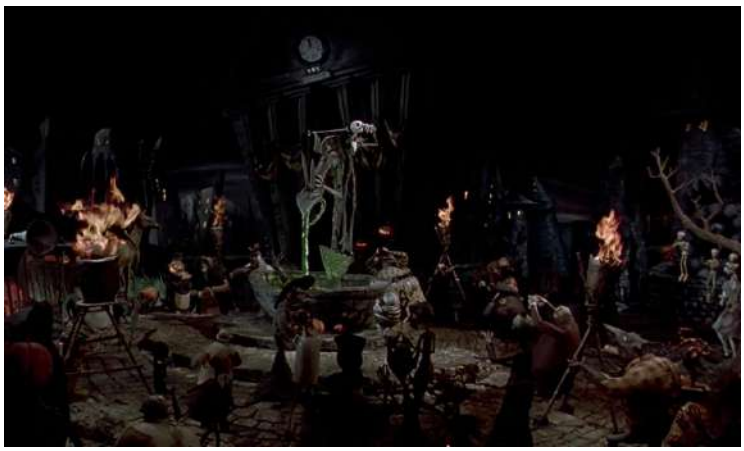
Infrastrutture da incubo speciale halloween : episodio 2

27 Ottobre 2023 - 7 min. read

Bimbi e bimbe di ogni età,
ecco qualcosa che vi stupirà!
Su, venite è propro qui!
È il paese di Halloween! - The nightmare before christmas

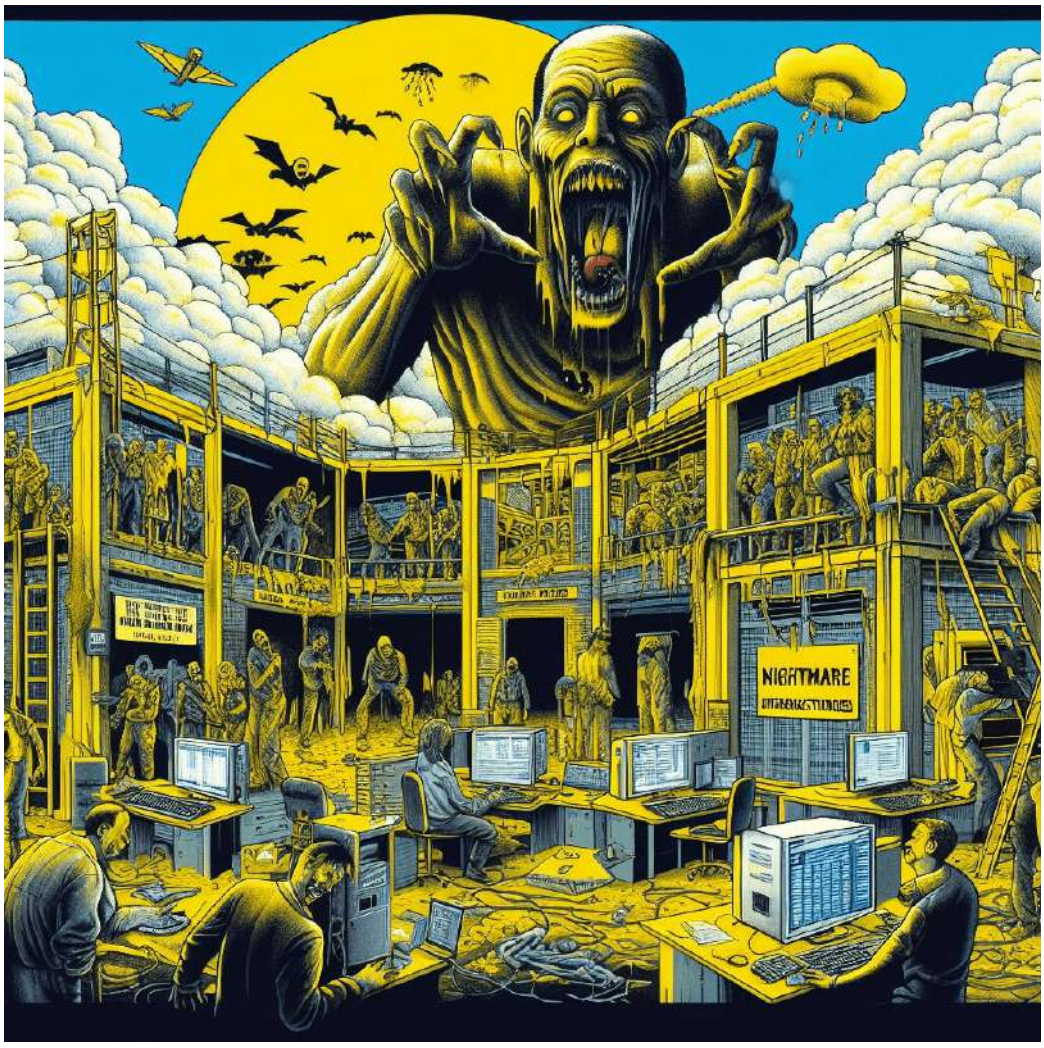


L'anno scorso, abbiamo visto alcune infrastrutture spaventose. Siete pronti per il nuovo episodio?

In questo articolo, vedremo storie su alcuni strani design e pratiche che abbiamo incontrato in questo ultimo anno. Sono anti-pattern del Cloud che diventeranno un incubo assoluto nel lungo termine.

Trattenete il respiro; stiamo per iniziare!

Gli zombie



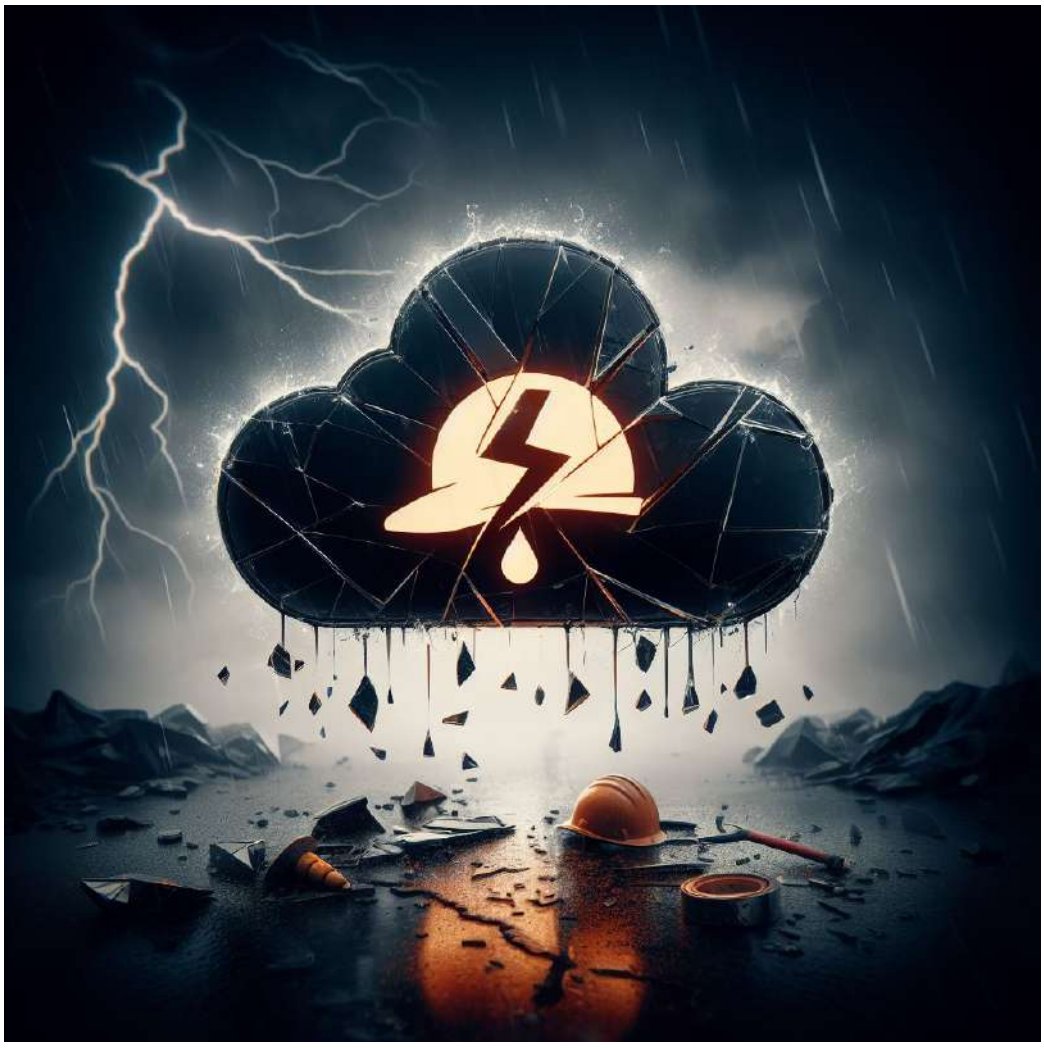
Il concetto moderno di zombi è influenzato dalla religione vodù haitiana, dove alcune persone credono che un dottore stregone possa far rivivere una persona morta come suo schiavo usando magia o una pozione segreta.

A volte, i task di ECS vengono resuscitati da un servizio, anche se dovrebbero essere sepolti (e fermati). L'abbiamo visto succedere in un ambiente di sviluppo quando un piccolo e grazioso container php aveva un problema a causa di un errore in una pipeline fallita. Il task lottava per restare vivo, ma il brutale healtcheck dell'Application Load Balancer gli sparava in testa; il servizio ECS faceva del suo meglio per far rivivere il task, riprendendolo dal repository ECR.

Nessuno si accorse del problema fino alla fine del mese quando la bolletta fatturato aumentò a oltre 800$ a causa di 16 Terabyte di traffico attraverso il NAT Gateway.

In questo caso, il circuit breaker di ECS (Deployment circuit breaker - Amazon ECS) era pronto ad aiutare, ma nessuno glielo chiese. Per evitare di creare zombi di ECS, coinvolgetelo la prossima volta che distribuite un container!

È questione di fiducia.



I'm lying when I say, "Trust me"
I can't believe this is true...
Trust hurts, why does trust equal suffering?"
Megadeth, Trust

Un antico proverbio italiano dice: "Dagli amici mi guardi Dio, che dai nemici mi guardo io".

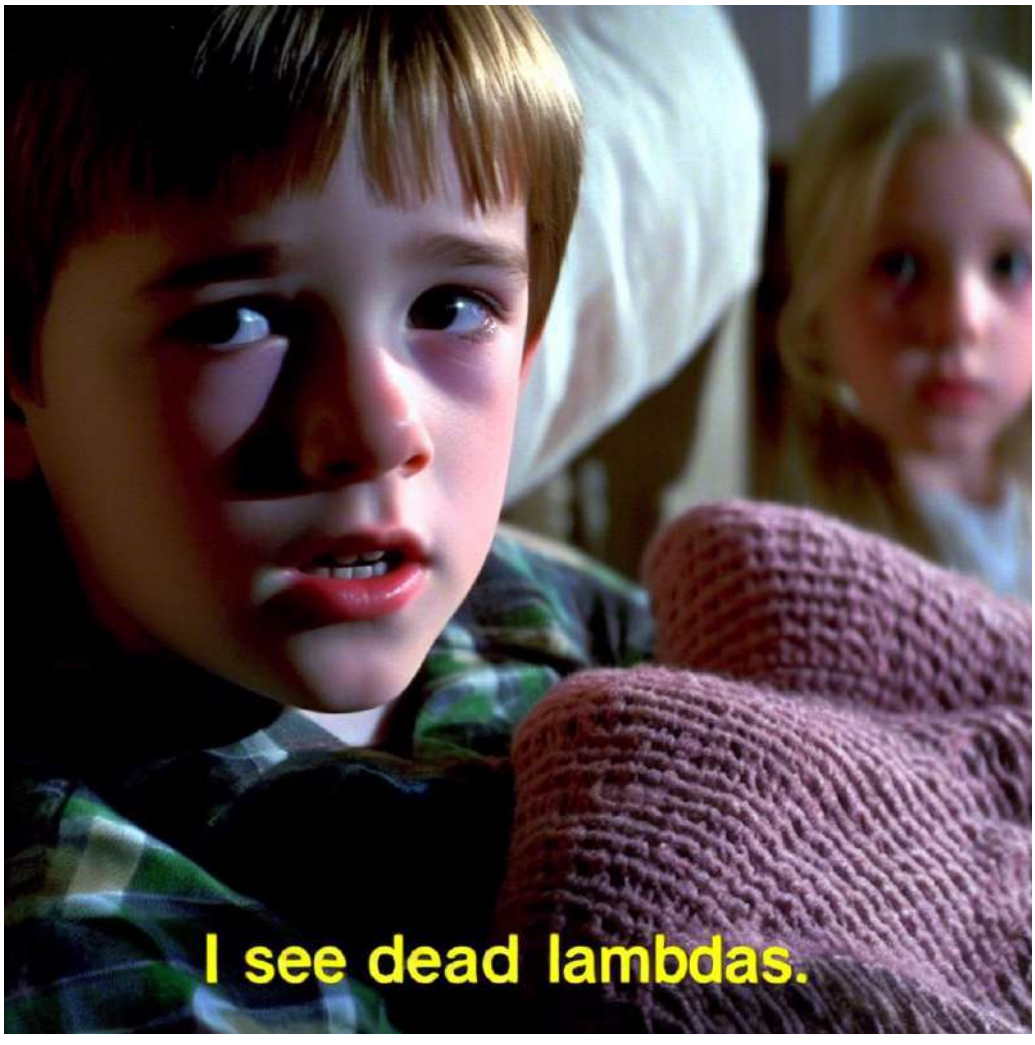
Questo è un episodio breve ma ancora più spaventoso. Mentre indagavamo su una pipeline fallita, abbiamo visto questa trust relationship in un ruolo con permessi di amministratore su ogni risorsa:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Principal": {
        "AWS": "*"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

Abbiamo prima staccato la policy e, come dimostrazione del problema, durante una chiamata con il cliente, abbiamo usato il nostro account AWS personale per assumere quel ruolo. Piuttosto spaventoso, eh?

Vedo le lambda morte



Esattamente come nel Sesto Senso..

Era una fredda notte d'inverno, e, durante una tempesta, il nostro cellulare di reperibilità iniziò a squillare disperatamente. Un'applicazione serverless lottava per la sopravvivenza, e il nostro API gateway restituiva errori 5xx. Il nostro collega si mise a indagare, e stranamente, tutto era tranquillo. Troppo tranquillo. Nessun log per la lambda associata alla rotta di API Gatewaty era presente in CloudWatch. Quando si facevano richieste con Postman o curl, tutto funzionava alla perfezione. Poiché tutto era tornato a funzionare, l'indagine fu rimandata al mattino seguente, ma... Dopo un'ora, il telefono ricominciò a squillare. E, ancora, nessuna traccia di problemi, nemmeno nei log.

Determinato a risolvere il mistero, un altro collega si unì all'indagine e, mentre revisionava la configurazione... Apparve improvvisamente! Il nostro cliente, in passato, aveva avuto qualche problema perché la lambda andava in timeout, così aveva "riservato un po' di capacità". Si scoprì che la reserved concurrency era impostata a 1.

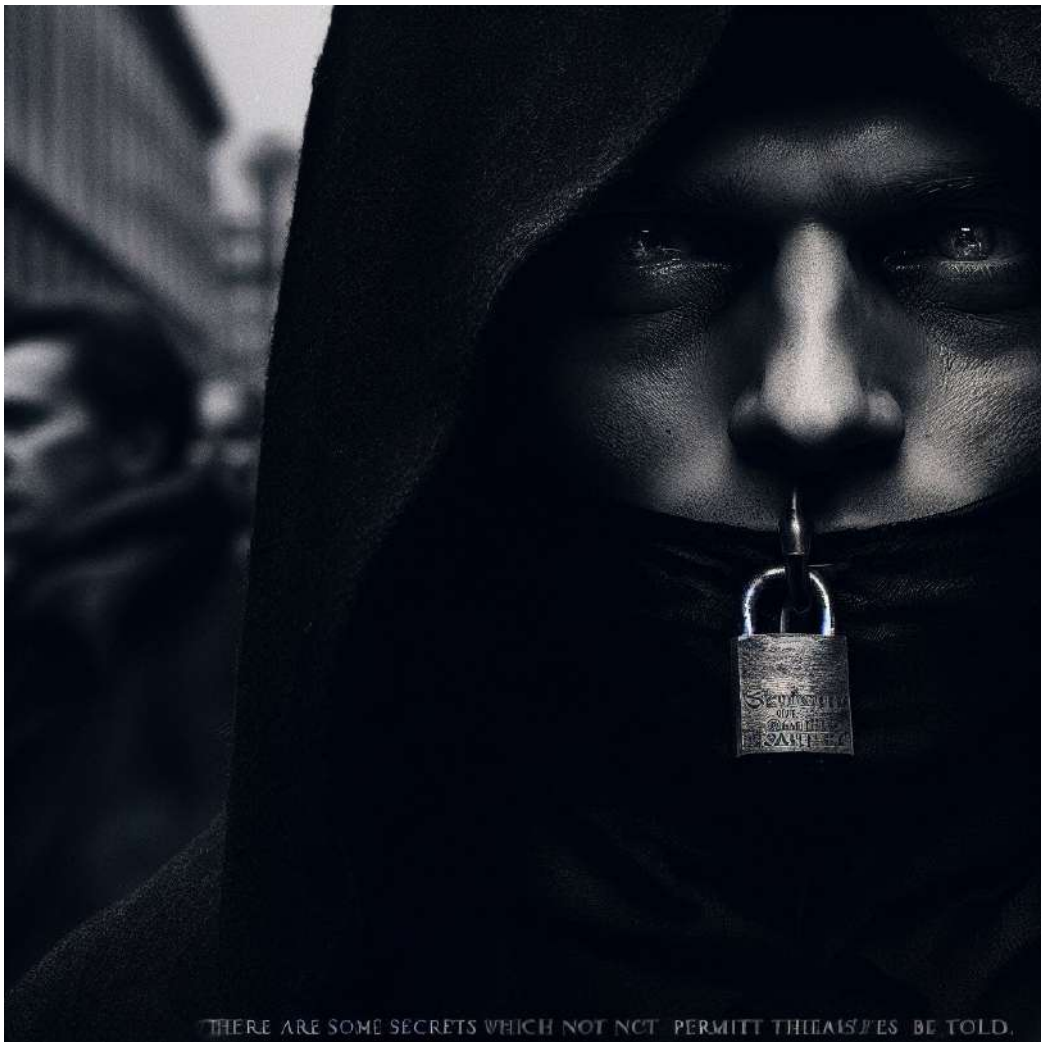
Secondo la documentazione di AWS: *"La reserved concurrency è il numero massimo di istanze concorrenti che allocabili alla funzione. Quando una lambda ha impostato la reserved concurrency, nessun'altra funzione può utilizzare la concorrenza"*.

Ma c'è un trucco: la reserved concurrency è anche il numero massimo di istanze lambda concorrenti che possono essere eseguite, quindi impostando questo valore a uno si limita e strozza effettivamente la lambda. In questo modo, se due utenti simultanei chiamano la rotta API, API Gateway restituirà un errore 5xx.

Dopo aver rimosso la concorrenza, tutto ha funzionato bene. Per avere delle lambda pronte a servire le richieste occorre considerare la reserved concurrency.

Questo articolo spiega come questi due parametri influenzano l'esecuzione e le prestazioni Lambda function scaling - AWS Lambda (amazon.com)..

Troppi segreti



“Esistono alcuni segreti che non si possono essere raccontati”
Edgar Allan Poe - L’uomo della folla

A volte, le infrastrutture possono essere perfette, ma possono comunque essere abusate. Questo è il caso di un replatform di un’applicazione e-commerce. Dopo una valutazione e una progettazione iniziali, abbiamo suggerito delle modifiche per integrarsi meglio in un ambiente cloud.

Sono state implementate sessioni stateless usando a Elasticache per Redis, scalabilità del database usando Aurora Serverless per MySQL, e, infine, è stata aumentata la sicurezza grazie all’uso di ruoli e SecretsManager per memorizzare le credenziali del database e le chiavi API esterne, invece dei soliti file di configurazione in chiaro.

Tutto andava bene quando, dopo un deployment, l’applicazione rallentava e fallivano gli health check del load balancer.

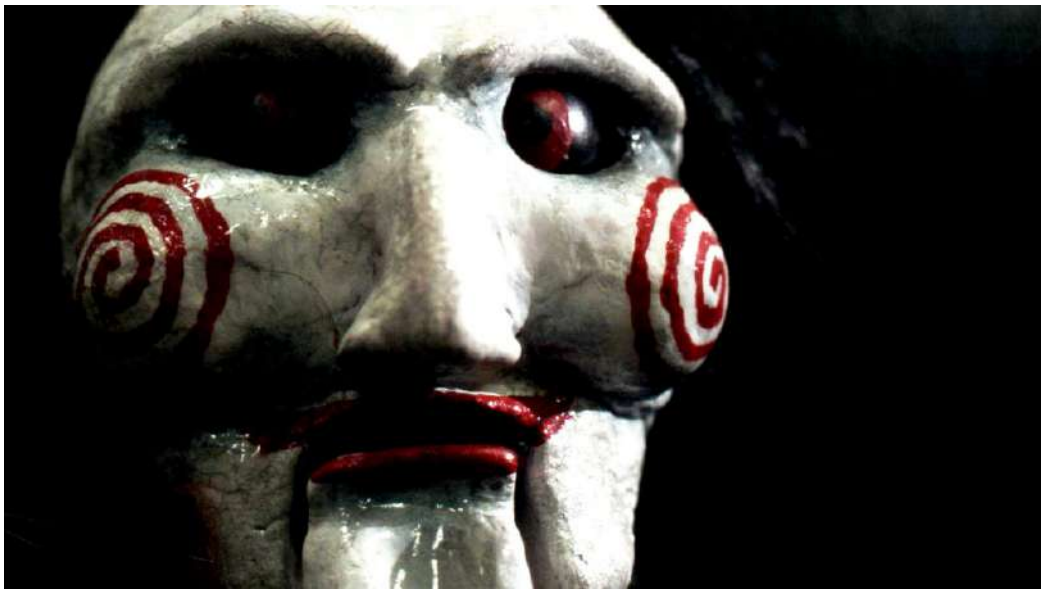
Si scoprì che, poiché SecretsManager era così facile e comodo da usare, veniva usato per tutto, anche per i parametri di configurazione ordinari come ad esempio i nomi dei bucket, gli endpoint e varie configurazioni.

Ogni richiesta a una pagina accedeva almeno una o due volte a diversi segreti, e, per peggiorare le cose, c'erano picchi di traffico dovuti alle campagne di marketing. Alla fine del mese, la dashboard del billing AWS ha mostrato 25.000.000 di chiamate API, con un costo aggiuntivo di 150$.

Ma c'è dell'altro: a volte, l'applicazione smetteva di rispondere perché il servizio dei metadati (usato per autenticare le chiamate API usando i ruoli IAM) iniziava a fare throttling delle richieste, pensando che l'applicazione stesse cercando di fare un DoS.

Dopo aver spiegato il problema e proposto una soluzione che utilizzasse parameter store e le variabili d'ambiente, tutto ha funzionato di nuovo alla perfezione, e il porting è andato bene.

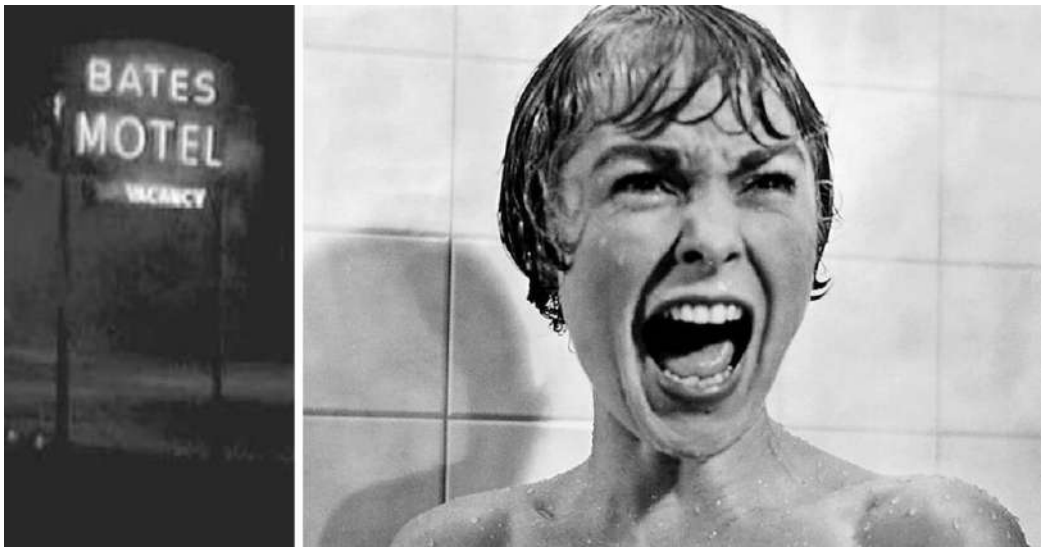
Alta inaffidabiltà



"Congratulazioni. Sei ancora vivo. La maggior parte delle persone è così ingrata di essere viva. Ma non tu. Non più"
- Saw.

Parlando di rifacimento delle applicazioni orientate al cloud, questo è un semplice errore che a volte vediamo accadere: se un'applicazione è altamente disponibile, per favore non legate le richieste vitali a un singolo point of failure (come un server FTP che gira su un'istanza EC2 per recuperare i file di configurazione). Questa è una storia breve, ma c'è molto dolore dietro.

Backup infernale



“Tutti impazziamo un po’ a volte.”
- Psycho (1960)

Il disaster recovery dei workload on-premise sul Cloud è un argomento attualissimo, soprattutto per le grandi imprese e le architetture on-premise.

Progettare una soluzione resiliente non è facile. Il primo passo è trovare una strategia di backup che possa adattarsi e garantire il giusto RPO e RTO. Una volta definita la strategia di backup e i requisiti, è il momento di trovare il software giusto che soddisfi le nostre esigenze.

In una grande impresa, la prima scelta è di usare la soluzione esistente, ma adattarla per funzionare sul Cloud. Nessun problema, purché ci sia almeno una forma di integrazione, tipicamente con Amazon Glacier o Amazon S3. Questo era il nostro caso, ma...

Per rendere le cose più resilienti, qualcuno ha installato il software su un’istanza EC2 in un account AWS dedicato e lo ha configurato per fare il backup delle macchine on-premise usando la connessione Direct Connect esistente e quindi attraverso l’attachment del Transit Gateway.

Si può già intuire che direzione può prendere il billing di AWS (spoiler: molto in alto)! Per peggiorare le cose, tutti gli endpoint erano centralizzati in un account di rete dedicato per una migliore gestibilità e osservabilità.

Quindi, per riassumere, un singolo gigabyte salvato da on-premise:

- Usava la connessione Direct Connect
- Attraversava l’attachment del transit gateway per raggiungere l’istanza EC nel Backup Account
- Attraversava ancora il Transit Gateway per raggiungere l’account di rete

- Attraversava l’endpoint S3 centralizzato

Alla fine del mese, c’è stato un picco di 6.000 $ nella sezione Networking della fattura AWS.

Cosa abbiamo imparato? Se qualcosa sembra facile e banale in un ambiente complesso, dovresti cercare meglio degli indizi.

Conclusioni

Anche quest’anno abbiamo avuto la nostra dose di storie horror. Come sempre, ogni errore non è intenzionale, e la cosa buona è che possiamo sempre imparare cose nuove, evitando di ripeterci in futuro.

Voglio ringraziare tutti i partecipanti all'AWS Community Day a Roma che hanno condiviso le loro storie dopo il mio speech sull’articolo dell’anno scorso. L’episodio del prossimo anno è già scritto! :)

Cosa avete trovato nei tuoi account AWS che vi ha fatto rabbrividire ?

Fatecelo sapere nei commenti! (Accettiamo anche e-mail anonime :D)

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!
