AWS 白皮書

使用 Amazon API Gateway 和 AWS Lambda 的 AWS Serverless 多層架構

使用 Amazon API Gateway 和 AWS Lambda 的 AWS Serverless 多層架構: AWS 白皮書

Table of Contents

此白皮書僅供歷史參考。有些內容可能已過時，有些連結可能無法使用。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。

使用 Amazon API Gateway 和 AWS Lambda 的 AWS Serverless 多層架構

發佈日期：2021 年 10 月 20 日 (文件修訂)

摘要

本白皮書說明如何使用 Amazon Web Services (AWS) 的創新來變更您設計多層架構的方式，並實作熱門模式，例如微服務、行動後端和單頁應用程式。架構師和開發人員可以使用 Amazon API Gateway AWS Lambda和其他 服務，以減少建立和管理多層應用程式所需的開發和操作週期。

您是 Well-Architected 嗎？

AWS Well-Architected Framework 可協助您了解在雲端建置系統時所做決策的優缺點。架構的六大支柱可讓您學習架構最佳實務，以設計和操作可靠、安全、有效率、經濟實惠且永續的系統。使用 AWS Well-Architected Tool免費提供的 AWS 管理主控台，您可以透過回答每個支柱的一組問題，根據這些最佳實務來檢閱工作負載。

在 Serverless Application Lens 中，我們著重於架構無伺服器應用程式的最佳實務 AWS。

如需雲端架構的更多專家指引和最佳實務，參考架構部署、圖表和白皮書，請參閱 AWS 架構中心。

簡介

多層應用程式（三層、n 層等）是幾十年來的基石架構模式，仍然是面向使用者的應用程式的熱門模式。雖然用來描述多層架構的語言有所不同，但多層應用程式通常包含下列元件：

- 呈現層：使用者直接互動的元件（例如網頁和行動應用程式 UIs)。
- 邏輯層：將使用者動作轉譯為應用程式功能（例如 CRUD 資料庫操作和資料處理）所需的程式碼。
- 資料層：存放應用程式相關資料的儲存媒體（例如資料庫、物件存放區、快取和檔案系統）。

多層架構模式提供一般架構，以確保分離和獨立可擴展的應用程式元件可以單獨開發、管理和維護（通常是由不同的團隊）。

由於這種模式，網路（層必須進行網路呼叫才能與另一個層互動）做為層之間的界限，開發多層應用程式通常需要建立許多未區分的應用程式元件。其中一些元件包括：

- 定義用於層之間通訊的訊息佇列的程式碼
- 定義應用程式程式設計界面 (API) 和資料模型的程式碼
- 確保適當存取應用程式的安全相關程式碼

所有這些範例都可以視為「樣板」元件，雖然在多層應用程式中是必要的，但其實作在各個應用程式之間不會有很大的差異。

AWS 提供多種服務，可建立無伺服器多層應用程式，大幅簡化將這類應用程式部署到生產環境的程序，並消除與傳統伺服器管理相關的額外負荷。Amazon API Gateway 是用於建立和管理 APIs 的服務，而 AWS Lambda是用於執行任意程式碼函數的服務，可以一起用於簡化強大的多層應用程式的建立。

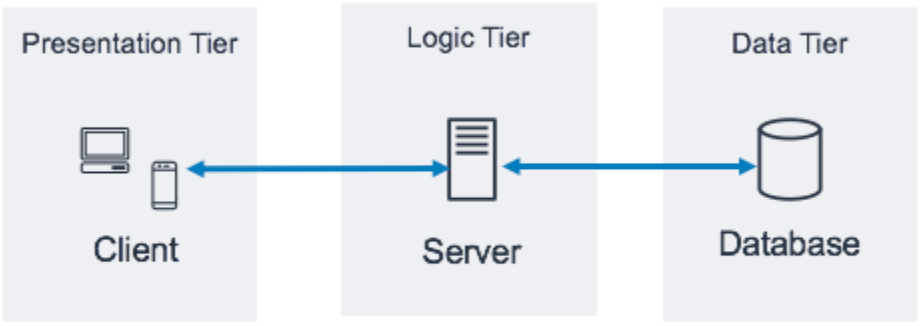
Amazon API Gateway 與 的整合 AWS Lambda 可讓使用者定義的程式碼函數直接透過 HTTPS 請求啟動。無論請求量為何，API Gateway 和 Lambda 都會自動擴展，以完全支援應用程式的需求（請參閱 API Gateway Amazon API Gateway 配額和可擴展性資訊的重要備註)。透過結合這兩個服務，您可以建立一個層，讓您只編寫對應用程式重要的程式碼，而不是專注於實作多層架構的各種其他不同層面，例如架構高可用性、撰寫用戶端 SDKs、伺服器和作業系統 (OS) 管理、擴展和實作用戶端授權機制。

API Gateway 和 Lambda 可建立無伺服器邏輯層。根據您的應用程式需求，AWS 也提供建立無伺服器呈現層（例如，使用 Amazon CloudFront 和 Amazon Simple Storage Service) 和資料層（例如，Amazon Aurora、Amazon DynamoDB) 的選項。

本白皮書著重於最熱門的多層架構範例，也就是三層 Web 應用程式。不過，您可以套用此多層模式遠超過典型的三層 Web 應用程式。

三層架構概觀

三層架構是多層架構最熱門的實作，由單一呈現層、邏輯層和資料層組成。下圖顯示簡單、通用三層應用程式的範例。



三層應用程式的架構模式

有許多絕佳的線上資源可讓您進一步了解一般三層架構模式。此白皮書著重於此架構使用 Amazon API Gateway 和 的特定實作模式 AWS Lambda。

無伺服器邏輯層

三層架構的邏輯層代表應用程式的大腦。這是使用 Amazon API Gateway 的地方，相較於傳統的伺服器型實作， AWS Lambda 可以產生最大的影響。這兩個服務的功能可讓您建置高可用性、可擴展性和安全的無伺服器應用程式。在傳統模型中，您的應用程式可能需要數千部伺服器；不過，透過使用 Amazon API Gateway AWS Lambda ，您不需要負責任何容量的伺服器管理。此外，透過將這些受管服務一起使用，您可以獲得以下好處：

- AWS Lambda:
  - 無需選擇、保護、修補或管理作業系統
  - 沒有大小、監控或擴展正確的伺服器
  - 減少過度佈建對您的成本造成的風險
  - 降低佈建不足對您的效能造成的風險

- Amazon API Gateway：
  - 簡化部署、監控和保護 APIs機制
  - 透過快取和內容交付改善 API 效能

AWS Lambda

AWS Lambda 是一種運算服務，可讓您執行任意程式碼函數，而無需佈建、管理或擴展伺服器。支援的語言包括 Python、Ruby、Java、Go 和 .NET。Lambda 函數會在受管、隔離的容器中執行，並啟動以回應事件，該事件可以是數個 AWS 可使用的程式設計觸發程序之一，稱為事件來源。如需支援的語言和事件來源的詳細資訊，請參閱 Lambda FAQs。

Lambda 的許多熱門使用案例圍繞事件驅動的資料處理工作流程，例如處理存放在 Amazon S3 中的檔案，或從 Amazon Kinesis 串流資料記錄。與 Amazon API Gateway 搭配使用時，Lambda 函數會執行典型 Web 服務的功能：它會啟動程式碼以回應用戶端 HTTPS 請求；API Gateway 充當邏輯層的前門，並 AWS Lambda 叫用應用程式碼。

您的商業邏輯在此，不需要伺服器

Lambda 要求您編寫稱為處理常式的程式碼函數，該函數會在由事件啟動時執行。若要將 Lambda 與 API Gateway 搭配使用，您可以設定 API Gateway 在對 API 提出 HTTPS 請求時啟動處理常式函數。

在無伺服器多層架構中，您在 APIs中建立的每個 API 都會與叫用所需商業邏輯的 Lambda 函數（以及其中的處理常式）整合。

使用 AWS Lambda 函數來編寫邏輯層可讓您定義所需的精細程度，以公開應用程式功能（每個 API 一個 Lambda 函數或每個 API 方法一個 Lambda 函數）。在 Lambda 函數中，處理常式可以聯絡任何其他相依性（例如，您使用程式碼、程式庫、原生二進位檔和外部 Web 服務上傳的其他方法），甚至是其他 Lambda 函數。

建立或更新 Lambda 函數需要將程式碼作為 Lambda 部署套件上傳至 Amazon S3 儲存貯體，或封裝程式碼作為容器映像以及所有相依性。函數可以使用不同的部署方法，例如 AWS 管理主控台、執行 AWS Command Line Interface (AWS CLI) 或執行基礎設施做為程式碼範本或架構，例如 CloudFormation、 AWS Serverless Application Model(AWS SAM) 或 AWS Cloud Development Kit (AWS CDK)。當您使用這些方法建立函數時，您可以指定部署套件中的哪個方法將做為請求處理常式。您可以針對多個 Lambda 函數定義重複使用相同的部署套件，其中每個 Lambda 函數在相同的部署套件中可能會有唯一的處理常式。

Lambda 安全性

若要執行 Lambda 函數，必須由 AWS Identity and Access Management (IAM) 政策允許的事件或服務叫用。使用 IAM 政策，您可以建立完全無法啟動的 Lambda 函數，除非您定義的 API Gateway 資源叫用它。這類政策可以使用各種 AWS 服務的資源型政策來定義。

每個 Lambda 函數都會擔任部署 Lambda 函數時指派的 IAM 角色。此 IAM 角色會定義 Lambda 函數可以互動的其他 AWS 服務和資源（例如，Amazon DynamoDB Amazon S3)。在 Lambda 函數的內容中，這稱為執行角色。

請勿在 Lambda 函數中存放敏感資訊。IAM 會透過 Lambda 執行角色處理對 AWS 服務的存取；如果您需要從 Lambda 函數內部存取其他登入資料（例如資料庫登入資料和 API 金鑰），您可以使用 AWS Key Management Service(AWS KMS) 搭配環境變數，或使用 AWS Secrets Manager 等服務，在不使用時保護此資訊的安全。

大規模效能

從 Amazon Elastic Container Registry (Amazon ECR) 或從上傳至 Amazon S3 的 zip 檔案提取的程式碼，會在 AWS 管理的隔離環境中執行。您不必擴展 Lambda 函數 - 每次函數收到事件通知時，都會在其運算機群中 AWS Lambda 放置可用容量，並使用您定義的執行時間、記憶體、磁碟和逾時組態執行程式碼。透過此模式，AWS 可以視需要啟動任意數量的 函式複本。

Lambda 型邏輯層的大小一律適合您的客戶需求。透過受管擴展和並行程式碼啟動快速吸收流量激增的能力，結合 Lambda pay-per-use定價，可讓您始終滿足客戶請求，同時不支付閒置的運算容量。

無伺服器部署和管理

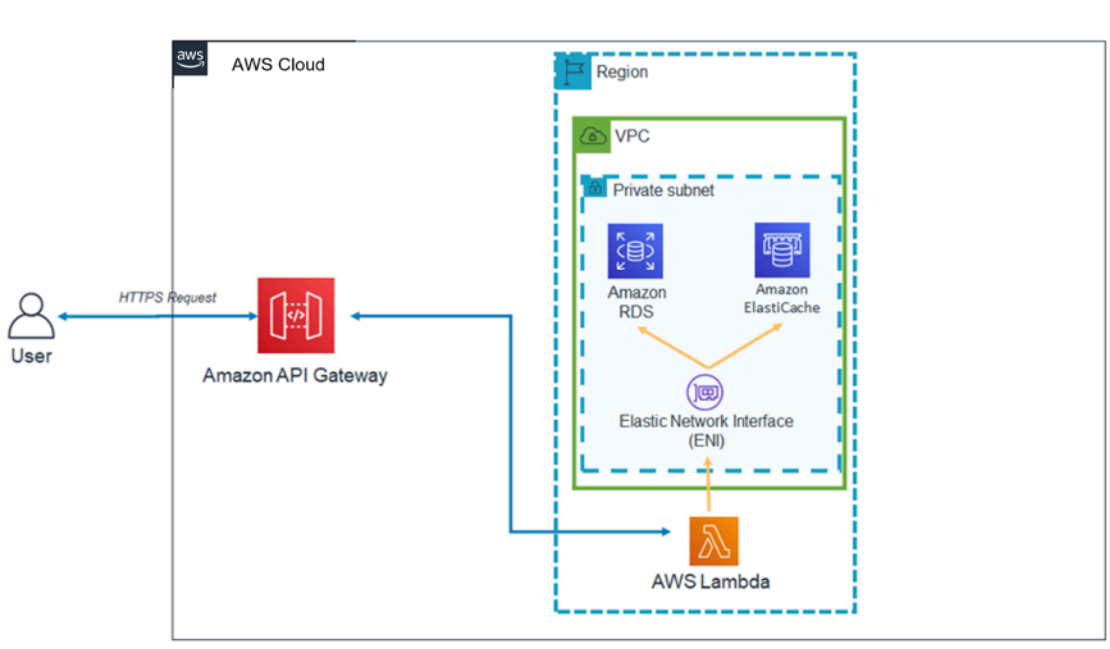
為了協助您部署和管理 Lambda 函數，請使用 AWS Serverless Application Model (AWS SAM)，這是一個開放原始碼架構，其中包含：

- AWS SAM 範本規格 - 用來定義函數並描述其環境、許可、組態和事件的語法，以簡化上傳和部署。
- AWS SAM CLI - 可讓您驗證 SAM 範本語法、在本機叫用函數、偵錯 Lambda 函數和部署套件函數的命令。

您也可以使用 AWS CDK，這是使用程式設計語言定義雲端基礎設施，並透過 CloudFormation 佈建雲端基礎設施的軟體開發架構。CDK 提供定義 AWS 資源的必要方式，而 AWS SAM 則提供宣告方式。

一般而言，當您部署 Lambda 函數時，會使用其指派的 IAM 角色所定義的許可來叫用它，並且能夠連接面向網際網路的端點。作為邏輯層的核心， AWS Lambda 是直接與資料層整合的元件。如果您的資料層包含敏感業務或使用者資訊，請務必確保此資料層已適當隔離 （在私有子網路中）。

如果您希望 Lambda 函數存取您無法公開的資源，例如私有資料庫執行個體，您可以將 Lambda 函數設定為連線至 AWS 帳戶中虛擬私有雲端 (VPC) 中的私有子網路。當您將函數連線至 VPC 時，Lambda 會為函數 VPC 組態中的每個子網路建立彈性網路界面，並使用彈性網路界面來私下存取您的內部資源。



VPC 內的 Lambda 架構模式

使用 Lambda 搭配 VPC 表示您的商業邏輯所依賴的資料庫和其他儲存媒體無法透過網際網路存取。VPC 也確保從網際網路與資料互動的唯一方法是透過您定義的 APIs 和您寫入的 Lambda 程式碼函數。

Amazon API Gateway

Amazon API Gateway 是一項全受管服務，可讓開發人員建立、發佈、維護、監控和保護任何規模 APIs。

用戶端（即簡報層）與使用標準 HTTPS 請求透過 API Gateway 公開APIs API 整合。透過 APIs Gateway 公開至服務導向的多層架構之 API 的適用性，是分離個別應用程式功能並透過 REST 端點公開此功能的能力。Amazon API Gateway 具有特定的功能和品質，可將強大的功能新增至邏輯層。

與 整合 AWS Lambda

Amazon API Gateway 同時支援 REST 和 HTTP 類型的 APIs。API Gateway API 由資源和方法組成。資源是應用程式可以透過資源路徑（例如，）存取的邏輯實體`/tickets`。方法對應至提交至 API 資源的 API 請求（例如，`GET /tickets`)。API Gateway 可讓您使用 Lambda 函數傳回每個方法，也就是說，當您透過 API Gateway 中公開的 HTTPS 端點呼叫 API 時，API Gateway 會叫用 Lambda 函數。

您可以使用代理整合和非代理整合來連接 API Gateway 和 Lambda 函數。

Proxy 整合

在代理整合中，整個用戶端 HTTPS 請求會依原狀傳送至 Lambda 函數。API Gateway 會將整個用戶端請求做為 Lambda 處理常式函數的事件參數傳遞，並將 Lambda 函數的輸出直接傳回給用戶端（包括狀態碼、標頭等）。

非代理伺服器整合

在非代理整合中，您可以設定如何將用戶端請求的參數、標頭和內文傳遞至 Lambda 處理常式函數的事件參數。此外，您可以設定如何將 Lambda 輸出翻譯回使用者。

> **Note**
>
> API Gateway 也可以代理到外部的其他無伺服器資源 AWS Lambda，例如模擬整合（適用於初始應用程式開發），以及直接代理到 S3 物件。

跨區域的穩定 API 效能

Amazon API Gateway 的每個部署都包含在幕後的 Amazon CloudFront 分佈。CloudFront 是一種內容交付服務，使用 Amazon 全球節點網路做為用戶端使用 API 的連線點。這有助於減少 API 的回應延遲。透過在全球使用多個節點，Amazon CloudFront 也提供對抗分散式阻斷服務 (DDoS) 攻擊案例的功能。如需詳細資訊，請參閱 AWS DDoS 彈性最佳實務白皮書。

您可以使用 API Gateway 將回應存放在選用的記憶體內快取中，以改善特定 API 請求的效能。這種方法不僅為重複的 API 請求提供效能優勢，還可以減少叫用 Lambda 函數的次數，從而降低整體成本。

使用內建功能鼓勵創新並減少額外負荷

建置任何新應用程式的開發成本是一項投資。使用 API Gateway 可以減少特定開發任務所需的時間，並降低總開發成本，讓組織能夠更自由地實驗和創新。

在初始應用程式開發階段，記錄和指標收集的實作通常被忽略，以更快地交付新的應用程式。將這些功能部署到在生產環境中執行的應用程式時，這可能會導致技術負債和營運風險。Amazon API Gateway 與 Amazon CloudWatch 無縫整合，可收集來自 API Gateway 的原始資料，並將其處理為可讀且幾近即時的指標，以監控 API 執行。API Gateway 也支援具有可設定報告和AWS X-Ray追蹤偵錯的存取記錄。這些功能都不需要編寫程式碼，而且可以在生產環境中執行的應用程式中進行調整，而不會對核心業務邏輯造成風險。

應用程式的整體生命週期可能不明，或已知短期。如果您的起點已包含 API Gateway 提供的受管功能，而且只有在 APIs 開始接收請求之後才會產生基礎設施成本，則可以更輕鬆地建立用於建置此類應用程式的商業案例。如需詳細資訊，請參閱 Amazon API Gateway 定價。

快速反覆運算，保持敏捷

使用 Amazon API Gateway 和 AWS Lambda 建置 API 的邏輯層，可讓您透過簡化 API 部署和版本管理，快速適應使用者群不斷變化的需求。

階段部署

當您在 API Gateway 中部署 API 時，您必須將部署與 API Gateway 階段建立關聯 – 每個階段都是 API 的快照，可供用戶端應用程式呼叫。使用此慣例，您可以輕鬆部署應用程式以開發、測試、階段或生產階段，並在階段之間移動部署。每次將 API 部署到階段時，您都會建立不同的 API 版本，並視需要還原。這些功能可讓現有功能和用戶端相依性繼續不受干擾，同時將新功能發佈為單獨的 API 版本。

與 Lambda 的解耦整合

API Gateway 和 Lambda 函數中的 API 整合可以使用 API Gateway 階段變數和 Lambda 函數別名來解耦。這可簡化並加速 API 部署。除了直接在 API 中設定 Lambda 函數名稱或別名，您可以在 API 中設定階段變數，以指向 Lambda 函數中的特定別名。在部署期間，變更階段變數值以指向 Lambda 函數別名，API 將在特定階段的 Lambda 別名後方執行 Lambda 函數版本。

Canary 版本部署

Canary Release 是一種軟體開發策略，其中部署了新版本的 API 以供測試之用，而基礎版本仍會部署為相同階段上正常操作的生產版本。在 Canary Release 部署中，總 API 流量隨機分成生產版本，以及具有預先設定比率的 Canary Release。可以為 Canary Release 部署設定 API Gateway APIs，以使用有限的一組使用者測試新功能。

自訂網域名稱

您可以為 API 提供直觀的商業易用 URL 名稱，而不是 API Gateway 提供的 URL。API Gateway 提供設定 APIs 自訂網域的功能。使用自訂網域名稱，您可以設定 API 的主機名稱，然後選擇多層基礎路徑 （例如 `myservice/cat/v1`、 `myservice`或 `myservice/dog/v2`），將替代 URL 映射至您的 API。

優先考慮 API 安全性

所有應用程式必須確保只有授權的用戶端才能存取其 API 資源。設計多層應用程式時，您可以利用 Amazon API Gateway 為保護邏輯層所做的多種不同方式：

傳輸安全性

所有對 APIs請求都可以透過 HTTPS 提出，以啟用傳輸中的加密。

API Gateway 提供內建 SSL/TLS 憑證 – 如果針對面向公眾APIs 使用自訂網域名稱選項，您可以使用 AWS Certificate Manager 提供自己的 SSL/TLS 憑證。API Gateway 也支援交互 TLS (mTLS) 身分驗證。相互 TLS 可增強 API 的安全性，並協助保護您的資料免受用戶端詐騙或中間man-in-the攻擊等攻擊。

API 授權

您在 API 中建立的每個資源/方法組合都會獲得唯一的 Amazon Resource Name (ARN)，可在 AWS Identity and Access Management (IAM) 政策中參考。

有三種一般方法可將授權新增至 API Gateway 中的 API：

- IAM 角色和政策：用戶端使用 AWS Signature 第 4 版 (SigV4) 授權和 IAM 政策進行 API 存取。相同的登入資料可以視需要限制或允許存取其他 AWS 服務和資源（例如，Amazon S3 儲存貯體或 Amazon DynamoDB 資料表）。
- Amazon Cognito 使用者集區：用戶端透過 Amazon Cognito 使用者集區登入並取得權杖，這些權杖包含在請求的授權標頭中。
- Lambda 授權方：定義 Lambda 函數，實作使用承載字符策略（例如 OAuth 和 SAML) 或使用請求參數來識別使用者的自訂授權機制。

存取限制

API Gateway 支援產生 API 金鑰，以及將這些金鑰與可設定的用量計劃建立關聯。您可以使用 CloudWatch 監控 API 金鑰用量。

API Gateway 支援 API 中每個方法的限流、速率限制和高載速率限制。

私有 API

使用 API Gateway，您可以建立私有 REST APIs，只能使用界面 VPC 端點從 Amazon VPC 中的虛擬私有雲端存取。這是您在 VPC 中建立的端點網路介面。

使用資源政策，您可以從選取的 VPCs存取，包括跨 AWS 帳戶。每個端點可用來存取多個私有 API。您也可以使用 AWS Direct Connect 在內部部署網路與 Amazon VPC 之間建立連線，並經由該連線來存取私有 API。

在所有情況下，進出您私有 API 的流量都會使用安全連線，且留在 Amazon 網路中，並與公有網際網路隔離。

使用 AWS WAF 的防火牆保護

面向網際網路APIs 容易遭受惡意攻擊。 AWS WAF 是一種 Web 應用程式防火牆，可協助保護 APIs 免受此類攻擊。它保護 APIs 免受 SQL Injection 和跨網站指令碼攻擊等常見 Web 入侵。您可以使用 AWS WAF搭配 API Gateway 來協助保護 APIs。

資料層

使用 AWS Lambda 做為邏輯層不會限制資料層中可用的資料儲存選項。Lambda 函數透過在 Lambda 部署套件中包含適當的資料庫驅動程式來連線至任何資料儲存選項，並使用 IAM 角色型存取或加密的登入資料（透過 AWS KMS 或 AWS Secrets Manager)。

為您的應用程式選擇資料存放區高度取決於您的應用程式需求。AWS 提供多種無伺服器和非伺服器資料存放區，可用來編寫應用程式的資料層。

無伺服器資料儲存選項

Amazon S3 是一種物件儲存服務，可提供業界領先的可擴展性、資料可用性、安全性和效能。

Amazon Aurora 是為雲端建置的 MySQL 相容和 PostgreSQL 相容關聯式資料庫，結合了傳統企業資料庫的效能和可用性，以及開放原始碼資料庫的簡單性和成本效益。Aurora 提供無伺服器和傳統用量模型。

Amazon DynamoDB 是金鑰值和文件資料庫，可在任何規模提供單一位數毫秒的效能。它是一個全受管、無伺服器、多區域、多活動、耐用的資料庫，具有內建安全性、備份和還原，以及適用於網際網路規模應用程式的記憶體內快取。

Amazon Timestream 是適用於 IoT 和操作應用程式的快速、可擴展、全受管時間序列資料庫服務，可讓您輕鬆地以關聯式資料庫成本的 1/10 每天儲存和分析數兆個事件。在 IoT 裝置、IT 系統和智慧型工業機器的崛起下，時間序列資料 - 測量隨著時間變化的資料 - 是成長速度最快的資料類型之一。

Amazon Quantum Ledger Database (Amazon QLDB) 是全受管總帳資料庫，可提供中央信任授權機構擁有的透明、不可變和密碼編譯可驗證的交易日誌。Amazon QLDB 會追蹤每個應用程式資料變更，並維護一段時間內完整且可驗證的變更歷史記錄。

Amazon Keyspaces （適用於 Apache Cassandra) 是一種可擴展、高可用性且受管的 Apache Cassandra 相容資料庫服務。透過 Amazon Keyspaces，您可以使用您目前使用的 AWS 相同 Cassandra 應用程式程式碼和開發人員工具，在 上執行 Cassandra 工作負載。您不需要佈建、修補或管理伺服器，也不需要安裝、維護或操作軟體。Amazon Keyspaces 是無伺服器，因此您只需支付使用的資源，而服務可以自動擴展和縮減資料表，以回應應用程式流量。

Amazon Elastic File System (Amazon EFS) 提供簡單、無伺服器、set-and-forget的彈性檔案系統，可讓您共享檔案資料，而無需佈建或管理儲存。它可以與 AWS 雲端服務和內部部署資源搭配使用，並且可隨需擴展至 PB，而不會中斷應用程式。使用 Amazon EFS，您可以在新增和移除檔案時自動擴展和

縮減檔案系統，無需佈建和管理容量來適應成長。Amazon EFS 可以使用 Lambda 函數掛載，這使得它成為 APIs的可行檔案儲存選項。

非伺服器資料儲存選項

Amazon Relational Database Service (Amazon RDS) 是一種受管 Web 服務，可讓您更輕鬆地使用任何可用的引擎 (Amazon Aurora、PostgreSQL、MySQL、MariaDB、Oracle 和 Microsoft SQL Server) 設定、操作和擴展關聯式資料庫，並在針對記憶體、效能或 I/O 最佳化的數種不同資料庫執行個體類型上執行。

Amazon Redshift 是雲端中全受管的 PB 級資料倉儲服務。

Amazon ElastiCache 是 Redis 或 Memcached 的全受管部署。無縫部署、執行和擴展與記憶體內資料存放區相容的熱門開放原始碼。

Amazon Neptune 是一種快速、可靠、全受管的圖形資料庫服務，可讓您輕鬆建置和執行可搭配高度連線資料集使用的應用程式。Neptune 支援熱門圖形模型 - 屬性圖形和 W3C 資源描述架構 (RDF) 及其個別的查詢語言，讓您能夠輕鬆建置查詢，以有效率地導覽高度連線的資料集。

Amazon DocumentDB （與 MongoDB 相容） 是一種快速、可擴展、高可用性且全受管的文件資料庫服務，可支援 MongoDB 工作負載。

最後，您也可以使用在 Amazon EC2 上獨立執行的資料存放區，做為多層應用程式的資料層

簡報層

簡報層負責透過透過網際網路公開的 API Gateway REST 端點與邏輯層互動。任何支援 HTTPS 的用戶端或裝置都可以與這些端點通訊，讓您的簡報層能夠靈活地採用多種形式（桌上型電腦應用程式、行動應用程式、網頁、IoT 裝置等）。根據您的需求，您的簡報層可以使用下列無 AWS 伺服器方案：

- Amazon Cognito - 無伺服器使用者身分和資料同步服務，可讓您快速有效地將使用者註冊、登入和存取控制新增至您的 Web 和行動應用程式。Amazon Cognito 擴展到數百萬使用者，並支援透過 SAML 2.0 登入社交身分提供者，例如 Facebook、Google 和 Amazon 以及企業身分提供者。
- Amazon S3 with CloudFront - 可讓您直接從 S3 儲存貯體提供靜態網站，例如單頁應用程式，而無需佈建 Web 伺服器。您可以使用 CloudFront 做為受管內容交付網路 (CDN)，以改善效能，並使用受管或自訂憑證啟用 SSL/TL。

AWS Amplify 是一組工具和服務，可搭配使用或單獨使用，協助前端 Web 和行動開發人員建置採用 技術的可擴展完整堆疊應用程式 AWS。Amplify 提供全受管服務，以在全球部署和託管靜態 Web 應用程式，由 Amazon 可靠的 CDN 提供，在全球有數百個據點，並具有可加速應用程式發行週期的內建 CI/CD 工作流程。Amplify 支援熱門的 Web 架構，包括 JavaScript、React、Angular、Vue、Next.js，以及 Android、iOS、React Native、Ionic 和 Flutter 等行動平台。根據您的聯網組態和應用程式需求，您可能需要讓 API Gateway APIs符合跨來源資源共用 (CORS)。CORS 合規允許 Web 瀏覽器直接從靜態網頁中調用您的 APIs。

當您使用 CloudFront 部署網站時，系統會提供您一個 CloudFront 網域名稱來連接您的應用程式（例如，`d2d47p2vcczkh2.cloudfront.net`)。您可以使用 Amazon Route 53 註冊網域名稱並將其導向至您的 CloudFront 分佈，或將已擁有的網域名稱導向至您的 CloudFront 分佈。這可讓使用者使用熟悉的網域名稱存取您的網站。請注意，您也可以使用 Route 53 將自訂網域名稱指派給 API Gateway 分佈，這可讓使用者使用熟悉APIs。

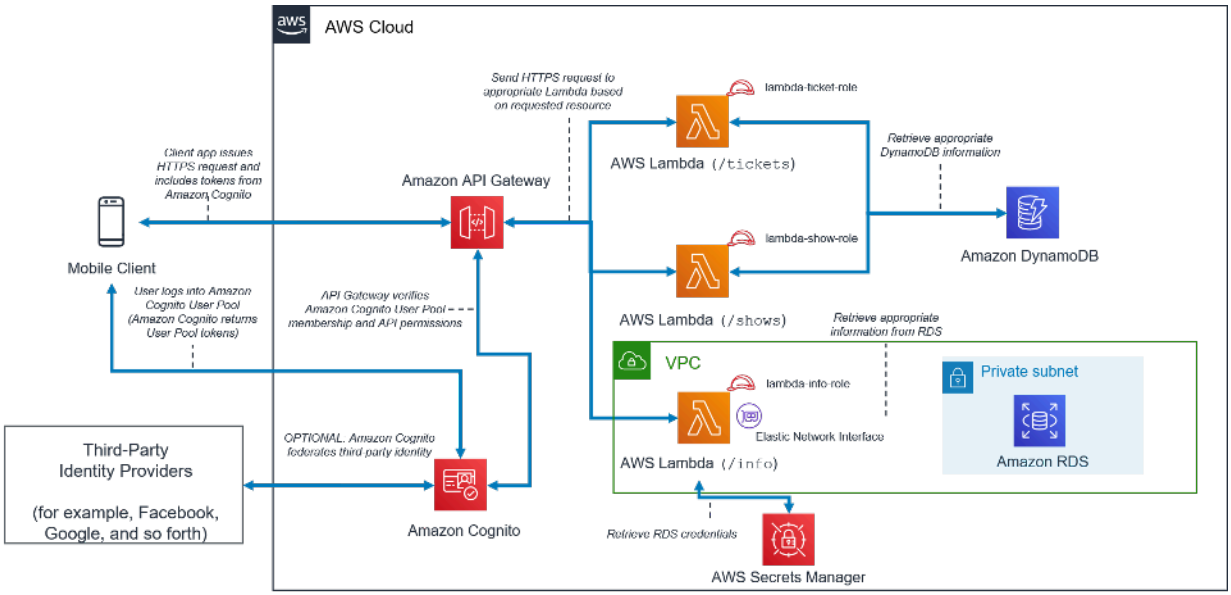
範例架構模式

您可以使用 API Gateway 和 AWS Lambda 作為邏輯層來實作熱門的架構模式。此白皮書包含利用 AWS Lambda邏輯層的最熱門架構模式：

- 行動後端 - 行動應用程式會與 API Gateway 和 Lambda 通訊，以存取應用程式資料。此模式可以延伸到未使用無伺服器 AWS 資源來託管簡報層資源的一般 HTTPS 用戶端（例如桌面用戶端、在 EC2 上執行的 Web 伺服器等）。
- 單一頁面應用程式 - Amazon S3 和 CloudFront 中託管的單一頁面應用程式會與 API Gateway 通訊 AWS Lambda 並存取應用程式資料。
- Web 應用程式 – Web 應用程式是一種一般用途、事件驅動的 Web 應用程式後端，可 AWS Lambda 搭配 API Gateway 用於其商業邏輯。它也會使用 DynamoDB 做為其資料庫，並使用 Amazon Cognito 進行使用者管理。所有靜態內容都是使用 Amplify 託管。

除了這兩種模式之外，本白皮書討論 Lambda 和 API Gateway 對一般微服務架構的適用性。微服務架構是一種熱門模式，雖然不是標準三層架構，但涉及解耦應用程式元件，並將其部署為彼此通訊的無狀態個別功能單位。

行動後端

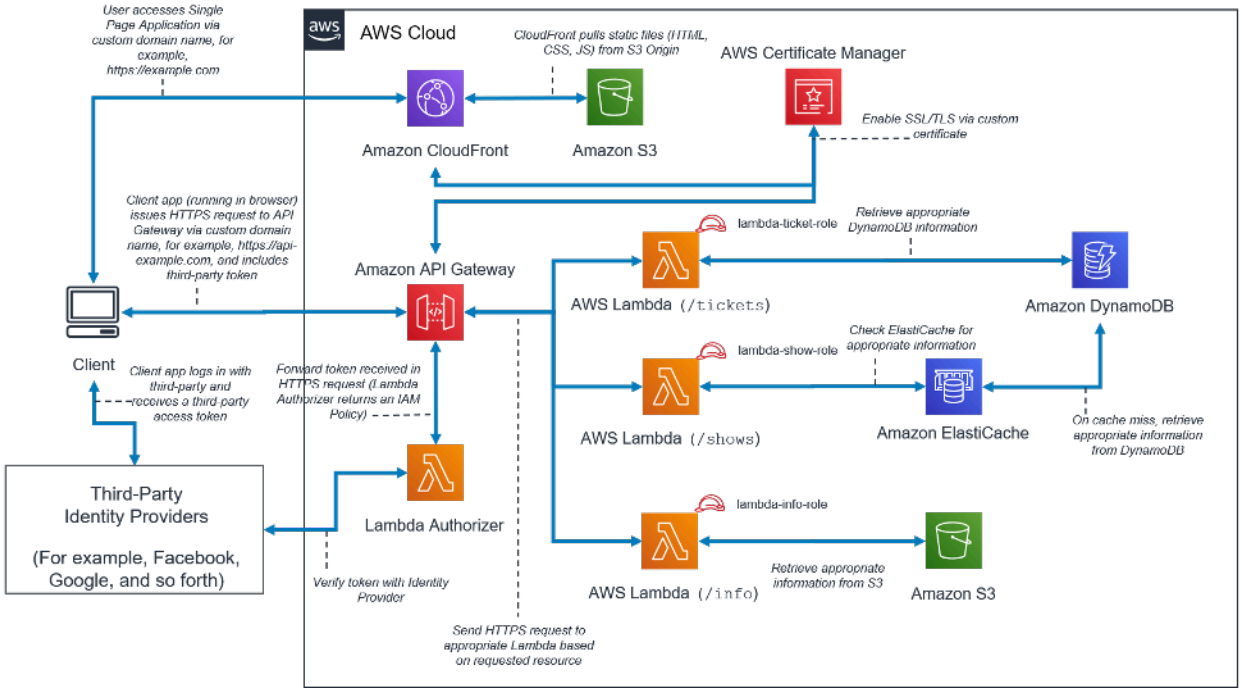


無伺服器行動後端的架構模式

表 1 - 行動後端層元件

層	元件
簡報	在使用者裝置上執行的行動應用程式。
Logic (邏輯)	Amazon API Gateway 搭配 AWS Lambda。 此架構顯示三個公開的服務 (`/tickets`、`/shows`和 `/info`)。API Gateway 端點由 Amazon Cognito 使用者集區保護 在此方法中，使用者會登入 Amazon Cognito 使用者集區（視需要使用聯合第三方），並接收用於授權 API Gateway 呼叫的存取和 ID 字符。 每個 Lambda 函數都會指派自己的 Identity and Access Management (IAM) 角色，以提供適當資料來源的存取權。
資料	DynamoDB 用於 `/tickets`和 `/shows`服務。 Amazon RDS 用於`/info`服務。此 Lambda 函數會從 AWS Secrets Manager 擷取 Amazon RDS 登入資料，並使用彈性網路界面存取私有子網路。

單頁應用程式



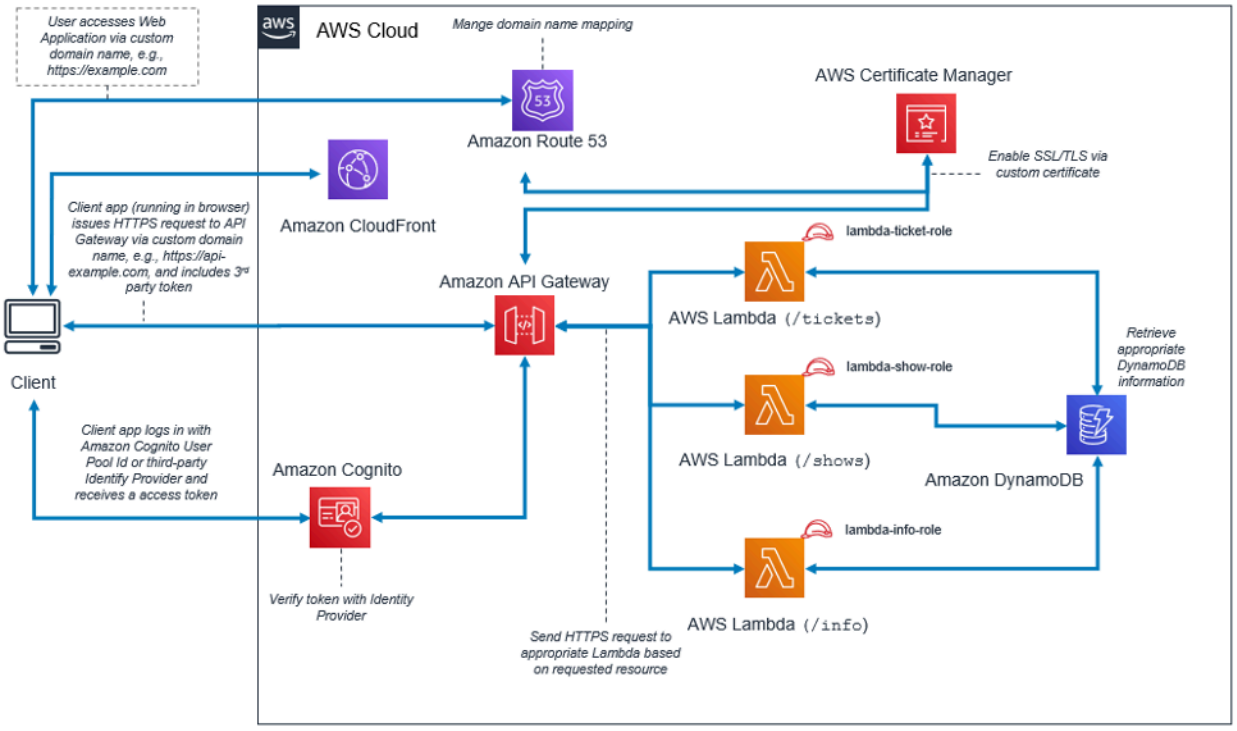
無伺服器單頁應用程式的架構模式

表 2 - 單頁應用程式元件

層	元件
簡報	Amazon S3 中託管的靜態網站內容，由 CloudFront 分發。 AWS Certificate Manager 允許使用自訂 SSL/TLS 憑證。
Logic (邏輯)	搭配 的 API Gateway AWS Lambda。 此架構顯示三個公開的服務 (`/tickets`、`/shows`和 `/info`)。API Gateway 端點由 Lambda 授權方保護。在此方法中，使用者透過第三方身分提供者登入，並取得存取權和 ID 字符。這些字符包含在 API Gateway 呼叫中，Lambda 授權方會驗證這些字符並產生包含 API 啟動許可的 IAM 政策。

層	元件
	每個 Lambda 函數都會指派自己的 IAM 角色，以提供適當資料來源的存取權。
資料	Amazon DynamoDB 用於 `/tickets`和 `/shows`服務。 `/shows` 服務會使用 Amazon ElastiCache 來改善資料庫效能。快取遺漏會傳送至 DynamoDB。 Amazon S3 用於託管 使用的靜態內容`/info service`。

Web 應用程式

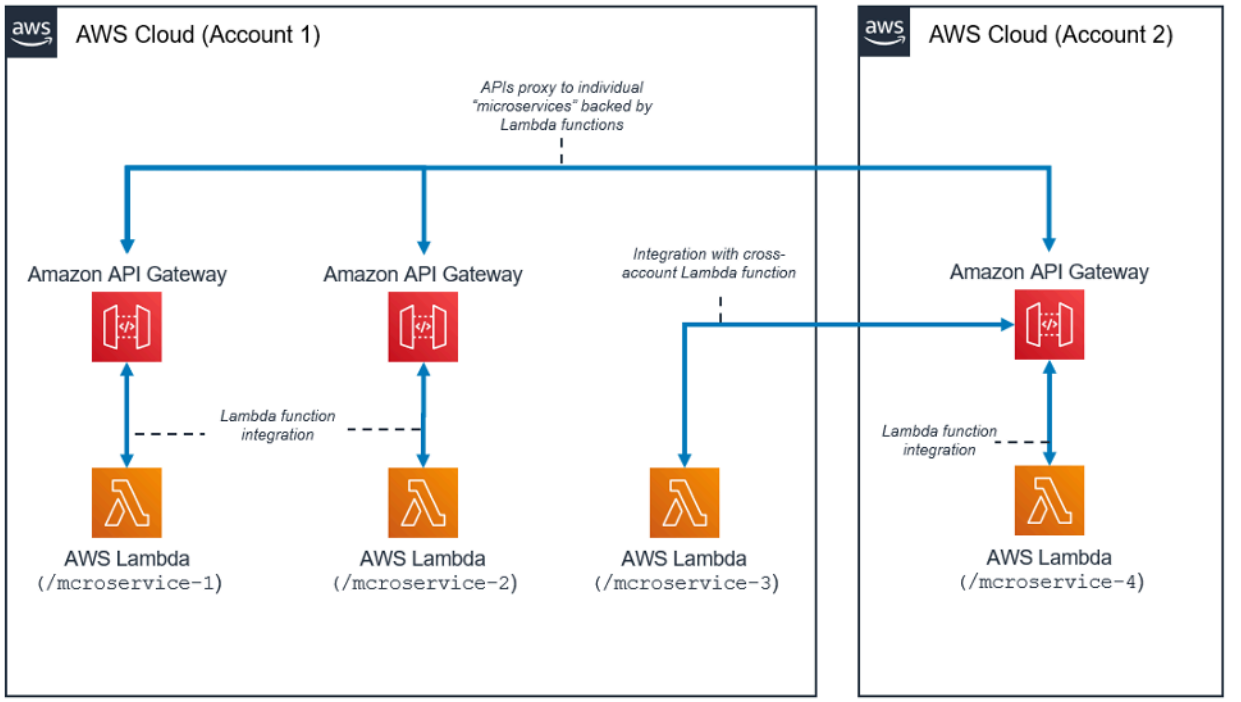


Web 應用程式的架構模式

表 3 - Web 應用程式元件

層	元件
簡報	前端應用程式是由 create-react-app 等 React 公用程式所產生的所有靜態內容 (HTML、CSS、JavaScript 和映像）。Amazon CloudFront 託管所有這些物件。使用 Web 應用程式時，會將所有資源下載到瀏覽器，並從該處開始執行。Web 應用程式會連線至呼叫 APIs後端。
Logic (邏輯)	邏輯層是使用 API Gateway REST APIs。 此架構會顯示多個公開的服務。有多個不同的 Lambda 函數，每個函數都會處理應用程式的不同層面。Lambda 函數位於 API Gateway 後方，可使用 API URL 路徑存取。 使用者身分驗證是使用 Amazon Cognito 使用者集區或聯合身分使用者提供者處理。API Gateway 使用與 Amazon Cognito 的立即可用整合。只有在使用者經過身分驗證後，用戶端才會收到 JSON Web Token (JWT) 權杖，然後在進行 API 呼叫時應使用該權杖。 每個 Lambda 函數都會指派自己的 IAM 角色，以提供適當資料來源的存取權。
資料	在此特定範例中，DynamoDB 用於資料儲存，但其他專用 Amazon 資料庫或儲存服務可根據使用案例和使用案例使用。

使用 Lambda 的微服務



使用 Lambda 的微服務架構模式

微型服務架構模式不受限於典型的三層架構；不過，這種熱門模式可以透過使用無伺服器資源來實現顯著的好處。

在此架構中，每個應用程式元件都會解耦並獨立部署和操作。使用 Amazon API Gateway 建立的 API，以及後續由 啟動的函數 AWS Lambda，就是您建置微服務所需的一切。您的團隊可以使用這些服務，將您的環境解耦和分段至所需的精細程度。

一般而言，微服務環境可能會帶來下列困難：建立每個新微服務時的重複額外負荷、伺服器密度和使用率最佳化的問題、同時執行多個微服務版本的複雜性，以及與許多個別服務整合的用戶端程式碼需求擴散。

當您使用無伺服器資源建立微服務時，這些問題變得較不難解決，在某些情況下，只是消失。無伺服器微服務模式可降低建立每個後續微服務的障礙 (API Gateway 甚至允許複製現有 APIs，以及在其他帳戶中使用 Lambda 函數）。最佳化伺服器使用率不再與此模式相關。最後，Amazon API Gateway 以多種熱門語言提供以程式設計方式產生的用戶端 SDKs，以減少整合開銷。

結論

多層架構模式鼓勵最佳實務，建立易於維護、解耦和擴展的應用程式元件。當您建立由 API Gateway 進行整合並在其中進行運算的邏輯層時 AWS Lambda，您會實現這些目標，同時減少實現這些目標的精力。這些服務共同為您的用戶端提供 HTTPS API 前端和安全環境，以套用您的商業邏輯，同時消除管理典型伺服器型基礎設施所涉及的額外負荷。

貢獻者

本文件的貢獻者包括：

- Andrew Baird，AWS 解決方案架構師
- Bryant Bost，AWS ProServe 顧問
- Stefano Buliani，AWS Mobile 技術資深產品經理
- Vyom Nagrani，AWS Mobile 資深產品經理
- AWS Mobile 資深產品經理 Ajay Nair
- Rahul Popat，全球解決方案架構師
- Brajendra Singh，資深解決方案架構師

深入閱讀

如需其他資訊，請參閱：

- AWS 白皮書和指南

文件修訂

若要收到此白皮書更新的通知，請訂閱 RSS 摘要。

變更	描述	日期
次要更新	錯誤修正和許多小幅變更。	2022 年 4 月 1 日
白皮書已更新	針對新的服務功能和模式進行更新。	2021 年 10 月 20 日
白皮書已更新	針對新的服務功能和模式進行更新。	2021 年 6 月 1 日
白皮書已更新	針對新服務功能更新 。	2019 年 9 月 25 日
初次出版	白皮書已發佈。	2015 年 11 月 1 日

注意

客戶有責任對本文件中的資訊進行自己的獨立評定。本文件：(a) 僅供參考，(b) 代表目前的 AWS 產品產品和實務，如有變更，恕不另行通知，且 (c) 不會從 AWS 及其附屬公司、供應商或授權方建立任何承諾或保證。AWS 產品或服務以「原樣」提供，不提供任何類型的保證、表示或條件，無論明示或暗示。AWS 對其客戶的責任與義務應由 AWS 協議管轄，本文並非 AWS 與其客戶之間的任何協議的一部分，也並非上述協議的修改。