AUTOSCALING LIKE A PRO: COME UTILIZZARE GLI AUTO SCALING LIFECYCLE HOOKS DI EC2.

Amazon EC2 | Amazon EC2 Auto Scaling | Lifecycle Hooks

beSharp | 4 Ottobre 2019

Vi è mai capitato di dover indagare le cause di un arresto inaspettato di un'istanza EC2 appartenente ad un Auto Scaling group o di doverla configurare prima di portarla nello stato InService?

Questi sono entrambi casi in cui i L**ifecycle Hooks di EC2** possono venirci in aiuto.

In questo articolo approfondiremo il loro funzionamento esaminando alcuni casi d'uso particolarmente significativi.

Partiamo con un breve recap delle principali caratteristiche del servizio AWS EC2 Auto Scaling.

EC2 Auto Scaling è un servizio offerto da Amazon Web Services che permette di **adattare il numero delle istanze EC2 in uso per garantire il corretto funzionamento delle applicazioni in base al reale andamento del traffico.** Le istanze sono logicamente raggruppate in **Auto Scaling groups.** Per ciascuno di questi gruppi è possibile impostare valori di **capacità minima, massima e desiderata.** Ad ogni Auto Scaling group possono essere associate particolari **policy di scaling.** Nel caso in cui nessuna policy sia associata, invece, il comportamento di default si traduce nel mantenimento di un numero di istanze healthy che coincide con il valore desiderato, inizialmente impostato. Per poter mantenere questo valore stabile, vengono eseguiti periodicamente **health check su ciascuna istanza** in modo tale che, per ogni istanza trovata in stato unhealthy, ne venga immediatamente avviata un'altra funzionante in sostituzione.

Oltre allo scaling manuale in cui i valori di capacità minima, massima e desiderata vengono impostati e aggiornati manualmente, esiste la possibilità di rendere lo **scaling automatico.**

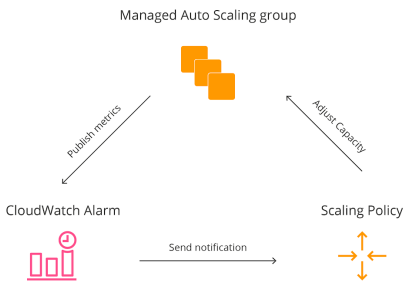
In particolare ci sono due diversi tipi di scaling automatico:

- **Scheduled scaling:** particolarmente indicato nel caso in cui si abbia una conoscenza accurata, precisa e prevedibile dell'andamento del traffico. In questo caso, l'Auto Scaling group aumenterà o diminuirà il numero di istanze in momenti specifici secondo regole di scheduling predefinite.
- **Dynamic Scaling:** se il traffico non è prevedibile, né dipendente da fattori conosciuti, lo Scaling di tipo dinamico è ciò che ci servirà per fronteggiare adeguatamente le variazioni nel volume di richieste.

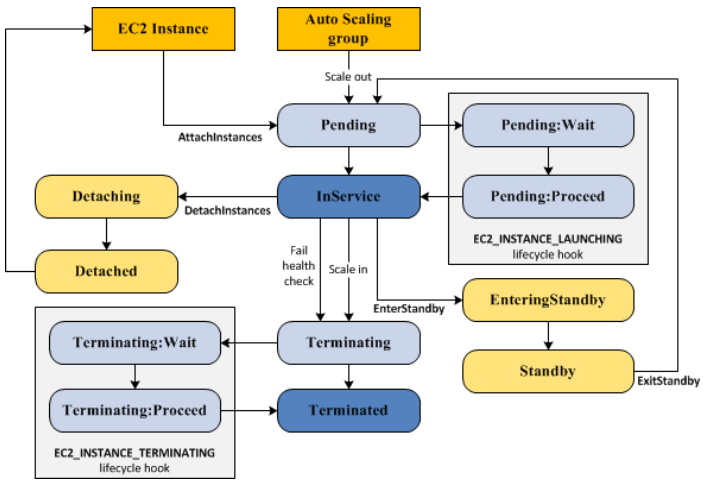
Per il Dynamic Scaling esistono **tre tipi di Policy di scaling:**

- **Simple Scaling;**
- **Step Scaling**, dipendente dalla gravità degli allarmi CloudWatch;
- **Target Tracking Scaling.** È possibile paragonare questo automatismo ad un Feedback Loop in cui viene calcolato il numero di istanze da aggiungere o rimuovere da un dato Auto Scaling group in base alla differenza tra una metrica registrata e il suo valore target.

Questo tipo di policy è particolarmente utile nel caso in cui le metriche di utilizzo siano proporzionali alla capacità dell'Autoscaling group.



Se un'istanza appartiene ad un Autoscaling group, il suo lifecycle risulterà differente da quello di ogni altra istanza EC2. Ecco una rappresentazione del Lifecycle di una istanza EC2 appartenente ad un Autoscaling group:



Cosa succede quando un evento di scale-out avvia una nuova istanza EC2?

Pending State

All'avvio di una nuova istanza EC2 questa assume , in risposta ad un evento di scale-out, **lo stato di "Pending".** In questo stato è possibile configurare l'istanza – preferibilmente in modo automatico – e assicurarsi che superi tutti gli health check prima del suo passaggio allo stato successivo:

InService

A partire da questo stato, l'EC2 potrà passare a uno dei seguenti stati di transizione:

- **Detaching** nel caso in cui risponda ad una richiesta di Detaching;
- **EnteringStandby,** nel caso in cui si scelga di metterla momentanemente in uno stato di standby;
- **Terminating,** nel caso di scale-in o nel caso in cui l'istanza fallisca una serie di health checks consecutivi sufficiente da essere considerata unhealthy;

Pending, Detaching, EnteringStandby e Terminating sono tutti **stati considerati di transizione** e anticipano rispettivamente gli **stati stabili di InService, Detached, StandBy e Terminated.**

Gli stati di Pending e Terminating possono entrambi essere momentaneamente sospesi grazie all'utilizzo degli EC2 Auto Scaling Lifecycle Hooks. Con essi, infatti, è possibile definire logiche custom da applicare ai processi di avvio e terminazione delle istanze.

Il Lifecycle Hook dello stato Pending prende il nome di **EC2_INSTANCE_LAUNCHING** e si occupa di gestire gli ulteriori stati **Pending:Wait** e **Pending:Proceed.**

Allo stesso modo, Il Lifecycle Hook dello stato Terminating chiamato **EC2_INSTANCE_TERMINATING** si occuperà di gestire gli stati **Terminating:Wait** e **Terminating:Proceed.**

In entrambi i casi, le istanze EC2 resteranno in Wait fino allo scadere del timeout a meno che non si decida di concludere in anticipo il Lifecycle Hook procedendo manualmente allo stato successivo.

Il risultato di un Lifecycle Hook potrà essere **CONTINUE** o **ABANDON.**

Per quanto riguarda il lifecycle di tipo EC2_INSTANCE_LAUNCHING, il risultato ottenuto decreterà il passaggio dell'istanza dallo stato InService o Terminating.

Nel caso di EC2_INSTANCE_TERMINATING, invece, il risultato non avrà conseguenze sul lifecycle dell'istanza EC2; l'istanza verrà in ogni caso terminata.

Quando è utile mettere in pausa gli stati Pending e Terminating?

Dipende dallo scenario in cui ci troviamo.

Andiamo nel dettaglio e vediamo insieme alcuni dei casi più comuni in cui, mantenere un'istanza nello stato Pending o Terminating può tornare utile.

Pending state hook

Instance configuration

Come spiegato in precedenza, è possibile pausare un lifecycle di un'istanza EC2 in Pending state prima che essa passi in stato InService.

Mantenere lo stato Pending di un'istanza risulta particolarmente importante quando il tempo di setup si protrae più del previsto. In questo modo è possibile impedire all'istanza di ricevere richieste quando ancora non è in grado di gestirle.

Ad esempio, il servizio CodeDeploy si basa sul lifecycle hook EC2_INSTANCE_LAUNCHING per gestire il deploy di applicazioni su istanze EC2 che appartengono a un Autoscaling group. Il tutto è perfettamente descritto in questo articolo.

È un tipo di comportamento che funziona anche su soluzioni custom attraverso, ad esempio, script resi disponibili negli User Data di un'istanza EC2.

Terminating state hook

Log files download

A: "Aiuto! La mia istanza EC2 è down!"

B: "Ok, vediamo qual è la causa dell'arresto"

A: "ehm.. tutti i log erano sull'istanza. Non abbiamo alcuna copia."

Una pratica utile e piuttosto comune a tal proposito consiste nell'utilizzo di un agent che esporti periodicamente i log in una destinazione diversa dall'istanza EC2. Questo è il modo migliore per evitare di trovarsi in situazioni simili. Ma non basta!

...

B: "Certo! L'agent per l'esportazione dei log era installato correttamente!"

A: "Probabilmente è andato in crash."

È qui che il lifecycle hook EC2_INSTANCE_TERMINATING corre in nostro aiuto: esso permette di **esportare i log in una nuova destinazione,** ad esempio un bucket S3.

Cosa è andato storto?

A volte risulta utile loggarsi sull'istanza EC2 per capirlo. Combinando insieme EC2 Auto Scaling Lifecycle Hooks, un Topic SNS e una Lambda Function è possibile **setuppare un sistema di notifiche che ci avvisi in caso di istanza unhealthy terminata.** In questo tipo di soluzione, la Lambda Function avrà il compito di esaminare il motivo del passaggio di un'istanza allo stato di Terminating distinguendo tra istanze healthy, ma non più necessarie e istanze terminate in quanto *unhealthy*.

Hands-on!

Vediamo insieme come **setuppare un EC2 Auto Scaling Lifecycle Hook** che consente di completare la configurazione dell'istanza prima che essa passi allo stato *InService*.

La configurazione consiste in uno User Data script definito nel Launch Template dell'Auto Scaling group.

```
#!/bin/bash
apt update && \
apt install snapd && \
snap install aws-cli --classic && \
apt install -y apache2 && \
ufw allow 'Apache' && \
INSTANCE_ID="`wget -q -O - http://instance-data/latest/meta-data/instance-id`" && \
/snap/bin/aws autoscaling complete-lifecycle-action --lifecycle-action-result CONTINU
E --instance-id $INSTANCE_ID --lifecycle-hook-name test-hook --auto-scaling-group-nam
e test-asg --region eu-west-1 || \
/snap/bin/aws autoscaling complete-lifecycle-action --lifecycle-action-result ABANDON
--instance-id $INSTANCE_ID --lifecycle-hook-name test-hook --auto-scaling-group-name
 test-asg --region eu-west-1
```

Lo User Data installerà un Apache Web Server su un'istanza Ubuntu 18.04. Una volta completata l'installazione, lo script concluderà la lifecycle action specificando CONTINUE come risultato, autorizzando l'Auto Scaling group a portare l'istanza in stato InService. Nel caso in cui qualcosa vada storto, il risultato fornito al comando complete-lifecycle-action sarà ABANDON. L'istanza sarà quindi terminata dall'Auto Scaling group.

In questo processo occorre porre particolare attenzione sui parametri **–lifecycle-hook-name** e **–auto-scaling-group-name** utilizzati nello script di configurazione; è necessario che essi corrispondano rispettivamente al **nome dell'EC2 Auto Scaling Lifecycle Hook** e al **nome dell'EC2 Auto Scaling group** che andremo a creare.

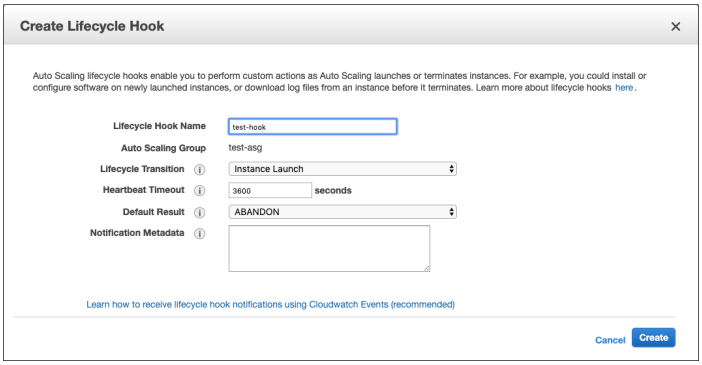
Prima di tutto, **creiamo il Launch Template** specificando come sorgente AMI **ubuntu/images/hvm-ssd/ubuntu-bionic-18.04-amd64-server-20190722.1** e come User Data lo script che abbiamo preparato durante lo step precedente.

A questo punto potremo creare un nuovo Auto Scaling group basato sul Launch template appena creato. Durante la configurazione dell'Auto Scaling group, specifichiamo **una o più subnet** che consentano all'istanza di installare Apache Web Server.

Prima di procedere, assicuriamoci che i valori di **min** e di **DesiredCapacity** iniziali siano impostati su **zero**. Il **parametro di max** iniziale potrà essere **settato a 1.**

Dalla tab del Lifecycle Hooks dell'Auto Scaling group clicchiamo su **Create Lifecycle Hook.**

Configuriamo ora il Lifecycle Hook come mostrato:



Per testare il comportamento del lifecycle hook, possiamo **incrementare il valore del parametro DesiredCapacity di uno.** Così facendo, noteremo che **una nuova istanza verrà aggiunta all'Auto Scaling group.** Dalla tab Auto Scaling group dell'istanza, dovremmo quindi notare nella colonna *Lifecycle* dell'istanza appena aggiunta lo stato di Pending:Wait. Questo significa che il Lifecycle dell'istanza rimarrà in stato Pending fino a che il comando complete-lifecycle-action non verrà invocato dallo script User Data.

Quando lo stato cambierà in InService, potremo accedere via SSH alla nuova istanza EC2 per controllare lo stato del Web Server Apache attraverso il comando

sudo systemctl status apache2

Se il Web Server Apache è stato configurato correttamente, otterremo una risposta simile alla seguente:

- apache2.service – The Apache HTTP Server

Loaded: loaded (/lib/systemd/system/apache2.service; enabled; vendor preset: enabled)

Drop-In: /lib/systemd/system/apache2.service.d

└─apache2-systemd.conf

Active: active (running) since Wed 2019-10-02 12:56:20 UTC; 3min 30s ago

Main PID: 3334 (apache2)

Tasks: 55 (limit: 1125)

CGroup: /system.slice/apache2.service

├─3334 /usr/sbin/apache2 -k start

├─3336 /usr/sbin/apache2 -k start

└─3337 /usr/sbin/apache2 -k start

Solo dopo aver superato la fase di configurazione, lo stato dell'istanza cambierà da Pending a InService.

Soddisfatti?

In questo articolo vi abbiamo mostrato il funzionamento dei Lifecycle Hooks descrivendo brevemente come essi possono tornarci utili durante lo sviluppo quotidiano.

Vi è capitato di utilizzarli in altre configurazioni? Raccontatecelo nei commenti!

Arrivederci al prossimo articolo!

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