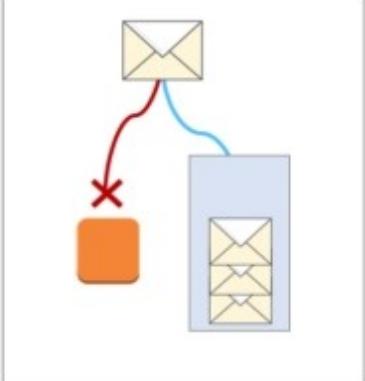


Amazon SQS 기능

aws training and certification

배달 못한 편지 대기열 지원



The diagram illustrates the concept of a Dead Letter Queue (DLQ). A blue rectangular box represents an Amazon SQS queue. An orange square icon with a red 'X' symbol is positioned below it, indicating a failed message. A red curved arrow points from the failed message to the queue, representing the message being rejected and sent to the DLQ.

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

배달 못한 편지 대기열(DLQ)은 처리되지 못한 메시지 대기열입니다. DLQ는 최대 처리 시도 수가 도달한 후에 메시지를 수신합니다. DLQ는 다른 Amazon SQS 대기열과 같습니다. 다른 SQS 대기열과 마찬가지로 DLQ로 메시지를 보내고 받을 수 있습니다. SQS API 및 SQS 콘솔에서 DLQ를 만들 수 있습니다.

Amazon SQS 기능

aws training and certification

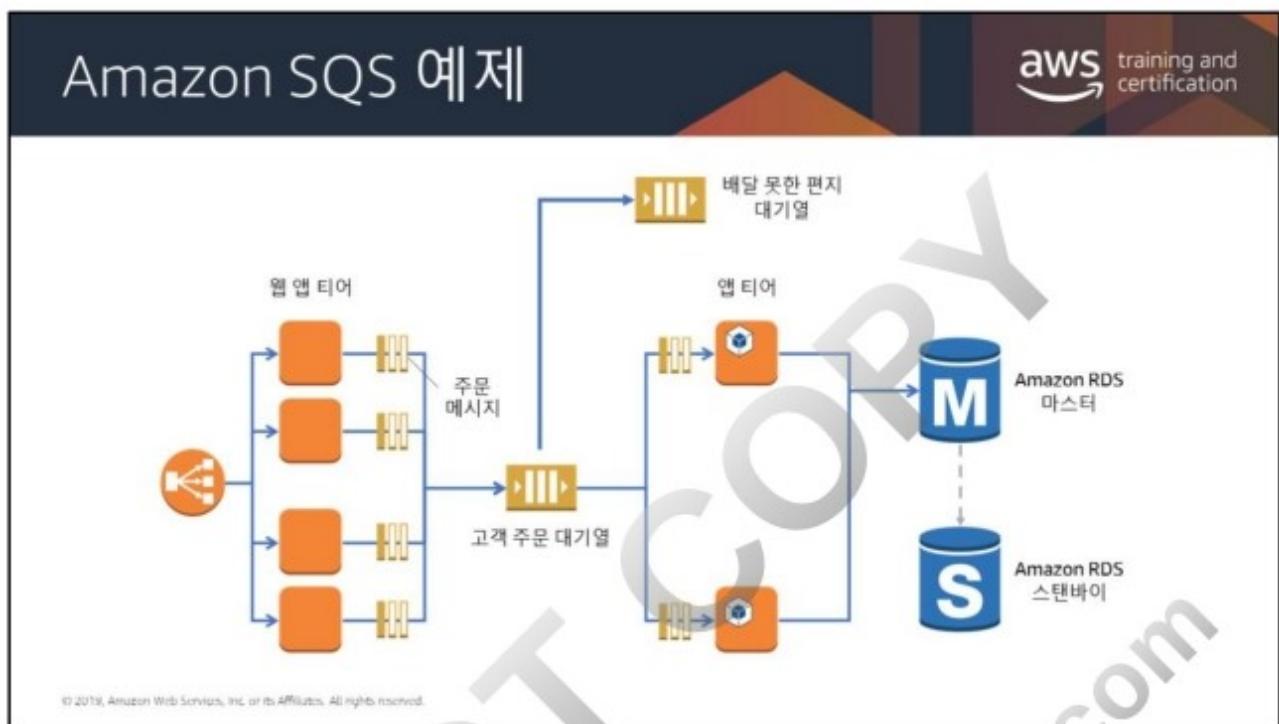
배달 못한 편지 대기열 지원 가시성 제한 시간

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

가시성 제한 시간이란 애플리케이션 구성 요소가 대기열에서 메시지를 가져온 후 해당 메시지가 다른 애플리케이션에는 보이지 않는 시간 간격을 가리킵니다. 메시지를 수신한 구성 요소는 일반적으로 이 가시성 제한 시간 동안 메시지를 처리한 다음, 이를 대기열에서 삭제합니다. 가시성 제한 시간은 여러 구성 요소가 같은 메시지를 처리하는 것을 방지합니다. 애플리케이션이 처리하는 데 시간이 더 필요한 경우, "보이지 않는" 제한 시간을 수정할 수 있습니다.



긴 폴링은 Amazon SQS 대기열에서 메시지를 검색하는 방법입니다. 짧은 폴링의 기본값은 폴링 중인 메시지 대기열이 비어 있더라도 즉시 반환됩니다. 다만 긴 폴링은 메시지가 메시지 대기열에 도달하거나 긴 폴링 제한 시간이 초과할 때까지 응답을 반환하지 않습니다. 긴 폴링을 사용하면 Amazon SQS 대기열에서 메시지가 제공되는 즉시 저렴한 방법으로 메시지를 검색할 수 있습니다.



SQS 대기열을 도입하여 주문 애플리케이션을 개선하는 방법을 알아봅니다. 대기열을 사용하면 처리 로직을 자체 구성 요소로 분리된 후, 웹 애플리케이션과 별도로 구분된 프로세스에서 실행할 수 있습니다. 이를 통해 시스템은 트래픽 급증에 보다 탄력적으로 대처할 수 있으며 비용 관리를 위해 필요한 만큼 신속하게 작업을 수행할 수 있습니다. 또한 주문을 메시지로 유지(임시 데이터베이스로 작동하는 대기열을 가지고)하기 위한 메커니즘을 갖추게 되었으며, 데이터베이스와의 트랜잭션의 범위를 스택 아래로 이동할 수 있습니다. 애플리케이션 예외 또는 트랜잭션 장애가 발생하면 SQS 대기열은 주문 처리를 중단하거나 Amazon SQS DLQ (Dead Letter Queue)로 리디렉션하여 나중에 다시 처리할 수 있습니다.

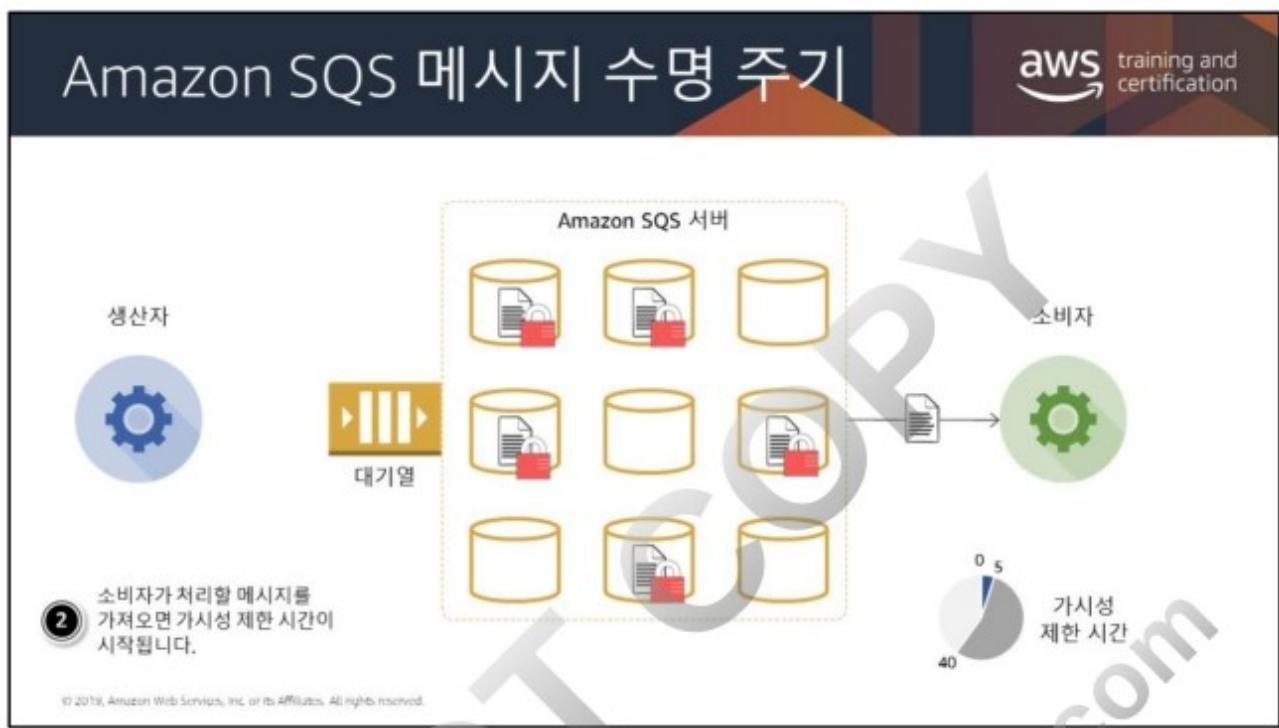
이 사용 사례에 관한 자세한 내용은

<https://aws.amazon.com/blogs/compute/building-loosely-coupled-scalable-c-applications-with-amazon-sqs-and-amazon-sns/>를 참조하십시오.



Amazon SQS (Amazon Simple Queue Service)는 웹 애플리케이션의 한 구성 요소가 생성하여 또 다른 구성 요소가 사용할 메시지를 웹 애플리케이션이 대기열에 넣을 수 있도록 허용하는 분산 대기열 시스템입니다. 대기열은 처리 대기 중인 메시지들의 임시 리포지토리이며 1~14 일간 메시지를 보관합니다(기본 설정된 보관 기간은 4일). Amazon SQS를 사용하면 애플리케이션의 구성 요소들을 분리하여 독립적으로 실행할 수 있습니다. 메시지는 형식에 관계없이 최대 256KB의 텍스트로 작성할 수 있습니다. Amazon SQS는 여러 생산자와 소비자가 같은 대기열에서 상호 작용하도록 지원합니다. Amazon SQS는 Amazon EC2, Amazon S3, Amazon ECS, AWS Lambda 및 Amazon DynamoDB 등 여러 AWS 서비스와 함께 사용할 수 있습니다.

Amazon SQS에서는 2가지 종류의 메시지 대기열을 제공합니다. 표준 대기열은 최대 처리량, 최선의 정렬 및 최소 1회 전송을 제공합니다. Amazon SQS FIFO 대기열은 메시지가 전송된 순서대로 정확히 한 번 제한된 처리량에 따라 처리될 수 있도록 설계되어 있습니다. 다음 시나리오는 생성에서 삭제까지 대기열에 있는 Amazon SQS 메시지의 수명 주기를 설명합니다. 여기서 생산자는 대기열에 메시지를 보내며 해당 메시지는 Amazon SQS 서버에 중복 배포됩니다.



소비자가 메시지를 처리할 준비가 완료되면 대기열에서 메시지를 검색합니다. 메시지는 처리하는 동안 대기열에 그대로 유지됩니다. 다른 소비자가 해당 메시지를 다시 처리하지 못하도록 Amazon SQS는 가시성 제한 시간을 설정합니다. 이는 Amazon SQS가 다른 소비자들이 해당 메시지를 수신하고 처리하는 것을 제한하는 시간을 의미합니다. 메시지의 가시성 제한 시간은 30초로 기본 설정됩니다. 이 사례에서는 제한 시간을 40초로 설정했습니다. 제한 시간은 최대 12시간입니다. 가시성 제한 시간이 만료되기 전에 소비자가 메시지를 삭제하지 않을 경우, 다른 소비자가 메시지를 볼 수 있게 되며 해당 메시지는 다시 처리될 수 있습니다. 일반적으로 가시성 제한 시간은 애플리케이션이 대기열에서 메시지를 처리하고 삭제하는 데 걸리는 최대 시간으로 설정해야 합니다.



Amazon SQS는 메시지를 자동으로 삭제하지 않습니다. Amazon SQS는 분산 시스템이므로 소비자가 메시지를 실제로 수신하는 것을 보장할 수 없습니다(예를 들면, 연결 문제가 있거나 소비자 애플리케이션의 문제가 있을 경우). 따라서 소비자는 수신하고 처리한 메시지를 대기열에서 삭제해야 합니다.

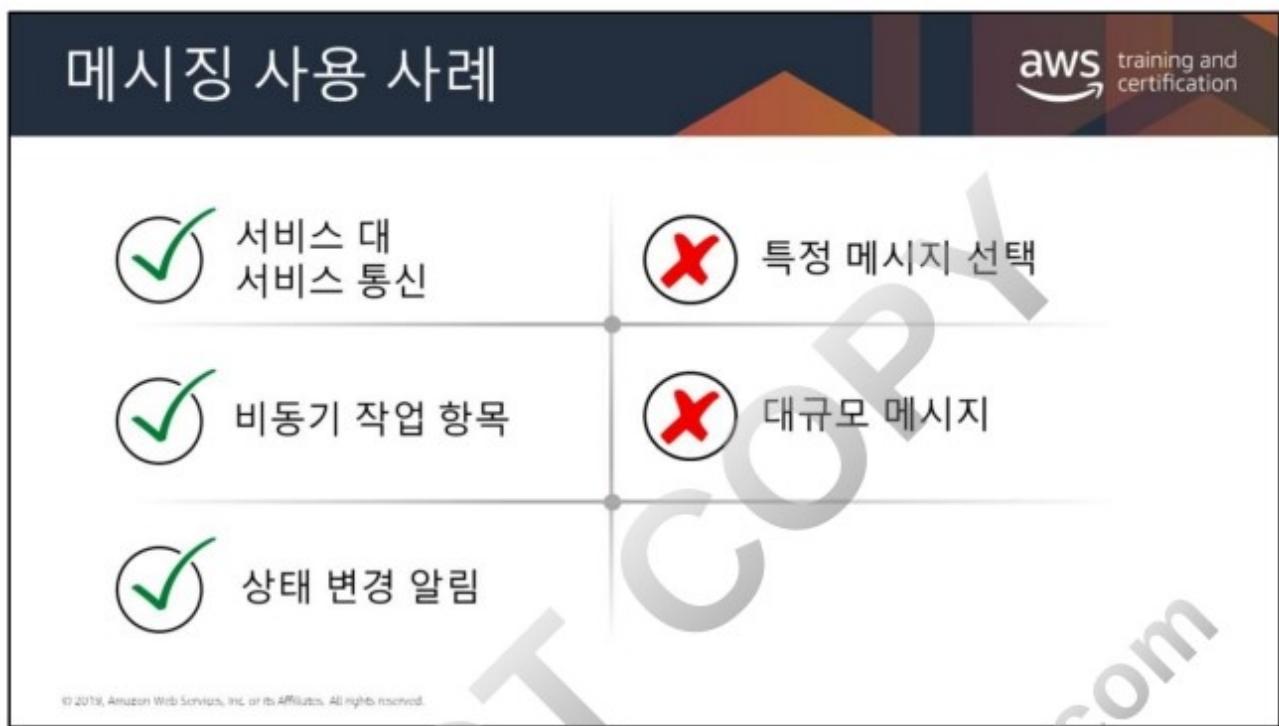
메시징 사용 사례

aws training and certification

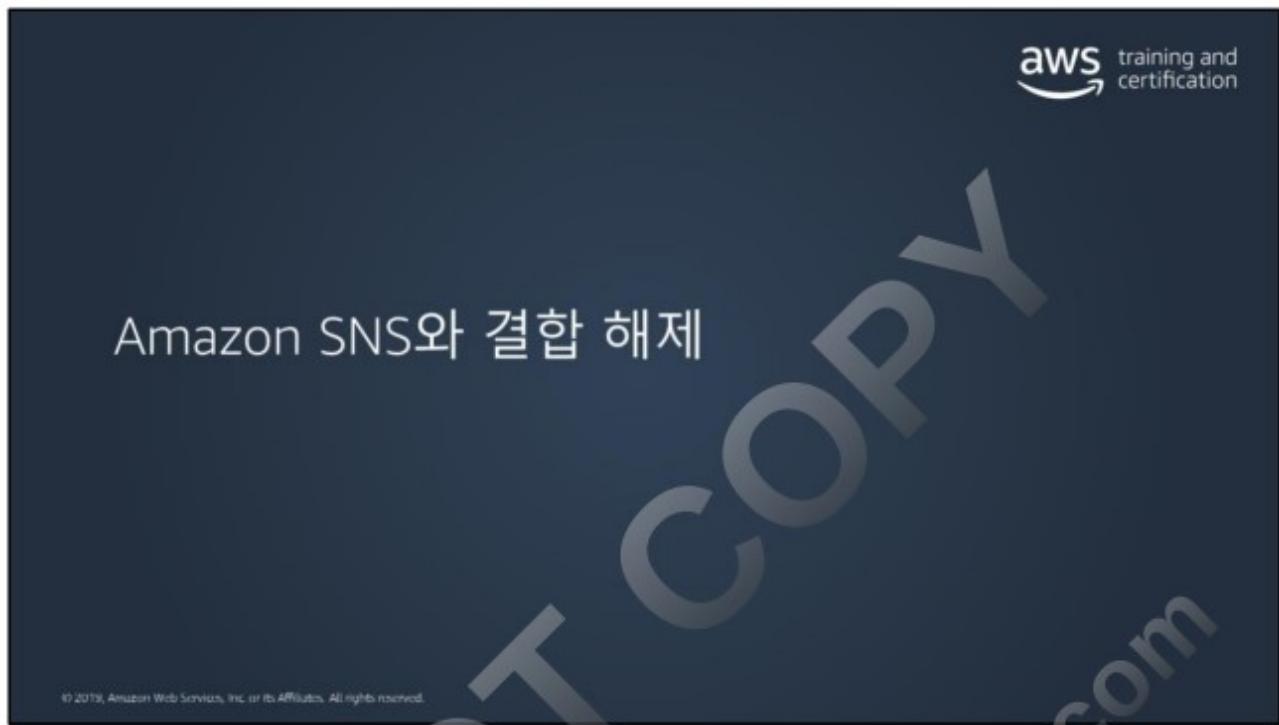
- 서비스 대
서비스 통신
- 비동기 작업 항목
- 상태 변경 알림

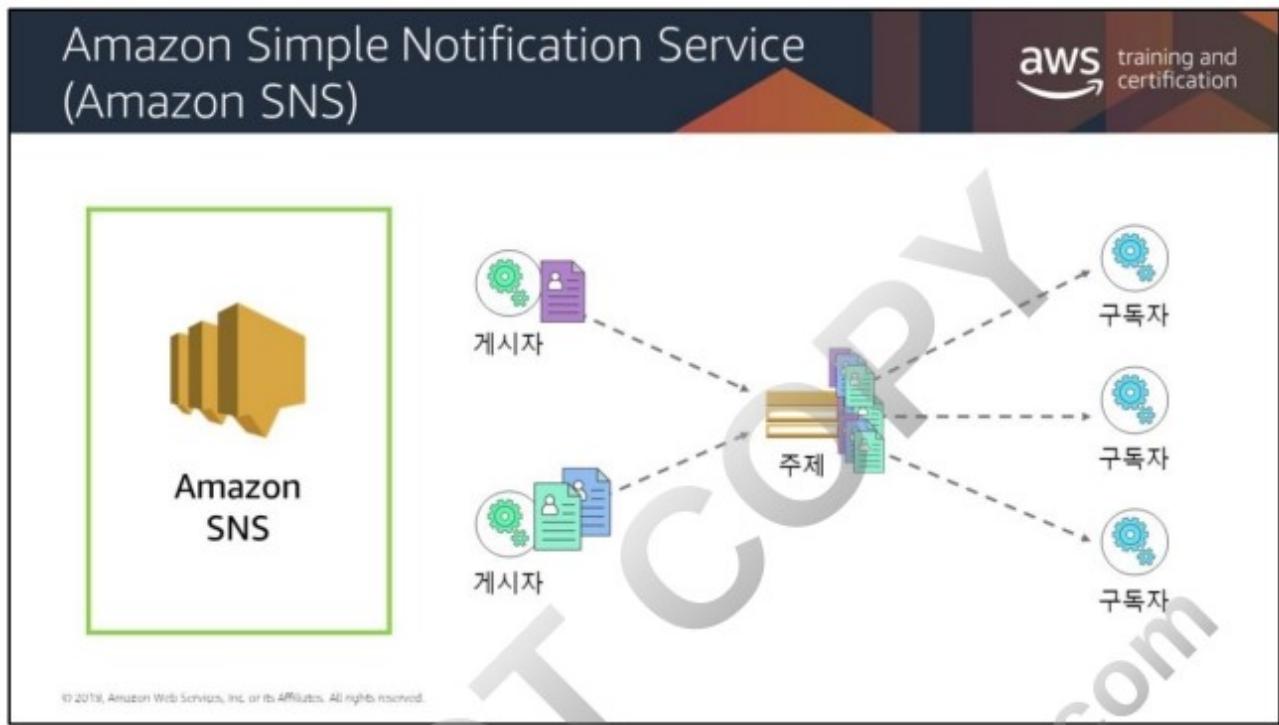
© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

메시징 서비스가 매우 적합한 몇 가지 일반적인 사용 사례가 있습니다. 서로 통신해야 하는 2개의 서비스 또는 시스템이 있는 경우가 그렇습니다. 웹 사이트(프런트 엔드)가 고객 관계 관리(CRM)시스템(백엔드)의 고객 배송 주소를 업데이트해야 한다고 생각해 보십시오. 그 대신 대기열을 설정해 프런트 엔드 웹 사이트 코드가 대기열에 메시지를 전송하도록 하고, 백엔드 CRM 서비스가 메시지를 소비하도록 할 수도 있습니다. 또는 호텔 예약 시스템이 예약을 취소해야 하는데 이 프로세스에 시간이 많이 소요된다고 가정해 보겠습니다. 그 대신 대기열에 메시지를 넣고 일부 호텔 예약 시스템이 해당 대기열에서 메시지를 소비하고 비동기 취소를 수행하도록 할 수 있습니다. 메시징 서비스는 변경 알림에도 적합합니다. 일부 리소스를 관리하는 서비스와 이러한 리소스의 변경에 대한 업데이트를 수신하는 그 밖의 서비스가 있습니다. 예를 들어 인벤토리 시스템은 일부 품목이 부족해 주문이 필요할 때 알림을 게시할 수 있습니다.



특정 기술이 사용 사례에 적합하지 않은 경우를 아는 것도 중요합니다. 메시징에는 일반적으로 보게 되는 고유한 안티 패턴이 있습니다. 특정 속성 집합과 일치하거나 심지어 애드혹 논리적 쿼리와 일치하는 메시지를 대기열에서 선택적으로 수신하는 기능이 아쉬울 수 있습니다. 예를 들어 서비스는 특정 속성이 있는 메시지를 요청하는데, 서비스가 전송한 다른 메시지에 대한 응답이 포함되어 있기 때문입니다. 이로 인해 아무도 폴링하지 않고 결코 소비되지 않는 메시지가 대기열에 있는 시나리오가 발생할 수 있습니다. 대부분의 메시징 프로토콜과 구현은 메시지의 크기가 적절할 때(수십 또는 수백 KB) 가장 효과적입니다. 메시지 크기가 커진다면, Amazon S3와 같은 전용 스토리지 시스템을 사용하고 해당 스토어에 있는 객체에 대한 참조를 메시지 자체에 넣어 전달하는 것이 가장 좋습니다.





Amazon Simple Notification Service (Amazon SNS)는 클라우드에서 손쉽게 알림 기능을 설정, 작동 및 전송할 수 있는 웹 서비스입니다. 이 서비스는 "게시-구독"(pub-sub) 메시징 패러다임을 따르며 "푸시" 메커니즘을 사용하여 클라이언트에 알림을 전달합니다.

사용자는 주제를 생성하고 어떤 게시자 및 구독자가 주제와 통신할 수 있는지를 결정하는 정책을 정의함으로써 주제에 대한 액세스를 제어합니다. 게시자는 자신이 생성한 주제 또는 게시할 권한이 있는 주제로 메시지를 보냅니다.

게시자는 각 메시지에 특정 대상 주소를 포함하는 대신 메시지를 해당 주제로 전송할 수 있습니다. Amazon SNS는 주제와 해당 주제를 구독하는 구독자 목록을 일치시켜 각각의 구독자에게 메시지를 전송합니다.

각 주제는 Amazon SNS 엔드포인트를 식별하는 고유한 이름을 가지므로, 게시자는 메시지를 게시하고 구독자는 알림을 받도록 등록할 수 있습니다. 구독자는 구독하는 주제에 게시된 모든 메시지를 수신하며, 특정 주제를 구독하는 모든 구독자는 동일한 메시지를 수신합니다.

Amazon SNS는 암호화된 주제를 지원합니다. 암호화된 주제에 메시지를 게시할 때 Amazon SNS는 메시지를 암호화하기 위해 AWS KMS (<https://aws.amazon.com/kms/>)에서 제공하는 고객 마스터 키(CMK)를 사용합니다.

Amazon SNS는 고객이 관리하는 CMK와 AWS가 관리하는 CMK를 지원합니다. Amazon SNS가 메시지를 수신하는 즉시 서버에서 256비트 AES-GCM 알고리즘을 사용한 암호화가 진행됩니다. 메시지는 내구성을 위해 여러 가용 영역(AZ)에 암호화된 형식으로 저장되며, Amazon Simple Queue Service (Amazon SQS) 대기열, AWS Lambda 함수, HTTP 및 HTTPS 웹후크 등 구독 중인 엔드포인트에 전달되기 직전에 해독됩니다.

<https://aws.amazon.com/blogs/compute/encrypting-messages-published-to-amazon-sns-with-aws-kms/>

Amazon SNS 구독 유형



Amazon SNS

- 이메일
- HTTP/HTTPS
- SMS(문자 서비스) 클라이언트
- Amazon SQS 대기열
- AWS Lambda 함수

© 2018, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

고객은 구독 요청 시 다음 전송 중 하나를 선택할 수 있습니다.

"Email" 또는 "Email-JSON" – 등록된 주소로 이메일 메시지가 전송됩니다.
Email-JSON은 알림을 JSON 객체로 전송하고, Email은 텍스트 기반
이메일로 전송합니다.

"HTTP" 또는 "HTTPS" – 구독자가 구독 등록 시 URL을 지정합니다. 알림은
HTTP POST를 통해 지정된 URL로 전송됩니다.

"SMS" – 등록된 전화번호로 SMS 문자 메시지가 전송됩니다.

"SQS" – 사용자는 SQS 표준 대기열을 엔드포인트로 지정할 수 있습니다.
Amazon SNS는 지정된 대기열에 알림 메시지를 추가합니다. 현재 FIFO
대기열은 지원되지 않습니다.

또한 메시지 사용자 지정을 처리하기 위해 AWS Lambda 함수에 메시지를 전송할
수 있으며, 메시지 지속성을 제공하거나 기타 AWS 서비스와 통신할 수도
있습니다.



Amazon SNS 알림을 사용하는 방법은 여러 가지가 있습니다.

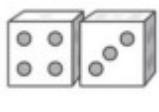
- 예를 들어, AWS Auto Scaling 그룹에 특정 변경 사항이 발생하는 등의 이벤트가 있을 경우, 사용자는 즉시 알림을 받을 수 있습니다.
- Amazon SNS를 사용하면 구독자에게 이메일 또는 SMS로 특정 뉴스 헤드라인을 푸시할 수 있습니다. 수신한 이메일 또는 SMS 문자에 흥미를 느낀 사람은 자세한 내용을 확인하기 위해 웹사이트를 방문하거나 애플리케이션을 시작할 수 있습니다.
- 업데이트가 가능함을 나타내는 알림을 앱으로 전송할 수 있습니다. 알림 메시지는 업데이트를 다운로드 및 설치하기 위한 링크를 포함할 수 있습니다.

Amazon SNS의 특성



aws training and certification

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

| | |
|---|----------------------|
|  | 하나의 게시된 메시지 |
|  | 리콜 옵션이 없음 |
|  | HTTP/HTTPS 재시도 |
|  | 주문 및 전달을 보장할 수 없음 |

모든 알림 메시지에는 게시 메시지가 하나만 포함됩니다.

Amazon SNS는 게시자가 주제에 게시한 메시지를 그 순서 그대로 전송하려고 시도합니다. 하지만 네트워크 문제로 인해 구독자에게 순서가 바뀌어 전송될 가능성도 없진 않습니다.

메시지가 성공적으로 전송되면, 이를 회수할 방법은 없습니다.

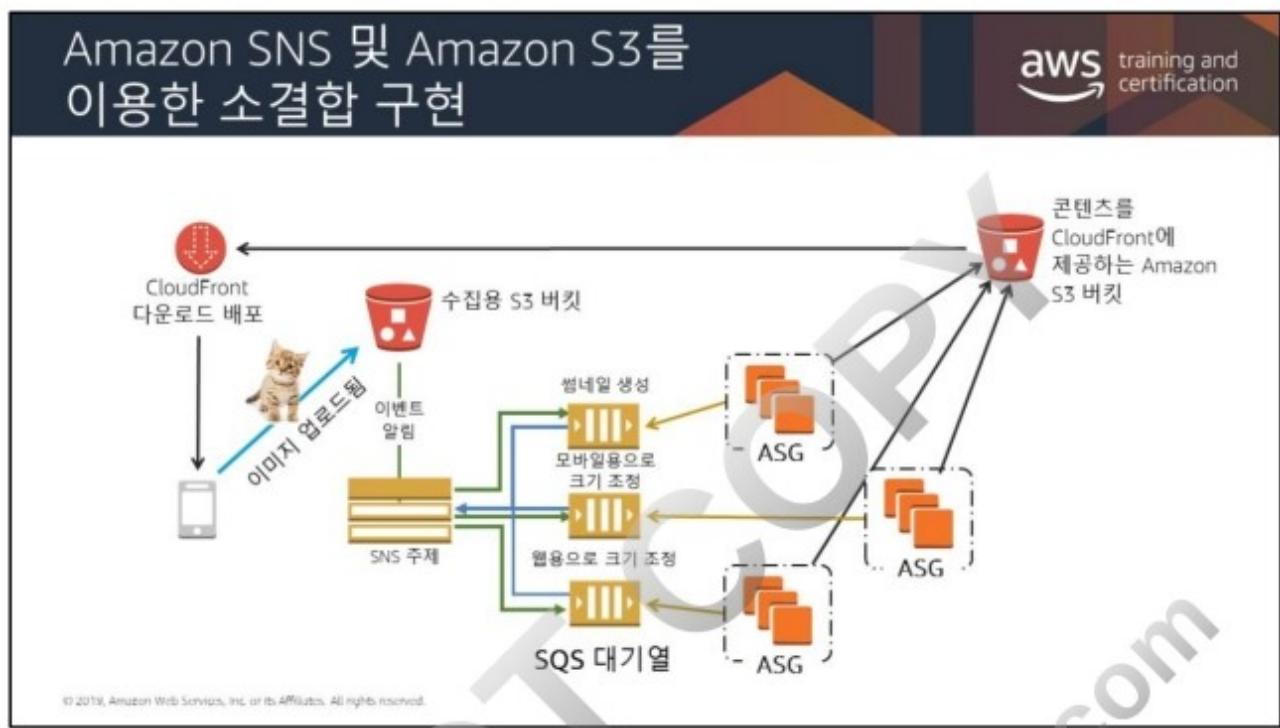
Amazon SNS 전송 정책을 사용해 재시도 패턴(선형, 기하학, 지수 백오프), 최대/최소 재시도 지연 및 기타 파라미터를 제어할 수 있습니다.

메시지가 유실되지 않도록 Amazon SNS에 게시된 모든 메시지는 여러 서버와 데이터 센터에 걸쳐 중복 저장됩니다.

Amazon SNS는 가장 크고 수요가 가장 많은 애플리케이션의 요구 사항에 부합하도록 설계되어, 애플리케이션이 언제든 무제한의 메시지를 게시할 수 있습니다.

Amazon SNS를 이용하면 서로 다른 디바이스의 애플리케이션과 최종 사용자가 모바일 푸시 알림(Apple, Google 및 Kindle Fire 디바이스), HTTP/HTTPS, Email/Email-JSON, SMS 또는 Amazon Simple Queue Service (SQS) 대기열, AWS Lambda 함수 등을 통해 알림을 수신할 수 있습니다.

Amazon SNS는 액세스 제어 메커니즘을 갖추고 있어 주제와 메시지가 무단 액세스로부터 확실하게 보호됩니다. 주제의 소유자가 주제 별로 일정한 정책을 수립해 주제를 게시하거나 구독할 수 있는 대상을 제한할 수 있습니다. 또한 전송 메커니즘을 HTTPS로 지정하여 알림을 암호화할 수도 있습니다.



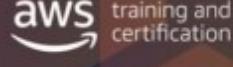
SNS에서는 주제를 사용하여 메시지 게시자를 구독자로부터 분리하고 여러 수신자에 대한 메시지를 동시에 팬아웃하며 애플리케이션에서 폴링을 제거할 수 있습니다.

SNS를 사용하면 단일 계정 내에서 메시지를 전송하거나 상이한 계정 내 리소스로 메시지를 전송하여 관리 경리를 생성할 수 있습니다.

Amazon EC2, Amazon S3 및 Amazon CloudWatch와 같은 AWS 서비스는 사용자의 SNS 주제로 메시지를 게시하여 이벤트 중심의 컴퓨팅 및 워크플로우를 트리거할 수 있습니다.

이 대체 시나리오에서는 Amazon S3에 이미지를 업로드하면 이벤트 알림이 트리거되고, 자동으로 메시지가 SNS 주제에 전송됩니다.

Amazon SNS는 Amazon SQS와 어떻게 다릅니까?

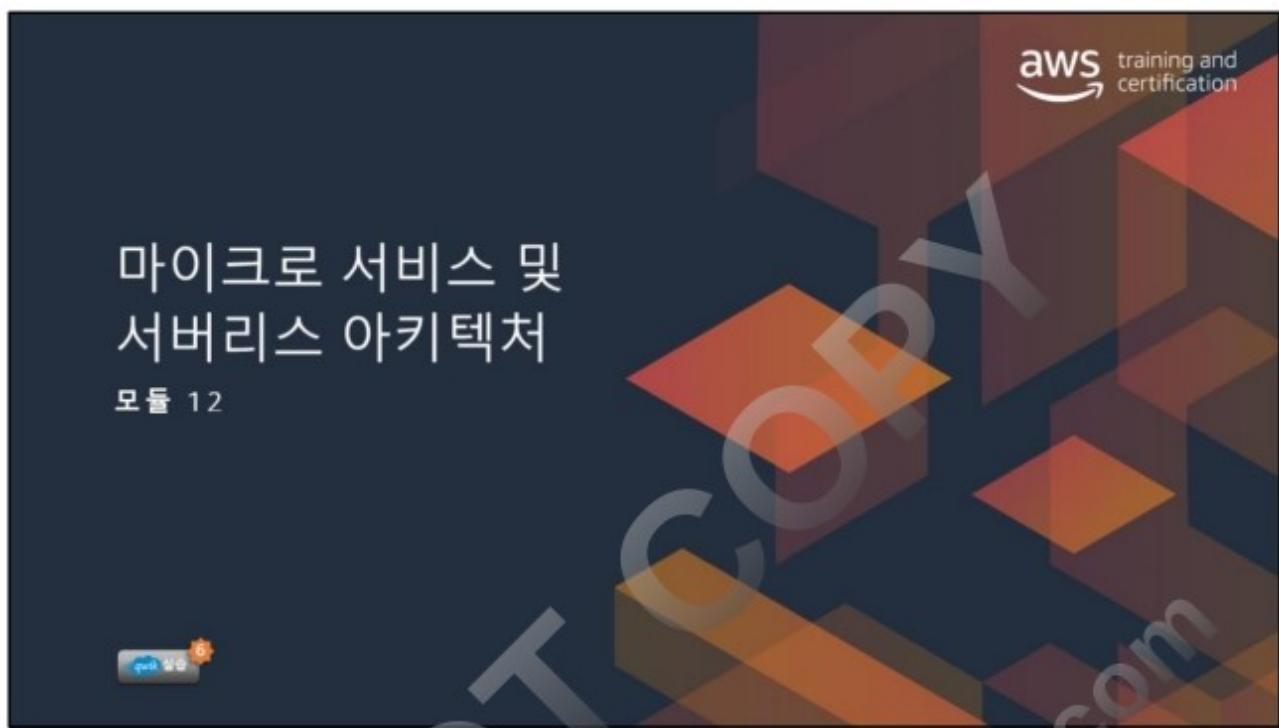


| | Amazon SNS (게시자/구독자) | Amazon SQS (생산자/소비자) |
|---------|-------------------------|-------------------------|
| 메시지 지속성 | 아니요 | 예 |
| 전송 메커니즘 | 푸시(수동적) | 풀링(능동적) |
| 생산자/소비자 | 게시/구독 | 송신/수신 |
| 배포 모델 | 일대다 | 일대일 |

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

- Amazon SNS를 사용하면 애플리케이션에서 푸시 메커니즘을 통해 타임 크리티컬 메시지를 여러 구독자에게 전송할 수 있습니다.
- Amazon SQS는 풀링 모델을 통해 메시지를 교환합니다. 즉 전송 및 수신 구성 요소가 결합 해제됩니다.
- Amazon SQS는 애플리케이션의 분산 구성 요소를 위한 유연성을 제공하므로 각 구성 요소를 동시에 사용하지 않고도 메시지를 송신 및 수신할 수 있습니다.





모듈 12



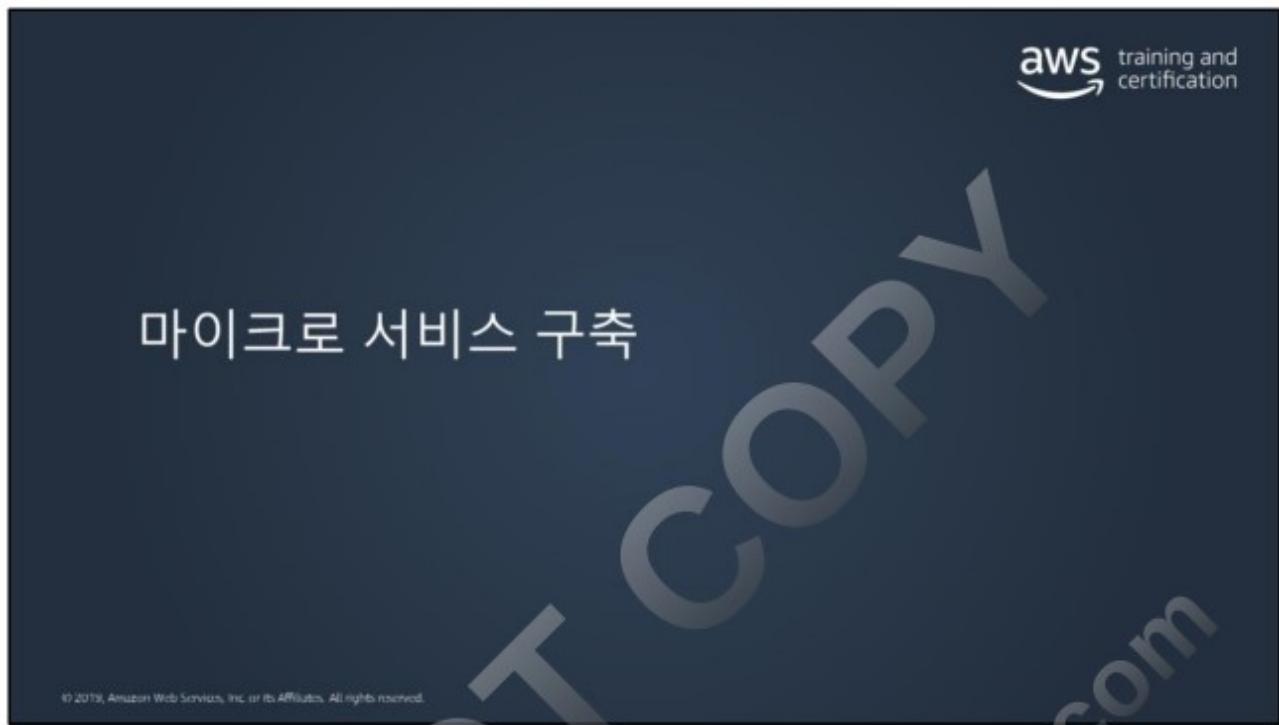
아키텍처 관련 문제

모놀리식 아키텍처가 분리되면 개별 구성 요소가 별도의 팀에서 관리되며, 이에 따라 한 팀에서 구성 요소를 변경하는 경우 충돌이 발생할 수 있습니다.

모듈 개요

- 마이크로 서비스 구축
- 컨테이너 서비스
- 서비스 환경 구현

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

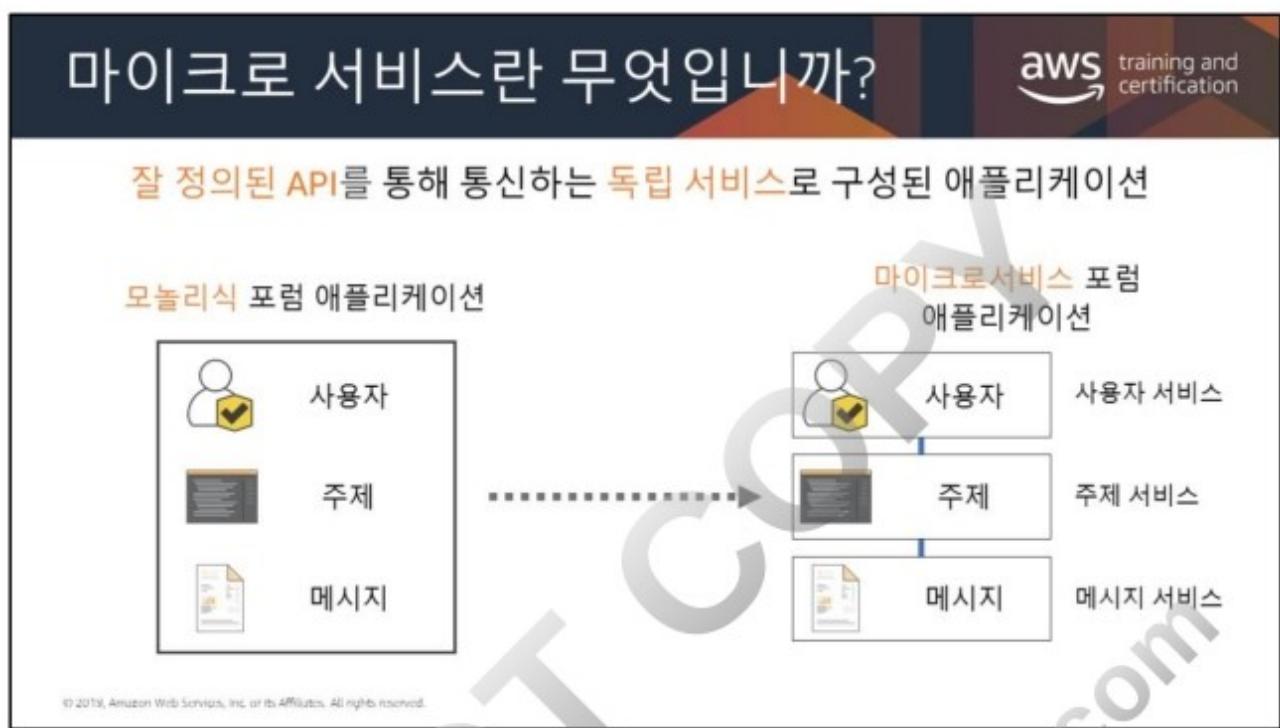


The screenshot shows a slide titled "마이크로 서비스란 무엇입니까?" (What is a microservice?). It features the AWS logo and the text "training and certification". Below the title, it says "잘 정의된 API를 통해 통신하는 독립 서비스로 구성된 애플리케이션" (Application composed of independent services communicating via well-defined APIs). A large watermark "NOT COPY" is diagonally across the slide. At the bottom left, there is a small copyright notice: "© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved."

재래식 모놀리식 애플리케이션에는 서로 긴밀히 결합된 모든 이동 및 작동 요소가 포함되어 있습니다. 한 요소에 장애가 발생할 경우 전체 애플리케이션이 중단됩니다. 수요가 급증할 경우 전체 아키텍처를 확장해야 합니다. 모놀리식 애플리케이션에 기능을 추가하는 과정은 시간이 흐를수록 복잡해집니다. 코드 베이스의 각 요소는 서로 잘 조화되어야 제대로 동기화됩니다.

마이크로서비스 아키텍처에서는 애플리케이션이 각 애플리케이션 프로세스를 서비스로 실행하는 독립적 구성 요소로 구축됩니다. 이러한 서비스는 경량 API를 사용하여 잘 정의된 인터페이스를 통해 통신합니다. 서비스는 비즈니스 기능을 위해 구축되고, 각 서비스는 단일 기능을 수행합니다. 서비스가 독립적으로 실행되므로 각 서비스를 업데이트, 배포 및 조정하며 애플리케이션의 특정 기능의 수요를 충족할 수 있습니다.

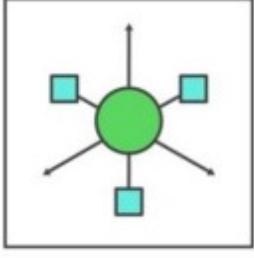
AWS 기반 마이크로서비스 아키텍처에 대한 자세한 내용은 다음을 참조하십시오.
<https://aws.amazon.com/microservices/>



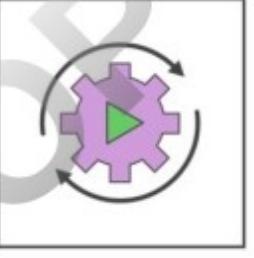
마이크로서비스의 특성

aws training and certification

자율적



전문적



© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

자율적

마이크로서비스 아키텍처의 각 구성 요소 서비스는 다른 서비스의 기능에 영향을 미치지 않고 개발, 배포, 운용 및 조정할 수 있습니다. 서비스가 다른 서비스와 어떤 코드 또는 구현도 공유할 필요가 없습니다. 개별 구성 요소 사이의 모든 통신은 잘 정의된 API를 통해 이루어집니다.

전문적

각 서비스는 일련의 기능을 제공하도록 설계되었으며 집중적으로 특정 문제를 해결합니다. 시간이 흐르면서 개발자가 서비스에 더 많은 코드를 기여하고 서비스가 복잡해지면 이를 더 작은 서비스로 분할할 수 있습니다.



컨테이너 솔루션을 생성하는 데 도움이 필요한 경우, AWS Container Competency 프로그램을 고려하십시오.

AWS 컨테이너 컴피던시는 AWS 컨테이너에서 워크로드를 실행하는 고객의 능력을 개선할 수 있도록 AWS와 통합되는 제품 또는 솔루션이 있는 APN 기술 파트너를 인정합니다. 이를 통해 사용자는 컨테이너의 모니터링, 로깅, 보안뿐 아니라 컨테이너에서 오케스트레이션 및 일정 예약, 인프라, 애플리케이션 빌드/테스트, 배포를 최적화할 수 있습니다.

APN 파트너가 AWS 컴피던시를 획득하려면 산업별 기술과 관련된 엄격한 기술적 검증을 거쳐야 합니다. 이러한 검증 덕에 고객은 AWS 파트너 네트워크 내 수십만 개의 APN 파트너 솔루션을 자신 있게 선택할 수 있습니다.

컨테이너에 대해 알아보겠습니다.

aws training and certification

The infographic features three icons: a yellow network-like icon labeled '반복 가능' (Replicable), an orange briefcase icon labeled '독립형 실행 환경' (Independent execution environment), and a yellow alarm clock icon labeled 'VM보다 더 빠른 처리 속도' (Faster processing speed than VM).

반복 가능

독립형 실행 환경

VM보다 더 빠른 처리 속도

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

마이크로서비스 지향 아키텍처의 이점은 강의가 진행되는 이 시점에도 여전히 실행 환경으로 가상 머신을 사용하고 있는 인프라 수준까지 확산되어야 합니다. 클라우드에서 VM을 실행하면 동적이고 탄력적인 환경을 구현할 수 있지만 마찰을 더 줄여야 할 수 있습니다. 클라우드에서 VM을 실행하면 동적이고 탄력적인 환경을 구현할 수 있지만 마찰을 더 줄여야 할 수 있습니다.

컨테이너란 무엇입니까?

aws training and certification

컨테이너

The diagram illustrates the components of a container. It shows a yellow rectangular box labeled "컨테이너" (Container) containing four items: "애플리케이션" (Application), "Dockerfile" (represented by a blue icon with code), "구성" (Configuration), and "OS에 연결" (Connect to OS).

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

컨테이너는 리소스 격리 프로세스에서 애플리케이션과 종속 항목을 실행하게 해주는 운영 시스템 가상화 방법입니다. 컨테이너를 사용함으로써 애플리케이션의 코드, 구성 및 종속 항목을 사용이 간편한 빌딩 블록으로 손쉽게 패키징할 수 있으며 환경 일관성, 운영 효율성, 개발자 생산성, 버전 제어를 제공합니다.

컨테이너 이미지는 컨테이너가 사용할 수 있는 파일 시스템의 스냅샷입니다. 예를 들어, Debian 운영 체제를 컨테이너 이미지로 보유할 수 있습니다. 그러한 컨테이너를 실행하면 컨테이너에서 유효하게 Debian 운영 체제를 사용하게 되는 것입니다. 또한 모든 코드 종속성을 컨테이너 이미지에 패키징하고 이를 코드 아티팩트로 사용할 수도 있습니다. 일반적으로 컨테이너 이미지는 공간 측면에서 가상 머신보다 훨씬 작습니다. 컨테이너 실행은 수백 밀리초밖에 걸리지 않습니다.

따라서 컨테이너를 사용하면 고속이고 휴대 가능하며 인프라에 구애받지 않는 실행 환경을 사용할 수 있습니다.

컨테이너는 어떤 문제를 해결할 수 있습니까?

aws training and certification

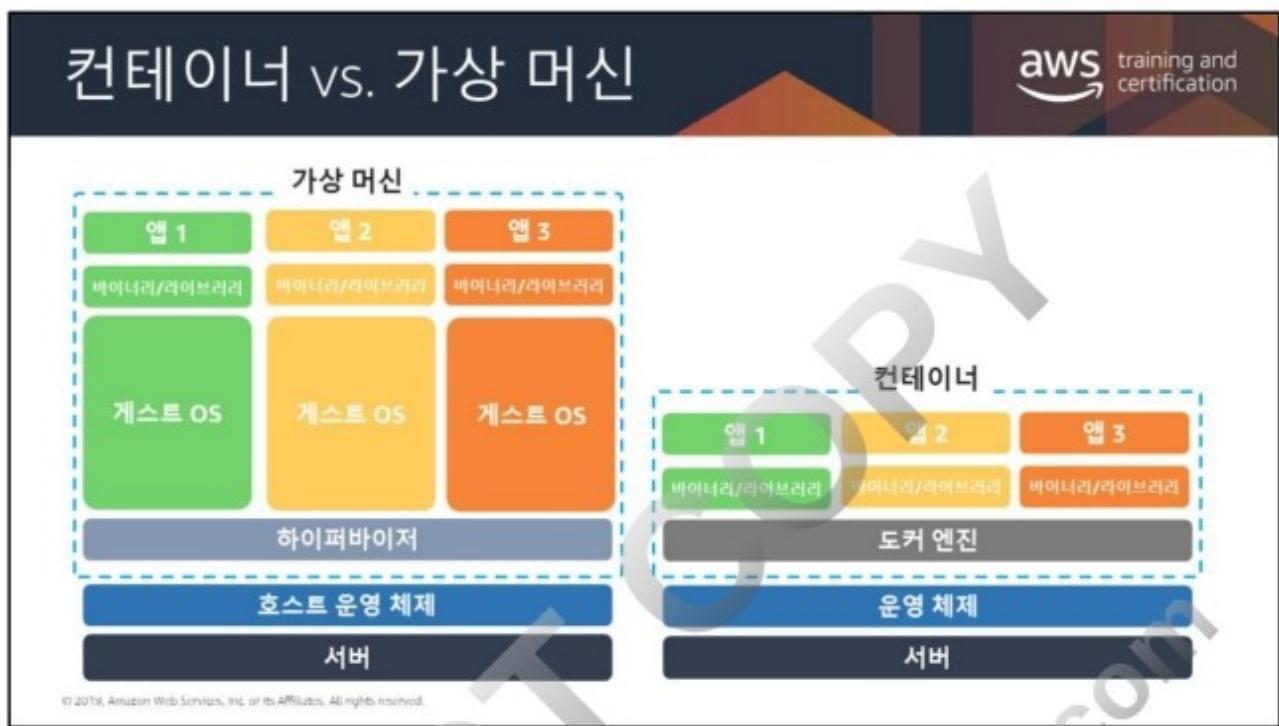
다양한 환경에서 소프트웨어를 안정적으로 실행

개발자 워크스테이션 프로덕션 테스트 환경

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

컨테이너는 애플리케이션을 배포 환경에 관계없이 빠르고 안정적으로 일관되게 배포할 수 있도록 해줍니다. 컨테이너는 리소스에 대한 좀 더 세분화된 제어가 가능하여 인프라의 효율성을 개선합니다.

사전 제작된 컨테이너 솔루션을 찾고 있다면, Amazon ECS 콘솔 또는 AWS Marketplace를 통해 AWS Marketplace for Containers에 방문해 개별 소프트웨어 판매자의 컨테이너 제품을 찾아 구매합니다. 이 제품들은 Amazon ECS, AWS Fargate, Amazon EKS 등 Docker와 호환되는 AWS 컨테이너 서비스에서 실행되는 검증된 상용 지원 제품입니다. 고성능 컴퓨팅, 보안 및 개발자 도구 등 카테고리별로 제품을 선택할 수 있습니다. 또한 컨테이너 애플리케이션을 관리, 분석 또는 보호하는 SaaS 제품도 있습니다. 자세한 내용은 <https://aws.amazon.com/marketplace/features/containers>를 참조하십시오.



컨테이너의 기능을 들으면 직관적으로 가상 머신과 비슷하다고 생각할 수 있습니다. 하지만 세부적인 부분이 다릅니다. 가장 큰 차이점은 하이퍼바이저가 필요 없다는 것입니다. 컨테이너는 적절한 Kernel 기능이 지원되고 도커 데몬이 있는 어떤 Linux 시스템에서나 실행할 수 있습니다. 이러한 특성으로 컨테이너는 휴대성이 매우 뛰어납니다. 노트북, VM, EC2 인스턴스 및 베어 메탈 서버 어디에든 호스팅할 수 있습니다.

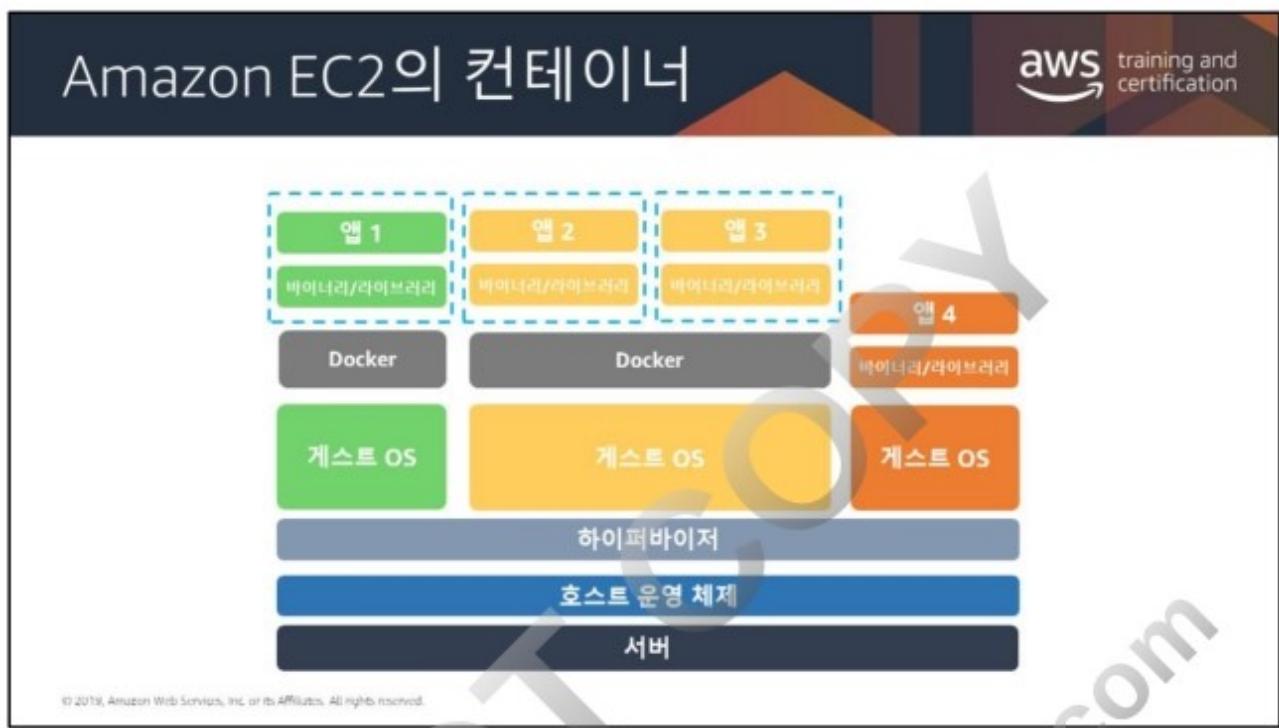
또한, 하이퍼바이저가 필요 없다는 것은 성능 오버헤드가 거의 발생하지 않는다는 뜻입니다. 프로세스가 Kernel과 직접 통신하며, 대체로 컨테이너 사일로에 대해서는 인식하지 못합니다. 대부분의 컨테이너는 몇 초 만에 부팅됩니다.

엔터프라이즈 내에는 컨테이너 및 가상 머신의 사용 사례와 그 차이점에 대한 많은 질문이 있습니다. 또한 AWS 내에서 컨테이너는 이제 Elastic Container Services, Docker, Elastic Beanstalk의 유ти리티를 통해 인프라의 핵심 부분이 되고 있습니다. 다음은 VM을 포함하여 Docker와 컨테이너의 차이점에 대한 간략한 개요입니다.

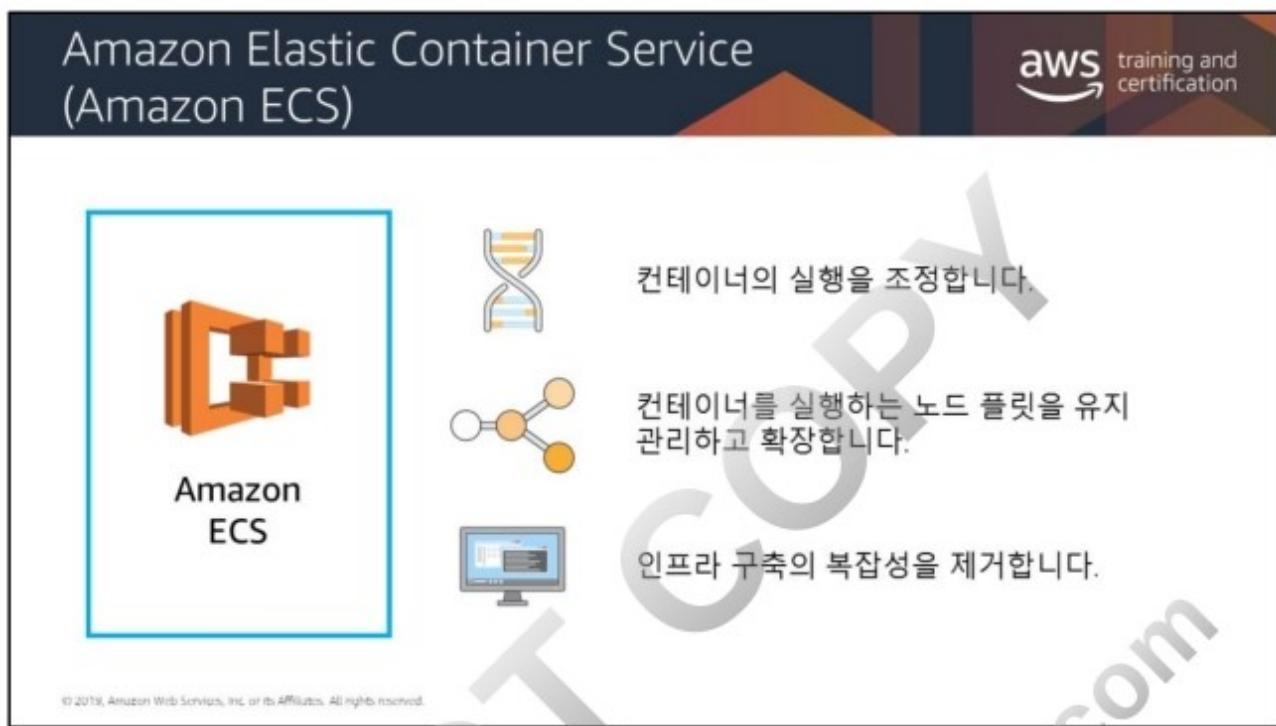
Docker, 컨테이너, VM, ECS의 개요는 다음을 참조하십시오.

<http://crmtrilogix.com/Cloud-Blog/AWS/Docker-Containers-VMs-and-ECS---an-overview/219>

DO NOT COPY
zlagusdbs@gmail.com



가상 머신 내 Amazon EC2의 컨테이너.



Amazon EC2 Container Service (Amazon ECS)는 도커 컨테이너를 지원하는 확장성과 성능이 뛰어난 컨테이너 관리 서비스로서, 서비스를 사용하여 Amazon EC2 인스턴스의 관리형 클러스터에서 애플리케이션을 손쉽게 실행할 수 있습니다.

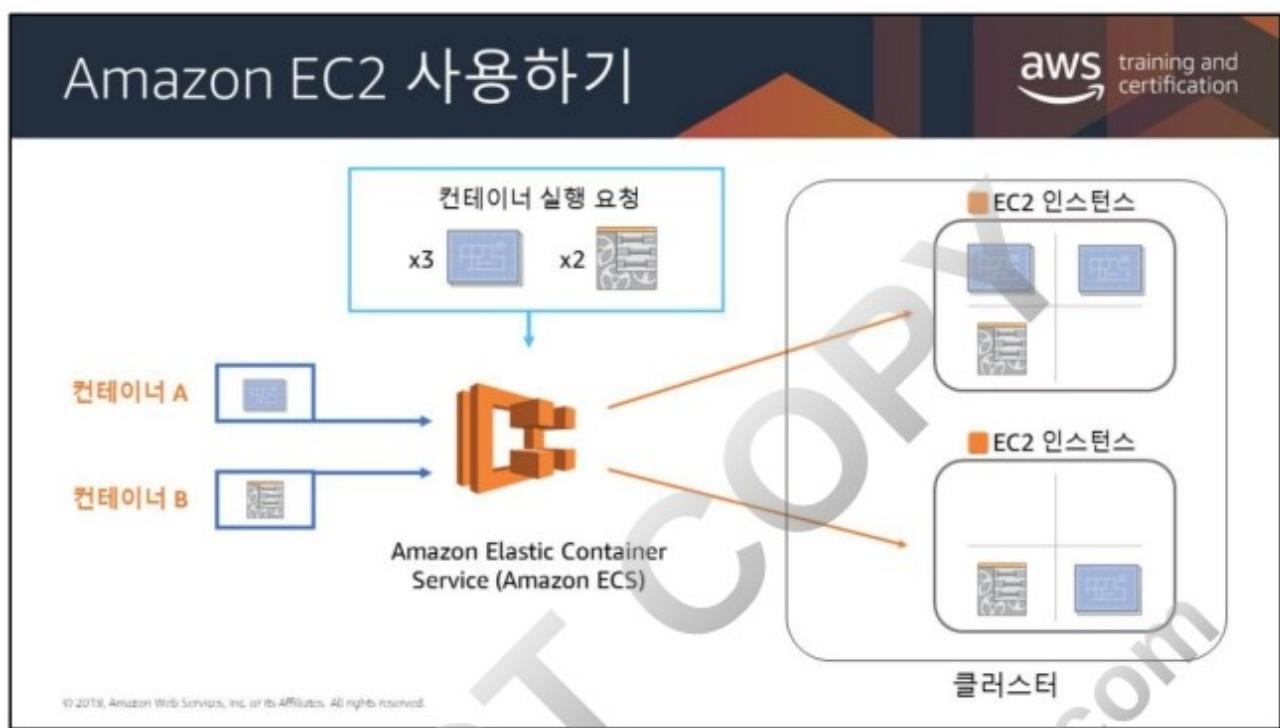
Amazon ECS는 컨테이너를 호스팅하기 위한 확장성이 뛰어난 클러스터 서비스로 다음과 같은 기능을 제공합니다.

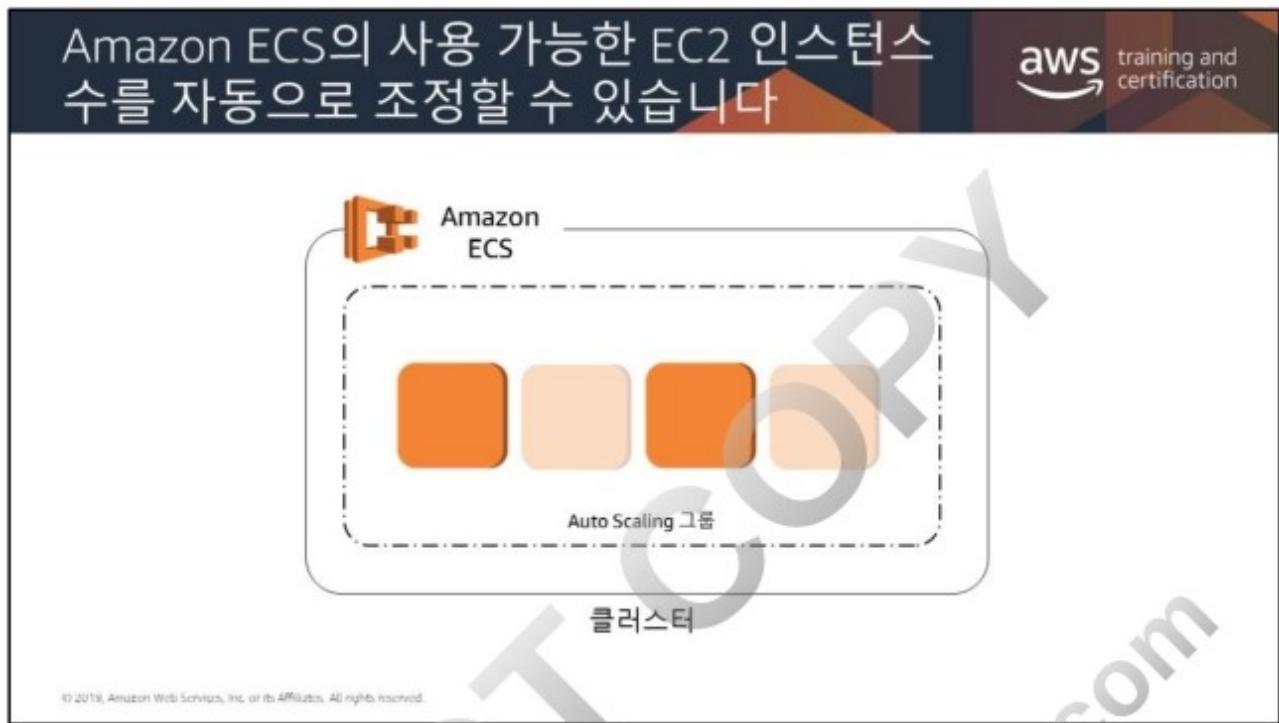
- 최대 수천 개의 인스턴스까지 확장할 수 있습니다.
- 컨테이너 배포를 모니터링합니다.
- 클러스터의 전체 상태를 관리합니다.
- 내장 스케줄러 또는 타사 스케줄러(예: Apache Mesos, Blox)를 사용하여 컨테이너 일정을 예약합니다.
- API를 사용하여 확장 가능합니다.
- Fargate 또는 EC2 시작 유형으로 시작 가능

클러스터는 스팟 인스턴스와 예약 인스턴스를 활용할 수 있습니다.

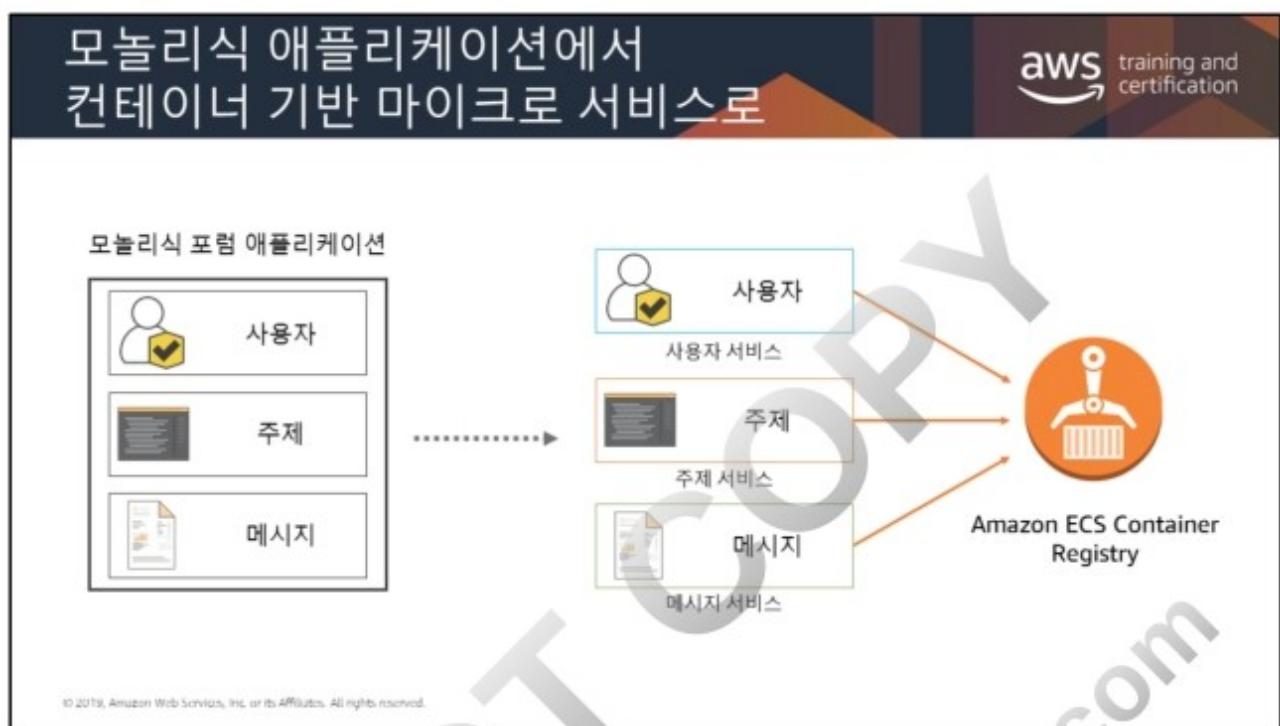
시작 유형에 대한 자세한 내용은 다음을 참조하십시오.

https://docs.aws.amazon.com/AmazonECS/latest/developerguide/launch_types.html

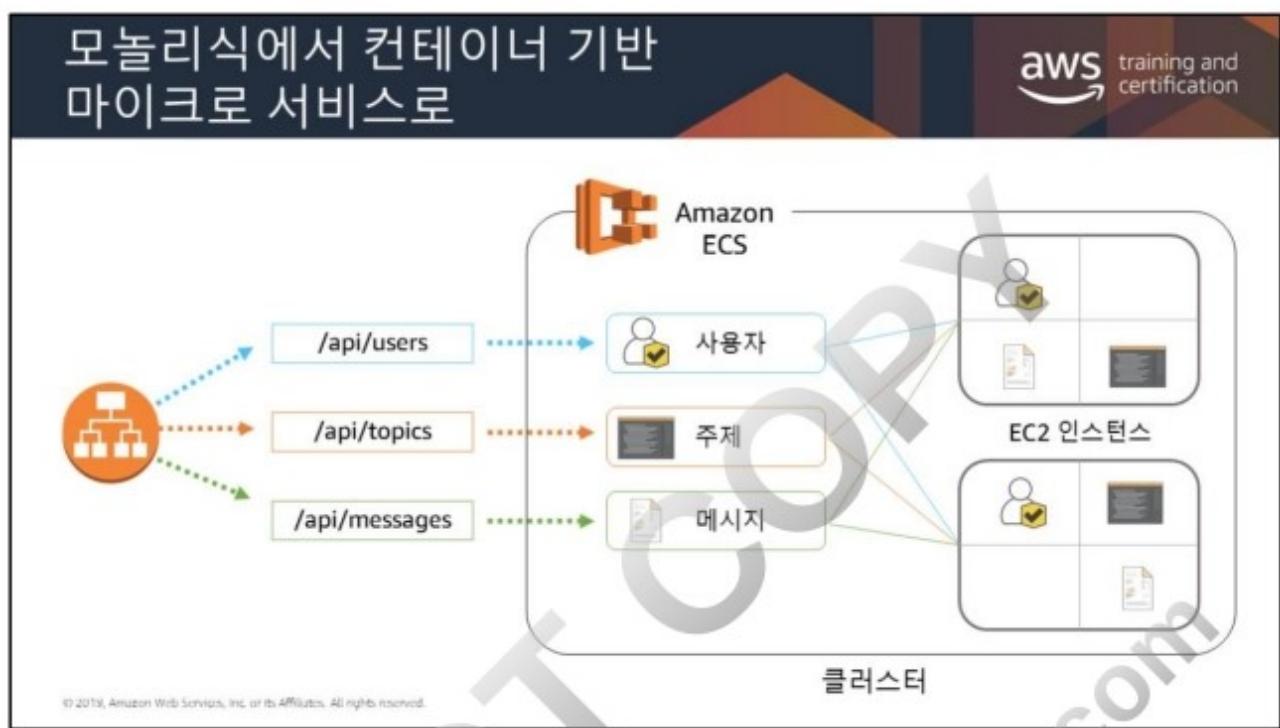




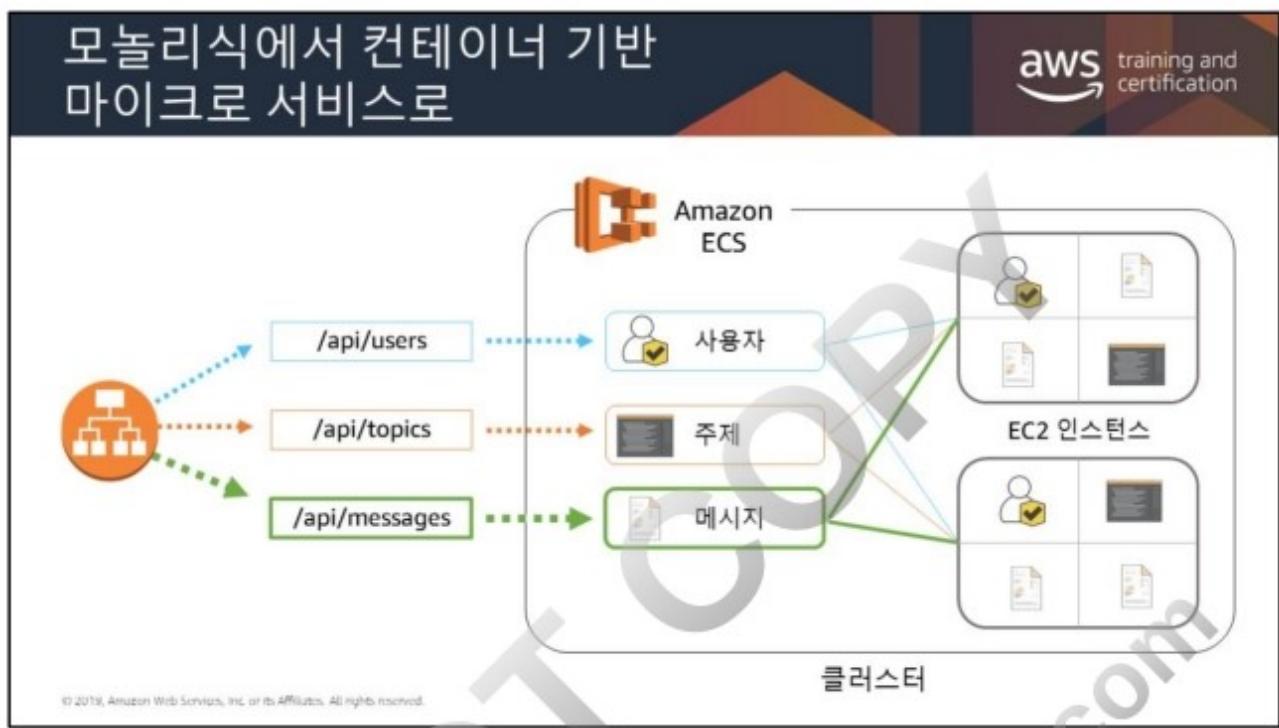
컨테이너 인스턴스를 제거하도록 Auto Scaling 그룹을 구성할 경우 제거된 컨테이너 인스턴스에서 실행되는 모든 작업이 중지됩니다. 작업이 서비스의 일부로 실행되는 경우 필요한 리소스(CPU, 메모리, 포트)가 사용 가능하면 Amazon ECS가 다른 인스턴스에서 해당 작업을 다시 시작합니다. 하지만 수동으로 시작한 작업은 자동으로 다시 시작되지 않습니다.



이 모놀리식 포럼 애플리케이션을 마이크로 서비스 접근 방식으로 전환하려면 코드를 캡슐화된 개별 서비스로 분할할 수 있습니다. 각 서비스가 제 기능을 완벽하게 수행하는지 확인한 다음, 이들 서비스를 Amazon ECS Container Registry에 등록합니다.



다음에는 Amazon ECS 내부에서 원래 애플리케이션의 이러한 요소 각각에 대해 서비스를 생성합니다. 그런 다음 이러한 서비스를 위한 대상 그룹 인스턴스를 등록합니다. 마지막으로 Amazon ECS 앱 서비스를 가리키는 대상 그룹을 포함하는 Application Load Balancer를 생성합니다.



AWS Cloud Map과 AWS App Mesh는 아키텍처 구축 및 문제 해결에 도움이 될 수 있습니다.

AWS Cloud Map은 완전 관리형 서비스로서, 모든 애플리케이션 리소스(예: 데이터베이스, 대기열, 마이크로서비스 및 기타 클라우드 리소스)를 사용자 지정 네임스페이스를 사용해 등록할 수 있습니다. 그러면 AWS Cloud Map은 리소스를 추가 및 등록할 때 수작업 매핑을 최소화하도록 리소스의 위치가 최신인지 확인하기 위해 상태를 지속적으로 확인합니다. AWS Cloud Map은 마이크로서비스와 애플리케이션의 서비스 검색, 지속적 통합, 상태 모니터링을 지원합니다. 자세한 내용은 다음을 참조하십시오.

- <https://aws.amazon.com/blogs/aws/aws-cloud-map-easily-create-and-maintain-custom-maps-of-your-applications/>
- <https://aws.amazon.com/cloud-map/>
- <https://www.youtube.com/watch?v=qTE1PbdY3hY>

AWS App Mesh은 모든 마이크로서비스에서 Amazon CloudWatch, AWS X-Ray 및 호환되는 AWS 파트너 및 커뮤니티 모니터링 및 추적 도구로 내보낼 수 있는 지표, 로그, 추적을 캡처합니다. 또한 마이크로서비스 간 트래픽 라우팅에 대한 사용자 지정 제어를 제공하여 애플리케이션의 배포, 장애 또는 조정을 지원합니다.

App Mesh는 애플리케이션 내에서 코드를 요구하거나 로드 밸런서를 사용하지 않고도 프록시를 통해 마이크로서비스를 서로 직접 연결하여 구성할 수 있습니다. App Mesh는 마이크로서비스 컨테이너와 함께 배포되는 오픈 소스 서비스 메시 프록시인 Envoy를 사용합니다. 자세한 내용은 다음을 참조하십시오.

- <https://aws.amazon.com/app-mesh/features/>
- <https://www.youtube.com/watch?v=qTE1PbdY3hY>

DO NOT COPY
zlagusdbs@gmail.com

AWS Fargate

aws training and certification

완전 관리형 컨테이너 서비스

- 클러스터 프로비저닝 및 관리
- 실행 시간 환경 관리
- 규모 조정

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

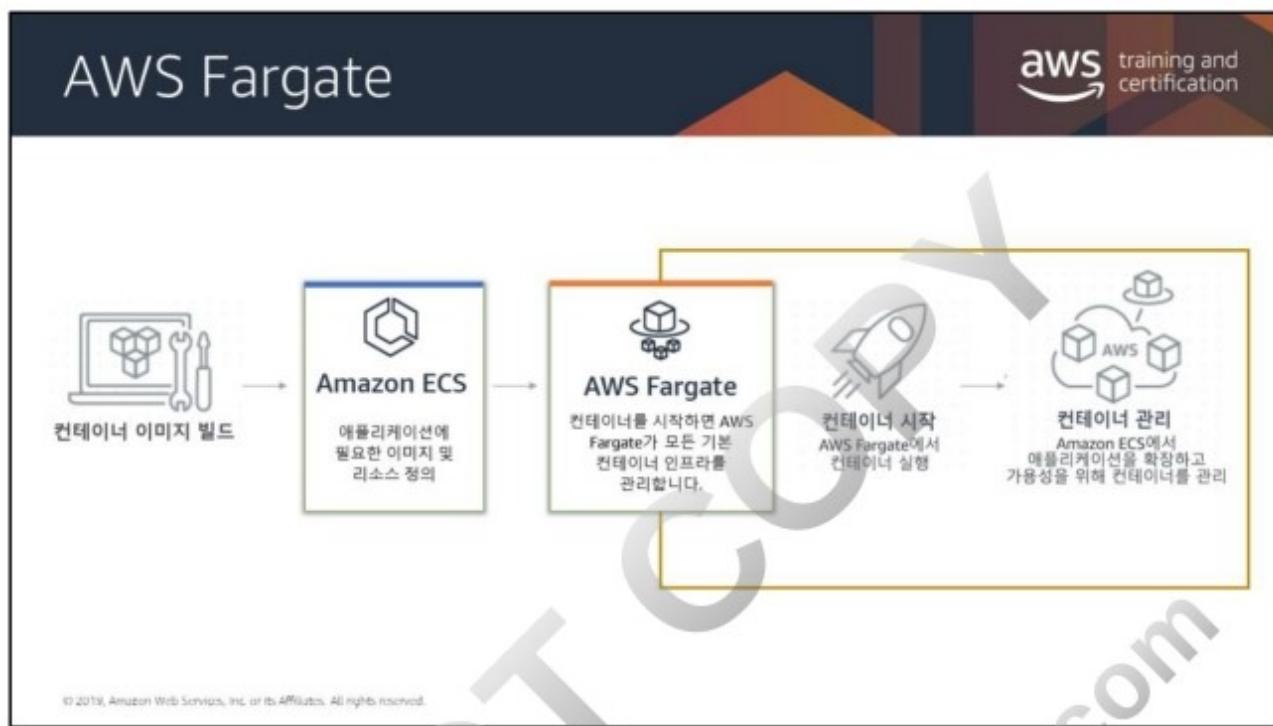
AWS Fargate는 서버 또는 클러스터를 관리할 필요 없이 [컨테이너](#)를 실행할 수 있게 해주는 Amazon ECS 및 Amazon Elastic Container Service for Kubernetes (Amazon EKS) 용 기술입니다. AWS Fargate를 사용하면 더 이상 컨테이너를 실행하기 위해 가상 머신을 프로비저닝, 구성 및 조정할 필요가 없습니다. 따라서 서버 유형을 선택하거나, 클러스터를 조정할 시점을 결정하거나, 클러스터 패킹을 최적화할 필요가 없습니다. AWS Fargate에서는 서버 또는 클러스터에 대해 고민하거나 상호 작용할 필요가 없습니다. Fargate를 사용하면 애플리케이션을 실행하는 인프라의 관리 대신 애플리케이션 설계 및 구축에 집중할 수 있습니다.

이 서비스는 Amazon ECS를 지원합니다.
AWS Fargate에 대한 자세한 내용은 다음을 참조하십시오.

<https://aws.amazon.com/fargate/>

Amazon EKS와의 통합에 대한 자세한 내용은 다음을 참조하십시오.

<https://aws.amazon.com/about-aws/whats-new/2018/11/aws-fargate-and-amazon-ecs-now-integrate-with-aws-cloud-map/>



Amazon ECS에는 Fargate 시작 유형과 EC2 시작 유형이라는 두 가지 모드가 있습니다. Fargate 시작 유형의 경우, 애플리케이션을 컨테이너로 패키징하고, CPU 및 메모리 요구 사항을 지정하고, 네트워킹 및 IAM 정책을 정의한 후 애플리케이션을 시작하기만 하면 됩니다. EC2 시작 유형의 경우, 컨테이너 애플리케이션을 실행하는 인프라에 대해 서버 수준의 좀 더 세분화된 제어를 할 수 있습니다. EC2 시작 유형에서는 Amazon ECS를 사용하여 서버 클러스터를 관리하고 서버에 컨테이너를 배치하는 일정을 예약할 수 있습니다. Amazon ECS는 클러스터 내 모든 CPU, 메모리 및 기타 리소스를 계속 추적하고, 지정한 리소스 요구 사항에 따라 컨테이너를 실행하기에 가장 적합한 서버를 찾습니다.



아키텍처가 효율적입니까?

하나의 기능만 수행하는 서비스를 지원하기 위해 전체 인스턴스를 사용하고 있습니까?

www API 간단한 앱

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

아키텍처가 효율적입니까?

aws training and certification

하나의 기능만 수행하는 서비스를 지원하기 위해 전체 인스턴스를 사용하고 있습니까?

www API 간단한 앱

다른 서비스를 활용하여 관리:

HA 및 FT 플랫 상태 모니터링 용량

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

서비스 컴퓨팅이란 무엇입니까?

aws training and certification

서버를 관리하지 않고 앱과 서비스를 구축하고 실행



© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

서비스 컴퓨팅이란 무엇입니까?

서비스 컴퓨팅을 사용하면 서버를 생각하지 않고 애플리케이션과 서비스를 구축하고 실행할 수 있습니다. 서비스 애플리케이션에서는 사용자가 서버를 프로비저닝, 확장, 관리할 필요가 없습니다. 거의 모든 유형의 애플리케이션 또는 백엔드 서비스를 위해 서비스 애플리케이션을 구축할 수 있으며, 애플리케이션을 고가용성으로 실행하고 확장하는 데 필요한 모든 것이 자동으로 처리됩니다.

서비스 컴퓨팅은 왜 사용합니까?

서비스 애플리케이션 구축은 개발자가 클라우드든 온프레미스든 서버 또는 런타임의 관리 및 운영에 신경을 쓰는 대신 핵심 제품에 집중할 수 있다는 것을 의미합니다. 이렇게 오버헤드가 줄어들면 개발자는 확장성과 안정성을 갖춘 훌륭한 제품을 개발하는 데 시간과 에너지를 쓸 수 있습니다.

자세한 내용은 다음을 참조하십시오. <https://aws.amazon.com/serverless/>

AWS Lambda



AWS Lambda

- 완전 관리형 컴퓨팅 서비스
- 상태 비저장 코드 실행
- Node.js, Java, Python, C#, Go, Ruby 지원
- 일정에서 또는 이벤트에 대한 응답으로 코드 실행
(예: Amazon S3 버킷 또는 Amazon DynamoDB 테이블의 데이터 변경)
- 엣지에서 실행 가능

© 2019, Amazon Web Services, Inc. or its Affiliates. All rights reserved.

AWS Lambda를 사용하면 서버를 프로비저닝하거나 관리할 필요 없이 코드를 실행할 수 있습니다. 이 서비스는 고가용성 컴퓨팅 인프라에서 코드를 실행하고 서버 및 운영 체제 유지 관리, 용량 프로비저닝 및 자동 조정, 코드 모니터링 및 로깅 등 모든 컴퓨팅 리소스 관리를 수행합니다. 사용자는 AWS Lambda가 지원하는 언어(현재 Node.js, Java, C#, Python, Ruby) 중 하나로 코드를 제공하기만 하면 됩니다.

Lambda@Edge는 Amazon CloudFront CDN에서 생성된 이벤트에 대한 응답으로 Lambda 함수를 실행하고 고가용성을 유지한 채로 최종 사용자에게 가장 가까운 AWS 엣지 로케이션에서 코드를 확장할 수 있습니다. Lambda 함수를 사용하여 다음 지점에서 CloudFront 요청 및 응답을 변경할 수 있습니다.

- 최종 사용자 요청
- 오리진 요청
- 오리진 응답
- 최종 사용자 응답

이를 통해 웹 사이트 보안 및 개인 정보 보호 강화, 엣지에 동적 애플리케이션 구축, SEO, 실시간 이미지 변환, 사용자 인증 및 권한 부여, 사용자 추적 및 분석, 기타 사용 사례가 가능합니다.

Lambda@Edge는 Node.js만 지원합니다

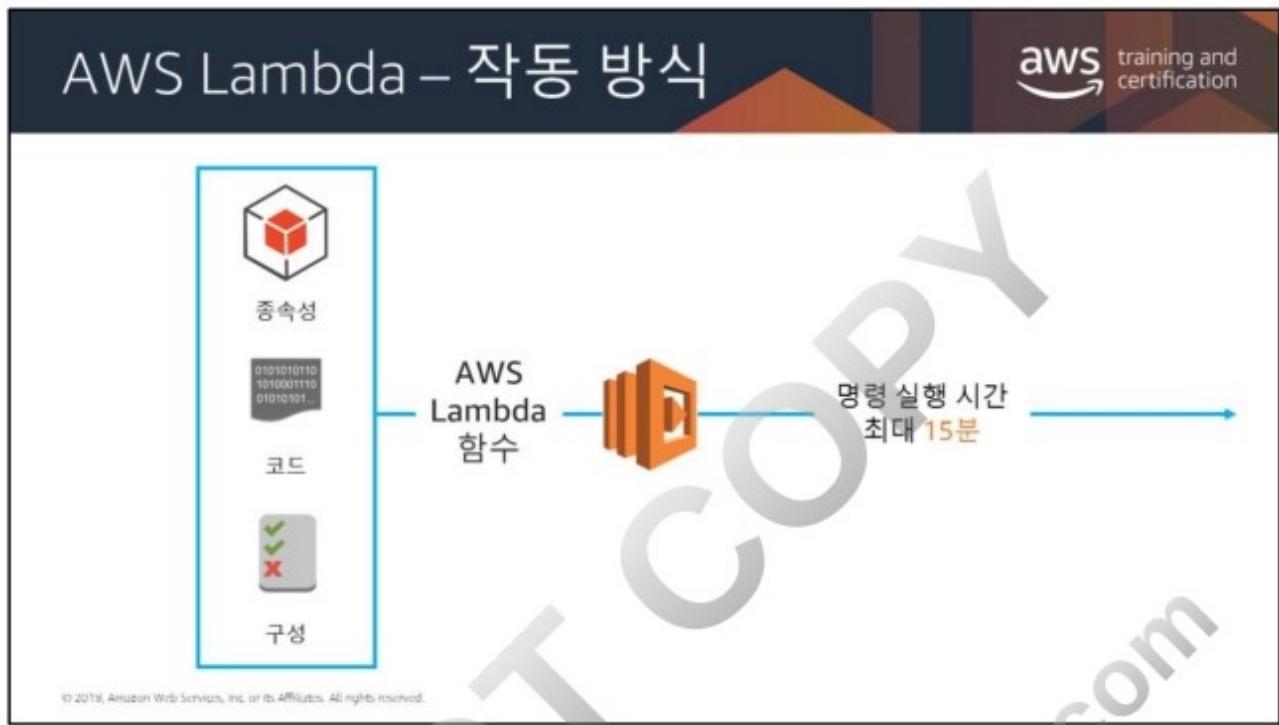
자세한 내용은 다음을 참조하십시오.

- <https://aws.amazon.com/lambda/edge/>
<https://docs.aws.amazon.com/lambda/latest/dg/lambda-edge.html>
- <https://aws.amazon.com/blogs/networking-and-content-delivery/adding-http-security-headers-using-lambdaedge-and-amazon-cloudfront/>

.NET Core 2.1 런타임을 사용하여 PowerShell Core 6.0에서 AWS Lambda 함수를 개발할 수도 있습니다. PowerShell 개발자는 AWS Lambda를 사용하여 PowerShell 환경 내에서 AWS 리소스를 관리하고 풍부한 자동화 스크립트를 작성할 수 있습니다.

자세한 내용은 다음을 참조하십시오.

- <https://aws.amazon.com/about-aws/whats-new/2018/09/aws-lambda-supports-powershell-core/>



AWS Lambda의 핵심 구성 요소는 이벤트 소스와 *Lambda* 함수입니다. 이벤트 소스는 이벤트를 게시하고, *Lambda* 함수는 이벤트를 처리하도록 사용자가 작성하는 사용자 지정 코드입니다. *Lambda*는 사용자 대신 *Lambda* 함수를 실행합니다.

Lambda 함수는 코드, 관련 종속성 및 구성으로 이루어집니다. 구성에는 이벤트를 수신하는 핸들러, 사용자 대신 *Lambda* 함수를 실행하기 위해 *AWS Lambda*가 맡을 수 있는 IAM 역할, 할당할 컴퓨팅 리소스, 실행 제한 시간 등의 정보가 포함됩니다.

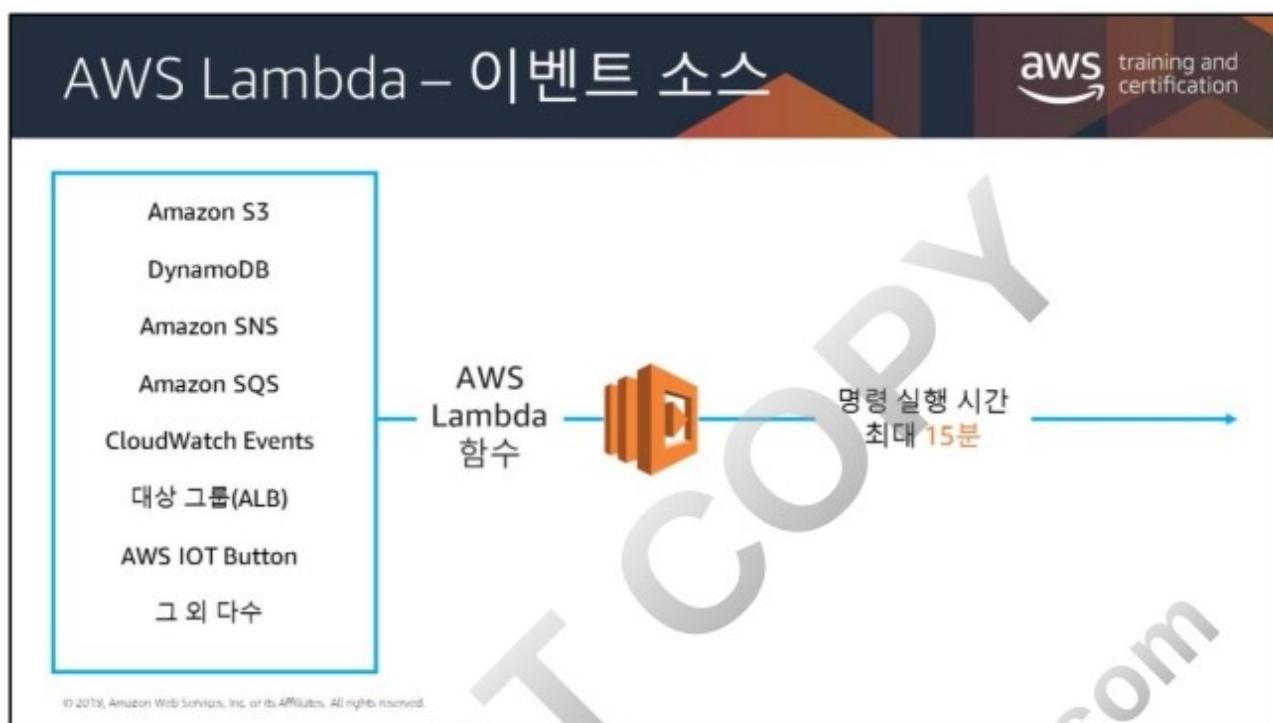
계층을 사용하면 *AWS Lambda* 함수 개발자가 패키지, 바이너리, 런타임 및 *Lambda* 함수에 필요한 그 밖의 파일을 함수 코드와 별개의 구성 요소로 유지할 수 있습니다. *Lambda* 함수를 생성할 때 함수의 실행 환경에 포함될 하나 이상의 계층을 지정할 수 있습니다. 이렇게 하면 여러 *Lambda* 함수에 분산된 동일한 파일의 사본을 유지할 필요가 없습니다. 예를 들어 Python으로 작성된 서비스 애플리케이션은 PyMySQL 같은 패키지를 사용하여 Amazon RDS MySQL 데이터베이스를 쿼리할 수 있습니다. 계층이 있는 경우 PyMySQL 패키지의 단일 사본만 유지하면 애플리케이션 내 모든 함수가 이를 사용할 수 있습니다.

Lambda 계층 제한:

- 단일 함수는 한 번에 최대 5개의 계층을 소비할 수 있습니다.
- 압축되지 않은 함수의 총 크기(계층 포함)는 압축되지 않은 배포 패키지 크기 제한인 250MB를 초과할 수 없습니다.

Lambda 계층은 리소스 수준 권한을 지원하며, 특정 AWS 계정, AWS Organizations 또는 모든 계정에서 공유할 수 있습니다. 계층은 함수 생성 도중이나 이후에 추가할 수 있으며, 필요한 경우 업데이트도 가능합니다. AWS Serverless Application Model (SAM)도 함수 전반의 계층 관리를 지원합니다.

함수 버전과 마찬가지로 계층은 개별 버전 및 해당 권한을 지원합니다. 게시된 계층 버전은 업데이트할 수 없습니다(버전 권한 변경 제외). 계층을 업데이트하려면 새 버전을 게시해야 합니다. 이렇게 하면 여러 함수에서 새 계층의 룰아웃을 제어할 수 있습니다.



ALB를 사용하여 HTTP/HTTPS를 통해 Lambda 함수에 트래픽을 전송할 수 있습니다. ALB가 콘텐츠 기반 라우팅이므로 ALB로 들어오는 요청의 호스트 또는 호스트 및 URL 경로를 기반으로 다른 Lambda 함수에 트래픽을 전송할 수도 있습니다. ALB 대상으로 Lambda 함수를 등록하면 로드 밸런서가 JSON 형식으로 Lambda 함수에 콘텐츠를 전달합니다. 기본적으로 Lambda 유형의 대상 그룹에 대한 상태 확인은 비활성화되어 있습니다.

<https://docs.aws.amazon.com/elasticloadbalancing/latest/application/lambda-functions.html>