



AWS 백서

AWS 기반 WordPress 모범 사례



AWS 기반 WordPress 모범 사례: AWS 백서

Copyright © 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

요약	1
요약	1
소개	2
간단한 배포	3
고려 사항	3
사용할 수 있는 방법	3
Amazon Lightsail	4
Amazon Lightsail 요금제 선택	4
WordPress 설치	4
장애 복구	5
성능 및 비용 효율성 향상	6
콘텐츠 전송 가속화	6
정적 콘텐츠 오프로드	7
동적 콘텐츠	7
데이터베이스 캐싱	8
바이트코드 캐싱	9
탄력적 배포	10
참조 아키텍처	10
웹 티어 확장	12
무상태 웹 티어	13
공유 스토리지(Amazon S3 및 Amazon EFS)	14
데이터 계층(Amazon Aurora 및 Amazon ElastiCache)	14
AWS 기반 WordPress High Availability by Bitnami 빠른 시작	15
결론	18
기여자	19
문서 개정	20
부록 A: CloudFront 구성	21
오리진 및 동작	21
CloudFront 배포 생성	21
부록 B: 플러그 인 설치 및 구성	25
AWS for WordPress 플러그 인	25
플러그 인 설치 및 구성	25
Amazon CloudFront 및 AWS Certificate Manager	27
콘텐츠 번역 및 발성	27

Amazon Pollycast로 팟캐스트	29
Amazon Alexa 디바이스를 통해 콘텐츠 읽기	30
정적 콘텐츠 구성	31
IAM 사용자 생성	31
Amazon S3 버킷 생성	31
정적 오리진 생성	33
부록 C: 백업 및 복구	34
부록 D: 새 플러그인 및 테마 배포	36
고지 사항	37

AWS 기반 WordPress 모범 사례

게시 날짜: 2021년 10월 19일([문서 개정](#))

요약

이 백서는 시스템 관리자에게 Amazon Web Services(AWS)에서 WordPress를 시작하는 방법과 배포의 비용 효율성 및 최종 사용자 환경을 개선하는 방법에 대한 구체적인 지침을 제공합니다. 또한 일반적인 확장성 및고가용성 요구 사항을 충족하는 데 필요한 참조 아키텍처도 개략적으로 설명합니다.

소개

WordPress는 PHP와 MySQL로 작성된 오픈 소스 블로그 도구 및 콘텐츠 관리 시스템(CMS)으로서, 개인 블로그에서 트래픽이 많은 웹 사이트까지 다양한 사이트를 지원하는 데 사용됩니다.

최초 버전의 WordPress가 2003년 릴리스되었을 때는 지금과 같은 탄력적이고 확장 가능한 클라우드 기반 인프라를 전제로 구축되지 않았습니다. WordPress 커뮤니티의 노력과 다양한 WordPress 모듈 릴리스를 통해 이러한 CMS 솔루션의 기능이 지속적으로 확장되고 있습니다. 이제 AWS 클라우드의 수많은 장점을 활용하는 WordPress 아키텍처를 구축할 수 있게 되었습니다.

간단한 배포

트래픽이 적은 블로그 또는 엄격한고가용성 요구 사항이 없는 웹 사이트의 경우 간단한 단일 서버 배포가 적합할 수 있습니다. 이 배포는 복원력 또는 확장성이 가장 높은 아키텍처는 아니지만 웹 사이트를 시작하고 실행하는 가장 빠르고 경제적인 방법입니다.

주제

- [고려 사항](#)
- [사용할 수 있는 방법](#)
- [Amazon Lightsail](#)

고려 사항

이 논의는 단일 웹 서버 배포에서 시작합니다. 때로는 웹 사이트가 단일 웹 서버 배포 규모를 넘어설 수 있습니다. 예를 들면 다음과 같습니다.

- WordPress 웹 사이트가 배포되는 가상 머신은 단일 실패 지점입니다. 이 인스턴스에 문제가 발생하면 웹 사이트의 서비스가 손실됩니다.
- 성능 향상을 위해 리소스를 확장하는 것은 '수직 확장'을 통해, 즉 WordPress 웹 사이트를 실행하는 가상 머신의 크기를 늘리는 방법으로만 달성할 수 있습니다.

사용할 수 있는 방법

AWS에는 가상 머신을 프로비저닝하기 위한 다양한 옵션이 있습니다. AWS에서 자체 WordPress 웹 사이트를 호스팅하는 세 가지 주요 방법이 있습니다.

- Amazon Lightsail
- Amazon Elastic Compute Cloud(Amazon EC2)
- AWS Marketplace

[Amazon Lightsail](#)은 가상 프라이빗 서버(Lightsail 인스턴스)를 빠르게 시작하여 WordPress 웹 사이트를 호스팅할 수 있는 서비스입니다. Lightsail은 고도로 구성 가능한 인스턴스 유형 또는 고급 네트워킹 기능이 필요하지 않은 경우 가장 쉽게 시작할 수 있는 방법입니다.

[Amazon EC2](#)는 크기 조정이 가능한 컴퓨팅 용량을 제공하는 웹 서비스로, 몇 분 만에 가상 서버를 시작할 수 있습니다. Amazon EC2는 Lightsail보다 많은 구성 및 관리 옵션을 제공하므로 고급 아키텍처에서 바람직합니다. 사용자는 EC2 인스턴스에 대한 관리 액세스 권한이 있으며 WordPress를 비롯해 선택한 모든 소프트웨어 패키지를 설치할 수 있습니다.

[AWS Marketplace](#)는 AWS에서 실행되는 소프트웨어를 검색, 구매 및 빠르게 배포할 수 있는 온라인 스토어입니다. 1-Click 배포를 사용하면 단 몇 분 만에 사전 구성된 WordPress 이미지를 자신의 AWS 계정에 있는 Amazon EC2로 시작할 수 있습니다. 즉각 실행할 수 있는 WordPress 인스턴스를 제공하는 Marketplace 공급 업체가 많이 있습니다.

이 백서에서는 단일 서버 WordPress 웹 사이트에 권장되는 구현인 Lightsail 옵션에 대해 설명합니다.

Amazon Lightsail

Lightsail은 간단한 가상 프라이빗 서버(VPS) 솔루션이 필요한 개발자, 소기업, 학생 등의 사용자가 AWS를 시작할 수 있는 가장 쉬운 방법입니다.

이 서비스는 복잡한 인프라 관리 요소를 상당 부분 사용자로부터 추상화합니다. 따라서 인프라 경험이 적은 경우 또는 웹 사이트 운영에 집중해야 하고 간단한 제품으로도 충분한 경우에 이상적인 출발점입니다.

Amazon Lightsail을 사용하면 Windows 또는 Linux/Unix 운영 체제와 WordPress를 비롯한 인기 있는 웹 애플리케이션을 선택하고 사전 구성된 템플릿에서 클릭 한 번으로 배포할 수 있습니다.

요구 사항이 증가하면 원활하게 초기 경계를 확장하여 추가 AWS 데이터베이스, 객체 스토리지, 캐싱 및 콘텐츠 배포 서비스에 연결할 수 있습니다.

Amazon Lightsail 요금제 선택

[Lightsail 요금제](#)는 WordPress 웹 사이트를 호스트하는 데 사용하는 Lightsail 리소스의 월간 비용을 정의합니다. 다양한 수준의 CPU 리소스, 메모리, 솔리드 스테이트 드라이브(SSD) 스토리지, 데이터 전송과 같은 다양한 사용 사례를 포괄할 수 있는 여러 가지 요금제가 있습니다. 웹 사이트가 복잡한 경우 더 많은 리소스를 제공하는 더 큰 인스턴스가 필요할 수 있습니다. 이를 위해 [웹 콘솔을 사용](#)하거나 [Amazon Lightsail CLI 설명서](#)에 설명된 대로 서버를 상위 요금제로 마이그레이션할 수 있습니다.

WordPress 설치

Lightsail은 WordPress와 같이 일반적으로 사용되는 애플리케이션을 위한 템플릿을 제공합니다. 이 템플릿은 필요한 소프트웨어가 대부분 사전 설치되어 제공되므로 자체 WordPress 웹 사이트를 실행하기

위한 좋은 출발점입니다. 브라우저 내 터미널 또는 자체 SSH 클라이언트를 사용하거나 WordPress 관리 웹 인터페이스를 통해 추가 소프트웨어를 설치하거나 소프트웨어 구성을 사용자 지정할 수 있습니다.

Amazon Lightsail은 GoDaddy Pro 사이트 제품과 제휴하여 WordPress 고객이 무료로 인스턴스를 쉽게 관리할 수 있도록 지원하고 있습니다. Lightsail WordPress 가상 서버는 빠른 성능과 보안을 위해 미리 구성되고 최적화되어 있으므로, WordPress 사이트를 즉시 시작하고 실행할 수 있습니다. WordPress 인스턴스를 여러 개 실행하는 고객은 운영 중인 모든 사이트를 업데이트, 유지 및 관리하려면 까다롭고 시간이 많이 든다는 것을 알고 있습니다. 이 통합을 통해 몇 번의 클릭만으로 몇 분 안에 여러 WordPress 인스턴스를 쉽게 관리할 수 있습니다.

Lightsail에서 WordPress를 관리하는 방법에 대한 자세한 내용은 [Amazon Lightsail 인스턴스에서 WordPress 사용 시작하기](#)를 참조하십시오. WordPress 웹 사이트를 사용자 지정했으면 인스턴스의 스냅샷을 생성하는 것이 좋습니다.

스냅샷은 Lightsail 인스턴스의 백업 이미지를 생성하는 방법입니다. 이 스냅샷은 시스템 디스크의 복사본으로, 원래 시스템 구성(메모리, CPU, 디스크 크기 및 디스크 전송 속도)도 저장합니다. 스냅샷은 잘못된 배포 또는 업그레이드 후 정상 작동이 확인된 구성으로 되돌아가는 데 사용할 수 있습니다.

이 스냅샷을 사용하면 필요한 경우 서버를 복구할 수 있을 뿐만 아니라 동일한 사용자 지정으로 새 인스턴스를 시작할 수도 있습니다.

장애 복구

단일 웹 서버는 단일 장애 지점이므로 웹 사이트 데이터가 백업되도록 해야 합니다. 앞서 설명한 스냅샷 메커니즘을 이 용도로도 사용할 수 있습니다. 장애를 복구하기 위해 최신 스냅샷에서 새 인스턴스를 복원할 수 있습니다. 복원 중에 손실될 수 있는 데이터의 양을 줄이려면 스냅샷이 가능한 한 최신 상태여야 합니다.

데이터 손실 가능성을 최소화하기 위해 스냅샷을 정기적으로 생성해야 합니다. Lightsail Linux/Unix 인스턴스의 자동 스냅샷을 예약할 수 있습니다. 단계는 [Amazon Lightsail에서 인스턴스 또는 디스크의 자동 스냅샷 활성화 또는 비활성화](#)를 참조하십시오.

고정 IP, 즉 Lightsail 계정 전용 고정 퍼블릭 IP 주소를 사용하는 것이 좋습니다. 인스턴스를 다른 인스턴스로 바꿔야 하는 경우 고정 IP를 새 인스턴스에 다시 할당할 수 있습니다. 그러면 인스턴스를 바꾸고 싶을 때마다 새 IP 주소를 지정하려고 외부 시스템(예: DNS 레코드)을 다시 구성할 필요가 없습니다.

성능 및 비용 효율성 향상

웹 사이트가 결국 단일 서버 배포 규모를 넘어설 수 있습니다. 이 경우 웹 사이트 성능을 개선하기 위한 옵션을 고려해야 할 수 있습니다. 확장 가능한 다중 서버 배포(이 백서의 뒷부분에서 설명)로 마이그레이션하기 전에 적용할 수 있는 여러 가지 성능 및 비용 효율성이 있습니다. 이러한 방법은 다중 서버 아키텍처로 이동하더라도 반드시 따라야 할 모범 사례입니다.

다음 섹션에서는 WordPress 웹 사이트의 성능 및 확장성을 향상할 수 있는 여러 옵션을 소개합니다. 일부 옵션은 단일 서버 배포에 적용할 수 있는 반면, 다른 옵션은 여러 서버의 확장성을 활용합니다. 이러한 수정은 대부분 하나 이상의 WordPress 플러그 인을 사용해야 합니다. 다양한 옵션을 사용할 수 있지만 [W3 Total Cache](#)는 이러한 수정 사항 중 많은 부분을 단일 플러그 인에 결합하여 널리 사용됩니다.

주제

- [콘텐츠 전송 가속화](#)
- [데이터베이스 캐싱](#)
- [바이트코드 캐싱](#)

콘텐츠 전송 가속화

모든 WordPress 웹 사이트는 정적 콘텐츠와 동적 콘텐츠를 혼합하여 제공해야 합니다. 정적 콘텐츠에는 이미지, JavaScript 파일 또는 스타일 시트가 포함됩니다. 동적 콘텐츠에는 WordPress PHP 코드를 사용하여 서버 측에서 생성된 모든 항목이 포함됩니다(예: 데이터베이스에서 생성되거나 각 최종 사용자에게 맞게 개인화된 사이트 요소).

최종 사용자 경험에서 한 가지 중요한 측면은 위에 언급한 콘텐츠를 전 세계 사용자에게 전송할 때 발생하는 네트워크 대기 시간입니다. 콘텐츠를 전송을 가속화하면 최종 사용자, 특히 전 세계에 지리적으로 분산된 사용자의 경험이 향상됩니다. 이는 Amazon CloudFront와 같은 콘텐츠 전송 네트워크(CDN)를 통해 달성할 수 있습니다.

[Amazon CloudFront](#)는 전 세계 여러 엣지 로케이션을 통해 짧은 대기 시간과 빠른 데이터 전송 속도로 콘텐츠를 배포할 수 있는 간편하고 비용 효율적인 방법을 제공하는 웹 서비스입니다. 최종 사용자 요청은 대기 시간을 줄이기 위해 자동으로 적합한 CloudFront [엣지 로케이션](#)으로 라우팅됩니다. 콘텐츠가 캐시될 수 있고(몇 초, 몇 분 또는 며칠 동안) 특정 엣지 로케이션에 이미 저장되어 있는 경우 CloudFront가 콘텐츠를 즉시 전송합니다. 콘텐츠를 캐시하지 않아야 하거나, 만료되었거나, 현재 해당 엣지 로케이션에 없는 경우 CloudFront는 CloudFront 구성에서 오리진(이 경우 Lightsail 인스턴스)이라

고 하는 하나 이상의 신뢰할 수 있는 원본에서 콘텐츠를 검색합니다. 이 검색은 최적화된 네트워크 연결을 통해 이루어지므로 웹 사이트의 콘텐츠 전송 속도를 높일 수 있습니다. 이 모델은 최종 사용자 경험을 개선하는 것 외에도 원본 서버의 로드를 줄이고 상당한 비용 절감을 가져올 수 있습니다.

정적 콘텐츠 오프로드

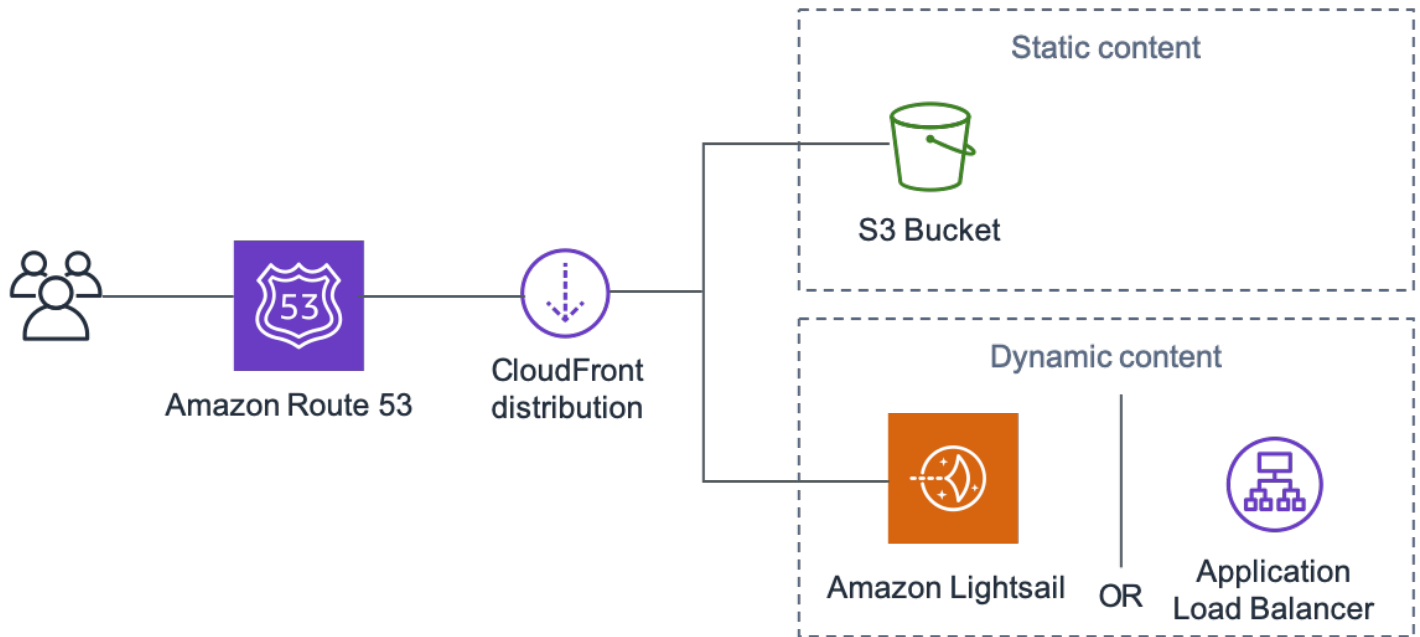
여기에는 CSS, JavaScript 및 이미지 파일이 포함됩니다(W WordPress 테마에 속하는 파일 또는 콘텐츠 관리자가 업로드한 미디어 파일). 이러한 파일은 모두 W3 Total Cache와 같은 플러그인을 사용하여 Amazon Simple Storage Service(Amazon S3)에 저장할 수 있으며 확장 가능하고 가용성이 뛰어난 방식으로 사용자에게 제공할 수 있습니다. [Amazon S3](#)는 확장성이 뛰어나고 안정적이며 대기 시간이 짧은 데이터 스토리지 인프라를 저렴한 비용으로 제공하며 REST API를 통해 액세스할 수 있습니다. Amazon S3는 객체를 여러 디바이스뿐만 아니라 AWS 리전의 여러 시설에 중복 저장하므로 고도의 내 구성을 제공합니다.

그러므로 이 워크로드를 Lightsail 인스턴스에서 오프로드하고 동적 콘텐츠 생성에 집중할 수 있다는 긍정적인 효과가 있습니다. 이는 서버의 로드를 줄여주며 무상태 아키텍처를 구축하는 중요한 단계입니다(자동 크기 조정을 구현하기 위한 사전 조건).

이후 Amazon S3를 CloudFront의 오리진으로 구성하여 이러한 정적 자산을 전 세계 사용자에게 더 효율적으로 전송할 수 있습니다. WordPress는 Amazon S3 및 CloudFront와 바로 통합되지는 않지만 다양한 플러그인이 이러한 서비스에 대한 지원을 제공합니다(예: W3 Total Cache).

동적 콘텐츠

동적 콘텐츠에는 서버 측 WordPress PHP 스크립트의 출력이 포함됩니다. WordPress 웹 사이트를 오리진으로 구성하여 CloudFront를 통해 동적 콘텐츠를 제공할 수도 있습니다. 동적 콘텐츠에는 개인화된 콘텐츠가 포함되므로 특정 HTTP 쿠키 및 HTTP 헤더를 요청의 일부로 사용자 지정 원본 서버에 전달하도록 CloudFront를 구성해야 합니다. CloudFront는 전달된 쿠키 값을 캐시에서 고유한 객체를 식별하는 키의 일부로 사용합니다. 캐싱 효율성을 극대화하려면 콘텐츠가 실제로 달라지는 HTTP 쿠키 및 HTTP 헤더만 전달하도록 CloudFront를 구성해야 합니다(예를 들어 웹 분석을 위해 클라이언트 측 또는 서드 파티 애플리케이션에서만 사용되는 쿠키는 아님).



Amazon CloudFront를 통한 전체 웹 사이트 전송

위 그림에는 두 가지 오리진이 포함되어 있습니다. 하나는 정적 콘텐츠용이고 다른 하나는 동적 콘텐츠용입니다. 구현에 대한 자세한 내용은 [부록 A: CloudFront 구성](#) 및 [부록 B: 플러그인 설치 및 구성](#)을 참조하십시오.

CloudFront는 표준 캐시 제어 헤더를 사용하여 특정 HTTP 응답을 캐시해야 하는지 여부 및 기간을 식별합니다. 웹 브라우저도 동일한 캐시 제어 헤더를 사용하여 보다 최적의 최종 사용자 환경을 위해 콘텐츠를 로컬로 캐시할 시기 및 기간을 결정합니다(예: 이미 다운로드된 .css 파일은 재방문자가 페이지를 볼 때마다 다시 다운로드되지 않음). 웹 서버 수준에서 캐시 제어 헤더를 구성하거나(예: .htaccess 파일 또는 httpd.conf 파일 수정을 통해) WordPress 플러그인(예: W3 Total Cache)을 설치하여 정적 및 동적 콘텐츠에 대해 이러한 헤더가 설정되는 방식을 지정할 수 있습니다.

데이터베이스 캐싱

데이터베이스 캐싱은 WordPress 같이 읽기 중심 애플리케이션 워크로드에 대한 대기 시간을 크게 줄이고 처리량을 늘릴 수 있습니다. 대기 시간이 짧은 액세스를 위해 자주 액세스하는 데이터를 메모리에 저장함으로써 애플리케이션 성능이 향상됩니다(예: I/O 집약적인 데이터베이스 쿼리의 결과). 캐시에서 결과가 제공되는 쿼리의 비율이 높으면 데이터베이스에 도달해야 하는 쿼리 수가 줄어들어 데이터베이스 실행과 관련된 비용이 절감됩니다.

WordPress에는 기본적으로 제한된 캐싱 기능이 있지만, 다양한 플러그인이 널리 채택된 메모리 객체 캐싱 시스템인 [Memcached](#)와의 통합을 지원합니다. W3 Total Cache 플러그인이 좋은 예입니다.

가장 간단한 시나리오에서는 웹 서버에 Memcached를 설치하고 그 결과를 새 스냅샷으로 캡처합니다. 이 경우 사용자가 캐시 실행과 관련된 관리 작업을 책임집니다.

또 다른 옵션은 [Amazon ElastiCache](#)와 같은 관리형 서비스를 활용하여 운영 부담을 피하는 것입니다. ElastiCache를 사용하면 클라우드에서 분산형 인 메모리 캐시를 더욱 간편하게 배포, 운영 및 조정할 수 있습니다. ElastiCache 클러스터 노드에 연결하는 방법에 대한 자세한 내용은 [Amazon ElastiCache 설명서](#)를 참조하십시오.

Lightsail을 사용 중이고 AWS 계정의 ElastiCache 클러스터에 비공개로 액세스하려는 경우 VPC 피어링을 사용하면 됩니다. VPC 피어링을 활성화하는 지침은 [Amazon Lightsail 외부의 AWS 리소스와 함께 작동하도록 Amazon VPC 피어링 설정](#)을 참조하십시오.

바이트코드 캐싱

PHP 스크립트는 실행될 때마다 구문 분석되고 컴파일됩니다. PHP 바이트코드 캐시를 사용하면 PHP 컴파일의 출력이 RAM에 저장되므로 동일한 스크립트를 반복해서 컴파일할 필요가 없습니다. 그러면 PHP 스크립트 실행과 관련된 오버헤드가 줄어들어 성능이 향상되고 CPU 요구 사항이 낮아집니다.

바이트코드 캐시는 WordPress를 호스트하는 모든 Lightsail 인스턴스에 설치할 수 있으며 로드를 크게 줄일 수 있습니다. PHP 5.5 이상에서는 해당 PHP 버전과 함께 번들로 제공되는 확장인 [OpCache](#)를 사용할 것을 권장합니다.

OpCache는 Bitnami WordPress Lightsail 템플릿에서 기본적으로 활성화되어 있으므로 추가 작업이 필요하지 않습니다.

탄력적 배포

웹 사이트에 단일 서버 배포가 충분하지 않은 시나리오는 많습니다. 이러한 상황에서는 확장 가능한 다중 서버 아키텍처가 필요합니다.

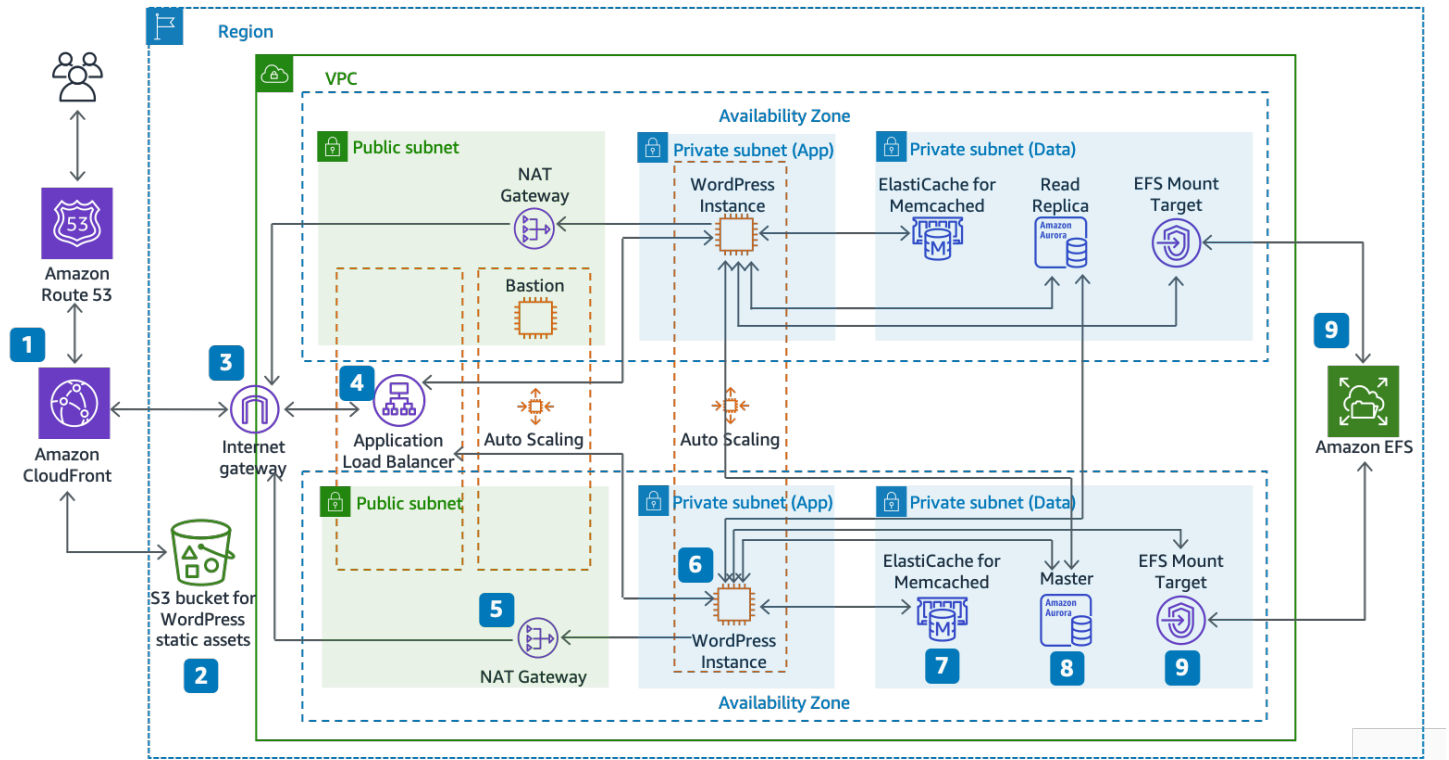
주제

- [참조 아키텍처](#)
- [웹 티어 확장](#)
- [무상태 웹 티어](#)
- [AWS 기반 WordPress High Availability by Bitnami 빠른 시작](#)

참조 아키텍처

GitHub에서 사용할 수 있는 [AWS 기반 WordPress 호스팅 참조 아키텍처](#)는 AWS에 WordPress를 배포하는 모범 사례를 간략하게 설명하며, 빠르게 시작하고 실행할 수 있는 일련의 AWS CloudFormation 템플릿을 포함합니다. 다음 아키텍처는 이 참조 아키텍처를 기반으로 합니다. 이 섹션의 나머지 부분에서는 아키텍처 선택 이유를 검토합니다.

GitHub의 기반 AMI는 2021년 7월에 Amazon Linux1에서 Amazon Linux2로 변경되었습니다. 하지만 S3의 배포 템플릿은 아직 변경되지 않았습니다. S3의 템플릿을 사용하여 참조 아키텍처를 배포하는데 문제가 있는 경우 GitHub의 템플릿을 사용하는 것이 좋습니다.



AWS 기반 WordPress 호스팅을 위한 참조 아키텍처

아키텍처 구성 요소

참조 아키텍처는 AWS 기반 WordPress 웹 사이트의 전체 모범 사례 배포를 보여줍니다.

- 이 아키텍처는 최종 사용자와 가까운 곳에서 콘텐츠를 캐시하여 전송 속도를 높이기 위해 Amazon CloudFront(1)의 엣지 캐싱부터 시작합니다.
- CloudFront는 S3 버킷(2)에서 정적 콘텐츠를 가져오고 웹 인스턴스 앞에 있는 Application Load Balancer(4)에서 동적 콘텐츠를 가져옵니다.
- 웹 인스턴스는 Amazon EC2 인스턴스(6)의 Auto Scaling 그룹에서 실행됩니다.
- ElastiCache 클러스터(7)는 응답 속도를 높이기 위해 자주 쿼리되는 데이터를 캐시합니다.

Amazon Aurora MySQL 인스턴스(8)는 WordPress 데이터베이스를 호스트합니다.

- WordPress EC2 인스턴스는 각 가용 영역의 EFS 탑재 대상(9)을 통해 Amazon EFS 파일 시스템의 공유 WordPress 데이터에 액세스합니다.
- 인터넷 게이트웨이는 VPC의 리소스와 인터넷 간의 통신을 가능하게 합니다.
- 각 가용 영역의 NAT 게이트웨이(5)는 프라이빗 서브넷(앱 및 데이터)의 EC2 인스턴스가 인터넷에 액세스할 수 있게 합니다.

Amazon VPC에는 퍼블릭(퍼블릭 서브넷) 및 프라이빗(앱 서브넷 및 데이터 서브넷) 두 가지 유형의 서브넷이 있습니다. 퍼블릭 서브넷에 배포된 리소스는 퍼블릭 IP 주소를 받으며 인터넷에서 공개적으로 볼 수 있습니다. Application Load Balancer(4)와 관리용 바스천 호스트가 여기에 배포됩니다. 프라이빗 서브넷에 배포된 리소스는 프라이빗 IP 주소만 받으므로 인터넷에 공개적으로 표시되지 않아 이러한 리소스의 보안이 향상됩니다. WordPress 웹 서버 인스턴스(6), ElastiCache 클러스터 인스턴스(7), Aurora MySQL 데이터베이스 인스턴스(8) 및 EFS 탑재 대상(9)은 모두 프라이빗 서브넷에 배포됩니다.

이 섹션의 나머지 부분에서는 각 고려 사항에 대해 자세히 설명합니다.

웹 티어 확장

단일 서버 아키텍처를 확장 가능한 다중 서버 아키텍처로 발전시키려면 5가지 주요 구성 요소를 사용해야 합니다.

- Amazon EC2 인스턴스
- Amazon Machine Image(AMI)
- 로드 밸런서
- 자동 크기 조정
- 상태 확인

AWS는 다양한 EC2 인스턴스 유형을 제공하므로 성능과 비용 모두에 가장 적합한 서버 구성을 선택할 수 있습니다. 일반적으로 컴퓨팅 최적화(예: C4) 인스턴스 유형이 WordPress 웹 서버에 적합한 선택이 될 수 있습니다. AWS 리전 내의 여러 가용 영역에 인스턴스를 배포하여 전체 아키텍처의 안정성을 높일 수 있습니다.

EC2 인스턴스를 완벽하게 제어할 수 있으므로 루트 액세스 권한으로 로그인하여 WordPress 웹 사이트를 실행하는 데 필요한 모든 소프트웨어 구성 요소를 설치하고 구성할 수 있습니다. 작업을 완료한 후 해당 구성을 AMI로 저장할 수 있으며, 이를 사용하여 모든 사용자 지정을 포함하는 새 인스턴스를 시작할 수 있습니다.

최종 사용자 요청을 여러 웹 서버 노드에 배포하려면 로드 밸런싱 솔루션이 필요합니다. AWS는 트래픽을 여러 EC2 인스턴스로 분산하는 고가용성 서비스인 [Elastic Load Balancing](#)을 통해 이러한 기능을 제공합니다. 웹 사이트는 HTTP 또는 HTTPS를 통해 사용자에게 콘텐츠를 제공하기 때문에 콘텐츠 라우팅 및 필요한 경우 여러 도메인에서 여러 WordPress 웹 사이트를 실행할 수 있는 기능을 갖춘 애플리케이션 계층 로드 밸런서인 Application Load Balancer를 사용하는 것이 좋습니다.

Elastic Load Balancing은 AWS 리전 내의 여러 가용 영역에 걸쳐 요청을 분산할 수 있도록 지원합니다. Application Load Balancer가 장애(예: 하드웨어 문제 또는 소프트웨어 충돌)가 발생한 개별 인스턴스로의 트래픽 전송을 자동으로 중지하도록 상태 확인을 구성할 수도 있습니다. 상태 확인에는 WordPress 관리자 로그인 페이지(/wp-login.php)를 사용하는 것이 좋습니다. 이 페이지는 웹 서버가 실행 중인지, 웹 서버가 PHP 파일을 올바르게 제공하도록 구성되어 있는지 모두 확인하기 때문입니다.

데이터베이스 및 캐시 리소스와 같은 다른 종속 리소스를 검사하는 사용자 지정 상태 확인 페이지를 작성할 수 있습니다. 자세한 내용은 Application Load Balancer 안내서에서 [대상 그룹에 대한 상태 확인](#)을 참조하십시오.

탄력성은 AWS 클라우드의 주요 특징입니다. 필요할 때 더 많은 컴퓨팅 용량(예: 웹 서버)을 시작하고 필요하지 않을 때는 더 적게 실행할 수 있습니다. [AWS Auto Scaling](#)은 이 프로비저닝을 자동화하여 수동 개입 없이 정의한 조건에 따라 Amazon EC2 용량을 확장 또는 축소할 수 있도록 지원하는 AWS 서비스입니다. 수요가 급증하면 사용하는 EC2 인스턴스 수를 원활히 늘려 성능을 유지하고 트래픽이 감소하면 자동으로 줄여 비용을 최소화하도록 AWS Auto Scaling를 구성할 수 있습니다.

또한 Elastic Load Balancing은 로드 밸런싱 회전에서 Amazon EC2 호스트를 동적으로 추가 및 제거하는 기능을 지원합니다. 또한 Elastic Load Balancing 자체는 로드 밸런싱 용량을 동적으로 늘리거나 줄여 수동 개입 없이 트래픽 수요에 맞게 조정합니다.

무상태 웹 티어

자동 크기 조정 구성에서 여러 웹 서버를 활용하려면 웹 티어가 무상태여야 합니다. 무상태 애플리케이션은 이전 상호 작용에 대한 정보가 필요하지 않고 세션 정보를 저장하지 않는 애플리케이션입니다. WordPress의 경우 이는 어떤 웹 서버가 요청을 처리했는지 관계없이 모든 최종 사용자가 동일한 응답을 받는다는 것을 의미합니다. 사용 가능한 컴퓨팅 리소스(즉, 웹 서버 인스턴스)에서 모든 요청을 처리할 수 있으므로 무상태 애플리케이션은 수평으로 확장할 수 있습니다. 이 용량이 더 이상 필요하지 않으면 개별 리소스를 안전하게 종료할 수 있습니다(실행 중인 작업이 고갈된 후). 이러한 리소스는 피어의 존재를 인식할 필요가 없습니다. 필요한 것은 워크로드를 각 피어로 분산하는 방법뿐입니다.

사용자 세션 데이터 스토리지와 관련하여 WordPress 코어는 클라이언트의 웹 브라우저에 저장된 쿠키에 의존하기 때문에 완전히 무상태입니다. 네이티브 PHP 세션에 의존하는 사용자 지정 코드(예: WordPress 플러그 인)를 설치하지 않았다면 세션 스토리지는 문제가 되지 않습니다.

그러나 WordPress는 원래 단일 서버에서 실행되도록 설계되었습니다. 따라서 서버의 로컬 파일 시스템에 일부 데이터를 저장합니다. 다중 서버 구성에서 WordPress를 실행할 때 웹 서버 간의 불일치 때문에 문제가 발생합니다. 예를 들어 사용자가 새 이미지를 업로드하는 경우 해당 이미지는 서버 중 하나에만 저장됩니다.

이는 중요한 데이터를 공유 스토리지로 이동하도록 기본 WordPress 실행 구성을 개선해야 하는 이유를 잘 보여줍니다. 모범 사례 아키텍처는 데이터베이스를 웹 서버 외부의 별도 계층으로 사용하며 공유 스토리지를 사용하여 사용자 업로드, 테마 및 플러그 인을 저장합니다.

공유 스토리지(Amazon S3 및 Amazon EFS)

기본적으로 WordPress는 사용자 업로드를 로컬 파일 시스템에 저장하므로 무상태가 아닙니다. 따라서 WordPress 설치 및 모든 사용자 지정(예: 구성, 플러그 인, 테마 및 사용자 생성 업로드)을 공유 데이터 플랫폼으로 이동하여 웹 서버의 로드를 줄이고 웹 티어를 무상태로 만들어야 합니다.

[Amazon Elastic File System](#)(Amazon EFS)은 EC2 인스턴스에서 사용할 수 있는 확장 가능한 네트워크 파일 시스템을 제공합니다. Amazon EFS 파일 시스템은 스토리지 서버에 분산되어 있는데 스토리지 수에 제약이 없기 때문에 파일 시스템이 탄력적으로 확장 가능하고 EC2 인스턴스에서 대량 병렬 액세스가 가능합니다. Amazon EFS의 분산 설계로 기존 파일 서버에 내재된 병목과 제약을 피할 수 있습니다.

전체 WordPress 설치 디렉터리를 EFS 파일 시스템으로 이동하고 부팅 시 각 EC2 인스턴스에 탑재하면, WordPress 사이트 및 모든 데이터가 한 EC2 인스턴스에 종속되지 않는 분산 파일 시스템에 자동으로 저장되므로 웹 티어가 완벽하게 무상태로 됩니다. 이 아키텍처의 이점은 새 인스턴스를 시작할 때마다 플러그 인 및 테마를 설치할 필요가 없으며 WordPress 인스턴스의 설치 및 복구 속도를 크게 높일 수 있다는 것입니다. 또한 이 문서의 [배포 고려 사항](#) 섹션에 설명된 대로 WordPress의 플러그 인 및 테마에 대한 변경 사항을 배포하기가 더 쉽습니다.

EFS 파일 시스템에서 웹 사이트를 실행할 때 성능을 최적화하려면 [WordPress용 AWS 참조 아키텍처](#)에서 Amazon EFS 및 OPcache에 대한 권장 구성 설정을 확인하십시오.

또한 이미지, CSS, JavaScript 파일과 같은 모든 정적 자산을 CloudFront 캐싱이 적용된 S3 버킷으로 오프로드할 수도 있습니다. 다중 서버 아키텍처에서 이 작업을 수행하는 메커니즘은 이 백서의 [정적 콘텐츠](#) 섹션에서 설명한 것처럼 단일 서버 아키텍처의 경우와 완전히 동일합니다. 이점도 단일 서버 아키텍처와 동일합니다. 정적 자산 제공과 관련된 작업을 Amazon S3 및 CloudFront로 오프로드할 수 있으므로 웹 서버는 동적 콘텐츠 생성에만 집중하고 웹 서버당 더 많은 사용자 요청을 처리할 수 있습니다.

데이터 계층(Amazon Aurora 및 Amazon ElastiCache)

WordPress 설치가 분산되고 확장 가능한 공유 네트워크 파일 시스템에 저장되고 정적 자산이 Amazon S3에서 제공되면 나머지 상태 유지 구성 요소인 데이터베이스에 집중할 수 있습니다. 스토리지 티어와 마찬가지로 데이터베이스는 단일 서버에 의존해서는 안 되므로 웹 서버 중 하나에서 호스트할 수 없습니다. 대신 WordPress 데이터베이스를 Amazon Aurora에서 호스트합니다.

[Amazon Aurora](#)는 고성능 상용 데이터베이스의 성능과 가용성에 오픈 소스 데이터베이스의 간편성과 비용 효율성을 결합하였으며 클라우드를 위해 구축된 MySQL 및 PostgreSQL 호환 관계형 데이터베이스입니다. Aurora MySQL은 데이터베이스 엔진을 SSD 지원 목적별 분산 스토리지 시스템과 긴밀하게 통합하여 MySQL의 성능 및 가용성을 향상합니다. 내결함성 및 자동 복구 기능이 있고, 3개의 가용 영역에 데이터 사본 6개를 복제하고, 99.99% 이상의 가용성을 제공하도록 설계되었으며, Amazon S3에 데이터를 지속적으로 백업합니다. Amazon Aurora는 장애를 복구하거나 데이터베이스 캐시를 다시 구축할 필요 없이 데이터베이스 장애를 자동 감지하고 재시작하도록 설계되었습니다.

Amazon Aurora는 메모리 최적화 및 버스트 가능한 인스턴스를 비롯하여 다양한 애플리케이션 프로필에 적합한 여러 [인스턴스 유형](#)을 제공합니다. 데이터베이스 성능을 향상하기 위해 대형 인스턴스 유형을 선택하여 더 많은 CPU 및 메모리 리소스를 제공할 수 있습니다.

Amazon Aurora는 기본 인스턴스와 [Aurora 복제본](#) 간의 장애 조치를 자동으로 처리하므로 관리자의 수동 개입 없이도 애플리케이션이 최대한 빨리 데이터베이스 작업을 재개할 수 있습니다. 장애 조치는 일반적으로 30초 이내에 처리됩니다.

Aurora 복제본을 하나 이상 생성했으면 클러스터 엔드포인트를 사용하여 기본 인스턴스에 연결하여 기본 인스턴스에 장애가 발생할 경우 애플리케이션이 자동으로 장애 조치하도록 합니다. 대기 시간이 짧은 읽기 전용 복제본을 3개의 가용 영역에 최대 15개까지 생성할 수 있습니다.

데이터베이스가 확장되면 데이터베이스 캐시도 확장해야 합니다. 앞서 [데이터베이스 캐싱](#) 섹션에서 설명한 것처럼 ElastiCache는 가용성을 높이기 위해 ElastiCache 클러스터의 여러 노드에 걸쳐 그리고 리전의 여러 가용 영역에 걸쳐 캐시를 확장하는 기능을 제공합니다. ElastiCache 클러스터를 확장할 때 구성 엔드포인트를 사용하여 연결하도록 캐싱 플러그 인을 구성해야 합니다. 그래야 WordPress가 새 클러스터 노드가 추가되면 사용을 시작하고 기존 클러스터 노드가 제거되면 사용을 중지할 수 있습니다. 또한 [PHP용 ElastiCache 클러스터 클라이언트](#)를 사용하도록 웹 서버를 설정하고 AMI를 업데이트하여 이 변경 사항을 저장해야 합니다.

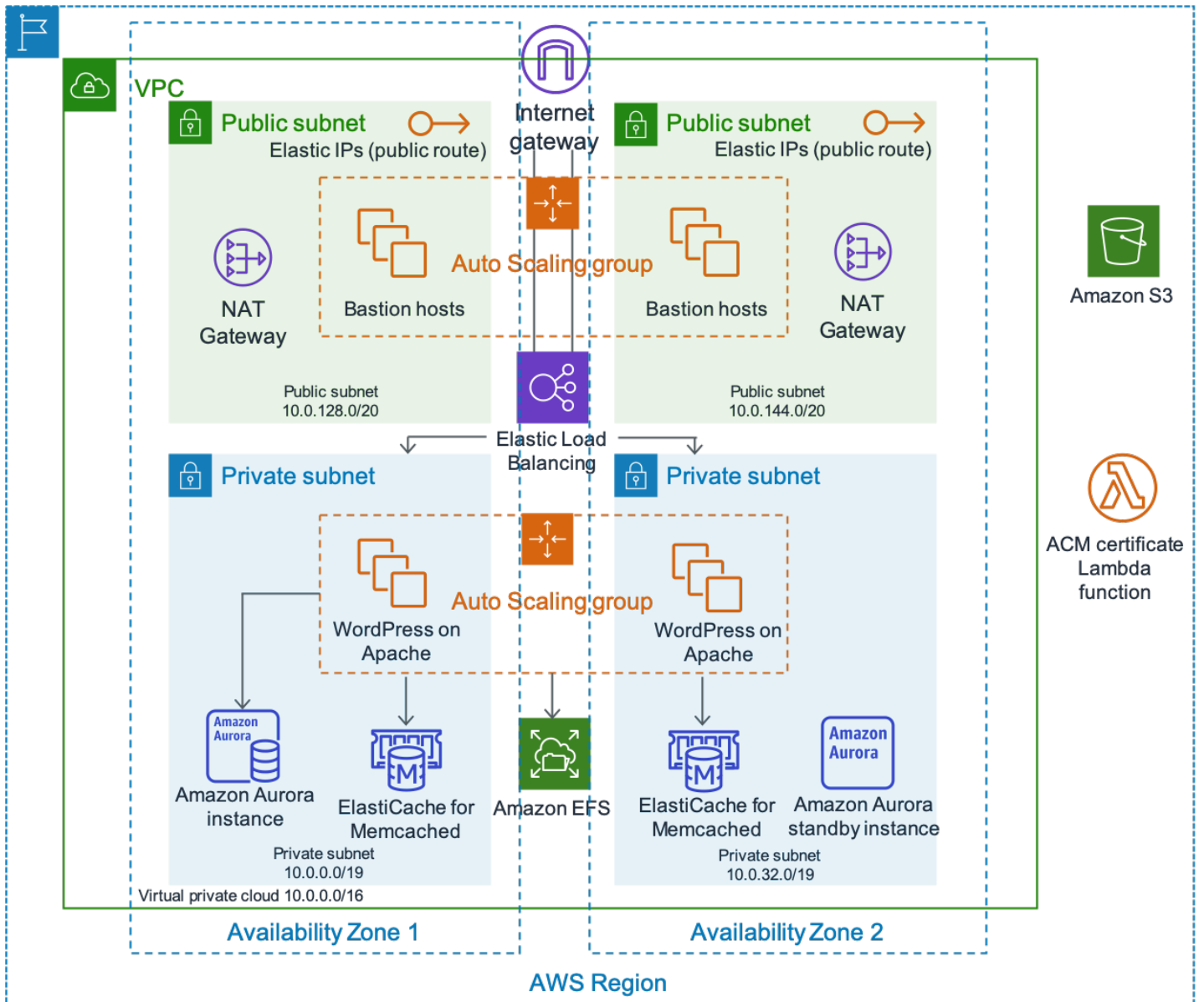
AWS 기반 WordPress High Availability by Bitnami 빠른 시작

빠른 시작은 고객이 보안 및고가용성을 위한 AWS 모범 사례에 따라 AWS에 인기 있는 기술을 배포하는 데 도움이 되도록 AWS 솔루션스 아키텍트와 파트너가 구축합니다. 이 액셀러레이터는 수백 단계의 수작업 절차를 몇 단계로 줄여주기 때문에 프로덕션 환경을 신속하게 구축하고 즉시 사용할 수 있습니다. 각 빠른 시작에는 배포를 자동화하는 AWS CloudFormation 템플릿과 아키텍처를 설명하고 단계별 배포 지침을 제공하는 안내서가 포함되어 있습니다.

[AWS 기반 WordPress High Availability by Bitnami 빠른 시작](#)은 AWS에서 다음과 같이 구성 가능한 환경을 설정합니다.

- 두 개의 가용 영역에 걸친 가용성이 뛰어난 아키텍처.*
- AWS 모범 사례에 따라 퍼블릭 및 프라이빗 서브넷으로 구성된 Virtual Private Cloud(VPC). 이는 배포를 위한 네트워크 인프라를 제공합니다.*
- 인터넷 액세스를 제공하는 인터넷 게이트웨이. 이 게이트웨이는 접속 호스트에서 트래픽을 주고받는 데 사용됩니다.*
- 퍼블릭 서브넷에서, 프라이빗 서브넷의 리소스에 대한 아웃바운드 인터넷 액세스를 허용하기 위한 관리형 NAT 게이트웨이.*
- 퍼블릭 서브넷에서, 퍼블릭 및 프라이빗 서브넷의 EC2 인스턴스에 인바운드 Secure Shell(SSH) 액세스를 허용하는 Auto Scaling 그룹의 Linux 배스천 호스트.*
- HTTP 및 HTTPS 요청을 여러 WordPress 인스턴스로 분산하기 위한 Elastic Load Balancing.
- 프라이빗 서브넷에서, Apache에 WordPress 애플리케이션을 호스팅하는 EC2 인스턴스. 이러한 인스턴스는고가용성을 보장하기 위해 Auto Scaling 그룹에서 프로비저닝됩니다.
- 프라이빗 서브넷에서, Amazon Relational Database Service(Amazon RDS)에 의해 관리되는 Amazon Aurora DB 인스턴스.
- 프라이빗 서브넷에서, WordPress 인스턴스 간에 자산(예: 플러그인, 테마, 이미지)을 공유하기 위한 Amazon Elastic File System(Amazon EFS).
- 프라이빗 서브넷에서, 데이터베이스 쿼리 캐싱을 위한 Amazon ElastiCache for Memcached 노드.

* 기존 VPC에 빠른 시작을 배포하는 템플릿에서는 별표가 표시된 작업을 건너뛰고 기존의 VPC 구성을 묻는 메시지를 표시합니다.



Bitnami에 의한 WordPress 고가용성 아키텍처

AWS 기반 WordPress High Availability by Bitnami를 배포하는 방법에 대한 자세한 설명은 이 문서의 범위를 벗어납니다. 구성 및 옵션은 [AWS 기반 WordPress High Availability by Bitnami](#)를 참조하십시오.

결론

AWS는 WordPress를 실행하기 위한 다양한 아키텍처 옵션을 제공합니다. 가장 간단한 옵션은 트래픽이 적은 웹 사이트를 위한 단일 서버 설치입니다. 보다 고급 웹 사이트의 경우 사이트 관리자는 가용성 및 확장성 측면에서 점진적으로 개선되는 몇 가지 다른 옵션을 추가할 수 있습니다. 관리자는 요구 사항과 예산에 가장 근접한 기능을 선택할 수 있습니다.

기여자

이 문서를 작성하는 데 도움을 주신 분들입니다.

- Paul Lewis, 솔루션스 아키텍트, Amazon Web Services
- Ronan Guilfoyle, 솔루션스 아키텍트, Amazon Web Services
- Andreas Chatzakis, 솔루션스 아키텍트 관리자, Amazon Web Services
- Jibril Touzi, 테크니컬 어카운트 관리자, Amazon Web Services
- 김학민, 마이그레이션 파트너 솔루션스 아키텍트, Amazon Web Services

문서 개정

이 백서의 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하세요.

업데이트 기록-변경	update-history-description	update-history-date
백서 업데이트됨	참조 아키텍처 및 AWS for WordPress 플러그 인을 수정하도록 업데이트되었습니다.	2021년 10월 19일
백서 업데이트됨	새로운 배포 방식 및 AWS for WordPress 플러그 인을 포함하도록 업데이트되었습니다.	2019년 10월 30일
백서 업데이트됨	Amazon Aurora 제품 메시지를 명확히 하도록 업데이트되었습니다.	2018년 2월 1일
백서 업데이트됨	최초 게시 이후 출시된 AWS 서비스를 포함하도록 업데이트되었습니다.	2017년 12월 1일
최초 게시	처음 게시되었습니다.	2014년 12월 1일

부록 A: CloudFront 구성

WordPress 웹 사이트에서 Amazon CloudFront를 사용할 때 최적의 성능 및 효율성을 얻으려면 제공하는 다양한 유형의 콘텐츠에 맞게 웹 사이트를 올바르게 구성하는 것이 중요합니다.

주제

- [오리진 및 동작](#)
- [CloudFront 배포 생성](#)

오리진 및 동작

[오리진](#)은 CloudFront가 엣지 로케이션을 통해 배포하는 콘텐츠에 대한 요청을 전송하는 위치입니다. 구현에 따라 하나 또는 두 개의 오리진이 있을 수 있습니다. 하나는 사용자 지정 오리진을 사용하는 동적 콘텐츠([단일 서버 배포 옵션](#)의 Lightsail 인스턴스 또는 [탄력적 배포 옵션](#)의 Application Load Balancer)용입니다. 정적 콘텐츠에 대해 CloudFront를 리디렉션할 두 번째 오리진이 있을 수 있습니다. 이전 [참조 아키텍처](#)의 S3 버킷입니다. Amazon S3를 배포의 오리진으로 사용하는 경우 [버킷 정책](#)을 사용하여 콘텐츠에 공개적으로 액세스할 수 있도록 해야 합니다.

[동작](#)을 사용하면 CloudFront가 콘텐츠를 캐시하는 방법을 제어하는 규칙을 설정하고, 결과적으로 캐시의 효과를 결정할 수 있습니다. 동작을 사용하여 웹 사이트에 액세스할 수 있는 프로토콜 및 HTTP 메서드를 제어할 수 있습니다. 또한 HTTP 헤더, 쿠키 또는 쿼리 문자열을 백엔드로 전달할지 여부(및 전달할 항목)를 제어할 수 있습니다. 동작은 특정 URL 경로 패턴에 적용됩니다.

CloudFront 배포 생성

배포를 따라 CloudFront 웹 배포를 생성합니다. 자동으로 생성된 기본 오리진 및 동작이 동적 콘텐츠에 사용됩니다. 4개의 추가 동작을 생성하여 정적 요청과 동적 요청이 모두 처리되는 방식을 추가로 사용자 지정합니다. 다음 표에는 다섯 가지 동작에 대한 구성 속성이 요약되어 있습니다. 이 수동 구성을 건너뛰고 [부록 B: 플러그인 설치 및 구성](#)에 설명되어 있는 AWS for WordPress 플러그인을 사용할 수도 있습니다. 이것이 WordPress 사이트를 가속화하도록 CloudFront를 구성하는 가장 쉬운 방법입니다.

표 1: CloudFront 동작 구성 속성 요약

속성	정적	동적(관리자)	동적(프런트 엔드)
경로(동작)	wp-content/*	wp-admin/*	기본값(*)

속성	정적	동적(관리자)	동적(프런트 엔드)
	wp-includes/*	wp-login.php	
프로토콜	HTTP 및 HTTPS	HTTPS로 리디렉션	HTTP 및 HTTPS
HTTP 메서드	GET, HEAD	모두	모두
HTTP 헤더	없음	모두	호스트 CloudFront-Forwarded-Proto CloudFront-Is-Mobile-Viewer CloudFront-Is-Tablet-Viewer CloudFront-Is-Desktop-Viewer
쿠키	없음	모두	comment_* wordpress_* wp-settings-*
쿼리 문자열	예(무효화)	예	예

기본 동작의 경우 다음 구성을 권장합니다.

- 최종 사용자가 HTTPS를 사용하여 CloudFront에 연결하는 경우 CloudFront도 HTTPS를 사용하여 오리진에 연결하여 엔드투엔드 암호화를 달성할 수 있도록 오리진 프로토콜 정책이 최종 사용자와 일치하도록 허용합니다. 이를 위해서는 로드 밸런서에 신뢰할 수 있는 SSL 인증서를 설치해야 합니다. 자세한 내용은 [CloudFront와 사용자 지정 오리진 간의 통신에 HTTPS 요구](#)를 참조하십시오.
- 웹 사이트의 동적 부분에는 GET 및 POST 요청이 모두 필요하므로(예: 주식 제출 양식에 POST를 지원하기 위해) 모든 HTTP 메서드를 허용합니다.

- WordPress 출력을 변경하는 쿠키만 전달합니다(예: >wordpress_*, wp-settings-* 및 comment_*). 목록에 없는 다른 쿠키에 의존하는 플러그 인을 설치한 경우 해당 목록을 확장해야 합니다.
- WordPress의 출력에 영향을 미치는 HTTP 헤더만 전달합니다(예: Host, CloudFront-Forwarded-Proto, CloudFront-is-Desktop-Viewer, CloudFront-is-Mobile-Viewer, CloudFront-is-Tablet-Viewer).
 - Host: 동일한 오리진에서 여러 WordPress 웹 사이트를 호스트할 수 있습니다.
 - CloudFront-Forwarded-Proto: HTTP 또는 HTTPS를 통해 액세스하는지에 따라 다른 버전의 페이지를 캐시할 수 있습니다.
 - CloudFront-is-Desktop-Viewer, CloudFront-is-Mobile-Viewer, CloudFront-is-Tablet-Viewer: 최종 사용자의 디바이스 유형에 따라 테마 출력을 사용자 지정할 수 있습니다.
- WordPress는 이러한 값을 기반으로 모든 쿼리 문자열을 캐시에 전달하며, 이들은 캐시된 객체를 무효화하는 데에도 사용할 수 있습니다.

웹 사이트를 사용자 지정 도메인 이름으로 제공하려는 경우(즉, *.cloudfront.net이 아님) 배포 설정의 대체 도메인 이름 아래에 적절한 URI를 입력합니다. 이 경우 사용자 지정 도메인 이름에 대한 SSL 인증서도 필요합니다. AWS Certificate Manager를 통해 SSL 인증서를 [요청](#)하고 CloudFront 배포에 대해 구성할 수 있습니다.

이제 동적 콘텐츠용 캐시 동작을 두 개 더 만듭니다. 하나는 로그인 페이지(경로 패턴: wp-login.php)를 위한 것이고 다른 하나는 관리 대시보드(경로 패턴: wp-admin/*)를 위한 것입니다. 이 두 동작의 설정은 다음과 같습니다.

- HTTPS 전용의 최종 사용자 프로토콜 정책을 적용합니다.
- 모든 HTTP 메서드를 허용합니다.
- 모든 HTTP 헤더를 기반으로 캐시합니다.
- 모든 쿠키를 전달합니다.
- 모든 쿼리 문자열을 기반으로 전달 및 캐시합니다.

이렇게 구성하는 이유는 웹 사이트에서 이 섹션이 고도로 개인화되어 있으며 일반적으로 사용자가 몇 명뿐이므로 캐싱 효율성이 주요 관심사가 아니기 때문입니다. 초점은 모든 쿠키 및 헤더를 오리진에 전달하여 설치된 플러그 인과의 호환성을 극대화하기 위해 구성을 단순하게 유지하는 것입니다.

부록 B에서 설명한 [AWS for WordPress 플러그 인](#)은 위의 구성을 충족하는 CloudFront 배포를 자동으로 생성합니다.

기본적으로 WordPress는 모든 것을 [단일 서버 배포](#)의 경우 블록 스토리지(Amazon EBS) 또는 [탄력적 배포](#)의 경우 파일 스토리지(Amazon EFS)인 웹 서버에 로컬로 저장합니다. 스토리지 및 데이터 전송 비용을 줄이는 것 외에도 정적 자산을 Amazon S3로 이동하면 확장성, 데이터 가용성, 보안, 성능이 확보됩니다. 정적 콘텐츠를 Amazon S3로 쉽게 이동할 수 있는 플러그 인이 몇 가지 있습니다. 그중 하나가 [부록 B: 플러그 인 설치 및 구성](#)에도 설명되어 있는 [W3 Total Cache](#)입니다.

부록 B: 플러그인 설치 및 구성

주제

- [AWS for WordPress 플러그인](#)
- [정적 콘텐츠 구성](#)

AWS for WordPress 플러그인

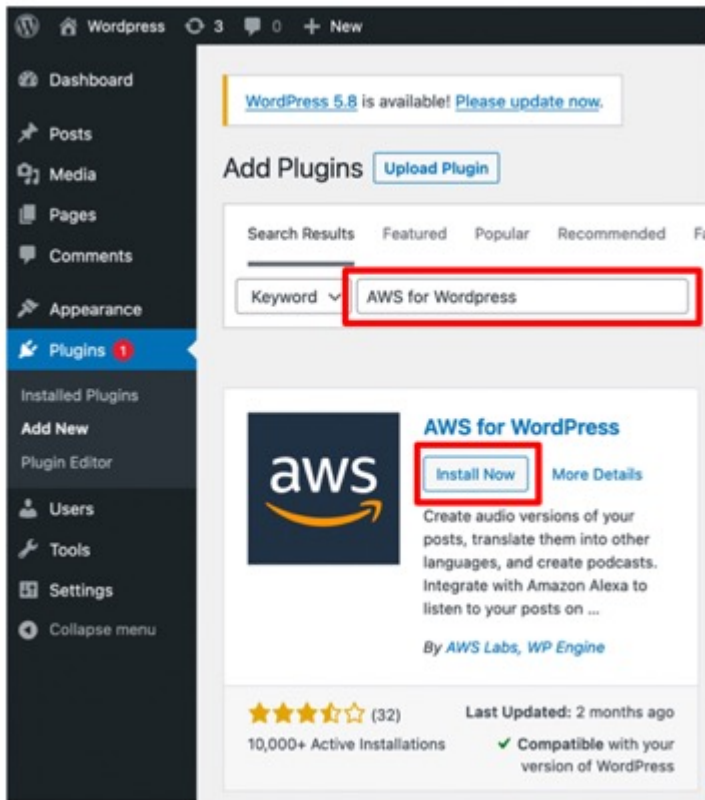
AWS for WordPress 플러그인은 AWS가 작성하고 적극적으로 유지 관리하는 유일한 WordPress 플러그인입니다. 고객은 이 플러그인을 사용하여 [Amazon CloudFront](#) 및 [AWS Certificate Manager\(ACM\)](#)를 WordPress 웹 사이트에 쉽게 구성하여 성능과 보안을 강화할 수 있습니다. 플러그인은 [Amazon Machine Learning](#) 서비스를 사용하여 콘텐츠를 하나 이상의 언어로 번역하고, 각 번역의 오디오 버전을 생성하고, Amazon Alexa 디바이스를 통해 WordPress 웹 사이트를 읽습니다.

이 플러그인은 [AWS 기반 WordPress High Availability by Bitnami 빠른 시작](#)에 이미 설치되어 있습니다.

플러그인 설치 및 구성

플러그인을 설치하려면

1. AWS for WordPress 플러그인을 사용하려면 플러그인을 위한 IAM 사용자를 만들어야 합니다. IAM 사용자는 AWS 서비스에 대한 API 호출을 수행할 권한이 있는 AWS 계정에 속한 사람 또는 애플리케이션입니다.
2. AWS 계정에 대한 인증 및 권한 부여를 제어하려면 [AWS Identity and Access Management\(IAM\)](#) 역할 또는 IAM 사용자가 필요합니다. 권한이 없는 사용자가 이 권한을 얻을 수 없게 하려면 IAM 사용자의 자격 증명을 보호합니다. 보안 액세스 키를 암호처럼 취급하고 안전한 장소에 보관하되, 다른 사람과 공유하지 마십시오. 암호와 마찬가지로 [액세스 키를 주기적으로 바꾸십시오](#). 실수로 보안 액세스 키가 유출되면 [즉시 삭제](#)합니다. 그런 다음 새 액세스 키를 생성하여 AWS for WordPress 플러그인에 사용할 수 있습니다.
3. WordPress 관리자 패널의 플러그인 메뉴에서 AWS for WordPress를 검색하고 지금 설치를 선택합니다.



4. 플러그 인이 설치되지 않는다면 사용자 권한 문제일 수 있습니다. WordPress 웹 서버에 연결하고 다음 지침을 완료하여 문제를 해결하십시오.
- WordPress 설치 디렉터리에서 wp-config.php 파일을 열고 wp-config.php 파일 끝에 다음 코드를 작성합니다.

```
define('FS_METHOD', 'direct');
```

- 다음 명령을 실행하여 쓰기 권한을 부여합니다.

```
chmod 777 <WordPress install directory>/wp-content
```

경고: 쓰기 권한을 777로 유지하는 것은 위험합니다. 이 권한을 777로 유지할 경우 누구나 이 폴더를 편집하거나 삭제할 수 있습니다. 플러그 인 작업을 완료한 후 쓰기 권한을 755 이하로 변경하십시오.

- 참조 아키텍처를 사용하는 경우 WordPress 설치 디렉터리는 '/var/www/wordpress/<site directory>'입니다.

모든 AWS for WordPress 설정에 대한 자세한 설명은 이 문서의 범위를 벗어납니다. 구성 및 옵션은 [AWS for WordPress 플러그 인 시작하기](#)를 참조하십시오.

Amazon CloudFront 및 AWS Certificate Manager

CloudFront 및 AWS Certificate Manager를 설정하려면

- 플러그인 메뉴에서 CloudFront를 선택하고 다음 파라미터를 입력합니다.
 - 오리진 도메인 이름: CloudFront가 웹 사이트의 콘텐츠를 가져오는 HTTP 원본 서버의 DNS 도메인입니다(예: example.com).
 - 대체 도메인 이름(CNAME): 방문자가 가속화된 웹 사이트 경험에 사용하는 도메인 이름입니다. 도메인 앞에 'www'를 사용하는 것이 좋습니다(예:www.example.com).
- 설정 시작을 선택하여 구성을 시작합니다.

플러그인은 ACM을 통해 CNAME에 대한 SSL 인증서를 자동으로 요청합니다. CNAME 항목으로 [DNS 레코드를 업데이트](#)하여 ACM 토큰의 유효성을 검사하면 플러그인이 [부록 A](#)에 정의된 모범 사례를 충족하는 CloudFront 배포를 생성합니다.

Note

AWS for WordPress 플러그인에는 CloudFront와 사용자 지정 오리진 간의 통신을 위해 HTTPS가 필요합니다. 오리진에 Origin 도메인 이름에 유효한 SSL 인증서가 있는지 확인합니다. 자세한 내용은 [CloudFront와 사용자 지정 오리진 간의 통신에 HTTPS 요구](#)를 참조하십시오.

콘텐츠 번역 및 발성

AWS for WordPress 플러그인을 사용하면 자동으로 텍스트를 다른 언어로 번역하고 텍스트 콘텐츠를 다국어 오디오 형식으로 변환할 수 있습니다. 이러한 기능은 Amazon Machine Learning 서비스를 통해 제공됩니다.

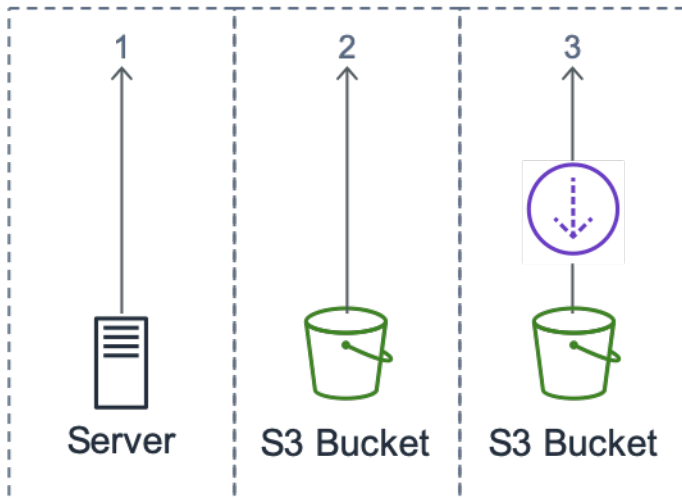
[Amazon Polly](#)는 텍스트를 생생한 음성으로 변환하는 서비스입니다. 다양한 언어로 수십 개의 음성이 제공되므로 여러 국가에서 적합한 음성을 선택하여 몰입적인 음성 지원 애플리케이션을 개발할 수 있습니다. 이 플러그인을 사용하면 Amazon Polly에서 지원하는 모든 음성 및 언어로 오디오 파일을 생성할 수 있습니다. 사용자의 방문자는 인라인 오디오 플레이어 및 모바일 애플리케이션을 사용하여 편리하게 오디오를 스트리밍할 수 있습니다.

기본적으로 플러그인은 새 오디오 파일을 웹 서버에 저장합니다. 파일을 Amazon S3 또는 Amazon CloudFront에 저장하도록 선택할 수 있습니다. 사용자는 오디오 파일을 저장하는 위치와 상관없이 동일한 청취 경험을 갖습니다. 브로드캐스트 위치만 변경됩니다.

- WordPress 서버에 저장된 오디오 파일의 경우 파일은 서버에서 직접 브로드캐스트됩니다.
- S3 버킷에 저장된 파일의 경우 해당 버킷에서 브로드캐스트됩니다.
- CloudFront를 사용하는 경우 파일은 Amazon S3에 저장되고 CloudFront와 함께 브로드캐스트됩니다.



Listening...

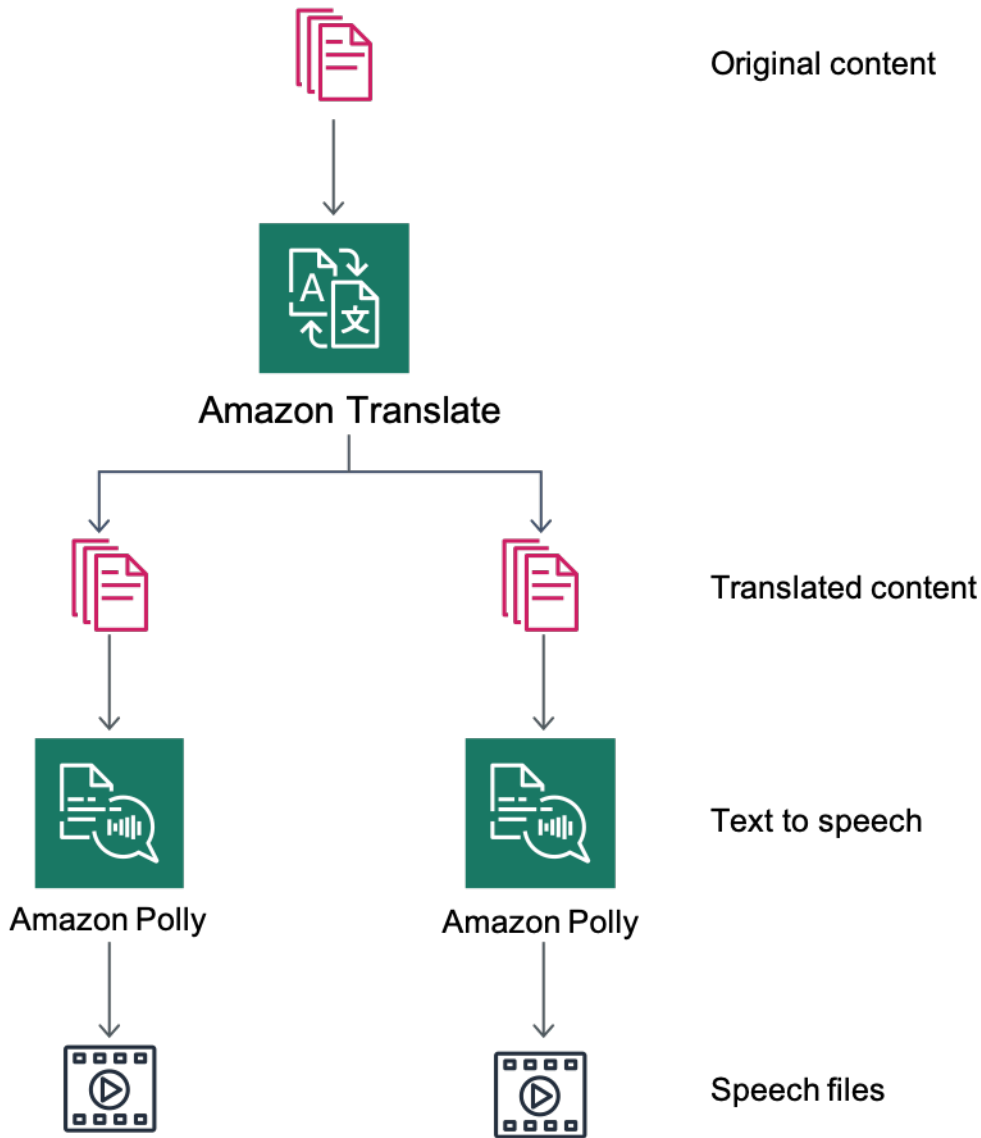


브로드캐스트 위치

[Amazon Translate](#)는 빠르고 저렴한 고품질 언어 번역을 제공하는 신경 기계 번역 서비스입니다. 다국어 콘텐츠 서비스는 사이트 소유자에게 탁월한 기회를 제공합니다. 웹에서는 영어가 지배적이지만 영어를 모국어로 구사하는 사람은 전체 온라인 사용자의 26%에 불과합니다.

WordPress 콘텐츠의 텍스트 및 오디오 버전을 다국어로 제공하면 광범위한 국제 사용자의 요구를 충족할 수 있습니다. 사용자는 다음을 수행하도록 플러그 인을 구성할 수 있습니다.

- 자동으로 다른 언어로 번역하고 게시 시 새 콘텐츠에 대한 각 번역의 오디오 녹음을 생성하거나 개별 게시물을 번역하고 녹음을 생성하도록 선택합니다.
- 보관된 콘텐츠를 다른 언어로 번역하고 각 번역에 대해 오디오 녹음을 생성합니다.
- Amazon Pollycast RSS 피드를 사용하여 오디오 콘텐츠 팟캐스트



콘텐츠 번역 및 텍스트를 음성으로 변환 개요

Amazon Pollycast로 팟캐스트

Amazon Pollycast 피드를 사용하면 방문자가 표준 팟캐스트 애플리케이션을 사용하여 사용자의 오디오 콘텐츠를 들을 수 있습니다. RSS 2.0 호환 Pollycast 피드는 iTunes와 같은 인기 있는 모바일 팟캐스트 애플리케이션 및 팟캐스트 디렉터리별로 팟캐스트를 집계하는 데 필요한 XML 데이터를 제공합니다.

AWS for WordPress 플러그 인을 설치하면 팟캐스트 구성 탭에서 XML 피드 생성을 활성화하는 옵션을 찾을 수 있습니다. 여기에는 여러 선택적 속성을 구성하는 옵션도 있습니다. 이 기능을 활성화하면 피드 링크가 전송됩니다.

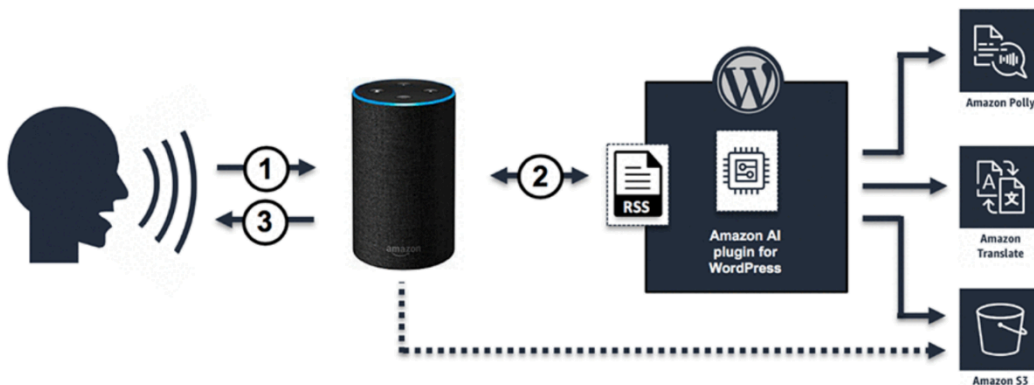
Amazon Alexa 디바이스를 통해 콘텐츠 읽기

Alexa 디바이스를 통해 WordPress 웹 사이트 및 블로그를 확장할 수 있습니다. 그러면 웹 사이트 제작자 및 작성자가 더 많은 대상에게 다가갈 수 있는 새로운 가능성이 열립니다. 또한 사용자는 Alexa에게 블로그를 읽도록 요청하기만 하면 좋아하는 블로그를 더 간편하게 들을 수 있습니다.

WordPress 웹 사이트를 Alexa에 공개하려면 다음을 활성화해야 합니다.

- AWS for WordPress 플러그 인.
- 텍스트를 음성으로 변환 및 Amazon Pollycast 기능. 이러한 기능은 Amazon Alexa가 사용하는 WordPress 사이트에서 RSS 피드를 생성합니다.
- Amazon S3는 텍스트를 음성으로 변환 파일의 기본 스토리지이므로 웹 사이트에서 보안 HTTPS 연결을 사용하여 피드를 Alexa에 공개하는 것이 중요합니다.

다음 다이어그램은 Alexa를 통해 웹 사이트를 공개하는 데 필요한 상호 작용 흐름 및 구성 요소를 보여줍니다.



Alexa를 통해 WordPress 웹 사이트를 공개하는 데 필요한 상호 작용의 흐름

1. 사용자가 예를 들어 Alexa, ask Demo Blog for the latest update(알렉사, 데모 블로그에 최신 업데이트를 요청해줘)라고 말해 새로운 Alexa 스킬을 호출합니다. 스킬 자체는 Alexa 스킬 블루프린트 중 하나를 사용하여 생성됩니다. 그러면 심층적인 기술 지식이 없더라도 Alexa 디바이스를 통해 스킬을 공개할 수 있습니다.
2. Alexa 스킬은 AWS for WordPress 플러그 인에서 생성된 통화 및 RSS 피드를 분석한 다음 최신 문서의 오디오 버전에 대한 링크를 반환합니다.
3. Alexa는 피드에서 제공하는 링크를 기반으로 Amazon S3에 저장된 오디오 파일을 재생하여 문서를 읽습니다.

플러그인 설치 및 기능 구성에 대한 자세한 단계별 안내서는 WordPress 마켓플레이스의 [플러그인 페이지](#)를 참조하십시오.

정적 콘텐츠 구성

기본적으로 WordPress는 모든 것을 [단일 서버 배포](#)의 경우 블록 스토리지(Amazon EBS) 또는 [탄력적 배포](#)의 경우 파일 스토리지(Amazon EFS)인 웹 서버에 로컬로 저장합니다. 스토리지 및 데이터 전송 비용을 줄이는 것 외에도 정적 자산을 Amazon S3로 이동하면 확장성, 데이터 가용성, 보안, 성능이 확보됩니다.

이 예에서는 W3 Total Cache(W3TC) 플러그인을 사용하여 Amazon S3에 정적 자산을 저장합니다. 그러나 유사한 기능을 가진 다른 플러그인도 있습니다. 대체 플러그인을 사용하려면 다음 단계를 적절히 조정할 수 있습니다. 다음 단계는 이 예제와 관련된 기능 또는 설정만 참조합니다. 모든 설정에 대한 자세한 설명은 이 문서의 범위를 벗어납니다. 자세한 내용은 [wordpress.org](#)의 [W3 Total Cache 플러그인 페이지](#)를 참조하십시오.

IAM 사용자 생성

Amazon S3에 정적 자산을 저장하려면 WordPress 플러그인을 위한 AWS Identity and Access Management(IAM) 사용자를 생성해야 합니다. 단계는 [AWS 계정에서 IAM 사용자 생성](#)을 참조하십시오.

참고: IAM 역할은 AWS 리소스에 대한 액세스를 보다 효과적으로 관리할 수 있는 방법을 제공하지만, 본 문서 작성 시점에는 W3 Total Cache 플러그인이 [IAM 역할](#)을 지원하지 않습니다.

사용자 보안 자격 증명을 기록해 두고 안전한 방식으로 저장합니다. 나중에 이러한 자격 증명도 필요합니다.

Amazon S3 버킷 생성

1. 먼저, 선택한 AWS 리전에 Amazon S3 버킷을 생성합니다. 단계는 [버킷 생성](#)을 참조하십시오. [자습서: Amazon S3에서 정적 웹 사이트 구성](#)에 따라 버킷에 대한 정적 웹 사이트 호스팅을 활성화합니다.
2. 앞서 생성한 IAM 사용자에게 지정된 S3 버킷에 대한 액세스 권한을 부여하는 IAM 정책을 생성하고 이 정책을 IAM 사용자에게 연결합니다. 다음 정책을 생성하는 단계는 [IAM 정책 관리](#)를 참조하십시오.

```
{
```

```

"Version": "2012-10-17",
"Statement": [
  {
    "Sid": "Stmt1389783689000",
    "Effect": "Allow",
    "Principal": "*",
    "Action": [
      "s3:DeleteObject",
      "s3:GetObject",
      "s3:GetObjectAcl",
      "s3:ListBucket",
      "s3:PutObject",
      "s3:PutObjectAcl"
    ],
    "Resource": [
      "arn:aws:s3:::wp-demo",
      "arn:aws:s3:::wp-demo/*"
    ]
  }
]
}

```

3. WordPress 관리자 패널에서 W3TC 플러그 인을 설치하고 활성화합니다.
4. 플러그 인 구성의 일반 설정 섹션으로 이동하여 브라우저 캐시 및 CDN이 모두 활성화되어 있는지 확인합니다.
5. CDN 구성의 드롭다운 목록에서 오리진 푸시: Amazon CloudFront를 선택합니다(이 옵션은 Amazon S3를 오리진으로 포함함).
6. 플러그 인 구성의 브라우저 캐시 섹션으로 이동하여 expires, cache control 및 entity tag (ETag) 헤더를 활성화합니다.
7. 또한 설정이 변경될 때마다 새 쿼리 문자열이 생성되고 객체에 추가되도록 설정 변경 후 객체 캐싱 방지(Prevent caching of objects after settings change) 옵션을 활성화합니다.
8. 플러그 인 구성의 CDN 섹션으로 이동하여 앞서 생성한 IAM 사용자의 보안 자격 증명과 S3 버킷의 이름을 입력합니다.
9. CloudFront URL을 통해 웹 사이트를 제공하는 경우 해당 상자에 배포 도메인 이름을 입력합니다. 그렇지 않으면 사용자 지정 도메인 이름에 하나 이상의 CNAME을 입력합니다.
10. 마지막으로 미디어 라이브러리를 내보내고 W3TC 플러그 인을 사용하여 wp-includes, 테마 파일 및 사용자 지정 파일을 Amazon S3에 업로드합니다. 이러한 업로드 기능은 CDN 구성 페이지의 일반 섹션에서 사용할 수 있습니다.

정적 오리진 생성

이제 정적 파일이 Amazon S3에 저장되었으므로 CloudFront 콘솔의 CloudFront 구성으로 돌아가서 Amazon S3를 정적 콘텐츠의 오리진으로 구성합니다. 이렇게 하려면 해당 용도로 생성한 S3 버킷을 가리키는 두 번째 오리진을 추가합니다. 그런 다음 동적 콘텐츠용 기본 오리진이 아닌 S3 오리진을 사용해야 하는 두 폴더(wp-content 및 wp-includes) 각각에 하나씩 두 개의 캐시 동작을 더 생성합니다. 둘 다 동일한 방식으로 구성합니다.

- HTTP GET 요청만 제공합니다.
- Amazon S3는 쿠키 또는 HTTP 헤더를 기반으로 출력을 변경하지 않으므로 CloudFront를 통해 오리진에 전달하지 않음으로써 캐싱 효율성을 개선할 수 있습니다.
- 이러한 동작은 정적 콘텐츠(파라미터를 허용하지 않음)만 제공하지만 그럼에도 쿼리 문자열을 오리진에 전달합니다. 이렇게 하면 쿼리 문자열을 버전 식별자로 사용하여 새 버전을 배포할 때 이전 CSS 파일 등을 즉시 무효화할 수 있습니다. 자세한 내용은 [Amazon CloudFront 개발자 안내서](#)를 참조하십시오.

Note

CloudFront 배포에 정적 오리진 동작을 추가한 후 순서를 확인하여 wp-admin/* 및 wp-login.php에 대한 동작이 정적 콘텐츠에 대한 동작보다 우선 순위가 높은지 확인합니다. 그렇지 않으면 관리자 패널에 액세스할 때 이상한 동작이 나타날 수 있습니다.

부록 C: 백업 및 복구

AWS에서의 장애 복구는 기존 호스팅 환경에 비해 더 신속하고 간편합니다. 예를 들어 하드웨어 장애에 대응하여 몇 분 만에 교체 인스턴스를 시작하거나, 여러 관리형 서비스에서 자동화된 장애 조치를 사용하여 일상적 유지 관리로 인한 재부팅의 영향을 배제할 수 있습니다.

그러나 데이터를 성공적으로 복구하려면 올바른 데이터를 백업하고 있어야 합니다. WordPress 웹 사이트의 가용성을 다시 설정하려면 다음 구성 요소를 복구할 수 있어야 합니다.

- 운영 체제(OS) 및 서비스 설치 및 구성(Apache, MySQL 등)
- WordPress 애플리케이션 코드 및 구성
- WordPress 테마 및 플러그인
- 업로드(예: 게시물용 미디어 파일)
- 데이터베이스 콘텐츠(게시물, 댓글 등)

AWS는 웹 애플리케이션 데이터 및 자산을 백업하고 복원하는 다양한 방법을 제공합니다.

이 백서에서는 앞서 Lightsail 스냅샷을 사용하여 인스턴스의 로컬 스토리지에 저장된 모든 데이터를 보호하는 방법에 대해 설명했습니다. WordPress 웹 사이트가 Lightsail 인스턴스에서만 실행되는 경우 일반 Lightsail 스냅샷만으로도 WordPress 웹 사이트 전체를 복구할 수 있습니다. 그러나 스냅샷에서 복원하면 마지막 스냅샷을 만든 이후 웹 사이트에 적용된 변경 내용이 손실됩니다.

다중 서버 배포에서는 다른 메커니즘을 사용하여 앞서 설명한 각 구성 요소를 백업해야 합니다. 각 구성 요소마다 백업 빈도 요구 사항이 다를 수 있습니다. 예를 들어 OS 및 WordPress 설치 및 구성은 사용자 생성 콘텐츠보다 훨씬 덜 자주 변경되므로 백업 빈도가 낮아도 데이터 손실 없이 복구할 수가 있습니다.

OS 및 서비스 설치 및 구성, WordPress 애플리케이션 코드 및 구성을 백업하기 위해 올바르게 구성된 EC2 인스턴스의 AMI를 생성할 수 있습니다. AMI는 인스턴스 상태의 백업 역할과 새 인스턴스를 시작할 때 템플릿 역할을 하는 두 가지 용도로 사용될 수 있습니다.

WordPress 애플리케이션 코드 및 구성을 백업하려면 AMI 외에 Aurora 백업도 사용해야 합니다.

웹 사이트에 설치된 WordPress 테마 및 플러그인을 백업하려면 해당 항목이 저장된 Amazon S3 버킷 또는 Amazon EFS 파일 시스템을 백업합니다.

- 테마 및 플러그인이 S3 버킷에 저장된 경우 [교차 리전 복제](#)를 활성화하여 기본 버킷에 업로드된 모든 객체가 다른 AWS 리전의 백업 버킷에 자동으로 복제되도록 할 수 있습니다. 교차 리전 복제를 사

용하려면 원본 및 대상 버킷 모두에서 [버전 관리](#)가 활성화되어야 합니다. 이는 추가로 보호 계층을 제공하며 버킷에 있는 객체를 이전 버전으로 되돌릴 수 있습니다.

- 테마 및 플러그 인이 EFS 파일 시스템에 저장된 경우 [Amazon EFS 파일 시스템 백업](#) 설명서 페이지에 설명된 대로 AWS Data Pipeline를 생성하여 프로덕션 EFS 파일 시스템에서 다른 EFS 파일 시스템으로 데이터를 복사할 수 있습니다. 이미 익숙한 백업 애플리케이션을 사용하여 EFS 파일 시스템을 백업할 수도 있습니다.
- 사용자 업로드를 백업하려면 앞서 설명한 WordPress 테마 및 플러그 인 백업 단계를 따라야 합니다.
- 데이터베이스 콘텐츠를 백업하려면 [Aurora 백업](#)을 사용해야 합니다. Aurora는 클러스터 볼륨을 자동으로 백업한 후 백업 보존 기간 동안 복구 데이터를 보관합니다. Aurora 백업은 지속적으로 누적되기 때문에 백업 보존 기간만 벗어나지 않는다면 어떤 시점으로든 신속히 복구할 수 있습니다. 백업 데이터를 쓰는 중에도 성능에 미치는 영향이나 데이터베이스 서비스 중단은 일어나지 않습니다. 백업 보존 기간은 1일에서 35일까지 지정할 수 있습니다. 스냅샷은 삭제할 때까지 유지되는 [수동 데이터베이스 스냅샷](#)을 생성할 수도 있습니다. 수동 데이터베이스 스냅샷은 장기 백업 및 아카이빙에 유용합니다.

부록 D: 새 플러그 인 및 테마 배포

계속 정적으로 유지되는 웹 사이트는 거의 없습니다. 대부분의 경우 공개적으로 사용 가능한 WordPress 테마 및 플러그 인을 정기적으로 추가하거나 최신 WordPress 버전으로 업그레이드합니다. 또는 사용자 지정 테마 및 플러그 인을 처음부터 개발하게 됩니다.

WordPress 설치를 구조적으로 변경할 때마다 예상치 못한 문제가 발생할 위험이 있습니다. 적어도 중요한 변경 사항(예: 새 플러그 인 설치)을 적용하기 전에는 애플리케이션 코드, 구성 및 데이터베이스를 백업하십시오. 비즈니스 또는 기타 가치가 있는 웹 사이트의 경우 먼저 별도의 준비 환경에서 이러한 변경 사항을 테스트하십시오. AWS를 사용하면 프로덕션 환경의 구성을 손쉽게 복제하고 전체 배포 프로세스를 안전한 방식으로 실행할 수 있습니다. 테스트를 마친 후에는 테스트 환경을 해체하면 되므로 해당 리소스에 대한 비용을 지불하지 않아도 됩니다. 나중에 이 백서에서 WordPress 관련 고려 사항에 대해 설명합니다.

일부 플러그 인은 wp_options 데이터베이스 테이블에 구성 정보를 작성하거나 데이터베이스 스키마 변경을 도입하고, 다른 플러그 인은 WordPress 설치 디렉터리에서 구성 파일을 생성합니다. 데이터베이스 및 스토리지를 공유 플랫폼으로 이동했기 때문에 추가 노력 없이도 실행 중인 모든 인스턴스에서 이러한 변경 사항을 즉시 사용할 수 있습니다.

WordPress에서 새 테마를 배포할 때 약간의 노력이 필요할 수 있습니다. Amazon EFS만 사용하여 모든 WordPress 설치 파일을 저장하는 경우 실행 중인 모든 인스턴스에서 새 테마를 즉시 사용할 수 있습니다. 그러나 정적 콘텐츠를 Amazon S3로 오프로드하는 경우 이러한 콘텐츠를 올바른 버킷 위치로 복사해야 합니다. W3 Total Cache와 같은 플러그 인은 해당 작업을 수동으로 시작할 수 있는 방법을 제공합니다. 또는 빌드 프로세스에서 이 단계를 자동화할 수 있습니다.

테마 자산은 CloudFront 및 브라우저에서 캐시될 수 있으므로 변경 사항을 배포할 때 이전 버전을 무효화할 방법이 필요합니다. 이를 위한 가장 좋은 방법은 객체에 일종의 버전 식별자를 포함하는 것입니다. 이 식별자는 날짜/시간 타임스탬프가 있는 쿼리 문자열이거나 임의의 문자열일 수 있습니다. W3 Total Cache 플러그 인을 사용하는 경우 미디어 파일의 URL에 추가된 미디어 쿼리 문자열을 업데이트할 수 있습니다.

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.

© 2021 Amazon Web Services, Inc. 또는 자회사. All rights reserved.