Quando il Serverless “gira” sui server: nuove opzioni per AWS Lambda e AWS Fargate con le Managed Instances

4 Marzo 2026 - 14 min. read

AWS Fargate | AWS Lambda | ECS Managed Instances | EKS Auto Mode | Lambda Managed Instances | Serverless

Per anni siamo rimasti intrappolati nella dicotomia del Cloud Computing: scegliere la semplicità del serverless con Lambda e Fargate, oppure accettare il peso operativo di gestire le istanze EC2 in autonomia. Entrambe le opzioni avevano il loro caso d’uso, ma nessuna delle due era perfetta.

Non conto più le volte in cui, alle conferenze, abbiamo parlato di Lambda vs Container o di Lambda vs Kubernetes (sono stato anche co-speaker su quest'ultimo argomento).

- *"Lambda è troppo costoso per workload costanti."*
- *"Fargate costa più di EC2 a parità di risorse."*
- *"Kubernetes è economico, ma devi strumentarlo, mantenerlo e non dimenticarti di aggiornarlo ogni 6 mesi. Buona fortuna con le API deprecation!"*
- *"Vuoi far girare tutto su EC2? Spero proprio che il tuo team abbia il tempo e le competenze per gestire istanze, AMI, patching e scaling."*

Queste frasi le abbiamo sentite in molti talk, e la conclusione era sempre il classico: "Devi conoscere il tuo workload e scegliere il mix di tecnologie che ti permetta di ottimizzare costi ed efficienza."

Alla fine del 2025, AWS ha cambiato le regole del gioco. Nel giro di pochi mesi sono stati annunciati Lambda Managed Instances, ECS Managed Instances e EKS Auto Mode.

Non sono aggiornamenti marginali: rappresentano un cambio nel modo in cui è possibile progettare i workload su AWS, tenendo conto dei costi.

Il vecchio paradigma: scegli la tua condanna

Dove eravamo? Siamo onesti: Lambda è brillante per architetture event-driven, API con traffico imprevedibile e funzioni che vengono eseguite poche volte al giorno.

Lambda scala a zero, quindi si paga solo per ciò che si usa. Ma far girare la stessa funzione in modo continuativo, con volumi elevati, finisce per incidere pesantemente sui costi, a causa del modello di prezzo pay-as-you-go.

Fargate ha problemi simili. Azzera completamente la gestione dei server, il che è fantastico per i team che non vogliono preoccuparsi del capacity planning. Ma separarsi dal mondo server comporta un costo direttamente misurabile nella bolletta mensile: le stesse risorse (CPU e RAM) costano di più rispetto alle controparti EC2 (soprattutto considerando ECS su EC2).

D'altra parte, gestire le proprie istanze EC2 offre il pieno controllo e un'economia molto migliore per workload costanti. In più è possibile scegliere tipi di istanze specifiche e specializzate e applicare gli sconti delle Reserved Instance.

Il compromesso? Il team è responsabile della manutenzione delle AMI, del security patching, dell'autoscaling e di tutto l'overhead operativo che comporta la gestione di un'infrastruttura "tradizionale".

Il vero costo di EC2 self-managed non è solo il prezzo dell'istanza: sono le ore di engineering spese in operazioni che potrebbero essere dedicate allo sviluppo di nuove feature.

L'anello mancante tra due mondi

Molti di noi lo pensavano da anni: avremmo bisogno di qualcosa a metà tra "fully serverless" e "gestisci tutto tu".

La risposta è arrivata in tre varianti poco prima e durante il re:Invent 2025:

Lambda Managed Instances

Le Lambda Managed Instances consentono di eseguire funzioni Lambda su istanze EC2 interamente gestite da AWS. È possibile scegliere tra diversi tipi di istanza, tra cui Graviton4, networking ad alta banda e calcolo GPU.

AWS gestisce provisioning, patching, scaling e lifecycle management.

Il modello di pricing è interessante:

- **Prezzo per richiesta**: $0.20 per milione (uguale alla Lambda standard)
- **Prezzo istanze EC2**: si applica il pricing EC2 standard
- **Costo di management**: 15% di sovrapprezzo sul prezzo EC2 on-demand
- **Prezzo per tempo di esecuzione**: a differenza di Lambda standard, non occorre pagare per GB-secondo

C'è però una differenza architetturale: le Lambda standard elaborano una sola invocazione per ciascun ambiente di esecuzione, mentre le Istanze Gestite possono gestire più invocazioni concorrenti nello stesso ambiente di esecuzione.

Questo modello a multi-concorrenza consente un migliore utilizzo delle istanze EC2 sottostanti, e poiché l'ambiente di esecuzione rimane lo stesso, è possibile ridurre i "cold start".

Quello che rende ancora più interessante questa implementazione è la possibilità di applicare EC2 Savings Plans e Reserved Instances alla parte compute sottostante.

Un Compute Savings Plan triennale può far risparmiare fino al 72% sul prezzo EC2 on-demand, ma occorre tenere presente che la management fee del 15% si calcola sul prezzo on-demand (non sul prezzo scontato).

Anche così, per workload ad alto volume, è possibile risparmiare in modo significativo rispetto alle Lambda standard (vedremo i conti più avanti).

C'è un ma (come sempre): le Lambda Managed Instances scalano in base all'utilizzo della CPU, non al tasso di invocazione. Ad esempio, se il traffico dovesse più che raddoppiare in 5 minuti, ci sarà throttling delle richieste mentre AWS provvede ad aggiungere istanze per far scalare l’ambiente.

ECS Managed Instances

Anche per le ECS Managed Instances, AWS effettua il provisioning e gestisce le istanze EC2, occupandosi di patching e scaling: occorre specificare solo i requisiti dei task (CPU, memoria, instance type).

Questa è una differenza rispetto a Fargate, dove è possibile scegliere solo l'architettura (Graviton o x86). Ce ne siamo accorti (con nostra sorpresa) durante un troubleshooting di problemi di prestazioni di un workload sensibile alla latenza.

A volte il task era ultrarapido, altre volte invece andava a singhiozzo, rallentando l'intera pipeline. Questo era dovuto proprio alla differenza tra la cpu assegnata al task, che non poteva essere scelta. Dopo aver individuato il problema, siamo tornati al "classico" ECS su EC2 e non si sono più verificati problemi di prestazioni.

Se avessimo avuto la possibilità di utilizzare le managed instances, avremmo trovato il giusto compromesso: più semplice di ECS su EC2 self-managed, più economico di Fargate e con accesso a hardware specializzato.

Il modello operativo è simile alle Lambda Managed Instances, ma con alcune differenze:

- Lifetime massimo dell'istanza di 14 giorni per sicurezza
- Patching automatico durante maintenance window configurabili
- Supporto per instance type specializzati, incluse le GPU
- Nessun accesso SSH/SSM alle istanze (per sicurezza)

Attenzione: Fargate esegue ogni task nella propria microVM isolata, il che è ottimo per la sicurezza, mentre le ECS Managed Instances possono ospitare più task, venendo meno all'isolamento. Esiste l'opzione "single-task mode" per un isolamento più stringente, ma ovviamente ci sarà un maggiore utilizzo delle risorse.

Inoltre, questo tipo di deployment avrebbe evitato il caso "The Undead" di "Infrastrutture da Incubo" perché le immagini dei container vengono scaricate dal registry solo se non sono già presenti nell'istanza, potenzialmente risparmiando $800 in costi di NAT Gateway.

Parlando dei costi, anche in questo tipo di deployment si applica una management fee aggiuntiva rispetto al costo delle istanze EC2.

EKS Auto Mode

EKS Auto Mode offre come sempre il control plane gestito, ma si spinge oltre. L'autoscaling della parte compute è gestito tramite Karpenter, il tool open-source che ottimizza l'utilizzo dei nodi e i costi. Gestisce anche il networking dei pod con la network policy enforcement, l'integrazione con il load-balancing e lo storage EBS CSI.

EKS Auto Mode offre:

- Lifetime massimo di 21 giorni per i nodi
- AMI immutabili con SELinux enforcing e root filesystem in sola lettura
- Nessun accesso SSH o SSM
- Completa gestione da parte di AWS, dall'assegnazione degli IP ai pod fino al supporto GPU

L'impatto pratico è significativo. Chiunque abbia gestito Kubernetes in produzione conosce i suoi punti dolenti: nodi bloccati in stato NotReady, esaurimento degli indirizzi IP a causa di configurazioni VPC CNI errate, upgrade falliti, il ciclo infinito di patching e di aggiornamento delle AMI.

EKS Auto Mode elimina la maggior parte di questi oneri operativi ricorrenti. Occorre però tenere presente che non si tratta di magia: il workload deve essere compatibile con questa modalità e richiede l'implementazione di check adeguati per la readiness e la definizione di una pod eviction policy che risponda ai requisiti di business; altrimenti si otterranno solo downtime, singhiozzi e altri problemi quando Karpenter ottimizza la distribuzione dei pod per mantenere i costi bassi.

Anche con EKS Auto Mode è prevista una management fee.

Altre opzioni: Lambda Durable Functions

Proprio mentre stavamo ancora digerendo le managed instances, AWS ha introdotto anche le Lambda Durable Functions.

Questa feature consente alle funzioni Lambda di eseguire workflow che durano fino a un anno, utilizzando checkpoint e replay automatici per gestire fallimenti e attese.

Le Lambda Durable Functions adottano un approccio code-first: la logica del workflow è scritta in Node.js o Python (attualmente solo questi due runtime gestiti sono supportati), e il Durable Execution SDK fornisce le primitive per il checkpointing e la pausa dell'esecuzione.

A questo indirizzo è presente un esempio di durable function.

Lambda Durable Function salva automaticamente un checkpoint quando l'esecuzione viene messa in pausa; quando riprende, il codice parte dall'inizio, ma salta i checkpoint già completati, utilizzando i risultati salvati anziché eseguire di nuovo tutte le operazioni.

Con le durable functions, il costo è dovuto al compute time e al numero di richieste, come al solito, ma non ci sono addebiti durante i periodi di attesa: un workflow può girare per 1 minuto e attendere per 24 ore; verrà fatturato solo 1 minuto.

Le durable operation (come avviare esecuzioni, completare step e creare attese) hanno però un costo aggiuntivo, come anche la quantità di dati scritti da queste operazioni (in GB) e la data retention durante e dopo l'esecuzione (in GB/month). Il periodo di retention dopo il completamento è configurabile da 1 a 90 giorni (il default è 14 giorni).

Quando ho visto l'annuncio, ho chiesto ai miei colleghi del team Cloud Native Development: "Le Durable Functions sostituiscono le Step Functions?"

La risposta è sfumata, perché, anche se sembrano sovrapporsi funzionalmente, il loro utilizzo dipende da preferenze di sviluppo e dai pattern architetturali utilizzati.

Le Step Functions usano un approccio "configuration first": definiscono i workflow in Amazon States Language (ASL), un DSL basato su JSON che coordina i servizi AWS con una minima quantità di codice. È possibile usare un visual workflow designer per visualizzare l'intera state machine a colpo d'occhio.

Le Step Function si integrano con i servizi AWS senza richiedere codice custom. Per esempio, è possibile avviare una Step Function quando un Amazon S3 Bucket viene reso pubblico, per notificare il team di sicurezza, che può effettuare una review. Nel frattempo, la funzione aspetta un'approvazione manuale per la remediation. Quando l'approvazione manuale viene ricevuta o rifiutata, vengono intraprese le azioni necessarie (ad esempio rendere il bucket privato o applicare un tag che escluda il bucket da un controllo futuro).

Con Step function non ci sono nemmeno restrizioni di runtime: mentre le Durable Functions supportano solo Python 3.13 e 3.14 e Node.js 22.x e 24.x, le Step Functions possono avviare Lambda con runtime custom, avviare task ECS, eseguire Glue job e molto altro.

Parlando di costi: gli standard workflow addebitano $25 per milione di state transition; gli Express Workflow addebitano un costo in base alla durata e al numero di esecuzioni.

La differenza tra le due tecnologie è così sottile che la documentazione AWS ha una pagina per guidare alla scelta del servizio giusto.

È possibile anche usare un approccio ibrido: Durable Function per la logica applicativa all'interno di Lambda (data processing, business rule, API orchestration) e Step Functions per workflow cross-service su diversi execution environment (ETL pipeline, approval workflow che attraversano più sistemi).

Altre considerazioni

Le Durable Functions e le Step Functions potrebbero sovrapporsi ai container long-running, quindi abbiamo ora altre tre opzioni?

Pensiamo però ai costi: un workflow (con Step Functions o Durable Functions) che attende 23 ore per l'approvazione e gira per 1 ora, costa solo per l'ora di lavoro effettivo. Con un container,invece, il costo sarà dovuto per tutte le 24 ore, a meno che di non costruire (e mantenere) una logica di pausa/ripresa.

Per altri workload, come ad esempio il video encoding, la training ML e lo stream processing, i container vincono ancora.

Quando usare cosa

Ok, abbiamo visto un sacco di nuove tecnologie, e, onestamente, lo scenario può sembrare più confuso di prima. Volevamo più opzioni, ma adesso scegliere il servizio giusto sembra più difficile.

Prima di riscrivere tutto da zero per usare le nuove tecnologie, dobbiamo sempre ricordare che le nostre scelte incidono sul business e sullo sforzo del team tecnico nel mantenere l'infrastruttura e le applicazioni.

Provare una nuova tecnologia può essere entusiasmante, ma occorre considerarne gli impatti e le implicazioni a lungo termine. Una tecnologia "vecchia" e ben collaudata non è peggio di una nuova.

Quando si sta sviluppando un nuovo prodotto, si tende tipicamente ad con carichi di lavoro sporadici, basati su eventi e con un traffico imprevedibile, permettendo di scalare fino a zero. In questo scenario, Lambda rappresenta la scelta ottimale.

Nel momento in cui il traffico diventa stabile e le invocazioni si accumulano, è possibile valutare le istanze gestite (managed instances), monitorando attentamente i picchi di traffico.

Per la gestione di workflow a lunga durata, si può optare per Durable Functions o Step Functions, a seconda dell'esperienza del team o dell'ambiente richiesto.

Per le applicazioni ad alto fabbisogno di risorse di calcolo (compute-intensive), è possibile scegliere tra ECS ed EKS in tre modalità: serverless (Fargate), serverful (istanze EC2 autogestite) o istanze gestite (managed instances).

Quando si hanno a disposizione team Operations altamente qualificati, è possibile ottenere risparmi sui costi di servizio e sulle management fee; tuttavia, quando è necessario ottimizzare i tempi e ridurre lo sforzo di manutenzione, le opzioni managed o serverless rappresentano una soluzione efficace.

Strategy	**Use Case**
Serverless (Fargate or Lambda)	- Development/test environments - Containers that truly need VM-level isolation per task - Bursty workloads with unpredictable traffic - Teams that want absolute simplicity and can absorb the cost premium
ECS And Lambda Managed Instances	- Steady-state workloads - Services running 24/7 or with predictable patterns - Workloads needing GPU or specialized compute

ECS on EC2	- When you need custom AMIs or specific kernel modules - Teams with platform engineering capacity who want maximum control
EKS Auto Mode	- Teams that need Kubernetes but lack deep K8s operational expertise - Workloads requiring the Kubernetes ecosystem (CRDs, operators, Helm) - Multi-cluster strategies where reducing operational burden matters - Production workloads where the 21-day node lifetime works with your architecture
EKS (self-managed)	- When you need custom node configurations or specific AMIs - Multi-cloud or hybrid deployments requiring portability - Teams with dedicated platform engineering who want full control - Advanced networking requirements beyond what Auto Mode provides
Long-running containers (ECS/EKS)	- Truly continuous processing (video encoding, stream processing) - Workflows that need state in memory throughout - CPU-bound workloads running non-stop - Legacy applications that weren't designed for pause/resume patterns
Lambda Durable Functions	- More familiarity with development tools - Code-first approach - Control using code instead of external tools
Step Functions	- Integration with a broad set of services is required

	- Visual workflow representation - Less code-centric knowledge is needed.

Per stimare i costi, dobbiamo considerare due dimensioni: i pattern di traffico e lo sforzo di gestione.

I conti in tasca

Usiamo dei numeri realistici per confrontare i costi. Considereremo due scenari con lo stesso volume di traffico ma pattern diversi (steady vs. burst).

Per determinare i costi, useremo due opzioni: on-demand e 1-Year Compute Savings Plans senza pagamento anticipato.

Come sempre, occorre precisare che stiamo considerando un singolo scenario, mentre il traffico, in casi reali, può variare enormemente a seconda del settore, dell'implementazione e di un milione di altri fattori.

In tutte e due le opzioni considereremo

- 50 milioni di request al mese
- Durata media di 200ms per request
- Requisito di 2GB di memoria

Scenario 1: traffico steady (gira 24/7)

Scenario 2: burst di traffico, distribuzione gaussiana con picco intorno alle 18

Scenario 1: Traffico Steady

- **Throughput**: 50.000.000 di richieste/mese, pari a 1,64 milioni al giorno, per un RPS di 19,03
- **Compute necessario**: 19,03 RPS × 0,2 s di durata = 3,8 esecuzioni concorrenti.
- **vCPU**: Assumendo che i 200 ms di durata siano CPU-bound, servono circa 4 vCPU.
- **Memoria**: 3,8 concorrenti × 2GB = ~8 GB RAM

Baseline infrastrutturale:

Per soddisfare i requisiti di CPU e garantire l'alta affidabilità utilizzando almeno 2 Availability Zone (AZ), è possibile considerare l'utilizzo di due istanze m7g.large.

- **Specifica dell'istanza**: m7g.large (2 vCPU, 8 GB di RAM).
- **Capacità totale**: 4 vCPU, 16 GB RAM (sufficiente per il workload).
- **Strategia HA**: Almeno 2 istanze, una per Availability Zone.

Nota: Il break-even point al quale la Lambda Standard diventa più economica è circa 15-20 milioni di request/mese per questo specifico profilo di memoria (2GB).

Se siete appassionati di calcoli, scriveteci! Vi manderemo il foglio con l'analisi dettagliata fatta sui costi per questo articolo ;)

Technology	On-Demand Cost (Monthly)	1-Year Savings Plan (No Upfront)	Cost components
ECS on EC2	**$119.14**	**$88.16**	Only Infrastructure
ECS Fargate	**$144.16**	**$108.12**	vCPU/GB Usage Rates
ECS Managed Instances	**$148.34**	**$117.36**	Infrastructure + variable management fee
EKS on EC2	**$192.14**	**$161.16**	$73 fixed cluster fee + infrastructure
EKS Auto Mode	**$206.44**	**$175.46**	$73 fixed + infrastructure + variable management fee
Lambda Managed Instances	**$215.51**	**$179.77**	Infrastructure (3 instances for HA by default) + management fees

			+ number of requests
EKS Fargate	**$217.16**	**$181.12**	$73 fixed cluster fee vCPU/GB Usage Rates
Lambda (Standard)	**$343.33**	**$293.33**	Duration + Requests

Scenario 2: Traffico Burst

Il volume di traffico è lo stesso (50M/mese), ma concentrato; il picco di richieste concorrenti sale a circa 14 (senza quindi problemi di concorrenza).

Usando Lambda o Fargate, il cambio di pattern di traffico non ha impatto sui costi: i costi dipendono dal totale delle richieste e dalla loro durata, che restano identici. (Per Fargate, assumiamo che l'auto-scaling gestisca il picco efficientemente).

In uno scenario basato su EC2, due istanze m7g.large offrono 4 vCPU, come abbiamo visto in precedenza. Il picco richiede circa 14 vCPU, quindi durante l'ora di punta occorre scalare aggiungendo 4 istanze in più e rimuovendole al termine, per un totale di circa 1000 instance-hour.

Nota: Assumiamo che le nostre policy di scaling siano in grado di gestire il traffico senza problemi. Questa è solo una stima di prezzo, non una valutazione delle prestazioni né un test di scalabilità. Altri scenari potrebbero avere requisiti diversi, orientando la scelta tecnologica in direzioni diverse.

Technology	On-Demand Cost (Monthly)	1-Year Savings Plan (No Upfront)
ECS on EC2	**$160.00**	**$118.00**
ECS Fargate	**$158.00**	**$119.00**
ECS Managed Instances	**$190.00**	**$148.00**
Lambda Managed Instances	**$225.00**	**$189.00**

EKS on EC2	**$233.00**	**$191.00**
EKS Auto Mode	**$245.00**	**$203.00**
Lambda (Standard)	**$343.33**	**$293.33**

Quindi, abbiamo un vincitore? Dobbiamo cambiare le nostre architetture?

No, perché, alla fine, la risposta è (come sempre)... dipende!

Le managed instances rappresentano una maturazione dei pattern di architettura cloud. Non dobbiamo cambiare la nostra architettura preferita, mantenendo le pratiche di sviluppo già in uso.

Le Managed Instances offrono una scelta che combina la semplicità del serverless e la flessibilità di EC2, con opzioni di ottimizzazione dei costi per workload che nel corso del tempo si sono evoluti in un prodotto con traffico stabile.

Sono perfette? No. Ogni workload trarrà beneficio da questo cambiamento? Assolutamente no. Ma le managed instances offriranno una maggiore efficienza dei costi e una maggiore semplicità operativa rispetto alle istanze EC2 self-managed.

Il paradigma è cambiato. Non dobbiamo più scegliere tra due estremi: abbiamo uno spettro di opzioni ampio per far combaciare le caratteristiche del workload, le capacità del team e la tolleranza ai compromessi tra costi e operazioni.

Per i miei workload in produzione, sto migrando le funzioni Lambda ad alto volume verso le Managed Instances e valutando le ECS Managed Instances per i servizi container che girano 24/7. I conti tornano, e ridurre il lavoro operativo ripetitivo è sempre prezioso.

E voi? Avete workload che potrebbero beneficiare delle managed instances? Siete bloccati nella trappola dei costi di Fargate? Fatecelo 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!
