방송기술의 클라우드 전략

Part 5. 클라우드를 활용한 서비스 구축

- MBC 예능연구소의 AWS 클라우드 도입 사례를 중심으로

글.
김기완 AWS 솔루션즈 아키텍트

연재목록

지난 호까지 클라우드의 미디어 워크로드 활용에 대해서 살펴보았다. 클라우드가 제공하는 혜택은 빠른 속도, 탄력적인 자원 활용을 통한 비용 효율성 극대화, 현실적으로 활용가능한 무한대의 서버, 컴퓨팅, 네트워크 자원, 글로벌 인프라 등 다양하다. 이러한 장점을 고려한다면, UHD, VR/AR, DVR, PVR 등 새로운 미디어 서비스들에 적극적으로 클라우드를 활용하여, 경쟁력 높은 서비스를 빠른 속도로 전 세계에 서비스할 수 있을 것이다.

하지만, 클라우드를 이용해 서비스를 기획할 경우 다양한 고려 요소들이 있다. 이를 위해서 AWS는 전사적, 혹은 큰 규모로 클라우드를 도입하기 위한 전략을 제공하는 컨설팅 서비스를 제공하고 있다. 또한 단위 업무별 전략을 돕기 위해 솔루션즈 아키텍트들이 고객과 함께 일하고 있다. 이러한 컨설팅 서비스 및 솔루션즈 아키텍트들을 활용하여 업무별 클라우드 이전 효과를 예측해 볼 수 있으며, 또한 비용 효율적인 시스템 구성을 위한 조언을 얻을 수 있다. 이번 호에서는 MBC 예능연구소의 사례를 바탕으로, 클라우드를 사용한 서비스 구성 방법에 대해서 살펴보자.

MBC 예능연구소(ent.mbc.co.kr)는 다양한 형태로 제작된 미디어 파일을 소셜 미디어를 활용하여 시청자들에게 폭넓게 공유하는 서비스이다. 휴대폰이나 핸디캠, 그 외 다양한 영상 장치를 통해 촬영된 미디어 파일을 웹에서 스트리밍 형태로 제공하는 것이 서비스의 핵심이다. 이러한 서비스를 기획하게 되면 처음 마주하게 되는 문제는 사용자 수에 대한 예측과 그에 따른 서버, 스토리지, 네트워크 사용량 예측이다. 만일 이러한 서비스를 처음부터 클라우드에서 설계한다면, 미리 예측하여 자원을 준비한 후 서비스를 구성할 필요가 없고, 대신 확장성이 뛰어난 AWS의 완전 관리형 서비스에 기반해 서비스를 구성할 수 있어 효율적이다.

이러한 서비스의 특성을 고려하여 미디어 파일을 업로드하고, 이를 웹/모바일 배포용으로 트랜스코딩 하는 과정이 클라우드에서 어떻게 만들어질 수 있는지에 대해서 살펴보자.

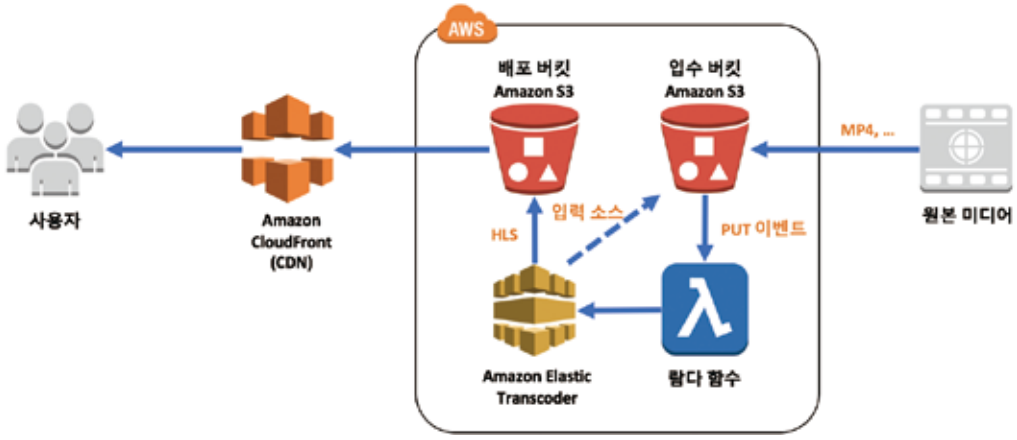

그림 1. 예능연구소의 미디어 트랜스코딩 작업

[그림 1]은 MBC 예능연구소의 실시간 미디어 트랜스코딩 서비스의 AWS 아키텍쳐이다. 제작된 소스 영상들은 먼저 Amazon S3(Simple Storage Service)라는 오브젝트 스토리지로 전송되어 저장된다. Amazon S3는 다음과 같은 특징을 갖는 스토리지 서비스이다.

- **높은 안정성 (99.999999999%)** | 여러 개의 물리적인 데이터센터에 데이터를 중복 저장
- **무제한 저장 기능** | 한 개의 오브젝트의 최대 크기 5TB, 개수 무제한
- **정적 웹 호스팅 기능** | 모든 오브젝트는 웹 URL을 가지고 웹 서비스 가능
- **라이프사이클 기능** | 가격 및 기능에 따라 세 개의 스토리지 티어로 나누어 자동으로 라이프사이클을 적용하여 비용 효율적인 스토리지 사용
- **버저닝 기능** | 오브젝트에 대한 안전한 보관 및 버전 관리
- **이벤트 기능** | 오브젝트가 생성되는 등 상태 변화가 생길 때 이벤트를 트리거. 이벤트를 통해서 Lambda 함수, Amazon SNS(Simple Notification Service), Amazon SQS(Simple Queue Service) 등과 연계해서 자동화된 워크플로우를 만들 수 있음
- **강력한 보안 기능** | 저장된 오브젝트들의 암호화, 전송 시 HTTPS 지원, 정해진 사용자/경로를 통해서만 오브젝트를 사용할 수 있는 접근 통제 기능 제공

앞의 기능에도 설명하였듯이, S3는 특정한 위치에 새로운 오브젝트가 생성되면 이벤트를 발생시킬 수 있다. 예능연구소의 경우 미디어 원본 파일의 업로드가 완료되는 순간 이벤트가 만들어지게 되는데, 이때 이 이벤트를 받아서 람다 함수가 구동될 수 있다.
람다 함수는, 물리적인 서버나 VM(가상 머신)을 할당하지 않고도 원하는 프로그램을 구동할 수 있는 서비스다. 대개 이벤트에 반응하여 구동되며, 이때 이벤트 소스로부터 필요한 이벤트 정보를 넘겨받는다. S3에서의 오브젝트 생성 이벤트에서는 생성된 오브젝트의 이름(Key) 및 메타데이터를 넘겨받게 된다

[그림 2]는 이러한 과정을 수행하는 샘플 파이썬 코드이다(물론, MBC에서 실제로 사용되는 코드는 아니다). 람다에서는 현재 파이썬, 자바스크립트, 자바, .NET을 사용할 수 있다. 람다 코드를 보면, lambda_handler()라는 함수를 볼 수 있으며, 이 함수는 람다 코드가 구동될 때 자동으로 실행된다. 이때 event와 context 두 개의 객체를 넘겨받게 되고, 이 중 event 객체를 통해서 이벤트로부터 전해지는 여러 파라미터들을 넘겨받을 수 있다. 그림에서 보이듯 event 객체로부터 입수된 원본 미디어 파일의 이름 및 위치(AWS 리전) 정보를 넘겨받고, 거기에 맞는 트랜스코딩 함수(start_ets_job)를 구동하고 있다.

김기완
AWS 솔루션즈 아키텍트

```python
import json
import boto3

pipeline_id = '1484456038659-osho6w'

def lambda_handler(event, context):

    print ("Processing lambda handler")

    try:
        if (event!=None and event.has_key('Records') and
            len(event.get('Records')) == 1 and
            event.get('Records')[0].has_key('s3') and
            event.get('Records')[0].get('s3').has_key('object') and
            event.get('Records')[0].get('s3').get('object').has_key('key')):

            # get S3 object key value
            event_object = event.get('Records')[0]
            # get AWS region
            s3_object = event.get('Records')[0].get('s3').get('object')

            input_key = s3_object.get('key')
            region = event_object.get('awsRegion')

            print ("Input media received : {0} from region {1}".format(input_key, region))
            print ("Starting ETS job to convert this media!")

            start_ets_job (region, input_key)

        else :
            return {'status' : 'ignored', 'message' : 'Invalid input'}
    except Exception as exception:
        return {'status' : 'error', 'message' : exception.message}
```

그림 2. 샘플 람다 코드 (S3로 입수된 파일의 트랜스코딩)

그림 3. 람다 코드 안에서 ETS 호출하기

다시 [그림 1]로 돌아가 보자. 원본 영상이 입수되고, 그때 이벤트가 발생하여 람다 함수가 호출된 이후 람다 함수 내부에서 트랜스코딩 작업을 시작하게 된다. 이때 Amazon Elastic Transcoder(ETS)를 사용하게 된다. ETS는 한 개의 입력 소스로부터 다양한 출력을 구성할 수 있도록 되어 있으며, SDK를 사용해 프로그램에서 사용할 수 있고, AWS 콘솔이나 API, CLI(Command Line Interface)로도 활용이 가능하도록 되어 있다.

실제로 ETS를 호출하는 코드는 람다 함수 안에 포함될 수 있다. 이때 경우 미리 ETS에서 만들어 둔 미디어 출력 템플릿들을 정의하고, 한 개의 입력 소스로부터 다양한 출력을 만드는 ETS 작업을 호출할 수 있다.

ETS가 작업을 완료하게 되면, 출력 파일은 다시 S3에 저장된다. [그림 1]에서는 출력 파일이 생성된 후 별다른 작업이 업데이트되어 있지 않지만, 필요하다면 출력 파일들이 생성될 때 발생한 이벤트를 통하여 CMS에 출력 결과를 보고하거나, 출력 파일에서 메타데이터를 추출하여 메타데이터 저장소를 업데이트하는 등의 작업을 할 수 있다.

이제 원하는 출력 파일들이 S3 상에 존재하므로 직접 웹에서 S3에 저장되어 있는 *.m3u8을 호출함으로써 HLS 스트리밍이 시작될 수 있다. HTML 5 기반의 플레이어에서 재생하는 경우, CORS (Cross Origin Resource Sharing) 설정에 주의해 주어야 하며, S3의 경우 해당 버킷 별로 CORS Policy를 입력할 수 있도록 하고 있다.
물론 S3는 스스로 확장성을 가지고 있다. 즉, 실제 사용자가 직접 네트워크 및 서비스에 필요한 서버 및 스토리지에 대한 확장성을 고려할 필요는 없다. 그렇지만 AWS에서 손쉽게 시작할 수 있는 Amazon CloudFront를 통해서 손쉽게 글로벌 사용자들에게 높은 성능으로 미디어 스트리밍을 할 수 있다.

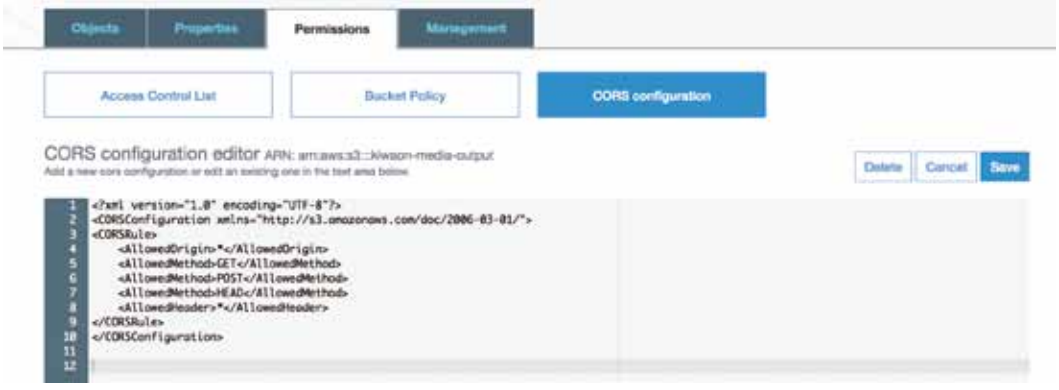

그림 4. S3 버킷의 CORS 설정

Amazon CloudFront는 글로벌 CDN 서비스로, 다음의 기능을 제공한다.

- 전 세계 73개 엣지 로케이션(Edge Location)에서 이미지, 미디어 파일, 웹 페이지 등 다양한 정적 파일을 캐싱하여 사용자의 위치에서 가장 응답시간이 빠른 지역에서 웹 서비스 제공
- 최적화된 리전 간 라우팅 및 Keepalive HTTP 세션에 대한 지원을 통하여 데이터베이스 및 백엔드 서비스를 사용하는 동적 콘텐츠에 대한 성능 향상
- Signed URL, Signed Cookie 등을 활용한 안전한 접근 제어
- AWS WAF(웹 방화벽) 및 AWS Shield(DDoS 대응 서비스)를 통한 웹 서비스 보호
- Lambda@Edge를 통하여 엣지에서의 컴퓨팅 사용
- AWS 콘솔 및 API/SDK를 사용하여 쉽게 사용할 수 있는 셀프 서비스 글로벌 CDN
- 웹 기반 라이브 스트리밍(HLS, DASH 등) 지원 및 RTMP VoD 지원

S3에 있는 미디어 배포본(HLS의 경우 .m3u8 및 ts 파일들)을 원본(Origin)으로 하여 Amazon CloudFront 글로벌 CDN을 활용하게 되면, 전 세계에 있는 사용자들이 자신과 가까운 곳에 있는 엣지로부터 해당 미디어 파일들을 다운로드하여 재생할 수 있게 된다. 따라서, 손쉽게 글로벌 미디어 스트리밍 서비스를 구성할 수 있다.

김기완
AWS 솔루션즈 아키텍트

이제 미디어 스트리밍 서비스(VoD)가 완료되었다. 물론 올바른 서비스를 위해서는 웹/모바일 백엔드 서비스 구현 및 CMS/메타데이터 처리 부분의 서비스 구현이 필요하다. 물론, 이러한 서비스 구현에서도 정적인 콘텐츠의 경우(이미지, 동영상, 정적 웹페이지) S3 및 CloudFront를 적극적으로 활용하여 사용되는 서버의 크기를 줄일 수 있다.

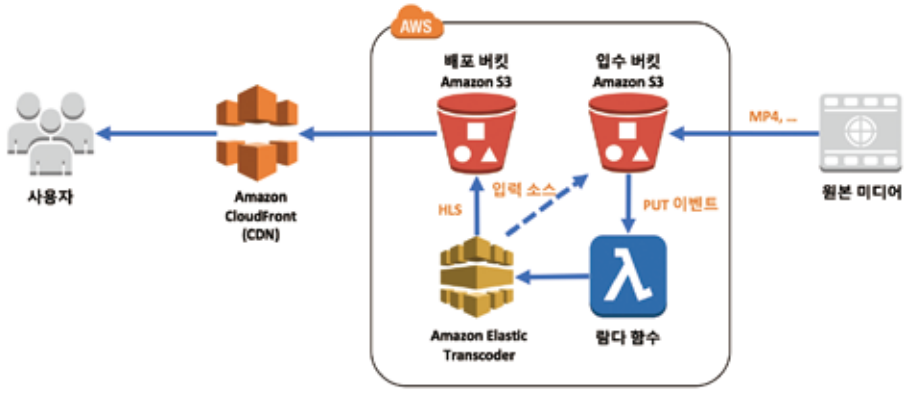

그림 5. 서버리스 아키텍처

[그림 5]는 [그림 1]과 동일하다. 위의 그림을 유심히 보면, 단 한 대의 서버도 사용하지 않는 것을 알 수 있다. S3, Lambda, ETS, CloudFront 모두 AWS의 완전 관리형 서비스들이다. 즉 사용자 입장에서는 서버를 관리할 필요가 없다는 것이다. 이러한 아키텍처를 서버리스 아키텍처라고 하며, 서버리스 아키텍처는 다음과 같은 장점을 갖는다.

- 보안적으로 우수하며, 관리 요소가 줄어든다. 서버 관리자는 더 이상 서버의 보안 취약성 및 패치를 신경쓸 필요가 없으며, 서버에 대한 접근제어, 로그분석 등을 할 필요가 없다.
- 탄력적이다. 사용량이 적거나 없는 경우 유지하는 자원이 없기 때문에 비용을 지불할 필요가 없다. 반대로 사용량이 급격하게 늘어나는 경우, 완전 관리형 서비스는 내재된 확장성을 바탕으로 손쉽게 늘어나는 수요에 대응할 수 있도록 해 준다.
- 관리 요소가 줄어들기 때문에 더 많은 인적/물적 자원을 서비스 개발에 쏟을 수 있도록 해 준다.
- 서버 및 애플리케이션에 대한 설치, 성능 점검, 기능 테스트 등의 과정을 거치지 않기 때문에 서비스를 매우 빠르게 개발하고 개선할 수 있게 해 준다.

이러한 서비스를 하드웨어 중심으로 설계한다고 생각해 보자. 스토리지를 관리하는 서버가 필요하고, 인코더 장비, 해외 서비스를 위한 자원들도 필요하다. 이러한 모든 장비들은 적절히 관리, 설정되어야 하며, 이러한 작업을 위한 전문 인력들도 배치해야 한다.
반면 AWS에 능숙한 사용자의 경우, 이와 같은 미디어 서비스를 구성하는 데 수 시간이면 충분하다. AWS는 또한 빠르게 고객들이 이와 같은 서비스를 구축할 수 있도록, 샘플 람다 함수들을 제공한다. github.com/awslabs을 통해 AWS가 제공하고 있는 다양한 서비스를 쉽게 이용할 수 있도록 샘플 환경들을 제공하고 있다.

이상으로 실제 AWS 클라우드를 도입해서 성공적으로 활용하고 있는, MBC 예능연구소의 사례를 바탕으로, 미디어 산업에서 클라우드를 사용한 서비스 구성 방법과 관련 혜택을 살펴보았다.

다음 호에서는 미디어 산업뿐만 아니라, 다양한 기업들의 관심을 받는 인공지능에 대해서 알아보기로 하자. 지난해 국내외적으로 주목을 받은 알파고의 성공에 힘입어 많은 인공지능 기반의 서비스들이 발표되고 있다. AWS의 사례를 중심으로 어떻게 인공지능 서비스를 만들 수 있는지에 대해 알아보기로 하자.