AWS ホワイトペーパー

# AWS での WordPress に関するベストプラクティス

# AWS での WordPress に関するベストプラクティス: AWS ホワイトペーパー

# Table of Contents

# AWS での WordPress に関するベストプラクティス

公開日: 2021 年 10 月 19 日 (改訂履歴)

## 要約

このホワイトペーパーでは、システム管理者に対し、アマゾン ウェブ サービス (AWS) で WordPress の使用を開始し、デプロイのコスト効率と、エンドユーザーのエクスペリエンスの両方を改善する方法に関して、具体的なガイダンスを提供します。また、一般的なスケーラビリティと高可用性の要件に対応する、リファレンスアーキテクチャの概要についても説明します。

# はじめに

WordPress は PHP と MySQL をベースとしたオープンソースのブログ作成ツールおよびコンテンツ管理システム (CMS) で、個人ブログからトラフィックが多いウェブサイトまで、あらゆる用途に使用されています。

WordPress の最初のバージョンは 2003 年にリリースされたため、伸縮自在でスケーラブルな最近のクラウドベースインフラストラクチャを念頭に置いた設計にはなっていません。WordPress コミュニティの活動とさまざまな WordPress モジュールのリリースによって、この CMS ソリューションの機能は拡張し続けてきました。現在では、AWS クラウドのさまざまなメリットを活用して、WordPress アーキテクチャを構築できます。

# シンプルなデプロイ

高可用性の厳しい要件がなく、トラフィックの少ないブログやウェブサイトの場合には、シングルサーバーのシンプルなデプロイが適しているでしょう。このデプロイメントは、最も耐障害性やスケーラブルなアーキテクチャというわけではありませんが、ウェブサイトを立ち上げて実行するための最も迅速で経済的な方法です。

トピック



## 考慮事項

まず、単一のウェブサーバーのデプロイから説明を開始します。これでは十分とは言えない状況もあります。次のような理由からです。

- WordPress ウェブサイトをデプロイした仮想マシンは、単一障害点となります。このインスタンスに問題が発生すると、ウェブサイトのサービスが行えなくなります。
- パフォーマンスを向上させるためのリソースのスケーリングは、「垂直スケーリング」でのみ行えます。つまり、WordPress ウェブサイトが動作している仮想マシンのサイズを増やすことによってのみ実現できます。

## 利用可能なアプローチ

AWS には、仮想マシンのプロビジョニングを行うためのさまざまなオプションがあります。AWS で独自の WordPress ウェブサイトをホストするには、主に次の 3 つの方法があります。

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

Amazon Lightsail は、仮想プライベートサーバー (Lightsail インスタンス) をすばやく起動して、WordPress ウェブサイトをホストできるサービスです。高度な設定が可能なインスタンスタイ

プや高度なネットワーク機能にアクセスする必要がない場合、Lightsail は最も簡単な開始方法となります。

Amazon EC2 は、サイズ変更可能なコンピューティング性能を提供するウェブサービスで、数分で仮想サーバーを立ち上げることができます。Amazon EC2 では、Lightsail よりも多くの設定と管理のオプションを提供しています。これらは、より高度なアーキテクチャーで使用することが望ましいオプションです。ユーザーは EC2 インスタンスに管理者としてアクセスできるので、WordPress を含め、自分で選択したどのようなソフトウェアパッケージでもインストールできます。

AWS Marketplace は、AWS で動作するソフトウェアを見つけて購入し、すぐにデプロイできるオンラインストアです。1-Click デプロイを使用すれば、設定済みの WordPress のイメージを、ご自身の AWS アカウントの Amazon EC2 で、わずか数分で直接起動することができます。Marketplace 多数のベンダーが、すぐに実行できる WordPress インスタンスを提供しています。

このホワイトペーパーでは、シングルサーバー WordPress ウェブサイトを実装するための推奨オプションとして、Lightsail を取り上げます。

# Amazon Lightsail

Lightsail は、デベロッパー、小規模ビジネス、学生、およびシンプルな仮想プライベートサーバー (VPS、virtual private server) ソリューションを必要とするその他のユーザーにとって、AWS の利用を開始するための、最も簡単な方法です。

このサービスは、インフラストラクチャ管理の複雑な要素の多くを、ユーザーが設定しなくとも済むようになっています。そのため、インフラストラクチャに関する経験が少ない場合、または自分のニーズでは単純化された製品で十分で、ウェブサイトの運営に専念したい場合に、理想的なスタートポイントとなります。

Amazon Lightsail では、Windows または Linux/Unix オペレーティングシステムと、WordPress を含む一般的なウェブアプリケーションから選択し、設定済みのテンプレートを使用して、ワンクリックでデプロイできます。

ニーズの増大に合わせて、当初の境界をスムーズに超えて、追加の AWS データベース、オブジェクトストレージ、キャッシング、コンテンツ配信サービスに接続できます。

## Amazon Lightsail の料金プランを選択する

Lightsail プランは、WordPress ウェブサイトをホストするために使用する Lightsail リソースの月額費用を定義します。CPU リソース、メモリ、ソリッドステートドライブ (SSD、solid-state drive) ス

トレージ、データ転送のレベルが異なる、さまざまなユースケースに対応できるプランが数多く用意されています。ウェブサイトが複雑な場合は、より多くのリソースを持つ大規模なインスタンスが必要になることがあります。これは、ウェブコンソールを使用するか、Amazon Lightsail CLI ドキュメントの説明に従って、サーバーをより大きなプランに移行することで実現できます。

## WordPress のインストール

Lightsail には、WordPress など一般的に使用されるアプリケーション用のテンプレートが用意されています。このテンプレートは、必要なソフトウェアのほとんどがプリインストールされた状態で付属しているので、独自の WordPress ウェブサイトを動作させる上での優れた出発点となります。ブラウザ内ターミナルまたは独自の SSH クライアントを使用して、または WordPress 管理ウェブインターフェイス経由で、追加のソフトウェアをインストールしたり、ソフトウェアの設定をカスタマイズしたりできます。

Amazon Lightsail は GoDaddy Pro Sites 製品とパートナーシップを結んでおり、WordPress のお客様が無料でインスタンスを簡単に管理できるよう支援しています。Lightsail WordPress の仮想サーバーは高速のパフォーマンスとセキュリティを実現するよう事前に設定および最適化されています。これにより WordPress サイトをすぐに立ち上げることができます。複数の WordPress インスタンスを実行しているお客様は、すべてのサイトを更新、維持、管理するには大変な作業と多くの時間が必要であることを実感しています。この統合により、数回クリックするだけで、複数の WordPress インスタンスを数分で簡単に管理できます。

Lightsail での WordPress の管理について詳しくは、 Amazon Lightsail インスタンスからの WordPress の使用開始を参照してください。WordPress ウェブサイトのカスタマイズが完了したら、インスタンスのスナップショットを作成することをお勧めします。

スナップショットは、Lightsail インスタンスのバックアップイメージを作成する方法です。システムディスクのコピーであり、元のマシン構成 (メモリ、CPU、ディスクサイズ、およびデータ転送レート) も格納されています。スナップショットは、デプロイやアップグレードの結果に問題があった場合、正常な状態であったことがわかっている設定に復元するために使用できます。

ここでスナップショットを作成しておけば、必要な場合にサーバーの状態を復元できるほか、同じカスタマイズを施した新しいインスタンスを起動するためにも使用できます。

## 障害からの復旧

単一のウェブサーバーは単一障害点なので、ウェブサイトのデータは必ずバックアップしておく必要があります。前述のスナップショットのメカニズムは、この目的で使用することもできます。障害から復旧するには、最新のスナップショットから新しいインスタンスを復元します。復元の際に失われ

るデータの量を少なくするために、スナップショットはできるだけ新しいものである必要があります。

データ損失の可能性を最小にするため、スナップショットは定期的に作成してください。Lightsail Linux/Unix インスタンスの自動スナップショットをスケジュールできます。手順については、Amazon Lightsail でのインスタンスまたはディスクの自動スナップショットの有効化または無効化を参照してください。

AWS では、静的 IP (Lightsail アカウント専用の固定のパブリック IP アドレス) の使用を推奨しています。インスタンスを別のもので置き換える必要がある場合は、その静的 IP を新しいインスタンスに割り当て直してください。このようにすれば、インスタンスを置き換えるたびに、(DNS レコードなどの) 外部のシステムを新しい IP アドレスをポイントするように設定し直さなくて済みます。

# パフォーマンスとコスト効率を改善する

やがて、シングルサーバーデプロイでは性能が不足するようになるかもしれません。その場合、ウェブサイトのパフォーマンスを改善するオプションを考慮する必要が生じるでしょう。マルチサーバーのスケーラブルなデプロイ (後ほどこのホワイトペーパーで扱います) に移行する前に、さまざまなパフォーマンスおよびコスト効率を適用できます。結局マルチサーバーアーキテクチャに移行することになったとしても、従っておくといい優れたプラクティスです。

以下のセクションでは、WordPress ウェブサイトのパフォーマンスとスケーラビリティの側面を改善できる、多くのオプションについて紹介します。シングルサーバーのデプロイにも適用できるものもあれば、マルチサーバーのスケーラビリティを活用できるものもあります。これらの変更の多くでは、1 つ以上の WordPress プラグインの使用が必要となります。さまざまなオプションがありますが、W3 Total Cache は、それらの多くの変更を 1 つのプラグインとしてまとめている、人気のある選択肢です。

トピック

- コンテンツ配信の加速
- データベースのキャッシング
- バイトコードキャッシング

## コンテンツ配信の加速

どの WordPress ウェブサイトでも、静的コンテンツと動的コンテンツをミックスして配信する必要があります。静的コンテンツには画像、JavaScript ファイル、スタイルシートなどが含まれます。動的コンテンツには、WordPress PHP コードを使用してサーバー側で生成したすべてのコンテンツが含まれます。データベースから生成した、または閲覧者ごとにパーソナライズした、サイトのエレメントなどがあります。

エンドユーザーのエクスペリエンスの点で重要な側面は、このようなコンテンツを世界中のユーザーに配信する際の、ネットワークレイテンシーです。これらのコンテンツの配信を加速すれば、エンドユーザー、特に世界中に散らばっているユーザーのエクスペリエンスは改善されます。これは、Amazon CloudFront などのコンテンツ配信ネットワーク (CDN) を使用して達成できます。

Amazon CloudFront は、世界中の複数のエッジロケーションを通じて、低レイテンシーで高速なデータ転送速度でコンテンツを配信するための、簡単で費用対効果の高い方法を提供するウェブサービスです。閲覧者のリクエストは適切な CloudFront エッジロケーションに自動的にルーティングさ

れ、レイテンシーが短縮されます。コンテンツが (数秒、数分、または数日間) キャッシュ可能で、既に特定のエッジロケーションに保存されている場合、CloudFront はそのコンテンツを即座に配信します。コンテンツがキャッシュされない場合、有効期限が切れている場合、または現在そのエッジロケーションにない場合、CloudFront は、CloudFront 設定ではオリジン (この場合は Lightsail インスタンス) と呼ばれる 1 つ以上の信頼できるソースからコンテンツを取得します。このような方法での取得は最適化されたネットワーク接続経由で行われ、ウェブサイトのコンテンツの配信速度を高速化します。説明したモデルでは、エンドユーザーエクスペリエンスの向上以外にも、オリジンサーバーの負荷も軽減され、大幅なコスト削減につながる可能性があります。

## 静的コンテンツのオフロード

これには CSS、JavaScript、および画像ファイルが含まれます。WordPress テーマの一部か、コンテンツ管理者がアップロードしたメディアファイルです。これらのファイルはすべて、W3 Total Cache などのプラグインを使用して Amazon Simple Storage Service (Amazon S3) に保存され、スケーラブルな可用性の高い方法でユーザーに提供できます。Simple Storage Service (Amazon S3) は、REST API を介してアクセスできる、スケーラブルで信頼性が高く、低レイテンシーのデータストレージインフラストラクチャを低コストで提供します。Simple Storage Service (Amazon S3) は、複数のデバイスだけでなく、AWS リージョン内の複数の施設にまたがってオブジェクトを冗長的に保存するため、非常に高いレベルの耐久性を提供します。

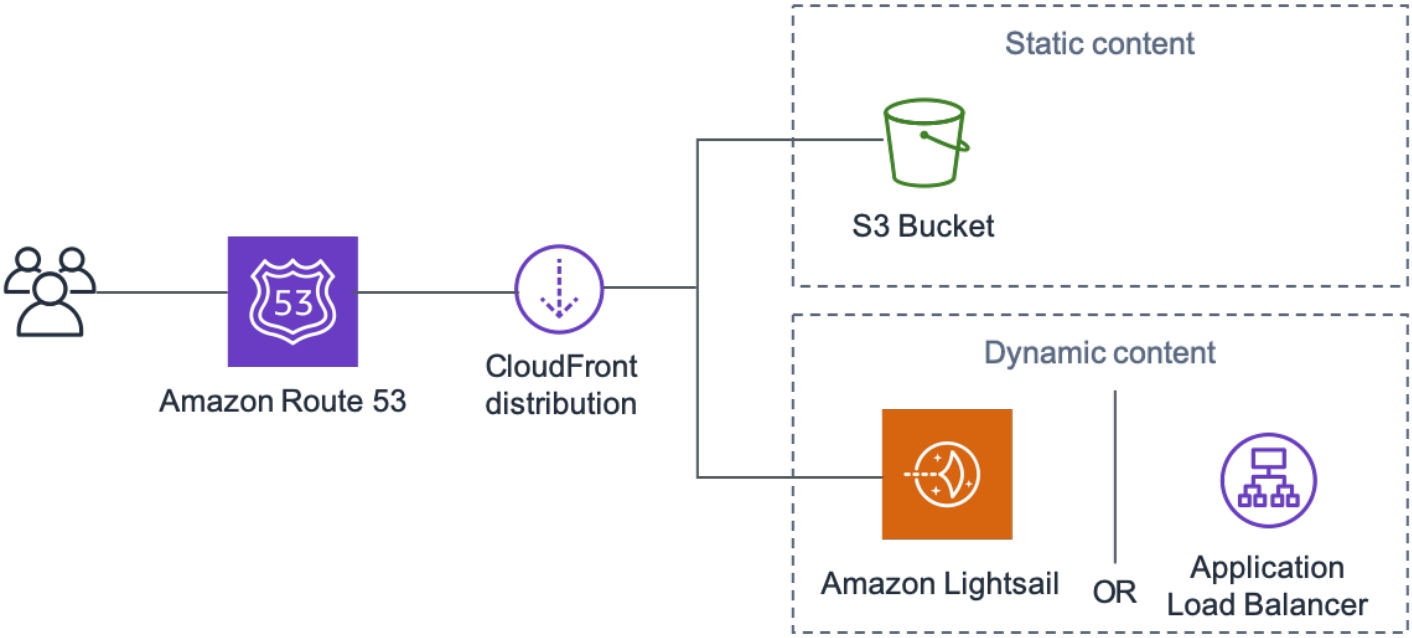
これには有益な副作用もあります。Lightsail インスタンスのワークロードをオフロードできるので、インスタンスは動的なコンテンツの生成に専念できます。これにより、サーバーの負荷が軽減され、ステートレスアーキテクチャ (オートスケーリングを実装する前の前提条件) を作成するための重要なステップとなります。

その後、Simple Storage Service (Amazon S3) を CloudFront のオリジンとして設定し、世界中のユーザーへの静的アセットの配信を改善できます。WordPress は、すぐに使用できるように Simple Storage Service (Amazon S3) や CloudFront と統合されているわけではありませんが、このようなサービスをサポートする多数のプラグインが存在します (W3 Total Cache など)。

## 動的コンテンツ

動的コンテンツには、サーバー側の WordPress PHP スクリプトの出力が含まれます。動的コンテンツは、WordPress ウェブサイトをオリジンにし、CloudFront を経由して提供することもできます。動的コンテンツにはパーソナライズされたコンテンツも含まれるため、カスタムのオリジンサーバーへのリクエストの一部として、特定の HTTP cookie と HTTP ヘッダーを転送するように CloudFront を設定しておく必要があります。CloudFront は転送された cookie の値を、キャッシュ内の一意のオブジェクトを識別するためのキーの一部として用います。キャッシュの効率を最大限に高めるに

は、(ウェブ分析用にクライアント側またはサードパーティーのアプリケーションでのみ使用される cookie ではなく) コンテンツを実際に変化させる HTTP cookie と HTTP ヘッダーのみを転送するように CloudFront を設定する必要があります。




Amazon CloudFront を介したウェブサイト全体の配信

上の図には 2 つのオリジンが含まれています。1 つは静的コンテンツ用で、もう 1 つは動的コンテンツ用です。実装の詳細については、「付録 A: CloudFront の設定」および「付録 B: プラグインのインストールと設定」を参照してください。

CloudFront は、特定の HTTP レスポンスをキャッシュするか、またどれくらいの期間かを判別するのに、標準のキャッシュ制御ヘッダーを使用します。同じキャッシュ制御ヘッダーは、ベストなエンドユーザーエクスペリエンスを実現するため、ウェブブラウザがコンテンツをいつ、どれほどの期間ローカルにキャッシュするかを決めるためにも用いられます (例えば、既にダウンロードした `.css` ファイルは、閲覧者がページにアクセスするたびに再度ダウンロードする必要はありません)。キャッシュ制御ヘッダーは、`.htaccess` ファイル、または `httpd.conf` ファイルの修正などによりウェブサーバーレベルで設定することもできますし、WordPress プラグイン (W3 Total Cache など) をインストールして、静的コンテンツと動的コンテンツの両方について、これらのヘッダーをどのように設定するかを決めることもできます。

## データベースのキャッシング

データベースのキャッシングは、レイテンシーを大幅に削減し、WordPress のような読み取りの多いアプリケーションワークロードのスループットを向上させることができます。アプリケーションの

パフォーマンスは、頻繁にアクセスされるデータ (I/O 負荷の高いデータベースクエリの結果など) をメモリ上に置いて、アクセスのレイテンシーを抑えることで改善できます。クエリの大部分がキャッシュから処理されると、データベースにヒットする必要のあるクエリの数が減り、データベースの実行に伴うコストが削減されます。

WordPress で最初から使えるキャッシング機能は限られたものですが、さまざまなプラグインが、広く採用されているメモリオブジェクトキャッシングシステムである Memcached との統合をサポートしています。W3 Total Cache プラグインがその良い例です。

最もシンプルなシナリオでは、ウェブサーバーに Memcached をインストールし、その結果を新しいスナップショットとしてキャプチャします。この場合、キャッシュの実行に関連する管理タスクはお客様が担当します。

もう 1 つの方法は、 Amazon ElastiCache などのマネージドサービスを活用して、運用上の負担を回避することです。ElastiCache を使用すると、分散型インメモリキャッシュをクラウド内で簡単にデプロイ、運用、スケーリングできます。ElastiCache クラスターノードに接続する方法の詳細については、Amazon ElastiCache のドキュメントに記されています。

Lightsail を使用していて、自分の AWS アカウントでプライベートに ElastiCache クラスターにアクセスする場合には、VPC ピアリングを使用できます。VPC ピアリングを有効にする手順については、Set up Amazon VPC peering to work with AWS resources outside of Amazon Lightsail を参照してください。

## バイトコードキャッシング

PHP スクリプトは、実行のたびに解析されて、コンパイルされます。PHP バイトコードキャッシュを使用すると、PHP コンパイルの出力が RAM に保存されるため、同じスクリプトを何度もコンパイルする必要がありません。これにより、PHP スクリプトの実行に関連したオーバーヘッドは少なくなり、パフォーマンスは向上して、CPU の要件は小さくなります。

バイトコードキャッシュは、WordPress をホストしているどの Lightsail インスタンスにもインストールでき、負荷を大きく減らします。PHP 5.5 以降では、OPcache を使用することを推奨します。これは対応する PHP バージョンにバンドルされている拡張機能です。

Bitnami WordPress Lightsail テンプレートでは OPcache はデフォルトで有効にされているので、特別な作業は必要ないことに注意してください。

# エラスティックなデプロイ

単一サーバーのデプロイではウェブサイトに不十分なシナリオが数多くあります。このような状況では、マルチサーバーでスケーラブルなアーキテクチャが必要です。

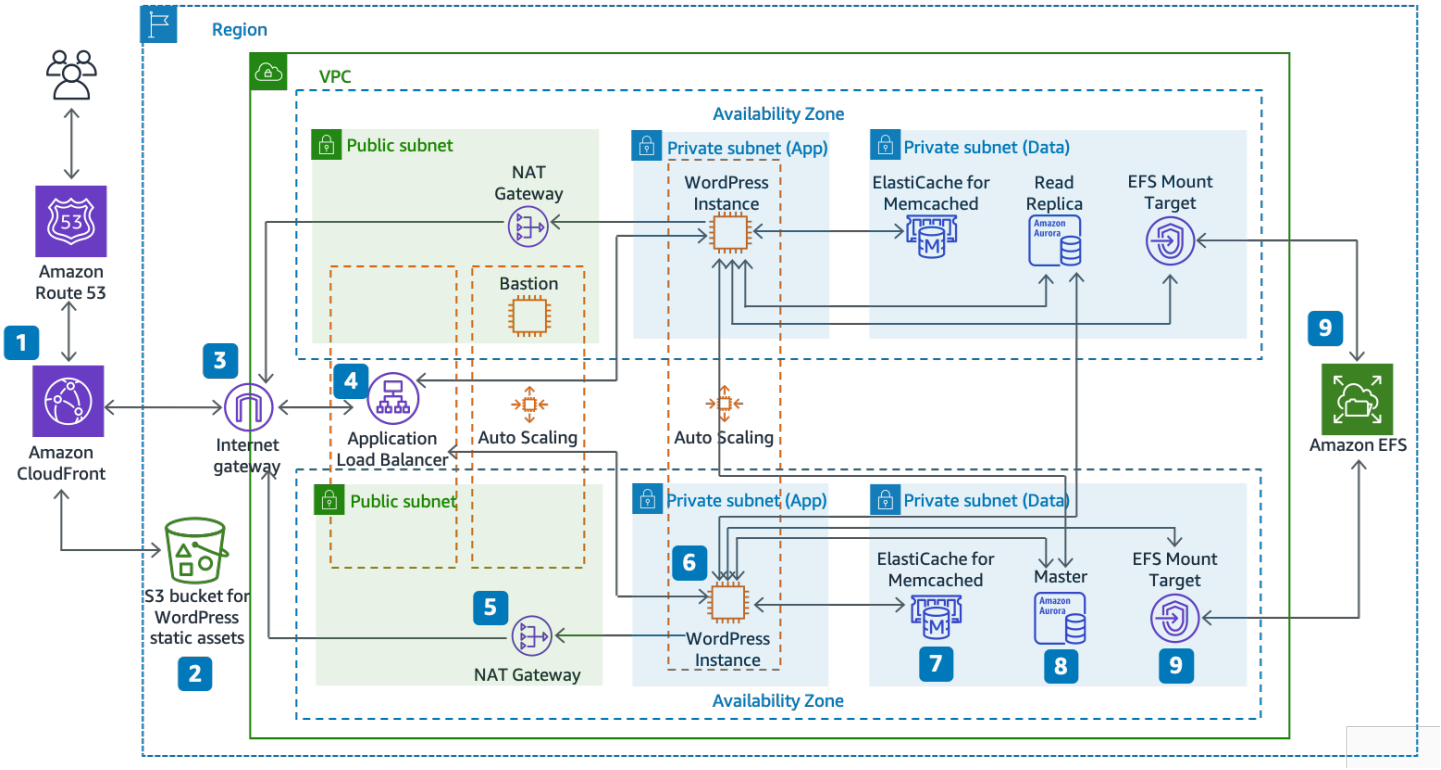
トピック

- リファレンスアーキテクチャ
- ウェブ層のスケーリング
- ステートレスなウェブ層
- AWS の WordPress High Availability by Bitnami クイックスタート

## リファレンスアーキテクチャ

GitHub で利用可能な Hosting WordPress on AWS reference architecture では、WordPress on AWS をデプロイするためのベストプラクティスを概説し、迅速に稼働させるための AWS CloudFormation テンプレートセットが含まれています。次のアーキテクチャは、そのリファレンスアーキテクチャに基づいています。このセクションの残りの部分では、このようなアーキテクチャを選択した理由についてレビューします。

GitHub のベース AMI は 2021 年 7 月に Amazon Linux1 から Amazon Linux2 に変更されました。ただし、S3 のデプロイテンプレートはまだ変更されていません。S3 でテンプレートを使用してリファレンスアーキテクチャをデプロイする際に問題が発生した場合は、GitHub でテンプレートを使用することをお勧めします。




WordPress on AWS をホストする際のリファレンスアーキテクチャ

## アーキテクチャのコンポーネント

リファレンスアーキテクチャでは、AWS 上に WordPress ウェブサイトをデプロイする際のベストプラクティスの全体像を描いています。

- Amazon CloudFront (1) のエッジにおけるキャッシングから始まっていますが、これは配信を高速にする目的で、エンドユーザーの近くにコンテンツをキャッシングするためです。
- CloudFront は静的コンテンツを S3 バケット (2) から、動的コンテンツをウェブインスタンスの前に設置された Application Load Balancer (4) から取り出します。
- ウェブインスタンスは Amazon EC2 インスタンスの Auto Scaling グループ (6) 内で動作します。
- ElastiCache クラスター (7) は、頻繁に問い合わせを受けるデータをキャッシュして、レスポンスを高速化します。

  Amazon Aurora MySQL インスタンス (8) は、WordPress データベースをホストします。
- WordPress EC2 インスタンスは、Amazon EFS ファイルシステム上の共有 WordPress データに、アベイラビリティーゾーンごとの EFS マウントターゲット (9) を経由してアクセスします。
- インターネットゲートウェイ (3) は、VPC のリソースとインターネットのリソース間のコミュニケーションを可能にします。

- アベイラビリティーゾーンごとの NAT ゲートウェイ (5) は、プライベートサブネット内の EC2 インスタンス (アプリケーションとデータ) がインターネットにアクセスするのを可能にします。

Amazon VPC 内には、2 種類のサブネットが存在します。パブリックのもの (パブリックサブネット) とプライベートのもの (アプリケーションサブネットとデータサブネット) です。パブリックサブネットにデプロイされたリソースにはパブリック IP アドレスが割り当てられ、インターネット上でパブリックに公開されます。Application Load Balancer (4) と管理用の要塞ホストはここにデプロイされます。プライベートサブネットにデプロイされたリソースにはプライベート IP アドレスだけが割り当てられるため、インターネット上に公開されることはありません。これによってこれらのリソースのセキュリティを高めます。WordPress ウェブサーバーインスタンス (6)、ElastiCache クラスターインスタンス (7)、Aurora MySQL データベースインスタンス (8)、および EFS マウントターゲット (9) は、すべてプライベートサブネットにデプロイされます。

このセクションの残りの部分では、これらの考慮事項について詳しく説明します。

# ウェブ層のスケーリング

シングルサーバーアーキテクチャをマルチサーバーでスケーラブルなアーキテクチャに進化させるには、5 つの主要コンポーネントを使用する必要があります。

- Amazon EC2 インスタンス
- Amazon マシンイメージ (AMI)
- ロードバランサー
- オートスケーリング
- ヘルスチェック

AWS にはさまざまな EC2 インスタンスタイプが用意されているため、パフォーマンスとコストの両面で最適なサーバー設定を選択できます。一般的に、WordPress ウェブサーバーには、コンピューティング最適化 (C4 など) インスタンスタイプが適しています。AWS リージョン内の複数のアベイラビリティーゾーンにインスタンスをデプロイして、アーキテクチャ全体の信頼性を高めることができます。

自分の EC2 インスタンスを完全にコントロールできるため、root アクセスでログインして WordPress ウェブサイトの実行に必要なすべてのソフトウェアコンポーネントをインストールおよび設定できます。完了したら、その設定を AMI として保存します。この AMI を使用して、行ったカスタマイズがすべて適用された状態で新しいインスタンスを起動できます。

エンドユーザーのリクエストを複数のウェブサーバーノードに分散するには、ロードバランシングソリューションが必要です。AWS は、複数の EC2 インスタンスにトラフィックを分散する可用性の高いサービスである Elastic Load Balancing を通じてこの機能を提供しています。ウェブサイトは HTTP または HTTPS 経由でユーザーにコンテンツを提供しているため、Application Load Balancer を使用することをお勧めします。Application Load Balancer は、コンテンツルーティング機能を備え、必要に応じて複数のドメインで複数の WordPress ウェブサイトを実行できるアプリケーションレイヤーのロードバランサーです。

Elastic Load Balancing はリクエストを 1 つの AWS リージョン内の複数のアベイラビリティーゾーンに分散する機能をサポートしています。また、ハードウェアの問題やソフトウェアのクラッシュなどにより障害が発生した個々のインスタンスへのトラフィックの送信を Application Load Balancer が自動的に停止するように、ヘルスチェックを設定することもできます。ヘルスチェックには、WordPress の管理ログインページ (`/wp-login.php`) を使用することを推奨します。このページでは、ウェブサーバーが動作しているか、そしてウェブサーバーが PHP ファイルを正し処理するように設定されているかの両方を確認できるからです。

データベースやキャッシュリソースなど、他の依存リソースをチェックするカスタムのヘルスチェックページを作成することもできます。詳細については、Application Load Balancer のガイドの「ターゲットグループのヘルスチェック」を参照してください。

伸縮性は AWS クラウドの主要な特性です。この特性により、必要になったときは (ウェブサーバーなどの) コンピューティング性能をより多く起動し、必要ないときには少なくすることが可能になっています。AWS Auto Scaling は AWS のサービスの 1 つで、このプロビジョニングを自動化し、手動による介入を必要とせずに、定義した条件に従って Amazon EC2 の容量をスケールできるようにします。需要の急増時には使用している EC2 インスタンスの数をシームレスに増やしてパフォーマンスを維持し、トラフィックが減少すると自動的に減らすように AWS Auto Scaling を設定することで、コストを最小限に抑えることができます。

Elastic Load Balancing はまた、ロードバランサーのローテーションに参加する Amazon EC2 ホスト数の動的な増減もサポートしています。Elastic Load Balancing 自体も、負荷分散容量を動的に増減し、手動での介入を必要とせずにトラフィック需要に合わせます。

## ステートレスなウェブ層

オートスケーリング設定で複数のウェブサーバーを利用するには、ウェブ層がステートレスである必要があります。ステートレスなアプリケーションとは、それまでのインタラクションを記憶しておく必要がなく、セッション情報を保存しないアプリケーションのことです。WordPress の場合、これはリクエストがどのサーバーで処理されたかに関わりなく、すべてのエンドユーザーが同じレスポン

スを受け取ることを意味しています。ステートレスなアプリケーションでは、すべてのリクエストが利用可能なコンピューティングリソース (つまりウェブサーバーインスタンス) のうちのどれであっても対応できるため、水平方向にスケールできます。その容量が不要になると、(実行中のタスクがクリアされた後で) 個別のリソースを安全に終了させることができます。これらのリソースは、各自のピアの存在を認識する必要はありません。必要なのは、ワークロードをそれらに対して配信する方法だけです。

ユーザーセッションのデータ保存に関しては、WordPress のコアは完全にステートレスです。クライアントのウェブブラウザに保存された cookie にのみ依存しているからです。セッションの保存は、WordPress プラグインなど、ネイティブな PHP セッションに依存するカスタムのコードがインストールされていない限り、問題とはなりません。

ただし、WordPress は本来シングルサーバーで動作するように設計されたものです。そのため、いくらかのデータをサーバーのローカルファイルシステムに保存します。WordPress をマルチサーバー設定で動作させる場合、ウェブサーバー間で不整合が生じるため、問題が発生します。例えば、あるユーザーが新しい画像をアップロードすると、それはいずれかのサーバーにのみ保存されます。

これが、重要なデータを共有ストレージに移動するために、デフォルトの WordPress 実行設定を改善する必要がある理由です。ベストプラクティスアーキテクチャでは、ウェブサーバーの外部に独立したレイヤーとしてデータベースがあり、共有ストレージを利用してユーザーのアップロード、テーマ、プラグインを保存します。

## 共有ストレージ (Simple Storage Service (Amazon S3) と Amazon EFS)

デフォルトでは、WordPress はユーザーがアップロードしたファイルをローカルファイルシステムに保存するので、ステートレスではありません。そのため、ウェブサーバーの負荷を軽減し、ウェブ層をステートレスにするために、インストールされた WordPress システムとユーザーによるすべてのカスタマイズ (設定、プラグイン、テーマ、ユーザーによるアップロードなど) を、共有データプラットフォームに移動する必要があります。

Amazon Elastic File System (Amazon EFS) は、EC2 インスタンスで使用できるスケーラブルなネットワークファイルシステムを提供します。Amazon EFS ファイルシステムは、数の制限を受けない複数のストレージサーバーに分散されるため、ファイルシステムの伸縮自在な拡張と、EC2 インスタンスからの大規模な並列アクセスが可能になります。Amazon EFS の分散設計により、従来のファイルサーバーに本質的に存在したボトルネックや制約を回避できます。

WordPress のインストールディレクトリ全体を EFS ファイルシステムに移行し、それを各 EC2 インスタンスの起動時にマウントすることにより、WordPress サイトとそのすべてのデータは自動的に、1 つの EC2 インスタンスに依存しない分散ファイルシステムに格納され、ウェブ層は完全にス

テートレスになります。このアーキテクチャの利点は、新規インスタンスを起動するたびにプラグインとテーマをインストールする必要がないことと、WordPress インスタンスのインストールと復旧を大幅に高速化できることです。このドキュメントの「デプロイに関する考慮事項」のセクションで説明されているように、WordPress のプラグインとテーマへの変更をデプロイすることもより容易になります。

EFS ファイルシステムから実行される際のウェブサイトのパフォーマンスを最適化するには、AWS Reference Architecture for WordPress に記載されている Amazon EFS と OPcache の推奨設定を確認してください。

また、画像、CSS、JavaScript ファイルなどすべての静的アセットを、CloudFront キャッシュを前面にした S3 バケットにオフロードするというオプションもあります。マルチサーバーアーキテクチャでこれを行うメカニズムは、このホワイトペーパーの「静的コンテンツ」セクションで説明されているように、シングルサーバーアーキテクチャの場合とまったく同じです。利点もシングルサーバーアーキテクチャの場合と同じです。静的アセットの処理に関連した作業を Simple Storage Service (Amazon S3) と CloudFront にオフロードできるため、ウェブサーバーは動的なコンテンツの生成のみに専念でき、ウェブサーバーごとにより多くのユーザーリクエストを処理できるようになります。

## データ層 (Amazon Aurora と Amazon ElastiCache)

WordPress のインストールシステムを分散型でスケーラブルな共有ネットワークファイルシステムに格納し、静的アセットを Simple Storage Service (Amazon S3) から提供されるようにしたため、次は、残っているステートフルなコンポーネントに注意を向けることができます。つまり、データベースです。ストレージ層の場合のように、データベースも単一のサーバーに依存するべきではありません。そのため、ウェブサーバーの中の 1 台にホストさせることはできません。代わりに、Amazon Aurora で WordPress データベースをホストしてください。

Amazon Aurora は、MySQL および PostgreSQL と互換性のあるリレーショナルデータベースで、クラウドに合わせて構築されており、オープンソースデータベースのシンプルさとコスト効率性を備え、高性能の商用データベースの可用性とパフォーマンスを併せ持ったデータベースです。Aurora MySQL は、データベースエンジンと、SSD を利用した特定目的用の分散ストレージシステムとを緊密に統合することによって、MySQL のパフォーマンスと可用性を向上させています。耐障害性と自己修復機能を備え、3 つのアベイラビリティーゾーンにわたって 6 つのデータコピーをレプリケートし、99.99% を超える可用性を実現するように設計されており、Simple Storage Service (Amazon S3) のデータを継続的にバックアップします。Amazon Aurora はデータベースのクラッシュを自動的に検出して再起動するように設計されています。クラッシュの復旧やデータベースキャッシュの再構築を行う必要はありません。

Amazon Aurora には、メモリ最適化インスタンスやバースト可能インスタンスなど、さまざまなアプリケーションプロファイルに適した多数のインスタンスタイプが用意されています。データベースのパフォーマンスを向上させるには、CPU およびメモリリソースを増やすラージインスタンスタイプを選択できます。

Amazon Aurora は、プライマリインスタンスと Aurora レプリカの間のフェイルオーバーを自動的に処理できるため、アプリケーションは管理者の手動介入なしでデータベースオペレーションをできる限り迅速に回復できます。フェイルオーバーに要する時間は通常 30 秒未満です。

少なくとも 1 つの Aurora レプリカを作成したら、クラスターエンドポイントを使用してプライマリインスタンスに接続し、プライマリインスタンスに障害が発生した場合にアプリケーションが自動的にフェイルオーバーできるようにします。3 つのアベイラビリティーゾーンにわたって、最大 15 個の低レイテンシーリードレプリカを作成できます。

データベースがスケールされると、データベースキャッシュもスケールする必要があります。「データベースのキャッシング」セクションで既に説明したように、ElastiCache には、ElastiCache クラスターの複数のノードや、リージョン内の複数のアベイラビリティーゾーンにまたがって、キャッシュをスケールして可用性を向上させる機能があります。ElastiCache クラスターをスケールする際には、キャッシングプラグインが設定エンドポイントを使用して接続するように設定されていることを確認してください。これは、新しいクラスターノードが追加されたときに WordPress がそれを使用し、削除される古いクラスターノードの使用を停止するようにするためです。また、ウェブサーバーが ElastiCache Cluster Client for PHP を使用するように設定し、AMI を更新してこの変更を保存する必要もあります。

# AWS の WordPress High Availability by Bitnami クイックスタート

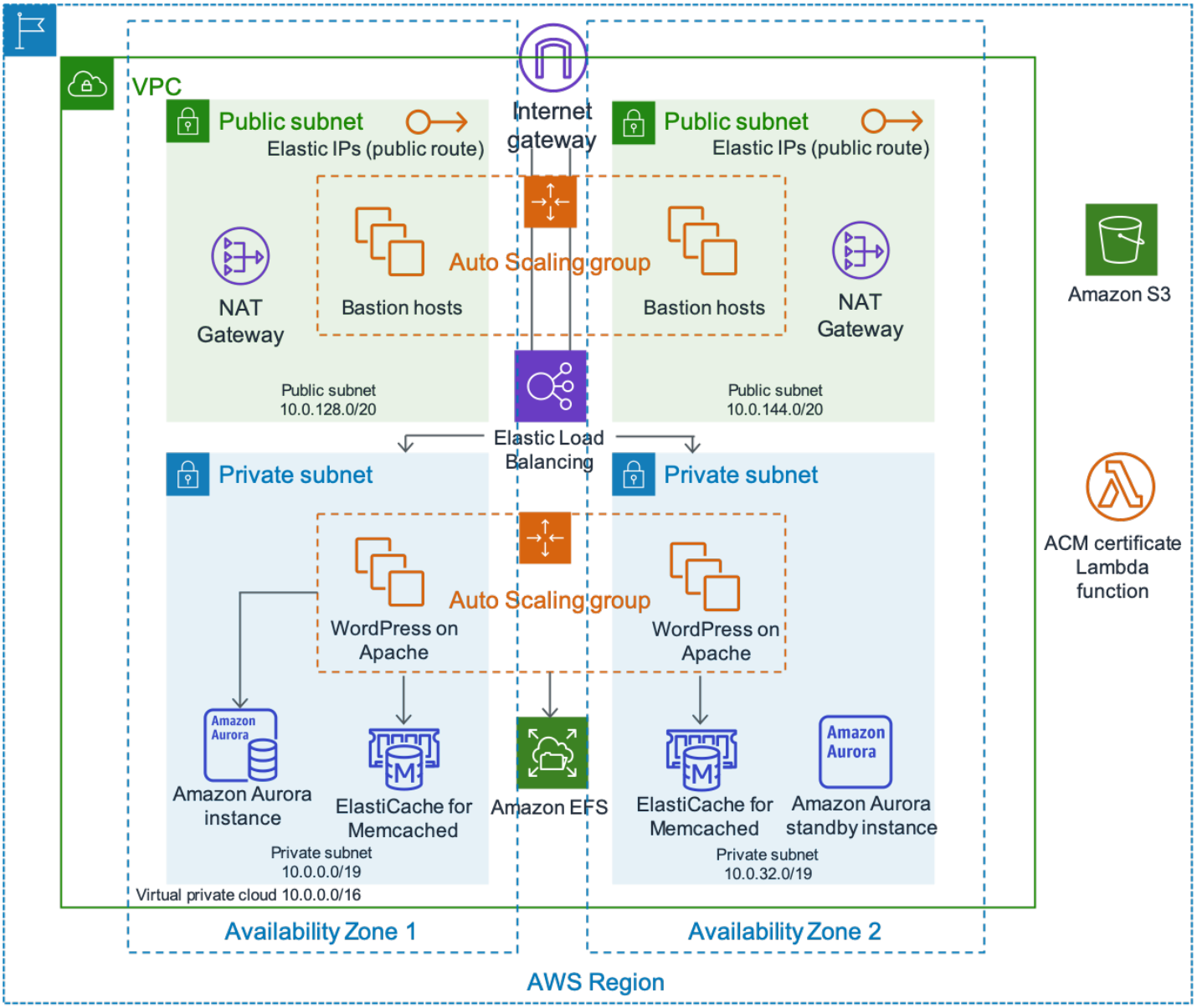
クイックスタートは、AWS ソリューションアーキテクトとパートナーによって構築され、セキュリティと高可用性を実現するための AWS のベストプラクティスに基づいて、貴社が AWS 上に一般的なテクノロジーをデプロイできるよう支援します。これらのアクセラレーターにより、何百もの手作業の手順がほんの数ステップに削減されるため、本番環境をすばやく構築して、すぐに使用を始めることができます。各クイックスタートには、デプロイを自動化する AWS CloudFormation テンプレート、およびアーキテクチャとステップバイステップのデプロイ手順が記載されたガイドが含まれています。

AWS の WordPress High Availability by Bitnami クイックスタートでは、以下の設定可能な環境を AWS にセットアップします。

- 2 つのアベイラビリティーゾーンにまたがる高可用性アーキテクチャ。*

- AWS のベストプラクティスに沿ってパブリックサブネットとプライベートサブネットが設定された仮想プライベートクラウド (VPC)。これが、デプロイにおけるネットワークインフラストラクチャとなります。*
- インターネットへのアクセスを提供するインターネットゲートウェイ。このゲートウェイは、要塞ホストによってトラフィックの送受信に使用されます。*
- パブリックサブネット内で、プライベートサブネットのリソース向けにアウトバウンドのインターネットアクセスを許可する、マネージド NAT ゲートウェイ。*
- パブリックサブネット内で、Auto Scaling グループ内の Linux 要塞ホスト。パブリックおよびプライベートサブネット内の EC2 インスタンスへのインバウンドの Secure Shell (SSH) アクセスを許可します。*
- 複数の WordPress インスタンスに HTTP および HTTPS リクエストを分散する Elastic Load Balancing。
- プライベートサブネット内で、Apache で WordPress アプリケーションをホストする EC2 インスタンス。これらのインスタンスは、高可用性の実現のために Auto Scaling グループでプロビジョニングされます。
- プライベートサブネット内で、Amazon Relational Database Service (Amazon RDS) によって管理される Amazon Aurora DB インスタンス。
- プライベートサブネット内で、WordPress インスタンス間でアセット (プラグイン、テーマ、画像など) を共有するための Amazon Elastic File System (Amazon EFS)。
- プライベートサブネット内で、データベースクエリをキャッシュするための Amazon ElastiCache for Memcached ノード。

* クイックスタートを既存の VPC にデプロイするテンプレートでは、アスタリスクの付いたタスクはスキップされ、既存の VPC 設定を求めるプロンプトが表示されます。




Bitnami による WordPress の高可用性アーキテクチャ

WordPress High Availability by Bitnami を AWS にデプロイする方法については、このドキュメントでは説明しません。設定とオプションについては、AWS の WordPress High Availability by Bitnami を参照してください。

# まとめ

AWS は、WordPress を実行するための多数のアーキテクチャオプションを提供しています。最もシンプルなオプションは、トラフィックの少ないウェブサイトに向いた、シングルサーバーインストールです。もっと性能の高いウェブサイトが必要な場合、サイト管理者は他のいくつかのオプションを追加できます。それぞれが、可用性やスケーラビリティの点での段階的な改善につながります。管理者は、自分の要件と予算に最もよくマッチする機能を選択することができます。

# 寄稿者

本書の作成における寄稿者

- アマゾン ウェブ サービス、ソリューションアーキテクト、Paul Lewis
- アマゾン ウェブ サービス、ソリューションアーキテクト、Ronan Guilfoyle
- アマゾン ウェブ サービス、ソリューションアーキテクトマネージャー、Andreas Chatzakis
- アマゾン ウェブ サービス、テクニカルアカウントマネージャー、Jibril Touzi
- アマゾン ウェブ サービス、移行パートナーソリューションアーキテクト、Hakmin Kim

# 改訂履歴

このホワイトペーパーの更新に関する通知を受け取るには、RSS フィードをサブスクライブしてください。

| update-history-change | update-history-description | update-history-date |
| --- | --- | --- |
| ホワイトペーパーの更新 | リファレンスアーキテクチャと AWS for WordPress プラグインが変更され更新。 | 2021 年 10 月 19 日 |
| ホワイトペーパーの更新 | 新しいデプロイアプローチと AWS for WordPress プラグインを含むように更新。 | 2019 年 10 月 30 日 |
| ホワイトペーパーの更新 | Amazon Aurora 製品のメッセージングを明確化するように更新。 | 2018 年 2 月 1 日 |
| ホワイトペーパーの更新 | 初公開後にリリースされた AWS のサービスを含めるように更新。 | 2017 年 12 月 1 日 |
| 初版公開 | 初公開。 | 2014 年 12 月 1 日 |

# 付録 A: CloudFront の設定

Amazon CloudFront を WordPress ウェブサイトで使用する際に最大のパフォーマンスと効率性が得られるようにするには、提供するコンテンツのタイプに応じて、ウェブサイトを適切に設定しておくことが重要です。

トピック



## オリジンとビヘイビア

オリジンとは、CloudFront がエッジロケーションを通じて配信するコンテンツのリクエストを送信する口ケーションです。実装に応じて、オリジンを 1 つまたは 2 つ持つことができます。1 つは、カスタムオリジンを使用する動的コンテンツ (シングルサーバーデプロイオプションの Lightsail インスタンス、またはエラスティックデプロイオプションの Application Load Balancer) 用です。静的コンテンツ用に CloudFront に指示する 2 つ目のオリジンがある場合があります。前述のリファレンスアーキテクチャでは、これは S3 バケットです。ディストリビューションのオリジンとして Simple Storage Service (Amazon S3) を使用する場合には、コンテンツをパブリックにアクセス可能にするために、バケットポリシーを使用する必要があります。

ビヘイビアを使用すると、CloudFront でコンテンツをキャッシュする方法を制御するルールを設定し、キャッシュの有効性を判断できます。ビヘイビアを使用すると、ウェブサイトにアクセスできるプロトコルと HTTP メソッドをコントロールできます。また、HTTP ヘッダー、cookie、またはクエリ文字列がバックエンドに渡されるかどうか (その場合、どれが渡されるか) をコントロールできます。ビヘイビアーは特定の URL パスパターンに適用されます。

## CloudFront ディストリビューションの作成

ディストリビューションに従って CloudFront ウェブディストリビューションを作成します。自動的に作成されるデフォルトのオリジンとビヘイビアが動的コンテンツに使用されます。静的および動的リクエスト両方を扱う方法をさらにカスタマイズするため、追加のビヘイビアを作成します。次の表は、5 つのビヘイビアの設定プロパティをまとめたものです。この手動設定を省略して、「付録 B: プラグインのインストールと設定」で説明されている AWS for WordPress plugin を使用することもできます。これは、WordPress サイトを加速させるように CloudFront を設定する最も簡単な方法です。

表 1: CloudFront ビヘイビアの設定プロパティの概要

| プロパティ | 静的 | 動的 (管理) | 動的 (フロントエンド) |
|---|---|---|---|
| パス (ビヘイビア) | `wp-content/*`<br>`wp-includes/*` | `wp-admin/*`<br>`wp-login.php` | デフォルト (*) |
| プロトコル | HTTP および HTTPS | HTTPS にリダイレクト | HTTP および HTTPS |
| HTTP メソッド | GET、HEAD | ALL | ALL |
| HTTP ヘッダー | なし | すべて | ホスト<br>CloudFront-Forwarded-Proto<br>CloudFront-Is-Mobile-Viewer<br>CloudFront-Is-Tablet-Viewer<br>CloudFront-Is-Desktop-Viewer |
| Cookie | なし | すべて | comment_*<br>wordpress_*<br>wp-settings-* |
| クエリ文字列 | はい (無効化) | はい | はい |

デフォルトのビヘイビアでは、AWS では以下の設定が推奨されます。

- オリジンのプロトコルポリシーでは Match Viewer (閲覧者に合わせる) を許可します。閲覧者が HTTPS を使用して CloudFront に接続した場合、CloudFront もそのオリジンに HTTPS で接続し

て、端から端まで接続全体が暗号化されるようにするためです。このためには、ロードバランサーに信頼済み SSL 証明書をインストールする必要があります。詳細については、「CloudFront と使用するカスタムオリジン間の通信で HTTPS を必須にする」を参照してください。

- ウェブサイトの動的部分が GET および POST リクエストの両方を要求した場合には、すべての HTTP メソッドを許可します (例えば、コメント送信フォームで POST をサポートするため)。
- >wordpress_*、wp-settings-*、comment_* など、WordPress の出力を変化させる cookie のみを転送します。このリストに掲載されていない他の cookie に依存するプラグインをインストールした場合には、このリストを拡張する必要があります。
- Host、CloudFront-Forwarded-Proto、CloudFront-is-Desktop-Viewer、CloudFront-is-Mobile-Viewer、CloudFront-is-Tablet-Viewer など、WordPress の出力に影響する HTTP ヘッダーのみを転送します。
  - Host では、複数の WordPress ウェブサイトを同じオリジンでホストできます。
  - CloudFront-Forwarded-Proto では、HTTP または HTTPS のどちらでアクセスされるかに応じて、異なるバージョンのページをキャッシュできます。
  - CloudFront-is-Desktop-Viewer、CloudFront-is-Mobile-Viewer、CloudFront-is-Tablet-Viewer では、エンドユーザーのデバイスタイプに基づいて、テーマの出力をカスタマイズできます。
- WordPress はこれらの値に依存しているため、すべてのクエリ文字列をその値に基づいてキャッシュに転送します。また、キャッシュされたオブジェクトの無効化にも使用できます。

カスタムのドメイン名 (つまり *.cloudfront.net 以外) を使用してウェブサイトを運営する場合には、ディストリビューション設定の [Alternate Domain Names] (代替ドメイン名) に適切な URI を入力してください。この場合には、カスタムドメイン名の SSL 証明書も必要です。AWS Certificate Manager で SSL 証明書をリクエストし、CloudFront ディストリビューションに対して設定できます。

ここで、動的コンテンツ用にさらに 2 つのキャッシュビヘイビアを作成します。1 つはログインページ用 (パスパターン: wp-login.php)、もう 1 つは管理ダッシュボード用 (パスパターン: wp-admin/*) です。これら 2 つのビヘイビアの設定は全く同じで、次のようになります。

- HTTPS のみの閲覧者プロトコルポリシーを強制。
- すべての HTTP メソッドを許可。
- すべての HTTP に基づいてキャッシュ。
- すべての cookie を転送。
- すべてのクエリ文字列に基づいて転送およびキャッシュ。

このように設定する理由は、ウェブサイトのこのセクションは高度にパーソナライズされたものとなり、通常は数人のユーザーにしか対応しないため、キャッシングの効率は主要な関心事とはならないためです。重点は、設定をシンプルなものに保ち、すべての cookie とヘッダーをオリジンに渡すことにより、インストールされているプラグインに関し、最大限の互換性が保たれるようにすることです。

付録 B で説明されている AWS for WordPress プラグインは、前述の設定を満たす CloudFront ディストリビューションを自動的に作成します。

デフォルトでは、WordPress はすべてをウェブサーバーにローカルで保存します。ウェブサーバーはシングルサーバーデプロイの場合はブロックストレージ (Amazon EBS)、エラスティックデプロイではファイルストレージ (Amazon EFS) です。静的アセットを Simple Storage Service (Amazon S3) に移行すると、ストレージとデータ転送のコストが削減されるだけでなく、スケーラビリティ、データ可用性、セキュリティ、パフォーマンスが向上します。静的コンテンツを Simple Storage Service (Amazon S3) に簡単に移動できるプラグインがいくつかあります。そのうちの 1 つは W3 Total Cache で、「付録 B: プラグインのインストールと設定」でも説明しています。

# 付録 B: プラグインのインストールと設定

トピック



## AWS for WordPress プラグイン

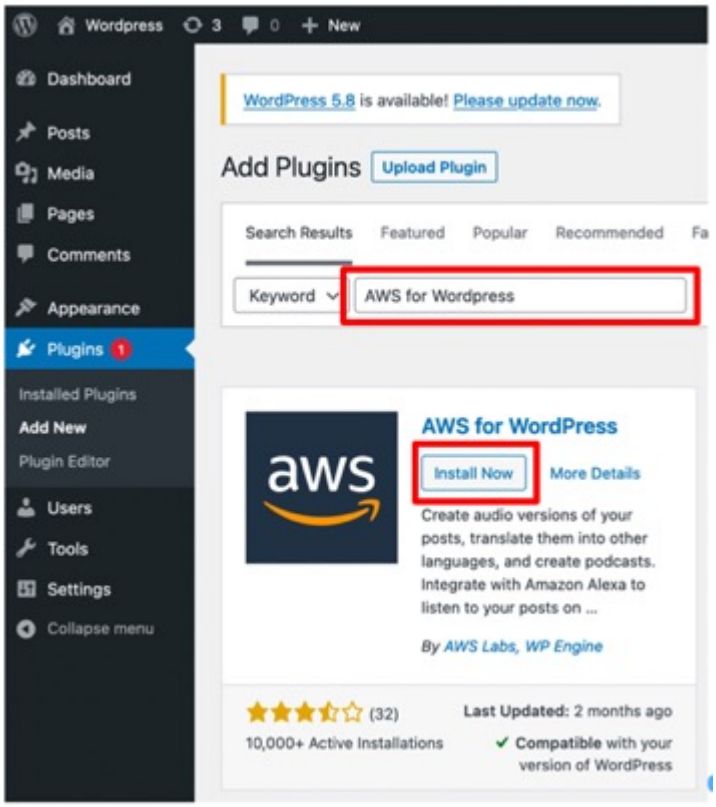
AWS for WordPress プラグインは、AWS によって作成され、積極的に保守されている唯一の WordPress プラグインです。これにより、お客様は Amazon CloudFront と AWS Certificate Manager (ACM) を WordPress ウェブサイトに簡単に設定して、パフォーマンスとセキュリティを強化できます。このプラグインは、Amazon Machine Learning サービスを使用してコンテンツを 1 つ以上の言語に翻訳し、各翻訳の音声バージョンを生成して、Amazon Alexa デバイスを介して WordPress ウェブサイトを読み取ります。

このプラグインは、AWS の WordPress High Availability by Bitnami クイックスタートに既にインストールされています。

### プラグインのインストールと設定

プラグインをインストールするには:

1. AWS for WordPress プラグインを使用するには、プラグイン用の IAM ユーザーを作成する必要があります。IAM ユーザーは、AWS のサービスへの API コールを行うアクセス許可がある、AWS アカウントの下にあるユーザーまたはアプリケーションです。
2. AWS アカウントの認証と認可をコントロールするには、AWS Identity and Access Management (IAM) ロールまたは IAM ユーザーが必要です。許可のないユーザーがこれらのアクセス許可を得られないようにするために、IAM ユーザーの認証情報を保護します。シークレットアクセスキーはパスワードのように扱います。安全な場所に保存し、誰とも共有しないでください。パスワードと同様に、アクセスキーを定期的にローテーションします。シークレットアクセスキーが誤って漏れた場合は、すぐに削除してください。その後、AWS for WordPress プラグインで使用する新しいアクセスキーを作成できます。
3. WordPress 管理パネルの [Plugins] (プラグイン) メニューで、AWS for WordPress を検索し、[Install Now] (今すぐインストール) を選択します。




4. プラグインのインストールがうまくいかない場合は、ユーザーアクセス許可に問題がある可能性があります。WordPress ウェブサーバーに接続し、次の手順を実行して問題を解決します。

   a. WordPress のインストールディレクトリにある `wp-config.php` ファイルを開き、`wp-config.php` ファイルの最後に次のコードを記述します。

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

   b. 以下のコマンドを起動して、書き込みアクセス許可を付与します。

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

   警告: 書き込みアクセス許可を 777 にしておくのは危険です。アクセス許可が 777 のままの場合、誰でもこのフォルダを編集または削除できます。プラグインの作業が完了したら、書き込みアクセス許可を 755 以下に変更してください。

   c. リファレンスアーキテクチャを使用する場合、WordPress のインストールディレクトリは `/var/www/wordpress/<site directory>` です。

このドキュメントでは、AWS for WordPress 設定をすべて詳しく説明することはしません。設定とオプションについては、「AWS for WordPress プラグインの開始方法」を参照してください。

## Amazon CloudFront と AWS Certificate Manager

CloudFront と AWS Certificate Manager をセットアップするには:

1. プラグインメニューで [CloudFront] を選択し、以下のパラメータを入力します。
   - オリジンドメイン名: CloudFront がウェブサイトのコンテンツを取得する HTTP オリジンサーバーの DNS ドメイン (`example.com` など)。
   - 代替ドメイン名 (CNAME): 訪問者がウェブサイトエクスペリエンスを高速化するために使用するドメイン名。AWS では、ドメインの前に 「www」を使用することをお勧めします (`www.example.com` など)。
2. [Initiate Setup] (セットアップの開始) を選択して設定を開始します。

プラグインは ACM 経由で CNAME の SSL 証明書を自動的にリクエストします。CNAME エントリで DNS レコードを更新して ACM トークンを検証すると、付録 A で定義されているベストプラクティスを満たす CloudFront ディストリビューションが作成されます。

> **Note**
>
> AWS for WordPress プラグインでは、CloudFront とカスタムオリジン間の通信に HTTPS が必要です。オリジンドメイン名に対して有効な SSL 証明書がオリジンにあることを確認してください。詳細については、「CloudFront と使用するカスタムオリジン間の通信で HTTPS を必須にする」を参照してください。

## コンテンツの翻訳と音声化

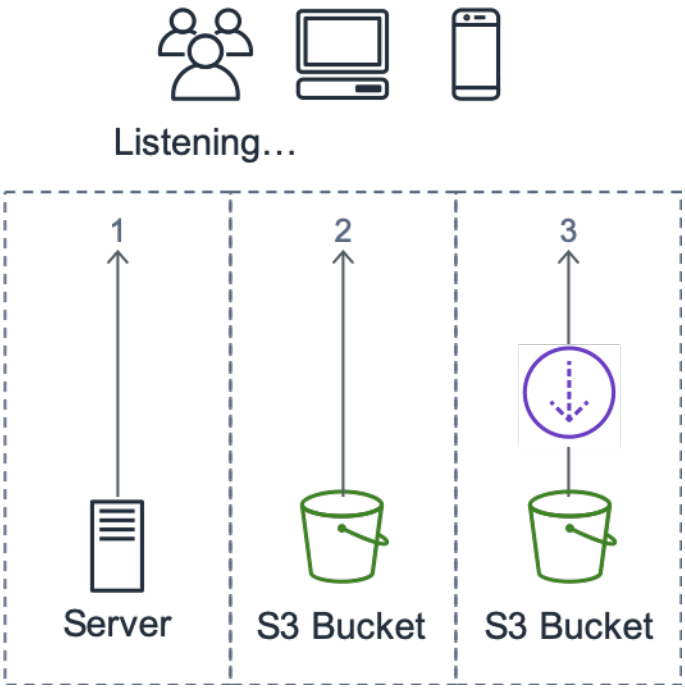
AWS for WordPress プラグインを使用すると、テキストをさまざまな言語に自動的に翻訳し、書かれたコンテンツを多言語オーディオ形式に変換できます。これらの機能は Amazon Machine Learning サービスによって強化されています。

Amazon Polly は、テキストをリアルな音声に変換するサービスです。何十種類もの音声を多数の言語でサポートしているため、最適な音声を選択して、さまざまな国で魅力的な音声対応アプリケーションを構築できます。このプラグインを使用して、Amazon Polly でサポートされている音声および言語のいずれかで、音声ファイルを作成します。訪問者は必要に応じて、インラインオーディオプレイヤーやモバイルアプリケーションを使用して音声をストリーミングできます。

デフォルトでは、プラグインは新しい音声ファイルをウェブサーバーに保存します。ファイルの保存先は Simple Storage Service (Amazon S3) または Amazon CloudFront のいずれかを選択できます。

ユーザーのリスニングエクスペリエンスは、音声ファイルの保存場所に関係なく同じです。ブロードキャストの場所が変更されるのみです。

- WordPress サーバーに保存された音声ファイルの場合、ファイルはサーバーから直接ブロードキャストされます。
- S3 バケットに保存されているファイルの場合、ファイルはそのバケットからブロードキャストされます。
- CloudFront を使用している場合、ファイルは Simple Storage Service (Amazon S3) に保存され、CloudFront でブロードキャストされます。



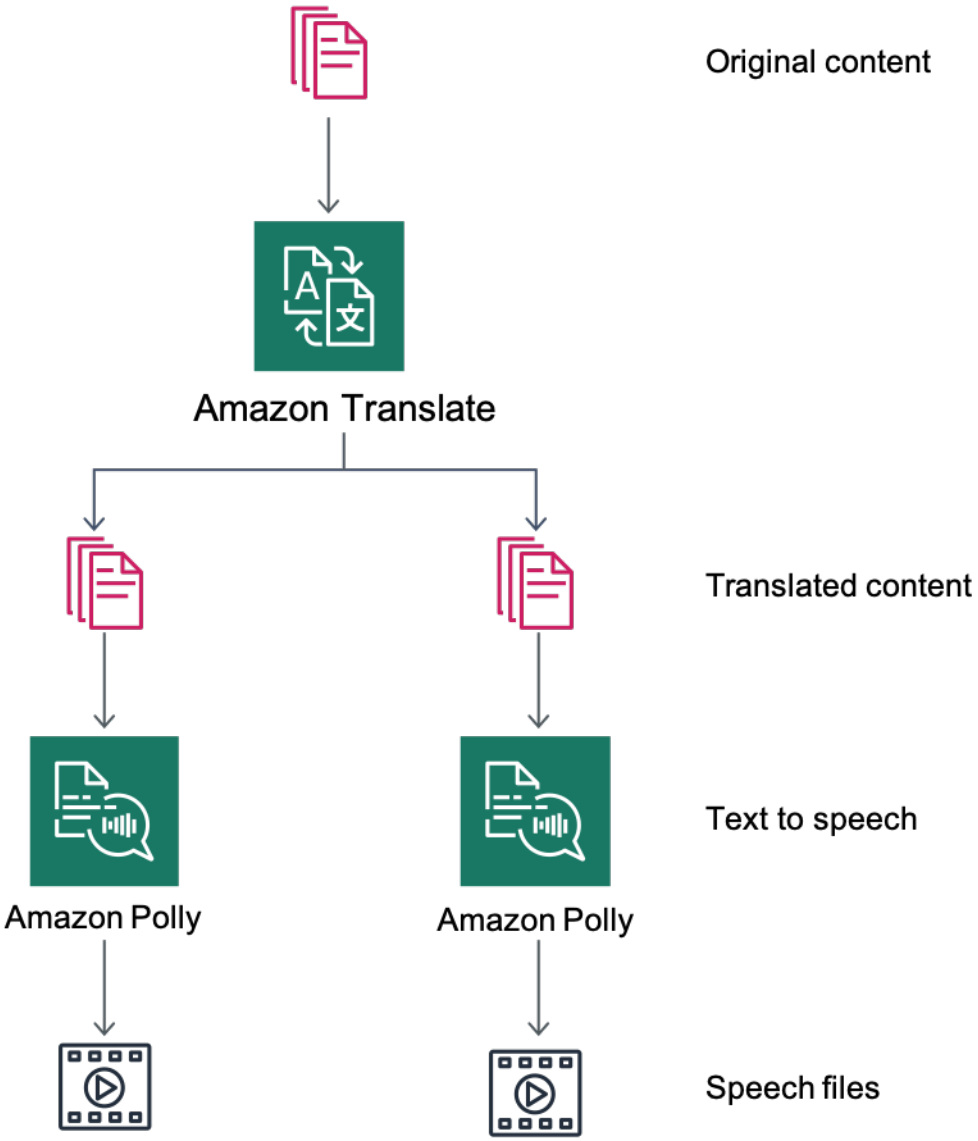

ブロードキャストロケーション

Amazon Translate は、高速で高品質な言語翻訳を手ごろな価格で提供する機械翻訳サービスです。多言語コンテンツを提供することは、サイト所有者にとって大きなチャンスです。英語はウェブの主要言語ですが、英語を母国語とする人はオンラインオーディエンス全体の 26% に過ぎません。

WordPress のテキスト版と音声版のコンテンツを複数の言語で提供することで、より多くの国際的な利用者のニーズを満たすことができます。プラグインを設定して、以下のことを実行できます。

- 新しいコンテンツの出版時に自動的に異なる言語に翻訳し、各翻訳の音声録音を作成する。または、個々の投稿を翻訳して録音を作成する
- アーカイブされたコンテンツを別の言語に翻訳し、翻訳ごとに音声録音を作成する

- Amazon Pollycast RSS フィードを使用して音声コンテンツをポッドキャストする




コンテンツ翻訳とテキスト読み上げの概要

## Amazon Pollycast を使用したポッドキャスト

Amazon Pollycast フィードを使用すると、ユーザーは標準のポッドキャストアプリケーションを使用して音声コンテンツを聴くことができます。RSS 2.0 準拠 Pollycast フィードは、iTunes など広く使用されているポッドキャストモバイルアプリケーションやポッドキャストディレクトリによるポッドキャスト集約に必要な XML データを提供します。

AWS for WordPress プラグインをインストールする際に、[Podcast] (ポッドキャスト) 設定タブに XML フィードの生成を有効にするオプションが表示されます。そこには、複数のオプションプロパティを設定するオプションもあります。機能を有効にすると、フィードを行うリンクが表示されます。

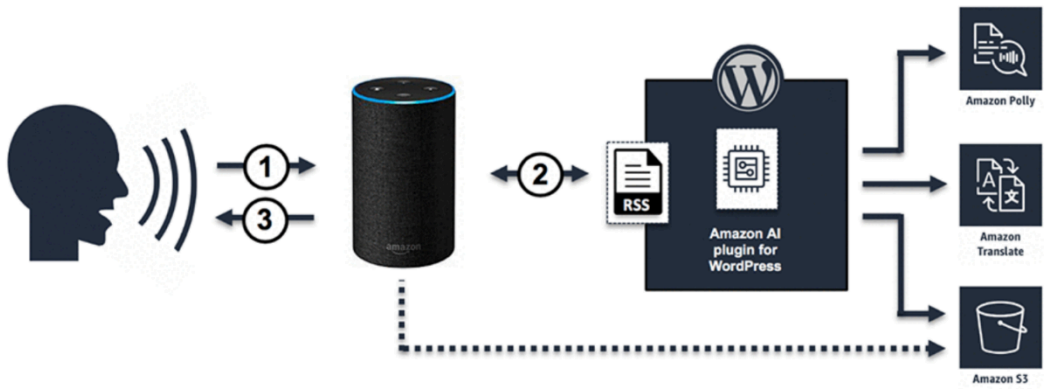
## Amazon Alexa デバイスでコンテンツを読む

WordPress のウェブサイトやブログは、Alexa デバイスを介して拡張できます。これにより、ウェブサイトの作成者や執筆者がさらに幅広いオーディエンスにリーチするための新しい可能性が開かれます。また、Alexa にお気に入りのブログの読み上げを頼むだけで、簡単に聴くことができます。

WordPress ウェブサイトを Alexa に公開するには、以下を有効にする必要があります。

- AWS for WordPress プラグイン。
- テキスト読み上げ機能と Amazon Pollycast 機能。これらの機能は、WordPress サイトに、Amazon Alexa によって消費される RSS フィードを生成します。
- Simple Storage Service (Amazon S3) がテキスト読み上げでファイルのデフォルトストレージであるため、ウェブサイトで安全な HTTPS 接続を使用してフィードを Alexa に公開することが重要です。

次の図表は、Alexa を通じてウェブサイトを公開するために必要なインタラクションとコンポーネントのフローを示しています。




WordPress ウェブサイトを Alexa で公開するために必要なインタラクションのフロー

1. ユーザーが、例えば「アレクサ、デモブログの最新記事を聞いて」と言って、新しい Alexa スキルを呼び出します。 スキル自体は、Alexa Skill ブループリントの 1 つを使用して作成されます。これにより、技術的な知識がなくても、Alexa デバイスを通じてスキルを公開できます。

2. Alexa スキルは、AWS for WordPress プラグインによって生成されたコールと RSS フィードを分析し、最新記事の音声バージョンへのリンクを返します。
3. フィードから提供されたリンクに基づいて、Alexa は Simple Storage Service (Amazon S3) に保存された音声ファイルを再生して記事を読み上げます。

プラグインとその機能をインストールして設定するための詳細なステップバイステップガイドについては、WordPress マーケットプレイスのプラグインページを参照してください。

# 静的コンテンツの設定

デフォルトでは、WordPress はすべてをウェブサーバーにローカルで保存します。ウェブサーバーはシングルサーバーデプロイの場合はブロックストレージ (Amazon EBS)、エラスティックデプロイではファイルストレージ (Amazon EFS) です。静的アセットを Simple Storage Service (Amazon S3) に移行すると、ストレージとデータ転送のコストが削減されるだけでなく、スケーラビリティ、データ可用性、セキュリティ、パフォーマンスが向上します。

この例では、W3 Total Cache (W3TC) プラグインを、Simple Storage Service (Amazon S3) 上の静的アセットを保存するために使用します。ただし、同様の機能を持つ他のプラグインも利用できます。別の方法を使用する場合は、次の手順を適宜調整できます。このステップでは、この例に直接関連した機能や設定についてのみ言及しています。このドキュメントでは、すべての設定を詳しく説明することはしません。詳細については、wordpress.org の W3 Total Cache プラグインについてのページを参照してください。

## IAM ユーザーの作成

Simple Storage Service (Amazon S3) に静的アセットを保存するには、WordPress プラグインの AWS Identity and Access Management (IAM) ユーザーを作成する必要があります。手順については、「AWS アカウントでの IAM ユーザーの作成」を参照してください。

注: AWS リソースへのアクセスを管理する点では IAM ロールの方が優れていますが、執筆の時点では W3 Total Cache プラグインは IAM ロールをサポートしていません。

ユーザーのセキュリティ認証情報を記録し、安全な方法で保管しておいてください。これらの認証情報は後ほど必要になります。

## Simple Storage Service (Amazon S3) バケットの作成

1. まず、選択した AWS リージョンに Simple Storage Service (Amazon S3) バケットを作成します。手順については、「バケットの作成」を参照してください。「チュートリアル: Amazon S3

での静的ウェブサイトの設定」に従って、バケットの静的ウェブサイトホスティングを有効にします。

2. 以前に作成した IAM ユーザーに指定された S3 バケットへのアクセス許可を付与する IAM ポリシーを作成して、そのポリシーを IAM ユーザーに添付します。以下のポリシーを作成する手順については、「IAM ポリシーを管理する」を参照してください。

```
{
"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. W3TC プラグインをインストールし、WordPress の管理パネルからアクティブにします。
4. プラグインの設定の [General Settings] (一般設定) セクションに移動し、[Browser Cache] (ブラウザキャッシュ) と [CDN] の両方が有効になっていることを確認します。
5. CDN 設定のドロップダウンリストから、[Origin Push: Amazon CloudFront] (オリジンプッシュ: Amazon CloudFront) を選択します (このオプションでは Simple Storage Service (Amazon S3) がオリジンになります)。
6. プラグインの設定の [Browser Cache] (ブラウザキャッシュ) セクションに移動し、[expires] (有効期限)、[cache control] (キャッシュコントロール)、および [entity tag (ETag)] (エンティティタグ (ETag)) のヘッダーを有効にします。

7. また、[Prevent caching of objects after settings change] (設定変更後のオブジェクトのキャッシュを防止する) オプションをアクティブにし、設定が変更されたときには必ず新しいクエリ文字列が生成され、オブジェクトの後に付加されるようにします。
8. プラグインの設定の [CDN] セクションに移動し、前に作成した IAM ユーザーのセキュリティ認証情報、および S3 バケットの名前を入力します。
9. ウェブサイトを CloudFront の URL で提供する場合には、関連するボックスにディストリビューション用のドメイン名を入力します。そうでない場合には、カスタムのドメイン名に応じた 1 つ以上の CNAME を入力します。
10. 最後に、メディアライブラリをエクスポートし、WP のインクルードファイル、テーマファイル、およびカスタムのファイルを、W3TC プラグインを使用して Simple Storage Service (Amazon S3) にアップロードします。これらのアップロード機能は、[CDN] 設定ページの [General] (一般) セクションから使用できます。

## 静的オリジンの作成

これで静的ファイルを Simple Storage Service (Amazon S3) に保存したので、CloudFront コンソールの CloudFront 設定に戻り、Simple Storage Service (Amazon S3) を静的コンテンツのオリジンとして設定します。そのために、この目的で作成した S3 バケットを指す 2 番目のオリジンを追加します。それから、さらに 2 つのキャッシュビヘイビアを追加します。それぞれは、動的コンテンツのデフォルトオリジンではなく、S3 オリジンを使用する 2 つのフォルダ (`wp-content` と `wp-includes`) のためのものです。両方を、次の同じ方法で設定します。

- HTTP GET リクエストのみを提供する。
- Simple Storage Service (Amazon S3) は cookie や HTTP ヘッダーに基づいて出力を変化させないため、CloudFront 経由でオリジンに転送しないことで、キャッシュの効率を向上させることができます。
- これらのビヘイビアは静的コンテンツ (パラメータを受け取らない) のみを処理しますが、クエリ文字列はオリジンに転送します。これは、クエリ文字列をバージョン識別子として使用して、例えば新しいバージョンをデプロイするときに古い CSS ファイルなどを即座に無効にするためです。詳細は、Amazon CloudFront デベロッパーガイドをご覧ください。

> **Note**
>
> CloudFront ディストリビューションに静的オリジンビヘイビアを追加したら、順序をチェックして、`wp-admin/*` と `wp-login.php` に対するビヘイビアが静的コンテンツに対するビ

ヘイビアよりも優先されることを確認してください。そうなっていなかった場合、管理者パネルにアクセスしたときに奇妙な動作をする可能性があります。

# 付録 C: バックアップと復旧

AWS での障害からの復旧は、従来のホスティング環境と比較して、高速かつ容易に行うことができます。例えば、ハードウェア障害が発生した場合には、数分以内に代わりのインスタンスを起動できます。または、多くのマネージドサービスで提供されている自動フェイルオーバーを活用すれば、定期的なメンテナンスに伴う再起動の影響をなくすことができます。

ただし、データを正常に復旧するには、正しいデータをバックアップしていることを確認する必要があります。WordPress ウェブサイトの可用性を再確立するたは、次のコンポーネントの復旧が可能になっている必要があります。

- オペレーティングシステム (OS) およびサービスのインストール先と設定 (Apache、MySQL、など)。
- WordPress アプリケーションコードと設定
- WordPress テーマとプラグイン
- アップロードされたコンテンツ (記事のメディアファイルなど)
- データベースコンテンツ (記事、コメントなど)

AWS では、ウェブアプリケーションのデータとアセットをバックアップおよび復元するためのさまざまな方法を提供しています。

Lightsail スナップショットを活用して、インスタンスのローカルストレージに保存されているすべてのデータを保護する方法については、このホワイトペーパーで既に説明しました。WordPress ウェブサイトが Lightsail インスタンスのみを使用して動作している場合には、定期的な Lightsail のスナップショットだけで、WordPress ウェブサイトの全体を復旧するには十分です。ただし、1 つのスナップショットから復元する場合には、スナップショットを最後に作成した後でウェブサイトに適用された変更はやはり失われてしまいます。

マルチサーバーデプロイでは、前述した各コンポーネントを、別のメカニズムを使用してバックアップする必要があります。それぞれのコンポーネントには、バックアップの頻度などの点で異なる要件があります。例えば、OS と WordPress のインストール先と設定の内容が変更される頻度は、ユーザーが生成するコンテンツよりもずっと少ないはずなので、バックアップの頻度が少なくても、復旧時にデータを失うことはないでしょう。

OS とサービスのインストール先と設定、および WordPress のアプリケーションコードと設定をバックアップするために、適切に設定された EC2 インスタンスの AMI を作成できます。AMI には 2

つの目的があります。インスタンスの状態のバックアップとしての役割と、新しいインスタンスを起動するときのテンプレートとしての役割です。

WordPress のアプリケーションコードと設定をバックアップするには、AMI と Aurora バックアップを活用する必要があります。

WordPress のテーマとウェブサイトにインストールしたプラグインをバックアップするには、それらが置かれている Simple Storage Service (Amazon S3) バケットまたは Amazon EFS ファイルシステムをバックアップします。

- S3 バケットに保存されているテーマとプラグインについては、クロスリージョンレプリケーションを有効にして、プライマリバケットにアップロードされたすべてのオブジェクトが別の AWS リージョンのバックアップバケットに自動的にレプリケートされるようできます。クロスリージョンレプリケーションでは、レプリケート元バケットとレプリケート先バケットの両方でバージョニングを有効にする必要があります。これにより、追加の保護レイヤーが提供され、バケット内の任意のオブジェクトの以前のバージョンに戻すことができます。
- EFS ファイルシステムに保存されているテーマとプラグインの場合、Backing up your Amazon EFS file systems のドキュメントページで説明されているように、本番環境の EFS ファイルシステムから別の EFS ファイルシステムにデータをコピーするための AWS Data Pipeline を作成できます。また、使い慣れたバックアップアプリケーションを使用して、EFS ファイルシステムをバックアップすることもできます。
- ユーザーがアップロードしたファイルについては、前述の、WordPress のテーマとプラグインのバックアップについて説明したステップに従ってください。
- データベースコンテンツをバックアップするには、Aurora バックアップを使用する必要があります。Aurora は、クラスターボリュームを自動的にバックアップし、バックアップ保持期間分の復元データを保持できます。Aurora のバックアップ機能は継続して行われる増分式のものなので、バックアップ保持期間内であれば、任意の時点の状態を素早く復元できます。バックアップデータが書き込まれるときに、データベースサービスのパフォーマンスに影響が出たり、中断が発生したりすることはありません。バックアップ保持期間は 1 ~ 35 日の範囲で指定できます。また、手動でデータベーススナップショットを作成することもできます。これは削除されるまで保持されます。手動によるデータベーススナップショットは、長期間のバックアップとアーカイブに役立ちます。

# 付録 D: 新しいプラグインとテーマのデプロイ

完全に静的なウェブサイトはほとんどありません。ほとんどの場合、公開されている WordPress テーマとプラグインを定期的に追加するか、新しい WordPress バージョンにアップグレードします。独自のカスタムテーマやプラグインをゼロから作成することもあるでしょう。

WordPress のインストール内容に構造的な変化を加えた場合、予期しない問題が発生するリスクがある程度存在します。少なくとも、重要な変更 (新しいプラグインのインストールなど) を適用する前に、アプリケーションコード、設定、データベースのバックアップを作成してください。ビジネスやその他の価値のあるウェブサイトの場合は、まず別のステージング環境でこれらの変更をテストします。AWS では、本番環境の設定のレプリケーションを簡単に作成し、デプロイプロセス全体を安全な方法で実行できます。テストが完了したら、テスト環境を破棄して、そのリソースに対する課金を停止できます。後ほどこのホワイトペーパーで、WordPress 固有の考慮事項について説明します。

プラグインの中には、設定情報を `wp_options` データベーステーブルに書き込む (またはデータベーススキーマの変更を導入する) ものがありますが、WordPress のインストールディレクトリに設定ファイルを作成するものもあります。データベースとストレージは共有プラットフォームに移動したので、これらの変更は、特に作業を行わなくても、実行中のすべてのインスタンスで直ちに利用できるようになります。

WordPress に新しいテーマをデプロイする際には、いくらかの作業が必要になることがあります。Amazon EFS にすべての WordPress インストールファイルを保存している場合には、新しいテーマは実行中のすべてのインスタンスで直ちに利用できるようになります。しかし、静的コンテンツを Simple Storage Service (Amazon S3) にオフロードしている場合には、これらのコピーを適切なバケットロケーションに処理する必要があります。W3 Total Cache のようなプラグインは、このタスクを手動で開始する方法を提供しています。または、このステップを構築プロセスの一部として自動化することもできます。

テーマのアセットは CloudFront とブラウザにキャッシュできるため、変更をデプロイする際には古いバージョンを無効にする方法が必要になります。これを実現するベストな方法は、オブジェクトに何らかのバージョン識別子を含めておくことです。この識別子には、日付/タイムスタンプ付きのクエリ文字列や、ランダムな文字列を指定できます。W3 Total Cache プラグインを使用する場合には、メディアファイルの URL に追加する、メディアクエリ文字列を更新することができます。

# 注意

お客様は、この文書に記載されている情報を独自に評価する責任を負うものとします。本書は、(a) 情報提供のみを目的とし、(b) AWS の現行製品と慣行について説明しており、これらは予告なしに変更されることがあり、(c) AWS およびその関連会社、サプライヤーまたはライセンサーからの契約上の義務や保証をもたらすものではありません。AWS の製品やサービスは、明示または暗示を問わず、一切の保証、表明、条件なしに「現状のまま」提供されます。お客様に対する AWS の責任は、AWS 契約により規定されます。本書は、AWS とお客様の間で締結されるいかなる契約の一部でもなく、その内容を修正するものでもありません。