AWS Elemental MediaConvert: advanced transcoding for streaming platforms

3 March 2023 - 9 min. read

AWS Elemental MediaConverter *Media Streaming*

Introduction

Have you ever wondered what is meant by transcoding?

Have you ever wondered how the big streaming platforms can provide you with such a complete and feature-rich service?

In this article, we will understand what transcoding is, how it works, and why transcoding is used. We will first face a classic solution using FFmpeg and then we will combine it with the AWS Elemental MediaConvert service.

What is transcoding and why is useful

We can define transcoding as the process that allows, given an audio/video source, to obtain multiple versions with different formats and resolutions. This elaboration can be done, both in real-time and in post-production.

Today there are many playback devices available and they differ from each other in resolution, size, and supported video codecs.

The only way we have to reach all devices is to transcode the contents of our platform.

Starting from content recorded in 4K we will create several versions with different resolutions, codecs, bitrates, etc.

Nowadays the ability to set the video resolution is considered an almost mandatory feature, especially if you intend to compete with the most popular streaming websites. This brings incredible benefits both for the users and for those who have to manage the platform.

If while playing a video the user decides to lower the resolution, this would reduce the consumption of his data network.

Another crucial activity, carried out during transcoding, is fragmentation.

This operation allows us to divide a single large video into sub-parts of the same duration. To organize the recomposition of the video, a specific file is used to orchestrate all the single fragments that will be transmitted by the media player.

This operation allows the buffering of the video, thus preventing the user from downloading the entire content to the device before "live/streaming" viewing. The most modern live streaming platforms use real-time fragmentation to allow users to have a fluid use of the broadcasted content.

Galactic guide for transcodificators

Let's consider the main video codecs according to the most popular formats and devices. Below is showed a table of the main video codecs used and their respective supported resolutions:

Codec	Description	Resolution
H.264/AVC	One of the most popular codecs, used for encoding high-definition video. It offers great video quality with low bandwidth.	HD - 720p con 1280x720px FullHD - 1080p 1920x1080px
H.265/HEVC	Successor of H.264, it offers even better video quality with lower bandwidth.	To 8k - 7680x4320px
VP9	Developed by Google, it is a free and open codec that offers video quality similar to H.265 (up to 4k) at a lower cost.	To 4k - 3840x2160px
AV1	Sviluppato da un consorzio di aziende tecnologiche, offre una qualità video simile a quella di H.265/HEVC a un costo inferiore	To 8k - 7680x4320px

MPEG-2	Developed by a consortium of technology companies, it offers video quality similar to H.265/HEVC at a lower cost	SD - 720x576px o 720x480px HD - 720p con 1280x720px FullHD - 1080p 1920x1080px
MPEG-4	Used for encoding video on the Internet and mobile devices, it offers good video quality with low bandwidth.	SD - 720x576px o 720x480px HD - 720p con 1280x720px FullHD - 1080p 1920x1080px
Theora	An open source video codec, it offers good video quality with low bandwidth.	SD - 720x576px o 720x480px HD - 720p con 1280x720px

Table of the main video codecs used and their respective supported resolutions

To convert our content into these formats we can use two approaches:

The "classic" approach

The term transcoding is associated by most users with the FFmpeg framework.

Quoting from the official site:

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.

Once installed on our machine we can use it to carry out transcoding operations locally.

As an example, if we wanted to decrease (or increase) the video file's bitrate to 24 we would use the following command:

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

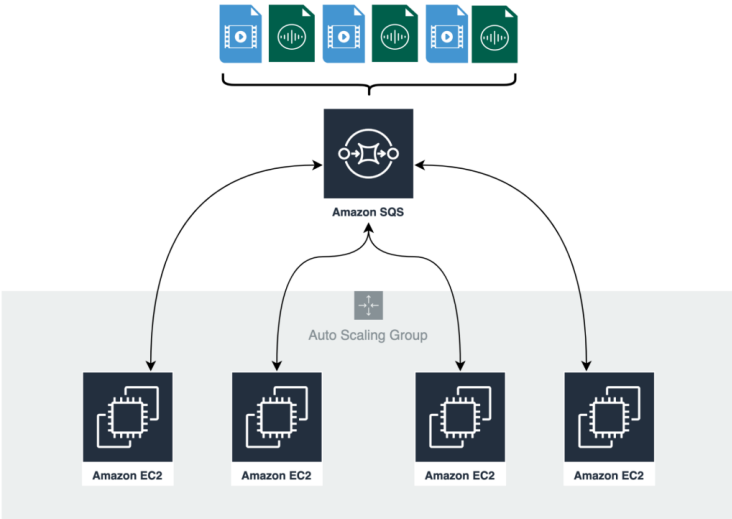
If we find ourselves with the need to transcode many sources into multiple formats, we could adopt a scalable solution using EC2, SQS, and autoscaling groups.

To process our videos we will use a battery of EC2 arranged within an autoscaling group; by doing so, our infrastructure will be capable of managing increases in traffic

(and as a consequence of the necessary processing) by autonomously creating new EC2s and managing the possible failure of one of the AWS AZs.

As a scaling metric, it will be enough for us to implement an SQS queue which will act as a buffer for the jobs to be performed, preventing us from excessive scaling when the requests are manageable with the current number of machines, and making the most of the computational resources that EC2 will make available. The SQS queue also provides other advantages such as the management of dead-letter queues, i.e. the possibility of recognizing jobs that have not been completed by our EC2 for any reason and then possibly retrying them.

To use this union of the two services, however, it will be necessary to create a custom metric on AWS since the number of elements in the queue is not natively recognized by the autoscaling groups.



SQS queue for parallelized transcoding on an EC2 fleet

Using the right types of EC2 we will certainly have the great computing power to complete our work, but we would run into several disadvantages:

- Non-parallelizable FFmpeg processing: two or more EC2 instances cannot share the same processing (coming from a single source) preventing the full exploitation of the computing power of the individual EC2 and therefore the use of CPU metrics for scaling;
- EC2 instance management: patching, disk space management, and instance fail must be managed personally.

AWS Elemental MediaConvert

When it comes to transcoding on AWS, AWS Elemental MediaConvert is the very first service we meet.

Using this fully managed service, we will have access to many features that will help both already experienced technicians and novices who are approaching the subject during the transcoding process.

Elemental MediaConvert is structured in JOBS TEMPLATE, JOBS, and QUEUES.

JOB TEMPLATE

Job templates create a blueprint that can be reused over and over again to optimally organize which and how many different formats we would like to derive from our source file.

During the creation of our templates, we will be provided with many features to "touch up" our video by adding particular sections, modifying formats, and codecs, adjusting the resolution and fragmentation if we want one, and many others that will allow us to obtain the best possible result from the source file.

As mentioned at the beginning of the article, we may need to create different formats for the same video perhaps by simply changing the resolution to be able to adapt to different playback devices. To avoid having to configure all these, and similar parameters, every time, we can create a template for each resolution we intend to produce content for.

Assuming that our main formats will be HD, FullHD, and 4K, at the end of our configurations we will have 3 templates ready to be recalled during processing by the MediaConvert service.

Eventually, we will also be able to decide to create a single template that will have different outputs divided by format, quality, and codec (HD, FullHD, 4K ....). t is up to the user to adopt the best practice according to his needs to efficiently produce his content.

JOBS

We can decide to create our jobs starting from:

- a previously created template. In this case, we will only have to set inputs and outputs in case they have not already been set in the template. When creating a new Job we will need to specify where the location of the files saved on S3;
- A JSON file that will contain all of our configurations. This practice could be very advantageous in the case of many template variants and could help a non-Elemental MediaConvert expert to approach them using a more common language such as JSON. The possibility of outsourcing our configuration into a file, allows also the creation of an ad hoc script/program to automatically fill in the fields of our JSON and then import it and finally have our JOB;
- A copy of a JSON resulting from an existing job. Once the job has been created we can read (and copy) the resulting JSON with all the configurations we have set and then duplicate our JOB to modify a parameter without having to create it from scratch.

QUEUE

The queues allow us to schedule the transcoding procedures that we would like to do on our contents and to parallelize the processes. Like for other AWS services, an on-demand model is available for paying only for the minutes of processing, along with a flat payment model that requires a 12-month processing charge.

Summing up: AWS Elemental MediaConvert vs FFmpeg

So, what's the best approach?

After analyzing the 2 transcoding approaches, we can say that choosing one or the other strongly depends on your specific needs.

On the one hand, using one or more EC2 would allow us to design a Highly Available infrastructure also capable of managing load peaks. In this scenario, costs would certainly be high, especially as the computational power required and the HA level increase.

On the other hand, a solution based on the fully managed AWS Elemental MediaConvert services would make us free from any infrastructure management issue and would allow us to benefit from a well-crafted GUI for all the transcoding setups.

This solution also has its drawbacks: the costs could increase dramatically. It would be necessary to keep the monthly invoices under control to understand when switching

from the classic on-demand to a reserved queue becomes a suitable choice. Although costs would be different according to the service chosen, we must say that such a comparison would be like comparing apples with oranges. The fact that the number of jobs to be performed may easily vary does not help us to quantify the difference in costs.

A plus for AWS Elemental MediaConvert is the possibility of natively integrating it with other AWS services to automate the entire flow. The easiest services to implement would certainly be S3 and event bridge which can help us to store our content and automatically start our transcodes when a new source is published, paving the way with automation and event-driven infrastructures.

Another big difference lies in the complexity that comes from the two approaches.

If on the one hand, AWS MediaConver guides us with default settings and with a fairly explanatory interface, on the other hand, FFmpeg provides us with many settings which, for a novice user, could be misleading and complicated.

Have you ever dealt with content streaming? Tell us about your experience in the comments!

See you soon on **#Proud2beCloud** for a brand-new article!

About Proud2beCloud

Proud2beCloud is a blog by beSharp, an Italian APN Premier Consulting Partner expert in designing, implementing, and managing complex Cloud infrastructures and advanced services on AWS. Before being writers, we are Cloud Experts working daily with AWS services since 2007. We are hungry readers, innovative builders, and gem-seekers. On Proud2beCloud, we regularly share our best AWS pro tips, configuration insights, in-depth news, tips&tricks, how-tos, and many other resources. Take part in the discussion!

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.Born as a hardware-addicted, “classic” system engineer, I also like jumping to the dark side: the Cloud! And you know the effect that this mix can make :) Hand-making is my first choice, but a bit of high-quality automation is welcome in my projects.My free time is split between my family and the music, both as a player, and sound engineer.
