AWS 無伺服器多層架構

使用 Amazon API Gateway 和 AWS Lambda

2015 年 11 月

©2015 年，Amazon Web Services, Inc. 或其附屬公司。保留所有權利。

注意

本文件資訊僅供參考。其內容為文件發佈日當時 AWS 的最新產品供應項目及實務方法，如有變更，恕不另行通知。客戶需自行獨立評估本文件資訊，任何 AWS 產品或服務皆以「現狀」提供，不包含任何明示或暗示性保證。本文件不提供任何來自 AWS、其附屬公司、供應商或授權人之任何保證、表示、契約承諾、條件或擔保。AWS 對其客戶的責任與義務由 AWS 協議進行控管，本文件並非 AWS 與其客戶之間的任何協議的一部分，也並非修改上述協議。

目錄

摘要

本白皮書說明 Amazon Web Services (AWS) 所提供的創新技術，如何改變了熱門模式 (例如微服務、行動後端和公有網站) 的多層架構設計方式。現在，架構師和開發人員可以使用 Amazon API Gateway 和 AWS Lambda 等實作模式，來降低建置與操作管理多層應用程式所需的開發工作和操作週期。

簡介

多層應用程式 (三層、n 層等) 數十年來已成為基本的架構模式。多層架構提供可供您遵循的良好準則，可確保能個別管理和維護去耦的可擴展應用程式元件（通常由不同團隊執行）。多層應用程式經常使用服務導向架構 (SOA) 方法建置以使用 Web 服務。在這項方法中，網路成為各層之間的界線然而，為應用程式建置新的 Web 服務層在許多方面並無差別。多層 Web 應用程式內的許多程式碼，是直接源自於模式本身。例如用來整合各層的程式碼、定義 API 和資料模型的程式碼 (用以使各層彼此了解)，以及安全相關的程式碼 (用以防止各層整合點不當暴露)。

Amazon API Gateway[1] 是一套可用於建置和管理 API 的服務；AWS Lambda[2] 則是用來執行任意程式碼功能的服務，這兩項服務可以搭配使用，以便簡化建置強大的多層應用程式時的相關工作。

Amazon API Gateway 與 AWS Lambda 的整合，讓使用者定義的程式碼功能，能夠透過使用者定義的 HTTPS 要求，來直接觸發。無論所需的要求數量為何，API Gateway 和 Lambda 都能夠自動擴展規模，以支援您應用程式確切的需求。結合這兩項服務將允許您建立應用程式層，以利撰寫對應用程式具有重要性的程式碼，*而非*專注於建置多層架構其他各種無分別的面向，例如建立高可用性的架構、撰寫用戶端 SDK、伺服器/作業系統 (OS) 的管理與擴展，以及建立用戶端的授權機制。

最近，AWS 推出了技術，讓您可建置 Lambda 功能，在 Amazon 虛擬私有雲端 (Amazon VPC)[3]內執行。這項功能擴大了結合 API Gateway 與 Lambda 的效益，納入各種需要網路隱私的使用案例。例如，當您需要整合包含敏感資訊的關聯式資料庫的 Web 服務時。Lambda 與 Amazon VPC 的整合，可間接擴展 Amazon API Gateway 的功能，因為開發人員能夠在後端的前方，自行定義一套 HTTPS API (可透過網際網路存取)，這套 API 隸屬於 Amazon VPC 的一部分，且保持隱私與安全性。您可以觀察這項強大的模式，為多層架構的各層所帶來的效益。本白皮書主要說明多層架構最熱門的範例 - **三層** Web 應用程式，不過，您可以將此一多層模式，套用到典型三層 Web 應用程式以外的地方。

三層架構概觀

三層架構是使用者導向應用程式的熱門模式。此架構的各層包括**展示層**、**邏輯層**和**資料層**。展示層代表使用者與其直接互動的元件 (例如網頁、行動應用程式 UI 等)。邏輯層包含的程式碼，可用來將展示層的使用者動作，轉譯為驅動應用程式動作的功能。資料層包含儲存媒體 (資料庫、物件存放區、快取、檔案系統等)，用來儲存與應用程式相關的資料。圖 1 為簡易三層應用程式的範例。

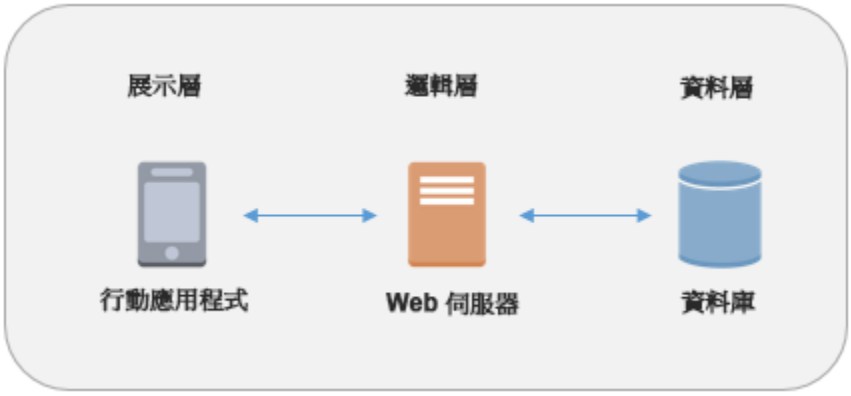

圖 1：簡易三層應用程式的架構模式

有許多絕佳的線上資源可供您學習，以進一步了解*一般*的三層架構 模式。本白皮書主要說明此種架構的特定建置模式 (使用 Amazon API Gateway 和 AWS Lambda)。

無需伺服器的邏輯層

三層架構中的邏輯層是應用程式的大腦。這也是為什麼，結合 Amazon API Gateway 和 AWS Lambda 來建置邏輯層的作法，會如此地具有革命性。這兩項服務的功能允許您建置無需伺服器的 生產應用程式，具備高度可用性、可擴展性和安全性。您的應用程式原本會使用數千台伺服器，但如果採用此項模式，則完全不需要任何伺服器。此外，搭配使用這些受管的服務，可提供下列效益：

不再需要選擇、保護、修補或管理作業系統。

不再需要針對伺服器選擇適當規模、進行監控或擴展。

不再因為過度佈建而產生更多成本風險。

不再因為佈建不足而產生效能風險。

此外，每項服務中還有特定的功能，可為多層架構模式帶來效益。

Amazon API Gateway

Amazon API Gateway 是完全受管的服務，可用來定義、部署和維護 API。用戶端可使用標準 HTTPS 要求來整合 API。Amazon API Gateway 顯然能夠適用於服務導向的多層架構，不過，也具備了特定的功能和特性，使其成為您邏輯層的強大優勢。

與 AWS Lambda 整合

Amazon API Gateway 為您的應用程式提供了簡單的方法 (HTTPS 要求)，來直接運用 AWS Lambda 的創新技術。API Gateway 形成了橋樑，連結您的展示層，以及您在 AWS Lambda 中所撰寫的功能。在使用您的 API 來定義用戶端/伺服器的關係後，用戶端 HTTPS 要求的內容就會被傳遞到 Lambda 功能以執行。這些內容包括要求的中繼資料、標頭與內文。

全球各地穩定的 API 效能

Amazon API Gateway 的每項安裝，其中都包含了 Amazon CloudFront[4] 分發服務。Amazon CloudFront 是一項內容交付 Web 服務，使用 Amazon 全球節點網路做為用戶端連結點，來與您的 API 整合。這有助於減少您 API 的整體回應時間的延遲。透過使用全球各地的多重節點，Amazon CloudFront 也讓您能夠對抗分散式阻斷服務攻擊 (DDoS)。如需詳細資訊，請參閱 AWS 對抗 DDoS 攻擊的最佳實務做法 (AWS Best Practices for Combatting DDoS Attacks)[5] 白皮書。

您可以使用 Amazon API Gateway，將回應儲存於選購的記憶體內快取中，以改善特定 API 要求的效能。這不只能為重複的 API 要求，提供效能方面的效益，也可減少後端的執行工作，進而降低您的整體成本。

鼓勵創新

建置任何新應用程式所需的開發工作，都是一項投資。您需要提出投資的正當理由，專案才能開始。透過減少開發工作和時間所需的投資金額，您就能夠將更多的資本，用來進行更多的實驗和創新。

對於許多的多層 Web 服務架構應用程式而言，使用者間的展示層很容易變得零散 (不同的行動裝置、網頁瀏覽器等)。這些使用者也往往沒有地域上的限制。不過，脫鉤的邏輯層，實體上並非是因為使用者才變得零散。所有的使用者皆仰賴執行您邏輯層的同一個基礎設施，因此也凸顯出基礎設施的重要性。在剛開始建置您的邏輯層時，經常會有人建議便宜行事的方法 (「我們不需要在初期推出時衡量指標」；「剛開始的使用量一定不高，所以我們以後再來擔心擴展的事」等)，來更快速地完成新的應用程式。當您必須針對正式營運系統中已經執行的應用程式，來部署這些變更時，這種做法就會造成技術的債務和營運風險。Amazon API Gateway 允許*您*更方便地完成工作，並更快速地推出應用程式，因為這項服務已經為您執行了這些工作。

應用程式的總計生命週期可能是未知的，或可能已知是短暫的。因此，針對新的多層應用程式來撰寫企畫案，可能是件困難的事。如果您的起點已經包含 Amazon API Gateway 所提供的受管功能，只有當您的 API 開始接收要求時，才會開始產生基礎設施的成本，那麼事情可能就會變得更容易。如需詳細資訊，請參閱 Amazon API Gateway 定價。[6]

快速更迭，保持靈活

新應用程式的使用者群，其定義可能仍然模糊 (數量、使用者模式等)。在使用者群的輪廓逐漸成形時，邏輯層必須保持靈活。您的應用程式和業務，應該能夠根據早期使用者不斷改變的期望，配合改變和調整。Amazon API Gateway 減少了 API 從開始到部署完成所需的開發周期。Amazon API Gateway 允許您建立模擬整合[7]，從 API Gateway 直接產生 API 回應 (用戶端應用程式可根據此 API Gateway 來開發)，同時開發完整的後端邏輯。這種做法不只能夠在首次部署 API 時帶來效益，當企業決定必須轉換應用程式 (和現有的 API)，以更快速地回應使用者需求時，也同樣能夠產生效益。API Gateway 和 AWS Lambda 可進行版本控制，如此現有的功能和客戶的依存關係可持續不受中斷，並同時以不同的 API/功能版本，推出新的功能。

安全性

將公有三層 Web 應用程式的邏輯層建置為 Web 服務，馬上就會產生安全性的問題。應用程式需確保只有具授權的用戶端，才能存取您的邏輯層 (邏輯層會透過網路暴露)。Amazon API Gateway 提供方法，讓您對後端的安全性感到放心，解決了安全的問題。控制存取的方法，不能仰賴為用戶端應用程式提供靜態的 API 金鑰字串，因為這些資料可以從用戶端取得，並且用於它處。您可以善用幾種不同的方法，這些方法運用了 Amazon API Gateway 來保護您的邏輯層：

所有對您 API 的要求，皆可透過 HTTPS 進行，以加密傳送中的資料。

您的 AWS Lambda 功能可以限制存取，只讓信任關係存在於 Amazon API Gateway 中的特定 API 和 AWS Lambda 中的特定功能之間。除了使用您選定要暴露的 API 以外，沒有其他任何方法能夠呼叫該 Lambda 功能。

Amazon API Gateway 允許您製作用戶端 SDK，來與您的 API 整合。該 SDK 也會針對需要身分驗證的 API，管理要求的簽署。如有需要，當您自行撰寫的程式碼要求進一步的身份驗證時，這些用戶端的身份驗證用 API 登入資料，會直接傳送到您的 AWS Lambda 功能。

您所建立的每項資源/方法組合 (包含於您的 API 中)，都會被賦予自己特定的 Amazon Resource Name (ARN)，可以在 AWS Identity and Access Management (IAM)[8] 政策中參照這些名稱。

這表示您的 API 和其他 AWS 所擁有的 API，都被視為一等公民。IAM 的政策可精細分級，對於使用 Amazon API Gateway 所製作的 API，也可參考 API 的特定資源/方法。

您在自己應用程式碼的範疇外所制定的 IAM 政策，可加強 API 的存取控管。這表示您不需要撰寫任何程式碼，來注意或落實這些存取權限級別。程式碼不能包含錯誤缺陷，如果有的話，也不能變成漏洞。

授權用戶端時，若使用 AWS Signature 第 4 版 (SigV4)[9] 授權與 IAM 政策，來控管 API 的存取，即可在需要時，使用同樣的登入資料，來限制或允許存取其他的 AWS 服務和資源 (例如，Amazon S3 儲存貯體或 Amazon DynamoDB 資料表)。

AWS Lambda

基本上，對於使用任何支援的語言所撰寫的任意程式碼 (Node、JVM 格式和 Python，截自 2015 年 11 月)，AWS Lambda 允許這些程式碼在回應事件時觸發。此事件可以是 AWS 所提供的數種程式化觸發器之一，這種觸發器稱為**事件來源** (請點選此處以參閱目前支援的事件資源[10])。AWS Lambda 的許多熱門使用案例，都跟事件驅動的資料處理工作流程有關，例如 Amazon Simple Storage Service (Amazon S3)[11] 中所儲存的處理檔案，或是 Amazon Kinesis[12] 的串流資料記錄。

搭配 Amazon API Gateway 使用時，AWS Lambda 功能可存在於典型 Web 服務的範疇中，也可以由 HTTPS 要求直接觸發。Amazon API Gateway 可以充當您邏輯層的前門，但現在您需要在這些 API 的後方執行邏輯。這就是 AWS Lambda 發揮作用之處。

您的商業邏輯在此處發揮作用

AWS Lambda 允許您撰寫稱為**處理器**的程式碼功能，此功能將在事件觸發時執行。例如，您可撰寫由事件觸發的處理器 (例如在出現對您 API 的 HTTPS 要求時)。Lambda 允許您撰寫模組化的處理器，這些處理器具備您所選定的精細分級層級 (每個 API 一個，或每個 API 方法一個)，並且可以個別更新、呼叫和變更。這些處理器接著即可隨意接觸自己所擁有的其他任何依存項目 (例如您連同程式碼一起上傳的其他功能、程式庫、原始二進位碼，或甚至是外部的 Web 服務)。Lambda 允許您在撰寫功能定義時，納入所有必需的依存項目。製作功能時，您會指定在部署套件中，將做為要求處理器的方法。您可以隨意地將同樣的部署套件，重複用於多個 Lambda 功能定義，而每項 Lambda function 功能，可能會在同一個部署套件中擁有獨特的處理器。在無需伺服器的多層架構模式中，您在 Amazon API Gateway 中所製作的每個 API，都將與 Lambda 功能 (和其中的處理器) 整合，而此一 Lambda 功能是執行商業邏輯所必需。

Amazon VPC 整合

您邏輯層的核心 AWS Lambda，將會成為與資料層直接整合的元件。由於資料層將會時常包含機密的業務或使用者資訊，因此資料層應該加以嚴格的保護。如果是您可以透過 Lambda 功能整合的 AWS 服務，則可使用 IAM 政策來控管存取。這些服務包括 Amazon S3、Amazon DynamoDB、Amazon Kinesis、Amazon Simple Queue Service (Amazon SQS)、Amazon 簡易通知服務 (Amazon SNS)，和其他的 AWS Lambda 功能等。不過，可能會有元件自行控管其存取，例如關聯式資料庫。對於這類元件，您可以透過在私有的聯網環境中進行部署 (例如 Amazon Virtual Private Cloud (Amazon VPC))[13]，來達成更高的安全性。

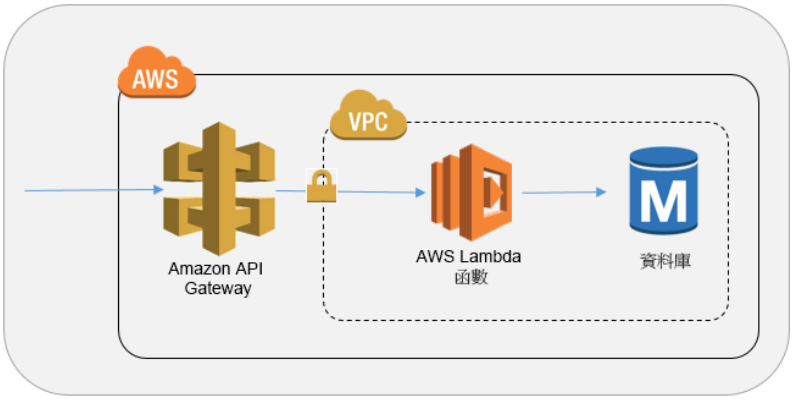

圖 2：使用 VPC 的架構模式

VPC 的使用，代表您的商業邏輯所依賴的資料庫和其他儲存媒體，無法從網際網路進行存取。VPC 也可確保，從網際網路與您資料互動的*唯一*方法，就是透過您所定義的 API，以及您所撰寫的 Lambda 程式碼功能。

安全性

Lambda 功能必須由事件或服務觸發才能執行，而在 IAM 政策中，會定義這些能夠觸發功能的事件或服務。您可以撰寫 Lambda 功能，使其只能由您所定義的 API Gateway 進行呼叫，否則就*完全*無法執行。您的程式碼只會被視為您有效使用案例的一部分來處理 (由您所製作的 API 定義)。

每項 Lambda 功能本身皆具備 IAM 的角色，而這必須透過 IAM 信任關係賦予。此等 IAM 角色，會定義您的 Lambda 功能可與之互動的其他 AWS 服務/資源 (例如 Amazon DynamoDB 資料表或 Amazon S3 儲存貯體)。您的功能所能存取的服務，將會從此功能本身以外定義和控制。這雖然是小細節，卻提供了強大的能力。此項做法允許您所撰寫的程式碼，不用再儲存或擷取 AWS 登入資料：這表示您不需要再將 API 金鑰寫入程式碼，也不需要再撰寫程式碼來將擷取金鑰，並將其儲存於記憶體中。讓您的 Lambda 功能呼叫所允許的服務 (由功能的 IAM 角色定義)，而服務本身就會為您管理好這件事。

資料層

使用 AWS Lambda 做為您的邏輯層，將允許您的資料層擁有廣泛的資料儲存選項。這些選項分為兩大類：Amazon VPC 託管的數據存放和 IAM 所實現的數據存放。AWS Lambda 能夠安全地和這兩種選項整合。

Amazon VPC 託管的數據存放

AWS Lambda 與 Amazon VPC 的整合，能夠以私有而安全的方法，讓功能與各種資料儲存技術整合。

Amazon RDS[14]

使用 Amazon Relational Database Service (Amazon RDS) 所提供的任一引擎。從您在 Lambda 中所撰寫的程式碼，直接連結至 Amazon RDS，就如同 Lambda 以外的程式碼，但好處是能夠簡單地與 AWS Key Management Service (AWS KMS) 整合，以加密資料庫的登入資料。

[Amazon ElastiCache][15]

將您的 Lambda 功能整合受管的記憶體內快取，以大幅提升您應用程式的效能。

[Amazon Redshift][16]

您可建置功能，使其透過安全的方式查詢企業資料倉儲庫，以製作報告、儀表板或擷取一次性的查詢結果。

由 [Amazon Elastic Compute Cloud (Amazon EC2)][17]所代管的私有 Web 服務。

您現有的應用程式，可能是在 VPC 內執行的私有 Web 服務。透過您邏輯上為私有的 VPC 網路，從 Lambda 功能進行 HTTP 要求。

IAM 實現的數據存放

由於 AWS Lambda 整合了 IAM，因此能夠使用 IAM，針對可使用 AWS API 直接運用的任何 AWS 服務，進行安全的整合。

[Amazon DynamoDB][18]

Amazon DynamoDB 是 AWS 可無限擴展的 NoSQL 資料庫。如果您希望能夠在幾毫秒內擷取資料記錄 (等於或小於 400KB，此為本文撰寫時的資訊)，可考慮使用 Amazon DynamoDB (無論資料庫的規模大小為何)。若使用 Amazon DynamoDB 精細定義的存取控制機制，您的 Lambda 功能可遵循最佳實務做法 - 查詢 DynamoDB 中的特定資料時擁有最小的權限。

[Amazon S3][19]

Amazon Simple Storage Service (Amazon S3) 提供網際網路規模的物件儲存服務。Amazon S3 的設計，適用於物件 99.999999999% 的存續時間，因此當您的應用程式需要便宜而高耐用性的儲存服務時，可考慮使用 Amazon S3。此外，Amazon S3 的設計，可在特定一年中，達成最高 99.99% 的物件可用性，因此當您的應用程式需要高可用性的儲存服務時，也可考慮使用 Amazon S3。存放於 Amazon S3 中的物件 (檔案、影像、日誌、任何二進位資料)，可透過 HTTP 直接存取。Lambda 功能可透過虛擬的私有終端節點，和 Amazon S3 進行安全的通訊，而 S3 內的資料可以受到限制，只讓與 Lambda 功能相關的 IAM 政策存取。

[Amazon Elasticsearch Service](20)

Amazon Elasticsearch Service (Amazon ES) 是 Elasticsearch 的受管版本，而 Elasticsearch 是一項熱門的搜尋與分析引擎。Amazon ES 提供受管的叢集佈建、故障偵測與節點更換功能；您可以利用 IAM 政策，來限制對 Amazon ES API 的存取。

展示層

Amazon API Gateway 開啟了展示層的各種可能性。任何具備 HTTPS 通訊能力的用戶端，皆可使用透過網際網路存取的 HTTPS API。下列清單包含常見的範例，您的應用程式展示層可以考慮使用：

行動應用程式：除了透過 Amazon API Gateway 和 AWS Lambda，與自訂的商業邏輯進行整合之外，您也可以使用 [Amazon Cognito](21)，做為建立和管理使用者身分識別資料的機制。

靜態的網路內容（例如在 Amazon S3 中所代管的檔案）：您可以讓您的 Amazon API Gateway API，相容於跨來源資源共享 (CORS) 標準。如此，Web 瀏覽器即可從靜態 Web 直接呼叫您的 API。

其他所有的 HTTPS 功能用戶端裝置：有許多互連的裝置可透過 HTTPS 進行通訊。對於您使用 Amazon API Gateway 所製作的 API，用戶端與其進行通訊的方式，並無獨特或專門之處，就只是 HTTPS。不需要特定的用戶端軟體或授權。

架構模式範例

您可以使用 Amazon API Gateway 和 AWS Lambda 做為組成邏輯層的連結機制，來建置下列熱門的架構模式。例如，有些 AWS 服務不需要由使用者管理自己的基礎設施，而我們只將使用此種 AWS 服務。

行動後端

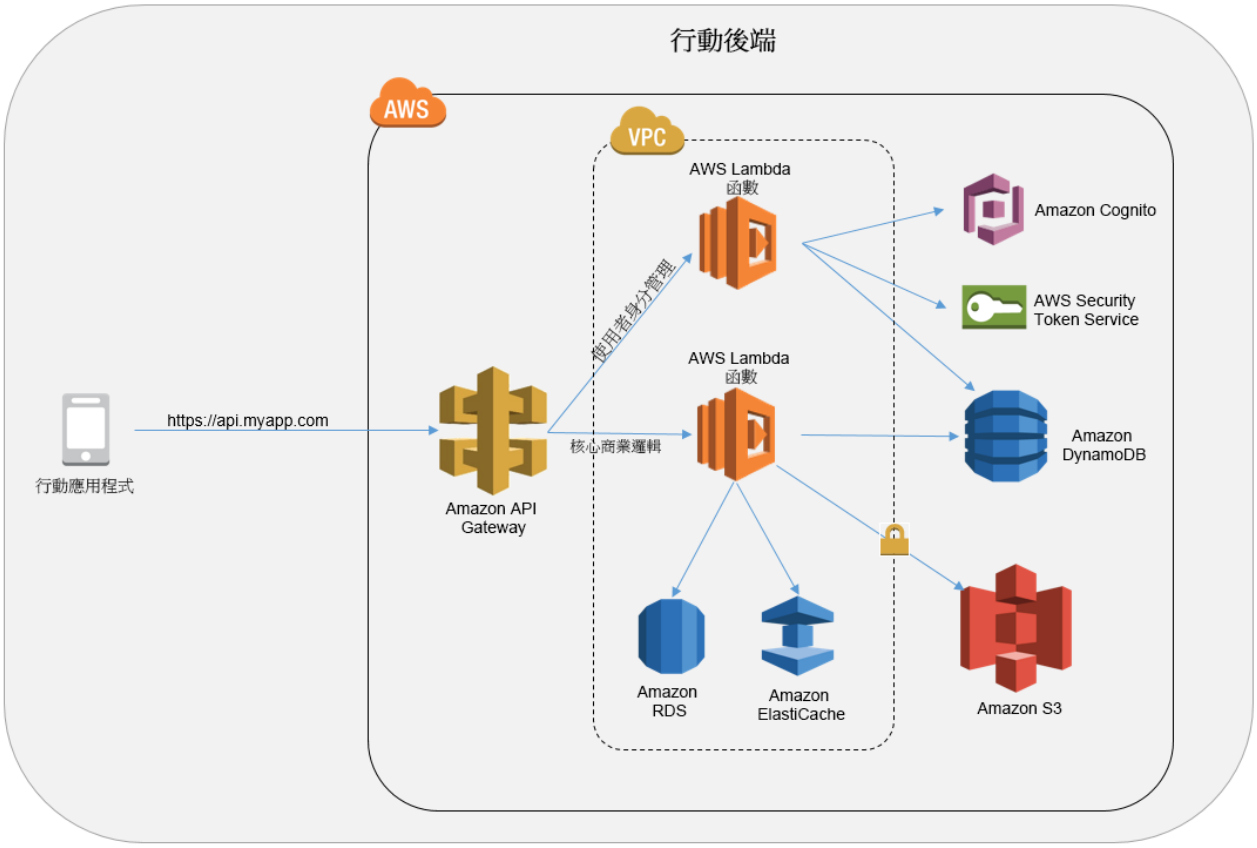

圖 3：適用於行動後端的架構模式

展示層：在每個使用者的智慧型手機上所執行的行動應用程式。

邏輯層：Amazon API Gateway 與 AWS Lambda。邏輯層由 Amazon CloudFront 分發服務全球發佈，成為每個 Amazon API Gateway API 的一部分。一組 Lambda 功能可能會特定連結於使用者/裝置身分管理與身份驗證，並由 Amazon Cognito 管理，Amazon Cognito 會提供與 IAM 的整合機制，以應用於臨時使用者的存取登入資料，以及熱門的第三方身份供應商。其他的 Lambda 功能，可以定義適用於您行動後端的核心商業邏輯。

資料層：如有需要，可運用各種資料儲存服務；選項如本白皮書前述的說明。

Amazon S3 代管網站

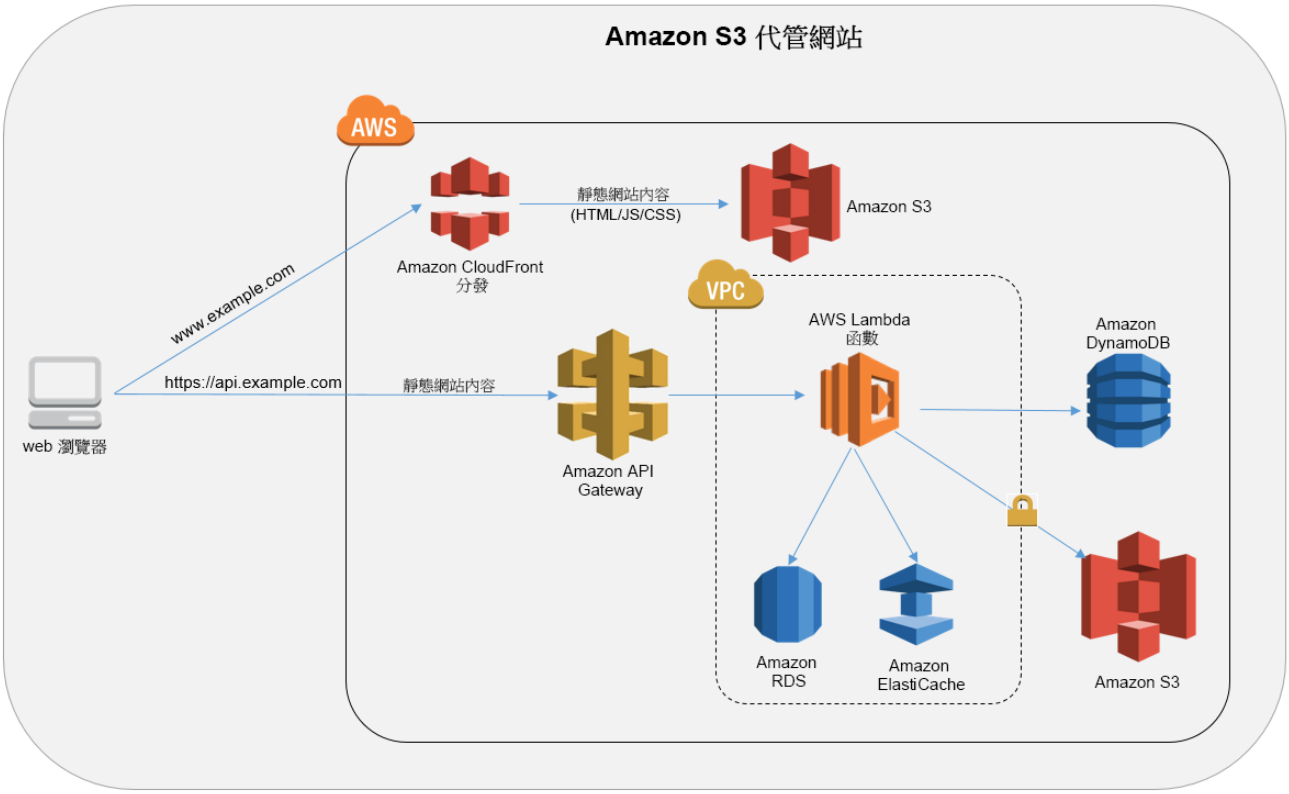

圖 4：適用於 Amazon S3 所代管靜態網站的架構模式

展示層：Amazon S3 中所代管的靜態網站內容，由 Amazon CloudFront 發佈。相較於將內容建置於伺服器式的基礎設施中，由 Amazon S3 代管靜態網站內容，是一項具成本效益的替代方案。不過，如果網站包含豐富的功能，則靜態的內容經常必須與動態的後端整合。

邏輯層：Amazon API Gateway 與 AWS Lambda。Amazon S3 中所代管的靜態網路內容，可以與 Amazon API Gateway 直接整合，而 Amazon API Gateway 可相容於 CORS 標準。

資料層：如有需要，可運用各種資料儲存服務；選項如本白皮書前述的說明。

微服務環境

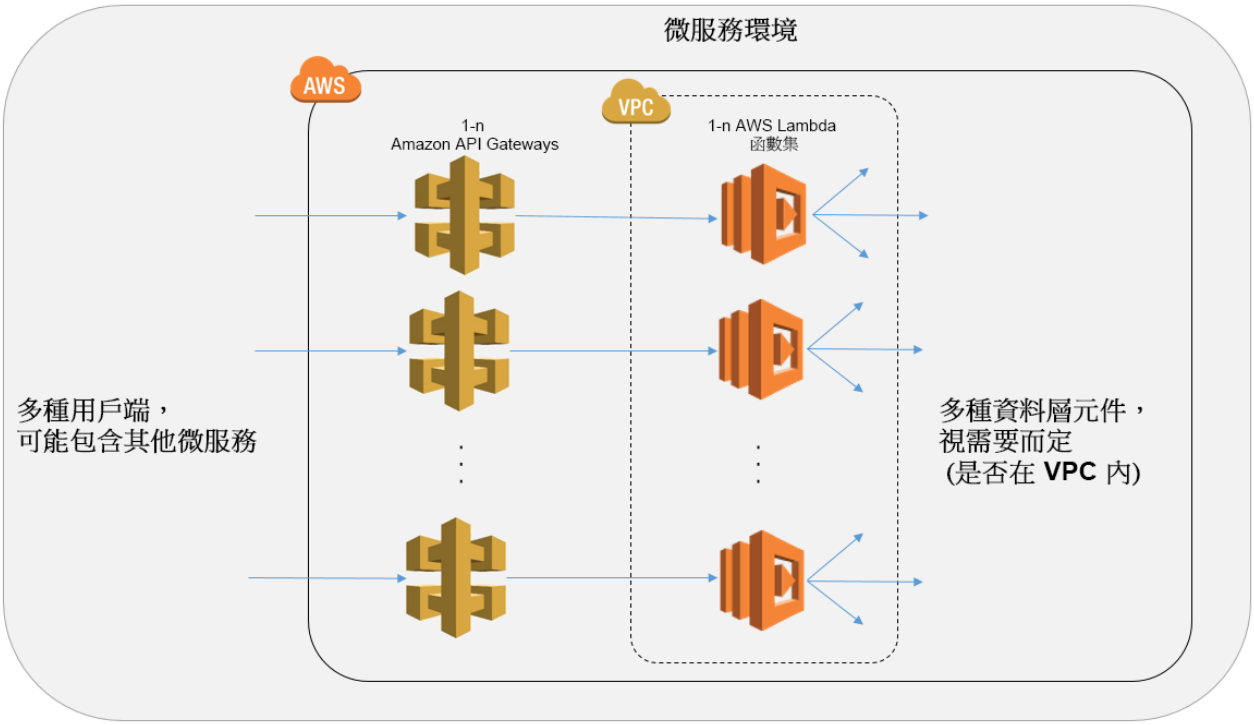

圖 5：適用於微服務環境的架構模式

微服務架構模式不限於本白皮書先前所說明的典型三層架構。在微服務的架構中，軟體元件大量地去耦，因此多層架構效益整個放大。建置微服務時，您只需要由 Amazon API Gateway 所製作的 API，以及後續由 AWS Lambda 所執行的功能。您的團隊可以隨意使用這些服務，讓您的環境脫鉤和裂解，以達到所要的精細分級程度。

一般而言，微服務環境可能會產生下列難題：在建置每個新的微服務時，重複產生的間接成本、伺服器密度/利用率最佳化的問題、同時執行多項微服務的多重版本，所產生的複雜性，以及用戶端程式碼整合許多不同服務的大量增長需求。

不過，當您採用 AWS 無伺服器模式來建置微服務時，前述的這些問題都變得更容易解決，在某些案例中，甚至問題根本消失。AWS 的微服務模式，減少了在建置後續各個微服務時，所出現的障礙 (Amazon API Gateway 甚至允許複製現有的 API)。採用此種模式，就不再需要達成最佳伺服器利用率。API Gateway 和 Lambda 皆實現了簡單的版本控制功能。最後，Amazon API Gateway 提供可透過程式產生的用戶端 SDK (可使用多種熱門的語言)，以降低整合的間接成本。

結論

多層架構模式促進了最佳實務做法，也就是建置易於維護、去耦和擴展的應用程式元件。如果建置的邏輯層，是透過 Amazon API Gateway 進行整合，並利用 AWS Lambda 進行運算，則您將能夠實現這些目標，同時減少達成這些目標所需的心力。結合這些服務，可提供用戶端的 HTTPS API 前端，以及 VPC 中的安全環境 (您可在其中執行商業邏輯)。如此，您就能善用許多熱門的使用案例，運用這些託管的服務，而不需自行管理典型的伺服器式基礎設施。

作者群

協力完成這份文件的個人與組織如下：

AWS 解決方案架構師 Andrew Baird

AWS Mobile 技術部資深產品經理 Stefano Buliani

AWS Mobile 資深產品經理 Vyom Nagrani

AWS Mobile 資深產品經理 Ajay Nair

備註

[1] http://aws.amazon.com/api-gateway/

[2] http://aws.amazon.com/lambda/

[3] https://aws.amazon.com/vpc/

[4] https://aws.amazon.com/cloudfront/

[5] https://d0.awsstatic.com/whitepapers/DDoS_White_Paper_June2015.pdf

[6] https://aws.amazon.com/api-gateway/pricing/

[7] http://docs.aws.amazon.com/apigateway/latest/developerguide/how-to-mock-integration.html

[8] http://aws.amazon.com/iam/

[9] http://docs.aws.amazon.com/general/latest/gr/signature-version-4.html

[10] http://docs.aws.amazon.com/lambda/latest/dg/intro-core-components.html#intro-core-components-event-sources

[11] https://aws.amazon.com/s3/

[12] https://aws.amazon.com/kinesis/

[13] https://aws.amazon.com/vpc/

[14] https://aws.amazon.com/rds/

[15] https://aws.amazon.com/elasticache/

[16] https://aws.amazon.com/redshift/

[17] https://aws.amazon.com/ec2/

[18] https://aws.amazon.com/dynamodb/

[19] https://aws.amazon.com/s3/storage-classes/

[20] https://aws.amazon.com/elasticsearch-service/

[21] https://aws.amazon.com/cognito/

amazon web services