AWS 백서

WordPress 의 에 대한 모범 사례 AWS

WordPress 의 에 대한 모범 사례 AWS: AWS 백서

Table of Contents

WordPress 의 에 대한 모범 사례 AWS

게시일: 2021년 10월 19일(문서 수정)

이 백서는 시스템 관리자에게 Amazon Web Services(AWS) WordPress 를 시작하는 방법과 배포의 비용 효율성과 최종 사용자 경험을 모두 개선하는 방법에 대한 구체적인 지침을 제공합니다. 또한 일반적인 확장성 및 고가용성 요구 사항을 해결하는 참조 아키텍처에 대해 간략하게 설명합니다.

귀사는 Well-Architected입니까?

AWS Well-Architected 프레임워크는 클라우드에서 시스템을 구축할 때 내리는 결정의 장단점을 이해하는 데 도움이 됩니다. 이 프레임워크를 사용하여 클라우드에서 안정적이고 안전하며 효율적이고 비용 효율적인 시스템을 설계하고 운영하기 위한 아키텍처 모범 사례를 살펴볼 수 있습니다. AWS Management Console에서 무료로 제공되는 AWS Well-Architected Tool를 사용하면 각 요소에 대한 일련의 질문에 답하여, 이러한 모범 사례와 비교하여 워크로드를 검토할 수 있습니다.

참조 아키텍처 배포, 다이어그램, 백서 등 클라우드 아키텍처에 대한 더 많은 전문가 지침과 모범 사례를 보려면 AWS 아키텍처 센터를 참조하세요.

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

엄격한 고가용성 요구 사항이 없는 트래픽이 적은 블로그 또는 웹 사이트의 경우 단일 서버를 간단하게 배포하는 것이 적합할 수 있습니다. 이 배포는 가장 복원력이 뛰어나거나 확장 가능한 아키텍처는 아니지만 웹 사이트를 시작하고 실행하는 가장 빠르고 경제적인 방법입니다.

주제

고려 사항

이 논의는 단일 웹 서버 배포로 시작됩니다. 예를 들어, 너머 성장하는 경우가 있을 수 있습니다.

- WordPress 웹 사이트가 배포되는 가상 머신은 단일 장애 지점입니다. 이 인스턴스에 문제가 있으면 웹 사이트에 대한 서비스가 손실됩니다.
- 성능을 개선하기 위한 리소스 확장은 “수직 확장”, 즉 WordPress 웹 사이트를 실행하는 가상 머신의 크기를 늘려야만 가능합니다.

사용 가능한 접근 방식

AWS 에는 가상 머신 프로비저닝을 위한 다양한 옵션이 있습니다. 에서 자체 WordPress 웹 사이트를 호스팅하는 세 가지 주요 방법이 있습니다AWS.

- Amazon Lightsail
- Amazon Elastic Compute Cloud(AmazonEC2)
- AWS Marketplace

Amazon Lightsail은 웹 사이트를 호스팅하기 위해 가상 프라이빗 서버(Lightsail 인스턴스)를 빠르게 시작할 수 있는 서비스입니다 WordPress. Lightsail은 구성 가능한 인스턴스 유형이나 고급 네트워킹 기능에 대한 액세스가 필요하지 않은 경우 가장 쉽게 시작할 수 있는 방법입니다.

AmazonEC2은 몇 분 안에 가상 서버를 시작할 수 있도록 크기 조정 가능한 컴퓨팅 용량을 제공하는 웹 서비스입니다. Amazon은 Lightsail보다 더 많은 구성 및 관리 옵션을 EC2 제공하며, 이는 고급 아키텍

처에 적합합니다. EC2 인스턴스에 대한 관리 액세스 권한이 있으며 를 포함하여 선택한 모든 소프트웨어 패키지를 설치할 수 있습니다 WordPress.

AWS Marketplace 는 에서 실행되는 소프트웨어를 찾고 구매하고 빠르게 배포할 수 있는 온라인 스토어입니다AWS. 1-Click 배포를 사용하여 단 몇 분 만에 자체 AWS 계정EC2의 Amazon으로 사전 구성된 WordPress 이미지를 직접 시작할 수 있습니다. WordPress 인스턴스를 제공하는 ready-to-run Marketplace 공급업체가 많습니다.

이 백서에서는 Lightsail 옵션을 단일 서버 WordPress 웹 사이트의 권장 구현으로 다룹니다.

Amazon Lightsail

Lightsail은 간단한 가상 프라이빗 서버(VPS) 솔루션이 필요한 개발자, 소기업, 학생 및 기타 사용자를 AWS 위한 가장 쉬운 시작 방법입니다.

이 서비스는 인프라 관리의 더 복잡한 요소 중 많은 부분을 사용자로부터 추출합니다. 따라서 인프라 경험이 적거나 웹 사이트 실행에 집중해야 하고 간소화된 제품이 필요에 맞는 경우 이상적인 출발점입니다.

Amazon Lightsail 을 사용하면 Windows 또는 Linux/Unix 운영 체제와 를 비롯한 인기 있는 웹 애플리케이션을 선택하고 사전 구성된 템플릿을 한 번의 클릭으로 WordPress배포할 수 있습니다.

요구 사항이 증가함에 따라 초기 경계를 부드럽게 벗어나 추가 AWS 데이터베이스, 객체 스토리지, 캐싱 및 콘텐츠 배포 서비스에 연결할 수 있습니다.

Amazon Lightsail 요금제 선택

Lightsail 계획은 WordPress 웹 사이트를 호스팅하는 데 사용하는 Lightsail 리소스의 월별 비용을 정의합니다. 다양한 수준의 CPU 리소스, 메모리, 솔리드 스테이트 드라이브(SSD) 스토리지 및 데이터 전송과 함께 다양한 사용 사례를 다루는 데 사용할 수 있는 여러 가지 계획이 있습니다. 웹 사이트가 복잡한 경우 리소스가 더 많은 더 큰 인스턴스가 필요할 수 있습니다. 웹 콘솔을 사용하거나 Amazon Lightsail CLI 설명서 에 설명된 대로 서버를 더 큰 계획으로 마이그레이션하여 이를 달성할 수 있습니다.

설치 WordPress

Lightsail은 와 같이 일반적으로 사용되는 애플리케이션에 대한 템플릿을 제공합니다 WordPress. 이 템플릿은 필요한 대부분의 소프트웨어와 함께 사전 설치되어 있으므로 자체 WordPress 웹 사이트를 실행하기 위한 좋은 출발점입니다. 브라우저 내 터미널 또는 자체 SSH 클라이언트를 사용하거나 관리 웹 인터페이스를 통해 WordPress 추가 소프트웨어를 설치하거나 소프트웨어 구성을 사용자 지정할 수 있습니다.

Amazon Lightsail은 GoDaddy Pro Sites 제품과 제휴하여 WordPress 고객이 인스턴스를 무료로 쉽게 관리할 수 있도록 지원합니다. Lightsail WordPress 가상 서버는 빠른 성능과 보안을 위해 사전 구성되고 최적화되어 있으므로 사이트를 즉시 쉽게 WordPress 시작하고 실행할 수 있습니다. 여러 WordPress 인스턴스를 실행하는 고객은 모든 사이트를 업데이트, 유지 관리 및 관리하는 것이 어렵고 시간이 많이 걸립니다. 이 통합을 사용하면 몇 번의 클릭만으로 몇 분 만에 여러 WordPress 인스턴스를 쉽게 관리할 수 있습니다.

Lightsail을 설치한 후 Lightsail WordPress 에서 관리하는 방법에 대한 자세한 내용은 Amazon Lightsail 인스턴스 WordPress 에서 사용 시작을 참조하세요. WordPress 웹 사이트 사용자 지정이 완료되면 인스턴스의 스냅샷을 찍는 것이 좋습니다.

스냅샷은 Lightsail 인스턴스의 백업 이미지를 생성하는 방법입니다. 시스템 디스크의 복사본이며 원래 시스템 구성(즉, 메모리, , CPU디스크 크기 및 데이터 전송 속도)도 저장합니다. 스냅샷은 잘못된 배포 또는 업그레이드 후 알려진 양호한 구성으로 되돌리는 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 백업되었는지 확인해야 합니다. 앞서 설명한 스냅샷 메커니즘을 이 용도로도 사용할 수 있습니다. 실패에서 복구하려면 최신 스냅샷에서 새 인스턴스를 복원할 수 있습니다. 복원 중에 손실될 수 있는 데이터의 양을 줄이려면 스냅샷이 가능한 최신 상태여야 합니다.

데이터 손실 가능성을 최소화하려면 스냅샷을 정기적으로 캡처해야 합니다. Lightsail Linux/Unix 인스턴스의 자동 스냅샷을 예약할 수 있습니다. 단계는 Amazon Lightsail 에서 인스턴스 또는 디스크에 대한 자동 스냅샷 활성화 또는 비활성화를 참조하세요.

AWS 에서는 고정 IP, 즉 Lightsail 계정 전용 고정 퍼블릭 IP 주소를 사용할 것을 권장합니다. 인스턴스를 다른 인스턴스로 교체해야 하는 경우 정적 IP를 새 인스턴스에 재할당할 수 있습니다. 이렇게 하면 인스턴스를 교체할 때마다 새 IP 주소를 가리키도록 외부 시스템(예: DNS 레코드)을 재구성할 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 Cache는 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 사용자에 맞게 개인화된 사이트 요소).

최종 사용자 경험에서 한 가지 중요한 측면은 위에 언급한 콘텐츠를 전 세계 사용자에게 전송할 때 발생하는 네트워크 대기 시간입니다. 콘텐츠 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 CloudFront는 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 로케이션으로 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

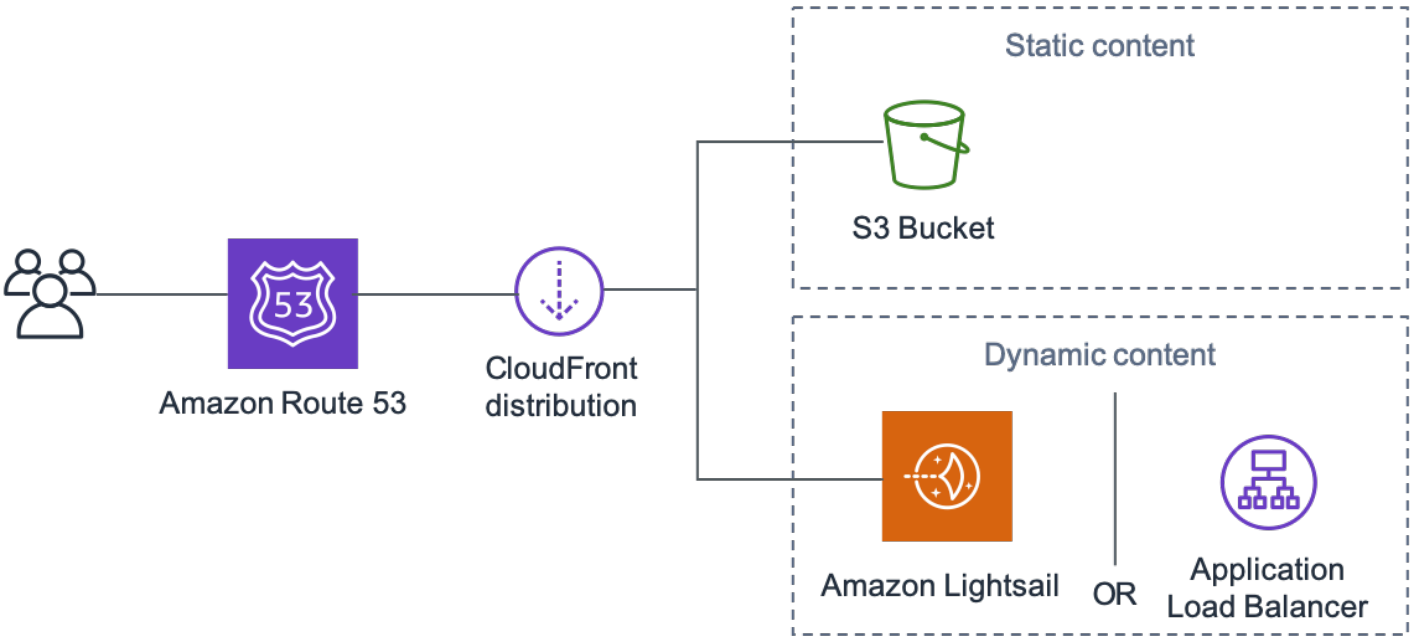
여기에는 CSS, JavaScript 및 이미지 파일이 포함됩니다(WordPress 테마에 속하는 파일 또는 콘텐츠 관리자가 업로드한 미디어 파일). 이러한 파일은 모두 W3 Total Cache와 같은 플러그 인을 사용하여 Amazon Simple Storage Service(Amazon S3)에 저장할 수 있으며 확장 가능하고 가용성이 뛰어난 방식으로 사용자에게 제공할 수 있습니다. Amazon S3는 확장성이 뛰어나고 안정적이며 대기 시간이 짧은 데이터 스토리지 인프라를 저렴한 비용으로 제공하며 REST API를 통해 액세스할 수 있습니다. Amazon S3는 객체를 여러 디바이스뿐만 아니라 AWS 리전의 여러 시설에 중복 저장하므로 고도의 내구성을 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 플러그 인이 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: 플러그 인 설치 및 구성을 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 플러그 인(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 플러그 인이 널리 채택된 메모리 객체 캐싱 시스템인 Memcached와의 통합을 지원합니다. W3 Total Cache 플러그 인이 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 ElastiCache와 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 설명서를 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 설정을 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 OpCache를 사용할 것을 권장합니다.

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

탄력적 배포

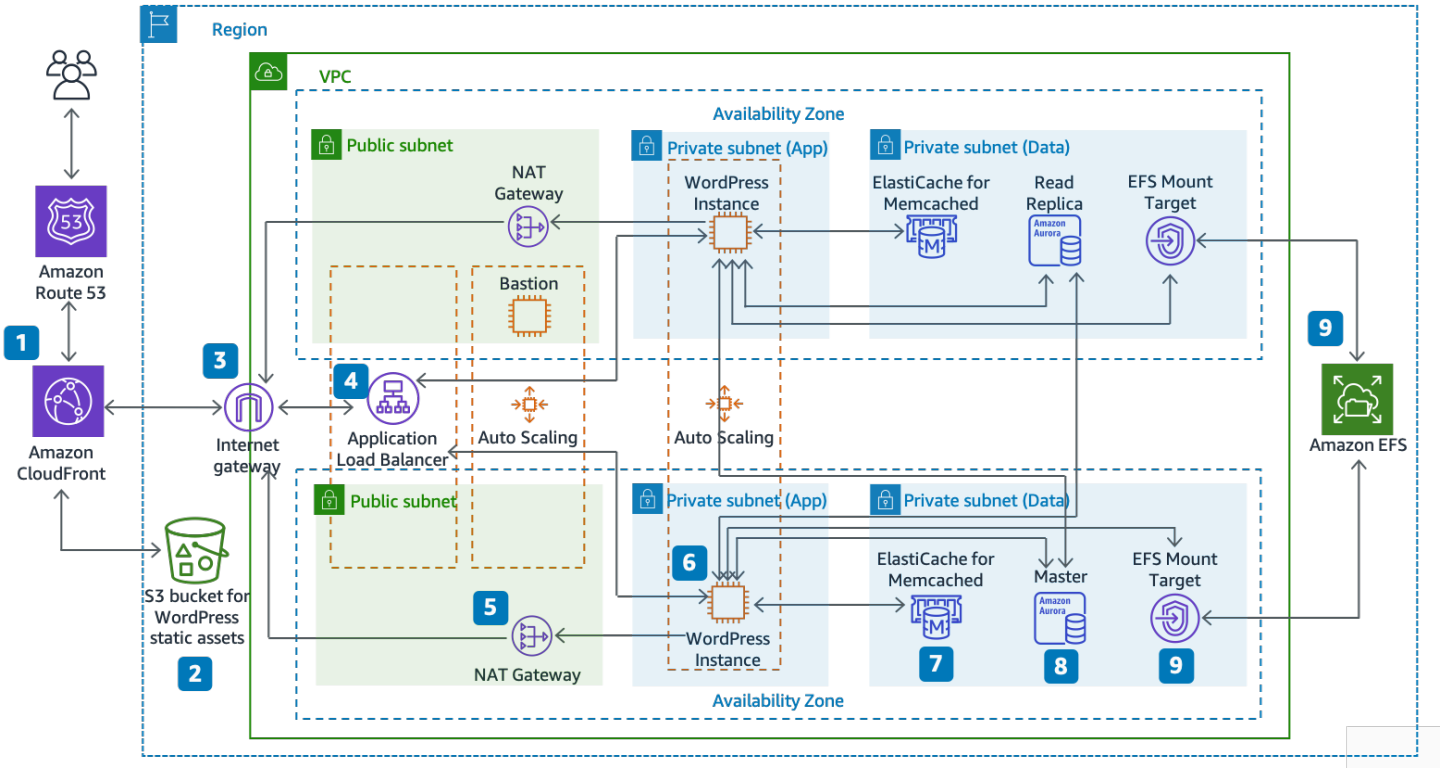
단일 서버 배포가 웹 사이트에 충분하지 않을 수 있는 많은 시나리오가 있습니다. 이러한 상황에서는 확장 가능한 다중 서버 아키텍처가 필요합니다.

주제

참조 아키텍처

에서 GitHub 사용할 수 있는 참조 AWS 아키텍처 WordPress 호스팅은 WordPress 에 배포하기 위한 모범 사례를 간략하게 설명하고 빠르게 시작하고 실행할 수 있는 AWS CloudFormation 템플릿 세트를 AWS 포함합니다. 다음 아키텍처는 해당 참조 아키텍처를 기반으로 합니다. 이 섹션의 나머지 부분에서는 아키텍처 선택의 이유를 검토합니다.

AMI 의 기반은 2021년 7월에 Amazon Linux1에서 Amazon Linux2로 GitHub 변경되었습니다. Linux2 하지만 S3의 배포 템플릿은 아직 변경되지 않았습니다. S3에서 템플릿과 함께 참조 아키텍처를 배포하는 데 문제가 있는 GitHub 경우 에서 템플릿을 사용하는 것이 좋습니다.



WordPress 에서 호스팅하기 위한 참조 아키텍처 AWS

아키텍처 구성 요소

참조 아키텍처는 의 WordPress 웹 사이트에 대한 전체 모범 사례 배포를 보여줍니다AWS.

- Amazon(1)에서 CloudFront 엣지 캐싱으로 시작하여 최종 사용자와 가까운 콘텐츠를 캐싱하여 더 빠르게 전송할 수 있습니다.
- CloudFront 는 S3 버킷(2)에서 정적 콘텐츠를 가져오고 웹 인스턴스 앞에 있는 Application Load Balancer(4)에서 동적 콘텐츠를 가져옵니다.
- 웹 인스턴스는 Amazon EC2인스턴스(6)의 Auto Scaling 그룹에서 실행됩니다.
- ElastiCache 클러스터(7)는 자주 쿼리되는 데이터를 캐싱하여 응답 속도를 높입니다.

  Amazon Aurora MySQL 인스턴스(8)는 데이터베이스를 호스팅합니다 WordPress.
- 인스턴스는 WordPress EC2 각 가용 영역의 EFS 탑재 대상(9)을 통해 Amazon EFS 파일 시스템의 공유 WordPress 데이터에 액세스합니다.
- Internet Gateway(3)를 사용하면 VPC 와 인터넷의 리소스 간에 통신할 수 있습니다.
- 각 가용 영역의NAT 게이트웨이(5)를 사용하면 프라이빗 서브넷(앱 및 데이터)의 EC2 인스턴스가 인터넷에 액세스할 수 있습니다.

Amazon VPC 내에는 퍼블릭(퍼블릭 서브넷)과 프라이빗(앱 서브넷 및 데이터 서브넷)의 두 가지 유형의 서브넷이 있습니다. 퍼블릭 서브넷에 배포된 리소스는 퍼블릭 IP 주소를 수신하고 인터넷에서 공개적으로 볼 수 있습니다. Application Load Balancer(4)와 관리를 위한 Bastion 호스트가 여기에 배포됩니다. 프라이빗 서브넷에 배포된 리소스는 프라이빗 IP 주소만 수신하므로 인터넷에서 공개적으로 볼 수 없으므로 해당 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 이러한 각 고려 사항에 대해 자세히 설명합니다.

웹 계층 크기 조정

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

- Amazon EC2 인스턴스
- Amazon Machine 이미지(AMIs)
- 로드 밸런서
- 자동 크기 조정
- 상태 확인

AWS 는 다양한 EC2 인스턴스 유형을 제공하므로 성능과 비용 모두에 가장 적합한 서버 구성을 선택할 수 있습니다. 일반적으로 컴퓨팅 최적화(예: C4) 인스턴스 유형은 WordPress 웹 서버에 적합한 선택일 수 있습니다. AWS 리전 내 여러 가용 영역에 인스턴스를 배포하여 전체 아키텍처의 신뢰성을 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. 완료되면 해당 구성을 로 저장할 수 AMI있습니다. 이 구성을 사용하여 모든 사용자 지정을 통해 새 인스턴스를 시작할 수 있습니다.

최종 사용자 요청을 여러 웹 서버 노드에 배포하려면 로드 밸런싱 솔루션이 필요합니다. AWS 는 트래픽을 여러 EC2 인스턴스에 분산하는 고가용성 서비스인 Elastic Load Balancing을 통해 이 기능을 제공합니다. 웹 사이트는 HTTP 또는 를 통해 사용자에게 콘텐츠를 제공하기 때문에 필요한 경우 콘텐츠 라우팅과 여러 도메인에서 여러 WordPress 웹 사이트를 실행할 수 있는 기능을 갖춘 애플리케이션 계층 로드 밸런서인 Application Load Balancer를 사용하는 HTTPS것이 좋습니다.

Elastic Load Balancing은 AWS 리전 내의 여러 가용 영역에 걸쳐 요청 배포를 지원합니다. Application Load Balancer가 실패한 개별 인스턴스로 트래픽을 자동으로 전송하지 않도록 상태 확인을 구성할 수도 있습니다(예: 하드웨어 문제 또는 소프트웨어 충돌로 인해). AWS 에서는 상태 확인을 위해 WordPress 관리자 로그인 페이지(`/wp-login.php`)를 사용하는 것이 좋습니다. 이 페이지는 웹 서버가 실행 중이고 웹 서버가 PHP 파일을 올바르게 제공하도록 구성되어 있는지 모두 확인하기 때문입니다.

데이터베이스 및 캐시 리소스와 같은 다른 종속 리소스를 확인하는 사용자 지정 상태 확인 페이지를 빌드하도록 선택할 수 있습니다. 자세한 내용은 Application Load Balancer Guide의 대상 그룹에 대한 상태 확인을 참조하세요.

탄력성은 AWS 클라우드의 주요 특성입니다. 필요할 때 더 많은 컴퓨팅 용량(예: 웹 서버)을 시작하고 그렇지 않을 때는 더 적게 실행할 수 있습니다. Amazon EC2 Auto Scaling은 수동 개입 없이 정의한 조건에 따라 Amazon EC2 용량을 늘리거나 줄일 수 있도록 이 프로비저닝을 자동화하는 데 도움이 되는 AWS 서비스입니다. Amazon EC2 Auto Scaling을 구성하여 수요 급증 중에 사용 중인 EC2 인스턴스 수가 원활하게 증가하여 성능을 유지하고 트래픽이 감소하면 자동으로 감소하여 비용을 최소화할 수 있습니다.

Elastic Load Balancing은 로드 밸런싱 교체에서 Amazon EC2 호스트의 동적 추가 및 제거도 지원합니다. 또한 Elastic Load Balancing 자체는 로드 밸런싱 용량을 동적으로 늘리고 줄여 수동 개입 없이 트래픽 수요에 맞게 조정할 수 있습니다.

상태 비저장 웹 계층

자동 조정 구성에서 여러 웹 서버를 활용하려면 웹 계층이 상태 비저장 상태여야 합니다. 상태 비저장 애플리케이션은 이전 상호 작용에 대한 지식이 필요하지 않고 세션 정보를 저장하지 않는 애플리케이션입니다. 의 경우 WordPress요청을 처리한 웹 서버에 관계없이 모든 최종 사용자가 동일한 응답을 받는다는 의미입니다. 사용 가능한 컴퓨팅 리소스(즉, 웹 서버 인스턴스)에서 요청을 서비스할 수 있으므로 상태 비저장 애플리케이션은 수평으로 확장할 수 있습니다. 해당 용량이 더 이상 필요하지 않은 경우 개별 리소스를 안전하게 종료할 수 있습니다(실행 중인 태스크가 드레이닝된 후). 이러한 리소스는 동료의 존재를 인식할 필요가 없습니다. 필요한 것은 워크로드를 배포하는 방법뿐입니다.

사용자 세션 데이터 스토리지의 경우 WordPress 코어는 클라이언트의 웹 브라우저에 저장된 쿠키에 의존하기 때문에 완전히 상태 비저장 상태입니다. 대신 기본 세션에 의존하는 사용자 지정 코드(예: WordPress 플러그인)를 설치하지 않는 한 PHP 세션 스토리지는 문제가 되지 않습니다.

그러나 WordPress 는 원래 단일 서버에서 실행되도록 설계되었습니다. 따라서 서버의 로컬 파일 시스템에 일부 데이터를 저장합니다. 다중 서버 구성 WordPress 에서 를 실행할 때 웹 서버 간에 불일치가

있기 때문에 문제가 발생합니다. 예를 들어 사용자가 새 이미지를 업로드하면 서버 중 하나에만 저장됩니다.

이는 중요한 데이터를 공유 스토리지로 이동하기 위해 기본 WordPress 실행 구성을 개선해야 하는 이유를 보여줍니다. 모범 사례 아키텍처에는 웹 서버 외부에 별도의 계층으로 데이터베이스가 있으며 공유 스토리지를 사용하여 사용자 업로드, 테마 및 플러그인을 저장합니다.

공유 스토리지(Amazon S3 및 Amazon EFS)

기본적으로 는 로컬 파일 시스템에 사용자 업로드를 WordPress 저장하므로 상태 비저장이 아닙니다. 따라서 웹 서버의 부하를 줄이고 웹 계층을 상태 비저장 상태로 만들려면 WordPress 설치 및 모든 사용자 사용자 지정(구성, 플러그인, 테마 및 사용자 생성 업로드 등)을 공유 데이터 플랫폼으로 이동해야 합니다.

Amazon Elastic File System(Amazon EFS)은 EC2 인스턴스와 함께 사용할 수 있는 확장 가능한 네트워크 파일 시스템을 제공합니다. Amazon EFS 파일 시스템은 제약 없는 수의 스토리지 서버에 분산되어 파일 시스템이 탄력적으로 확장되고 EC2 인스턴스에서 대규모 병렬 액세스를 허용합니다. Amazon의 분산 설계는 기존 파일 서버에 내재된 병목 현상과 제약을 EFS 방지합니다.

전체 WordPress 설치 디렉터리를 EFS 파일 시스템으로 이동하고 부팅할 때 각 EC2 인스턴스에 탑재하면 WordPress 사이트와 모든 데이터가 하나의 EC2 인스턴스에 종속되지 않는 분산 파일 시스템에 자동으로 저장되므로 웹 계층이 완전히 상태 비저장 상태가 됩니다. 이 아키텍처의 이점은 새 인스턴스를 시작할 때마다 플러그인과 테마를 설치할 필요가 없으며 WordPress 인스턴스의 설치 및 복구 속도를 크게 높일 수 있다는 것입니다. 또한 이 문서의 배포 고려 사항 섹션에 설명된 WordPress대로 에서 플러그인 및 테마에 대한 변경 사항을 배포하는 것이 더 쉽습니다.

EFS 파일 시스템에서 실행할 때 웹 사이트의 성능을 최적화하려면 Amazon 및 AWS 참조 아키텍처 WordPressEFSOPcache에서 권장 구성 설정을 확인하세요.

또한 이미지, CSS및 JavaScript 파일과 같은 모든 정적 자산을 CloudFront 캐싱이 앞에 있는 S3 버킷으로 오프로드할 수 있습니다. 이 백서의 정적 콘텐츠 섹션에서 설명한 대로 다중 서버 아키텍처에서 이 작업을 수행하는 메커니즘은 단일 서버 아키텍처와 정확히 동일합니다. 이점은 단일 서버 아키텍처와 동일합니다. 정적 자산을 Amazon S3 및 에 제공하는 것과 관련된 작업을 오프로드하여 웹 서버 CloudFront가 동적 콘텐츠만 생성하는 데 집중하고 웹 서버당 더 많은 사용자 요청을 제공할 수 있습니다.

데이터 계층(Amazon Aurora 및 Amazon ElastiCache)

분산되고 확장 가능하며 공유된 네트워크 파일 시스템에 저장된 WordPress 설치와 Amazon S3에서 제공되는 정적 자산을 사용하면 나머지 상태 저장 구성 요소인 데이터베이스에 집중할 수 있습니다. 스

토리지 계층과 마찬가지로 데이터베이스는 단일 서버에 의존해서는 안 되므로 웹 서버 중 하나에서 호스팅할 수 없습니다. 대신 Amazon Aurora에서 WordPress 데이터베이스를 호스팅합니다.

Amazon Aurora는 클라우드용으로 구축된 MySQL and PostgreSQL 호환 관계형 데이터베이스로, 고급 상용 데이터베이스의 성능과 가용성을 오픈 소스 데이터베이스의 단순성과 비용 효율성과 결합합니다. Aurora MySQL는 데이터베이스 엔진을 가 지원하는 특수 설계된 분산 스토리지 시스템과 긴밀하게 통합하여 내SQL 성능과 가용성을 높입니다SSD. 내결함성 및 자체 복구 기능을 갖추고 있으며, 3개의 가용 영역에 걸쳐 6개의 데이터 사본을 복제하고, 99.99% 이상의 가용성을 위해 설계되었으며, Amazon S3에서 데이터를 지속적으로 백업합니다. Amazon Aurora는 충돌 복구 또는 데이터베이스 캐시 재구축 없이 데이터베이스 충돌을 자동으로 감지하고 다시 시작하도록 설계되었습니다.

Amazon Aurora는 메모리 최적화 및 버스트 가능한 인스턴스를 포함하여 다양한 애플리케이션 프로파일에 적합한 다양한 인스턴스 유형을 제공합니다. 데이터베이스의 성능을 개선하기 위해 대규모 인스턴스 유형을 선택하여 더 많은 CPU 및 메모리 리소스를 제공할 수 있습니다.

Amazon Aurora는 기본 인스턴스와 Aurora 복제본 간의 장애 조치를 자동으로 처리하므로 애플리케이션이 수동 관리 개입 없이 데이터베이스 작업을 최대한 빨리 재개할 수 있습니다. 장애 조치에는 일반적으로 30초 미만이 소요됩니다.

하나 이상의 Aurora 복제본을 생성한 후 클러스터 엔드포인트를 사용하여 기본 인스턴스에 연결하여 기본 인스턴스가 실패할 경우 애플리케이션이 자동으로 장애 조치될 수 있도록 합니다. 3개의 가용 영역에서 지연 시간이 짧은 읽기 전용 복제본을 최대 15개까지 생성할 수 있습니다.

데이터베이스 규모가 조정됨에 따라 데이터베이스 캐시도 확장해야 합니다. 앞에서 데이터베이스 캐싱 섹션에서 설명한 대로 ElastiCache 에는 가용성 향상을 위해 ElastiCache 클러스터의 여러 노드와 리전의 여러 가용 영역에 걸쳐 캐시를 확장하는 기능이 있습니다. ElastiCache 클러스터를 확장할 때 가 새 클러스터 노드를 추가할 때 사용하고 이전 클러스터 노드를 제거할 때 사용을 중지할 WordPress 수 있도록 구성 엔드포인트를 사용하여 연결하도록 캐싱 플러그인을 구성해야 합니다. 클러스터 ElastiCache 클라이언트PHP를 사용하도록 웹 서버를 설정하고 이 변경 사항을 저장하도록 AMI를 업데이트해야 합니다.

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 수정

이 백서의 업데이트에 대한 알림을 받으려면 RSS 피드를 구독하세요.

변경 사항	설명	날짜
백서 업데이트	참조 아키텍처 및 WordPress 플러그인AWS을 수정하도록 업데이트되었습니다.	2021년 10월 19일
백서 업데이트	새로운 배포 접근 방식과 WordPress 플러그인AWS을 포함하도록 업데이트되었습니다.	2019년 10월 30일
백서 업데이트	Amazon Aurora 제품 메시징을 명확히 하기 위해 업데이트되었습니다.	2018년 2월 1일
백서 업데이트	첫 번째 게시 이후 시작된 AWS 서비스를 포함하도록 업데이트되었습니다.	2017년 12월 1일
최초 게시	처음 게시되었습니다.	2014년 12월 1일

부록 A: CloudFront 구성

웹 CloudFront WordPress 사이트에서 Amazon을 사용할 때 최적의 성능과 효율성을 얻으려면 제공되는 다양한 유형의 콘텐츠에 맞게 웹 사이트를 올바르게 구성하는 것이 중요합니다.

주제

오리진 및 동작

오리진은 가 엣지 로케이션을 통해 배포하는 콘텐츠에 대한 요청을 CloudFront 보내는 로케이션입니다. 구현에 따라 오리진이 하나 또는 두 개 있을 수 있습니다. 사용자 지정 오리진을 사용하는 동적 콘텐츠(단일 서버 배포 옵션 의 Lightsail 인스턴스 또는 탄력적 배포 옵션 의 Application Load Balancer)용 1개. 정적 콘텐츠에 대해 로 직접 CloudFront 보낼 두 번째 오리진이 있을 수 있습니다. 이전 참조 아키텍처 에서 이 는 S3 버킷입니다. Amazon S3를 배포의 오리진으로 사용하는 경우 버킷 정책을 사용하여 콘텐츠에 공개적으로 액세스할 수 있도록 해야 합니다.

동작을 사용하면 가 콘텐츠를 CloudFront 캐시하는 방법을 제어하는 규칙을 설정하고 캐시의 효과를 확인할 수 있습니다. 동작을 사용하면 웹 사이트에 액세스할 수 있는 프로토콜과 HTTP 방법을 제어할 수 있습니다. 또한 HTTP 헤더, 쿠키 또는 쿼리 문자열을 백엔드에 전달할지 여부(그리고 전달할 경우 어느 문자열인지)를 제어할 수 있습니다. 동작은 특정 URL 경로 패턴에 적용됩니다.

CloudFront 배포 생성

배포에 따라 CloudFront 웹 배포를 생성하면 자동으로 생성된 기본 오리진 및 동작이 동적 콘텐츠에 사용됩니다. 정적 요청과 동적 요청을 모두 처리하는 방식을 추가로 사용자 지정하려면 네 가지 추가 동작을 생성합니다. 다음 표에는 다섯 가지 동작의 구성 속성이 요약되어 있습니다.

표 1: CloudFront 동작에 대한 구성 속성 요약

속성	정적	동적(관리자)	동적(프런트 엔드)
경로(행동)	`wp-content/*` `wp-includes/*`	`wp-admin/*` `wp-login.php`	기본값(*)

속성	정적	동적(관리자)	동적(프런트 엔드)
프로토콜	HTTP 및 HTTPS	리디렉션 대상 HTTPS	HTTP 및 HTTPS
HTTP 메서드	GET, HEAD	ALL	ALL
HTTP 헤더	NONE	ALL	Host CloudFront-Forwarded-Proto CloudFront-Is-Mobile-Viewer CloudFront-Is-Tablet-Viewer CloudFront-Is-Desktop-Viewer
쿠키	NONE	ALL	의견_* wordpress_* wp-settings-*
쿼리 문자열	YES (무효화)	YES	YES

기본 동작의 경우 는 다음 구성을 AWS 권장합니다.

- 오리진 프로토콜 정책이 뷰어와 일치하도록 허용하여 시청자가 를 사용하여 에 CloudFront 연결하면 HTTPS도 를 사용하여 오리진에 HTTPS CloudFront 연결하여 암호화를 달성 end-to-end합니다. 이렇게 하려면 로드 밸런서에 신뢰할 수 있는 SSL 인증서를 설치해야 합니다. 자세한 내용은 CloudFront 및 사용자 지정 오리진 간의 HTTPS 통신 요구 섹션을 참조하세요.
- 웹 사이트의 동적 부분에 GET 및 POST 요청이 모두 필요하므로(예: POST 의견 제출 양식 지원) 모든 HTTP 방법을 허용합니다.

- , 및 와 같이 WordPress 출력을 변경하는 쿠키만 전달합니다>wordpress_*wp-settings-*comment_*. 목록에 없는 다른 쿠키에 종속되는 플러그인을 설치한 경우 해당 목록을 확장해야 합니다.
- , WordPress, Host, , 등의 출력에 영향을 미치는 HTTP 헤더만 전달합니다CloudFront-Forwarded-ProtoCloudFront-is-Desktop-ViewerCloudFront-is-Mobile-ViewerCloudFront-is-Tablet-Viewer.
  - Host 는 여러 WordPress 웹 사이트를 동일한 오리진에서 호스팅할 수 있도록 허용합니다.
  - CloudFront-Forwarded-Proto 에서는 HTTP 또는 를 통해 액세스되는지 여부에 따라 다양한 버전의 페이지를 캐시할 수 있습니다HTTPS.
  - CloudFront-is-Desktop-Viewer, CloudFront-is-Mobile-Viewer를 CloudFront-is-Tablet-Viewer 사용하면 최종 사용자의 디바이스 유형에 따라 테마의 출력을 사용자 지정할 수 있습니다.
- 모든 쿼리 문자열을 해당 값에 따라 캐시에 전달합니다. WordPress 는 이러한 쿼리 문자열을 사용하므로 캐시된 객체를 무효화하는 데 사용할 수도 있습니다.

사용자 지정 도메인 이름(즉, 가 아님*.cloudfront.net)으로 웹 사이트를 제공하려면 배포 설정의 대체 도메인 이름URIs에 적절한 를 입력합니다. 이 경우 사용자 지정 도메인 이름에 대한 SSL 인증서도 필요합니다. AWS Certificate Manager를 통해 SSL 인증서를 요청하고 CloudFront 배포에 대해 구성할 수 있습니다.

이제 동적 콘텐츠에 대해 두 가지 캐시 동작을 추가로 생성합니다. 하나는 로그인 페이지(경로 패턴: wp-login.php)용이고 다른 하나는 관리자 대시보드(경로 패턴:)용입니다wp-admin/*. 이러한 두 동작은 다음과 같이 정확히 동일한 설정을 갖습니다.

- 의 뷰어 프로토콜 정책HTTPS만 적용합니다.
- 모든 HTTP 메서드를 허용합니다.
- 모든 HTTP 헤더를 기반으로 캐시합니다.
- 모든 쿠키를 전달합니다.
- 모든 쿼리 문자열을 기반으로 전달 및 캐시합니다.

이 구성의 이유는 웹 사이트의 이 섹션이 고도로 개인화되어 있고 일반적으로 사용자 수가 적기 때문에 캐싱 효율성이 주요 관심사가 아니기 때문입니다. 모든 쿠키와 헤더를 오리진에 전달하여 설치된 플러그인과의 호환성을 극대화하기 위해 구성을 간단하게 유지하는 것이 중요합니다.

기본적으로 는 단일 서버 배포의 경우 블록 스토리지(Amazon EBS)이고 탄력적 배포의 경우 파일 스토리지(Amazon EFS)인 모든 것을 웹 서버에 로컬로 WordPress 저장합니다. 스토리지 및 데이터 전송 비용을 줄이는 것 외에도 정적 자산을 Amazon S3로 이동하면 확장성, 데이터 가용성, 보안 및 성능이 제공됩니다. 정적 콘텐츠를 Amazon S3로 쉽게 이동할 수 있는 몇 가지 플러그인이 있습니다. 그 중 하나는 W3 Total Cache 이며, 부록 B: 플러그인 설치 및 구성 에서도 다룹니다.

부록 B: 정적 콘텐츠 구성

기본적으로 는 단일 서버 배포의 경우 블록 스토리지(Amazon EBS)이고 탄력적 배포의 경우 파일 스토리지(Amazon EFS)인 모든 것을 웹 서버에 로컬로 WordPress 저장합니다. ??? 스토리지 및 데이터 전송 비용을 줄이는 것 외에도 정적 자산을 Amazon S3로 이동하면 확장성, 데이터 가용성, 보안 및 성능이 제공됩니다.

이 예제에서는 W3 Total Cache(W3TC) 플러그인을 사용하여 Amazon S3에 정적 자산을 저장합니다. 그러나 유사한 기능을 가진 다른 플러그인도 사용할 수 있습니다. 대안을 사용하려는 경우 그에 따라 다음 단계를 조정할 수 있습니다. 이 단계는 이 예제와 관련된 기능 또는 설정만 참조합니다. 모든 설정에 대한 자세한 설명은 이 문서의 범위를 벗어납니다. 자세한 내용은 wordpress.org W3 Total Cache 플러그인 페이지를 참조하세요.

사용자 생성

Amazon S3에 정적 자산을 저장하려면 WordPress 플러그인에 대한 사용자를 생성해야 합니다. 단계는 AWS 계정에서 사용자 생성을 참조하세요.

참고: 역할은 AWS 리소스에 대한 액세스를 관리하는 더 나은 방법을 제공하지만 작성 시 W3 Total Cache 플러그인은 역할을 지원하지 않습니다.

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

Amazon S3 버킷 생성

1. 먼저 선택한 AWS 리전에서 Amazon S3 버킷을 생성합니다. 단계는 버킷 생성을 참조하세요. 자습서: Amazon S3에서 정적 웹 사이트 구성에 따라 버킷에 대한 정적 웹 사이트 호스팅을 활성화합니다.
2. 정책을 생성하여 이전에 생성한 사용자에게 지정된 S3 버킷에 대한 액세스 권한을 제공하고 정책을 사용자에게 연결합니다. 다음 정책을 생성하는 단계는 정책 관리를 참조하세요.

```
{
"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 플러그인을 설치하고 활성화합니다.
4. 플러그인 구성의 일반 설정 섹션으로 이동하여 브라우저 캐시와 가 모두 활성화CDN되어 있는지 확인합니다.
5. CDN 구성의 드롭다운 목록에서 오리진 푸시: Amazon CloudFront(이 옵션은 Amazon S3를 오리진으로 사용)을 선택합니다.
6. 플러그인 구성의 브라우저 캐시 섹션으로 이동하여 만료 , 캐시 제어 및 엔터티 태그(ETag) 헤더를 활성화합니다.
7. 또한 설정이 변경될 때마다 새 쿼리 문자열이 생성되고 객체에 추가되도록 설정 변경 후 객체 캐싱 방지 옵션을 활성화합니다.
8. 플러그인 구성의 CDN 섹션으로 이동하여 이전에 생성한 사용자의 보안 자격 증명과 S3 버킷의 이름을 입력합니다.
9. 를 통해 웹 사이트를 제공하는 경우 관련 상자에 배포 도메인 이름을 CloudFront URL입력합니다. 그렇지 않으면 사용자 지정 도메인 이름(들)CNAMEs에 하나 이상을 입력합니다.
10. 마지막으로, 미디어 라이브러리를 내보내고 W3TC 플러그인을 사용하여 Wp 포함, 테마 파일 및 사용자 지정 파일을 Amazon S3에 업로드합니다. 이러한 업로드 함수는 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 목적으로 생성한 S3 버킷을

가리키는 두 번째 오리진을 추가합니다. 그런 다음 동적 콘텐츠의 기본 오리진이 아닌 S3 오리진을 사용해야 하는 두 폴더(`wp-content` 및 `wp-includes`) 각각에 대해 하나씩 두 개의 캐시 동작을 추가로 생성합니다. 동일한 방식으로 두 가지를 모두 구성합니다.

- HTTP GET 요청만 제공합니다.
- Amazon S3는 쿠키 또는 HTTP 헤더를 기반으로 출력을 변경하지 않으므로 를 통해 오리진에 전달하지 않아 캐싱 효율성을 개선할 수 있습니다 CloudFront.
- 이러한 동작은 정적 콘텐츠(파라미터를 허용하지 않음)만 제공하지만 쿼리 문자열을 오리진에 전달합니다. 이렇게 하면 쿼리 문자열을 버전 식별자로 사용하여 새 버전을 배포할 때 이전 CSS 파일과 같이 즉시 무효화할 수 있습니다. 자세한 내용은 Amazon CloudFront 개발자 안내서를 참조하세요.

> **Note**
>
> CloudFront 배포에 정적 오리진 동작을 추가한 후 순서를 확인하여 `wp-admin/*` 및 의 동작이 정적 콘텐츠의 동작보다 우선순위`wp-login.php`가 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 플러그 인
- 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 올바로 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 플러그 인을 백업하려면 해당 항목이 저장된 Amazon S3 버킷 또는 Amazon EFS 파일 시스템을 백업합니다.

- 테마 및 플러그 인이 S3 버킷에 저장된 경우 교차 리전 복제를 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 관리가 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 백업을 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 스냅샷을 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 문서의 정보를 독립적으로 평가할 책임이 있습니다. 이 문서: (a) 정보 제공용이며, (b) 예고 없이 변경될 수 있는 현재 AWS 제품 제공 및 관행을 나타내며, (c) AWS 및 그 계열사, 공급업체 또는 라이선스 제공자로부터 약속 또는 보증을 생성하지 않습니다. AWS 제품 또는 서비스는 명시적이든 묵시적이든 어떤 종류의 보증, 표현 또는 조건 없이 “있는 그대로” 제공됩니다. 고객에 AWS 대한 의 책임과 의무는 AWS 계약에 의해 관리되며, 이 문서는 AWS 와 고객 간의 계약의 일부이거나 수정하지 않습니다.

AWS 용어집

최신 AWS 용어는 AWS 용어집 참조 의 AWS 용어집을 참조하세요.

기계 번역으로 제공되는 번역입니다. 제공된 번역과 원본 영어의 내용이 상충하는 경우에는 영어 버전이 우선합니다.