AWS ホワイトペーパー

WordPress での のベストプラクティス AWS

WordPress での のベストプラクティス AWS: AWS ホワイトペーパー

Table of Contents

要約 1
Well-Architected の実現状況の確認 1
はじめに 2
シンプルなデプロイ 3
考慮事項 3
使用可能なアプローチ 3
Amazon Lightsail 4
Amazon Lightsail 料金プランの選択 4
のインストール WordPress 5
障害からの復旧 5
パフォーマンスとコスト効率を改善する 7
コンテンツ配信の加速 7
静的コンテンツのオフロード 8
動的コンテンツ 8
データベースのキャッシング 9
バイトコードキャッシング 10
Elastic デプロイ 11
リファレンスアーキテクチャ 11
ウェブ層のスケーリング 13
ステートレスウェブ層 14
共有ストレージ (Amazon S3 と Amazon EFS) 15
データ階層 (Amazon Aurora と Amazon ElastiCache) 16
まとめ 18
寄稿者 19
ドキュメントの改訂 20
付録 A: CloudFront 設定 21
オリジンと動作 21
CloudFront ディストリビューションの作成 21
付録 B: 静的コンテンツ設定 25
ユーザーの作成 25
Amazon S3 バケット作成 25
静的オリジンの作成 27
付録 C: バックアップと復旧 28
付録 D: 新しいプラグインとテーマのデプロイ 30

注意 31
AWS 用語集 32
..... xxxiii

WordPress での のベストプラクティス AWS

公開日: 2021 年 10 月 19 日 (ドキュメントの改訂)

このホワイトペーパーでは、Amazon Web Services (AWS) WordPress で を使い始める方法と、デプロイのコスト効率とエンドユーザーエクスペリエンスの両方を向上させる方法に関する具体的なガイダンスをシステム管理者に提供します。また、一般的なスケーラビリティと高可用性要件に対処するリファレンスアーキテクチャについても概説しています。

Well-Architected の実現状況の確認

AWS Well-Architected フレームワークは、クラウド内でのシステム構築に伴う意思決定の長所と短所を理解するのに役立ちます。このフレームワークの 6 つの柱により、信頼性、安全性、効率、費用対効果、持続可能性の高いシステムを設計および運用するための、アーキテクチャのベストプラクティスを確認できます。AWS マネジメントコンソール で無料で提供されている AWS Well-Architected Tool を使用すると、柱ごとに一連の質問に答えることで、これらのベストプラクティスに照らしてワークロードを評価できます。

クラウドアーキテクチャに関する専門的なガイダンスやベストプラクティス (リファレンスアーキテクチャのデプロイ、図、ホワイトペーパー) については、AWS アーキテクチャセンターを参照してください。

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

トラフィックの少ないブログや、高可用性に関する厳格な要件のないウェブサイトでは、単一のサーバーを簡単にデプロイするのが適している場合があります。このデプロイは、最も回復力やスケーラブルなアーキテクチャではありませんが、ウェブサイトを稼働させる最も迅速かつ経済的な方法です。

トピック

- 考慮事項
- 使用可能なアプローチ
- Amazon Lightsail

考慮事項

この説明は、1 つのウェブサーバーのデプロイから始まります。場合によっては、次のような状況が発生することがあります。

- WordPress ウェブサイトがデプロイされている仮想マシンは、単一障害点です。このインスタンスに問題があると、ウェブサイトのサービスが失われます。
- 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」、つまり WordPress ウェブサイトを実行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。

使用可能なアプローチ

AWS には、仮想マシンをプロビジョニングするためのさまざまなオプションがあります。独自の WordPressウェブサイトを でホストするには、主に 3 つの方法がありますAWS。

- 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ウェブサイトを WordPressホストできるサービスです。Lightsail は、高度に設定可能なインスタンスタイプや高度なネットワーク機能へのアクセスが必要ない場合に、最も簡単に開始できます。

Amazon EC2 は、数分以内に仮想サーバーを起動できるように、サイズ変更可能なコンピューティング容量を提供するウェブサービスです。Amazon EC2は Lightsail よりも多くの設定と管理オプションを提供します。これは、より高度なアーキテクチャで理想的です。EC2 インスタンスへの管理アクセス権があり、 を含む任意のソフトウェアパッケージをインストールできます WordPress。

AWS Marketplace は、 で実行されるソフトウェアを検索、購入、すばやくデプロイできるオンラインストアですAWS。1-Clickデプロイを使用して、事前設定された WordPress イメージを自分のAWSアカウントEC2で Amazon に直接わずか数分で起動できます。 WordPress インスタンスを提供する ready-to-run Marketplace ベンダーは多数あります。

このホワイトペーパーでは、単一のサーバー WordPress ウェブサイトの推奨実装として Lightsail オプションについて説明します。

Amazon Lightsail

Lightsail は、シンプルな仮想プライベートサーバー (VPS) ソリューションを必要とするデベロッパー、中小企業、学生、その他のユーザーAWSにとって最も簡単な開始方法です。

このサービスは、インフラストラクチャ管理のより複雑な要素の多くをユーザーから抽象化します。したがって、インフラストラクチャの使用経験が少ない場合や、ウェブサイトの実行に集中する必要があり、簡素化された製品で十分な場合、これは理想的な出発点です。

Amazon Lightsail では、Windows または Linux/Unix オペレーティングシステム、および などの一般的なウェブアプリケーションを選択し WordPress、事前設定されたテンプレートからワンクリックでデプロイできます。

ニーズが拡大するにつれて、初期境界からスムーズに踏み出し、追加のAWSデータベース、オブジェクトストレージ、キャッシュ、コンテンツ配信サービスに接続できるようになります。

Amazon Lightsail 料金プランの選択

Lightsail プランは、 WordPress ウェブサイトのホストに使用する Lightsail リソースの月額コストを定義します。リソース、メモリ、ソリッドステートドライブ (SSD) ストレージCPU、データ転送のレベルが異なるさまざまなユースケースをカバーするための計画が多数あります。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より大きなインスタンスが必要になる場合があります。これを実現するには、ウェブコンソールを使用して、または Amazon Lightsail CLIドキュメント で説明されているように、サーバーをより大きなプランに移行します。

のインストール WordPress

Lightsail は、 などの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を提供します WordPress。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ているため、独自の WordPress ウェブサイトを実行するための素晴らしい出発点です。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するか、管理ウェブインターフェイスを使用して WordPress、追加のソフトウェアをインストールしたり、ソフトウェア設定をカスタマイズしたりできます。

Amazon Lightsail は Pro Sites GoDaddy 製品とパートナーシップを結び、 WordPress お客様がインスタンスを無料で簡単に管理できるようにします。Lightsail WordPress 仮想サーバーは、高速なパフォーマンスとセキュリティのために事前設定および最適化されているため、 WordPress サイトをすぐに起動して実行できます。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、保守、管理するのが難しく、時間がかかります。この統合により、数回のクリックだけで、複数の WordPress インスタンスを数分で簡単に管理できます。

インストール後の Lightsail WordPress での の管理の詳細については、Amazon Lightsail インスタンス WordPress からの使用開始」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、元のマシン設定 (メモリ、、CPUディスクサイズ、データ転送レート) も保存されます。スナップショットは、不適切なデプロイまたはアップグレード後に、既知の適切な設定に戻すために使用できます。

このスナップショットを使用すると、必要に応じてサーバーを復旧できますが、同じカスタマイズで新しいインスタンスを起動することもできます。

障害からの復旧

単一のウェブサーバーは単一の障害点であるため、ウェブサイトデータがバックアップされていることを確認する必要があります。前述のスナップショットメカニズムは、この目的でも使用できます。失敗から復旧するには、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できます。復元中に失われる可能性のあるデータの量を減らすには、スナップショットをできるだけ最新にする必要があります。

データ損失の可能性を最小限に抑えるために、スナップショットを定期的に作成していることを確認してください。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。

ステップについては、Amazon Lightsail 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 は、静的 IP を使用することをお勧めします。これは、Lightsail アカウント専用の固定パブリック IP アドレスです。インスタンスを別のインスタンスに置き換える必要がある場合は、静的 IP を新しいインスタンスに再割り当てできます。これにより、インスタンスを置き換えるたびに新しい IP アドレスを指すように外部システム (DNSレコードなど) を再設定する必要はありません。

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。

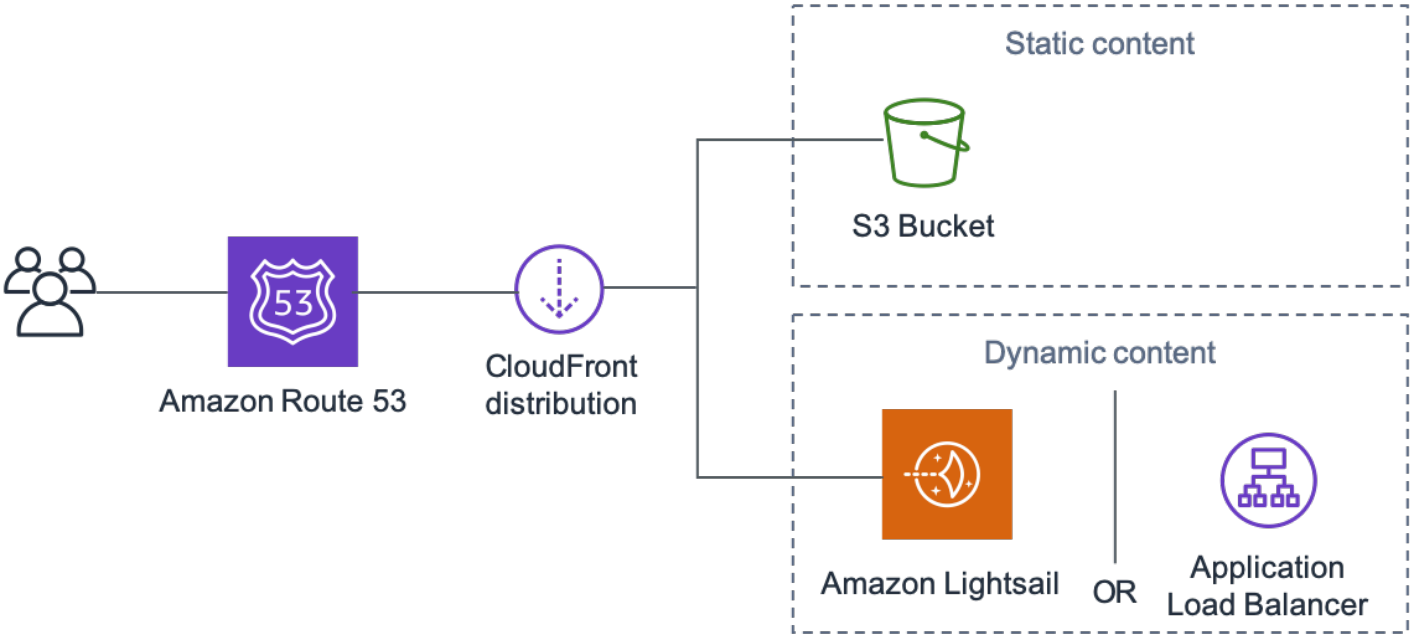
これには有益な副作用もあります。Lightsail インスタンスのワークロードをオフロードできるので、インスタンスは動的なコンテンツの生成に専念できます。これにより、サーバーの負荷が軽減され、ステートレスアーキテクチャ (オートスケーリングを実装する前の前提条件) を作成するための重要なステップとなります。

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。

Elastic デプロイ

単一サーバーのデプロイがウェブサイトにとって十分ではないシナリオが多数あります。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。

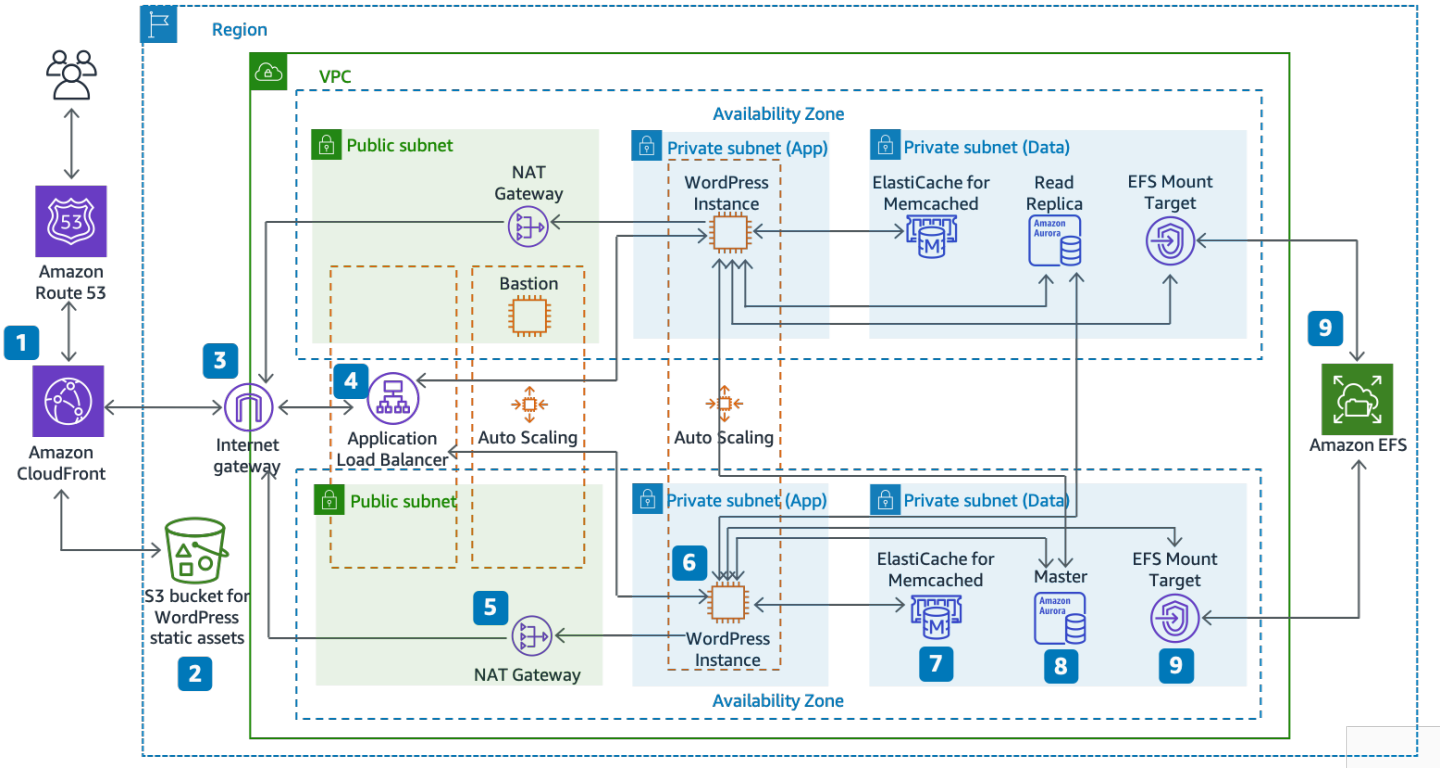
トピック

- リファレンスアーキテクチャ
- ウェブ層のスケーリング
- ステートレスウェブ層

リファレンスアーキテクチャ

で GitHub利用可能な AWS 参照アーキテクチャ WordPress のホスティングには、 WordPress にデプロイするためのベストプラクティスの概要AWSと、すぐに起動して実行するための AWS CloudFormation テンプレートの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、アーキテクチャの選択の背後にある理由を確認します。

AMI の ベースは、202Linux1 年 7 月に Amazon Linux1 から Amazon Linux2 に変更 GitHub 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 GitHub がある場合は、 でテンプレートを使用することをお勧めします。



WordPress でホストするためのリファレンスアーキテクチャ AWS

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は、 WordPress のウェブサイトの完全なベストプラクティスデプロイを示していますAWS。

- Amazon (1) での CloudFrontエッジキャッシュから始めて、エンドユーザーに近いコンテンツをキャッシュし、配信を高速化します。
- CloudFront は、S3 バケット (2) から静的コンテンツを取得し、ウェブインスタンスの前に Application Load Balancer (4) から動的コンテンツを取得します。
- 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) で実行されます。
- ElastiCache クラスター (7) は、頻繁にクエリされ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) はデータベースを WordPressホストします。
- インスタンスは WordPress EC2、各アベイラビリティーゾーンのEFSマウントターゲット (9) を介して Amazon EFS ファイルシステムの共有 WordPress データにアクセスします。
- Internet Gateway (3) では、 VPCとインターネット内のリソース間の通信が可能になります。

- 各アベイラビリティーゾーンのNATゲートウェイ (5) は、プライベートサブネット (アプリとデータ) のEC2インスタンスがインターネットにアクセスできるようにします。

Amazon VPC 内には、パブリック (パブリックサブネット) とプライベート (アプリケーションサブネットとデータサブネット) の 2 種類のサブネットがあります。パブリックサブネットにデプロイされたリソースはパブリック IP アドレスを受け取り、インターネット上で公開されます。管理用の Application Load Balancer (4) と Bastion ホストが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はプライベート IP アドレスのみを受け取るため、インターネットでは公開されないため、これらのリソースのセキュリティが向上します。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)、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、これらの各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、次の 5 つの主要な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 (AMIs)
- ロードバランサー
- Auto Scaling
- ヘルスチェック

AWS は、パフォーマンスとコストの両方に最適なサーバー設定を選択できるように、さまざまなEC2インスタンスタイプを提供します。一般的に、コンピューティング最適化 (C4 など) インスタンスタイプは、 WordPress ウェブサーバーにとって適切な選択です。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。

EC2 インスタンスを完全に制御できるため、ルートアクセスを使用してログインして、 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 として保存できます。これを使用してAMI、行ったすべてのカスタマイズで新しいインスタンスを起動できます。

エンドユーザーリクエストを複数のウェブサーバーノードに配信するには、負荷分散ソリューションが必要です。AWS は、トラフィックを複数のEC2インスタンスに分散する高可用性サービスである Elastic Load Balancingを通じてこの機能を提供します。ウェブサイトは HTTPまたは を介してユーザーにコンテンツを提供しているためHTTPS、コンテンツルーティングと、必要に応じて異なるドメインで複数の WordPress ウェブサイトを実行する機能を備えたアプリケーションレイヤーロードバランサーである Application Load Balancer を使用することをお勧めします。

Elastic Load Balancing は、AWSリージョン内の複数のアベイラビリティーゾーンにリクエストを分散できます。Application Load Balancer が失敗した個々のインスタンスへのトラフィックの送信を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 (ハードウェアの問題やソフトウェアクラッシュなど) 。AWS は、ヘルスチェックに WordPress 管理者ログインページ (`/wp-login.php`) を使用することをお勧めします。このページでは、ウェブサーバーが実行中であることと、ウェブサーバーがPHPファイルを正しく処理するように設定されていることの両方が確認されます。

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ヘルスチェックページを構築できます。詳細については、Application Load Balancer 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。

Elasticity は AWS クラウドの主要な特徴です。必要なときにより多くのコンピューティングキャパシティ (ウェブサーバーなど) を起動し、必要ないときに実行を減らすことができます。Amazon EC2 Auto Scaling は、このプロビジョニングを自動化して、手動で介入することなく、定義した条件に従って Amazon EC2容量をスケールアップまたはスケールダウンするのに役立つ AWS サービスです。Amazon EC2 Auto Scaling を設定して、需要の急増時に使用する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少するようにすることで、コストを最小限に抑えることができます。

Elastic Load Balancing は、ロードバランシングローテーションからの Amazon EC2ホストの動的な追加と削除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て、手動による介入なしでトラフィックの需要に適応します。

ステートレスウェブ層

自動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。ステートレスアプリケーションは、以前のインタラクションに関する知識を必要とせず、セッション情報を保存しないアプリケーションです。の場合 WordPress、これは、どのウェブサーバーがリクエストを処理したかに関係なく、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ます。ステートレスアプリケーションは水平方向にスケーリングできます。これは、

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) によってリクエストを処理できるためです。その容量が不要になった場合、個々のリソースを安全に終了できます (実行中のタスクがドレインされた後) 。これらのリソースは、ピアの存在を認識する必要はありません。必要なのは、ワークロードを分散する方法だけです。

ユーザーセッションデータストレージの場合、 WordPress コアはクライアントのウェブブラウザに保存されている Cookie に依存するため、完全にステートレスです。代わりにネイティブセッションに依存するカスタムコード (WordPress プラグインなど) をインストールしない限り、PHPセッションストレージは問題ではありません。

ただし、当初 WordPress は 1 つのサーバーで実行されるように設計されています。その結果、一部のデータはサーバーのローカルファイルシステムに保存されます。マルチサーバー設定 WordPress で を実行すると、ウェブサーバー間で不整合があるため、問題が発生します。例えば、ユーザーが新しいイメージをアップロードした場合、そのイメージはいずれかのサーバーにのみ保存されます。

これは、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を示しています。ベストプラクティスアーキテクチャには、データベースがウェブサーバー外の別のレイヤーとしてあり、共有ストレージを使用してユーザーアップロード、テーマ、プラグインを保存します。

共有ストレージ (Amazon S3 と Amazon EFS)

デフォルトでは、 は WordPress ユーザーアップロードをローカルファイルシステムに保存するため、ステートレスではありません。したがって、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には、 WordPress インストールとすべてのユーザーカスタマイズ (設定、プラグイン、テーマ、ユーザー生成アップロードなど) 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、無制限の数のストレージサーバーに分散されているため、ファイルシステムは伸縮自在に成長し、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 の分散設計は、従来のファイルサーバーに固有のボトルネックや制約EFSを回避します。

WordPress 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動し、起動時に各EC2インスタンスにマウントすることで、 WordPress サイトとそのすべてのデータは、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自動的に保存され、ウェブ層が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、新しいインスタンスの起動ごとにプラグインやテーマをインストールしなくても、 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。また、このドキュメントのデプロイに関する考慮事項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に変更をデプロイする方が簡単です。

EFS ファイルシステムから実行するときにウェブサイトのパフォーマンスを最適化するには、 のリファレンスアーキテクチャOPcacheで Amazon EFSおよび の推奨設定を確認してください。 AWS WordPress

また、イメージ、、CSS JavaScript ファイルなどのすべての静的アセットを、 CloudFront キャッシュを前にして S3 バケットにオフロードすることもできます。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、単一サーバーアーキテクチャの場合とまったく同じです。利点は、単一サーバーアーキテクチャの場合と同じです。静的アセットの提供に関連する作業を Amazon S3 および にオフロードできるため CloudFront、ウェブサーバーは動的コンテンツのみの生成に集中し、ウェブサーバーあたりのユーザーリクエスト数を増やすことができます。

データ階層 (Amazon Aurora と Amazon ElastiCache)

Amazon S3 から提供される分散されたスケーラブルな共有ネットワークファイルシステムおよび静的アセットに WordPress インストールを保存すると、残りのステートフルコンポーネントであるデータベースに注意を払うことができます。ストレージ層と同様に、データベースは単一のサーバーに依存しないため、いずれかのウェブサーバーでホストす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ます。

Amazon Aurora は、クラウド用に構築された MySQL および PostgreSQL 互換のリレーショナルデータベースです。ハイエンドの商用データベースのパフォーマンスと可用性と、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を兼ね備えています。Aurora MySQL は、データベースエンジンを にバックアップされた専用の分散ストレージシステムと緊密に統合することで、パフォーマンスSQLと可用性を向上させますSSD。耐障害性と自己修復性を備え、3 つのアベイラビリティーゾーンに 6 つのデータの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、Amazon S3 でデータを継続的にバックアップします。Amazon Aurora は、クラッシュリカバリやデータベースキャッシュの再構築を必要とせずに、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。

Amazon Aurora には、メモリ最適化インスタンスやバースト可能なインスタンスなど、さまざまなアプリケーションプロファイルに適したインスタンスタイプが多数用意されています。データベースのパフォーマンスを向上させるには、大規模なインスタンスタイプを選択して、より多くの CPU および メモリリソースを提供できます。

Amazon Aurora は、プライマリインスタンスと Aurora レプリカ間のフェイルオーバーを自動的に処理するため、アプリケーションは手動による管理介入なしでデータベースオペレーションをできるだけ早く再開できます。フェイルオーバーは通常 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が失敗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で最大 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すると、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。データベースキャッシュセクションで前述したように、 には、可用性を向上させるために、 ElastiCache クラスター内の複数のノードとリージョン内の複数のアベイラビリティーゾーンにキャッシュをスケーリングする機能 ElastiCache があります。 ElastiCache クラスターをスケーリングするときは、設定エンドポイントを使用して接続するようにキャッシュプラグインを設定し、 が新しいクラスターノードを追加時に WordPress 使用できるようにし、古いクラスターノードを削除時に使用を停止します。また、 ElastiCache のクラスタークライアントPHPを使用するようにウェブサーバーを設定し、この変更を保存するAMIように を更新する必要があります。

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

ドキュメントの改訂

このホワイトペーパーの更新について通知を受け取るには、RSSフィードをサブスクライブします。

変更	説明	日付
ホワイトペーパーの更新	WordPress プラグインAWSのリファレンスアーキテクチャと を変更するために更新されました。	2021 年 10 月 19 日
ホワイトペーパーの更新	WordPress プラグインAWS用の新しいデプロイアプローチと を含めるように更新しました。	2019 年 10 月 30 日
ホワイトペーパーの更新	Amazon Aurora 製品メッセージングを明確にするために更新されました。	2018 年 2 月 1 日
ホワイトペーパーの更新	最初の発行以降に開始されたAWSサービスを含めるように更新されました。	2017 年 12 月 1 日
初版発行	初めて公開されました。	2014 年 12 月 1 日

付録 A: CloudFront 設定

WordPress ウェブサイト CloudFront で Amazon を使用する際に最適なパフォーマンスと効率を得るには、提供されるさまざまなタイプのコンテンツに合わせてウェブサイトを正しく設定することが重要です。

トピック

- オリジンと動作
- CloudFront ディストリビューションの作成

オリジンと動作

オリジンは、エッジロケーションを介して配信されるコンテンツに対するリクエスト CloudFront を送信する場所です。実装に応じて、1 つまたは 2 つのオリジンを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 (単一サーバーデプロイオプション の Lightsail インスタンス、または Elastic デプロイオプション の Application Load Balancer) 用です。静的コンテンツ CloudFront に対して にリダイレクトする 2 番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。Amazon S3 をディストリビューションのオリジンとして使用する場合は、バケットポリシーを使用してコンテンツをパブリックにアクセスできるようにする必要があります。

動作により、 がコンテンツを CloudFront キャッシュする方法を制御するルールを設定し、キャッシュの効果を決定できます。動作により、ウェブサイトがアクセスできるプロトコルとHTTP方法を制御できます。また、HTTPヘッダー、Cookie、またはクエリ文字列をバックエンドに渡すかどうか (渡す場合はどの文字列を渡すか) を制御できます。動作は特定の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すると、自動的に作成されたデフォルトのオリジンと動作が動的コンテンツに使用されます。静的リクエストと動的リクエストの両方の処理方法をさらにカスタマイズするために、4 つの追加の動作を作成します。次の表は、5 つの動作の設定プロパティをまとめたものです。

表 1: CloudFront 動作の設定プロパティの概要

プロパティ	静的	動的 (管理者)	動的 (フロントエンド)
パス (動作)	`wp-content/*` `wp-includes/*`	`wp-admin/*` `wp-login.php`	デフォルト (*)
プロトコル	HTTP および HTTPS	リダイレクト先 HTTPS	HTTP および HTTPS
HTTP メソッド	GET, HEAD	ALL	ALL
HTTP ヘッダー	NONE	ALL	ホスト CloudFront-Forwarded-Proto CloudFront-Is-Mobile-Viewer CloudFront-Is-Tablet-Viewer CloudFront-Is-Desktop-Viewer
cookie	NONE	ALL	comment_* wordpress_* wp-settings-*
クエリ文字列	YES (無効化)	YES	YES

デフォルトの動作では、 は次の設定AWSを推奨します。

- オリジンプロトコルポリシーを Match Viewer に許可すると、ビューワーが を使用して CloudFrontに接続するとHTTPS、 HTTPSも を使用してオリジン CloudFront に接続し、暗号化を実現 end-to-endできます。そのためには、信頼できるSSL証明書をロードバランサーにインストー

ルする必要があることに注意してください。詳細については、「 CloudFront とカスタムオリジン間の通信の要件HTTPS」を参照してください。

- ウェブサイトの動的部分には GETと の両方のPOSTリクエスト (コメントPOST送信フォームをサポートするなど) が必要なため、すべてのHTTPメソッドを許可します。
- 、、 などwp-settings-*、 WordPress 出力を変更する Cookie >wordpress_*のみを転送しますcomment_*。リストに含まれていない他の Cookie に依存するプラグインをインストールした場合は、そのリストを拡張する必要があります。
- 、 WordPress、、、 など、 の出力に影響するHTTPヘッダーのみを転送しますHostCloudFront-Forwarded-ProtoCloudFront-is-Desktop-ViewerCloudFront-is-Mobile-ViewerCloudFront-is-Tablet-Viewer。
  - 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 経由でアクセスされるかどうかに応じて、さまざまなバージョンのページをキャッシュできますHTTPS。
  - CloudFront-is-Desktop-Viewer、 CloudFront-is-Tablet-Viewer を使用するとCloudFront-is-Mobile-Viewer、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。
- すべてのクエリ文字列を値に基づいてキャッシュに転送します。これら WordPress に依存するため、キャッシュされたオブジェクトを無効にするためにも使用できます。

カスタムドメイン名 (ではない*.cloudfront.net) でウェブサイトを提供する場合は、ディストリビューション設定のURIs代替ドメイン名に適切な を入力します。この場合、カスタムドメイン名のSSL証明書も必要です。AWS Certificate Manager を介して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することができます。

次に、動的コンテンツ用にさらに 2 つのキャッシュ動作を作成します。1 つはログインページ用 (パスパターン: wp-login.php) 、もう 1 つは管理者ダッシュボード用 (パスパターン:) ですwp-admin/*。これらの 2 つの動作は、次のようにまったく同じ設定になります。

- ビューワープロトコルポリシーのみを適用しますHTTPS。
- すべてのHTTPメソッドを許可します。
- すべてのHTTPヘッダーに基づいてキャッシュします。
- すべての Cookie を転送します。
- 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します。

この設定の背後にある理由は、ウェブサイトのこのセクションが高度にパーソナライズされており、通常、少数のユーザーしかいないため、キャッシュ効率が主な関心事ではないためです。すべての Cookie とヘッダーをオリジンに渡すことで、インストールされたプラグインとの互換性を最大限に高めるために、設定をシンプルに保つことが焦点です。

デフォルトでは、 はすべてウェブサーバーにローカルに WordPress 保存されます。ウェブサーバーは、単一サーバーデプロイ で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。 ???ストレージとデータ転送コストを削減するだけでなく、静的アセットを Amazon S3 に移行すると、スケーラビリティ、データの可用性、セキュリティ、パフォーマンスも向上します。静的コンテンツを Amazon S3 に移動しやすくするプラグインがいくつかあります。その 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 でも説明されています。

付録 B: 静的コンテンツ設定

デフォルトでは、 はすべてウェブサーバーにローカルに WordPress 保存されます。ウェブサーバーは、単一サーバーデプロイ 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 の場合はファイルストレージ (Amazon EFS) です。 ???ストレージとデータ転送コストを削減するだけでなく、静的アセットを Amazon S3 に移行すると、スケーラビリティ、データの可用性、セキュリティ、パフォーマンスも向上します。

この例では、W3 Total Cache (W3TC) プラグインを使用して Amazon S3 に静的アセットを保存します。ただし、同様の機能を持つ他のプラグインもあります。代替手段を使用する場合は、それに応じて次のステップを調整できます。ステップは、この例に関連する機能または設定のみを参照します。すべての設定の詳細な説明は、本書の範囲外です。詳細については、wordpress.org の W3 Total Cache プラグインページを参照してください。

ユーザーの作成

Amazon S3 に静的アセットを保存するには、 WordPress プラグインのユーザーを作成する必要があります。ステップについては、AWSアカウント でのユーザーの作成を参照してください。

注： ロールはAWSリソースへのアクセスを管理するより良い方法を提供しますが、書き込み時には W3 Total Cache プラグインはロール をサポートしていません。

ユーザーセキュリティ認証情報を書き留めて安全な方法で保存します。後でこれらの認証情報が必要です。

Amazon S3 バケット作成

1. まず、選択したAWSリージョンに Amazon S3 バケットを作成します。ステップについては、「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. 以前に作成したユーザーに指定された S3 バケットへのアクセスを提供するポリシーを作成し、そのポリシーをユーザーにアタッチします。次のポリシーを作成する手順については、「ポリシーの管理」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. WordPress 管理パネルから W3TC プラグインをインストールしてアクティブ化します。
4. プラグインの設定の全般設定セクションを参照し、ブラウザキャッシュと の両方CDN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、オリジンプッシュ: Amazon CloudFront (このオプションでは Amazon S3 をオリジンとして指定します) を選択します。
6. プラグインの設定のブラウザキャッシュセクションを参照し、有効期限が切れる 、キャッシュコントロール 、エンティティタグ (ETag) ヘッダーを有効にします。
7. また、設定変更後のオブジェクトのキャッシュの防止オプションを有効にして、設定が変更されるたびに新しいクエリ文字列が生成され、オブジェクトに追加されるようにします。
8. プラグインの設定CDNのセクションを参照して、以前に作成したユーザーのセキュリティ認証情報と S3 バケットの名前を入力します。
9. 経由でウェブサイトを提供している場合は CloudFront URL、関連するボックスにディストリビューションドメイン名を入力します。それ以外の場合は、カスタムドメイン名 (複数可) CNAMEs に 1 つ以上を入力します。
10. 最後に、メディアライブラリをエクスポートし、wp-includes、テーマファイル、カスタムファイルを W3TC プラグインを使用して Amazon S3 にアップロードします。これらのアップロード関数は、CDN設定ページの全般セクションで使用できます。

静的オリジンの作成

静的ファイルが Amazon S3 に保存されました。 CloudFront コンソール CloudFront の設定に戻り、静的コンテンツのオリジンとして Amazon S3 を設定します。これを行うには、その目的のために作成した S3 バケットを指す 2 番目のオリジンを追加します。次に、動的コンテンツのデフォルトのオリジンではなく S3 オリジンを使用する必要がある 2 つのフォルダ (`wp-content` と `wp-includes`) ごとに 1 つずつ、さらに 2 つのキャッシュ動作を作成します。両方を同じ方法で設定します。

- HTTP GET リクエストのみを提供します。
- Amazon S3 は Cookie やHTTPヘッダーに基づいて出力を変更しないため、 経由でオリジンに転送しないことでキャッシュ効率を向上させることができます CloudFront。
- これらの動作は静的コンテンツ (パラメータを受け入れない) のみを提供するという事実にもかかわらず、クエリ文字列を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、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、すぐに無効化できるようにするためです。詳細については、「Amazon CloudFront デベロッパーガイド」を参照してください。

> **Note**
>
> 静的オリジン動作を CloudFront ディストリビューションに追加したら、 の動作`wp-admin/*`と静的コンテンツの動作よりも優先順位`wp-login.php`が高いことを確認する順序を確認します。それ以外の場合、管理者パネルにアクセスすると異常な動作が表示されることがあります。

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、本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ており、(b) 通知なしに変更される可能性がある現在のAWS製品の提供と慣行を表し、(c) AWS およびその関連会社、サプライヤー、ライセンサーからのコミットメントや保証を一切作成しません。AWS 製品またはサービスは、明示的か黙示的かにかかわらず、保証、表明、またはいかなる種類の条件も伴わず、現状有姿で提供されます。顧客AWSに対する の責任と責任はAWS契約によって管理され、本書は AWSとその顧客との間の契約の一部でも、変更もありません。

AWS 用語集

最新の AWS 用語については、 AWS の用語集 リファレンスのAWS 用語集を参照してください。

翻訳は機械翻訳により提供されています。提供された翻訳内容と英語版の間で齟齬、不一致または矛盾がある場合、英語版が優先します。