AWS Elemental MediaConvert: transcodifica avanzata per piattaforme di streaming

3 Marzo 2023 - 9 min. read

AWS Elemental MediaConvert | AWS Elemental MediaConverter | Media Streaming

Introduzione

Vi siete mai chiesti che cosa si intende quando si parla di transcodifica?

Vi siete mai chiesti come le grandi piattaforme di streaming riescano a fornirvi un servizio così completo e ricco di funzionalità?

In questo articolo capiremo cos'è, come si fa, e perché si utilizza la transcodifica; affronteremo prima una soluzione classica con l'utilizzo di FFmpeg e successivamente la accosteremo al servizio AWS Elemental MediaConvert.

Cos'è la transcodifica e perché serve

Possiamo definire transcodifica il processo che permette, data una sorgente audio/video, di ottenerne versioni con differenti formati e qualità. Il processo di elaborazione può essere fatto, sia in real-time sia in post produzione.

Al giorno d'oggi i dispositivi di riproduzione sono molteplici e differiscono tra loro in risoluzione, dimensioni e codec video supportati.

L'unica possibilità che abbiamo per raggiungere tutti i dispositivi è transcodificare i contenuti della nostra piattaforma.

Partendo da un contenuto realizzato in 4K andremo a realizzarne più versioni differenti per risoluzione, codec , bitrate ecc.

Oggi la possibilità di impostare la risoluzione del video è ormai considerata una feature quasi obbligatoria, soprattutto se si intende competere con le piattaforme più in voga. Questo porta con sé incredibili benefici sia per l'utente, che per chi gestisce la piattaforma.

Se durante la riproduzione di un video l'utente decidesse di abbassare la risoluzione, ad esempio, questo ridurrebbe il consumo della sua rete dati.

Un'altra cruciale attività, svolta in fase di transcodifica, è la frammentazione.

Questa operazione ci permette di suddividere un unico grande video in sottoparti della stessa durata. Per organizzare la ricomposizione del video viene utilizzato uno specifico file adibito ad orchestrare tutti i singoli frammenti che verranno trasmessi dal player.

Quest'operazione permette il buffering del video, evitando così all'utente di scaricarsi sul dispositivo l'intero contenuto prima della visualizzazione "live/streaming".

Le più moderne piattaforme di streaming live adoperano la frammentazione in real time così da consentire una fruizione fluida del contenuto in onda.

Guida galattica per transcodificatori

Prendiamo in considerazione i principali codec video in funzione dei formati e dei dispositivi più diffusi. Sotto una tabella dei principali codec video utilizzati e le rispettive risoluzioni supportate:

Codec	Descrizione	Risoluzione
H.264/AVC	Uno dei codec più diffusi, utilizzato per la codifica di video ad alta definizione . Offre un'ottima qualità video con una larghezza di banda ridotta.	HD - 720p con 1280x720px FullHD - 1080p 1920x1080px
H.265/HEVC	Successore di H.264, offre una qualità video ancora migliore con una larghezza di banda ridotta.	fino a 8k - 7680x4320p

		x
VP9	Sviluppato da Google, è un codec aperto e gratuito che offre una qualità video simile a quella di H.265 (fino a 4k) a un costo minore.	Fino a 4k - 3840x2160p x
AV1	Sviluppato da un consorzio di aziende tecnologiche, offre una qualità video simile a quella di H.265/HEVC a un costo inferiore	fino a 8k - 7680x4320p x
MPEG-2	Utilizzato per la codifica di DVD e TV digitale, offre una buona qualità video a una larghezza di banda moderata.	SD - 720x576px o 720x480px HD - 720p con 1280x720px FullHD - 1080p 1920x1080px
MPEG-4	Utilizzato per la codifica di video su Internet e dispositivi mobili, offre una buona qualità video con una larghezza di banda ridotta.	SD - 720x576px o 720x480px HD - 720p con 1280x720px FullHD - 1080p 1920x1080px
Theora	Un codec video open source, offre una buona qualità video con una larghezza di banda ridotta.	SD - 720x576px o 720x480px HD - 720p con 1280x720px

Principali codec video e rispettive risoluzioni supportate.

Per convertire il nostro contenuto in questi formati possiamo utilizzare due metodi:

Il metodo “classico”

Il termine transcodifica, dalla maggior parte degli utilizzatori, è associato al framework FFmpeg.

Citando dal sito ufficiale:

FFmpeg is the leading multimedia framework, able to decode, encode, transcode, mux, demux, stream, filter and play pretty much anything that humans and machines have created. It supports the most obscure ancient formats up to the cutting edge. No matter if they were designed by some standards committee, the community or a corporation. It is also highly portable: FFmpeg compiles, runs, and passes our testing infrastructure FATE across Linux, Mac OS X, Microsoft Windows, the BSDs, Solaris, etc. under a wide variety of build environments, machine architectures, and configurations.

Una volta installato sulla nostra macchina, potremo utilizzarlo per effettuare le operazioni di transcodifica in locale.

A titolo di esempio, se volessimo diminuire (o aumentare) il bitrate a 24 di un file video utilizzeremmo il seguente comando:

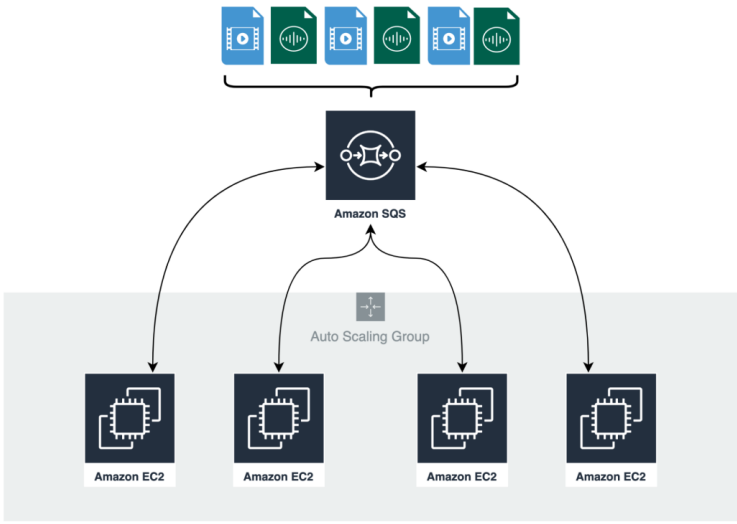
```
ffmpeg -i input.avi -r 24 output.avi
```

Se ci trovassimo con la necessità di transcodificare numerose sorgenti in molteplici formati, potremmo adottare una soluzione scalabile utilizzando EC2, SQS e autoscaling group.
Per elaborare i nostri video utilizzeremo una batteria di EC2 disposte all'interno di un autoscaling group; facendo ciò, la nostra infrastruttura sarà capace di gestire sia aumenti del traffico (e quindi delle elaborazioni necessarie) creando in maniera autonoma nuove EC2, sia 'eventuale fallimento di una delle AZ di AWS.

Come metrica di scaling ci basterà implementare una coda SQS che andrà a fare da buffer per i lavori da eseguire permettendoci di non scalare eccessivamente qualora le richieste fossero gestibili con l'attuale numero di macchine, e di sfruttare al meglio le risorse computazionali che le EC2 ci metteranno a disposizione.

La coda SQS, inoltre, fornisce anche altri vantaggi come la gestione delle dead-letter queue, ovvero la possibilità di riconoscere i lavori che per un qualunque motivo non sono stati portati a termine dalle nostre EC2 e quindi eventualmente riproporlo per poter effettuare un nuovo tentativo. Per utilizzare questa unione dei due servizi però

sarà necessario creare una metrica custom su AWS dato che il numero di elementi nella coda non è riconosciuto nativamente dagli autoscaling group.



Coda SQS per transcodifica parallelizzata su una flotta di EC2.

Utilizzando le giuste tipologie di EC2 avremmo sicuramente una grande potenza di calcolo per completare i nostri lavori, ma andremmo ad incappare in diversi svantaggi:

- Elaborazione di FFmpeg non parallelizzabile: due o più istanze EC2 non possono condividere la stessa elaborazione (proveniente da un'unica sorgente). Questo impedisce di sfruttare a pieno la potenza di calcolo delle singole EC2 e quindi l'utilizzo della metrica di CPU per lo scaling;
- Gestione dell'istanza EC2: patching, gestione spazio disco e fail dell'istanza vanno gestiti personalmente;

AWS Elemental MediaConvert

Quando si parla di transcodifica su AWS sicuramente ci si è imbattuti in AWS MediaConvert.

Utilizzando questo servizio completamente gestito, avremo accesso a moltissime funzionalità che ci aiuteranno durante il nostro processo di transcodifica sia da tecnici già esperti, che da novizi che si stanno approcciando alla tematica.

Elemental MediaConvert è strutturato in JOBS TEMPLATE, JOBS e QUEUE.

JOB TEMPLATE

Il Job Template crea un blueprint riutilizzabile più e più volte per organizzare in maniera ottimale quali e quanti formati diversi vorremmo ricavare dal nostro file sorgente.

Nel corso della creazione dei nostri template ci verranno fornite moltissime funzionalità per “ritoccare” il nostro video come l’aggiunta di particolari sezioni, la modifica di formati e codec, la regolazione della risoluzione e della frammentazione qualora ne volessimo una e molte altre che ci permetteranno di ricavare il miglior risultato possibile dal file d’origine.

Come citato all’inizio dell’articolo, potremmo aver bisogno di creare diversi formati per lo stesso video, magari cambiandone semplicemente la risoluzione così da potersi adattare al dispositivo di riproduzione. Per evitare tutte le volte di configurare tutti questi e altri parametri, potremmo creare un template per ognuna delle risoluzioni per cui intendiamo produrre dei contenuti.

Supponendo che i nostri principali formati saranno HD, FullHD e 4K, al termine delle nostre configurazioni avremo 3 template pronti per essere richiamati in fase di elaborazione dal servizio MediaConvert.

Eventualmente potremmo anche decidere di creare un unico template che avrà al suo interno diversi output suddivisi per formato, qualità e codec (HD, FullHD, 4K ….). Qui è demandato al gestore della piattaforma selezionare la pratica che più lo aiuta a presentare efficientemente i suoi contenuti.

JOBS

Possiamo decidere di creare i nostri job partendo da:

- un template creato precedentemente dove dovremo solo impostare input e output nel caso in cui non siano già stati impostati nel template. Quando creeremo un nuovo Job dovremo specificare dove sono salvati i file su S3;
- Un file JSON che conterrà tutte le nostre configurazioni. Questa pratica è vantaggiosa quando si hanno tra le mani moltissime varianti di template e può aiutare un tecnico non esperto di Elemental MediaConvert ad approcciarvisi utilizzando un linguaggio più comune come quello JSON. Avendo la possibilità di esternalizzare su un file le nostre configurazioni, sarà possibile anche creare uno

script/programma ad hoc che compilerà automaticamente i campi del nostro JSON per poi importarlo e finalmente avere il nostro JOB;

- Copiare il JSON risultante da un job esistente. Una volta creato il job abbiamo la possibilità di leggere (e copiare) il JSON risultante con tutte le configurazioni che abbiamo impostato e quindi duplicare il nostro JOB per modificare pochi parametri senza doverlo rifare da zero.

QUEUE

Le queue ci consentono di schedulare le procedure di transcodifica che vorremmo fare sui nostri contenuti e di pararrelillazere le lavorazioni. Come per altri servizi esiste sia la modalità *on-demand* pagata a minuti di elaborazione, oppure una soluzione *riservata* dove dovremo impegnarci a pagare 12 mesi di elaborazione.

Questa distinzione diventa molto vantaggiosa quando si inizia ad utilizzare Elemental MediaConvert in maniera massiva e si desidera risparmiare sulle bollette di fine mese.

Tirando le somme

Quindi, quale soluzione utilizzare?

Dopo aver analizzato i 2 approcci di transcodifica, possiamo affermare che entrambe le possibilità sono più che valide; la loro implementazione dipende dalle tue necessità specifiche.
Da una parte utilizzare una o più EC2 ci permetterebbe di progettare una infrastruttura in HA e in grado di gestire i picchi di carico, ma dovremmo fare i conti con costi sicuramente più elevati, soprattutto all'aumentare della potenza computazionale necessaria e del livello di HA che si intende adottare.

Dall'altra parte,una soluzione completamente managed basata su AWS Elemental MediaConvert soppianterebbe tutti i problemi di gestione dell'infrastruttura e ci darebbe la possibilità di utilizzare una GUI ben realizzata per effettuare tutta la configurazione della transcodifica desiderata.

Sebbene i costi siano diversi in base alla soluzione scelta, dobbiamo tenere in considerazione che un confronto di questo tipo sarebbe come mettere vicine mele con pere. Sicuramente la facilità con cui potrebbe variare il numero di lavori da eseguire non aiuta a quantificare una possibile disparità di costi.

Un fattore che potrebbe spostare l'ago della bilancia verso l'utilizzo di Elemental MediaConvert è la possibilità di integrarlo nativamente con altri servizi di AWS per automatizzare tutto il flusso. I più facili da implementare sarebbero sicuramente S3 e event bridge che ci aiuterebbero a conservare i nostri contenuti e a far partire automaticamente le nostre trascodifiche alla pubblicazione di una nuova sorgente.

Un'altra grossa differenza sta nella complessità che deriva dall'utilizzo dei due approcci. Se da un lato AWS MediaConver ci guida con delle impostazioni di fabbrica e con un interfaccia abbastanza esplicativa, dall'altra parte FFmpeg ci mette a disposizione moltissime impostazioni che, per un utente alle prime armi, potrebbero risultare fuorvianti e complicate.

Vi siete mai cimentati nello streaming di contenuti? Raccontateci nei commenti come ci siete riusciti!

A presto con un nuovo 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!

Riccardo Fragnelli

DevOps @ beSharpI was born on-prem as a Dev before landing on the "Cloud side of IT". With AWS I discovered a whole new branch of IT that fascinates me more and more; I'm always ready for the next big thing!I'm the fussiest man I know on earth and quite lazy. I like spending my free time jumping between video games and RPGs.

Antonio Callegari

DevOps Engineer @ beSharp. Nasco sistemista "classico" innamorato di hardware, ma passo volentieri al lato oscuro: il Cloud! Preferisco sempre le cose fatte a mano, ma non disdegno un po' di sana automazione (se fatta con criterio...)Nel tempo libero allestisco e calco palchi e mi dedico alla mia famiglia