AWS 白皮书

带有 Amazon API Gateway 和 AWS Lambda 的 AWS 无服务器多层架构

带有 Amazon API Gateway 和 AWS Lambda 的 AWS 无服务器多层架构: AWS 白皮书

Table of Contents

本白皮书仅供历史参考。有些内容可能已过时，有些链接可能不可用。

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。

带有 Amazon API Gateway 和 AWS Lambda 的 AWS 无服务器多层架构

发布日期：2021 年 10 月 20 日 (文档修订)

摘要

本白皮书说明了如何使用 Amazon Web Services (AWS) 的创新来改变您设计多层架构和实现常用模式（例如微服务、移动后端和单页应用程序）的方式。架构师和开发人员可以使用 Amazon API Gateway 和其他服务来缩短创建和管理多层应用程序所需的开发和操作周期。 AWS Lambda

您的架构是否良好？

当您在云端构建系统时，AWS Well-Architected Framework 可助您了解所作决策的利弊。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。

在无服务器应用程序视角中，我们重点介绍在上架构无服务器应用程序的最佳实践。 AWS

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。

简介

几十年来，多层应用程序（三层、n 层等）一直是基础架构模式，并且仍然是面向用户的应用程序的流行模式。尽管用于描述多层架构的语言各不相同，但多层应用程序通常由以下组件组成：

- 演示层：用户直接与之交互的组件（例如，网页和移动应用程序 UIs）。
- 逻辑层：将用户操作转换为应用程序功能（例如，CRUD 数据库操作和数据处理）所需的代码。
- 数据层：存储与应用程序相关的数据的存储介质（例如数据库、对象存储、缓存和文件系统）。

多层架构模式提供了一个通用框架，以确保可以单独开发、管理和维护分离且可独立扩展的应用程序组件（通常由不同的团队开发、管理和维护）。

由于这种网络模式（一层必须进行网络调用才能与另一层交互）充当各层之间的边界，因此开发多层应用程序通常需要创建许多无差别的应用程序组件。其中一些组件包括：

- 用于定义层间通信的消息队列的代码
- 定义应用程序编程接口 (API) 和数据模型的代码
- 与安全相关的代码，可确保对应用程序的适当访问

所有这些示例都可以被视为 “样板” 组件，这些组件虽然在多层应用程序中是必要的，但它们在实现上从一个应用程序到另一个应用程序的差异不大。

AWS 提供了许多支持创建无服务器多层应用程序的服务，极大地简化了将此类应用程序部署到生产环境的过程，并消除了与传统服务器管理相关的开销。Amazon API Gatew ay 是一项用于创建和管理 APIs的服务 AWS Lambda，也是一项用于运行任意代码函数的服务，可以一起使用，以简化强大的多层应用程序的创建。

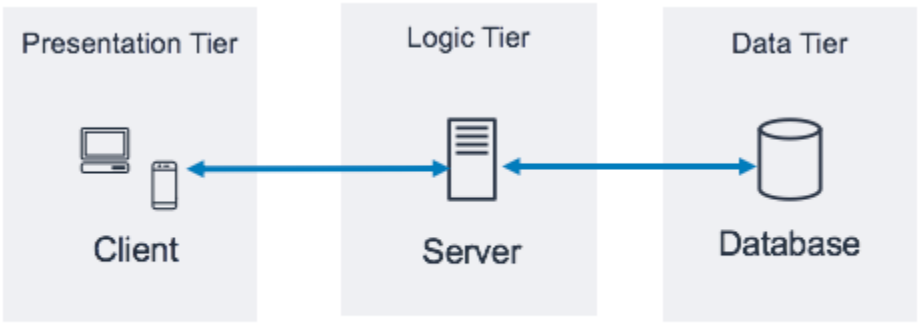
Amazon API Gateway 与的集成 AWS Lambda 使用户定义的代码函数可以直接通过 HTTPS 请求启动。无论请求量如何，API Gateway 和 Lambda 都会自动扩展以完全支持您的应用程序的需求（有关可扩展性信息，请参阅 API Gateway Amazon API Gateway 配额和重要说明）。通过组合这两项服务，您可以创建一个层，使您能够仅编写对应用程序至关重要的代码，而不必专注于实现多层架构的其他各种不同方面，例如设计高可用性、编写客户端 SDKs、服务器和操作系统 (OS) 管理、扩展以及实现客户端授权机制。

API Gateway 和 Lambda 支持创建无服务器逻辑层。根据您的应用程序要求，AWS 还提供创建无服务器演示层（例如，使用亚马逊 CloudFront和亚马逊简单存储服务）和数据层（例如 Amazon A urora、Amazon DynamoD B）的选项。

本白皮书重点介绍最受欢迎的多层架构示例，即三层 Web 应用程序。但是，您可以应用这种多层模式，远远超出典型的三层 Web 应用程序。

三层架构概述

三层架构是多层架构中最常用的实现，由单个表示层、逻辑层和数据层组成。下图显示了一个简单、通用的三层应用程序的示例。



三层应用程序的架构模式

有许多很棒的在线资源可供您详细了解一般的三层架构模式。本白皮书重点介绍使用 Amazon API Gateway 和 AWS Lambda该架构的具体实现模式。

无服务器逻辑层

三层架构的逻辑层代表了应用程序的大脑。这是使用 Amazon API Gateway 的地方，与 AWS Lambda 基于服务器的传统实施相比，其影响最大。这两项服务的功能使您能够构建高度可用、可扩展和安全的无服务器应用程序。在传统模式中，您的应用程序可能需要数千台服务器；但是，通过使用 Amazon API Gateway， AWS Lambda 您无需负责任何容量的服务器管理。此外，通过将这些托管服务一起使用，您将获得以下好处：

- AWS Lambda:
  - 无需选择、保护、修补或管理操作系统
  - 没有服务器可以调整大小、监控或扩展
  - 降低因过度配置而导致的成本风险
  - 降低了因配置不足而对性能造成的风险

- 亚马逊 API Gateway：
  - 简化的部署、监控和保护机制 APIs
  - 通过缓存和内容交付提高了 API 性能

AWS Lambda

AWS Lambda 是一项计算服务，使您无需预置、管理或扩展服务器即可运行任意代码函数。支持的语言包括 Python、Ruby、Java、Go 和.NET。Lambda 函数在托管、隔离的容器中运行，是为了响应事件而启动的，该事件可能是几个 AWS 可用的编程触发器之一，称为事件源。有关支持的语言和事件源的更多信息，请参阅 Lambda FAQs。

Lambda 的许多常见用例都围绕着事件驱动的数据处理工作流程，例如处理存储在 Amazon S3 中的文件或从 A ma zon Kinesis 流式传输数据记录。与 Amazon API Gateway 配合使用时，Lambda 函数执行典型网络服务的功能：它启动代码以响应客户端 HTTPS 请求；API Gateway 充当逻辑层的前门，并 AWS Lambda 调用应用程序代码。

您的业务逻辑在这里，无需服务器

Lambda 要求您编写名为处理程序的代码函数，这些函数将在事件启动时运行。要将 Lambda 与 API Gateway 配合使用，您可以将 API Gateway 配置为在向您的 API 发出 HTTPS 请求时启动处理程序函

数。在无服务器多层架构中， APIs 您在 API Gateway 中创建的每个函数都将与调用所需业务逻辑的 Lambda 函数（以及其中的处理程序）集成。

使用 AWS Lambda 函数组成逻辑层，您可以定义公开应用程序功能所需的粒度级别（每个 API 一个 Lambda 函数或每个 API 方法一个 Lambda 函数）。在 Lambda 函数中，处理程序可以访问任何其他依赖项（例如，您使用代码、库、原生二进制文件和外部 Web 服务上传的其他方法），甚至是其他 Lambda 函数。

创建或更新 Lambda 函数需要将代码作为 Lambda 部署包以 zip 文件形式上传到 Amazon S3 存储桶，或者将代码与所有依赖项一起打包为容器映像。这些函数可以使用不同的部署方法，例如 AWS 管理控制台、running AWS Command Line Interface (AWS CLI) 或运行基础设施作为代码模板或框架，例如CloudFormation、AWS Serverless Application Model(AWS SAM) 或AWS Cloud Development Kit (AWS CDK)。使用这些方法中的任何一种创建函数时，需要指定部署包中的哪个方法将用作请求处理程序。您可以将同一个部署包重复用于多个 Lambda 函数定义，其中每个 Lambda 函数在同一个部署包中可能有一个唯一的处理程序。

Lambda 安全

要运行 Lambda 函数，必须由 AWS Identity and Access Management (IAM) 策略允许的事件或服务调用该函数。使用 IAM 策略，您可以创建一个 Lambda 函数，除非由您定义的 API Gateway 资源调用，否则该函数根本无法启动。可以在各种 AWS 服务中使用基于资源的策略来定义此类策略。

每个 Lambda 函数都扮演一个 IAM 角色，该角色是在部署 Lambda 函数时分配的。此 IAM 角色定义了您的 Lambda 函数可以与之交互的其他 AWS 服务和资源（例如，亚马逊 DynamoDB Amazon S3）。在 Lambda 函数的上下文中，这称为执行角色。

请勿在 Lambda 函数中存储敏感信息。IAM 通过 Lambda 执行角色处理对 AWS 服务的访问；如果您需要从 Lambda 函数内部访问其他证书（例如数据库凭证和 API 密钥），则可以将 AWS Key Management Service(AWS KMS) 与环境变量一起使用，或者使用诸如 Secrets Manager AWS之类的服务在不使用时确保这些信息的安全。

大规模性能

从亚马逊弹性容器注册表 (Amazon ECR) 或上传到 Amazon S3 的 zip 文件中作为容器镜像提取的代码在由 AWS 管理的隔离环境中运行。您不必扩展 Lambda 函数——每次您的函数收到事件通知时，都会在其计算队列中 AWS Lambda 找到可用容量，并使用您定义的运行时间、内存、磁盘和超时配置运行您的代码。通过这种模式，AWS 可以根据需要启动任意数量的函数副本。

基于 Lambda 的逻辑层大小始终适合您的客户需求。通过托管扩展和并发代码启动快速吸收激增的流量，再加上 pay-per-use Lambda 定价，使您能够始终满足客户请求，同时无需为空闲计算容量付费。

无服务器部署和管理

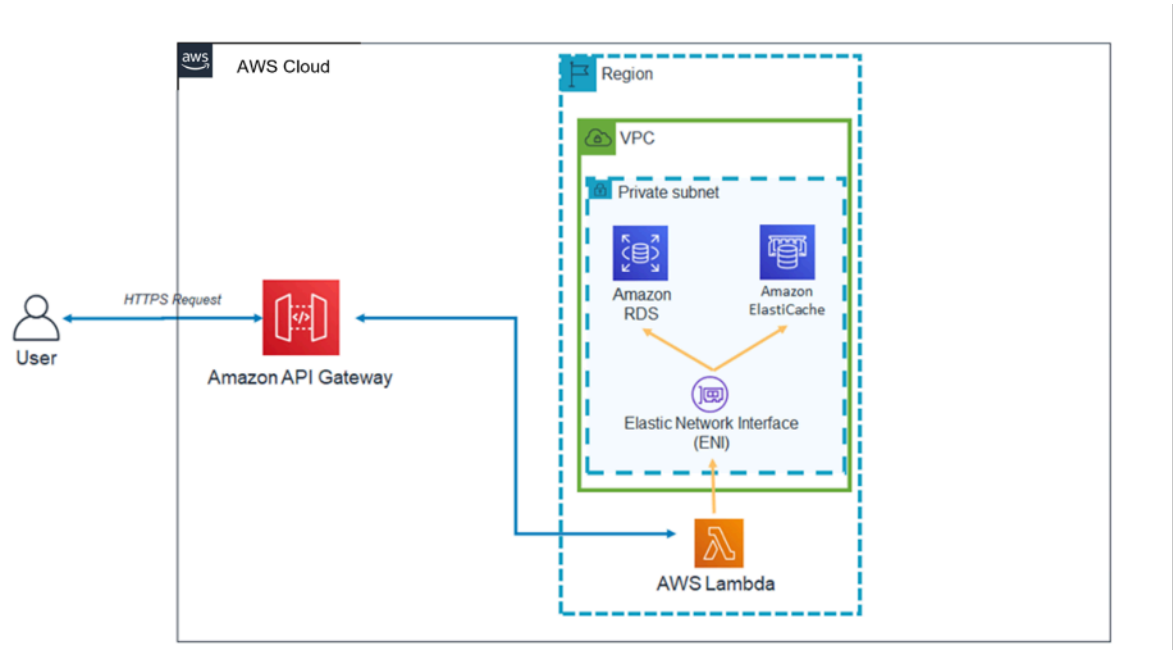
为了帮助您部署和管理 Lambda 函数，请使用 AW S 无服务器应用程序模型 (AWS SAM)，这是一个开源框架，包括：

- AWS SAM 模板规范- 用于定义您的函数并描述其环境、权限、配置和事件的语法，以简化上传和部署。
- AWS SAM CLI- 允许您验证 SAM 模板语法、在本地调用函数、调试 Lambda 函数和部署包函数的命令。

您也可以使用 AWS CDK，这是一个软件开发框架，用于使用编程语言定义云基础架构并通过它进行配置 CloudFormation。CDK 提供了一种定义 AWS 资源的必要方式，而 AWS SAM 提供了一种声明方式。

通常，当您部署 Lambda 函数时，使用由其分配的 IAM 角色定义的权限调用该函数，并且能够访问面向互联网的终端节点。作为逻辑层的核心 AWS Lambda ，组件直接与数据层集成。如果您的数据层包含敏感的业务或用户信息，则务必确保该数据层被适当隔离（在私有子网中）。

如果您希望 Lambda 函数访问无法公开的资源，例如私有数据库实例，则可以将 Lambda 函数配置为连接到 AWS 账户中虚拟私有云 (VPC) 中的私有子网。当您将函数连接到 VPC 时，Lambda 会为函数的 VPC 配置中的每个子网创建一个弹性网络接口，弹性网络接口用于私密访问您的内部资源。



VPC 内部的 Lambda 架构模式

将 Lambda 与 VPC 结合使用意味着您的业务逻辑所依赖的数据库和其他存储媒体可能无法通过互联网访问。VPC 还确保通过互联网与您的数据进行交互的唯一方法是通过您定义的 APIs 和您编写的 Lambda 代码函数。

Amazon API Gateway

Amazon API Gateway 是一项完全托管的服务，使开发人员能够以任何规模创建、发布、维护、监控和保护 APIs 。

客户端（即演示层）使用标准 HTTPS 请求与通过 API Gateway APIs 公开的内容集成。通过 API Gateway APIs 公开到面向服务的多层架构的适用性在于能够分离应用程序功能的各个部分，并通过 REST 端点公开此功能。Amazon API Gateway 具有特定的功能和品质，可以为您的逻辑层添加强大的功能。

与集成 AWS Lambda

Amazon API Gateway 同时支持 REST 和 HTTP 类型的 APIs。API Gateway API 由资源和方法组成。资源是应用程序可以通过资源路径访问的逻辑实体（例如，`/tickets`）。方法对应于提交给 API 资源的 API 请求（例如，`GET /tickets`）。API Gateway 允许您使用 Lambda 函数支持每种方法，也就是说，当您通过 API Gateway 中公开的 HTTPS 终端节点调用 API 时，API Gateway 会调用 Lambda 函数。

您可以使用代理集成和非代理集成连接 API Gateway 和 Lambda 函数。

代理集成

在代理集成中，整个客户端 HTTPS 请求按原样发送到 Lambda 函数。API Gateway 将整个客户端请求作为 Lambda 处理程序函数的事件参数传递，Lambda 函数的输出将直接返回给客户端（包括状态码、标头等）。

非代理集成

在非代理集成中，您可以配置如何将客户端请求的参数、标头和正文传递给 Lambda 处理程序函数的事件参数。此外，您还可以配置 Lambda 输出如何转换回给用户。

> **Note**
>
> API Gateway 还可以代理到外部的其他无服务器资源 AWS Lambda，例如模拟集成（对初始应用程序开发很有用），并直接代理到 S3 对象。

各地区的 API 性能稳定

Amazon API Gateway 的每次部署都包含一个亚马逊 CloudFront发行版。 CloudFront是一项内容分发服务，它使用亚马逊的全球边缘站点网络作为使用您的 API 的客户的连接点。这有助于缩短 API 的响应延迟。通过在全球范围内使用多个边缘站点，Amazon CloudFront 还提供了抵御分布式拒绝服务 (DDoS) 攻击场景的能力。有关更多信息，请查看 AW DDo S 弹性最佳实践白皮书。

您可以使用 API Gateway 将响应存储在可选的内存缓存中，从而提高特定 API 请求的性能。这种方法不仅可以为重复的 API 请求带来性能优势，还可以减少调用 Lambda 函数的次数，从而降低总体成本。

利用内置功能鼓励创新并减少开销

构建任何新应用程序的开发成本都是一项投资。使用 API Gateway 可以减少某些开发任务所需的时间并降低总开发成本，从而使组织能够更自由地进行实验和创新。

在应用程序的初始开发阶段，为了更快地交付新应用程序，通常会忽略日志记录和指标收集的实施。在将这些功能部署到生产环境中运行的应用程序时，这可能会导致技术负债和运营风险。Amazon API Gateway 与亚马逊无缝集成 CloudWatch，亚马逊从 API Gateway 收集原始数据并将其处理为可读的、近乎实时的指标，用于监控 API 的执行情况。API Gateway 还支持带有可配置报告的访问记录和调试AWS X-Ray跟踪。这些功能中的每一个都不需要编写代码，并且可以在生产环境中运行的应用程序中进行调整，而不会对核心业务逻辑造成风险。

应用程序的总体生命周期可能未知，也可能已知其寿命很短。如果您的起点已经包含 API Gateway 提供的托管功能，并且仅在 APIs 开始收到请求后才产生基础设施成本，则可以更轻松地为构建此类应用程序创建商业案例。有关更多信息，请参阅 Amazon API Gateway 定价。

快速迭代，保持敏捷

使用 Amazon API Gateway 并 AWS Lambda 构建 API 的逻辑层，可以简化 API 部署和版本管理，从而快速适应用户群不断变化的需求。

阶段部署

在 API Gateway 中部署 API 时，必须将部署与 API Gateway 阶段相关联——每个阶段都是 API 的快照，可供客户端应用程序调用。使用此惯例，您可以轻松地将应用程序部署到开发、测试、阶段或生产阶段，并在各个阶段之间移动部署。每次将 API 部署到某个阶段时，都要创建不同版本的 API，必要时可以恢复该版本。这些功能使现有功能和客户端依赖关系能够不受干扰地继续运行，同时新功能将作为单独的 API 版本发布。

与 Lambda 的解耦集成

API Gateway 中的 API 和 Lambda 函数之间的集成可以使用 API Gateway 阶段变量和 Lambda 函数别名来解耦合。这简化并加快了 API 的部署。您可以在 API 中配置阶段变量，该变量可以指向 Lambda 函数中的特定别名，而不是直接在 API 中配置 Lambda 函数名称或别名。在部署过程中，将阶段变量值更改为指向 Lambda 函数别名，API 将在特定阶段在 Lambda 别名后面运行 Lambda 函数版本。

Canary 版本部署

Canary 发布是一种软件开发策略，其中部署新版本的 API 用于测试目的，而基础版本仍作为生产版本部署，以便在同一阶段进行正常操作。在金丝雀版本部署中，API 总流量随机分为生产版本和带有预配置比例的金丝雀版本。 APIs 在 API Gateway 中，可以针对金丝雀版本部署进行配置，以便在有限的用户群中测试新功能。

自定义域名

您可以为 API 提供直观的商业友好型网址名称，而不是 API Gateway 提供的网址。API Gateway 提供了为配置自定义域名的功能 APIs。使用自定义域名，您可以设置 API 的主机名，然后选择多级基本路径（例如`myservicemyservice/cat/v1`、或`myservice/dog/v2`）将备用 URL 映射到您的 API。

优先考虑 API 安全

所有应用程序都必须确保只有经过授权的客户才能访问其 API 资源。在设计多层应用程序时，您可以利用 Amazon API Gateway 通过几种不同的方式来保护您的逻辑层：

过境安全

向您的所有请求均 APIs 可通过 HTTPS 提出，以便在传输过程中启用加密。

API Gateway 提供内置 SSL/TLS 证书 — 如果使用面向公众的自定义域名选项 APIs，则可以使用 AWS Certificate Manager 提供自己的 SSL/TLS 证书。API Gateway 还支持双向 TLS (mTLS) 身份验

证。双向 TLS 可增强您的 API 的安全性，并有助于保护您的数据免受诸如客户端欺骗或 man-in-the中间攻击之类的攻击。

API 授权

您在 API 中创建的每个 resource/method 组合都会被授予一个唯一的 Amazon 资源名称 (ARN)，该名称可以在 AWS Identity and Access Management (IAM) 策略中引用。

在 API Gateway 中为 API 添加授权的通用方法有三种：

- IAM 角色和策略：客户使用 AWS 签名版本 4 (Sigv4) 授权和 IAM 策略进行 API 访问。相同的证书可以根据需要限制或允许访问其他 AWS 服务和资源（例如 Amazon S3 存储桶或 Amazon DynamoDB 表）。
- Amazon Cognito 用户池：客户通过 Amazon Co gnito 用户池登录并获取令牌，这些令牌包含在请求的授权标头中。
- Lambda 授权方：定义一个 Lambda 函数，该函数实现使用不记名令牌策略（例如 OAuth 和 SAML）或使用请求参数来识别用户的自定义授权方案。

访问限制

API Gateway 支持生成 API 密钥以及将这些密钥与可配置的使用计划关联。您可以使用监控 API 密钥的使用情况 CloudWatch。

API Gateway 支持您的 API 中每种方法的限流、速率限制和突发速率限制。

私人 APIs

使用 API Gateway，您可以创建私有 REST APIs ，该私有 REST 只能通过接口 VPC 终端节点从 Amazon VPC 中的虚拟私有云进行访问。这是您在 VPC 中创建的端点网络接口。

使用资源策略，您可以启用或拒绝从选定的终端节点 VPCs 和 VPC 终端节点访问您的 API，包括通过 AWS 账户进行访问。每个端点可用于访问多个私有端点 APIs。您还可以使用 AWS Direct Connect 建立从本地网络到 Amazon VPC 的连接，并通过该连接访问您的私有 API。

在所有情况下，您的私有 API 的流量使用安全的连接，不会离开 Amazon 网络；它与公有 Internet 隔离开来。

使用 AWS WAF 进行防火墙保护

面向互联网的用户容易 APIs 受到恶意攻击。 AWS WAF 是一种 Web 应用程序防火墙，可帮助防范 APIs 此类攻击。它可以 APIs 防止常见的 Web 漏洞，例如 SQL 注入和跨站脚本攻击。您可以AWS WAF与 API Gateway 配合使用来帮助保护 APIs。

数据层

AWS Lambda 用作逻辑层不会限制数据层中可用的数据存储选项。Lambda 函数通过在 Lambda 部署包中包含相应的数据库驱动程序来连接到任何数据存储选项，并使用基于 IAM 角色的访问权限或加密证书（通过或 Sec AWS KMS rets Manager）。 AWS

为您的应用程序选择数据存储在很大程度上取决于您的应用程序要求。AWS 提供了许多无服务器和非服务器数据存储，您可以使用它们来构成应用程序的数据层。

无服务器数据存储选项

Amazon S3 是一项对象存储服务，提供行业领先的可扩展性、数据可用性、安全性和性能。

Amazon Aurora 是一款兼容 MySQL 且兼容 PostgreSQL 的关系数据库，专为云而构建，它将传统企业数据库的性能和可用性与开源数据库的简单性和成本效益相结合。Aurora 提供无服务器和传统使用模式。

Amazon DynamoDB 是一个键值和文档数据库，在任何规模下都能提供个位数的毫秒级性能。它是一个完全托管、无服务器、多区域、多活动、耐用的数据库，具有针对互联网规模应用程序的内置安全性、备份和恢复以及内存缓存。

Amazon Timestream 是一项快速、可扩展、完全托管的时间序列数据库服务，适用于物联网和运营应用程序，可轻松存储和分析每天数万亿个事件，成本仅为关系数据库的十分之一。在物联网设备、IT 系统和智能工业机器兴起的推动下，时间序列数据（衡量事物如何随时间推移而变化的数据）是增长最快的数据类型之一。

Amazon Quantum Ledger 数据库 (Amazon QLDB) 是一个完全托管的账本数据库，它提供由中央可信机构拥有的透明、不可变且可加密验证的交易日志。Amazon QLDB 会跟踪每一次应用程序数据更改，并维护一段时间内完整且可验证的更改历史记录。

Amazon Keyspaces（适用于 Apache Cassandra）是一项可扩展、高度可用且托管的 Apache Cassandra 兼容数据库服务。借助 Amazon Keyspaces，您可以使用与当今相同的 Cassandra 应用程序代码和开发者工具 AWS 来运行 Cassandra 工作负载。您无需配置、修补或管理服务器，也不必安装、维护或操作软件。Amazon Keyspaces 是无服务器的，因此您只需为使用的资源付费，并且该服务可以根据应用程序流量自动向上和向下扩展表。

Amazon Elastic Fil e System (Amazon EFS) 提供了一个简单 set-and-forget、无服务器的弹性文件系统，让您无需预置或管理存储即可共享文件数据。它可以与 AWS 云服务和本地资源配合使用，并且

可在不中断应用程序的情况下按需扩展到 PB 级。借助 Amazon EFS，您可以在添加和删除文件时自动扩展和缩小文件系统，无需为适应增长而配置和管理容量。Amazon EFS 可以使用 Lambda 函数挂载，这使其成为可行的文件存储选项。 APIs

非无服务器数据存储选项

Amazon Relat ional Database Service（Amazon RDS）是一项托管网络服务，它使用任何可用的引擎（亚马逊 Aurora、PostgreSQL、MySQL、MariaDB、Oracle 和微软 SQL Server），在针对内存、性能或 I/O 进行了优化的几种不同的数据库实例类型上运行，可以更轻松地设置、操作和扩展关系数据库。

Amazon Redshift 是一项完全托管的 PB 级云端数据仓库服务。

Amazon ElastiCache 是 Redis 或 Memcached 的完全托管部署。无缝部署、运行和扩展与开源兼容的常用内存数据存储。

Amazon Neptune 是一项快速、可靠、完全托管的图形数据库服务，可轻松构建和运行适用于高度互联数据集的应用程序。Neptune 支持流行的图形模型（属性图和 W3C 资源描述框架 (RDF)）及其各自的查询语言，使您能够轻松构建能够高效浏览高度互联的数据集的查询。

Amazon DocumentDB（兼容 MongoDB）是一种快速、可扩展、高度可用且完全托管的文档数据库服务，支持 MongoDB 工作负载。

最后，您还可以使用在 Amazon 上独立运行的数据存储 EC2 作为多层应用程序的数据层

演示层

演示层负责通过互联网公开的 API Gateway REST 端点与逻辑层进行交互。任何支持 HTTPS 的客户端或设备都可以与这些端点通信，从而使您的演示层可以灵活地采用多种形式（桌面应用程序、移动应用程序、网页、物联网设备等）。根据您的要求，您的演示层可以使用以下 AWS 无服务器产品：

- Amazon Cognito-一项无服务器用户身份和数据同步服务，可让您快速高效地将用户注册、登录和访问控制添加到您的 Web 和移动应用程序。Amazon Cognito 可扩展到数百万用户，支持通过 Facebook、Google 和亚马逊等社交身份提供商以及通过 SAML 2.0 登录企业身份提供商。
- Amazon S3 with CloudFront-使您能够直接从 S3 存储桶提供静态网站，例如单页应用程序，而无需配置 Web 服务器。您可以 CloudFront 用作托管内容分发网络 (CDN)，以提高性能并启用 SSL/TL 托管或自定义证书。

AWS Amplify是一组工具和服务，可以一起使用或单独使用，用于帮助前端 Web 和移动开发人员构建可扩展的全栈应用程序，由 AWS... 提供支持。Amplify 提供完全托管的服务，用于在全球部署和托管静态 Web 应用程序，由 Amazon 的可靠内容分发网络提供服务，在全球拥有数百个接入点，内置 CI/CD 工作流程可加快您的应用程序发布周期。Amplify 支持流行的网络框架 JavaScript，包括 React、Angular、Vue、Next.js，以及包括安卓、iOS、React Native、Ionic 和 Flutter 在内的移动平台。根据您的网络配置和应用程序要求，您可能需要使您的 API Gateway APIs 符合跨域资源共享 (CORS) 标准。CORS 合规性允许 Web 浏览器直接 APIs 从静态网页中调用您的。

当您使用部署网站时 CloudFront，系统会为您提供一个用于访问您的应用程序的 CloudFront 域名（例如，`d2d47p2vcczkh2.cloudfront.net`）。您可以使用 Amazon Route 53 注册域名并将其定向到您的 CloudFront 分配，或者将已拥有的域名指向您的 CloudFront 分配。这使用户能够使用熟悉的域名访问您的网站。请注意，您还可以使用 Route 53 将自定义域名分配给 API Gateway 分配，这样用户就 APIs 可以使用熟悉的域名进行调用。

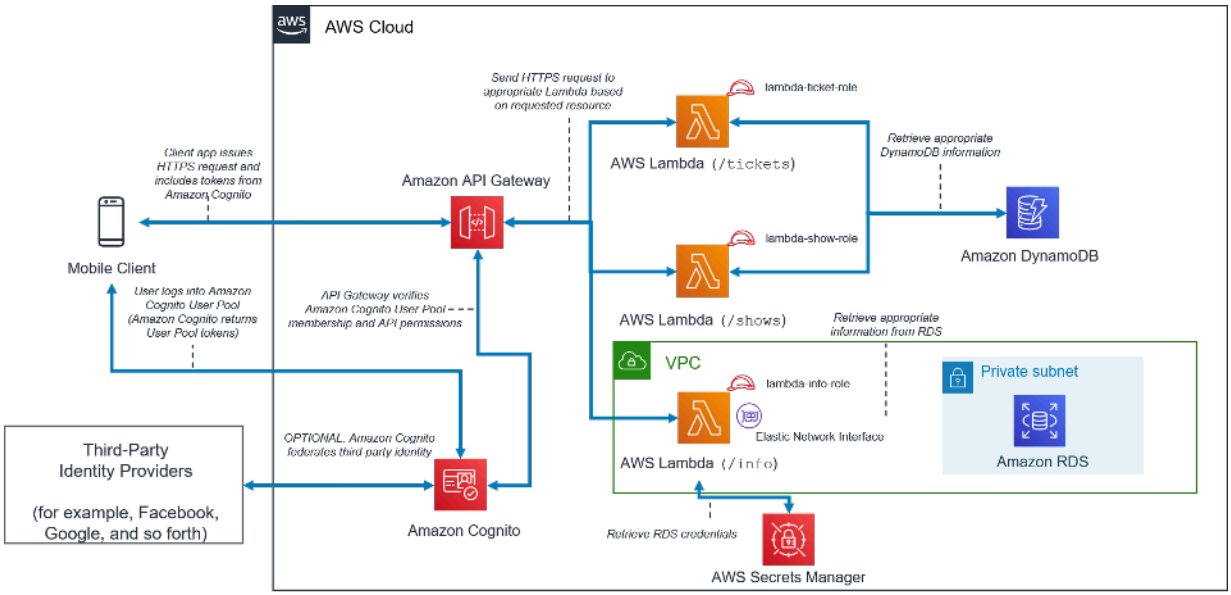
架构模式示例

您可以使用 API Gateway 和 AWS Lambda 作为您的逻辑层来实现常用的架构模式。本白皮书包括利用 AWS Lambda基于逻辑层的最流行的架构模式：

- 移动后端- 移动应用程序与 API Gateway 和 Lambda 通信以访问应用程序数据。此模式可以扩展到不使用无服务器 AWS 资源托管演示层资源（例如桌面客户端、运行的 Web 服务器等）的通用 HTTPS 客户端。 EC2
- 单页应用程序-托管在 Amazon S3 中的单页应用程序， CloudFront 可与 API Gateway 通信并 AWS Lambda 访问应用程序数据。
- Web 应用程序 — Web 应用程序是一个通用、事件驱动的 Web 应用程序后端，其业务逻辑与 API AWS Lambda Gateway 配合使用。它还使用 DynamoDB 作为其数据库，使用 Amazon Cognito 进行用户管理。所有静态内容均使用 Amplify 托管。

除了这两种模式外，本白皮书还讨论了 Lambda 和 API Gateway 对通用微服务架构的适用性。微服务架构是一种流行的模式，尽管它不是标准的三层架构，但它涉及解耦应用程序组件，并将它们部署为相互通信的无状态的独立功能单元。

移动后端

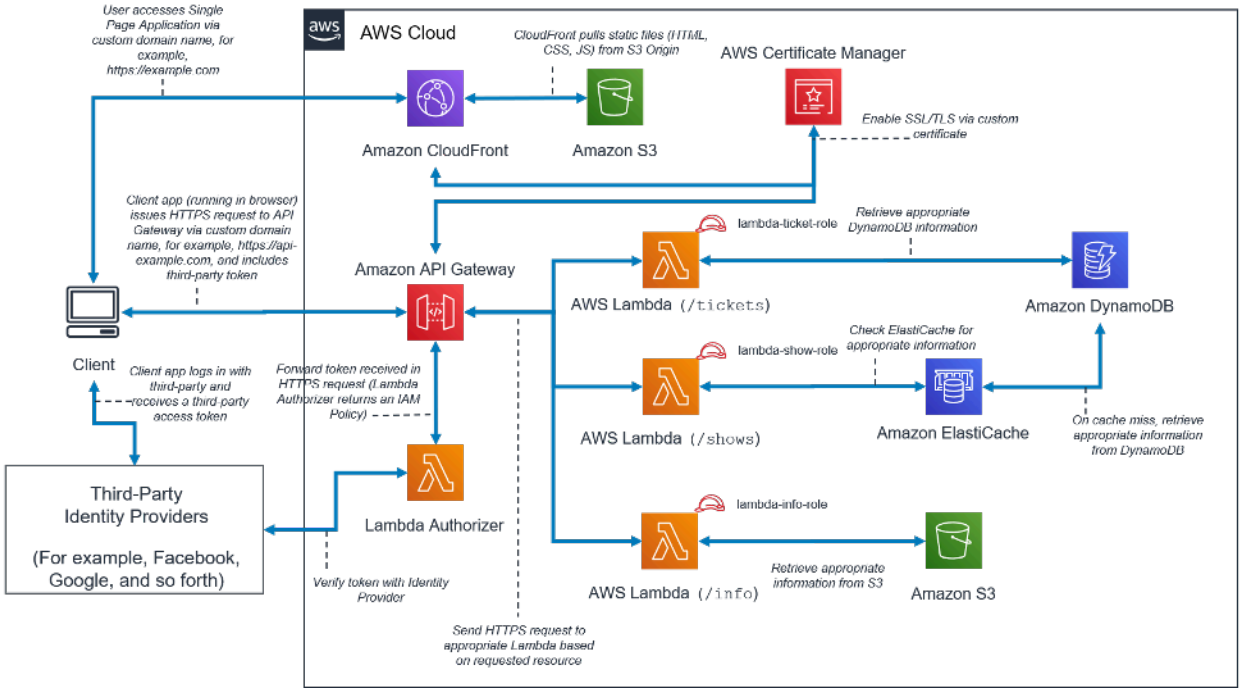


无服务器移动后端的架构模式

表 1-移动后端层组件

套餐	组件
演示	在用户设备上运行的移动应用程序。
逻辑	带有 Amazon API Gateway AWS Lambda。 此架构显示了三个公开的服务（`/tickets/shows`、和`/info`）。API Gateway 终端节点由 Amazon Cognito 用户池保护。通过这种方法，用户登录到 Amazon Cognito 用户池（必要时使用联合第三方），并获得用于授权 API Gateway 调用的访问权限和 ID 令牌。 每个 Lambda 函数都被分配了自己的身份和访问管理 (IAM) 角色，以提供对相应数据源的访问权限。
数据	DynamoDB 用于和服务。`/tickets` `/shows` 该`/info`服务使用亚马逊 RDS。此 Lambda 函数从 Secrets AWS Manager 检索 Amazon RDS 凭证，并使用弹性网络接口访问私有子网。

单页应用程序



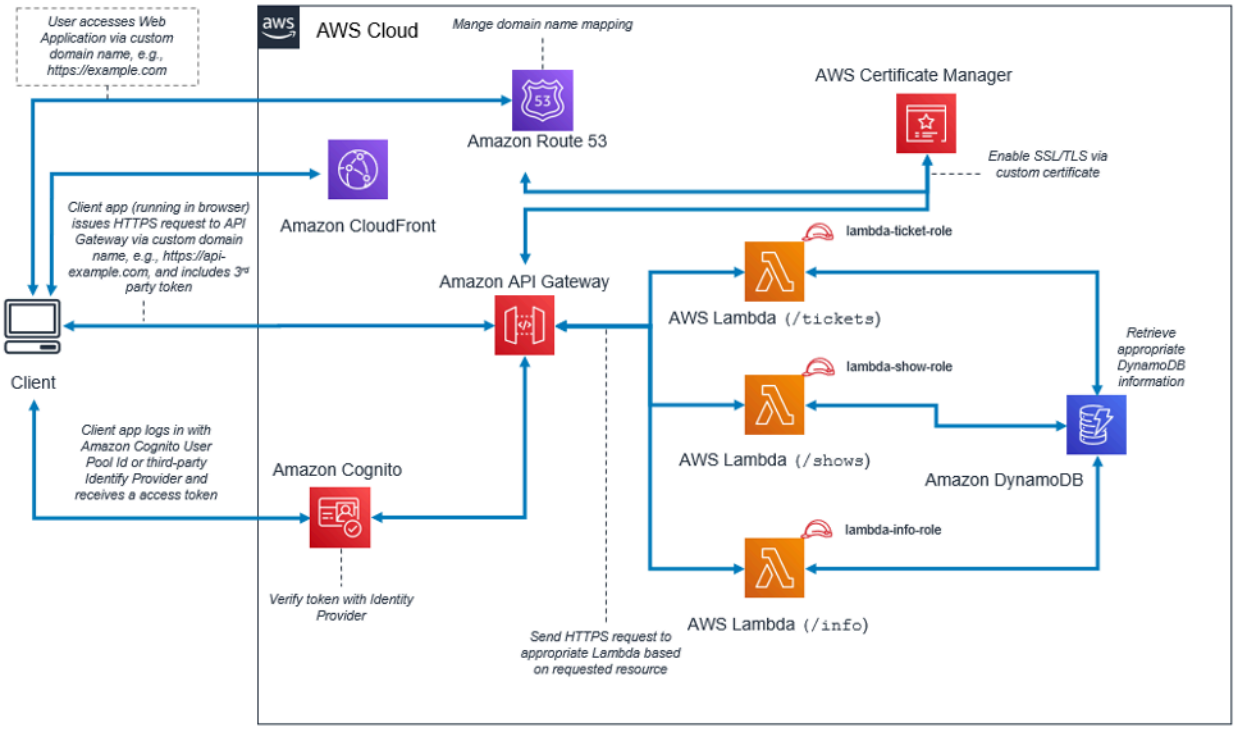
无服务器单页应用程序的架构模式

表 2-单页应用程序组件

套餐	组件
演示	在 Amazon S3 中托管的静态网站内容，由分发 CloudFront。 AWS Certifice Manager 允许使用自定义 SSL/TLS 证书。
逻辑	API Gateway 带有 AWS Lambda。 此架构显示了三个公开的服务（`/tickets`/`shows`、和`/info`）。API Gateway 终端节点由 Lambda 授权机构保护。在此方法中，用户通过第三方身份提供商登录并获取访问权限和 ID 令牌。这些令牌包含在 API Gateway 调用中，Lambda 授权机构会验证这些令牌并生成包含 API 启动权限的 IAM 策略。

套餐	组件
	每个 Lambda 函数都被分配了自己的 IAM 角色，以提供对相应数据源的访问权限。
数据	Amazon DynamoDB 用于和服务。`/tickets` / `shows` 该`/shows`服务使用Amazon ElastiCache 来提高数据库性能。缓存失误将发送到 DynamoDB。 Amazon S3 用于托管所使用的静态内容`/info service`。

Web 应用程序

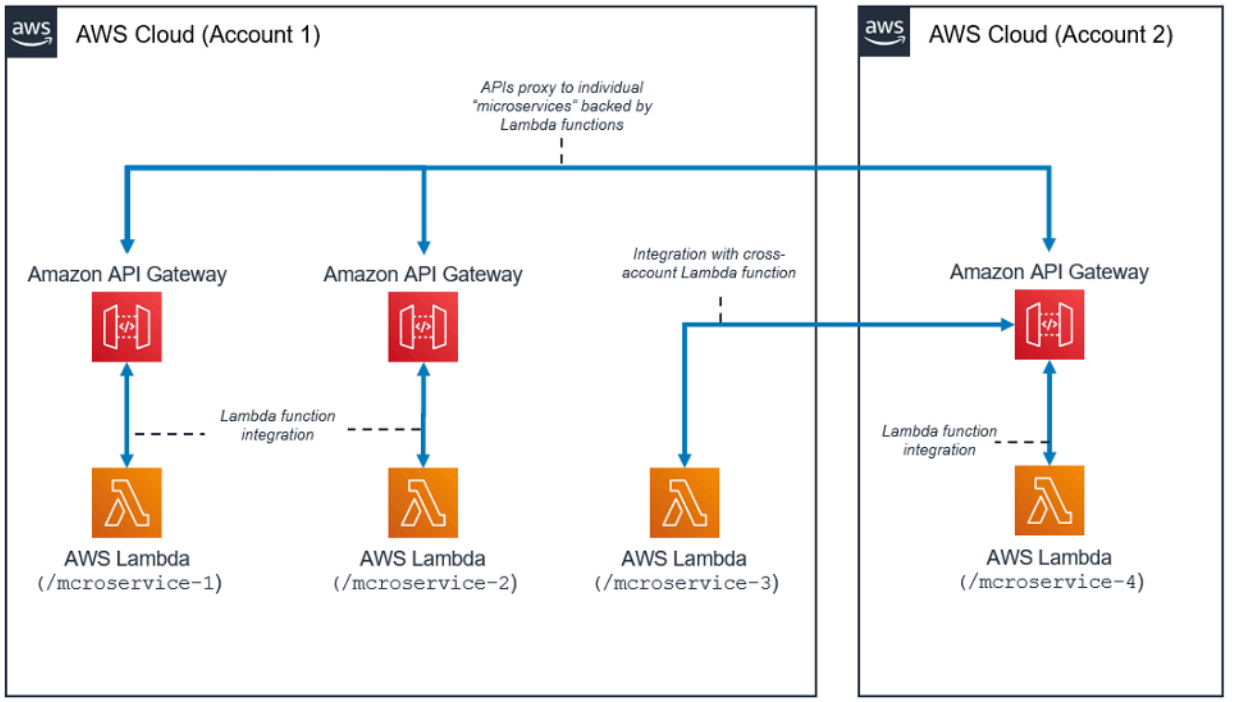


Web 应用程序的架构模式

表 3-Web 应用程序组件

套餐	组件
演示	前端应用程序是所有静态内容（HTML、CSS JavaScript 和图像），它们都是由 React 工具生成的，比如 create-react-app。Amazon CloudFront 托管了所有这些对象。使用 Web 应用程序时，会将所有资源下载到浏览器并从那里开始运行。Web 应用程序连接到后端，调用 APIs。
逻辑	逻辑层是使用 API Gateway REST 前面的 Lambda 函数构建的。 APIs 此架构显示了多个公开的服务。有多个不同的 Lambda 函数，每个函数处理应用程序的不同方面。Lambda 函数位于 API Gateway 之后，可使用 API 网址路径进行访问。 用户身份验证是使用 Amazon Cognito 用户池或联合用户提供商来处理的。API Gateway 使用与亚马逊 Cognito 的开箱即用集成。只有在用户通过身份验证后，客户端才会收到 JSON Web 令牌 (JWT) 令牌，然后在调用 API 时应使用该令牌。 每个 Lambda 函数都被分配了自己的 IAM 角色，以提供对相应数据源的访问权限。
数据	在此特定示例中，DynamoDB 用于数据存储，但可以根据用例和使用场景使用其他专门构建的 Amazon 数据库或存储服务。

采用 Lambda 的微服务



使用 Lambda 的微服务的架构模式

微服务架构模式并不局限于典型的三层架构；但是，这种流行的模式可以从使用无服务器资源中获得显著的好处。

在这种架构中，每个应用程序组件都是分离的，并且是独立部署和运行的。构建微服务所需要的只是使用 Amazon API Gateway 创建的 API 以及随后由 AWS Lambda其启动的函数。您的团队可以使用这些服务将您的环境解耦并分割到所需的粒度级别。

通常，微服务环境可能会带来以下困难：创建每项新微服务的重复开销、优化服务器密度和利用率方面的问题、同时运行多个微服务的多个版本的复杂性，以及与许多独立服务集成的客户端代码要求激增。

当你使用无服务器资源创建微服务时，这些问题变得不那么难解决，在某些情况下甚至会消失。无服务器微服务模式降低了创建每个后续微服务的门槛（API Gateway 甚至允许克隆现有 APIs微服务并在其他账户中使用 Lambda 函数）。优化服务器利用率已不再与这种模式相关。最后，Amazon API Gateway 提供了多种常用语言以编程方式生成的客户端 SDKs ，以减少集成开销。

结论

多层架构模式鼓励采用创建易于维护、分离和扩展的应用程序组件的最佳实践。当您创建一个逻辑层，在该层中由 API Gateway 进行集成并在其中进行计算时 AWS Lambda，您可以实现这些目标，同时减少实现这些目标所需的精力。这些服务共同为您的客户端提供了 HTTPS API 前端，并提供了一个安全的环境来应用您的业务逻辑，同时消除了管理典型的基于服务器的基础架构所涉及的开销。

贡献者

本文档的贡献者包括：

- Andrew Baird，AWS 解决方案架构师
- Bryant Bost，AWS 顾问 ProServe
- Stefano Buliani，AWS 移动技术高级产品经理
- Vyom Nagrani，AWS 移动高级产品经理
- Ajay Nair，AWS 移动高级产品经理
- 全球解决方案架构师 Rahul Popat
- 布拉金德拉·辛格，高级解决方案架构师

阅读更多内容

如需了解其他信息，请参阅：

- AWS 白皮书和指南

文档修订

要获得有关白皮书更新的通知，请订阅 RSS 源。

变更	说明	日期
次要更新	自始至终都修复了错误并进行了许多细微的更改。	2022 年 4 月 1 日
已更新白皮书	更新了新的服务功能和模式。	2021 年 10 月 20 日
已更新白皮书	更新了新的服务功能和模式。	2021 年 6 月 1 日
已更新白皮书	更新了新的服务功能。	2019 年 9 月 25 日
初次发布	已发布白皮书。	2015 年 11 月 1 日

版权声明

客户有责任对本文档中的信息进行单独评测。本文档：(a) 仅供参考；(b) 代表当前提供的 AWS 产品和实操，如有更改，恕不另行通知；并且 (c) AWS 及其附属机构、供应商或许可方不做任何承诺或保证。AWS 产品或服务“按原样”提供，不提供任何形式的保证、陈述或条件，无论是明示还是暗示。AWS 对其客户承担的责任和义务受 AWS 协议制约，本文档不是 AWS 与客户直接协议的一部分，也不构成对该协议的修改。