AWS 백서

모범 사례 설계 패턴: Amazon S3 성능 최적화

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

모범 사례 설계 패턴: Amazon S3 성능 최적화: AWS 백서

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon의 상표 및 브랜드 디자인은 Amazon 외 제품 또는 서비스와 함께, Amazon 브랜드 이미지를 떨어뜨리거나 고객에게 혼동을 일으킬 수 있는 방식으로 사용할 수 없습니다. Amazon이 소유하지 않은 기타 모든 상표는 Amazon과 제휴 관계이거나 관련이 있거나 후원 관계 여부에 관계없이 해당 소유자의 자산입니다.

Table of Contents

모범 사례 설계 패턴: Amazon S3 성능 최적화

최초 게시 날짜: 2019년 6월(문서 개정)

요약

Amazon S3에서 스토리지를 업로드 및 검색하는 애플리케이션을 구축할 때 AWS 모범 사례 가이드에 따라 성능을 최적화합니다. AWS는 더 자세한 성능 설계 패턴도 제공합니다.

소개

애플리케이션은 Amazon S3의 스토리지를 업로드하고 검색할 때 요청 성능에서 초당 수천 회의 트랜잭션을 쉽게 달성할 수 있습니다. Amazon S3은 높은 요청 빈도로 자동으로 조정됩니다. 예를 들어, 애플리케이션은 버킷의 접두사마다 초당 3,500개 이상의 PUT/COPY/POST/DELETE 요청 및 5,500개 이상의 GET/HEAD 요청을 달성할 수 있습니다. 버킷의 접두사 수에는 제한이 없습니다. 읽기를 병렬화하여 읽기 또는 쓰기 성능을 향상시킬 수 있습니다. 예를 들어, Amazon S3 버킷에서 접두사 10개를 만들어 읽기를 병렬화하는 경우 읽기 성능을 초당 읽기 요청 55,000개로 조정할 수 있습니다.

Amazon S3의 일부 데이터 레이크 애플리케이션은 페타바이트 이상의 데이터에 걸쳐 실행되는 쿼리에서 수백만 또는 수십억 개의 객체를 검색합니다. 이러한 데이터 레이크 애플리케이션으로 단일 인스턴스에서 최대 100Gb/s가 될 수 있는 Amazon EC2 인스턴스의 네트워크 인터페이스 사용을 극대화하는 단일 인스턴스 전송 속도를 달성합니다. 그런 다음 이러한 애플리케이션은 여러 인스턴스에서 처리량을 집계하여 초당 여러 테라비트를 얻습니다.

또 다른 예는 소셜 미디어 메시징 애플리케이션처럼 대기 시간에 민감한 애플리케이션입니다. 이러한 애플리케이션은 약 100~200밀리초의 일관된 작은 객체 대기 시간(큰 객체의 경우 첫 번째 바이트 출력 대기 시간)을 달성할 수 있습니다.

다른 AWS 서비스도 다른 애플리케이션 아키텍처의 성능을 가속화하는 데 도움이 됩니다. 예를 들어, 단일 HTTP 연결 또는 한 자릿수 밀리초 대기 시간에 비해 더 높은 전송 속도를 원한다면 Amazon S3으로 캐싱하기 위해 Amazon CloudFront 또는 Amazon ElastiCache를 사용합니다.

또한 클라이언트와 S3 버킷 간 장거리에 걸쳐 고속 데이터 전송을 원할 경우 Amazon S3 Transfer Acceleration을 사용합니다. Transfer Acceleration은 CloudFront에서 전 세계에 분산된 엣지 로케이션을 사용하여 지리적으로 먼 거리 간 데이터 전송을 가속화합니다.

Amazon S3 워크로드에서 AWS Key Management Service를 통한 서버 측 암호화(SSE-KMS)가 사용되는 경우 사용 사례에 지원되는 요청 빈도에 대한 자세한 내용은 AWS Key Management Service 개발자 가이드의 AWS KMS 제한 단원을 참조하세요.

다음 주제에서는 Amazon S3을 사용하는 애플리케이션의 성능을 최적화하기 위한 모범 사례 가이드와 설계 패턴에 대해 설명합니다.

이 가이드는 Amazon S3 성능 최적화에 대한 이전 가이드를 대체합니다. 예를 들어, 이전의 Amazon S3 성능 가이드에서는 잦은 데이터 검색 성능을 최적화하기 위해 해시된 문자를 사용하여 접두사 이름을 임의로 지정할 것을 권장했습니다. 그러나 더 이상 성능을 위해 접두사 이름을 무작위로 지정할 필요가 없으며 접두사에 대해 순차적 날짜 기반의 이름을 지정할 수 있습니다. Amazon S3의 성능 최적화에 대한 최신 정보는 성능 가이드 및 성능 설계 패턴을 참조하세요.

Amazon S3 성능 가이드

Amazon S3에서 애플리케이션의 성능을 최상으로 유지하려면 다음 가이드를 따르는 것이 좋습니다.

주제

성능 측정

성능을 최적화할 때는 네트워크 처리량, CPU 및 Dynamic Random Access Memory(DRAM) 요구 사항을 살펴보세요. 이처럼 다양한 리소스에 대한 요구 조건의 조합에 따라 다른 Amazon EC2 인스턴스 유형을 평가할 가치가 있습니다. 인스턴스 유형에 대한 자세한 내용은 Linux 인스턴스용 Amazon EC2 사용 설명서의 인스턴스 유형을 참조하십시오.

성능을 측정할 때 HTTP 분석 도구를 사용하여 DNS 조회 시간, 대기 시간 및 데이터 전송 속도를 살펴보는 것도 도움이 됩니다.

스토리지 연결 수평 확장

여러 연결에 걸쳐 요청을 분산시키는 것은 성능을 수평으로 확장하는 일반적인 설계 패턴입니다. 고성능 애플리케이션을 빌드할 때는 Amazon S3을 기존의 스토리지 서버와 같은 단일 네트워크 엔드포인트가 아닌 아주 큰 분산 시스템처럼 생각하십시오. Amazon S3으로 여러 건의 동시 요청을 보내 최상의 성능을 달성할 수 있습니다. Amazon S3에서 액세스할 수 있는 대역폭을 최대화하기 위해 이러한 요청을 별도의 연결로 분산합니다. Amazon S3은 버킷에 대한 연결 수 제한이 없습니다.

바이트 범위 가져오기 사용

GET Object 요청에서 Range HTTP 헤더를 사용하면 객체에서 바이트 범위를 가져와 지정된 부분만 전송할 수 있습니다. Amazon S3에 대한 동시 연결을 사용하여 동일한 객체 내에서 서로 다른 바이트

범위를 가져올 수 있습니다. 그러면 전체 객체 요청 한 건에 비해 집계 처리량을 높일 수 있습니다. 더 작은 범위의 큰 객체를 가져와 요청이 중단되었을 때 애플리케이션의 재시도 횟수를 개선할 수도 있습니다. 자세한 내용은 객체 가져오기를 참조하세요.

바이트 범위 요청의 일반적인 크기는 8MB 또는 16MB입니다. 객체가 멀티파트 업로드를 사용해 PUT되는 경우 최상의 성능을 위해 동일한 파트 크기(또는 최소한 파트 경계에 맞춤)로 GET하는 것이 좋습니다. GET 요청은 개별 파트(예: `GET ?partNumber=N`)를 직접 처리할 수 있습니다.

대기 시간에 민감한 애플리케이션 요청 재시도

공격적인 제한 시간과 재시도는 일관된 대기 시간을 유지하는 데 도움이 됩니다. Amazon S3의 큰 규모로 볼 때, 첫 번째 요청이 느린 경우 재시도된 요청은 다른 경로를 취하여 신속하게 성공할 가능성이 높습니다. AWS SDK에는 특정 애플리케이션의 허용 오차에 따라 튜닝할 수 있는 구성 가능한 제한 시간 및 재시도 값이 있습니다.

동일한 AWS 리전에서 Amazon S3(스토리지)와 Amazon EC2(컴퓨팅) 결합

S3 버킷 이름은 전 세계적으로 고유하지만 각 버킷은 버킷을 만들 때 선택한 리전에 저장됩니다. 성능을 최적화하려면, 가급적 같은 AWS 리전의 Amazon EC2 인스턴스에서 버킷에 액세스하는 것이 좋습니다. 그러면 네트워크 대기 시간 및 데이터 전송 비용을 줄일 수 있습니다.

데이터 전송 요금에 대한 자세한 내용은 Amazon S3 요금을 참조하십시오.

거리에 의해 발생하는 대기 시간을 최소화하기 위해 Amazon S3 Transfer Acceleration 사용

Amazon S3 Transfer Acceleration을 사용하면 지리적으로 거리가 먼 클라이언트와 S3 버킷 간에 파일을 빠르고 쉽고 안전하게 전송할 수 있습니다. Transfer Acceleration은 Amazon CloudFront에서 전 세계에 분산된 엣지 로케이션을 활용합니다. 엣지 로케이션에 도착한 데이터는 최적화된 네트워크 경로를 통해 Amazon S3으로 라우팅됩니다. Transfer Acceleration은 여러 대륙에 걸쳐 기가바이트 또는 테라바이트 용량의 데이터를 정기적으로 전송하는 데 이상적입니다. 전 세계의 중앙 집중식 버킷에 업로드하는 클라이언트에도 유용합니다.

Amazon S3 Transfer Acceleration 속도 비교 도구를 사용하면 Amazon S3 리전에서 속도를 높인 경우와 그렇지 않은 경우의 업로드 속도를 비교할 수 있습니다. 이 속도 비교 도구는 멀티파트 업로드를

통해 Amazon S3 Transfer Acceleration을 사용하거나 사용하지 않으면서 브라우저에서 여러 Amazon S3 리전으로 파일을 전송합니다.

최신 버전의 AWS SDK 사용

AWS SDK는 Amazon S3 성능을 최적화하기 위한 다양한 권장 가이드를 기본적으로 지원합니다. SDK는 애플리케이션 내에서 Amazon S3을 활용할 수 있는 더 간단한 API를 제공하며 최신 모범 사례를 따르기 위해 정기적으로 업데이트됩니다. 예를 들어 SDK에는 HTTP 503 오류에 대한 요청을 자동으로 다시 시도하는 로직이 포함되어 있으며 느린 연결에 응답하고 적응하는 코드에 투자하고 있습니다.

SDK는 또한 경우에 따라 바이트 범위 요청을 사용하여 초당 수천 건의 요청을 달성하기 위해 수평 조정 연결을 자동화하는 Transfer Manager를 제공합니다. 최신 성능 최적화 기능을 사용하려면 최신 버전의 AWS SDK를 사용하는 것이 중요합니다.

HTTP REST API 요청을 사용하여 성능을 최적화 할 수도 있습니다. REST API를 사용할 때는 SDK에 포함된 것과 동일한 모범 사례를 따라야 합니다. 느린 요청에 대해 제한 시간 및 재시도를 허용하고 객체 데이터를 병렬로 가져올 수 있도록 다중 연결을 허용합니다. REST API 사용에 대한 자세한 내용은 Amazon Simple Storage Service API 참조를 참조하십시오.

Amazon S3의 성능 설계 패턴

Amazon S3에서 스토리지를 업로드하고 검색할 애플리케이션을 설계할 때, 최상의 애플리케이션 성능을 얻기 위해 모범 사례 설계 패턴을 사용합니다. 또한 애플리케이션 아키텍처를 계획할 때 고려해야 할 성능 가이드도 제공합니다.

성능을 최적화하려면 다음과 같은 설계 패턴을 사용할 수 있습니다.

주제

- 자주 액세스하는 콘텐츠에 캐싱 사용
- 대기 시간에 민감한 애플리케이션의 제한 시간 및 재시도
- 높은 처리량을 위한 수평 크기 조정 및 요청 병렬화
- Amazon S3 Transfer Acceleration을 사용하여 지리적으로 분산된 데이터 전송 가속화

자주 액세스하는 콘텐츠에 캐싱 사용

Amazon S3에 데이터를 저장하는 많은 애플리케이션은 사용자가 반복적으로 요청하는 데이터의 "작업 집합"을 제공합니다. 워크로드가 공통 객체 집합에 반복적인 GET 요청을 보내는 경우 Amazon CloudFront, Amazon ElastiCache 또는 AWS Elemental MediaStore와 같은 캐시를 사용하여 성능을 최적화할 수 있습니다. 캐시를 성공적으로 채택하면 대기 시간이 짧아지고 데이터 전송률이 높아질 수 있습니다. 그리고 캐싱을 사용하는 애플리케이션은 Amazon S3에 직접 요청을 거의 보내지 않아 요청 비용을 줄일 수 있습니다.

Amazon CloudFront는 지리적으로 분산된 대규모 PoP(Point of Presence) 집합에서 Amazon S3의 데이터를 투명하게 캐시하는 고속 콘텐츠 전송 네트워크(CDN)입니다. 여러 리전 또는 인터넷을 통해 객체에 액세스할 수 있는 경우 CloudFront를 사용하면 객체에 액세스하는 사용자 가까이에서 데이터를 캐싱할 수 있습니다. 이로 인해 인기 있는 Amazon S3 콘텐츠를 고성능으로 전달할 수 있습니다. CloudFront에 대한 자세한 내용은 Amazon CloudFront 개발자 안내서를 참조하십시오.

Amazon ElastiCache는 관리형 인 메모리 캐시입니다. ElastiCache를 사용하면 메모리에 객체를 캐시하는 Amazon EC2 인스턴스를 프로비저닝할 수 있습니다. 이 캐싱은 GET 대기 시간이 크게 감소하고 다운로드 처리량이 상당히 증가합니다. ElastiCache를 사용하려면 캐시를 핫 객체로 채우고 Amazon S3에서 요청하기 전에 캐시에 핫 객체가 있는지 확인하도록 애플리케이션 로직을 수정합니다. ElastiCache를 사용하여 Amazon S3 GET 성능을 향상시키는 예는 블로그 게시물 Amazon ElastiCache for Redis를 사용한 Amazon S3 가속화를 참조하십시오.

AWS Elemental MediaStore는 Amazon S3의 비디오 워크플로와 미디어 전송 전용으로 제작된 캐싱 및 콘텐츠 배포 시스템입니다. MediaStore는 비디오 전용 통합 스토리지 API를 제공하며, 성능에 민감한 비디오 워크로드에 권장됩니다. 미디어 스토어에 대한 자세한 내용은 AWS Elemental MediaStore 사용 설명서를 참조하십시오.

대기 시간에 민감한 애플리케이션의 제한 시간 및 재시도

애플리케이션이 Amazon S3에서 재시도를 해야 한다는 응답을 수신하는 경우가 있습니다. Amazon S3은 버킷 및 객체 이름을 연결된 객체 데이터에 매핑합니다. 애플리케이션의 요청 속도가 높으면(일반적으로 적은 수의 객체에 대해 초당 5,000건 이상의 연속 요청) HTTP 503 slowdown 응답이 수신될 수 있습니다. 이러한 오류가 발생하면 각 AWS SDK는 지수 백오프를 사용하여 자동 재시도 로직을 구현합니다. AWS SDK를 사용하지 않는 경우 HTTP 503 오류 수신 시 재시도 로직을 구현해야 합니다. 자세한 내용은 Amazon Web Services 일반 참조에서 AWS에서의 오류 재시도 횟수 및 지수 백오프를 참조하십시오.

Amazon S3은 새로운 연속 요청 속도에 따라 자동으로 조정되어 동적으로 성능을 최적화합니다. Amazon S3이 새로운 요청 속도에 대해 내부적으로 최적화하는 동안 최적화가 완료될 때까지 일시적으로 HTTP 503 요청 응답을 받게 됩니다. Amazon S3이 새로운 요청 속도에 대해 성능을 내부적으로 최적화한 후에는 일반적으로 모든 요청이 재시도 없이 처리됩니다.

대기 시간에 민감한 애플리케이션의 경우 Amazon S3은 느린 작업을 추적하고 적극적으로 재시도할 것을 권고합니다. 요청을 재시도 할 때 Amazon S3에 대한 새 연결을 사용하고 새로운 DNS 조회를 수행하는 것이 좋습니다.

다양한 크기의 요청(예: 128MB 이상)을 생성할 때, 달성 중인 처리량을 추적하고 요청 중 가장 느린 5%를 재시도하는 것이 좋습니다. 일반적으로 대기 시간 중간값이 수십 밀리초 범위인 작은 요청(예: 512KB 미만)을 생성할 때는 2초 후 GET 또는 PUT 작업을 재시도하는 것이 좋습니다. 추가 재시도가 필요할 경우 가장 좋은 방법은 백오프입니다. 예를 들어 2초 후 재시도하고, 그로부터 4초 후 두 번째로 재시도하는 것이 좋습니다.

애플리케이션이 Amazon S3에 고정 크기 요청을 하는 경우, 각 요청에 대해보다 일관된 응답 시간을 기대할 것입니다. 이 경우 간단한 전략은 요청 중 가장 느린 1%를 확인하고 재시도하는 것입니다. 흔히 한 번의 재시도만으로도 대기 시간을 줄이는 데 효과적입니다.

서버 측 암호화에 AWS Key Management Service(AWS KMS)를 사용하는 경우 사용 사례에 지원되는 요청 속도에 대한 정보는 AWS Key Management Service 개발자 가이드의 할당량을 참조하세요.

높은 처리량을 위한 수평 크기 조정 및 요청 병렬화

Amazon S3은 매우 큰 분산 시스템입니다. 규모를 활용하려면 병렬 요청을 Amazon S3 서비스 엔드포인트까지 수평으로 조정하는 것이 좋습니다. 이 유형의 조정 방법은 Amazon S3 내에서 요청을 분산할 뿐만 아니라, 네트워크를 통해 여러 경로에 걸쳐 부하를 분산하기에도 좋습니다.

높은 처리량 전송의 경우, Amazon S3은 다중 연결을 사용하여 병렬로 데이터를 GET 또는 PUT하는 애플리케이션 사용을 권고합니다. 예를 들어 AWS Java SDK의 Amazon S3 Transfer Manager에서 지원하며, 다른 AWS SDK의 대부분은 비슷한 구조를 제공합니다. 일부 애플리케이션의 경우 서로 다른 애플리케이션 스레드나 다른 애플리케이션 인스턴스에서 동시에 여러 요청을 실행하여 병렬 연결을 구현할 수 있습니다. 취할 수 있는 가장 좋은 방법은 애플리케이션과 액세스 중인 객체의 구조에 따라 다릅니다.

AWS SDK에서 전송 관리를 사용하는 대신 AWS SDK를 사용하여 GET 및 PUT 요청을 직접 실행할 수 있습니다. 이 방법을 사용하면 워크로드를 보다 직접적으로 조정할 수 있으며 SDK의 재시도 지원과 발생할 수 있는 HTTP 503 응답 처리 기능을 계속 활용할 수 있습니다. 일반적으로 리전 내 큰 객체를 Amazon S3에서 Amazon EC2로 다운로드할 때 객체의 바이트 범위에 대한 동시 요청을 8~16MB의 세부 수준으로 수행하는 것이 좋습니다. 원하는 각각의 85~90MB/s의 네트워크 처리량에 대해 한 건의 동시 요청을 하십시오. 10Gb/s 네트워크 인터페이스 카드(NIC)를 포화시키려면 별도의 연결을 통해 약 15건의 동시 요청을 사용할 수 있습니다. 더 많은 연결을 통해 동시 요청을 수직 확장하여 25Gb/s 또는 100Gb/s NIC와 같은 더 빠른 NIC를 포화시킬 수 있습니다.

동시에 생성할 요청 수를 조정할 때 성능 측정이 중요합니다. 한 번에 하나의 요청으로 시작하는 것이 좋습니다. 달성되는 네트워크 대역폭과 애플리케이션이 데이터 처리에 사용하는 다른 리소스의 사용을 측정하십시오. 그런 다음 병목 리소스(즉, 사용률이 가장 높은 리소스), 따라서 유용할 가능성이 있는 요청 수를 식별할 수 있습니다. 예를 들어 한 번에 하나의 요청을 처리하면 CPU 사용량이 25%가되며 최대 4개의 동시 요청을 수용할 수 있습니다.

측정은 필수적이며, 요청 속도가 증가함에 따라 리소스 사용을 확인할 필요가 있습니다.

애플리케이션에서 REST API를 사용하여 Amazon S3에 직접 요청을 제출하는 경우 HTTP 연결 풀을 사용하고 일련의 요청에 대해 각각의 연결을 다시 사용하는 것이 좋습니다. 요청별 연결 설정을 피하면 요청마다 TCP slow-start 및 SSL(Secure Sockets Layer) 핸드셰이크를 수행할 필요가 없습니다. REST API 사용에 대한 자세한 내용은 Amazon S3 REST API 소개를 참조하세요.

마지막으로, DNS에 주의를 기울여 Amazon S3 IP 주소의 광범위한 풀로 요청이 확산되고 있는지 다시 확인하는 것이 중요합니다. DNS는 수많은 IP 엔드포인트 목록을 통해 Amazon S3 주기에 대해 쿼리합니다. 그러나 단일 IP 주소를 재사용하는 캐싱 해석기 또는 애플리케이션 코드는 주소 다양성과 그에 따른 로드 밸런싱의 이점을 얻지 못합니다. `netstat` 명령줄 도구와 같은 네트워크 유틸리티 도구

는 Amazon S3과 통신하는 데 사용되는 IP 주소를 표시할 수 있으며, 사용할 DNS 구성에 대한 가이드를 제공합니다. 이러한 가이드에 대한 자세한 내용은 요청 라우팅을 참조하세요.

Amazon S3 Transfer Acceleration을 사용하여 지리적으로 분산된 데이터 전송 가속화

Amazon S3 Transfer Acceleration은 Amazon S3을 사용하여 전 세계에 분산된 클라이언트와 리전 애플리케이션 간의 물리적 거리로 인해 발생하는 대기 시간을 최소화하거나 없애는 데 효과적입니다. Transfer Acceleration은 데이터 전송을 위해 CloudFront에서 전 세계에 분산된 엣지 로케이션을 사용합니다. AWS 엣지 네트워크는 50개 이상의 로케이션에 존재합니다. 현재는 CloudFront를 통해 콘텐츠를 배포하고 Amazon Route 53에 대해 수행된 DNS 쿼리에 신속하게 응답하는 데 사용됩니다.

또한 엣지 네트워크는 Amazon S3과 주고 받는 데이터 전송을 가속화합니다. 대륙 내부 및 대륙 간에 데이터를 전송하고, 인터넷 연결이 빠르며, 대용량 객체를 사용하거나 업로드할 콘텐츠가 많은 애플리케이션에 이상적입니다. 엣지 로케이션에 도착한 데이터는 최적화된 네트워크 경로를 통해 Amazon S3으로 라우팅됩니다. 일반적으로 Amazon S3 리전에서 멀어질수록 Transfer Acceleration을 사용하면 더 높은 속도 향상을 기대할 수 있습니다.

새 버킷 또는 기존 버킷에 Transfer Acceleration을 설정할 수 있습니다. 별도의 Amazon S3 Transfer Acceleration 엔드포인트를 사용하여 AWS 엣지 로케이션을 사용할 수 있습니다. Transfer Acceleration이 클라이언트 요청 성능에 유익한지 테스트하는 가장 좋은 방법은 Amazon S3 Transfer Acceleration 속도 비교 도구를 사용하는 것입니다. 네트워크 구성 및 조건은 때때로 달라지며 위치에 따라 다릅니다. 따라서 Amazon S3 Transfer Acceleration으로 업로드 성능이 향상될 가능성이 있는 전송에 대해서만 요금이 부과됩니다. 다른 AWS SDK에서 Transfer Acceleration을 사용하는 방법에 대한 자세한 내용은 Amazon S3 Transfer Acceleration 예제 단원을 참조하세요.

기여자

이 문서를 작성하는 데 도움을 주신 분들입니다.

- Mai-Lan Tomsen Bukovec, Amazon S3 VP
- Andy Warfield, Amazon S3 선임 수석 엔지니어
- Tim Harris, Amazon S3 수석 엔지니어

문서 개정

이 백서의 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하세요.

update-history-change	update-history-description	update-history-date
업데이트	기술 정확성을 검토했습니다.	2021년 3월 10일
최초 게시	최초 게시	2019년 6월 1일

고지 사항

고객은 본 문서에 포함된 정보를 독자적으로 평가할 책임이 있습니다. 본 문서는 (a) 정보 제공만을 위한 것이며, (b) 사전 고지 없이 변경될 수 있는 현재의 AWS 제품 제공 서비스 및 사례를 보여 주며, (c) AWS 및 자회사, 공급업체 또는 라이선스 제공자로부터 어떠한 약정 또는 보증도 하지 않습니다. AWS 제품 또는 서비스는 명시적이든 묵시적이든 어떠한 종류의 보증, 진술 또는 조건 없이 '있는 그대로' 제공됩니다. 고객에 대한 AWS의 책임과 법적 책임은 AWS 계약서에 준하며 본 문서는 AWS와 고객 간의 계약에 포함되지 않고 계약을 변경하지도 않습니다.

© 2020 Amazon Web Services, Inc. 또는 자회사. All rights reserved.