02

EBS 기술인협회 스터디

# 『넷플릭스 기술 연구회』 6부

EBS 기술인협회 스터디 『넷플릭스 기술 연구회』

EBS 기술인협회 기고가 벌써 마지막 회차로 다가왔습니다. 그동안 소개되었던 다양한 토픽들을 종합해 보면 넷플릭스는 의사결정, 콘텐츠 제작, 사용자 서비스 개선, IT 기술 증진 및 오픈소스 활성화와 같이 정말 방송기술의 모든 부분에 대해 일찍부터 고민해 오고 발전시켜 왔다는 것을 확인할 수 있었습니다. 이번 회차에서는 'Netflix에서 컨테이너 사용의 진화', 'HDR을 사용한 넷플릭스 UI 경험 강화' 두 가지 토픽을 다뤄본 후에 넷플릭스 기술 연구회를 진행하면서 느낀 시사점에 대해 말씀드리고자 합니다.

## Netflix에서 컨테이너 사용의 진화

### 원본 정보

| | |
|---|---|
| 제목 | The Evolution of Container Usage at Netflix<br>(넷플릭스에서 컨테이너 사용의 진화) |
| URL | https://netflixtechblog.com/the-evolution-of-container-usage-at-netflix-3abfc096781b |

그림 1. 원문 QR-Code

글로벌 가입자 수가 2억 3,100만 명에 육박하는 넷플릭스는 현재 '클라우드 네이티브' 환경을 기반으로 서비스를 개발하고 배포하고 있습니다. '클라우드 네이티브'란, 클라우드의 장점을 최대한 활용하여 정보 시스템을 구축 및 실행하는 환경으로 정의하고 있습니다. 넷플릭스도 처음 시작은 '클라우드 네이티브' 기반이 아니었습니다. 그들만의 데이터 센터를 가지고 있었으며 물리 서버를 중심으로 서비스가 개발되고 배포되었습니다.

그러나 2008년도에 데이터베이스 손상으로 인해 3일간 넷플릭스 서비스가 지연되는 상황이 발생하게 되면서 넷플릭스는 안정적이고 수평 확장이 가능한 분산 시스템으로 전환이 필요하다는 것을 인지합니다. 그리고 7년이라는 시간에 걸쳐서 클라우드 마이그레이션 작업을 완료하고, 넷플릭스 서비스는 데이터센터에서 아마존 AWS로 완전히 이전하는 데 성공했습니다. 또한 MSA(Microservices Architecture) 구조로 넷플릭스 서비스 개발 구조를 바꾸게 됩니다.

앞에서 언급한 대로 '클라우드 네이티브' 환경의 큰 특징 중 하나는 'MSA(Microservices Architecture)' 구조로 서비스를 개발하고 배포한다는 것입니다. MSA는 특정 비즈니스 로직에만 집중하여 개발하고 배포하는 개념입니다. 전체 애플리케이션을 한 덩어리로 개발하고 배포하던 기존 방식과 다르게 각 서비스마다 개발, 패키징, 빌드 및 배포 스케줄을 유연하게 가져갈 수 있고 요구사항이 발생하거나 수정사항이 생길 경우 바로바로 서비스에 적용할 수 있는 것입니다. 하단의 그림을 보면 기존 개발방식인 '모놀리식 아키텍처'와 '마이크로서비스 아키텍처'의 차이를 확인하실 수 있습니다.

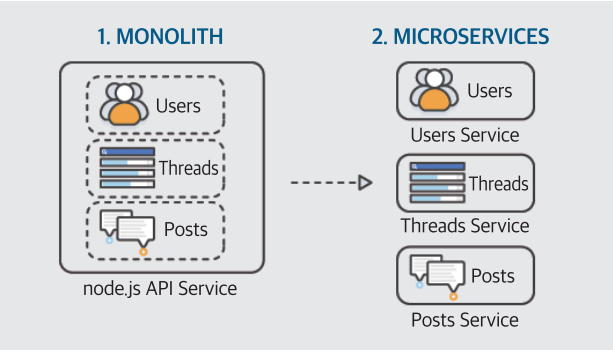


그림 2. Monolith 모델과 Microservices 모델 / 출처 : 아마존 AWS

넷플릭스는 2008년부터 2016년까지 이용자가 8배 이상 증가하는 상황에도 MSA 환경 구축을 기반으로 고객들에게 안정적인 동영상 서비스를 제공할 수 있었습니다. 또한 넷플릭스가 경험한 MSA 전환 기술을 Netflix OSS(Open Source Software)를 통해 오픈소스로 공개하면서 노하우를 전 세계에 공유하고 있습니다.

이제 MSA 구조의 중심이라고 할 수 있는 컨테이너에 대해 조금 더 살펴보도록 하겠습니다. 컨테이너가 등장한 가장 큰 배경은 애플리케이션 이식성 이슈라고 할 수 있습니다. 아무리 개발환경에서 좋은 서비스를 개발했다고 하더라도 이 서비스를 실제 운영환경에 배포하려고 하면 온갖 오류에 직면하는 문제점이 발생합니다. 오류의 가장 큰 이유는 각 환경에서 사용하는 운영체제들이 모두 다르고, 그에 따라 기술정책 및 명령어도 달라지기 때문입니다. 이를 해결하기 위해서 고안된 기술이 컨테이너 기술입니다. 즉 어떠한 컴퓨팅 환경이라도 배포된 SW 서비스를 오류 없이 실행할 수 있도록 해주는 기술이라고 생각하시면 됩니다.

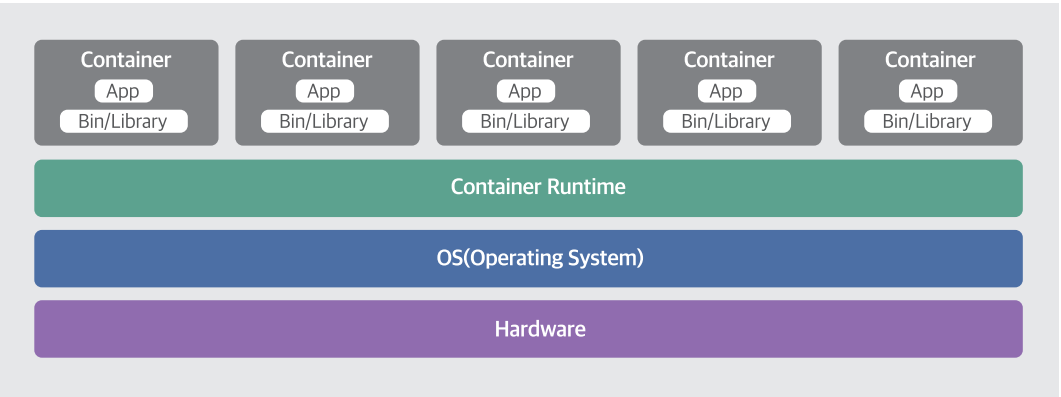


그림 3. 컨테이너 기본 구조

위 그림이 컨테이너 기반의 기본 구조입니다. 컨테이너를 구동하는 환경을 제공해 주는 Container Runtime(Ex. Docker) 위에서 각각의 컨테이너들이 독립적인 서비스를 구동하는 방식입니다. MSA 기반인 넷플릭스 또한 수많은 컨테이너들이 동시에 구동되면서 거대한 넷플릭스 플랫폼을 구성한다고 보시면 됩니다.

넷플릭스는 아마존 AWS의 클라우드 컴퓨팅 서비스인 AWS EC2(Amazon Elastic Compute Cloud) 가상머신을 기반으로 컨테이너를 운영하고 있습니다. 컨테이너를 생성할 때 이미지라는 것을 사용하게 되는데, 이미지는 소스코드, 라이브러리, 종속성 및 응용 프로그램을 실행하는 데 필요한 것들을 포함하는 불변 파일입니다. AWS는 AMI(Amazon Machine Image)를 제공해 주기 때문에 컨테이너 구동 환경에 적합한 인스턴스* AMI를 EC2에 적용하기만 하면 쉽게 컨테이너 운영 환경을 구축할 수 있습니다. 넷플릭스 또한 강력한 AWS 서비스를 적극 활용해서 자신들의 MSA 서비스를 구동하고 있습니다.

※ **인스턴스** : EC2에서 구성할 수 있는 컴퓨팅 스펙(Ex. CPU, RAM, 그래픽카드 등)

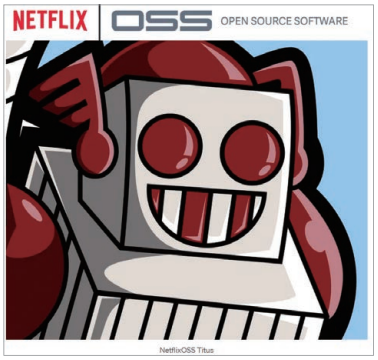

그림 4. Titus 로고

MSA를 위해 사용되는 컨테이너가 한두 개가 아니라 엄청난 규모로 사용된다면 매번 직접 유지보수하고 이슈 사항에 대처하는 것은 굉장히 어려운 일입니다. 그렇기 때문에 수많은 컨테이너들의 배포 및 관리, 확장, 네트워킹을 자동화하여 관장하는 솔루션을 '컨테이너 오케스트레이션'이라고 부릅니다. 오케스트레이션의 경우 쿠버네티스와 같은 유명한 솔루션들이 있지만 넷플릭스는 Titus라고 하는 솔루션을 개발해서 사용하고 있습니다. Titus는 완전히 새로 개발된 솔루션이 아니라 Apache Mesos라는 오픈소스를 기반으로 개발된 넷플릭스 OSS(Open Source Software)이며, 기사에 따르면 Titus는 대략 1주일에 거의 300만 개 이상의 컨테이너를 띄우고 관리하고 있습니다.

넷플릭스에서 설명하는 Titus의 구조가 복잡해서 조금 간단하게 [그림 6]으로 설명드리면 Titus Scheduler가 전체 시스템을 관장하는 제어판의 역할을 수행한다고 보면 됩니다. Titus Scheduler에서 대표적으로 수행하는 기능들은 다음과 같습니다.

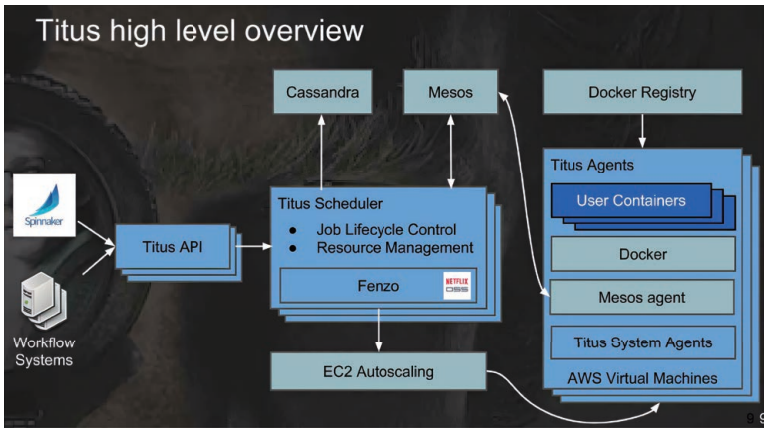

그림 5. Titus high level Overview / 출처 : Netflix

1) 컨테이너가 죽게 되면 재빠르게 정상 컨테이너로 대체하는 컨테이너 모니터링
2) 서비스 사용자 수의 증감에 따라 컨테이너의 사용량을 조절해서 리소스를 관리하는 오토 스케일링
3) 서비스를 중단시키지 않고 업데이트를 수행하는 롤링 업데이트

제어판의 통제를 토대로 하위에 있는 Titus Agent 내부에서 작업을 수행하는 구조로 이해하시면 될 것 같습니다. Agents는 전에 언급드렸던 AWS EC2 가상머신 기반으로 동작하며 Agents 내부에 Docker와 컨테이너가 작동하고 있는 구조입니다. 쿠버네티스를 공부해 보셨거나 조금 다뤄보셨던 분들은 거의 동일한 구조라는 점을 바로 알 수 있을 것입니다.


그림 6. Titus의 간단한 구조

넷플릭스는 Titus를 이용해서 수많은 컨테이너를 효율적으로 관리하고 있고, 안정적인 OTT 서비스를 사용자들에게 제공하고 있습니다. MSA의 개념을 실제 업무에 적용하는 일이 정말 만만치 않은 어려운 작업인데 오랜 시간 동안 준비하고 안정적인 MSA를 구축한 넷플릭스의 기술 노하우를 조금이나마 들여다볼 수 있었습니다.

## HDR을 사용한 넷플릭스 UI 경험 강화

### 원본 정보

| | |
|---|---|
| 제목 | Enhancing the Netflix UI Experience with HDR<br>(HDR을 사용한 넷플릭스 UI 경험 강화) |
| URL | https://medium.com/netflix-techblog/enhancing-the-netflix-ui-experience-with-hdr-1e7506ad3e8 |

그림 7. 원문 QR-Code

이번 포스팅에서는 HDR과 색 영역, 그리고 넷플릭스에서 HDR을 어떤 방식으로 구별해 사용하는지 간략하게 말씀드리고자 합니다. 우리가 현재 흔히 접할 수 있는 기술은 SDR(Standard Dynamic Range)과 HDR(High Dynamic Range)입니다. 우선 Dynamic Range란 특정 이미지에서 가장 밝은 부분과 가장 어두운 부분의 차이를 말합니다. SDR과 HDR은 아래 사진으로 쉽게 비교해 보실 수 있습니다.

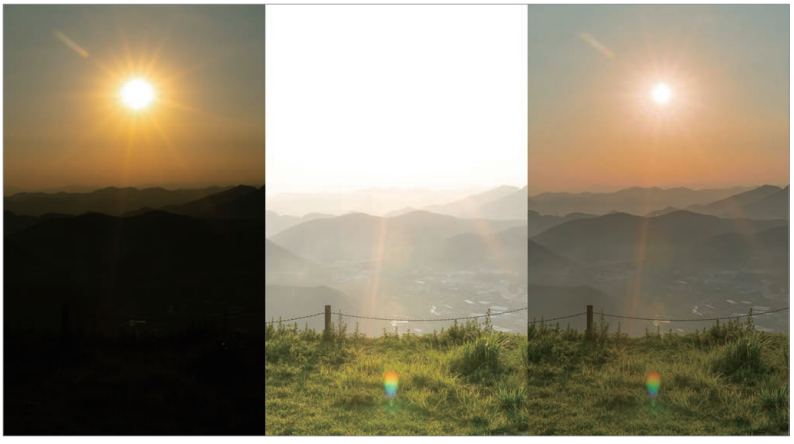

그림 8. SDR, HDR 비교

첫 번째 사진은 SDR을 사용해 하늘을 기준으로 촬영, 두 번째 사진은 SDR을 사용해 땅을 기준으로 촬영, 마지막 사진은 HDR을 사용해 촬영한 사진입니다. 첫 번째 사진을 보시면 노을 부분은 잘 표현되었으나 땅 부분이 너무 어두워 구별하기 힘들고 두 번째 사진은 반대로 땅 부분이 잘 표현되었으나 하늘 부분은 구별이 힘든 것을 볼 수 있습니다. 하지만 실제로 우리가 풍경을 볼 때는 밝은 부분과 어두운 부분 다 인식할 수 있으므로 가장 우리의 눈에 가깝게 촬영하기 위한 기술을 HDR이라고 생각하면 쉽게 이해할 수 있을 것 같습니다.

다음으로는 색 영역에 대해 말씀드리겠습니다.

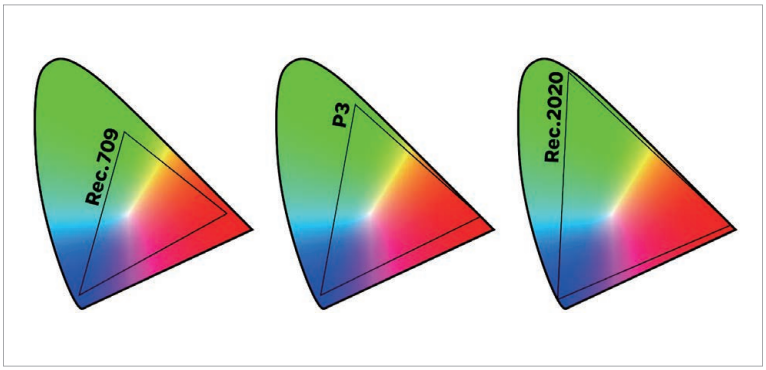


그림 9. 색 영역 비교

좌측의 3가지 사진에서 삼각형으로 표시된 부분이 디스플레이가 재현 가능한 색상의 범위를 수치로 나타낸 Rec.709, DCI-P3, Rec.2020이라는 각각의 색 영역입니다. 실제 사용하고 있는 색 영역은 더 많지만 방송에서 볼 수 있는 대표적인 색 영역 3가지를 간략하게 설명하겠습니다.

먼저 색 영역을 정의한 단체에서 확인해 보면 Rec.709, 2020 색 영역은 ITU-R(국제전기 통신연합)에서 발표한 영역으로 방송, TV 쪽에서 사용하고 있는 영역입니다. Rec.709는 보통 HD, Rec.2020은 UHD에서 사용합니다.

다음으로 DCI-P3는 DCI(Digital Cinema Initiatives)의 약자로 할리우드의 영화 제작 스튜디오들이 참여해 발표한 색 영역으로 현재 디지털 영화 또는 갤럭시나 애플 등의 모바일 장치에서 많이 사용하고 있는 영역입니다. 흔히 SDR에서는 Rec.709, HDR에서는 P3 또는 Rec.2020 색 공간을 사용합니다.

이제 넷플릭스에서 어떻게 HDR을 사용해 UI 강화를 연구하는지 알아보겠습니다.

**원문**

We would love at this point to be able to show you an actual HDR image on your screen to show you exactly what brighter, more saturated colors look like, but there isn't any current technology that lets us do that (which is the basis of this whole problem).

One way to imagine this is to picture a single frame of video from our app that contains pixels outside of what its SDR equivalent source can express.

**To visualize these differences in expressibility, our team first started off by implementing a debug feature within the Netflix app that is able to highlight HDR pixels in real-time on the screen.**

**번역**

지금 이 시점에서 실제 HDR 이미지를 화면에 보여드려 더 밝고 넓은 색영역을 사용하는 색상이 어떻게 보이는지 정확히 보여드리고 싶지만, 현재 그런 기술이 없습니다.(이것이 전체 문제의 근본입니다).

이를 확인하는 한 가지 방법은, SDR로 표현할 수 있는 범위를 벗어난 픽셀이 포함된 소스의 비디오 프레임을 보는 것입니다.

**표현 가능성의 차이를 시각화하기 위해, 우리 팀은 먼저 Netflix 앱 내에서 실시간으로 화면에 HDR 픽셀을 하이라이트를 표시할 수 있는 디버그 기능을 구현했습니다.**


그림 10. 본문 사진

**원문**

The above are screenshots with the HDR highlighter feature enabled during video playback of our HDR-packed Netflix Original — Altered Carbon.

**We grayscale the entire screen, then highlight each "HDR pixel" with various colors, typically with stronger colors for pixels that we consider "more HDR" (or farther away from what's possible with SDR).**

**We highlight pixels based on the following conditions:**

The pixel is out of SDR color space.
…

**번역**

위의 스크린샷은 Altered Carbon이라는 Netflix 오리지널 작품에서 HDR 포함된 동영상 재생 중에 HDR 하이라이트 기능이 활성화된 상태입니다.

**우리는 전체 화면을 그레이스케일로 만들고, 각 "HDR 픽셀"을 다양한 색상으로 하이라이트 표시합니다. 일반적으로 "더 HDR"(또는 SDR의 색영역과 밝기 범위를 벗어난 픽셀)로 간주되는 픽셀에는 강한 색상을 사용합니다.**

**우리는 다음 조건에 따라 픽셀을 강조 표시합니다:**

해당 픽셀이 SDR 색상 공간을 벗어납니다.
…

본문 내용 및 [그림 10]을 확인해 보시면 넷플릭스에서는 화면을 그레이 스케일로 변환한 후 SDR이 표현할 수 없는 범위, 즉 Rec.709 색 영역과 밝기 범위를 벗어나는 픽셀을 크게 3가지로 구분했습니다. 청록색은 Rec.709 표준을 벗어나는 색상, 분홍색은 SDR 표준을 벗어나는 밝기, 마지막으로 노란색은 Rec.709 표준을 넘어가는 색상과 SDR 표준을 넘어가는 밝기를 동시에 가진 픽셀입니다.
이렇게 HDR로 인지할 수 있는 색상 구분을 한 후에, UI에 HDR 이미지를 활용하기 위한 적정 포맷 요구사항을 넷플릭스 자체적으로 연구하기 시작했습니다. HDR 이미지 사용을 위해서 1. Color profile, 2. Higher bit-depth, 3. Codec support, 4. Tooling support 이렇게 4가지 요구사항을 검토했다고 합니다. 요구사항을 토대로 HDR 이미지를 조건에 맞는 적정한 포맷으로 정의한 후 압축된 이미지 파일에 ICC profile을 생성할 수 있는 툴을 직접 개발했다고 합니다.

※ **ICC profile** : 색 입력 장치나 색 출력 장치의 특성을 구현하는 데이터의 집합으로 국제 컬러 협회(ICC)가 공표한 표준을 따른다. 프로파일들은 특정한 장치의 색 특성을 정의하거나 색 공간, PCS(프로파일 연결 공간)의 매핑 정의에 필요한 요구 사항을 보여준다.

| Format | Max Depth | Lossy / Lossless | Has Alpha | Encoded File Size | HDR Picture Quality | Decode Speed | Implementation Support |
|---|---|---|---|---|---|---|---|
| PNG | 16-bit | Lossless | Yes | Very Large | Perfect (Lossless) | Fast | Excellent |
| JPEG | 8-bit | Lossy | No | Small | Poor (8-bit) | Fast | Excellent |
| WebP | 8-bit | Both | Yes | Very Small | Poor (8-bit) | Medium | Very Good |
| JPEG2000 | 32-bit | Both | Yes | Small | Excellent | Slow | Good |
| JPEG-XR | 32-bit | Both | Yes | Very Small | Good | Slow | Poor |
| HEIF | 16-bit | Both | Yes | Very Small | Excellent | Slow | Poor |

**표 1. Format table / 출처 : Netflix tech blog**

툴을 활용해서 이미지를 처리할 때 Source image 포맷(HDR 이미지를 포토샵에서 Export 한 이미지)은 PNG, Compressed delivery image 포맷(실제 디바이스에서 디코딩 및 렌더링하기 위해 다운로드하는 이미지)은 JPEG2000을 채택하여 HDR 이미지를 처리했다고 밝혔습니다.
해당 토픽은 HDR 이미지를 UI에 완전하게 표현하기 위한 초창기의 고민을 담았기 때문에 지금은 위에 언급한 포맷이 아니라 다른 포맷(Ex. HEIF)을 사용하여 더욱 효율적으로 HDR 이미지를 다루고 있을 수도 있습니다.


그림 11. 넷플릭스 오리지널 <IRON FIST> 화면. HDR 이미지가 적용된 UI

넷플릭스 오리지널 <IRON FIST> 서비스 화면을 보시면 처음에 언급했던 HDR 하이라이트의 기능을 이용해서 자신들의 UI에 HDR 이미지로 서비스를 제공하고 있다는 증거를 보여주고 있습니다. HDR 이미지를 처리하고 ICC profile을 정의하는 넷플릭스 솔루션 툴에 대한 내용을 더 확인하고 싶었지만 토픽에는 더 언급되어 있지 않았습니다. 하지만 HDR 이미지를 서비스하는 과정에 정말 많은 시행착오가 있었으며 방법을 찾아 나가는 넷플릭스 기술진들의 노력을 엿볼 수 있는 토픽이었습니다.

## 연재를 마치며

EBS 기술인협회 넷플릭스 기술 연구회에서는 약 10개월이라는 기간 동안 넷플릭스 기술 블로그의 토픽들을 번역하고, 더 나아가서 우리 방송기술 업무에 접목하는 방법을 생각해 보는 시간을 가졌습니다. 스터디를 진행하면서 넷플릭스의 기술 노하우 분석을 통해 시야를 넓힌 부분도 있지만, 기술을 끊임없이 발전시킬 수 있도록 회사의 방향성을 설립하고 업무 환경을 구축한 넷플릭스의 시스템에 대해 더 큰 시사점을 느낀 것 같습니다. 현재 상황에 만족하고 더 나아가지 않았다면 지금의 넷플릭스 또한 없었을 것입니다. 생각은 누구나 할 수 있지만 정말 적극적으로 행동에 옮겨서 기술을 발전시켜 나가는 것은 아무나 실천할 수 없다고 생각합니다. 넷플릭스는 행동으로 보여주었고, DVD 대여 회사에서 전 세계적인 OTT 서비스 회사로 거듭날 수 있었습니다. 물론 넷플릭스와 우리 공사는 분명히 다른 회사이지만 기술에 대한 태도와 지향점을 벤치마킹하는 것이 필요하다고 생각합니다. 끊임없이 발전하는 방송기술에 뒤처지지 않도록 노력이 필요하다는 것을 느끼며 이번 연재를 마치도록 하겠습니다. 6개월에 걸쳐서 긴 글을 읽어주신 독자분들에게 감사의 말씀을 드립니다.
마지막으로, 넷플릭스 기술 블로그에는 지금까지 연재한 토픽 이외에도 수많은 토픽이 많이 있으니 더 많은 자료를 보고 싶으신 분들은 넷플릭스 기술 블로그 홈페이지를 방문해 보시면 많은 도움이 될 것이라 믿습니다.

URL https://netflixtechblog.com