AWS 白皮書

使用 Amazon API Gateway 和 AWS Lambda 的 AWS 無伺服器多層架構

使用 Amazon API Gateway 和 AWS Lambda 的 AWS 無伺服器多層架構：AWS 白皮書

Table of Contents

使用 Amazon API Gateway 和 AWS Lambda 的 AWS 無伺服器多層架構

出版日期：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，改變您設計多層架構和實作熱門模式 (例如微型服務、行動後端和單頁應用程式) 的方式。架構師和開發人員可以使用 Amazon API Gateway、AWS Lambda 和其他服務來減少建立和管理多層應用程式所需的開發和作業週期。

簡介

多層應用程式 (三層、n 層等) 幾十年來持續是基礎架構模式，並且仍然是面向使用者的應用程式的熱門模式。雖然用於描述多層架構的語言各不相同，但多層應用程式通常由以下元件組成：

- 表示層：使用者直接與其互動的元件 (例如，網頁和行動應用程式使用者介面)。
- 邏輯層：將使用者動作轉換為應用程式功能 (例如，CRUD 資料庫作業和資料處理) 所需的程式碼。
- 資料層：儲存與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個通用架構，以確保解偶和獨立可擴展的應用程式元件可以單獨開發、管理和維護 (通常由不同的團隊)。

由於此模式，網路 (一層必須進行網路呼叫才能與另一層互動) 充當層之間的邊界，開發多層應用程式往往需要建立許多無差別的應用程式元件。其中一些元件包括：

- 定義層之間通訊訊息佇列的程式碼
- 定義應用程式介面 (API) 和資料模型的程式碼
- 可確保對應用程式進行適當存取的安全性相關程式碼

所有這些範例都可以被視為「樣板」元件，這些元件雖然在多層應用程式中是必要的，但其實作方式在不同的應用程式之間沒有很大差異。

AWS 提供許多服務，使得您可以建立無伺服器多層應用程式，大幅簡化將此類應用程式部署到生產環境的程序，並去除與傳統伺服器管理相關的負擔。Amazon API Gateway 是一項用於建立和管理 API 的服務，而 AWS Lambda 則是用於執行任意程式碼功能的服務，這兩者可以搭配使用，以簡化穩健的多層應用程式的建立。

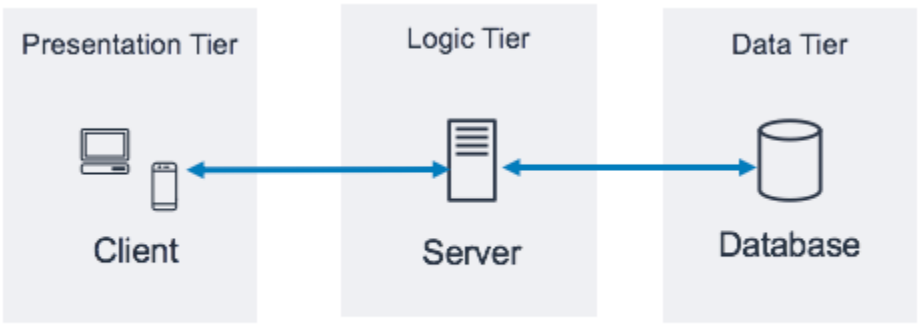
Amazon API Gateway 與 AWS Lambda 的整合，使得使用者定義的程式碼功能可以透過 HTTPS 請求直接啟動。無論請求量如何，API Gateway 和 Lambda 都會自動擴展，以支援您的應用程式的確切需求 (如需可擴展性資訊，請參閱 API Gateway Amazon API Gateway 配額和重要說明)。透過結合這兩項服務，您可以建立一個層，使您只能編寫對應用程式重要的程式碼，而不是著重於實作多層架構的其他各種不同無差別的面向，例如，針對高可用性而架構、編寫用戶端 SDK、伺服器和作業系統 (OS) 管理、擴展和實作用戶端授權機制。

API Gateway 和 Lambda 可實現無伺服器邏輯層的建立。根據您的應用程式需求，AWS 還提供選項用於建立無伺服器表示層 (例如，使用 Amazon CloudFront 和 Amazon Simple Storage Service) 和資料層 (例如，Amazon Aurora、Amazon DynamoDB)。

本白皮書聚焦於最熱門的多層架構範例，即三層 Web 應用程式。但是，您可以將此多層模式運用於典型的三層 Web 應用程式之外。

三層架構概觀

三層架構是多層架構中最熱門的實作，由單一表示層、一個邏輯層和一個資料層組成。下圖顯示一個簡單的通用三層應用程式範例。



三層應用程式的架構模式

有許多絕佳的線上資源，您可以在其中瞭解關於一般三層架構模式的相關資訊。本白皮書聚焦於使用 Amazon API Gateway 和 AWS Lambda 針對此架構的特定實作模式。

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 的位置，而相較於傳統基於伺服器的實作，AWS Lambda 可以有最大的影響。這兩項服務的功能可讓您建置高度可用、可擴展且安全的無伺服器應用程式。在傳統模式中，您的應用程式可能需要數千部伺服器；但是，透過使用 Amazon API Gateway 和 AWS Lambda，您不需負責任何容量伺服器的管理。此外，透過結合使用這些受管服務，您可以獲得以下優點：

- AWS Lambda：
  - 無需選擇、保護、修補或管理作業系統
  - 無需選擇適當大小的伺服器、監視或擴展
  - 降低過度佈建帶來的成本風險
  - 降低佈建不足帶來的效能風險

- Amazon API Gateway：
  - 簡化部署、監控和保護 API 的機制
  - 透過快取和內容交付提高 API 效能

AWS Lambda

AWS Lambda 是一項運算服務，可讓您以任何支援的語言 (Node.js、Python、Ruby、Java、Go、.NET，如需詳細資訊，請參閱 Lambda 常見問答集) 執行任意程式碼函數，而無需佈建、管理或擴展伺服器。Lambda 函數會在受管的隔離容器中執行，並且會為了回應事件而啟動，其可以是 AWS 讓其可用的數個以程式設計的觸發程序之一，稱為事件來源。 如需所有事件來源，請參閱 Lambda 常見問答集。

Lambda 的許多熱門使用案例都是有關事件驅動的資料處理工作流程，例如，處理儲存在 Amazon S3 中的檔案或從 Amazon Kinesis 的串流資料記錄。與 Amazon API Gateway 結合使用時，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 可充當邏輯層的前門，並且 AWS Lambda 會叫用應用程式碼。

您的商業邏輯就在這裡，無需伺服器

Lambda 要求您編寫程式碼函數，稱為處理常式，其將在由事件啟動時執行。若要使用 Lambda 搭配 API Gateway，您可以將 API Gateway 設定為在發生對 API 的 HTTPS 請求時啟動處理常式函數。在

無伺服器多層架構中，您在 API Gateway 中建立的每個 API，都將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 函數來組成邏輯層，可讓您定義公開應用程式功能所需的精細程度 (每個 API 一個 Lambda 函數或每個 API 方法一個 Lambda 函數)。在 Lambda 函數內，處理常式可以向外連接任何其他相依性 (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)，或甚至其他 Lambda 函數。

建立或更新 Lambda 函數需要將程式碼以 Lambda 部署套件的形式放在一個 zip 檔案中上傳到 Amazon S3 儲存貯體，或將程式碼與所有相依性封裝為容器映像。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) 或執行基礎設施即程式碼範本或架構，例如，AWS CloudFormation、AWS Serverless Application Model (AWS SAM) 或 AWS Cloud Development Kit (AWS CDK)。使用上述任何方法建立函數時，您可以指定要將部署套件內的哪個方法充當請求處理常式。您可以為多個 Lambda 函數定義重複使用相同的部署套件，其中的每個 Lambda 函數可能在相同部署套件內具有唯一的處理常式。

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該函數。使用 IAM 政策，可以建立除非由您定義的 API Gateway 資源叫用，否則無法啟動的 Lambda 函數。此類政策可以在各種 AWS 服務中使用基於資源的政策來定義。

每個 Lambda 函數都會擔任一個 IAM 角色，在部署 Lambda 函數時指派。此 IAM 角色會定義您的 Lambda 函數可以與其互動的其他 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。

請勿將敏感資訊儲存在 Lambda 函數中。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 函數內部存取其他憑證 (例如，資料庫憑證和 API 金鑰)，則可以使用 AWS Key Management Service (AWS KMS) 搭配環境變數，或者使用服務 (例如 AWS Secrets Manager)，以確保此資訊在未使用時的安全。

大幅提高效能

以容器映像形式從 Amazon Elastic Container Registry (Amazon ECR) (Amazon ECR) 或從上傳到 Amazon S3 的 zip 檔案中提取的程式碼，會在由 AWS 管理的隔離環境中執行。您不需擴展 Lambda 函數 - 每次函數收到事件通知，AWS Lambda 都會在其計算機群內找到可用容量，並使用您定義的執行時間、記憶體、磁碟和逾時組態來執行程式碼。利用此模式，AWS 可以視需要啟動任意數量的函數副本。

基於 Lambda 邏輯層的大小始終會迎合您的客戶需求。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 按使用付費定價，讓您始終能夠滿足客戶的請求，同時無需為閒置的運算容量付費。

無伺服器部署和管理

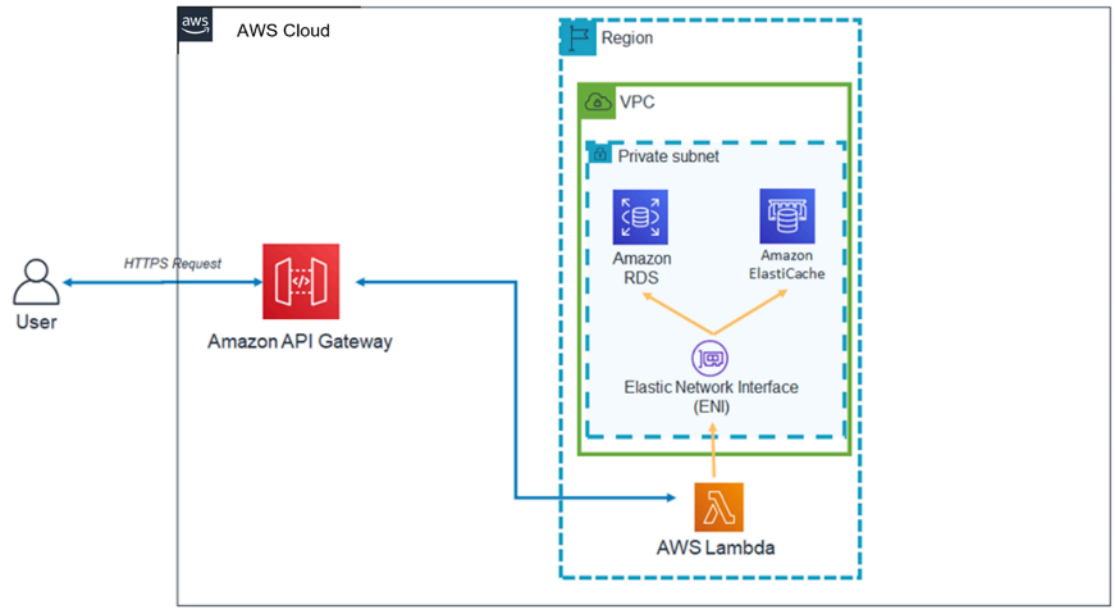
為協助您部署和管理 Lambda 函數，請使用 AWS SAM AWS Serverless Application Model (AWS SAM)，這是一項開放原始碼架構，其中包括：

- AWS SAM 範本規格 - 用於定義您的函數並描述其環境、許可、組態和事件以簡化上傳和部署的語法。
- 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。

您還可以使用 AWS CDK，其為軟體開發架構，用於使用程式設計語言定義雲端基礎設施，並透過 CloudFormation 佈建它。CDK 提供定義 AWS 資源的指令式方式，而 AWS SAM 則提供宣告方式。

通常，部署 Lambda 函數時，它會使用其獲指派的 IAM 角色定義的許可來叫用該函數，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的商業或使用者資訊，請務必確保此資料層被適當隔離 (在私有子網路中)。

如果希望 Lambda 函數存取無法對外公開的資源 (如私有資料庫執行個體)，則可以將 Lambda 函數設定為連接到 AWS 帳戶的 Virtual Private Cloud (VPC) 中的私有子網路。將函數連接到 VPC 時，Λ 會為函數 VPC 組態中的每個子網路建立一個彈性網路介面，而彈性網路介面可用於私密存取您的內部資源。



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可能無法透過網際網路存取。VPC 還可確保從網際網路上與您的資料進行互動的唯一方法是透過您定義的 API 和您編寫的 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的 API。

用戶端 (即表示層) 使用標準 HTTPS 請求與透過 API Gateway 公開的 API 整合。透過 API Gateway 公開的 API 對服務導向的多層架構的適用性，是能夠隔離應用程式功能的個別部分，並透過 REST 端點公開此功能。Amazon API Gateway 具有特定的功能和特色，可為您的邏輯層新增強大的功能。

與 AWS Lambda 整合

Amazon API Gateway 同時支援 REST 和 HTTP 類型的 API。API Gateway API 是由資源和方法構成。資源是應用程式可透過資源路徑存取的邏輯實體 (例如，`/tickets`)。方法會與提交給 API 資源的 API 請求對應 (例如，`GET /tickets`)。API Gateway 可讓您使用 Lambda 函數支援每個方法，也就是說，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連接 API Gateway 和 Lambda 函數。

代理整合

在代理整合中，整個用戶端 HTTPS 請求會依原樣傳送到 Lambda 函數。API Gateway 會將整個用戶端請求作為 Lambda 處理常式函數的事件參數傳遞，而 Lambda 函數的輸出會直接傳回到用戶端 (包括狀態代碼、標頭等)。

非代理整合

在非代理整合中，您可以設定如何將用戶端請求的參數、標頭和本文傳遞到 Lambda 處理常式函數的事件參數。此外，您還可以設定將 Lambda 輸出轉譯回使用者的方式。

> **Note**
>
> API Gateway 還可以代理到 AWS Lambda 外部的其他無伺服器資源，例如，模擬整合 (對初始應用程式開發很實用)，以及直接代理到 S3 物件。

跨區域的穩定 API 效能

Amazon API Gateway 的每一個部署本質上都包括一個 Amazon CloudFront 分發。CloudFront 是一項內容交付服務，它使用 Amazon 的全球節點網路作為使用 API 的用戶端的連接點。這有助於減少 API 的回應延遲。透過使用全球的多個節點，Amazon CloudFront 還提供抵禦分散式阻斷服務 (DDoS) 攻擊案例的功能。如需詳細資訊，請檢閱 AWS DDoS 彈性最佳實務白皮書。

透過使用 API Gateway 將回應儲存在選用的記憶體內快取中，您可以提高特定 API 請求的效能。此方法不僅可為重複的 API 請求提供效能優點，還可以減少叫用 Lambda 函數的次數，從而降低您的整體成本。

透過內建功能鼓勵創新並減少負擔

建置任何新應用程式的開發成本都是一項投資。使用 API Gateway 可以減少特定開發工作所需的時間，並降低總開發成本，使得組織能夠更自由地試驗和創新。

在初始應用程式開發階段期間，通常會忽略記錄和指標收集的實作，以便更快地交付新應用程式。在將這些功能部署到生產中執行的應用程式時，這可能會導致技術負債和作業風險。Amazon API Gateway 與 Amazon CloudWatch 無縫整合，能夠從 API Gateway 收集原始資料並處理為可讀取、近乎即時的指標，以監控 API 執行情況。API Gateway 還支援具有可設定報告的存取記錄，以及用於偵錯的 AWS X-Ray 追蹤。這些功能的每一項都不需要編寫程式碼，並且可以於生產中執行的應用程式中進行調整，而不會對核心商業邏輯造成風險。

應用程式的整體生命週期可能未知，或可能已知會是短暫。如果您的起點已經包含 API Gateway 提供的受管功能，並且您僅在 API 開始接收請求後才會產生基礎設施成本，那麼為建置此類應用程式建立商業案例會更加簡單。如需詳細資訊，請參閱 Amazon API Gateway 定價。

快速反覆，保持敏捷性

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

預備部署

在 API Gateway 中部署 API 時，必須將該部署與 API Gateway 階段相關聯，每個階段都是 API 的快照，並且可供用戶端應用程式呼叫。使用此慣例，您可以輕鬆地將應用程式部署到開發、測試、預備或生產階段，並在各個階段之間移動部署。每次將 API 部署到某個階段時，都會建立不同版本的 API，必要時可以還原該版本。這些功能使得現有功能和用戶端相依性可不受干擾地繼續，同時將新功能以單獨的 API 版本形式發佈。

與 Lambda 的解偶整合

API Gateway 中的 API 和 Lambda 函數之間的整合可以使用 API Gateway 階段變數和 Lambda 函數別名來解偶。這可簡化並加速 API 部署的速度。不要直接在 API 中設定 Lambda 函數名稱或別名，而是可以在 API 中設定預備變數，其可以指向 Lambda 函數中的特定別名。部署期間，將預備變數值變更為指向 Lambda 函數別名，而 API 將針對特定階段於 Lambda 別名後面執行該 Lambda 函數版本。

Canary 版本部署

Canary 版本是一種軟體開發策略，此策略會部署 API 的新版本以供測試之用，而基本版本仍會部署為生產版本以供相同階段正常作業之用。在 Canary 版本部署中，總 API 流量會隨機分為生產版本以及具有預先設定比率的 Canary 版本。可以設定 API Gateway 中的 API，讓 Canary 版本部署得以使用一組有限的使用者測試新功能。

自訂網域名稱

您可以為 API 提供直覺的商業易記 URL 名稱，而非 API Gateway 提供的 URL。API Gateway 提供為 API 設定自訂網域的功能。使用自訂網域名稱，您就能設定 API 的主機名稱，並選擇多層級基本路徑 (例如，`myservice`、`myservice/cat/v1` 或 `myservice/dog/v2`) 將替代 URL 對應至您的 API。

API 安全性優先

所有應用程式都必須確保只有獲授權的用戶端才能存取其 API 資源。在設計多層應用程式時，您可以利用 Amazon API Gateway 幫助保護您的邏輯層的數個不同方式：

傳輸安全

對 API 的所有請求都可以透過 HTTPS 提出，以啟用傳輸中加密。

API Gateway 提供內建的 SSL/TLS 憑證，如果對面向公眾的 API 使用自訂網域名稱選項，您可以使用 AWS Certificate Manager 來提供自己的 SSL/TLS 憑證。API Gateway 還支援交互 TLS (mTLS) 身分驗證。交互 TLS 可增強 API 的安全性，並有助於保護您的資料免受用戶端詐騙或中間人攻擊等攻擊。

API 授權

您在 API 中建立的每個資源/方法組合都會被授與唯一的 Amazon 資源名稱 (ARN)，該名稱 (ARN) 可在 AWS Identity and Access Management (IAM) 政策中參考。

要新增授權至 API Gateway 中的 API，有三個一般方法：

- IAM 角色和政策：用戶端會對 API 存取使用 AWS 簽章第 4 版 (SigV4) 授權和 IAM 政策。相同憑證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字符，其包含在請求的授權標頭中。
- Lambda 授權者：定義一個 Lambda 函數，其會實作使用承載字符策略 (例如，OAuth 和 SAML) 的自訂授權方案，或使用請求參數來識別使用者。

存取限制

API Gateway 支援產生 API 金鑰，並將這些金鑰與可設定的用量計劃相關聯。您可以使用 CloudWatch 監控 API 金鑰的使用情況。

API Gateway 對 API 中的每一方法支援調節、速率限制和爆量率限制。

私有 API

使用 API Gateway，您可以透過介面 VPC 端點建立只能從 Amazon VPC 中的虛擬私有雲端存取的私有 REST API。這是您在 VPC 中建立的端點網路介面。

您可利用資源政策啟用或拒絕選定的 VPC 與 VPC 端點 (包括其他 AWS 帳戶) 存取 API。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

面向網際網路的 API 容易受到惡意攻擊。AWS WAF 是一個 Web 應用程式防火牆，有助於保護 API 免受此類攻擊。它可以保護 API 免受常見的網頁入侵，別如 SQL 插入和跨網站指令碼攻擊。您可以使用 AWS WAF 搭配 API Gateway，以幫助保護 API。

資料層

使用 AWS Lambda 作為邏輯層不會限制資料層中可用的資料儲存選項。Lambda 函數會透過在 Lambda 部署套件中包含適當的資料庫驅動程式，來連接到任何資料儲存選項，並使用以 IAM 角色為基礎的存取或加密憑證 (透過 AWS KMS 或 AWS Secrets Manager)。

為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 提供許多無伺服器和非無伺服器資料存放區，您可以使用這些存放區來組成應用程式的資料層。

無伺服器資料儲存選項

Amazon S3 是一項物件儲存服務，提供領先業界的可擴展性、資料可用性、安全性和效能。

Amazon Aurora 是專為雲端建置的 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與經濟實惠優勢。Aurora 同時提供無伺服器和傳統使用模式。

Amazon DynamoDB 是一個鍵值和文件資料庫，可針對任何規模提供十毫秒內級別的效能。它是一個全受管、無伺服器、多區域、多主機、耐用的資料庫，針對網際網路規模的應用程式提供內建安全性、備份和還原以及記憶體內快取。

Amazon Timestream 是一個適用於 IoT 和營運應用程式的快速、可擴展、全受管的時間序列資料庫服務，可輕鬆存放和分析每天數兆個事件，其成本僅為關聯式資料庫的十分之一。在 IoT 裝置、IT 系統和智慧型工業機器的推動下，時間序列資料 (衡量事物如何隨時間變化的資料) 是增長最快的資料類型之一。

Amazon Quantum Ledger Database (Amazon QLDB) 是全受管總帳資料庫，可針對集中信任單位所有的交易日誌提供透明、不可變及採用密碼演算法的可驗證交易日誌。Amazon QLDB 追蹤每個及所有應用程式資料變更，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) 是一種具可擴展性、高可用性且受管的 Apache Cassandra 相容資料庫服務。透過 Amazon Keyspaces，您可以在 AWS 上使用與今天相同的 Cassandra 應用程式程式碼和開發人員工具執行 Cassandra 工作負載。您無需佈建、修補或管理伺服器，而且您不必安裝、維護或操作軟體。Amazon Keyspaces 是無伺服器服務，因此，您只需對所用的資源付費，該服務可自動根據應用程式流量擴展和縮減表格。

Amazon Elastic File System (Amazon EFS) 提供簡單、無伺服器、一勞永逸的彈性檔案系統，讓您可以共享檔案資料，而無須佈建或管理儲存體。它可以與 AWS 雲端服務和內部部署資源搭配使用，並且

可以在不中斷應用程式的情況下隨需擴展至 PB 級。使用 Amazon EFS，您可以在新增和移除檔案時自動增長和縮減檔案系統，而無須佈建和管理容量來適應增長。Amazon EFS 可以使用 Lambda 函數掛載，這使其成為 API 的可行檔案儲存選項。

非無伺服器資料儲存選項

Amazon Relational Database Service (Amazon RDS) 是一項受管 Web 服務，它讓您更容易設定、操作和擴展關聯式資料庫，方法是使用任何可用引擎 (Amazon Aurora、PostgreSQL、MySQL、MariaDB、Oracle 和 Microsoft SQL Server)，並在針對記憶體、效能或 I/O 最佳化的數個不同資料庫執行個體類型上執行。

Amazon Redshift 是一項雲端中全受管的 PB 級資料倉儲服務。

Amazon ElastiCache 是 Redis 或 Memcached 一項完全受管的部署。無縫部署、執行和擴展熱門的開放原始碼相容記憶體內資料儲存。

Amazon Neptune 是快速、可靠的全受管圖形資料庫服務，可讓您輕鬆建置及執行搭配高度連線資料集運作的應用程式。Neptune 支援熱門的圖形模型 (屬性圖形和 W3C 資源描述架構 (RDF)) 及其各自的查詢語言，使您能夠輕鬆建置有效導覽高度連線資料集的查詢。

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

最後，您還可以使用在 Amazon EC2 上獨立執行的資料存放區作為多層應用程式的資料層

表示層

表示層負責與邏輯層互動，經由透過網際網路公開的 API Gateway REST 端點。任何支援 HTTPS 的用戶端或裝置都可以與這些端點通訊，使得您的表示層有靈活性可採取多種形式 (桌面應用程式、行動應用程式、網頁、IoT 裝置等)。根據您的要求，您的表示層可以使用以下 AWS 無伺服器產品：任何支援 HTTPS 的用戶端或裝置都可以與這些端點通訊，使得您的表示層有靈活性可採取多種形式 (桌面應用程式、行動應用程式、網頁、IoT 裝置等)。根據您的要求，您的表示層可以使用以下 AWS 無伺服器產品：

- Amazon Cognito - 無伺服器使用者身分和資料同步服務，使您能夠快速且有效地對您的 Web 和行動應用程式進行新增使用者註冊、登入和存取控制。Amazon Cognito 可擴展到數百萬使用者，並支援透過 SAML 2.0 使用社交身分供應商 (例如 Facebook、Google 和 Amazon) 以及企業身分供應商進行登入。
- Amazon S3 與 CloudFront - 讓您能夠直接從 S3 儲存貯體為靜態網站 (如單頁應用程式) 提供服務，而無需佈建 Web 伺服器。您可以將 CloudFront 用作受管內容交付網路 (CDN)，以提高效能並使用受管或自訂憑證啟用 SSL/TL。

AWS Amplify 是一套可以結合使用或單獨使用的工具和服務，用於協助前端 Web 和行動開發人員採用 AWS 技術建置可擴展的完整堆疊應用程式。Amplify 提供用於在全球範圍內部署和託管靜態 Web 應用程式的全受管服務，透過 Amazon CDN 全球範圍內的數百個連接點為這些應用程式提供服務，以及透過內建的 CI/CD 工作流程來加快應用程式的發佈週期。Amplify 支援熱門的 Web 架構，包括 JavaScript、React、Angular、Vue、Next.js，同時支援各種行動平台，包括 Android、iOS、React Native、Ionic 和 Flutter。根據您的聯網組態和應用程式需求，您可能需要讓 API Gateway API 符合跨來源資源共享 (CORS)。CORS 合規會允許 Web 瀏覽器直接從靜態網頁內叫用您的 API。

使用 CloudFront 部署網站時，會為您提供一個 CloudFront 網域名稱，用來連接您的應用程式 (例如，`d2d47p2vcczkh2.cloudfront.net`)。您可以使用 Amazon Route 53 來註冊網域名稱，並將其導向到您的 CloudFront 分發，或將已擁有的網域名稱導向到您的 CloudFront 分發。這使得使用者可以使用熟悉的網域名稱存取您的網站。請注意，您還可以使用 Route 53 將自訂網域名稱指派給 API Gateway 分發，這會使得使用者能夠使用熟悉的網域名稱叫用 API。

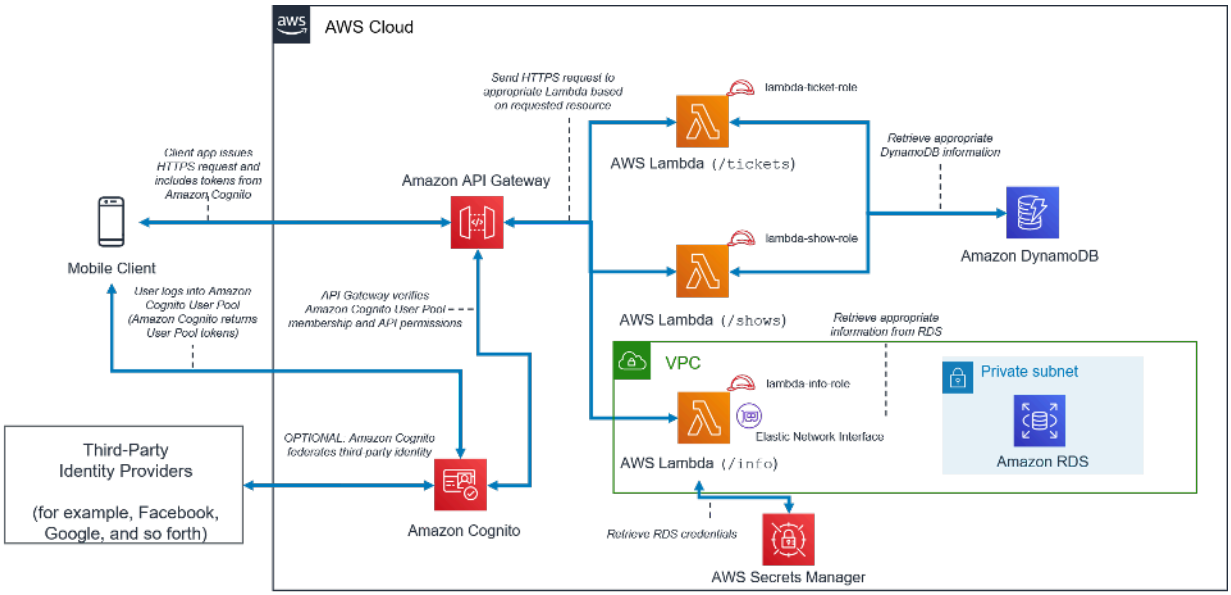
範例架構模式

您可以使用 API Gateway 和 AWS Lambda 做為邏輯層來實作熱門的架構模式。本白皮書包括利用基於 AWS Lambda 邏輯層的最熱門架構模式：

- 行動後端 - 一種行動應用程式，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 資源來託管表示層資源的通用 HTTPS 用戶端 (例如，桌面用戶端、EC2 上執行的 Web 伺服器等)。
- 單頁應用程式 - 託管在 Amazon S3 和 CloudFront 中的單頁應用程式，會與 API Gateway 和 AWS Lambda 通訊，以存取應用程式資料。
- 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，其使用 AWS Lambda 搭配 API Gateway 以達成其商業邏輯。它還將 DynamoDB 用作資料庫，並將 Amazon Cognito 用於使用者管理。所有靜態內容都使用 Amplify 託管。

除了這兩種模式之外，本白皮書還會討論 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的模式，雖然不是標準的三層架構，但會涉及解耦應用程式元件和將其部署為可彼此通訊的無狀態個別功能單位。

行動後端

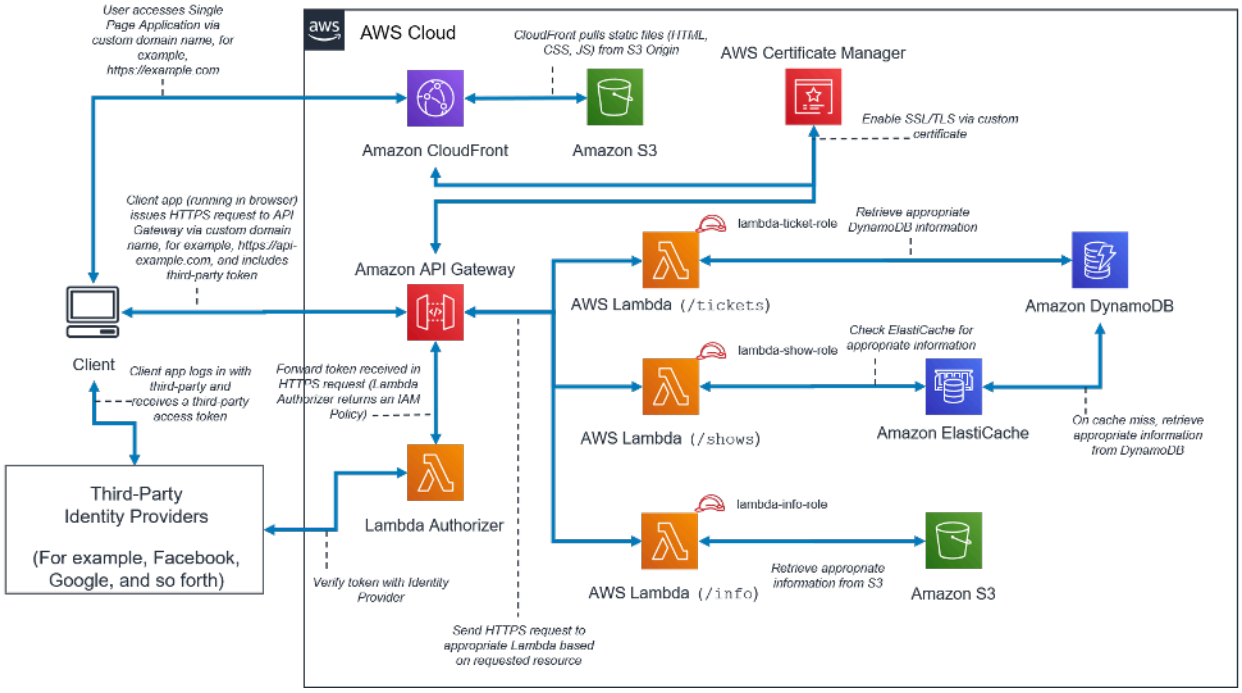


無伺服器行動後端的架構模式

表 1 - 行動後端層元件

層	元件
表示	在使用者裝置上執行的行動應用程式。
邏輯	Amazon API Gateway 與 AWS Lambda。 此架構顯示三項公開的服務 (`/tickets`、`/shows` 和 `/info`)。API Gateway 端點會由 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 (必要時使用聯合身分第三方)，並獲得用於授權 API Gateway 呼叫的存取權和 ID 字符。 每個 Lambda 函數都會獲指派其 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 會用於 `/info` 服務。此 Lambda 函數會從 AWS Secrets Manager 擷取 Amazon RDS 憑證，並使用彈性網路介面以存取私有子網路。

單頁應用程式



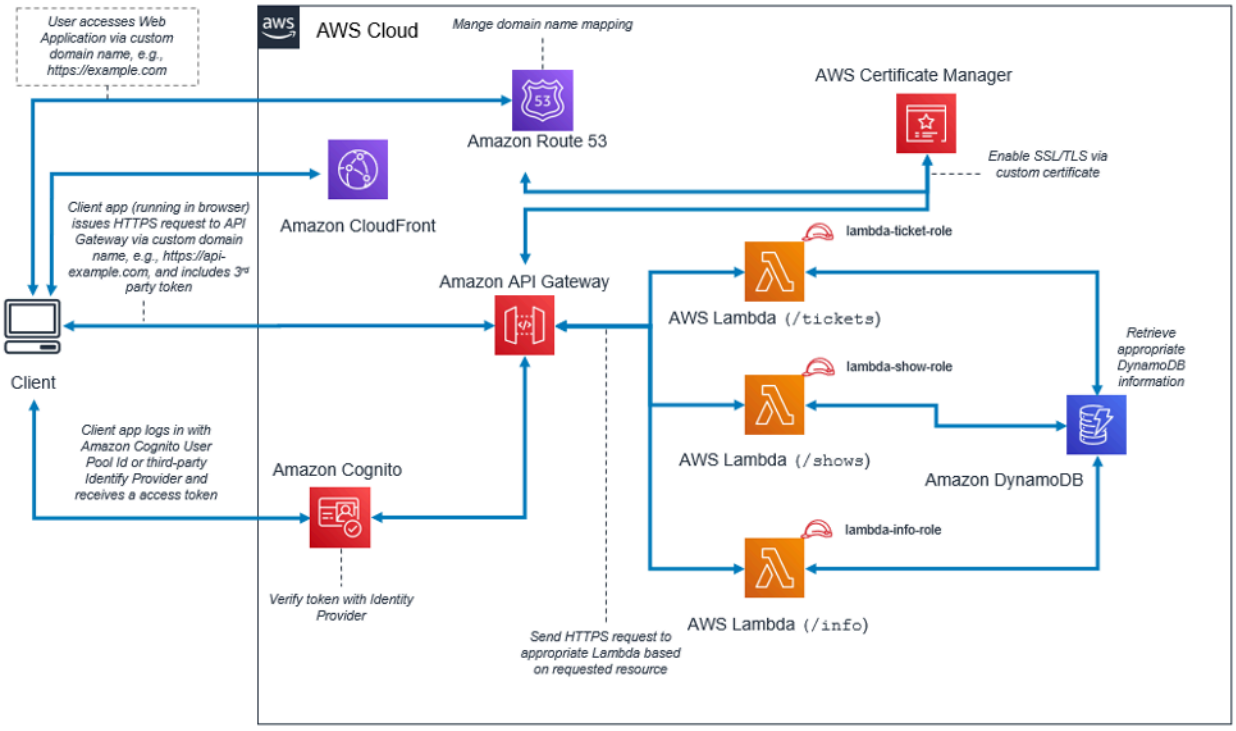
無伺服器單頁應用程式的架構模式

表 2 - 單頁應用程式元件

層	元件
表示	在 Amazon S3 中託管的靜態網站內容，由 CloudFront 分配。 AWS Certificate Manager 允許使用自訂 SSL/TLS 憑證。
邏輯	API Gateway 與 AWS Lambda。 此架構顯示三項公開的服務 (`/tickets`、`/shows` 和 `/info`)。API Gateway 端點會由 Lambda 授權方保護。在此方法中，使用者會透過第三方身分提供者登入，並取得存取權和 ID 字符。這些字符會包含在 API Gateway 呼叫中，而 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 函數都會獲指派其 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` 服務使用 Amazon ElastiCache 來提高資料庫效能。快取遺漏數會傳送到 DynamoDB。 Amazon S3 會用來託管 `/info service` 使用的靜態內容。

Web 應用程式

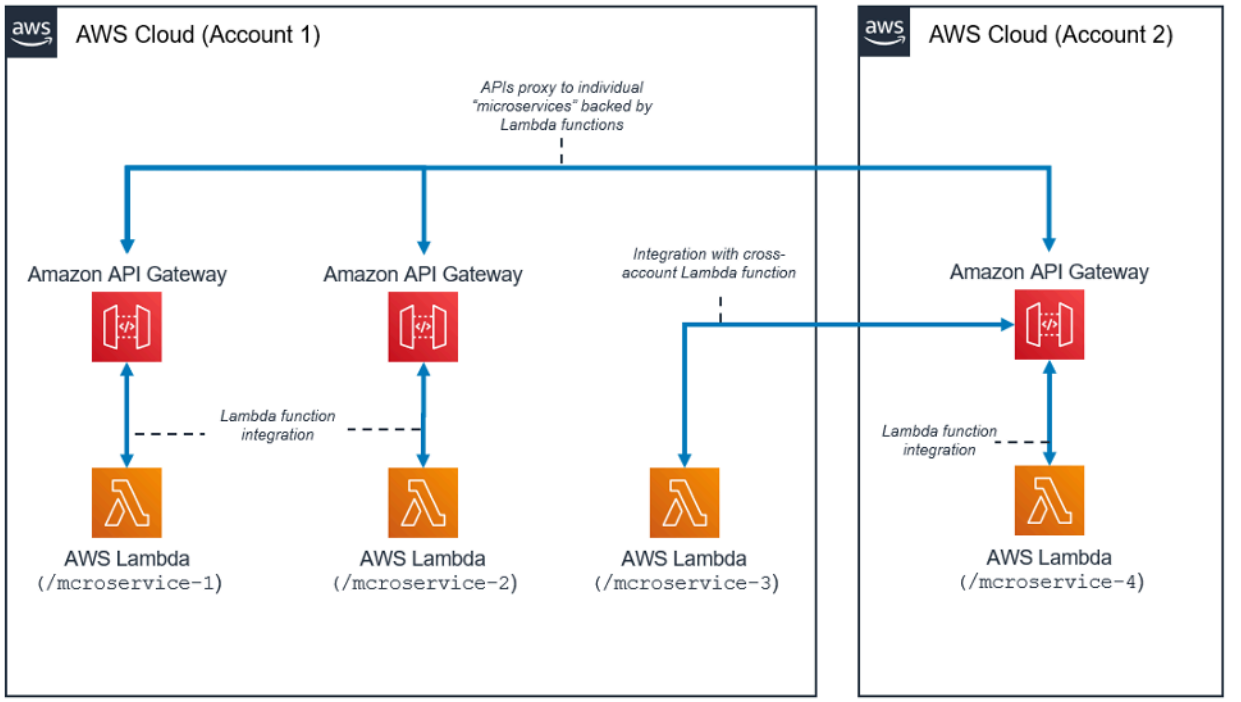


Web 應用程式的架構模式

表 3 - Web 應用程式元件

層	元件
表示	前端應用程式全部為靜態內容 (HTML、CSS、JavaScript 和影像)，其由 create-react-app 之類的 React 公用程式產生。Amazon CloudFront 會託管這所有物件。Web 應用程式在使用時，會將所有資源下載到瀏覽器，並從瀏覽器開始執行。Web 應用程式會連線到呼叫 API 的後端。
邏輯	邏輯層是使用 Lambda 函數建置，前端為 API Gateway REST API。 此架構顯示多項公開的服務。有多個不同的 Lambda 函數，每個函數都處理應用程式的不同面向。Lambda 函數位於 API Gateway 後方，並且可使用 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來處理。API Gateway 使用與 Amazon Cognito 的立即可用的整合。只有在使用者通過身分驗證後，用戶端才會收到 JSON Web 字符 (JWT) 字符，然後在進行 API 呼叫時應使用該字符。 每個 Lambda 函數都會獲指派其 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 會用於資料存放區，但根據使用案例和使用情形，也可以使用其他專用的 Amazon 資料庫或儲存服務。

微服務與 Lambda



使用 Lambda 的微服務的架構模式

微服務架構模式並不侷限於典型的三層架構。但是，此熱門模式可以從使用無伺服器資源實現顯著的優點。

在此架構中，每個應用程式元件都是解偶並獨立部署和作業。使用 Amazon API Gateway 建立的 API，以及後續由 AWS Lambda 啟動的功能，是要建置微服務您所需的一切。您的團隊可以使用這些服務將您的環境解偶並分段為所需的精細程度。

一般來說，微服務環境會帶來以下困難：建立每個新微服務的重複負擔、最佳化伺服器密度和利用率的問題、同時執行多個微服務的多個版本的複雜性，以及用戶端程式碼與許多不同服務整合的要求激增。

使用無伺服器資源建立微服務時，這些問題會變得不那麼難以解決，而在某些情況下，問題便消失了。無伺服器微服務模式降低了建立每個後續微服務的障礙 (API Gateway 甚至允許複製現有的 API，並在其他帳戶中使用 Lambda 函數)。使用此模式，最佳化伺服器使用率即不再相關。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 SDK，以減少整合負擔。

結論

多層架構模式鼓勵建立易於維護、解偶和擴展的應用程式元件的最佳實務。建立由 API Gateway 整合並在 AWS Lambda 中計算的邏輯層時，您可以實現這些目標，同時減少達成這些目標所需的工作量。這些服務共同為您的用戶端提供 HTTPS API 前端和一個安全的環境來運用商業邏輯，同時去除管理典型基於伺服器的基礎設施所涉及的負擔。

作者群

此文件的作者包括：

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

深入閱讀

如需詳細資訊，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。

update-history-change	update-history-description	update-history-date
白皮書已更新	針對新服務功能和模式更新。	2021 年 10 月 20 日
白皮書已更新	針對新服務功能和模式更新。	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

聲明

客戶應負責對本文件中的資訊自行進行獨立評估。本文件：(a) 僅供參考之用，(b) 代表目前的 AWS 產品供應與實務，如有變更恕不另行通知，以及 (c) 不構成 AWS 及其附屬公司、供應商或授權人的任何承諾或保證。AWS 產品或服務以「現況」提供，不提供任何明示或暗示的擔保、主張或條件。AWS 對其客戶之責任與義務，應受 AWS 協議之約束，且本文件並不屬於 AWS 與其客戶間之任何協議的一部分，亦非上述協議之修改。