Cloud Databases with Amazon Web Services

Project for Course INFO H-415 Advanced Databases
with prof. Esteban Zimányi
2015-2016

Sarah Van Bogaert & Carlos Zamalloa

Introduction

A “cloud database” is any kind of database that runs over the cloud system platform. Its main characteristic is its scalability because it is based on on demand computing as well as its flexibility thanks to the many different applications found in a cloud service. For this project, we used the service of Amazon, called Amazon Web Services (AWS), which offered us a 12 month free trial period. To learn how to use some of the AWS features, we created a small web application, where a user can search for movies and see their trailers.

In a first chapter, we briefly describe AWS as well as the two solutions offered by Microsoft and Google. In chapter 2, our application is described as well as how we proceeded with AWS. In that chapter is also presented our general experience with the AWS software.

1. AWS and Third Party Solutions

We introduce here the three major players in the cloud services platform: Amazon Web Services, Microsoft Azure and Google Cloud Platform, for the reason that they share most of today's market.

Amazon has a very strong background with the development of AWS. While AWS was officially launched in 2006, It is one of the first companies that went into the business of cloud computing. Customers of AWS today include NASA, Netflix and the CIA. Microsoft Azure was launched in 2008 as Windows Azure while Google Cloud Storage was launched in 2010. Since the Google solution is newer, it doesn't offer yet as much applications and flexibility as the other two.

The cloud database systems are part of a global solution that can share virtual computers for cloud computing (like Amazon EC2) and simple cloud storage (like Amazon S3). Besides, the global solution provides APIs to easily access the different parts of the system. About relational databases, which represent only a small part of the total services offered, Microsoft Azure, the solution of Microsoft uses the relational model database of MS SQL-Server running in the cloud. The correspondence in Amazon Web Services is called Amazon Relational Database Service (Amazon RDS) where can be found, as a difference with Microsoft, many known database systems like MySQL, MS SQL-Server and Oracle. Besides MySQL, Google offers NoSQL as well as part of their solution.

One of the advantages of using cloud databases is the scalability offered by these kind of services. Let's imagine that we are a company of a service of video in streaming as Netflix, Amazon Instant Video or Hulu. If the growth of our company goes faster or slower than expected we don't have problems to manage that as the computer force and storage is handled by Amazon itself. Let's now take the example of Airbnb, the sharing rental service that is being more and more used lately: they only need 5 people in the IT department because AWS is in charge of everything concerning the database.

About the cost there is a variation between the different solutions but we cannot make a direct comparison because the cost depends on many different factors. As an example, the charge of prices are divided in the following cases in AWS:

-General Purpose: having a base price of around 0.0050 american dollars per one virtual CPU and 512Mb of RAM, the price is raised to 0.01 american dollars for the servers in Japan.

- GPU instances: from 0.65 dollars per hour, this module is oriented as the name says, for graphic demanding power. One example of solution based on this method is the Cloud Gaming, like PlayStation Now.

- Memory Optimized: beginning at 0.175 american dollars per hour using 1 SSD hard drive of 32 Gb.

- Storage Optimized: at 0.853 american dollars per hour using one SSD of 800 Gb.

We decided to use Amazon for our project for the reason that they offer a trial period of 12 months with most of the services activated. On the contrary, if we would have used Microsoft Azure, we would have paid since the first minute to create and manipulate a database in the cloud.

In figures 1.1, 1.2 and 1.3, you will find an overview of all applications offered by the three solutions. Inside one system, the services can interact with each other but can also be used independently. The different services offer enough tools and flexibility to make a robust application.

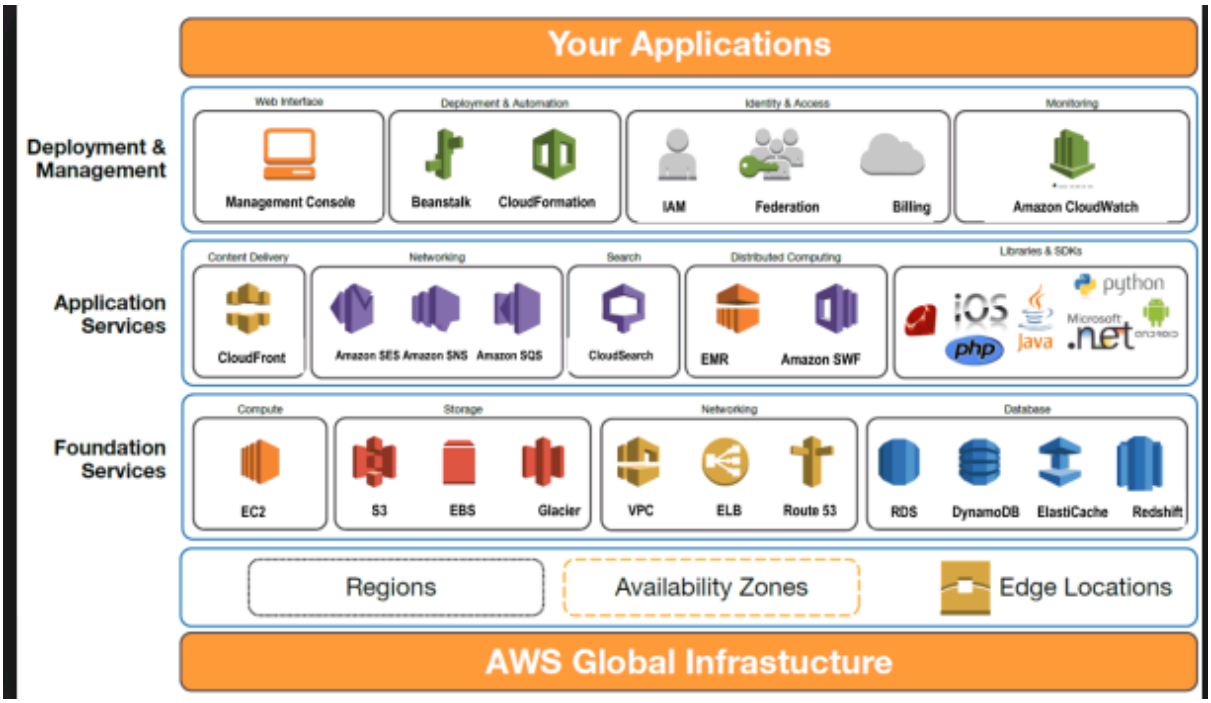


Figure 1.1 : Amazon Web Services Architecture

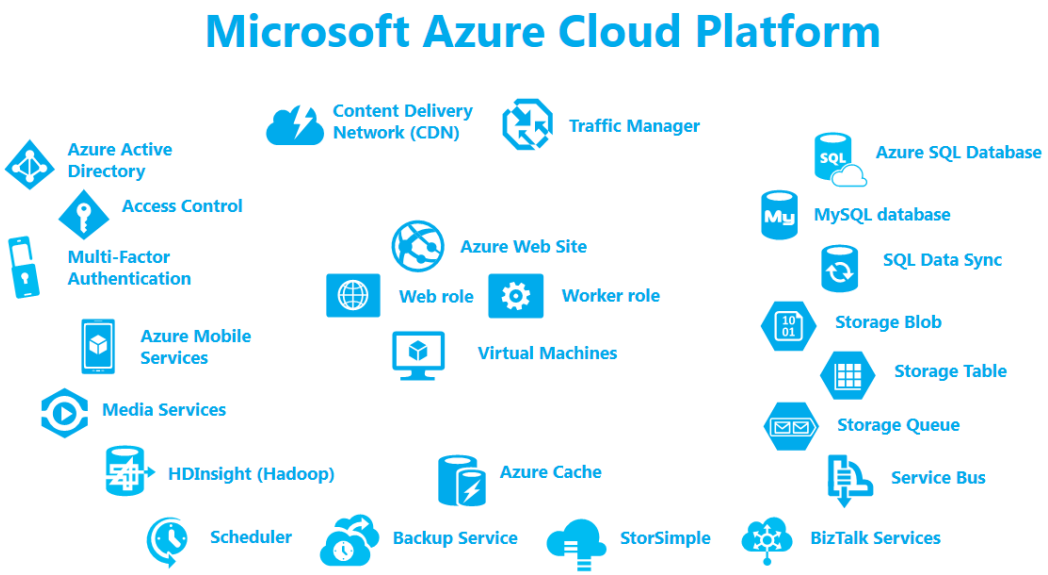


Figure 1.2: Microsoft Azure Services

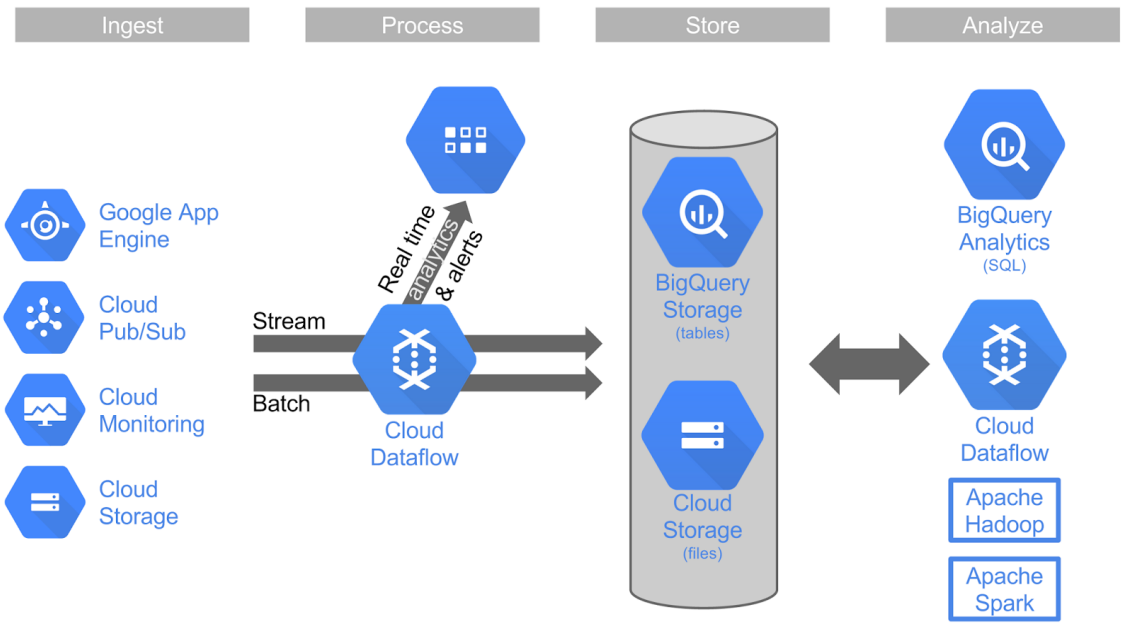


Figure 1.3 Google Cloud Platform

2. Our Movies and Trailers Web App

http://trailersapp-dev.elasticbeanstalk.com/movies/search

Our application consists of a simple website where one can search for movies and visualize their trailers. The two pages of the website are shown in figure 1 and 2 respectively.

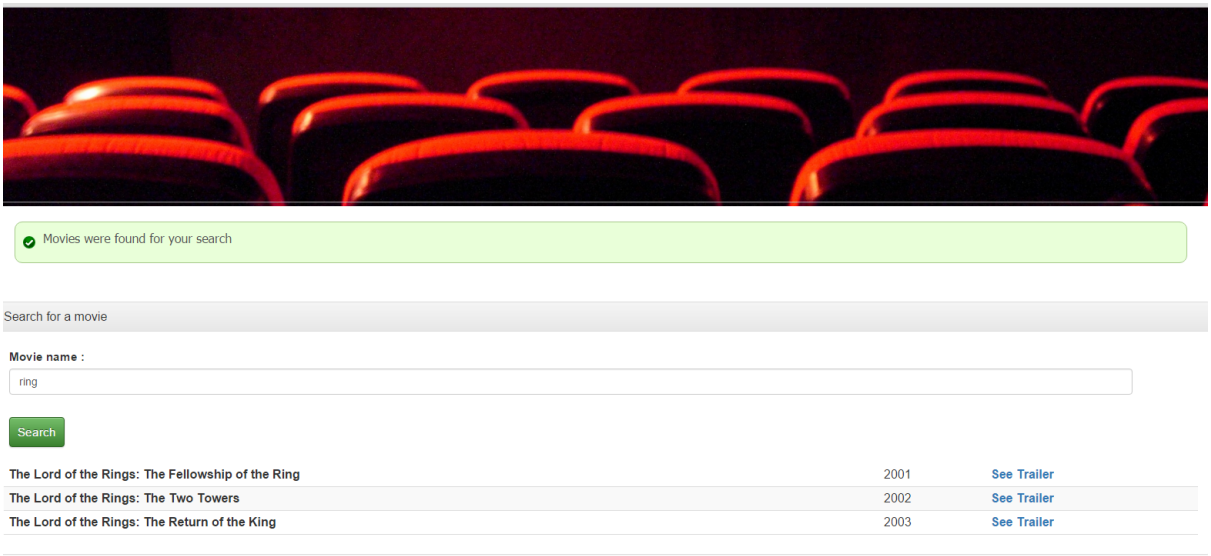


Figure 2.1: Main page of our application

Figure 2.2: Page shown when clicking on “see trailer”

Our application uses Amazon S3 to store the trailers, Amazon Cloudfront for the streaming, Amazon RDS for the relational database and finally, Amazon Elastic Beanstalk as a server for the web page. Below is explained in more details how we used these services for our Movies and Trailers app.

During this project, we worked with the AWS management console to manipulate the different services of Amazon. However, solutions exist where the management console is integrated in the browser, i.e. to make it easier to install and work with Amazon S3 buckets and objects. Examples are S3Fox for Firefox [3] or Extended S3 Browser for Chrome.

RDS

Amazon RDS is a relational database system running with many different SQL systems inside the cloud of Amazon. We created a MySQL database for the purpose of this project, as it is the default service offered by AWS. As our application is very simple and only shows the title of the movies, the year they were launched and the trailer, we only needed one table in our database (see figure 2.3). Please note that in this table only the trailer filename is stored and not the mp4 file itself, which is stored in Amazon S3.

Advantages

- Using amazon RDS in the application was like using any other database system with the advantages of Amazon RDS: possibility to easily clone, scale out, make automatic backups, etc.

- Creating a database and make it work is very simple with the default configuration.

Disadvantages

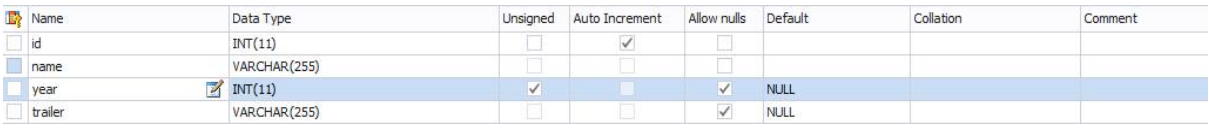
- The time to create the database was longer than 15 minutes. As the AWS application is working in the cloud, we cannot predict such timings. However, this timing problem may be due to the fact that we are using a trial-version of the software that offers less computing power.

- With the trial version of AWS, some options were not allowed while creating the database.

- Making reference to the initial creation of the database, someone who has a knowledge of a traditional UI server or console would need some time to adapt to the new interface provided by Amazon Web Services.

- The creation of the MySQL database took around 5 minutes before it was available for usage. This may be caused by the fact that we use a free version of the software.

- The execution of the queries is very slow. For instance, a query of less than 1 Mb (768 kb/3411 lines) took 7:12 minutes. However, this may again be due to the trial version of the software we use.



Name	Data Type	Unsigned	Auto Increment	Allow nulls	Default	Collation	Comment
id	INT(11)	☐	☑	☐			
name	VARCHAR(255)	☐	☐	☐			
year	INT(11)	☑	☐	☑	NULL		
trailer	VARCHAR(255)	☐	☐	☑	NULL		

Figure 2.3: Table “Movies” in our relational database

CloudFront

Amazon CloudFront is a content delivery web service we handled for the streaming of our videos. It integrates with other AWS services like Amazon S3. In our case, we stored our videos with Amazon S3 (a simple cloud storage service explained below) where CloudFront could fetch them to render them in a user friendly way in our web application. The tutorial from [1] explains in details how to use CloudFront in combination with S3 and JW Player media player. We proceeded in three steps:

1. Upload the JW Player files in the Amazon S3 Bucket, i.e. in the same folder where the videos are stored (see figure 2.4).
2. Create CloudFront Web and RTMP Distributions.
3. Embed the videos in the web application

Please note that RTMP is the protocol owned by Adobe (previously Macromedia) for video streaming.

Figure 2.5 shows in more details how to proceed for step 3. To use the video files stored in the S3 bucket, only their filenames are needed as well as the web and RTMP distribution names from CloudFront.

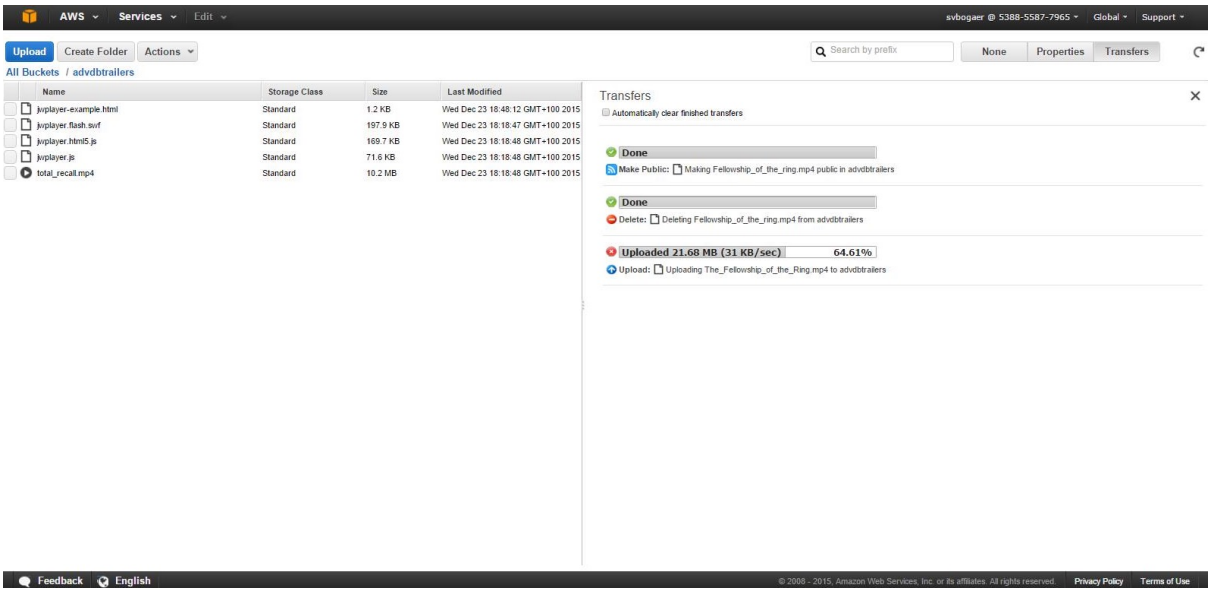


Figure 2.4. The Amazon S3 bucket used for the Movies and Trailer application

```
<HTML>
<HEAD>
<TITLE>Amazon CloudFront Streaming with JW Player 6</TITLE>

<!-- Call the JW Player JavaScript file, jwplayer.js.
Replace WEB-DISTRIBUTION-DOMAIN-NAME with the domain name of your
CloudFront web distribution, for example, d1234.cloudfront.net
(begins with "d"). This causes a browser to download the JW Player file
before streaming begins.
-->

<script type='text/javascript' src='https://WEB-DISTRIBUTION-DOMAIN-NAME/jwplayer.js'></script>

</HEAD>

<BODY>
<H1>This video is streamed by CloudFront and played by JW Player 6.</H1>

<!-- Replace RTMP-DISTRIBUTION-DOMAIN-NAME with the domain name of your
RTMP distribution, for example, s5678.cloudfront.net (begins with "s").

Replace VIDEO-FILE-NAME with the name of your .mp4 or .flv video file,
including the .mp4 or .flv filename extension. For example, if you uploaded
my-vacation.mp4, enter my-vacation.mp4. You might need to prepend "mp4:" to the
name of your video file, for example, mp4:my-vacation.mp4.
-->

<div id='mediaplayer'></div>
<script type="text/javascript">
   jwplayer('mediaplayer').setup({
      file: "rtmp://RTMP-DISTRIBUTION-DOMAIN-NAME/cfx/st/VIDEO-FILE-NAME",
      width: "720",
      height: "480"
   });
</script>
```

Figure 2.5. Integration of S3 and CloudFront through web application (from [1])

As we are working the JWPlayer mediaplayer which is implemented for Adobe Flash, our web application only correctly renders the trailer videos in web browsers like Google Chrome and Internet Explorer, which still support the copyright implementation of RTMP. Using another implementation as video over HTTP or HLS would have enabled to use other browsers working with HTML5 for video rendering. The rendering of videos with HLS would have required the use of Amazon Elastic Transcoder, as explained in [2], to transcode the movie files into HLS format. However, figure 2.6 shows that HLS is only supported in the latest versions of some browsers and is not supported on the default browser of Android phones.

The problem with the streaming protocols today is that they are not regulated by third party organizations and that every company uses a different protocol. However, the aim of this project was to show the potential offered by Amazon AWS more than the conflict between the different copyright technologies that are running nowadays.

Advantages:

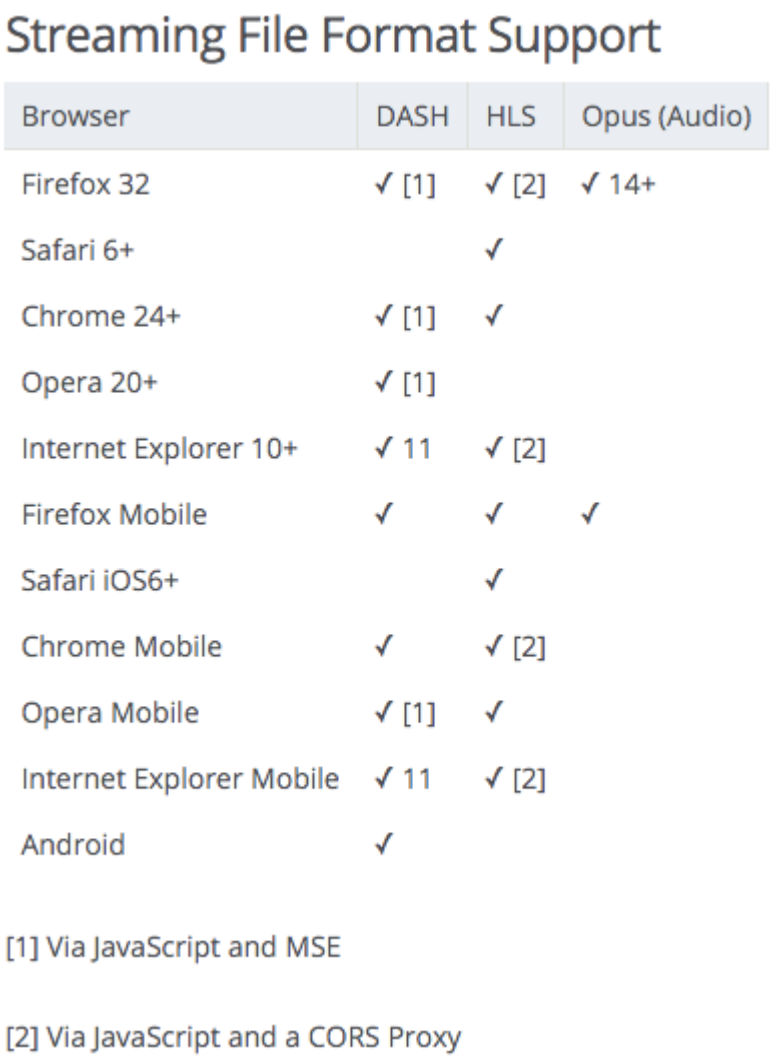
- The content is delivered dynamically, giving portions of the stream of the video following rules set during the configuration of the CloudFront distributions.

- CloudFront uses its own analytics center which allows to have a better view about the speed and performance of the video delivery. This enables for instance to compare between the speeds given by Amazon servers in different locations around the globe.

- As mentioned in the previous point, AWS has a server in each continent and the user can choose which server to use depending on its location.

Disadvantages

- More than a disadvantage of Cloudfront we found a problem with the regulations of the current protocols of streaming, as the standard HTML5 is still under development and the panorama is still chaotic when we speak about copyright management systems.



Streaming File Format Support

Browser	DASH	HLS	Opus (Audio)
Firefox 32	✓ [1]	✓ [2]	✓ 14+
Safari 6+		✓	
Chrome 24+	✓ [1]	✓	
Opera 20+	✓ [1]		
Internet Explorer 10+	✓ 11	✓ [2]	
Firefox Mobile	✓	✓	✓
Safari iOS6+		✓	
Chrome Mobile	✓	✓ [2]	
Opera Mobile	✓ [1]	✓	
Internet Explorer Mobile	✓ 11	✓ [2]	
Android	✓		

[1] Via JavaScript and MSE

[2] Via JavaScript and a CORS Proxy

Figure 2.6 Showing support for HLS Streaming in latest browsers

Amazon S3

Amazon S3 is a simple and cheap storage system on the cloud for scripts, videos, etc. The flow and the interaction with the rest of the console AWS is very important. Indeed, S3 is not only a shared hosting server or a cloud storage service (like Google Drive or Dropbox) but it is also a module integrated with the rest of the tools of AWS to make a complete solution. It disposes of a S3 API to easily access and upload files in S3 buckets, while using an object oriented approach. However, the API is not used in our application because CloudFront (as shown in figure 2.5) enables to directly access the files through their S3 filename.

Please note, as was said before, that the Movies and Trailers application stores the filename of the trailers in an AWS RDS database while the movie trailers themselves are stored in Amazon S3. This means that there is no direct relation in our

application between Amazon RDS and Amazon S3, i.e. we could have used any other relational database to store the trailer filenames.

Another comment we can add is that our application only retrieves data from S3 while putting data is done with the Amazon Management Console outside the Movies and Trailers app. However, it would have been possible for users to upload their own movie trailers on the website. In that case, the application would have manipulated the S3 API to upload the trailers in the S3 Bucket (for details of how to do this, see [8]). Nevertheless, we chose not to implement this option because that would have required to differentiate between logged in users and other users (to keep track of who posted what) and would have required some additional CakePHP programming outside the scope of this project.

Advantages:

- In Amazon S3 there are 3 different types of storage, either standard, infrequent access or "glacier" (very infrequent access). The two last solutions are of course cheaper than the standard storage. Besides, files can easily be moved from one type of storage to another.

- In the S3 bucket properties, it is possible to choose the option "requester pays" to request a payment from the user to watch videos.

- Data in S3 can be encrypted at the server-side.

Disadvantages

- With Amazon management console, uploading files in S3 takes a lot of time (around 25 kB/s). However, several files can be loaded at the same time, so the uploading can be done in the background. The reason of this slow speed could again be the use of a trial version of the software where the upload speed may be limited.

Elastic Beanstalk

Deploying our web application on the cloud was easily done with Elastic Beanstalk. We used CakePHP as a framework for PHP. The details on how to proceed are explained in [7]. We developed our website locally with Apache and once it was implemented, the application was easily deployed using the Elastic Beanstalk EB Command Line Interface (CLI) 3.x and Git.

Conclusion

When using the free trial version of AWS, we realized that we were not free to select all possible configurations of Amazon, i.e. when creating a database. However, we were able to develop our application using the default configuration. Furthermore, knowing and understanding all possible options offered by Amazon would require much time which we didn't have. Therefore, the free trial version we worked with was satisfactory for us and enabled us to learn the basic functioning of AWS.

We have learned from this project that a cloud database system like Amazon AWS has many advantages. The two most important ones are scalability as well as the integration of many different services in a same cloud system. The learning curve to use these systems is quite low, i.e. to use CloudFront was very easy relatively to other solutions. Another point is that such a system is very flexible. Indeed, Amazon AWS enable the customer to choose between different solutions : either the customer itself manages its resources (like calculating the number of CPU needed) or he asks Amazon AWS to do that for him. If the customer chooses the second solution, his costs towards Amazon increase a little but on the other hand his cost in IT infrastructure and human resources can decrease considerably since AWS does most of the job for him. To know if working with Amazon AWS can be advantageous for a company requires a full analysis of the cost gains and losses, which must be done case by case and depends on many different factors. Furthermore, we can add that the global use of such cloud systems can have a good impact on the Ecology because, as said in [4], "cloud computing is all about virtualization, multi-tenancy, and shared resources that provide more service for the amount of energy expended when compared to in-house, single tenant solutions". As a conclusion, the cloud solution is a promising field that is adopted more and more by companies around the world.

Bibliography

[1]http://docs.aws.amazon.com/AmazonCloudFront/latest/DeveloperGuide/TutorialStreamingJWPlayer.html
[2]http://www.jwplayer.com/blog/encoding-hls-with-amazon-elastic-transcoder/
[3]https://aws.amazon.com/developertools/Amazon-S3/771
[4]http://blog.caspio.com/paas-in-action/top-benefits-of-database-cloud-computing/
[5]https://aws.amazon.com/es/blogs/aws/using-amazon-cloudfront-for-video-streaming/
[6]http://www.slideshare.net/AmazonWebServices/dat203-28463220
[7]http://docs.aws.amazon.com/elasticbeanstalk/latest/dg/create_deploy_PHP_cakePHP.html
[8]http://docs.aws.amazon.com/AmazonS3/latest/dev/UploadObjSingleOpPHP.html