

[Home](#) > [Architecting](#)

Nightmare Infrastructures: come sopravvivere a un'apocalisse zombie sul Cloud

28 Ottobre 2022 - 10 min. read

[AWS Well-Architected](#)

[Cloud adoption](#)

[DevOps](#)

Jack Skeleton, in "The Nightmare Before Christmas", cantava: "Cos'è? Cos'è? Ma che colore è?"

E di fronte a certe infrastrutture, è capitato di dirlo anche a noi.



Potreste già aver capito che, in questo articolo, andremo a descrivere alcune strane infrastrutture con cui abbiamo avuto a che fare nel passato, raccontando storie spaventose su pattern e pratiche per nulla appropriati per il cloud e che, a lungo andare, hanno trasformato innocue infrastrutture in Zombie assetati di cervelli, pronti ad invadere il Cloud.

Non è nostra intenzione puntare il dito e deridere nessun design in particolare: i requisiti e le condizioni al contorno possono far sì che alcune soluzioni siano pressoché obbligatorie; il nostro intento è farvi fare una risata: il mondo là fuori è già abbastanza spaventoso!

(Se siete nostri clienti, state leggendo questa storia e vi riconoscete: sì, stiamo parlando di voi ! :D)

Non vi resta che andare a prendere i pop corn e la vostra coperta preferita: stiamo per iniziare i nostri racconti horror!



TestProdDev

"La strada per l'inferno è lastricata di buone intenzioni"



Cosa c'è di male nell'applicare un piccolo fix direttamente in produzione perché non c'è tempo di provarlo nell'ambiente di dev o di test ?

Ovviamente la patch rimarrà solamente in produzione, facendo in modo che gli altri ambienti siano disallineati. Dopo qualche tempo, tutti si saranno dimenticati di quel piccolo, insignificante pezzo di codice, che rimarrà in silenzio ad aspettare il momento giusto per gustarsi la vendetta: al rilascio di una nuova funzionalità sicuramente

emergerà un bug solo nell'ambiente di produzione causato dalla modifica (la legge di Murphy è sempre valida).

Siccome si tratta dell'ambiente di produzione, un altro "piccolo" fix sarà rilasciato, aumentando il disallineamento.

La storia si ripeterà fino a che ogni nuovo sviluppo sarà applicato direttamente all'ambiente di produzione perché non ci sarà modo di essere sicuri che il software possa essere provato in modo affidabile negli altri ambienti. Siamo "finalmente" arrivati all'ambiente di **testproddev**.

Il vero incubo è riuscire a capire dove le cose smettono di funzionare e perché a volte funzionano! In altri casi esistono ambienti di test, ma che non sono propriamente di test.

Parafrasando Boskov: *"Produzione è quando cliente urla"*

Tutto ha inizio un Martedì mattina, con un drift nell'ambiente di test. Dopo aver valutato come risolvere il disallineamento ed aver applicato le modifiche su CloudFormation, si è materializzato l'incubo "Update Rollback Failed". Una ulteriore analisi ci ha fatto scoprire il problema alla base di tutto: qualcuno ha cancellato una task definition di un container Fargate ECS, che è stato poi aggiornato da CloudFormation. In questi casi l'unica soluzione possibile è cancellare l'intero stack e ricrearlo da zero.

"Nessun problema", abbiamo pensato: per fortuna siamo nell'ambiente di test e stiamo usando IAC, cosa mai potrebbe andare male?

Cinque minuti dopo aver cancellato lo stack, i nostri Slack in ufficio hanno cominciato a suonare all'unisono: l'ambiente di "Test" era usato per ospitare servizi usati da partner del cliente. In questo caso la fortuna ci ha aiutato: tutti i dati erano salvati nei backup e in soli 15 minuti l'infrastruttura è stata ripristinata da zero.

Cosa abbiamo imparato? Mai fidarsi di qualcosa che ha "test" nel nome.

In qualsiasi caso: se un ambiente è essenziale per il business è sempre meglio considerarlo tale usando nomi come "produzione", "production", "prod" o qualcosa che comunque significhi: "Maneggiare con cura, attenzione!"

Il Cloud semplifica il deploy delle infrastrutture: usiamo questa caratteristica a nostro vantaggio!

Tutto può essere automatizzato usando IAC, i costi diventano un problema marginale: con il modello di pagamento on-demand si possono fare esperimenti e cancellare tutto alla fine dei test, minimizzando le spese. Usare delle pipeline per Continuous Integration e Continuous Deployment fa in modo che tutto sia sincronizzato.

AWS CloudFormation, CodeBuild, CodePipeline e CodeDeploy sono servizi chiave per fare sì che tutto sia aggiornato e coerente in ambienti differenti.

Microservizi... giganti

Se affermo che la mia immaginazione, alquanto stravagante, produsse le visioni simultanee di un polipo, di un drago e di una caricatura umana, non sarò infedele allo spirito della cosa. Una testa polposa, tentacolare, sormontava un corpo grottesco e squamoso, munito di ali rudimentali; ma era il profilo generale del tutto che lo rendeva sconvolgente e spaventoso in massimo grado. Alle spalle della figura si intuiva vagamente uno sfondo architettonico di dimensioni ciclopiche. - H.P. Lovecraft - Il richiamo di Chtulu



Un microservizio è, per definizione, codice che funziona in modo cooperativo con altri elementi, permettendo ai team di sviluppo di concentrarsi maggiormente su singoli problemi. Ad esempio, un microservizio può occuparsi dell'invio di email ai clienti, mentre un'altro farà sì che l'inventario sia aggiornato a fronte di un pagamento andato a buon fine.

Un design a microservizi fa sì che il software sia facilmente manutenibile ed occupi poche risorse computazionali.

Ci è capitato di trovare un container Docker con allocati 8 gigabyte di ram e 4 cpu: il tutto per far funzionare una applicazione Liferay usata come CRM, gestore di pagamenti con relativo invio di mail. In questo caso non si trattava di un microservizio, ma di un elefante che tentava di guidare una 500!

Alcuni framework enterprise tendono ad essere per loro natura monolitici. Se necessitate di tecnologie di questo genere non fatevi indurre nella tentazione di usare container solamente per avere una strategia di deploy più semplice: non è così. Le istanze EC2 con Autoscaling Group non sono alla moda come i container Docker e le Lambda, ma possono essere la soluzione migliore in casi analoghi.

Per citare il Tenente Combo: "un'ultima cosa": no, **usare un cluster EKS o, peggio ancora, un cluster Kubernetes su EC2 non fa sì che si stiano usando i microservizi.** State "semplicemente" aggiungendo complessità e costi alla vostra infrastruttura.

Essere "Cloud-native" è più semplice quando si sviluppa qualcosa da zero. Trasferire una applicazione su AWS con un lift and shift senza adattarla al paradigma cloud non rende il software cloud-ready, scalabile, né altamente affidabile.

Istanze EC2 per contenuti web statici

"Abominevole! Ahah! Non è incredibile? A voi io sembro "abominevole"? E perché non potevano chiamarmi "l'adorabile" uomo delle nevi o "il simpatico" uomo delle nevi, per la miseria! Io sono un tipo socievole!"



Ammettiamolo: a volte la tentazione di usare una tecnologia che già padroneggiamo al posto di servizi gestiti meno conosciuti è forte. Un esempio tipico è un server Apache per ospitare un sito web statico.

Questo approccio, però, fa crescere la manutenzione ed anche la bolletta AWS mensile, anche senza usare un Autoscaling Group ed un Load Balancer per rendere la soluzione altamente affidabile.

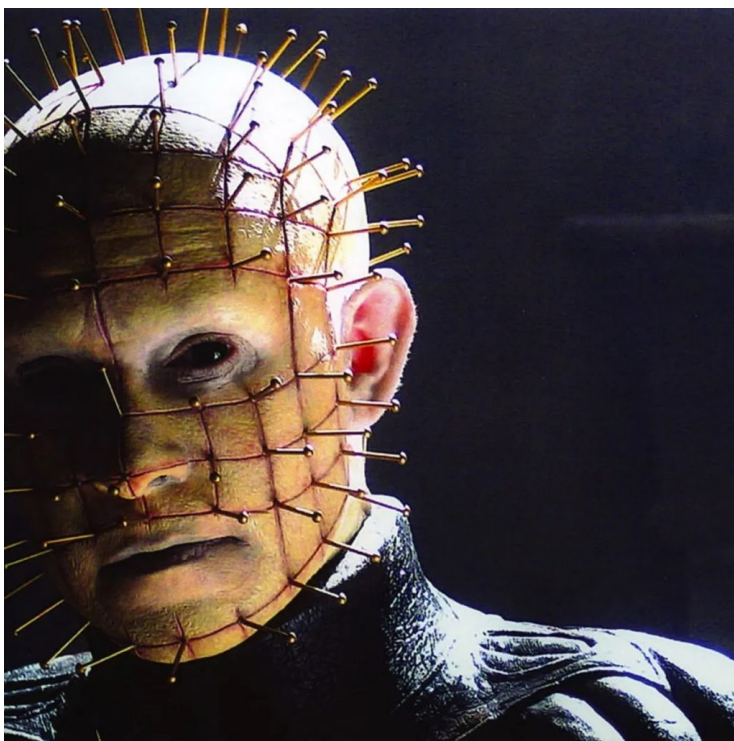
Ci è capitato di dover fare modifiche alla configurazione di due webserver Apache che si occupavano di servire pagine statiche.

I due server, per ragioni storiche e di SEO, avevano una configurazione con 60.000 regole di rewrite, combinate con 13.000 condizioni. Una quantità enorme di regole era ovviamente ridondante e interferiva con le altre. A seconda della pagina la risposta poteva richiedere anche 5 secondi prima di essere servita, nonostante le istanze EC2 utilizzate fossero due m5.2xlarge. Quando qualcosa andava storto, abilitare il log generava circa 2.000 righe la cui analisi non era semplice.

Usando CloudFront ed S3 il risparmio sarebbe stato considerevole, con il vantaggio di avere a disposizione una Content Distribution Network globale. Per la gestione dei redirect sono sufficienti una Lambda@edge ed una tabella Dynamo.

Contenuto statico generato sul backend (con finale a sorpresa)

"Niente lacrime per favore; non si deve sprecare così la sofferenza." - PinHead, Hellraiser.



Quando il backend genera pagine statiche, sprechiamo cicli di CPU, soldi ed energia. Per ridurre il carico di un application server monolitico, si può provare ad usare

CloudFront.

Una cosa che non ci stancheremo mai di dire: per favore, fate in modo che script e utility (o anche rotte applicative) non siano messe in sottocartelle che potrebbero essere raggiunte ed indicizzate per sbaglio dai motori di ricerca (anche configurare robots.txt in modo che siano esclusi spesso non è una soluzione).

Abbiamo ricevuto la telefonata di un cliente finito nel panico perché la notte il database di produzione era stato cancellato.

Dopo aver investigato, abbiamo notato l'esistenza di uno script PHP (utility.php) usato dagli sviluppatori per resettare velocemente l'ambiente di sviluppo locale. Per rendere le cose più veloci, era in una directory il cui contenuto era visualizzabile e, per attivarlo, bastava una semplice GET.

Il deploy dell'applicazione era fatto a mano copiando i file sul server, per cui una svista ha fatto sì che la cartella con le utility finisse online. Quando un motore di ricerca ha scoperto il nuovo percorso, ovviamente, ha richiesto il file utility.php con una GET e... Il database di produzione ha smesso di esistere!

Igor in Frankenstein Junior avrebbe detto: *"Potrebbe essere peggio, potrebbe piovere!"*

Servizi unmanaged: lo stateless stateful

"Si paga per quel che si ottiene, si ottiene ciò per cui si paga... e prima o poi quel che ti appartiene torna a te." - Stephen King, It



Una installazione MySQL su Docker sembra una buona idea per gestire gli aggiornamenti e tenere allineate le versioni fra l'ambiente locale (gestito con docker-compose) e l'ambiente di produzione.

Cosa c'è di meglio di una istanza EC2 con Docker a bordo per semplificarci la vita?

Un container, per definizione, è stateless. Quindi, per conservare i dati, dovremo definire dei volumi. Ovviamente i dati devono essere salvati, quindi occorre un sistema di backup consistente (che non può quindi essere un semplice snapshot). A questo punto dovremo iniziare a sviluppare script di backup e ad usare job di cron per la pianificazione.

In breve tempo ci ritroveremo a spendere più tempo per gestire la nostra installazione: ad ogni cambio di versione di mysql andrà fatto l'aggiornamento delle tabelle avviando le utility all'interno del container con docker exec, lo spazio su disco andrà aumentato, i limit dei container andranno adattati...

Se calcoliamo il Total Cost of Ownership (TCO) ci rendiamo conto che un database Aurora RDS in configurazione Multi-AZ è molto più conveniente!

A questo proposito, il nostro Alessio ha scritto un [ottimo articolo su servizi stateful e stateless](#).

Lo stateless stateful: il ritorno.

"Errare humanum est, perseverare autem diabolicum" - Sant' Agostino



Ok, metteremo i dati su un filesystem condiviso! EFS è una buona soluzione, può scalare, ha i backup integrati ed ha un modello di pagamento a consumo.

In questo caso, abbiamo visto girare una applicazione PHP su container Docker in istanze ECS (in autoscaling group) con tutto il codice applicativo ospitato su EFS.

Sappiamo che a volte un filesystem condiviso è necessario per ospitare dati condivisi (come ad esempio [in questo caso](#)).

Una applicazione PHP, anche se piccola, non può però essere ospitata su EFS a causa del modo in cui l'interprete interagisce con il filesystem (e perché le IOPs saranno molto minori rispetto ad un disco EBS).

Questi consigli da parte di AWS possono aiutare, ma i risultati non sono garantiti:

- [Optimizing WordPress performance with Amazon EFS](#)
- [Why is my EFS file system performance slow?](#)

La vita è molto più semplice quando i file applicativi sono memorizzati direttamente sul container, i file statici sono ospitati su S3 e l'esecuzione si appoggia su di un cluster ECS Fargate !

Lo stateless stateful: la resurrezione.

"Per me si va ne la città dolente,

per me si va ne l'eterno dolore,

per me si va tra la perduta gente"

Inferno, Canto III - Dante



Ok, seguiamo i vostri consigli, quindi useremo ECS Fargate per fare il deploy di... un Cluster Cassandra! Fargate è interamente gestito, quindi non avremo problemi di sicuro.

Un task Fargate ha storage **effimero**. Di default, lo spazio allocato è 20Gb, può essere aumentato fino a 200 (così possiamo perdere più dati quando il container si riavvia...). Usare EFS è comunque una scelta non corretta: le performance potrebbero non essere affidabili se non si usa la modalità provisioned (che però aumenta i costi in modo considerevole).

In aggiunta a questo, ci saranno costi maggiori a causa del traffico fra differenti Availability Zone (per fare sì che il database sia distribuito ed affidabile): ogni inserimento e rimozione di record genera traffico, per cui anche solo un traffico modesto di circa 10 Gb per ora può portare ad un costo di circa 150 euro mensili.

Amazon KeySpaces, in questo caso, è un servizio gestito e che pesa molto meno sulla bolletta.

"... e quindi uscimmo a riveder le stelle" - Inferno - Dante

Quest'ultimo racconto conclude il nostro viaggio dell'orrore nelle infrastrutture da incubo.

Siamo riusciti a farvi correre un brivido lungo la schiena ? Avete qualcosa di più spaventoso da raccontarci ? Fateci sapere nei commenti!

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!



Damiano Giorgi

Ex sistemista on-prem, pigro e incline all'automazione di task noiosi. Alla ricerca costante di novità tecnologiche e quindi passato al cloud per trovare nuovi stimoli. L'unico hardware a cui mi dedico ora è quello del mio basso; se non mi trovate in ufficio o in sala prove provate al pub o in qualche aeroporto!

Copyright © 2011-2022 by beSharp spa - P.IVA IT02415160189