AWS 白皮书

WordPress 上的最佳实践 AWS

WordPress 上的最佳实践 AWS: AWS 白皮书

Table of Contents

WordPress 上的最佳实践 AWS

发布日期：2021 年 10 月 19 日 (文档修订)

本白皮书为系统管理员提供了有关如何开始使用 Amaz WordPress on Web Services (AWS) 以及如何提高部署成本效益和最终用户体验的具体指导。它还概述了一种可满足常见可扩展性和高可用性要求的参考架构。

您的架构是否良好？

当您在云端构建系统时，AWS Well-Architected Framework 可帮助您了解所做决策的利弊。利用此框架的六个支柱，您可以了解到设计和运行可靠、安全、高效、经济有效且可持续的系统的架构最佳实践。您可以使用 AWS 管理控制台 免费提供的 AWS Well-Architected Tool，回答与每个支柱相关的一组问题，即可根据这些最佳实践检查自己的工作负载。

有关云架构的更多专家指导和最佳实践（参考架构部署、图表和白皮书），请参阅 AWS 架构中心。

引言

WordPress 是一种基于 PHP 和 MySQL 的开源博客工具和内容管理系统 (CMS)，可用于为从个人博客到高流量网站的任何内容提供支持。

第一版 WordPress 发布于 2003 年，在当初设计时，还没有弹性和可扩展性的现代云基础设施的概念。随着 WordPress 社区工作的推进和各种 WordPress 模块的发布，该 CMS 解决方案的功能不断扩展。现在，您可以构建一个利用 AWS 云的众多优势的 WordPress 架构。

简单部署

对于低流量博客或没有严格高可用性要求的网站，简单部署一台服务器可能是合适的。这种部署并不是最具弹性或可扩展性的架构，但它是启动和运行网站的最快、最经济的方式。

主题

注意事项

本讨论从单个 Web 服务器部署开始。在某些情况下，你可能会长出来，例如：

- 部署 WordPress 网站的虚拟机是单点故障。此实例出现问题会导致您的网站中断服务。
- 只能通过 “垂直扩展”（即增加运行 WordPress 网站的虚拟机的大小）来扩展资源以提高性能。

可用方法

AWS有许多不同的配置虚拟机的选项。您可以通过以下三种主要方式托管自己的 WordPress网站 AWS：

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

Amazon Lightsail 是一项服务，可让您快速启动虚拟专用服务器（Lightsail 实例）来托管网站。WordPress如果您不需要高度可配置的实例类型或访问高级联网功能，Lightsail 是开始使用的最简单的方法。

Amazon EC2 是一项网络服务，可提供可调整的计算容量，因此您可以在几分钟内启动虚拟服务器。与Lightsail相比，亚马逊EC2提供了更多的配置和管理选项，而Lightsail在更高级的架构中是理想的。您拥有EC2实例的管理权限，并且可以安装您选择的任何软件包，包括 WordPress。

AWS Marketplace是一个在线商店，您可以从中查找、购买和快速部署在上运行的软件AWS。您可以使用 1-Click 部署，在短短几分钟内通过自己的AWS账户将预配置EC2的 WordPress 映像直接发布到 Amazon。有许多 Marketplace 供应商提供 ready-to-run WordPress 实例。

本白皮书介绍了 Lightsail 选项作为单 WordPress 服务器网站的推荐实现方案。

Amazon Lightsail

Lightsail AWS 为开发人员、小型企业、学生以及需要简单虚拟私有服务器 (VPS) 解决方案的其他用户提供了一种最简单的方法以开始使用。

该服务将基础设施管理中许多更复杂的元素从用户身上抽象出来。因此，如果您的基础架构经验较少，或者需要专注于运营网站，而简化的产品足以满足您的需求，那么这是一个理想的起点。

借助 Amazon Lightsail，您可以选择 Windows 或 Linux/Unix 操作系统和流行的网络应用程序，包括这些应用程序 WordPress，只需单击一下预配置的模板即可部署它们。

随着需求的增长，您可以顺利地超越最初的界限，连接到其他AWS数据库、对象存储、缓存和内容分发服务。

选择亚马逊 Lightsail 定价计划

Lightsail 计划定义了你用来托管网站的 Lightsail 资源的每月费用。 WordPress 有许多计划可以涵盖各种用例，包括不同级别的CPU资源、内存、固态硬盘 (SSD) 存储和数据传输。如果您的网站很复杂，则可能需要一个具有更多资源的大型实例。您可以通过使用 Web 控制台或按照 Amazon Lightsail CLI 文档中所述将服务器迁移到更大的计划来实现这一目标。

正在安装 WordPress

Lightsail 为常用应用程序提供了模板，例如。 WordPress此模板是运行自己的 WordPress 网站的绝佳起点，因为它已预先安装了您需要的大部分软件。您可以使用浏览器内终端或自己的SSH客户端，或者通过 WordPress管理 Web 界面安装其他软件或自定义软件配置。

Amazon Lightsail 与 GoDaddy Pro Sites 产品建立了合作关系，可帮助 WordPress 客户免费轻松管理其实例。Lightsail WordPress 虚拟服务器经过预配置和优化，可实现快速性能和安全性，让您的 WordPress 网站立即轻松启动和运行。运行多个 WordPress 实例的客户发现更新、维护和管理所有站点既困难又耗时。通过这种集成，您只需点击几下即可在几分钟内轻松管理多个 WordPress 实例。

有关在安装 Lightsail 后对其 WordPress 进行管理的更多信息，请参阅 WordPress 从您的亚马逊 Lightsail 实例开始使用。完成 WordPress 网站自定义后，我们建议您拍摄实例的快照。

快照是创建 Lightsail 实例的备份映像的一种方式。它是系统磁盘的副本，还存储原始计算机配置（即内存CPU、磁盘大小和数据传输速率）。部署或升级失败后，可以使用快照恢复到已知的正确配置。

此快照允许您在需要时恢复服务器，也可以启动具有相同自定义项的新实例。

从故障中恢复

单个 Web 服务器是单点故障，因此您必须确保您的网站数据已备份。前面描述的快照机制也可以用于此目的。要从故障中恢复，您可以从最新的快照中恢复新实例。为了减少恢复过程中可能丢失的数据量，您的快照必须尽可能新。

为了最大限度地减少数据丢失的可能性，请确保定期拍摄快照。你可以安排你的 Lightsail Linux/Unix 实例的自动快照。有关步骤，请参阅在 Amazon Lightsail 中启用或禁用实例或磁盘的自动快照。

AWS建议您使用静态 IP：专用于您的 Lightsail 账户的固定公共 IP 地址。如果您需要用另一个实例替换您的实例，可以将静态 IP 重新分配给新实例。这样，您就不必在每次要替换实例时重新配置任何外部系统 (如DNS记录) 以指向新的 IP 地址。

提高性能和成本效益

您最终可能会超出单服务器部署容量。在这种情况下，您可能需要考虑提高网站性能的选项。在迁移到可扩展的多服务器部署（本文稍后将讨论）之前，您可以采取许多改进性能和提高成本效益的方法。这些是您无论如何都应该遵循的最佳实践，即使迁移到多服务器架构也是如此。

以下各节介绍了许多可以改善 WordPress 网站的性能和可扩展性的选项。有些可以应用于单服务器部署，有些则利用了多服务器的可扩展性。其中许多修改都需要使用一个或多个 WordPress 插件。尽管选项很多，但 W3 Total Cache 是一种受欢迎的选择，它将许多修改整合到一个插件中。

主题

- 加快内容交付
- 数据库缓存
- 字节码缓存

加快内容交付

任何 WordPress 网站都需要提供静态和动态混合内容。静态内容包括镜像、JavaScript 文件或样式表。动态内容包括使用 WordPress PHP 代码在服务器端生成的任何内容，例如，从数据库生成的或针对每个查看者提供个性化内容的网站元素。

最终用户体验的一个重要方面是，在向世界各地的用户交付以前的内容时可能出现网络延迟。加快交付以前的内容可改善最终用户的体验，尤其是分布在全球各地的用户的体验。为此，可以借助 Amazon CloudFront 等内容分发网络 (CDN)。

Amazon CloudFront 是一项 Web 服务，它提供通过全球多个边缘站点分发内容的方式，具有低延迟、数据传输快、简单且经济高效的优点。查看者的请求将自动路由到合适的 CloudFront 边缘站点，缩短了延迟时间。如果内容可以缓存（几秒钟、几分钟甚至几天），并且已经存储在特定边缘站点，则 CloudFront 可以很快交付它。如果内容不能缓存，或者已经过期，或者目前不在该边缘站点上，CloudFront 将在 CloudFront 配置中从一个或多个真实来源（在本例中为 Lightsail 实例）检索内容。这种检索通过优化的网络连接完成，因此交付网站上内容的速度加快。除了改善最终用户体验，所讨论的模型还减少了源服务器上的负载，并有可能显著节省成本。

静态内容卸载

这包括 CSS、JavaScript 和镜像文件，不管它们是属于 WordPress 主题还是属于内容管理员上传的媒体文件。所有这些文件都可以使用 W3 Total Cache 之类的插件存储在 Amazon Simple Storage

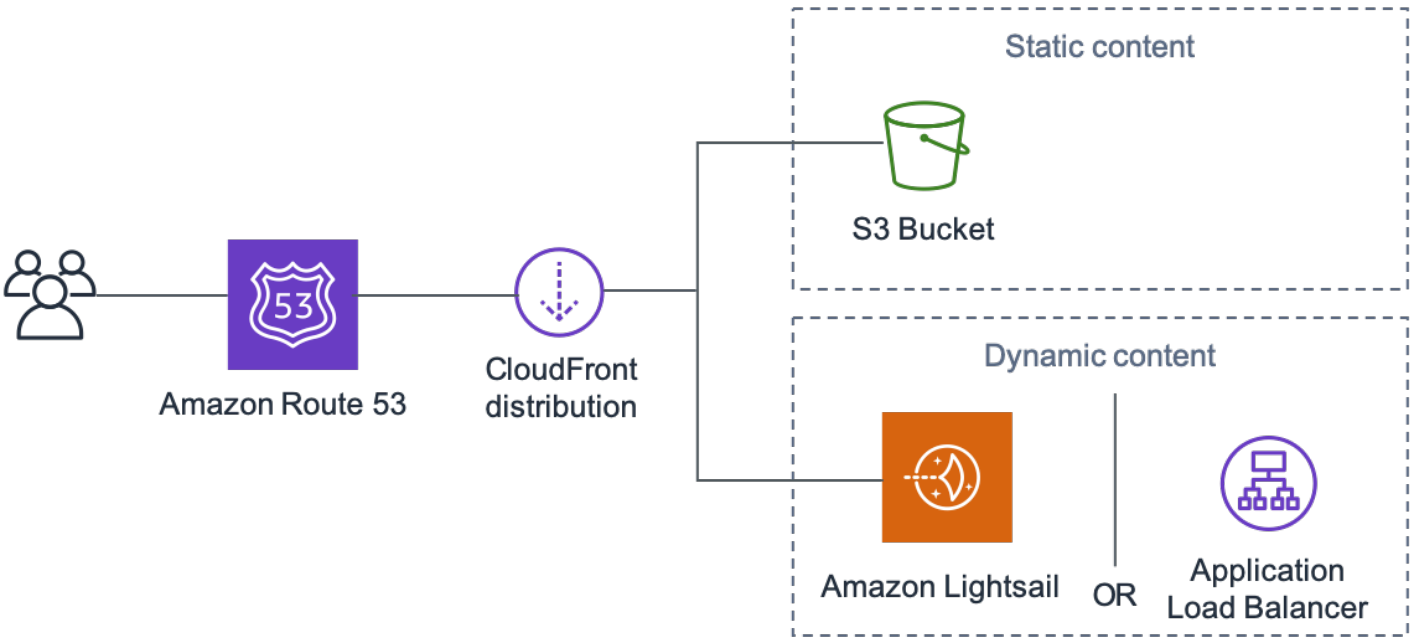
Service (Amazon S3) 中，并且可以采用可扩展且高度可用的方式提供给用户。Amazon S3 以低成本提供高度可扩展、可靠且低延迟的数据存储基础设施，可通过 REST API 访问该设施。Amazon S3 以冗余方式存储您的对象，不仅存储在多台设备上，还会跨一个 AWS 区域的多个设施存储，这带来了极高的持久性。

此举的积极效果是，从 Lightsail 实例上卸载了这一工作负载，让它专注于动态内容的生成。这减轻了服务器负载，是迈向创建无状态架构（实现自动扩展的先决条件）的重要一步。

随后，您可以将 Amazon S3 配置为 CloudFront 的源，以改进向全球用户交付这些静态资产的过程。尽管 WordPress 未与 Amazon S3 和 CloudFront 整合到开箱即用的程度，但各种插件增加了对这些服务（例如 W3 Total Cache）的支持。

动态内容

动态内容包括服务器端 WordPress PHP 脚本的输出。将 WordPress 网站配置为源后，也可以通过 CloudFront 提供动态内容。因为动态内容包含个性化内容，所以您需要将 CloudFront 配置为将某些 HTTP Cookie 和 HTTP 标头作为请求的一部分转发到自定义源服务器。CloudFront 使用转发的 Cookie 值作为标识其缓存中唯一对象的密钥的一部分。为确保最大限度提高缓存效率，应将 CloudFront 配置为仅转发那些真正改变内容的 HTTP Cookie 和 HTTP 标头（不是仅在客户端使用的或被第三方应用程序使用的 Cookie，例如用于网络分析的 Cookie）。



通过 Amazon CloudFront 交付整个网站

前图包括两个源：一个静态内容源，一个动态内容源。有关实现的详细信息，请参阅附录 A：CloudFront 配置和附录 B：插件的安装和配置。

CloudFront 使用标准缓存控制标头来确定是否应缓存特定 HTTP 响应以及缓存多长时间。Web 浏览器也使用相同的缓存控制标头来决定何时在本地缓存内容以及缓存多长时间，以获得更好的最终用户体验（例如，在下载一个 `.css` 文件后，每次回访者查看页面时都不会重新下载该文件）。您可以在 Web 服务器级别配置缓存控制标头（例如，通过 `.htaccess` 文件或修改 `httpd.conf` 文件），或安装 WordPress 插件（例如 W3 Total Cache）来指示如何为静态和动态内容设置这些标头。

数据库缓存

数据库缓存可以显著减少延迟和提高读取密集型应用程序工作负载（如WordPress）的吞吐量。通过将经常访问的数据片段存储在内存中以实现低延迟访问（例如，I/O 密集型数据库查询的结果），可以提高应用程序的性能。当很大一部分查询由高速缓存处理时，需要访问数据库的查询数量就会减少，从而降低与运行数据库相关的成本。

尽管 WordPress 开箱即用的缓存功能有限，但各种插件都支持与 Memcached（一种广泛采用的内存对象缓存系统）集成。W3 Total Cache 插件就是一个很好的例子。

在最简单的情况下，您可以在 Web 服务器上安装 Memcached，并将结果捕获为新快照。在本例中，您负责与运行缓存相关的管理任务。

另一种选择是利用 Amazon ElastiCache 等托管服务来避免这种运营负担。ElastiCache 让您在云环境中轻松部署、操作和扩展分布式内存中的缓存。您可以在 Amazon ElastiCache 文档中找到有关如何连接到 ElastiCache 集群节点的信息。

如果您正在使用 Lightsail 并希望私下访问您的 AWS 账户中的 ElastiCache 集群，可以使用 VPC 对等连接。有关启用 VPC 对等连接的说明，请参阅设置 Amazon VPC 对等连接以使用 Amazon Lightsail 之外的 AWS 资源。

字节码缓存

每次运行一个 PHP 脚本时，都会解析和编译该脚本。通过使用 PHP 字节码缓存，PHP 编译的输出存储在 RAM 中，这样就无需反复编译相同的脚本。这减少了与执行 PHP 脚本相关的开销，改进了性能并降低了 CPU 要求。

字节码缓存可以安装在任何托管 WordPress 的 Lightsail 实例上，因此大大减少了其负载。对于 PHP 5.5 及更高版本，AWS 建议使用 OpCache，这是该 PHP 版本随附的一个扩展功能。

请注意，Bitnami WordPress Lightsail 模板中默认启用 OpCache，无需进一步操作。

弹性部署

在许多情况下，单服务器部署可能不足以满足您的网站需求。在这些情况下，您需要一个多服务器、可扩展的架构。

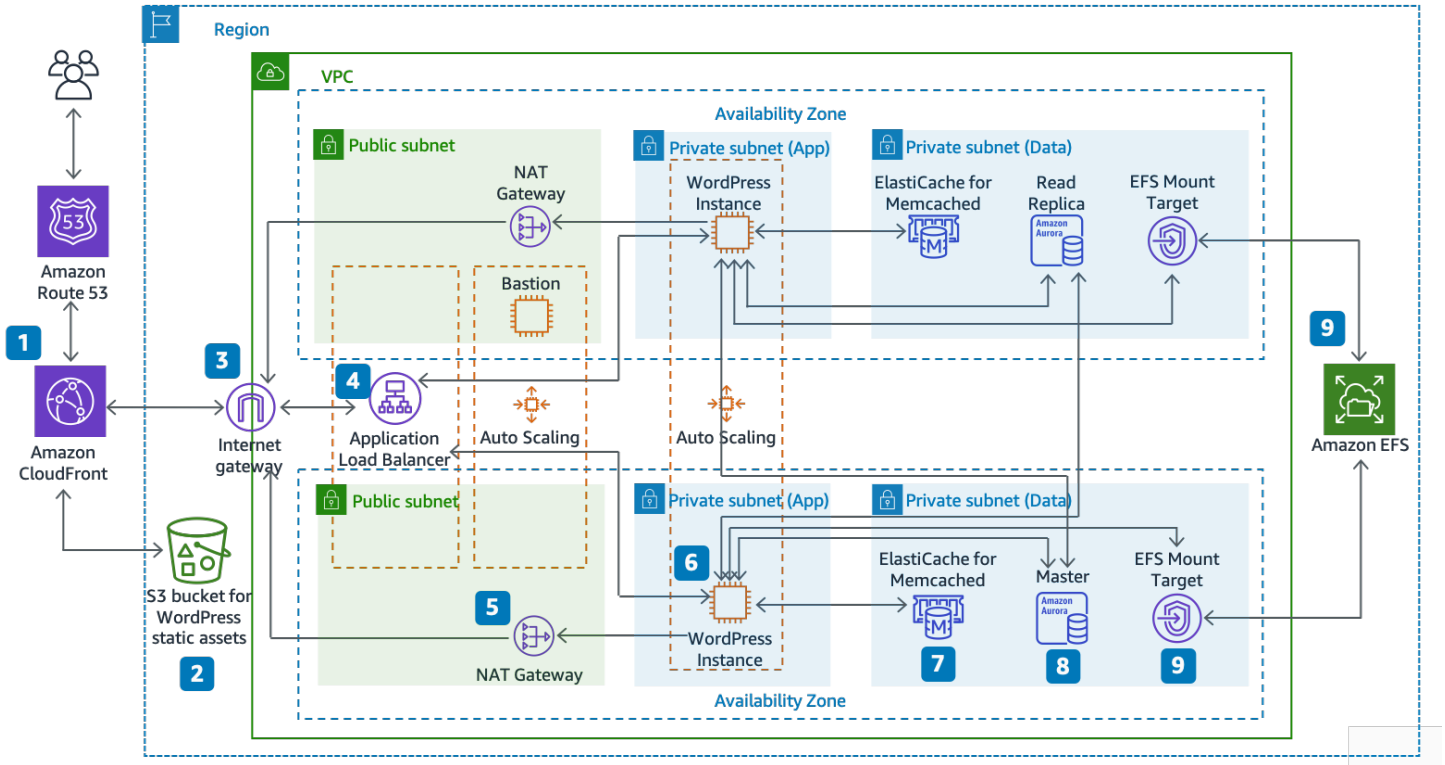
主题

- 参考架构
- 扩展 Web 层
- 无状态数

参考架构

WordPress 上的 Hosting on AWS 参考架构 GitHub概述了部署 WordPress 的最佳实践，AWS并包括一组可以让你快速启动和运行的 AWS CloudFormation 模板。以下架构基于该参考架构。本节的其余部分将回顾架构选择背后的原因。

2021 GitHub 年 7 AMI 月，其基础从亚马逊 Linux1 更改为亚马逊 Linux2。但是，S3 的部署模板尚未更改。 GitHub 如果在 S3 上使用模板部署参考架构时遇到问题，建议使用中的模板。



用于托管 WordPress 的参考架构 AWS

架构组件

参考架构说明了 WordPress 网站上网站的完整最佳实践部署AWS。

- 它从 Amazon CloudFront (1) 中的边缘缓存开始，将内容缓存到靠近最终用户的地方，以便更快地交付。
- CloudFront 从 S3 存储桶 (2) 中提取静态内容，从 Web 实例前面的 Application Loa d Balancer (4) 中提取动态内容。
- 网络实例在由 Amazon EC2 实例组成的 A uto Scaling 组中运行 (6)。
- 集 ElastiCache 群 (7) 会缓存经常查询的数据以加快响应速度。

  A mazon Aurora 我的SQL实例 (8) 托管 WordPress数据库。
- 这些 WordPress EC2实例通过每个可用区中的EFS挂载目标 (9) 访问 Amazon EFS 文件系统上的共享 WordPress 数据。
- In ternet Gateway (3) 可实现您VPC和互联网中的资源之间的通信。
- NAT每个可用区中的@@ 网关 (5) 允许私有子网（应用程序和数据）中的EC2实例访问互联网。

Amazon 内VPC存在两种类型的子网：公有子网（公有子网）和私有子网（应用程序子网和数据子网）。部署到公有子网中的资源将获得一个公有 IP 地址，并且将在互联网上公开可见。此处部署了 Application Load Balancer (4) 和一台用于管理的 Bastion 主机。部署到私有子网中的资源只能获得私有 IP 地址，因此在互联网上不可见，从而提高了这些资源的安全性。WordPress Web 服务器实例 (6)、ElastiCache集群实例 (7)、Aurora My SQL 数据库实例 (8) 和EFS挂载目标 (9) 都部署在私有子网中。

本节的其余部分将更详细地介绍其中的每一个注意事项。

扩展 Web 层

要将单服务器架构演变为多服务器、可扩展的架构，必须使用五个关键组件：

- 亚马逊EC2实例
- Amazon 系统映像 (AMIs)
- 负载均衡器
- 自动扩缩
- 运行状况检查

AWS提供多种EC2实例类型，让您能够根据性能和成本选择最佳服务器配置。一般而言，计算优化（例如 C4）实例类型可能是 Web 服务器的不错选择。 WordPress 您可以跨AWS区域内的多个可用区部署实例，以提高整体架构的可靠性。

由于您可以完全控制自己的EC2实例，因此您可以使用 root 权限登录，以安装和配置运行 WordPress 网站所需的所有软件组件。完成后，您可以将该配置另存为AMI，用它来启动具有您所做的所有自定义项的新实例。

要将最终用户请求分发到多个 Web 服务器节点，您需要一个负载平衡解决方案。AWS通过 Elastic Load Bal ancing 提供此功能，这是一项高度可用的服务，可将流量分配到多个EC2实例。由于您的网站通过HTTP或向用户提供内容HTTPS，因此我们建议您使用 Application Load Balancer，这是一种具有内容路由功能的应用程序层负载均衡器，并且能够在需要时在不同的域上运行多个 WordPress 网站。

Elastic Load Balancing 支持在一个AWS区域内的多个可用区之间分配请求。您还可以配置运行状况检查，以便 Application Load Balancer 自动停止向出现故障（例如，由于硬件问题或软件崩溃）的单个实例发送流量。AWS建议使用 WordPress 管理员登录页面 (`/wp-login.php`) 进行运行状况检查，因为该页面既确认 Web 服务器正在运行，也确认 Web 服务器已配置为正确提供PHP文件。

您可以选择构建一个自定义运行状况检查页面，用于检查其他依赖资源，例如数据库和缓存资源。有关更多信息，请参阅《App lication Load Balancer 指南》中的目标群体的健康检查。

弹性是AWS云的关键特征。您可以在需要时启动更多的计算容量（例如 Web 服务器），而在不需要时可以减少运行的容量。Amazon A EC2 uto Scaling 是一项 AWS 服务，可帮助您自动进行此配置，从而根据您定义的条件向上或向下扩展您的 Amazon EC2 容量，无需手动干预。您可以配置 Amazon A EC2 uto Scaling，使您使用的EC2实例数量在需求高峰期间无缝增加以保持性能，并在流量减少时自动减少，从而最大限度地降低成本。

Elastic Load Balancing 还支持在负载平衡轮EC2换中动态添加和移除亚马逊主机。Elastic Load Balancing 本身还可以动态增加和减少负载平衡容量，以适应流量需求，无需人工干预。

无状态数

要在自动扩展配置中利用多个 Web 服务器，您的 Web 层必须处于无状态状态。无状态应用程序是指不需要知道以前的交互并且不存储会话信息的应用程序。如果是 WordPress，这意味着无论哪个 Web 服务器处理了他们的请求，所有最终用户都会收到相同的响应。无状态应用程序可以横向扩展，因为任何请求都可以由任何可用的计算资源（即 Web 服务器实例）提供服务。当不再需要该容量时，可以安全地终止任何单个资源（在耗尽正在运行的任务之后）。这些资源不需要意识到同行的存在——所需要的只是一种将工作量分配给他们的方法。

在用户会话数据存储方面， WordPress 核心是完全无状态的，因为它依赖于存储在客户端 Web 浏览器中的 Cookie。除非您安装了任何依赖本机会话的自定义代码（例如 WordPress 插件），否则不会考虑PHP会话存储。

但是， WordPress 最初是为在单台服务器上运行而设计的。因此，它将一些数据存储在服务器的本地文件系统上。在多服务器配置 WordPress 中运行时，这会产生问题，因为各个 Web 服务器之间存在不一致性。例如，如果用户上传了一张新图像，则该图像仅存储在其中一台服务器上。

这说明了为什么我们需要改进默认的 WordPress运行配置才能将重要数据移动到共享存储。最佳实践架构将数据库作为 Web 服务器之外的独立层，并利用共享存储空间来存储用户上传的内容、主题和插件。

共享存储（亚马逊 S3 和亚马逊EFS）

默认情况下，将用户上传的内容 WordPress 存储在本地文件系统上，因此不是无状态的。因此，我们需要将 WordPress 安装和所有用户自定义设置（例如配置、插件、主题和用户生成的上传）转移到共享数据平台中，以帮助减少 Web 服务器的负载并使 Web 层处于无状态状态。

Amazon Elastic File System（亚马逊EFS）提供可扩展的网络文件系统，用于EC2实例。Amazon EFS 文件系统分布在数量不受限制的存储服务器上，使文件系统能够弹性增长，并允许从实例进行大规模并行访问EC2。Amazon 的分布式设计EFS避免了传统文件服务器固有的瓶颈和限制。

通过将整个 WordPress 安装目录移动到EFS文件系统上，并在每个实例启动时将其安装到每个EC2实例中，您的 WordPress 站点及其所有数据将自动存储在不依赖任何一个EC2实例的分布式文件系统上，从而使您的 Web 层完全处于无状态状态。这种架构的好处是，您无需在每次启动新实例时都安装插件和主题，并且可以显著加快 WordPress 实例的安装和恢复。如本文档的 “部署注意事项” 部分所述WordPress，在中部署对插件和主题的更改也更加容易。

为确保您的网站在EFS文件系统上运行时获得最佳性能，请查看 Amaz OPcache on EFS 和AWS参考架构的推荐配置设置 WordPress。

您还可以选择将所有静态资产（例如图像和 JavaScript 文件）卸载到前面有 CloudFront 缓存的 S3 存储桶。CSS如本白皮书的 “静态内容” 部分所述，在多服务器架构中执行此操作的机制与单服务器架构完全相同。其好处与单服务器架构相同，您可以将与提供静态资产相关的工作转移到 Amazon S3，从而使您的 Web 服务器能够仅专注于生成动态内容 CloudFront，并在每台 Web 服务器上处理更多用户请求。

数据层（亚马逊 Aurora 和亚马逊 ElastiCache）

通过将 WordPress 安装存储在分布式、可扩展、共享的网络文件系统上，并由 Amazon S3 提供静态资产，您可以将注意力集中在剩下的有状态组件：数据库。与存储层一样，数据库不应依赖于任何一台服务器，因此不能将其托管在其中一个 Web 服务器上。而应将 WordPress 数据库托管在亚马逊 Aurora 上。

Amazon Aurora 是一种与 My SQL 和 Postgre SQL 兼容的关系数据库，结合了高端商用数据库的性能和可用性，同时还具有开源数据库的简单性和成本效益。Aurora My 将数据库引擎与由专门构建的分布式存储系统紧密集成，来SQL提高我的SQL性能和可用性。SSD它具有容错能力和自我修复功能，可在三个可用区复制六个数据副本，可用性超过 99.99%，并且可以持续备份您在 Amazon S3 中的数据。Amazon Aurora 旨在自动检测数据库崩溃并重新启动，无需进行崩溃恢复或重新构建数据库缓存。

Amazon Aurora 提供了多种实例类型，以适应不同的应用程序配置，包括内存优化型实例和可突发实例。要提高数据库的性能，您可以选择大型实例类型来提供更多CPU和内存资源。

Amazon Aurora 会自动处理主实例和 Aurora 副本之间的故障转移，以便应用程序尽快恢复数据库操作而无需手动管理干预。失效转移到失效转移通常可在不到 30 秒的时间内完成。

创建至少一个 Aurora 副本后，使用集群终端节点连接到您的主实例，以便在主实例出现故障时您的应用程序能够自动进行故障转移。您可以跨三个可用区域创建多达 15 个低延迟只读副本。

随着数据库的扩展，数据库缓存也需要扩展。如前面的 “数据库缓存” 部分所述， ElastiCache 它具有跨 ElastiCache 集群中的多个节点以及跨区域的多个可用区扩展缓存的功能，以提高可用性。在扩展 ElastiCache 集群时，请确保将缓存插件配置为使用配置终端节点进行连接，以便 WordPress 可以在添加新集群节点时使用它们，并在移除旧集群节点后停止使用它们。您还必须将 Web 服务器设置为使用ElastiCache群集客户端，PHP并更新您的服务器AMI以存储此更改。

总结

AWS 提供了许多用于运行 WordPress 的架构选项。最简单的选项就是安装一台服务器，这适用于低流量网站。对于更高级的网站，网站管理员可以添加其他几个选项，每个选项都代表了可用性和可扩展性方面的持续改进。管理员可以选择最符合其要求和预算的功能。

贡献者

本文档的贡献者包括：

- Paul Lewis，Amazon Web Services 解决方案构架师
- Ronan Guilfoyle，Amazon Web Services 解决方案构架师
- Andreas Chatzakis，Amazon Web Services 解决方案架构经理
- Jibril Touzi，Amazon Web Services 技术客户经理
- Hakmin Kim，Amazon Web Services 迁移合作伙伴解决方案构架师

文档修订

如需获取有关本白皮书更新的通知，请订阅 RSS Feed 源。

变更	说明	日期
已更新白皮书	更新为修改参考架构和AWS WordPress 插件。	2021 年 10 月 19 日
已更新白皮书	更新以包括新的部署方法和 AWS WordPress 插件。	2019 年 10 月 30 日
已更新白皮书	更新以澄清亚马逊 Aurora 产品信息。	2018 年 2 月 1 日
已更新白皮书	已更新，包括自首次发布以来推出的AWS服务。	2017 年 12 月 1 日
初次发布	首次发布。	2014 年 12 月 1 日

附录 A： CloudFront 配置

为了在您的 WordPress 网站中 CloudFront 使用Amazon时获得最佳性能和效率，请务必针对所提供的不同类型的内容正确配置网站。

主题

- 起源和行为
- CloudFront 发行版创建

起源和行为

源站是指向其通过边缘站点分发的内容 CloudFront 发送请求的位置。根据你的实现情况，你可以有一个或两个来源。一个用于使用自定义来源的动态内容（单服务器部署选项中的 Lightsail 实例或弹性部署选项中的 Application Load Balancer）。您的静态内容可能还有第二个来源可以 CloudFront 定向。在前面的参考架构中，这是一个 S3 存储桶。当您使用 Amazon S3 作为分配的源时，需要使用存储桶策略使内容可公开访问。

行为允许您设置规则来控制 CloudFront 缓存内容的方式，进而确定缓存的有效性。行为允许您控制访问网站的协议和HTTP方法。它们还允许您控制是否将HTTP标头、cookie 或查询字符串传递到后端（如果是，则传递哪些字符串）。行为适用于特定的URL路径模式。

CloudFront 发行版创建

通过关注分配来创建 CloudFront Web 分配，自动创建的默认来源和行为将用于动态内容。创建另外四个行为，以进一步自定义处理静态和动态请求的方式。下表总结了这五种行为的配置属性。

表 1： CloudFront 行为的配置属性摘要

属性	静态	动态（管理员）	动态（前端）
路径（行为）	`wp-content/*` `wp-includes/*`	`wp-admin/*` `wp-login.php`	默认 (*)
协议	HTTP和 HTTPS	重定向至 HTTPS	HTTP和 HTTPS
HTTP方法	GET, HEAD	ALL	ALL

属性	静态	动态（管理员）	动态（前端）
HTTP标题	NONE	ALL	Host CloudFront-Forwarded-Proto CloudFront-Is-Mobile-Viewer CloudFront-Is-Tablet-Viewer CloudFront-Is-Desktop-Viewer
Cookie	NONE	ALL	评论_* wordpress_* wp-settings-*
查询字符串	YES（无效）	YES	YES

对于默认行为，AWS建议使用以下配置：

- 允许 Origin Protocol 策略与 Viewer 匹配，这样HTTPS，如果查看者 CloudFront 连接到 CloudFront 使用，也可以使用HTTPS它连接到您的源，从而实现 end-to-end加密。请注意，这需要您在负载均衡器上安装可信SSL证书。有关详细信息，请参阅HTTPS要求在 CloudFront 和您的自定义来源之间进行通信。
- 允许所有HTTP方法，因为网站的动态部分需要同时使用GET和POST请求（例如，支持POST评论提交表单）。
- 仅转发会改变 WordPress 输出的 Cookie；例如>`wordpress_*`、`wp-settings-*`和`comment_*`。如果您安装了任何依赖列表中未列出的其他 Cookie 的插件，则必须扩展该列表。
- 仅转发影响输出的HTTP标题 WordPress，例如、`Host`、`CloudFront-Forwarded-Proto`、`CloudFront-is-Desktop-Viewer`、`CloudFront-is-Mobile-Viewer`、和`CloudFront-is-Tablet-Viewer`：

  - `Host`允许将多个 WordPress 网站托管在同一个源上。
  - `CloudFront-Forwarded-Proto`允许根据是通过HTTP还是访问页面来缓存不同版本的页面HTTPS。
  - `CloudFront-is-Desktop-Viewer`，`CloudFront-is-Mobile-Viewer`，`CloudFront-is-Tablet-Viewer`允许您根据最终用户的设备类型自定义主题的输出。
- 根据查询字符串的值将所有查询字符串转发到缓存中，因为 WordPress 依赖这些值，它们也可以用来使缓存的对象失效。

如果您想使用自定义域名（即不是`*.cloudfront.net`）为网站提供服务，请在 “分发设置” 的 “备用域名” URIs 下输入相应的域名。在这种情况下，自定义域名还需要一个SSL证书。您可以通过 Certificate Manager 申请SSLAWS证书，并针对 CloudFront 分发进行配置。

现在，为动态内容再创建两个缓存行为：一个用于登录页面（路径模式:`wp-login.php`），另一个用于管理员仪表板（路径模式:`wp-admin/*`）。这两种行为的设置完全相同，如下所示：

- 强制执行 “HTTPS仅限查看者协议” 策略。
- 允许所有HTTP方法。
- 基于所有HTTP标题进行缓存。
- 转发所有 Cookie。
- 根据所有 Cookie 转发和缓存。

这种配置背后的原因是，网站的这一部分是高度个性化的，通常只有几个用户，因此缓存效率不是首要考虑的问题。重点是保持配置简单，通过将所有 cookie 和标头传递给源站来确保与任何已安装插件的最大兼容性。

默认情况下，将所有内容 WordPress 存储在本地的 Web 服务器上，即用于单服务器部署的块存储 (AmazonEBS) 和用于弹性部署的文件存储 (AmazonEFS)。除了降低存储和数据传输成本外，将静态资产转移到 Amazon S3 还提供可扩展性、数据可用性、安全性和性能。有几个插件可以轻松地将静态内容移至 Amazon S3；其中之一是 W3 Total Cache，附录 B：插件安装和配置中也对此进行了介绍。

附录 B：静态内容配置

默认情况下，将所有内容 WordPress 存储在本地网络服务器上，即用于单服务器部署的块存储 (AmazonEBS) 和用于弹性部署的文件存储 (AmazonEFS)。除了降低存储和数据传输成本外，将静态资产转移到 Amazon S3 还提供可扩展性、数据可用性、安全性和性能。

在此示例中，W3 总缓存 (W3TC) 插件用于在 Amazon S3 上存储静态资产。但是，还有其他具有类似功能的插件可用。如果您想使用替代方案，可以相应地调整以下步骤。这些步骤仅涉及与此示例相关的功能或设置。本文档讨论不包含所有设置的说明。有关更多信息，请参阅 wordp ress.org 上的 W3 Total Cache 插件页面。

创建用户

您需要为 WordPress 插件创建一个用户，以便在 Amazon S3 中存储静态资产。有关步骤，请参阅在您的AWS账户中创建用户。

注意：角色提供了一种更好的AWS资源访问管理方式，但在撰写本文时，W3 Total Cache 插件不支持角色。

记下用户安全凭证，并以安全的方式存储它们，以后您需要这些证书。

Amazon S3 存储桶创建

1. 首先，在您选择的AWS区域创建 Amazon S3 存储桶。有关步骤，请参阅创建存储桶。按照教程：在 Amazon S3 上配置静态网站，为存储桶启用静态网站托管。
2. 创建策略以向先前创建的用户提供对指定 S3 存储桶的访问权限，并将该策略附加到该用户。有关创建以下策略的步骤，请参阅管理策略。

```
{
"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. 从 WordPress 管理面板安装并激活 W3TC 插件。
4. 浏览到插件配置的 “常规设置” 部分，并确保浏览器缓存和CDN均已启用。
5. 从CDN配置的下拉列表中选择 O rigin Push：Amazon CloudFront（此选项以 Amazon S3 作为其来源）。
6. 浏览到插件配置的 “浏览器缓存” 部分，并启用过期、缓存控制和实体标签 (ETag) 标头。
7. 还要激活 “设置更改后防止缓存对象” 选项，这样每当更改任何设置时，都会生成一个新的查询字符串并将其附加到对象中。
8. 浏览到插件配置CDN部分，输入您之前创建的用户的安全凭证以及 S3 存储桶的名称。
9. 如果您通过提供网站服务 CloudFront URL，请在相关框中输入分发域名。否则，请输入一个或多个CNAMEs自定义域名。
10. 最后，使用 W3TC 插件导出媒体库并将 wp-includes、主题文件和自定义文件上传到 Amazon S3。这些上传功能可在CDN配置页面的 “常规” 部分中找到。

静态原点创建

现在，静态文件已存储在 Amazon S3 上，请返回 CloudFront 控制台中的 CloudFront 配置，将 Amazon S3 配置为静态内容的来源。为此，请添加指向您为此目的创建的 S3 存储桶的第二个源。然后再创建两个缓存行为，分别针对两个文件夹（`wp-content`和`wp-includes`），这两个文件夹（和）应使用 S3 源而不是动态内容的默认来源。以相同的方式配置两者：

- 仅HTTPGET处理请求。
- Amazon S3 不会根据 Cookie 或HTTP标头改变其输出，因此您可以通过不通过将它们转发到源站来提高缓存效率 CloudFront。

- 尽管这些行为仅提供静态内容（不接受任何参数），但您仍会将查询字符串转发到源。这样，您就可以使用查询字符串作为版本标识符，以便在部署新版本时立即使旧CSS文件失效。有关更多信息，请参阅《Amazon CloudFront 开发者指南》。

> **Note**
>
> 将静态起源行为添加到您的 CloudFront 分配后，请检查顺序以确保静态内容的`wp-admin/*`行为优先级高于静态内容的行为。`wp-login.php`否则，您可能会在访问管理面板时看到奇怪的行为。

附录 C：备份和恢复

与传统托管环境相比，在 AWS 中，故障恢复更快且更容易。例如，您可以在几分钟内启动替换实例来应对硬件故障，或者，在我们的许多托管式服务中利用自动故障转移来消除由于例行性维护导致的重启影响。

但是，为了成功恢复数据，仍需要确保备份了正确的数据。要重新建立 WordPress 网站的可用性，必须能够恢复以下组件：

- 操作系统 (OS) 以及服务安装和配置（Apache、MySQL 等）
- WordPress 应用程序代码和配置
- WordPress 主题和插件
- 上传（例如，帖子的媒体文件）
- 数据库内容（帖子、评论等）

AWS 提供了多种方法来备份和还原 Web 应用程序数据及资产。

本白皮书之前讨论了如何利用 Lightsail 快照来保护存储在实例本地存储中的所有数据。如果您的 WordPress 网站仅在 Lightsail 实例上运行，那么定期拍摄的 Lightsail 快照足够恢复您的整个 WordPress 网站。但是，如果您从某个快照还原，则仍会丢失自上次拍摄快照以来应用于网站的所有更改。

在多服务器部署中，需要使用不同的机制备份前面讨论的每个组件。每个组件可能对备份频率的要求不同，例如，操作系统和 WordPress 的安装和配置会比用户生成的内容的改动频率低很多，因此，备份频率低一点也不会在恢复时丢失数据。

要备份操作系统和服务的安装和配置以及 WordPress 应用程序代码和配置，您可以创建正确配置的 EC2 实例的 AMI。AMI 有两个用途：一是作为实例状态的备份，二是在启动新实例时作为模板。

要备份 WordPress 应用程序代码和配置，需要利用 AMI 和 Aurora 备份。

要备份网站上安装的 WordPress 主题和插件，应备份 Amazon S3 存储桶或存储它们的 Amazon EFS 文件系统。

- 对于存储在 S3 存储桶中的主题和插件，您可以启用跨区域复制，使上传到主存储桶的所有对象都自动复制到另一个 AWS 区域的备份存储桶。跨区域复制要求在源存储桶和目标存储桶上都启用版本控制，这提供了一个额外保护层，并允许您还原到存储桶中任何给定对象的某个之前版本。

- 对于存储在 EFS 文件系统中的主题和插件，您可以创建一个 AWS Data Pipeline，以便将数据从生产 EFS 文件系统复制到另一个 EFS 文件系统，如备份 Amazon EFS 文件系统文档页面中所述。您还可以使用任何熟悉的备份应用程序来备份 EFS 文件系统。
- 要备份用户上传的内容，则应遵循前述的备份 WordPress 主题和插件的步骤。
- 要备份数据库内容，需要使用 Aurora 备份。Aurora 自动备份您的集群卷，并在备份保留期内一直保留还原数据。Aurora 备份是连续且递增的，所以您可以快速还原到备份保留期内的任何时间点。在写入备份数据时，数据库服务的性能不会受影响或中断。您可以指定备份保留期为 1 到 35 天。您还可以创建手动数据库快照，它们会保留到您删除它们为止。手动数据库快照对于长期备份和存档很有用。

附录 D：部署新插件和主题

很少有网站会一直处于静态。大多数情况下，您需要定期添加公开可用的 WordPress 主题和插件，或升级到更新的 WordPress 版本。在其他情况下，您需要从头开始开发自己的自定义主题和插件。

每当您对 WordPress 安装进行结构性改动时，都会存在引入不可预见问题的风险。因此，在应用任何重大更改（例如安装新插件）前，至少应备份应用程序代码、配置和数据库。对于具有商业价值或其他价值的网站，请先在隔离的暂存环境中测试这些更改。借助 AWS，可以轻松复制生产环境的配置，并以安全的方式运行整个部署过程。完成测试后，只需删除测试环境并停止为这些资源付费即可。本白皮书稍后会讨论一些特定于 WordPress 的注意事项。

有些插件将配置信息写入 `wp_options` 数据库表（或引入数据库架构更改），有些则在 WordPress 安装目录中创建配置文件。因为我们已将数据库和存储迁移到共享平台，所以这些更改能立即供您的所有正在运行的实例使用，无需执行更多操作。

在 WordPress 中部署新主题时，可能需要额外执行一些操作。如果您仅使用 Amazon EFS 来存储所有 WordPress 安装文件，那么新主题将立即可用于所有正在运行的实例。但是，如果您将静态内容卸载到 Amazon S3 中，则必须将这些内容的副本置于正确的存储桶位置。像 W3 Total Cache 这样的插件为您提供了一种手动启动该任务的方法。您也可以在构建过程中自动执行此步骤。

由于主题资产可以缓存在 CloudFront 和浏览器中，所以，您需要一种方法来使旧版本在部署更改时失效。实现此目标的最佳方法是在对象中包含某种版本标识符。此标识符可以是带有日期时间戳的查询字符串，也可以是随机字符串。如果您使用 W3 Total Cache 插件，则可以更新附加到媒体文件的 URL 的媒体查询字符串。

版权声明

AWS 术语表

有关最新 AWS 术语，请参阅《AWS 词汇表 参考资料》中的AWS 词汇表。

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。