nemmeno a farlo apposta, **da poco è disponibile un'immagine di Codebuild MacOS**.

Perché non esplorare subito questa nuova possibilità? Quali vantaggi porta? Come si compara con ciò che abbiamo fatto noi?

A costo di tardare con l'uscita dell'articolo, abbiamo deciso di integrare queste considerazioni ed eccoci qui con il contenuto che state leggendo ora.

Senza ulteriori premesse, entriamo nel vivo dell'argomento partendo proprio dalla soluzione di AWS.

'La maggior parte dei problemi citati qui sopra sono dovuti all'attuale immaturità del

servizio che, essendo appena uscito, ha ancora delle limitazioni; limitazioni che sicuramente col tempo verranno mitigate.'

AWS CodeBuild per MacOS

Con l'uscita dell'immagine Codebuild MacOS abbiamo adesso la possibilità di **integrare in una Pipeline una macchina con sistema operativo MacOS** senza dover creare metodi di integrazione custom e quindi facilitando di gran lunga la velocità di rilascio dell'infrastruttura. Questo è sicuramente il primo grande vantaggio della soluzione di AWS.

Ma vediamo con ordine tutti i pro e contro:

PRO

Serverless

Codebuild è già predisposto all'essere Serverless. Questo ci permetterebbe di avere sempre a disposizione una macchina per poter rilasciare le nostre applicazioni senza preoccuparci di un eventuale disservizio e senza dover gestire manualmente configurazioni per l'alta affidabilità.

Cosa, quest'ultima, che dovrebbe invece essere gestita in altre situazioni, per esempio avendo una seconda macchina in una differente 'Availability Zone'.

Gestione e costi

Codebuild ci offre la possibilità di utilizzare la stessa Fleet per diversi progetti, e quindi poter ripartire le spese su più applicazioni, diminuendone il singolo costo.

Sarà però necessario avere attenzione riguardo a come gestirne il setup e la sua configurazione in modo dinamico, così da utilizzare gli stessi script ad ogni lancio, cambiando solo delle variabili.

Questo porterà a lungo andare e su più progetti vantaggi sia economici, che gestionali.

CONTRO

Una sola immagine

Le immagini, o meglio, l'immagine disponibile è solamente una, con una versione ben specifica di XCode (Programma che serve per creare il file .IPA da poter poi rilasciare nell'app store).

Questo porta a diverse criticità:

- Non avere sempre a disposizione l’ultima versione del software
- Non avere tutti i programmi installati (Come IXGuard)
- Mantenere l’ambiente di sviluppo in locale allineato con quello in cloud limitando gli sviluppatori ad utilizzare una versione specifica e perdendo gli aggiornamenti che portano maggiore sicurezza e velocità.

Costi fissi

Questa versione specifica di CodeBuild ha una modalità di pagamento differente dal classico pay-per-use tipico delle immagini più comuni (Come quelle Linux).

La differenza deriva dall’avere come hardware una macchina MAC fisica, il che richiederà di dover creare una Codebuild Fleet; la quale avrà un costo anche se non utilizzata attivamente.

Questo non dovrebbe essere un grosso problema, dato che l’alternativa è una istanza MAC fisica...purtroppo però il costo di Codebuild è maggiore rispetto a quello di una istanza EC2, specialmente se con dei saving plan a lungo termine.

Complessità

Personalmente mi aspettavo fossero integrati alcuni script di uso comune per lo sviluppo di un'app all’interno dell’immagine di Codebuild. Sarebbe stato così più facile svolgere alcune operazioni necessarie, come la configurazione e l’installazione dei ‘provisioning profile’, i quali sono richiesti per rilasciare un’applicazione nell’AppStore.

È quindi necessario crearne alcuni in autonomia.

Da poco AWS ci fornisce un ulteriore strumento, ovvero la diretta integrazione tra CodeBuild e GitLab Runners. Questo potrebbe risultare utile quando vi è gia’ un ambiente GitLab per il proprio Software Version Control. In questo caso sarebbe quindi possibile integrare il tutto direttamente tra i due providers, riducendo il costo e gestendo con più facilità l’istanza tramite GitLab.

La nostra soluzione

"Ai nostri tempi...", ovvero qualche mese fa, non avevamo niente per poter gestire in modo autonomo la creazione di un'applicazione iOS, dunque abbiamo cercato di capire dove fosse possibile integrare automatismi e quali.

Tutto questo pensare, oltre che farci venire fame e sete, ci ha portato ad una soluzione 100% personalizzata che è attualmente in uso.

Siccome uno dei prerequisiti del cliente era **mantenere tutto sullo stesso ambiente AWS**, non abbiamo potuto utilizzare le GitHub actions, che avrebbero facilitato la nostra integrazione con l'ambiente iOS. Questo perché sono direttamente integrate con una macchina munita di ambiente iOS.

Oltre all'ambiente, gli altri vantaggi sono il maggior tempo per progredire come tecnologia ed essere raffinata e il maggiore supporto della community che quindi porta ad avere più esempi e integrazioni già fatte.

Detto ciò, siamo dovuti partire dal presupposto che l'unico modo per rilasciare un'applicazione iOS su AWS utilizzando servizi managed e senza importare strumenti di terze parti, come un'immagine Docker MacOS, fosse **utilizzare un'istanza EC2 MAC**.

Abbiamo dunque dovuto capire come integrare questa macchina dentro ad una Pipeline, dato che non vi sono integrazioni già fatte da AWS. La strada che abbiamo ritenuto essere la migliore è stata quella di utilizzare una **StepFunction** per orchestrare le varie operazioni necessarie per costruire e rilasciare l'applicazione.

L'idea era quella di controllare se la singola macchina fosse già in utilizzo da qualche altro ambiente e, nel caso, attendere che la build fosse finita così da evitare di raddoppiare i tempi di rilascio per entrambi;

Il dubbio successivo riguardava come comunicare direttamente con la macchina tramite questa StepFunction.

Confrontando i requisiti con le nostre possibilità siamo giunti ad una conclusione interrogando la macchina tramite SSM con l'API di 'SendCommand'.

I comandi servivano per poter lanciare i vari script .SH necessari per i vari step di predisposizione della macchina e di rilascio dell'applicazione.

Come abbiamo fatto per la soluzione di AWS, vediamo Pro e Contro anche della nostra:

PRO

Personalizzazione

Una soluzione costruita da zero, come tutte le cose, ci permette di crearla come più ci aggrada e con ciò che riteniamo davvero utile. In questo caso, ad esempio, siamo stati in

grado di aggiungere altri applicativi come IXGuard e mantenere sempre la versione più aggiornata di XCode così da facilitare anche il lavoro degli sviluppatori e andare in contro alle esigenze del cliente.

Vantaggi economici

Il costo risulta essere più basso nel confronto singola istanza e singolo CodeBuild, grazie all'utilizzo di molti servizi serverless, come AWS Lambda e AWS Step Function. Mentre il costo dell'EC2 può essere abbattuto dai Saving Plan.

CONTRO

Overhead gestionale

Sicuramente gestire la logica di integrazione con l'istanza EC2 è molto più complicato rispetto a utilizzare una soluzione Out Of The Box come AWS CodeBuild fornitoci da AWS, ma al contempo offre la possibilità di avere una nostra macchina e una nostra integrazione ci permette di avere un controllo e flessibilità ben maggiori come indicato sopra.

Singola AZ

L'istanza EC2 è una singola istanza fisica in una singola AZ e ciò va contro i principi dell'alta disponibilità; nel caso in cui una AZ dovesse andare fuori servizio la nostra macchina non potrebbe essere più utilizzata provocando disservizio.

Gestione degli aggiornamenti

Un'altra problematica è il caso in cui ci fosse la necessità di eseguire un update, o upgrade del sistema operativo o di un applicativo; si andrebbe certamente incontro a disservizio. Questo perché dopo un aggiornamento di una 'Major Version' qualche funzionalità potrebbe cambiare e portare a degli errori imprevisti, richiedendo dunque del tempo per risolverli, dato che non è possibile prevederli prima dell'upgrade.

Per risolvere la problematica relativa gli aggiornamenti abbiamo trovato molto utile avere **una seconda istanza spenta con una AMI preconfigurata e allineata con la macchina di produzione**. In questo modo, potrà essere utilizzata per testare gli aggiornamenti e poi decidere con consapevolezza come procedere, essendo gli aggiornamenti sempre pianificabili.

È anche possibile lasciare l'istanza sempre operativa, ma questo comporta dei costi non trascurabili; dipende quindi da cosa ritenete essere più importante.

Conclusioni

Esiste una soluzione migliore rispetto all'altra?

Forse...ma non l'abbiamo trovata. La risposta, come sempre, è "dipende": se cercate una soluzione rapida e ben integrata, CodeBuild potrebbe essere la scelta giusta. Se invece avete bisogno di controllo e flessibilità, l'istanza EC2 con Step Functions e Lambda potrebbe essere più adatta.

Entrambe le soluzioni, sia la nostra che quella di AWS, comunque richiedono la **creazione e la gestione degli script** necessari per installare i prerequisiti, per poter creare il file .IPA e per poterlo poi rilasciare.

Questa risulta essere la parte più complessa, laboriosa e di difficile gestione.

Se volete qualche consiglio o esempio fatecelo sapere. Sarebbe una perfetta "Parte II" di questo articolo!

Per il momento è tutto.

Arrivederci al prossimo articolo su Proud2beCloud!

About Proud2beCloud

Proud2beCloud è il blog di beSharp, APN Premier Consulting Partner italiano esperto nella progettazione, implementazione e gestione di infrastrutture Cloud complesse e servizi AWS avanzati. Prima di essere scrittori, siamo Solutions Architect che, dal 2007, lavorano quotidianamente con i servizi AWS. Siamo innovatori alla costante ricerca della soluzione più all'avanguardia per noi e per i nostri clienti. Su Proud2beCloud condividiamo regolarmente i nostri migliori spunti con chi come noi, per lavoro o per passione, lavora con il Cloud di AWS. Partecipa alla discussione!

Stefano Salvaneschi

DevOps Engineer @ beSharp. Sono la classica persona che non accetta che una domanda possa rimanere senza risposta. Sia quando ne faccio (molto spesso!), sia quando ne ricevo. Appassionato di IT fin da quando ero un bambino, sono da poco approdato nel mondo del Cloud.

Nel tempo libero amo giocare ai videogames, a dungeons & dragons e ad altri giochi di ruolo tipicamente "nerd"... il tutto accompagnato da una buona birra!
