AWS 白皮書

實作 AWS 上的微型服務

實作 AWS 上的微型服務: AWS 白皮書

Table of Contents

實作 AWS 上的微型服務

出版日期： 2021 年 11 月 9 日 (文件歷史記錄和貢獻者)

摘要

微型服務是一種架構與組織方法，旨在加速已建立之軟體開發的部署週期、促進創新與權責分工、改善軟體應用程式的可維護性和可擴展性，以及使用靈活的方法來協助團隊獨立運作，藉此擴展提供軟體和服務的組織。利用微型服務方法，軟體包含了各個小型服務，服務之間透過定義明確且可獨立部署的應用程式介面 (API) 進行通訊。這些服務皆由小型的自主團隊負責。這個靈活的方法是擴展您組織的成功關鍵。

AWS 客戶建置微型服務時的三種常見模式為：API 導向、活動導向和資料串流。本白皮書介紹了微型服務的三種方法，並摘要說明這些方法的共同特性，探討建置微型服務的主要挑戰，及產品團隊可如何使用 Amazon Web Services (AWS)，來克服這些挑戰。

由於本白皮書中討論的各種主題性質相當複雜，包括資料存放區、非同步通訊和服務探索，除了在進行架構選項之前提供的指導之外，建議您考慮其應用程式的特定要求及使用案例。

簡介

微型服務架構並非全新的軟體工程方法，而是將成功的和經過驗證的概念加以彙總組合，例如：

- 敏捷式軟體開發
- 服務導向架構
- API 優先設計
- 持續整合/持續交付 (CI/CD)

在許多的情況中，微型服務採用 12 要素應用程式的設計模式。

本白皮書首先說明高度可擴展、容錯型微型服務架構 (使用者介面、微型服務實作與資料存放區) 的不同面向，以及運用容器技術，在 AWS 上建置此架構。接著，其會建議適合的 AWS 服務，這些服務可用來建置典型的無伺服器微型服務架構，以降低營運的複雜度。

無伺服器定義為操作模式，採用的原則如下：

- 無須佈建或管理基礎設施

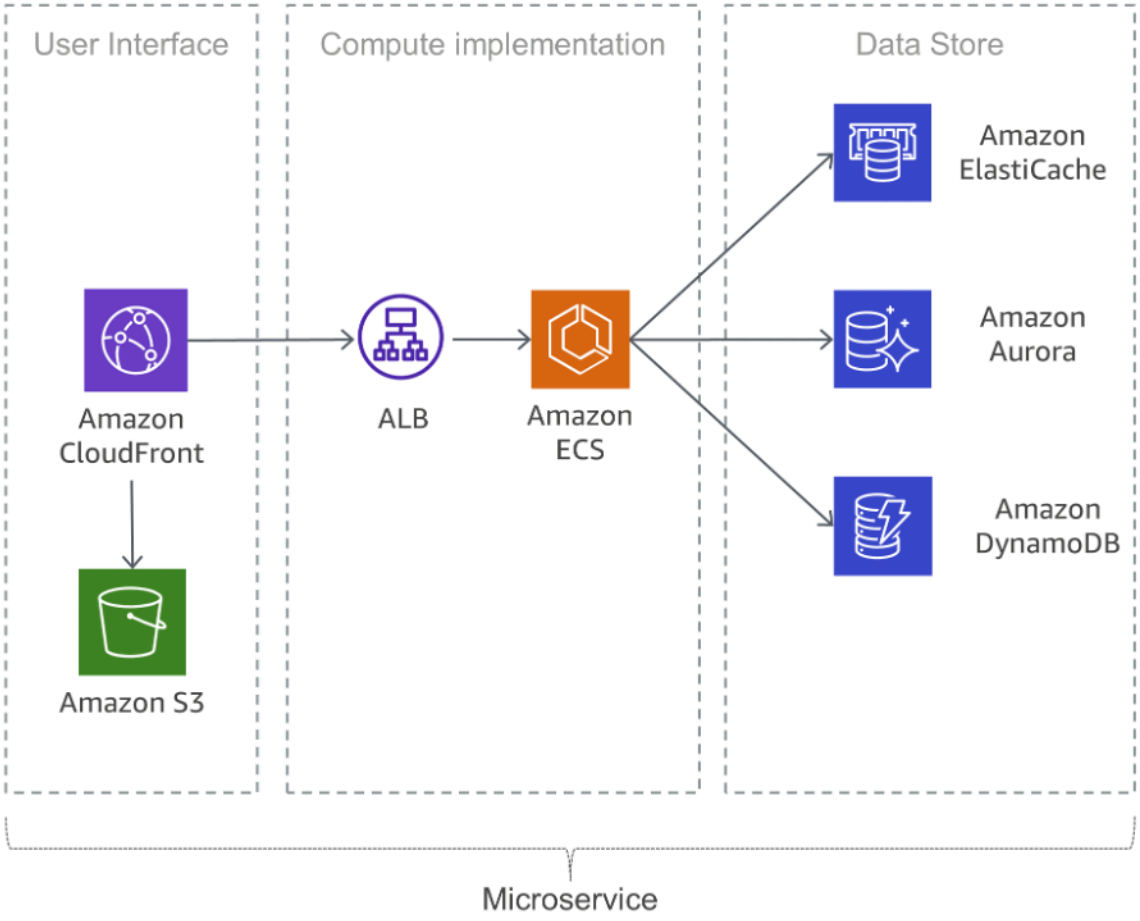
- 依照消耗單位自動調整規模
- 按價值付費計費模式
- 內建可用性與容錯能力

最後，本白皮書涵蓋整體系統，並說明微型服務架構的跨服務面向，例如分散式監控與稽核、資料一致性和非同步通訊。

本白皮書僅著重於 AWS 雲端中執行的工作負載。其不包含混合案例或遷移策略。如需有關遷移的詳細資訊，請參閱容器遷移方法白皮書)。

AWS 上的微型服務架構

典型的整合型應用程式是使用不同分層所建置，也就是使用者介面 (UI) 層、業務層和持久層。微型服務架構的中心概念，是將功能劃分為連貫的垂直項目，這並非是依技術層，而是透過建置特定的領域來劃分。下圖顯示在 AWS 上典型微型服務應用程式的參考架構。



AWS 上的典型微型服務應用程式

主題

- 使用者介面
- 微型服務
- 資料存放區

使用者介面

現代的 Web 應用程式，通常會使用 JavaScript 架構來建置單一頁面的應用程式，此等應用程式使用表現層狀態轉移 (REST) 或 RESTful API 進行通訊。靜態的網站內容，可使用 Amazon Simple Storage Service (S3) 和 Amazon CloudFront 加以提供。

由於微型服務會從距離用戶端最近的節點來為用戶端提供資源，且從快取或是與原始伺服器具有最佳化連線的代理伺服器來取得回應，因此可大幅地減少延遲。不過，運作時彼此接近的微型服務，並不會受益於內容交付網路。在某些情況中，此方法可能反而會增加更多的延遲。最佳的實務做法是建置其他的快取機制，以減少繁瑣，並且將延遲降到最低。如需詳細資訊，請參閱 the section called “繁瑣性” 主題。

微型服務

API 是微型服務的前門，這表示 API 會做為一組可程式設計介面背後的應用程式邏輯，通常是 RESTful Web 服務 API。此 API 會接受和處理來自用戶端的呼叫，也可能會建置一些功能，例如流量管理、請求篩選、路由、快取和身分驗證與授權。

微型服務實作

AWS 具備整合式建置區塊，可支援微型服務的開發。使用 AWS Lambda 和 Docker 容器搭配 AWS Fargate 是常見的兩種方法。

利用 AWS Lambda，您可上傳程式碼，然後讓 Lambda 來處理執行所需的一切作業，並調整執行的規模，以滿足您的實際需求曲線，提供高可用性。無須管理基礎設施。Lambda 支援多種程式設計語言，並且可從其他 AWS 服務呼叫，或是從任何 Web 或行動應用程式直接呼叫。AWS Lambda 的其中一項最大優點，就是移動迅速：您可以專注在自己的商業邏輯上，因為 AWS 會管理安全性和擴展。Lambda 的主導性方法促進了可擴展平台。

如果想要減少部署作業所需投注的營運心力，一個常見的方法就是使用容器來進行部署。基於下列的優點，Docker 等容器技術，在最近幾年日益受到歡迎：可攜性、生產力和效率等。容器的學習曲線可能很陡峭，而您必須思考 Docker 影像和監控的安全性修正。Amazon Elastic Container Service (Amazon ECS) 和 Amazon Elastic Kubernetes Service (Amazon EKS) 讓您無需安裝、操作及擴充自己的叢集管理基礎設施。使用 API 呼叫，您可啟動和停止啟用 Docker 的應用程式、查詢叢集的完整狀態，及使用許多孰悉的功能，例如安全群組、負載平衡器、Amazon Elastic Block Store (Amazon EBS) 磁碟區和 AWS Identity and Access Management (IAM) 角色等。

AWS Fargate 是一種無伺服器運算引擎，適用於同時使用 Amazon ECS 和 Amazon EKS 的容器。使用 Fargate，您完全不用操心，為容器應用程式佈建足夠的運算資源。Fargate 可以啟動數十萬個容器，並輕鬆擴展以執行您最關鍵任務的應用程式。

Amazon ECS 支援容器放置策略與限制，以自訂 Amazon ECS 放置和終止工作的方式。工作放置位置的限制是一項規則，在放置工作時會考量這項規則。您可將屬性 (基本上就是鍵值/值組) 與容器執行個體建立關聯，然後根據這些屬性，配合限制來放置工作。例如，您可以利用限制並根據執行個體類型或容量放置某些微型服務，例如運用 GPU 的執行個體。

Amazon EKS 會執行最新版本的開放原始碼 Kubernetes 軟體，所以您可使用 Kubernetes 社群中所有的現有外掛程式和工具。在 Amazon EKS 上執行的應用程式與在任何標準 Kubernetes 環境上執行的應用程式完全相容，不論是在現場部署的資料中心還是在公有雲端中執行。Amazon EKS 將 IAM 與 Kubernetes 整合在一起，讓您可以使用 Kubernetes 中的原生驗證系統註冊 IAM 實體。完全無須手動設定憑證，即可向 Kubernetes 控制平面進行驗證。IAM 的整合可讓您提供對 Kubernetes 控制平面公開端點的精細存取權，使用 IAM 直接向控制平面本身進行驗證。

用於 Amazon ECS 和 Amazon EKS 中的 Docker 影像可以存放於 Amazon Elastic Container Registry (Amazon ECR) 中。Amazon ECR 讓您不需運作和擴展基礎設施，來支援容器的登錄檔。

持續整合和持續交付 (CI/CD) 是最佳實務做法，也是 DevOps 計劃至關重要的一環，能實現快速軟體變更，同時維護系統穩定性和安全性。但是，這超出了本白皮書的範圍。如需詳細資訊，請參閱在 AWS 上實作持續整合與持續交付白皮書。

私有連結

AWS PrivateLink 是一項具有高可用性和可擴展性的技術，可讓您將您的 Virtual Private Cloud (VPC) 私下連線至支援的 AWS 服務、由其他 AWS 帳戶託管的服務 (VPC 端點服務)，以及支援的 AWS Marketplace 合作夥伴服務。您不需要網際網路閘道、網路位址轉譯裝置、公有 IP 地址、AWS Direct Connect 連線或 VPN 連線，即可與服務進行通訊。您 VPC 與 服務之間的流量都會保持在 Amazon 網路的範圍內。

私有連結是提高微型服務架構隔離性和安全性的絕佳方法。例如，微型服務可部署於完全獨立的 VPC 中，在負載平衡器前面，並透過 AWS PrivateLink 端點顯現於其他微型服務。利用此設定使用 AWS PrivateLink，進出微型服務的網路流量絕對不會周遊於公有網際網路。此種隔離的某個使用案例包括處理敏感資料 (如 PCI、HIPPA 和歐盟/美國隱私盾) 等服務的法規合規。此外，無須防火牆規則、路徑定義或路由表，AWS PrivateLink 可連線不同帳戶和 Amazon VPC 的微型服務；簡化網路管理。使用 PrivateLink，軟體即服務 (SaaS) 供應商和 ISV 也可為其微型服務型解決方案提供完整的作業隔離和安全存取。

資料存放區

資料存放區是用來長久保留微型服務所需的資料。熱門的工作階段資料存放區，是 Memcached 或 Redis 等記憶體內快取。AWS 在受管的 Amazon ElastiCache 服務中，同時提供了這兩項技術。

將快取放置在應用程式伺服器和資料庫之間，是一項常見的機制，用以減輕從資料庫的讀取工作負載，進而可讓資源用來支援更多的寫入操作。快取亦可改善延遲的情況。

關聯式資料庫仍然非常熱門，用來儲存結構化資料與業務物件。AWS 經由 Amazon Relational Database Service (Amazon RDS) 以受管服務的形式，提供六種資料庫引擎 (Microsoft SQL Server、Oracle、MySQL、MariaDB、PostgreSQL 和 Amazon Aurora)。

不過，關聯式資料庫的設計無法不受限地擴展，因此若要運用技術來讓這種資料庫支援大量的查詢，會變得困難和費時。

相較於關聯式資料庫的一致性，NoSQL 資料庫在設計時，已經更加著重於可擴展性、效能和可用性。其中一個重要元素，是 NoSQL 資料庫通常不會強制執行嚴格的結構描述。資料會分散到可水平擴展的分割區，並使用分割區索引鍵擷取。

由於個別微型服務設計只專注於執行一個任務，因此通常擁有簡化的資料模型，可能適合 NoSQL 的持久性。請務必了解，NoSQL 資料庫的存取模式和關聯式資料庫的並不相同。例如，資料表無法合併。如果這是必要的，必須在應用程式中實作此邏輯。您可使用 Amazon DynamoDB 建立資料庫表格，以存放和擷取任意數量的資料，並處理任何規模的請求流量。DynamoDB 提供一位數毫秒延遲的效能，不過，在某些使用案例中，需要以微秒為時間單位來回應。Amazon DynamoDB Accelerator (DAX) 提供快取功能來存取資料。

DynamoDB 還提供自動規模調整功能，可動態調整吞吐容量，以因應實際的流量。但是，有時會因為您應用程式中短期的大量活動高峰而難以或無法進行容量規劃。針對這類情況，DynamoDB 提供隨需選項，提供依要求逐次計費的簡便方式。DynamoDB 隨需選項能立即處理每秒數千個要求，無須進行容量規劃。

降低營運的複雜性

先前說明於本白皮書中的架構已使用受管服務，但仍需管理 Amazon Elastic Compute Cloud (Amazon EC2) 執行個體。藉由使用完全無伺服器的架構，針對微型服務的執行、維護與監控，進一步減少所需的營運工作。

主題

- API 實作
- 無伺服器微型服務
- 災難復原
- 高可用性
- 部署 Lambda 型應用程式

API 實作

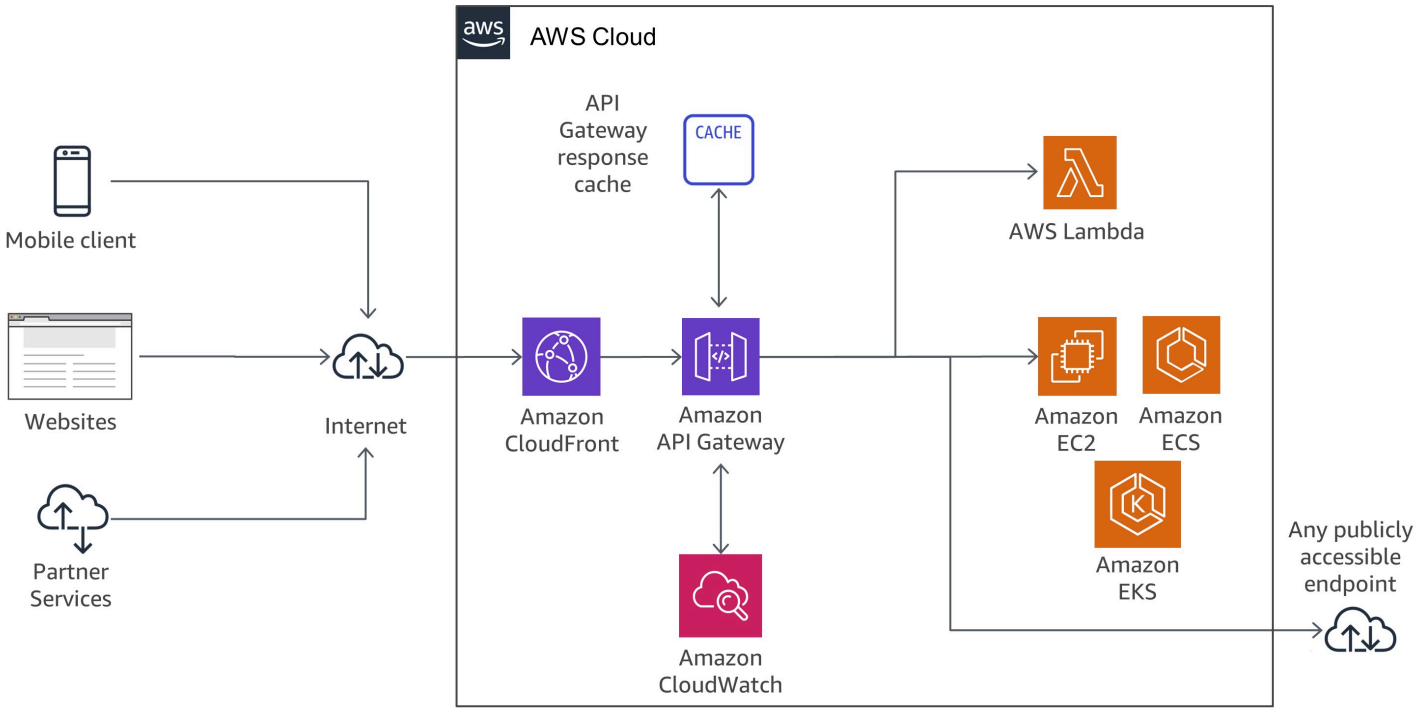
針對 API 進行架構的建立、部署、監控、持續改善與維護作業，是十分耗時的工作。有時候需要執行不同版本的 API，以確保所有用戶端的回溯相容性。開發週期的各個階段 (例如，開發、測試和生產)，每進一步都需要投注加倍的營運心力。

授權對所有 API 而言，是一項至關重要的功能，但通常涉及複雜的建置和重複的工作。API 發佈並變得熱門時，接下來的挑戰就是，針對使用 API 的第三方開發人員，進行其生態系統的管理、監控和從中獲利。

其他重要的功能和挑戰，包括調節請求的數量以保護後端服務、建立 API 回應的快取、處理請求與回應的轉換，以及使用 Swagger 等工具來產生 API 的定義和文件。

Amazon API Gateway 克服了這些挑戰，並針對 RESTful API 的建立與維護作業，降低了運作的複雜性。API Gateway 可讓您使用 AWS API 或 AWS 管理主控台，透過匯入 Swagger 的定義，以程式設計方式建立您自己的 API。在 Amazon EC2、Amazon ECS、AWS Lambda 上或任何內部部署環境中執行的任何 Web 應用程式，皆可使用 API Gateway 做為其門戶。基本上，API Gateway 可讓您執行 API，而不需要管理伺服器。

下圖說明 API Gateway 如何處理 API 呼叫，以及如何與其他的元件互動。來自行動裝置、網站或其他後端服務的請求，會轉傳到最靠近的 CloudFront 網路連接點 (PoP)，以將延遲減到最少，並提供最佳的使用者體驗。



API 閘道呼叫流程

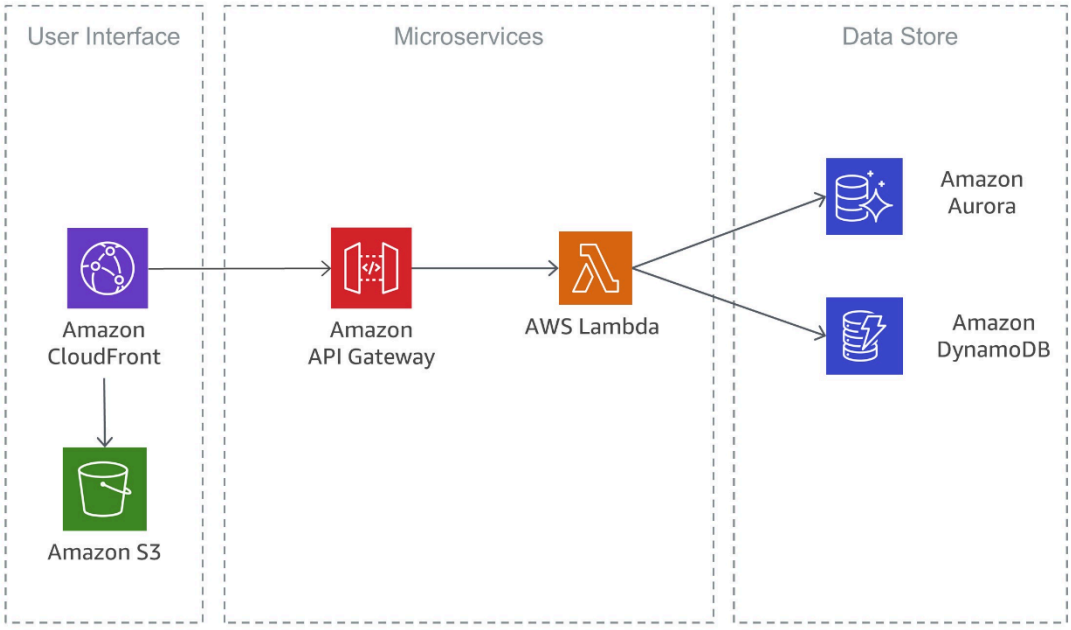
無伺服器微型服務

「無伺服器比沒有伺服器更容易管理」。

擺脫伺服器是消除營運複雜性的絕佳方法。

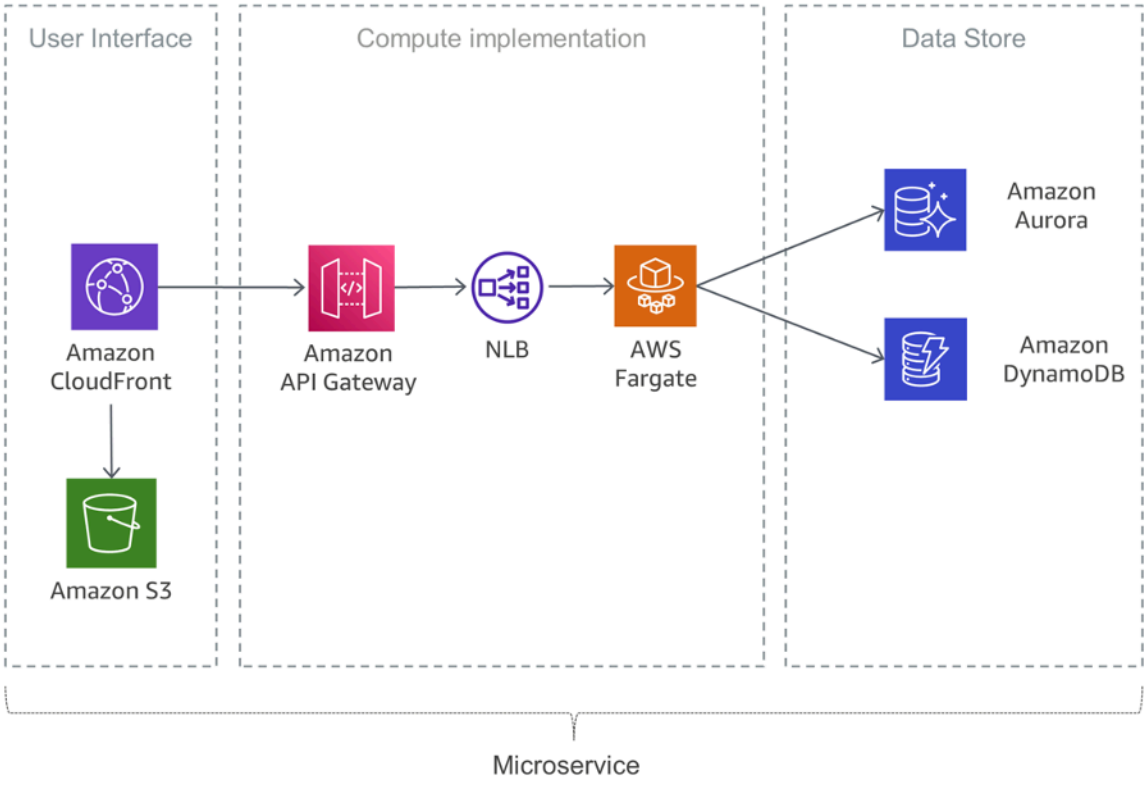
Lambda 已與 API Gateway 高度整合。從 API Gateway 向 Lambda 發出同步呼叫的功能，讓您能夠建立完全無伺服器的應用程式，如需詳細說明，請參閱 Amazon API Gateway 開發人員指南。

下圖顯示 AWS Lambda 無伺服器微型服務的架構，其中完整的服務是由受管服務建置而成，這可消除針對擴展和高可用性進行設計時的架構負擔，並減少在執行與監控微型服務的基礎設施時，所需投注的營運心力。



使用 AWS Lambda 的無伺服器微型服務

下圖中顯示了也採用無伺服器服務的類似實作。在這個架構中，Docker 容器與 Fargate 搭配使用，所以無須操心基礎設施。除了 DynamoDB 之外，也使用了 Amazon Aurora Serverless，這是針對 Amazon Aurora (MySQL 相容版本) 隨需自動調整規模的組態，其中資料庫將會根據應用程式的需求，自動啟動、關閉或擴展容量。



使用 Fargate 的無伺服器微型服務

災難復原

如先前於本白皮書的簡介中所提及，典型的微型服務應用程式是使用 12 要素應用程式模式進行實作。程序章節指出「12 要素程序為無狀態且不共用。任何需要保留的資料皆必須存放於有狀態的備份服務中，通常是個資料庫。」

對於典型的微型服務架構，這表示災難復原的主要重點應該置於維護應用程式狀態的下游服務上。例如，這些可為檔案系統、資料庫或佇列。建立災難復原策略時，組織通常會規劃復原時間點目標和復原點目標。

復原時間點目標是服務中斷與恢復服務之間的最大可接受延遲。此目標會決定可接受的服務無法使用的時間長度，為組織所定義。

復原點目標是自上次資料復原點之後的最大可接受時間長度。這目標會決定最後一個復原點與服務中斷之間可接受的資料遺失，且為組織所定義。

如需詳細資訊，請參閱 AWS 上的工作負載災難復原：雲端中的復原白皮書。

高可用性

本節將詳細介紹不同運算選項的高可用性。

Amazon EKS 在多個可用區域內執行複數 Kubernetes 控制和資料平面執行個體以確保高可用性。Amazon EKS 會自動偵測並替換有問題的控制平面執行個體，而且提供自動化版本升級及修補。此控制平面包含至少兩個 API 伺服器節點和三個 etcd 節點，並在一個區域內跨三個可用區域執行。Amazon EKS 使用 AWS 區域 架構，來維持高可用性。

Amazon ECR 會在高可用性及高效能架構中託管映像，讓您在可用區域間可靠地為容器應用程式部署映像。Amazon ECR 與 Amazon EKS、Amazon ECS 和 AWS Lambda 搭配使用，簡化從開發到生產的工作流程。

Amazon ECS 是一項區域服務，可在一個 AWS 區域 中跨多個可用區域，以高可用性的方式簡化執行容器。Amazon ECS 包含多個排程策略，可根據您的資源需要 (例如，CPU 或 RAM) 與可用性需求，將容器置於各叢集。

AWS Lambda 會在多個可用區域中執行您的函式，確保單一區域的服務中斷時，其可用於處理事件。如果您將函式設定為連接到您帳戶中的 Virtual Private Cloud (VPC)，請在多個可用區域中指定子網路，以確保高可用性。

部署 Lambda 型應用程式

您可使用 AWS CloudFormation 指定、部署以及設定無伺服器應用程式。

AWS Serverless Application Model (AWS SAM) 是個定義無伺服器應用程式的簡單方法。AWS SAM 由 CloudFormation 原生支援，定義了簡化的語法以表達無伺服器資源。若要部署應用程式，您需於應用程式中指定所需的資源、使用 CloudFormation 範本來指定資源相關的權限政策、將部署的成品做成套件，然後部署範本即可。根據 AWS SAM，SAM Local 是個 AWS Command Line Interface (AWS CLI) 工具，提供環境供您先在本機開發、測試和分析無伺服器的應用程式後，再上傳至 Lambda 執行時間。您可使用 AWS SAM Local 建立本機測試環境，來模擬 AWS 執行階段環境。

分散式系統元件

在介紹 AWS 如何解決個別微型服務的相關挑戰後，焦點將會移至跨服務的挑戰，例如服務探索、資料一致性、非同步通訊和分散式監控與稽核。

主題

服務探索

使用微型服務架構的主要挑戰之一，是讓服務彼此探索與互動。微型服務架構的分散式特點，不僅讓服務之間更難以進行通訊，也帶來了其他挑戰，例如檢查這些系統的運作狀態，和宣布新應用程式可用的時間。此外，您也必須決定儲存中繼資料存放區資訊的方式與位置，例如應用程式可以使用的組態資料。在本節中會探討幾種技術，這些技術可針對微型服務型架構，在 AWS 上進行服務探索。

DNS 型服務探索

Amazon ECS 現在包含整合式服務探索功能，可讓您的容器化服務輕鬆探索彼此並進行互連。

過去，為了確保服務能探索彼此並進行互連，您必須根據 Amazon Route 53、AWS Lambda 和 ECS Event Stream 設定和執行自己的服務探索系統，或將每個服務連接至負載平衡器。

Amazon ECS 會使用 Route 53 Auto Naming API 建立和管理服務名稱的登錄。系統會自動將名稱對應至一組 DNS 記錄，因此您可以在程式碼中依名稱來參照服務，然後撰寫 DNS 查詢，使名稱在執行階段解析為服務的端點。您可以在服務的任務定義中指定運作狀態檢查條件，Amazon ECS 會確保服務查詢只會傳回運作狀態良好的服務端點。

此外，您還可將統一服務探索運用於 Kubernetes 管理的服務。為了啟用此項整合，AWS 協力完成外部 DNS 專案，也就是一個 Kubernetes 育成中心專案。

另一個選項是運用 AWS Cloud Map 的功能。AWS Cloud Map 擴展了 Auto Naming API 的功能：為資源 (例如，網際網路通訊協定 (IP)、統一資源定位器 (URL) 和 Amazon Resource Name (ARN) 提供

服務登錄，並將更快變更傳播功能和使用屬性以縮小探索資源集合的能力，提供給 API 型服務探索機制。現有 Route 53 Auto Naming 資源會自動升級為 AWS Cloud Map。

第三方軟體

建置服務探索功能的另一個方法，是使用第三方軟體，例如 HashiCorp Consul、etcd 或 Netflix Eureka。這三個例子全都是分散式、可靠的機碼值存放區。HashiCorp Consul 使用了 AWS Quick Start 來設置靈活、可擴展的 AWS 雲端環境，並使用您所選擇的組態，自動啟動 HashiCorp Consul。

服務網格

在進階微型服務架構中，實際的應用程式可包含數百個、甚至數千個服務。應用程式最為複雜的部分通常不是服務本身，而是這些服務之間的通訊。服務網格是處理服務間通訊的額外分層，負責監控和控制微型服務架構中的流量。這可讓任務 (如服務探索) 得以由此層完全處理。

一般來說，一個服務網格會分為一個資料平面和控制平面。資料平面由一組智慧型代理所組成，這些智慧型代理與應用程式程式碼一起部署為特殊補充代理，用於攔截微型服務間的網路通訊。控制平面負責與代理進行通訊。

服務網格是公開透明的，也就是說，應用程式開發人員無須了解這個額外分層，也不需要變更現有應用程式程式碼。AWS App Mesh 是提供應用程式層級聯網的服務網格，可讓您的服務跨多種類型的運算基礎設施，彼此進行通訊。App Mesh 會將您的服務通訊方式標準化，為您提供完整的能見度，並確保您應用程式的高可用性。

您可使用 AWS App Mesh 搭配在 AWS Fargate、Amazon ECS、Amazon EKS 和 AWS 上自我管理型 Kubernetes 上執行的現有或新的微型服務。App Mesh 可以單一應用程式的形式，針對跨叢集執行的微型服務、協同運作系統，或針對 VPC 監控和控制通訊，而不需要變更任何程式碼。

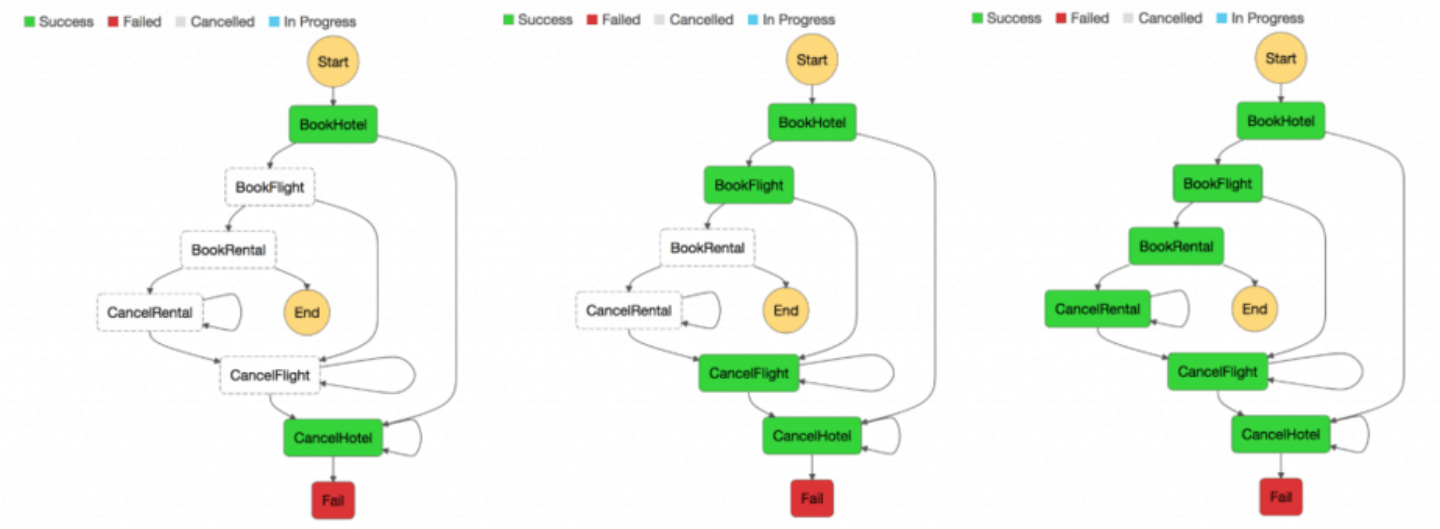
分散式資料管理

整合型應用程式的背後，通常具有大型的關聯式資料庫，此資料庫會定義所有應用程式元件共同的單一資料模型。如果使用微型服務方法，建置分散式和各自獨立元件的目標，會因為此等中央資料庫而無法達成。每個微型服務元件，都應該擁有自己的資料持久層。

不過，分散式資料管理也帶來了新的挑戰。根據 CAP 定理的推論，分散式的微型服務架構，本質上就在一致性與效能之間做了權衡取捨，需要追求最終的一致性。

在分散式系統中，商業交易可能橫跨多個微型服務。由於不能使用單一 ACID 交易，所以可能會產生部分執行的情況。在此情況下，我們將需要某些控制邏輯，來重做已處理的交易。而分散式 Saga 模

式經常用於此目的。發生商業交易失敗時，Saga 會協調一系列的補償性交易，以復原先前交易所做的變更。AWS Step Functions 可讓您輕鬆實作 Saga 執行協調器，如下圖所示。



Saga 執行協調器

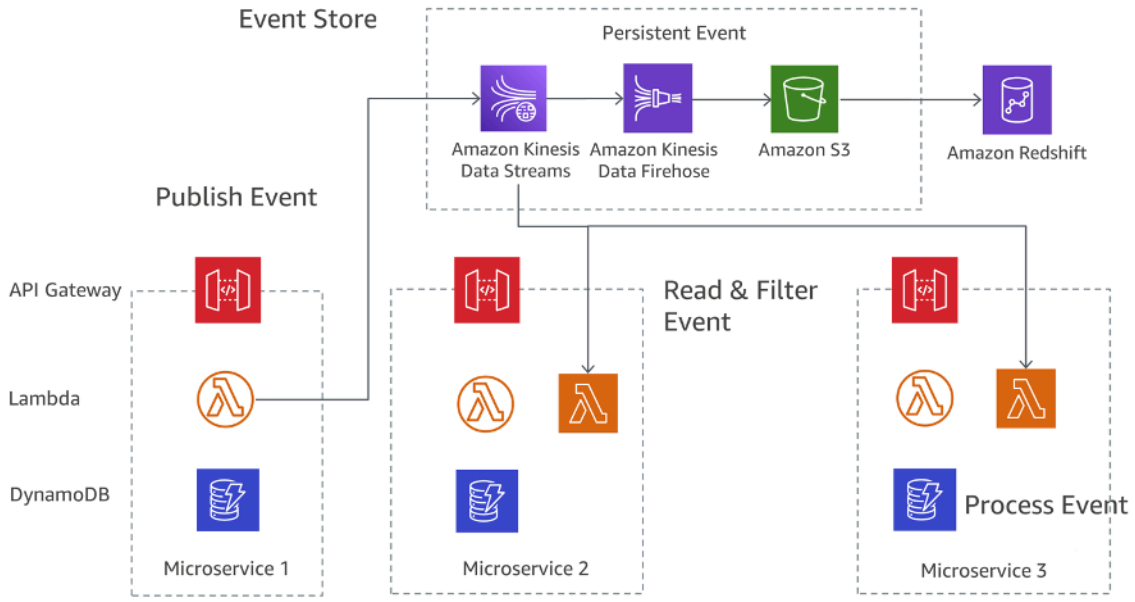
使用核心資料管理工具和程序，來策劃建置重要參考資料的集中存放區，這為微型服務提供了方法，將重要的資料進行同步，亦可回復狀態。使用 Lambda 搭配排程的 Amazon CloudWatch 事件，可讓您建置簡單的清除與重複資料刪除機制。

狀態的變更，經常會影響不只一個微型服務。在這些案例中，事件來源已經過驗證為實用的模式。事件來源的核心概念，是將每次應用程式的變更，以事件記錄的方式呈現和保存。事件來源並未保存應用程式的狀態，而是將資料儲存為事件的串流。資料庫的交易記錄與版本控制系統，是廣為人知的兩個事件來源範例。事件來源具有幾項優點：可以判斷和重建任何時間點的狀態。這自然地就會產生持久的稽核軌跡，也有助於進行除錯。

在微型服務架構的脈絡中，事件來源可利用發佈/訂閱模式，將應用程式的不同部分解耦，並且將相同的事件資料，饋送給不同微型服務的不同資料模型。事件來源經常搭配 命令、查詢、權責、隔離 (CQRS) 模式使用，將讀取與寫入工作負載分開，並同時針對讀取與寫入作業，進行效能、可擴展性和安全性的最佳化。在傳統的資料管理系統中，會對同一個資料儲存庫執行指令與查詢。

下圖顯示如何在 AWS 上實作事件來源模式。Amazon Kinesis Data Streams 會做為集中事件存放區的主要元件，將應用程式的變更擷取為事件，並將這些事件保存於 Amazon S3 中。該圖顯示三種不同的微型服務，這些服務由 Amazon API Gateway、AWS Lambda 和 Amazon DynamoDB 組成。箭頭代表事件的流程：當微型服務 1 發生事件狀態變更時，會藉由將訊息寫入 Kinesis Data Streams，來發佈事件。所有的微型服務，都會在 AWS Lambda 中，執行自己的 Kinesis Data Streams 應用程式，來讀取訊息的副本、根據微型服務的相關性篩選此副本，也可能轉傳此副本，以進行進一步的處理。如果您的函式傳回錯誤，Lambda 會不斷重試批次直到處理成功或資料過期。若要避免停滯的碎片，您

可設定事件來源映射以較小的批次大小進行重試、限制重試次數，或捨棄太舊的記錄。若要保留已捨棄的事件，您可設定事件來源映射將失敗批次的相關詳細資訊傳送至 Amazon Simple Queue Service (Amazon SQS) 佇列或 Amazon Simple Notification Service (Amazon SNS) 主題。



AWS 上的事件來源模式

Amazon S3 可以將所有的事件，持久地儲存到所有的微型服務，在除錯、回復應用程式狀態或進行應用程式變更的稽核時，也是真確事實的單一來源。有兩個主要原因可能會導致多次將記錄交付至您的 Kinesis Data Streams 應用程式：生產者重試和消費者重試。您的應用程式必須預料並妥善因應多次處理個別記錄的問題。

組態管理

在具有數十種不同服務的典型微型服務架構中，每個服務皆需要存取數個下游服務和基礎設施元件，其會對服務公開資料。範例可為訊息佇列、資料庫和其他微型服務。其中一個關鍵挑戰是以一致的方式設定每個服務，以提供有關連線至下游服務和基礎設施的資訊。此外，組態還應包含服務運行環境的相關資訊，且無須重新啟動應用程式以使用新的組態資料。

12 要素應用程式的第三個原則涵蓋了這個主題：「環境變數中的 12 要素應用程式商店配置 (通常縮寫為 env vars 或 env)。」對於 Amazon ECS，環境變數可使用環境容器定義參數映射至 docker run 的 `--env` 選項，將環境變數傳遞至容器。使用 `environmentFiles` 容器定義參數列出內含環境變數的一個或多個檔案，可將環境變數批量傳遞至您的容器。檔案必須託管於 Amazon S3 中。於 AWS Lambda 中，執行階段可讓您的程式碼使用環境變數，並設定其他環境變數，這些變數包含函數和叫用

請求的相關資訊。對於 Amazon EKS，您可於相應 pod 之組態資訊清單的 env 欄位中定義環境變數。使用 env-variables 的另一種方法是使用 ConfigMap。

非同步通訊與輕量型傳訊

在傳統的整合型應用程式中，通訊相當地簡單：應用程式的某個部分，使用方法呼叫或內部事件發送機制，以與其他的部分進行通訊。如果使用拆分去耦的微型服務，來建置相同的應用程式，則應用程式不同部分之間的通訊，就必須使用網路通訊來進行。

REST 型通訊

在建置微型服務之間的同步通訊時，HTTP/S 通訊協定是最熱門的方式。在大部分情況中，RESTful API 使用 HTTP 做為傳輸層。REST 架構模式使用無狀態通訊、統一的界面與標準方法。

利用 API Gateway，即可建立做為前門使用的 API，以供應用程式存取資料、商業邏輯或後端服務的功能。API 開發人員可以建立 API，進而存取 AWS 或其他 Web 服務，以及 AWS 雲端中所存放的資料。使用 API Gateway 服務所定義的 API 物件，是一組資源和方法。

資源是 API 網域內的型別物件，而且可能有相關聯的資料模型，或是和其他資源具有關係。每個資源皆可設定為回應一個或多個標準方法，也就是標準的 HTTP 動詞/方法，例如 GET、POST 或 PUT。REST API 可部署至不同的階段、進行版本控制，也可複製至新的版本。

涉及接受和處理多達數十萬個並行 API 呼叫 (包括流量管理、授權與存取控制、監控和 API 版本管理) 的所有任務均由 API Gateway 處理。

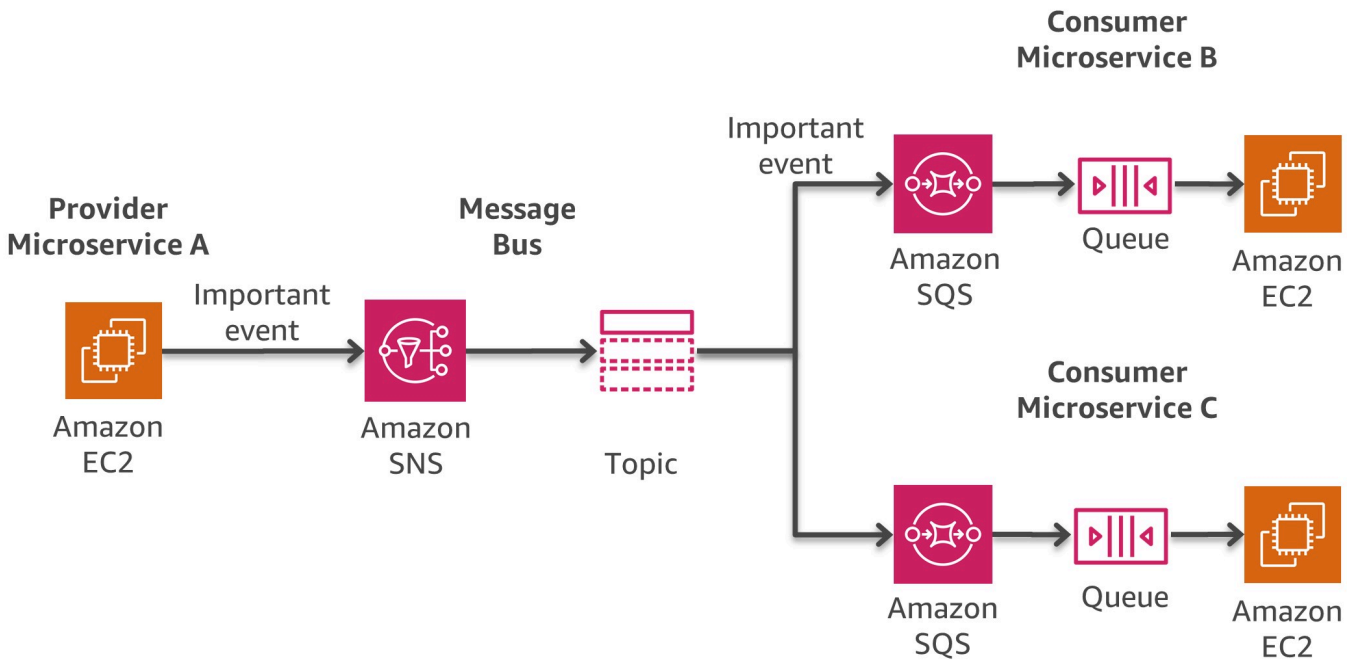
非同步傳訊和事件傳遞

訊息傳遞為用來建置微型服務間之通訊功能的另一個模式。服務之間會透過佇列交換訊息，以進行通訊。這種溝通模式的一個主要優點，就是無須具有服務探索功能，且服務是鬆散耦合的。

同步系統會緊密耦合，也就是說，某個同步下游相依項中的問題，會立即影響到上游呼叫者。上游呼叫者的重試動作會快速擴散並擴大問題。

AWS 會根據特定需求 (如通訊協定)，提供不同的服務來協助實作此模式。其中一項可能的實作是使用 Amazon Simple Queue Service (Amazon SQS) 佇列或 Amazon Simple Notification Service (Amazon SNS) 的組合。

這兩種服務皆緊密地協同運作：Amazon SNS 可讓應用程式透過「推送」機制，向多個訂閱者傳送訊息。透過同時使用 Amazon SNS 與 Amazon SQS，您可以將一個訊息傳送給多個使用者。下圖將會示範 Amazon SNS 和 Amazon SQS 的整合。



AWS 上的訊息匯流排模式

如果訂閱 SNS 主題的 Amazon SQS 佇列，您可以向該主題發佈訊息，Amazon SNS 接著就將訊息傳送至訂閱的 Amazon SQS 佇列。包含主旨與訊息的訊息，會和 JSON 格式的中繼資料資訊，一起發佈到該主題。

使用跨內部應用程式、第三方 SaaS 應用程式和 AWS 服務的事件來源建置事件驅動架構的另一個選項是 Amazon EventBridge。作為全受管事件匯流排服務，EventBridge 接收來自不同來源的事件，根據路由規則識別目標，並對該目標 (包括 AWS Lambda、Amazon SNS 和 Amazon Kinesis Streams 等) 提供幾近即時的資料。還可在交付前使用輸入變換器自訂入站事件。

若要更快速地開發事件驅動應用程式，EventBridge 結構描述登錄檔會收集並組織結構描述，包括 AWS 服務所產生的所有事件結構描述。客戶還可定義自訂的結構描述，或使用推斷結構描述選項自動發現結構描述。然而，總的來說，所有這些功能的潛在權衡取捨是相對較高的 EventBridge 交付延遲值。此外，EventBridge 的預設輸送量和配額可能需要根據使用案例透過支援請求加以提升。

不同的實作策略是採用 Amazon MQ，若現有軟體使用開放標準 API 和通訊協定進行傳訊 (包括 JMS、NMS、AMQP、STOMP、MQTT 和 WebSocket)，便可使用 Amazon MQ。Amazon SQS 會公開自訂 API，這意味著，例如，假設您想將現有的應用程式從內部部署環境移轉到 AWS，您將需要變更程式碼。在許多情況下，使用 Amazon MQ 就不需要進行這項作業。

Amazon MQ 會進行熱門開放原始碼訊息代理程式 ActiveMQ 的管理和維護。系統會針對高可用性和訊息耐久性而自動佈建基礎設施，以支援您應用程式的可靠性。

協同運作與狀態管理

在統籌工作流程時，若牽涉到多個微型服務，則微型服務的分散式特性，會使得這項任務充滿挑戰。開發人員可能會嘗試將協同運作用的程式碼，直接加到服務中。這種做法應該避免，因其會導致更緊密的耦合，而讓快速替換個別服務變得更加困難。

您可使用 AWS Step Functions，利用執行不同功能的個別元件，來建置應用程式。Step Functions 提供狀態機器，此機器會隱藏服務協同運作的複雜性，例如錯誤處理、序列化和並行化。這可讓您快速地擴展和變更應用程式，同時避免服務中加入額外的統籌程式碼。

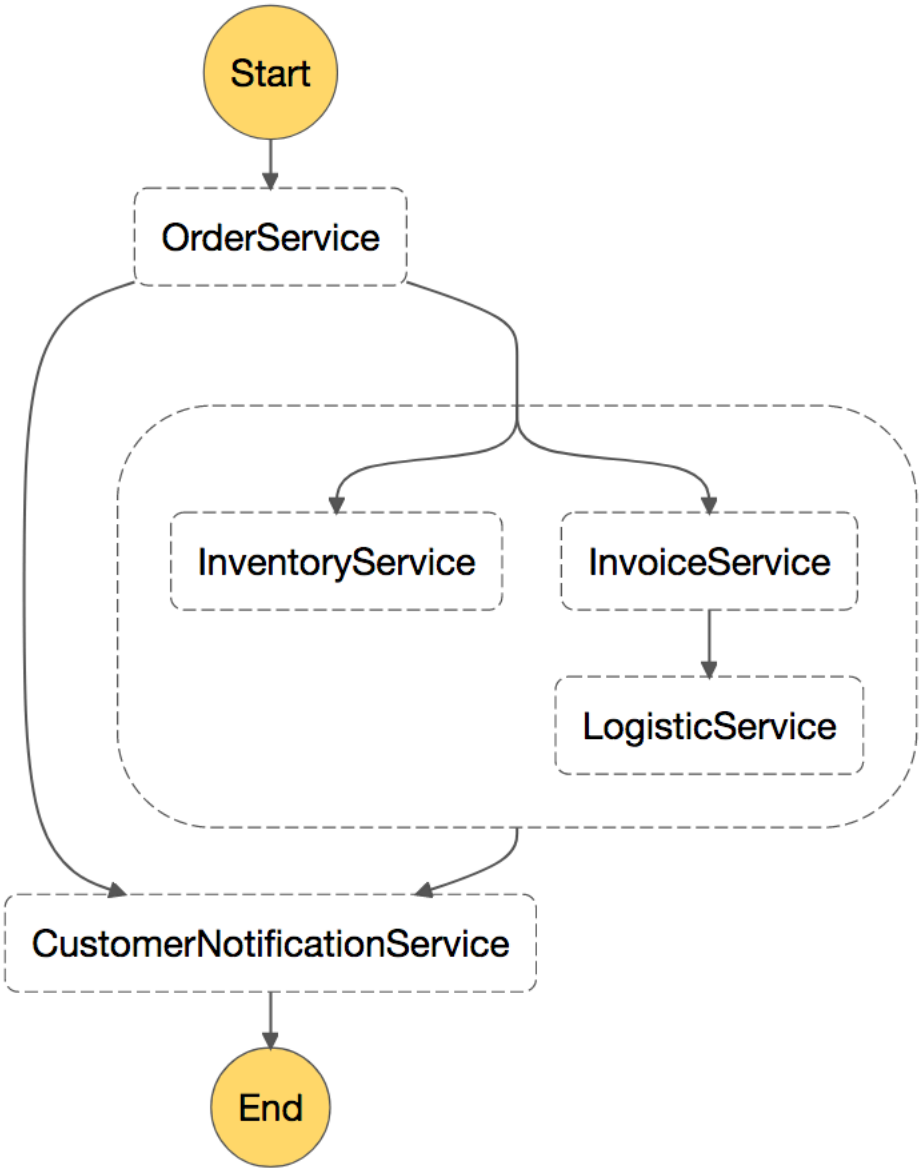
Step Functions 是可靠的方式，可用來統籌元件，並逐步執行應用程式的函數。Step Functions 提供圖形化的主控台，可排列應用程式的元件，並將這些元件以視覺化的方式呈現為一連串的步驟，讓您輕鬆地建置和執行分散式服務。

Step Functions 會自動啟動且追蹤每個步驟，並在發生錯誤時重試，讓您的應用程式可如預期依序執行。Step Functions 會記錄每個步驟的狀態，如果有哪個環節出了差錯，您就能迅速診斷並解決問題。您甚至不需要撰寫程式碼，即可變更和新增步驟，來讓您的應用程式進化，更快速地進行創新。

Step Functions 是 AWS 無伺服器平台的一部分，可根據運算資源 (例如 Amazon EC2、Amazon EKS 和 Amazon ECS)，以及 Amazon SageMaker 和 AWS Glue 等其他服務，來協同運作 Lambda 函數與應用程式。Step Functions 會為您管理運作和底層的基礎設施，以協助確保您的應用程式無論規模皆可使用。

如要建置工作流程，Step Functions 會使用 Amazon 狀態語言。工作流程可以包含連續或並行的步驟，和分支的步驟。

下圖顯示微型服務架構的範例工作流程，此流程結合了連續和並行的步驟。此種工作流程也可以透過 Step Functions API 或 API Gateway 來叫用。



使用 Step Functions 叫用的微型服務工作流程範例

分散式監控

微型服務架構包含分散的許多不同部分，這些部分必須受到監控。您可以使用 Amazon CloudWatch 來收集與追蹤指標、集中和監控日誌檔、設定警示，以及自動對 AWS 環境的變更做出反應。CloudWatch 可以監控各種 AWS 資源，例如 Amazon EC2 執行個體、DynamoDB 資料表、Amazon RDS 資料庫執行個體，也可監控應用程式和服務產生的自訂指標以及應用程式產生的所有日誌檔。

監控

您可以利用 CloudWatch 來全面了解整個系統的資源使用率、應用程式效能和運作狀態。CloudWatch 提供可靠、可擴展的彈性監控解決方案，可讓您在短時間內立即上手使用。您再也不需要設定、管理和擴展自己的監控系統與基礎設施。在微型服務架構中，使用 CloudWatch 來監控自訂指標的功能，提供了額外的效益，因為開發人員可以決定應從每個服務收集哪些指標。除此之外，還可根據自訂指標來進行動態擴展。

除了 Amazon CloudWatch 之外，您還可使用 CloudWatch 容器洞察來收集、彙總和總結容器化的應用程式和微型服務中的指標和日誌。CloudWatch 容器洞察會自動收集多種資源 (如 CPU、記憶體、磁碟和網路) 的指標，並在叢集、節點、Pod、任務和服務層級彙總為 CloudWatch 指標。使用 CloudWatch 容器洞察，您可存取 CloudWatch 容器洞察儀表板指標。其還提供診斷資訊，例如容器重新啟動故障，協助您快速隔離和解決這些故障。您也可以為容器洞察收集的指標設定 CloudWatch 警示。

容器洞察可用於 Amazon ECS、Amazon EKS 和 Amazon EC2 上的 Kubernetes 平台。Amazon ECS 支援包含對 Fargate 的支援。

另一個熱門選項，尤其是針對 Amazon EKS，是使用 Prometheus。Prometheus 是開放原始碼監控和警示工具組，常用於結合 Grafana 來視覺化收集的指標。許多 Kubernetes 元件會將指標儲存於 `/metrics` 中，而 Prometheus 會定期消除這些指標。

Amazon Managed Service for Prometheus (AMP) 為 Prometheus 相容監控服務，可讓您大規模監控容器化應用程式。透過 AMP，您可使用開放原始碼 Prometheus 查詢語言 (PromQL) 來監控容器化工作負載的效能，無需針對管理擷取、儲存和查詢操作指標所需的基礎設施進行管理。您可以從 Amazon EKS 和 Amazon ECS 環境中，使用 AWS Distro for OpenTelemetry 或 Prometheus 伺服器作為收集代理程式，來收集 Prometheus 指標。

AMP 通常與 Amazon Managed Service for Grafana (AMG) 結合使用。無論指標存放於何處，AMG 皆可輕鬆查詢、視覺化、提醒和了解指標。您可以透過 AMG，分析指標、日誌和追蹤，無須佈建伺服器、設定和更新軟體，或在生產環境中進行保護和擴展 Grafana 所涉及的繁重工作。

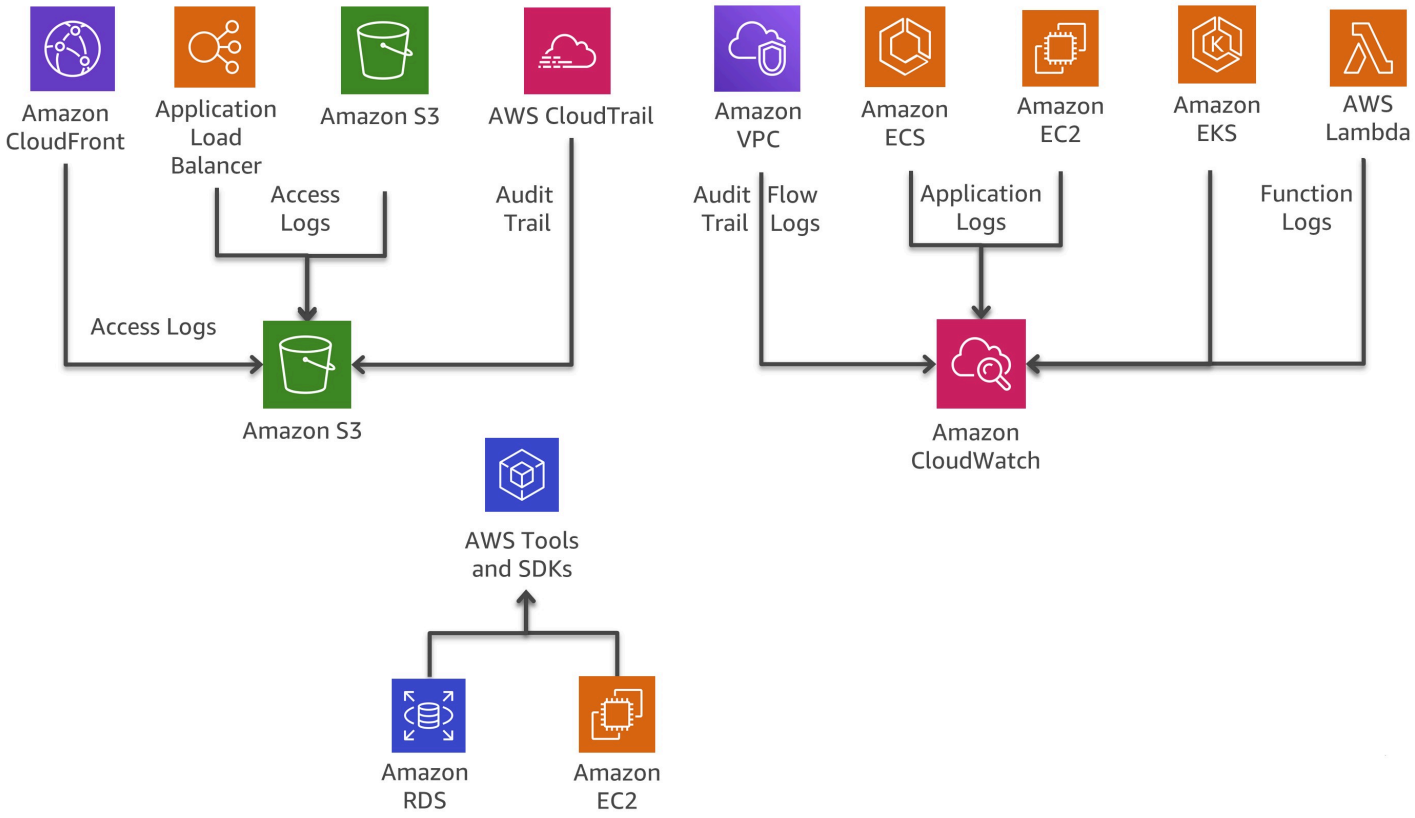
集中日誌

一致的記錄對故障診斷和找出問題至關重要。微型服務可讓團隊推出比以往更多的版本，並鼓勵工程團隊在生產環境中實驗新的功能。若要逐步改進應用程式，了解對客戶的影響極為重要。

依預設，大多數的 AWS 服務都會將日誌檔集中。AWS 上日誌檔的主要目的地是 Amazon S3 和 Amazon CloudWatch Logs。若為在 Amazon EC2 執行個體上執行的應用程式，可使用常駐程式，將

日誌檔傳送至 CloudWatch Logs。Lambda 函數原本就會將其日誌輸出傳送到 CloudWatch Logs，而 Amazon ECS 也包含對 awslogs 日誌驅動程式的支援功能，可將容器日誌集中至 CloudWatch Logs。對 Amazon EKS 來說，Fluent Bit 或 Fluentd皆可將叢集內個別執行個體中的日誌轉送至集中式日誌 CloudWatch Logs，在此處會使用 Amazon OpenSearch Service 和 Kibana 將日誌合併，以進行更高階層的報告。由於其較小的佔用空間和效能優勢，因此推薦使用 Fluent Bit 來代替 FluentD。

下圖顯示某些服務的記錄功能。然後，團隊即可使用 Amazon OpenSearch Service 和 Kibana 等工具，來搜尋和分析這些日誌。Amazon Athena 可用來對 Amazon S3 中的集中日誌檔，進行一次性的查詢。



AWS 服務的記錄功能

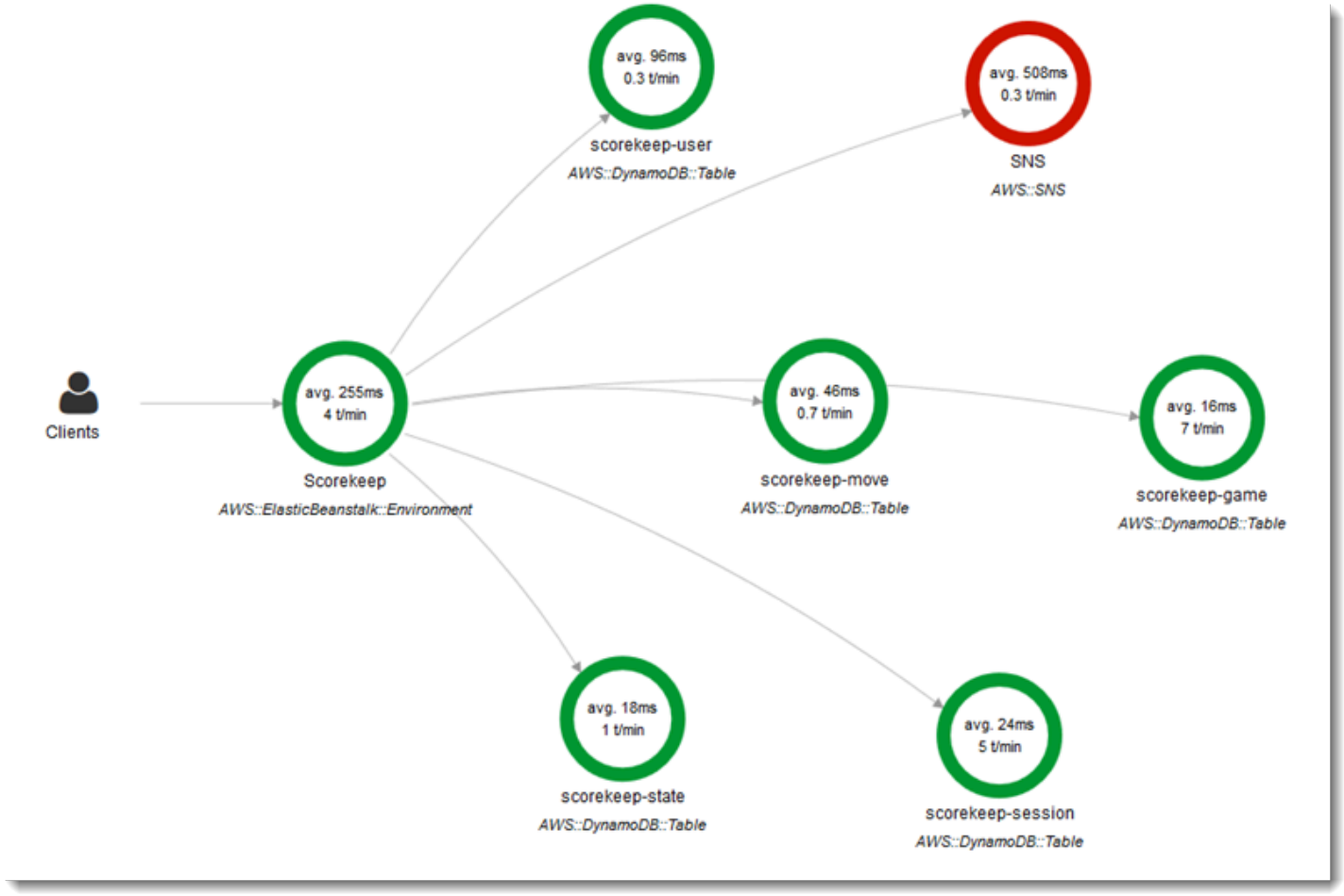
分散式追蹤

在許多情況中，一組微型服務會協同運作，來處理請求。試想有一個複雜的系統，其中包含數十個微型服務，而在呼叫鏈中，這些服務的其中一個發生了錯誤。即使正確地記錄了每個微型服務，並且將日誌整合於集中的系統，想要找到相關的所有日誌訊息，也可能很困難。

AWS X-Ray 的中心概念，是使用相互關聯的 ID，與特定事件鏈相關的所有請求和訊息，都會附加此一獨特的識別符。當請求命中第一個內建 X-Ray 的服務 (例如，Application Load Balancer 或 API

Gateway) 時，追蹤 ID 會新增至 HTTP 請求的特定追蹤標頭中 (此標頭名稱為 `X-Amzn-Trace-Id`)，並加入回應之中。經由 X-Ray SDK，任何微型服務皆可讀取，但也可新增或更新此標頭。

X-Ray 可搭配 Amazon EC2、Amazon ECS、Lambda 和 AWS Elastic Beanstalk 使用。您可以使用 X-Ray，搭配這些服務上部署的應用程式，這些應用程式使用 Java、Node.js 和 .NET 撰寫。



AWS X-Ray 服務地圖

Epsagon 為全受管 SaaS，其包含對所有 AWS 服務、第三方 API (透過 HTTP 呼叫) 及其他常見服務 (如 Redis、Kafka 和 Elastic) 進行追蹤。Epsagon 服務包括監控功能、對最常用服務的提醒，及對程式碼進行每次呼叫的酬載可視性。

AWS Distro for OpenTelemetry 是安全、生產就緒型、AWS 支援的 OpenTelemetry 專案發行版本。作為 Cloud Native Computing Foundation 的一員，AWS Distro for OpenTelemetry 會提供開放原始碼 API、程式庫和代理程式來收集分散的追蹤和指標，以執行應用程式監控。使用 AWS Distro for OpenTelemetry，您只需檢測應用程式一次，即可將相關指標和追蹤傳送至多個 AWS 和合作夥伴監控解決方案。使用自動檢測代理程式來收集追蹤，無須變更您的程式碼。AWS Distro for OpenTelemetry

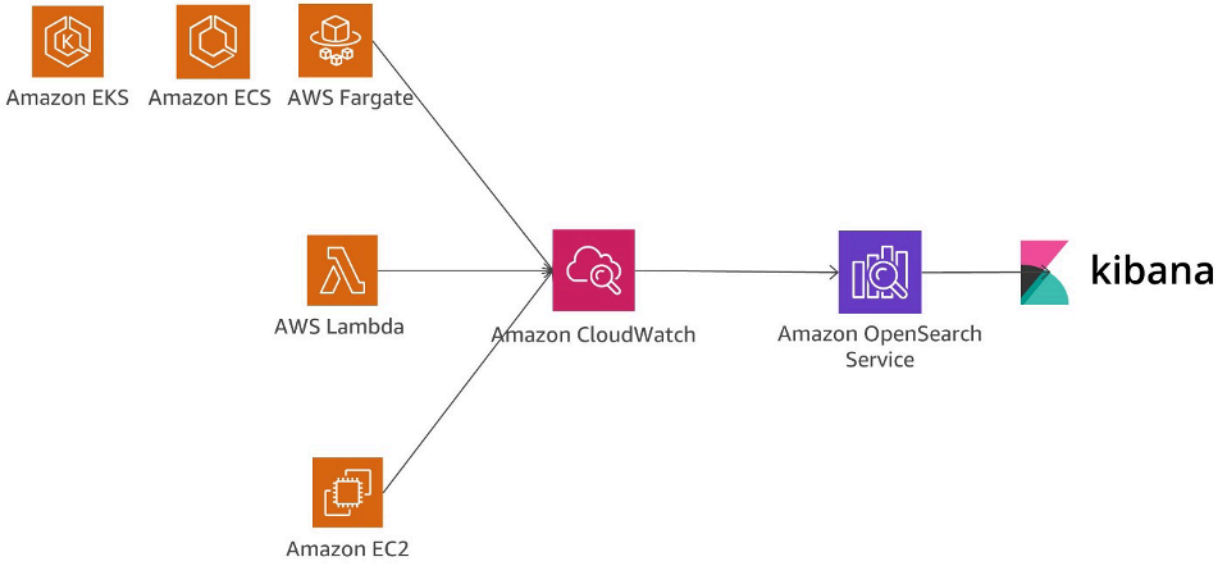
也會收集您 AWS 資源和受管服務的中繼資料，以將應用程式效能資料與基礎設施資料相互關聯，減少解決問題所需的平均時間。使用 AWS Distro for OpenTelemetry 來檢測在以下程式上執行或內部部署的應用程式：Amazon EC2、Amazon ECS、Amazon EKS on Amazon EC2 上的 Amazon EKS、Fargate 和 AWS Lambda。

AWS 上的日誌分析選項

搜尋、分析日誌資料，並以視覺化的方式呈現，這是了解分散式系統的一個重要面相。Amazon CloudWatch Logs Insights 可讓您即時探索、分析和視覺化日誌。這可讓您針對運作問題進行疑難排解。分析日誌檔的另一個選項，是使用 Amazon OpenSearch Service 搭配 Kibana。

Amazon OpenSearch Service 可用來進行全文字搜尋、結構化搜尋、分析，和這三種作業的組合。Kibana 是一項開放原始碼的資料視覺化外掛程式，適用於與 Kibana 無縫整合的 Amazon OpenSearch Service。

下圖示範使用 Amazon OpenSearch Service 和 Kibana 進行日誌分析。CloudWatch Logs 可設定透過 CloudWatch Logs 訂閱，將日誌記錄以接近即時的速度，串流到 Amazon OpenSearch Service。Kibana 會以視覺化的方式顯示資料，並針對 Amazon OpenSearch Service 中的資料存放區，呈現便利的搜尋介面。這項解決方案可搭配 ElastAlert 等軟體使用，來建置警示系統，於偵測到資料中存在異常、突峰或其他關切的模式時，傳送 SNS 通知、電子郵件和建立 JIRA 問題工作單。



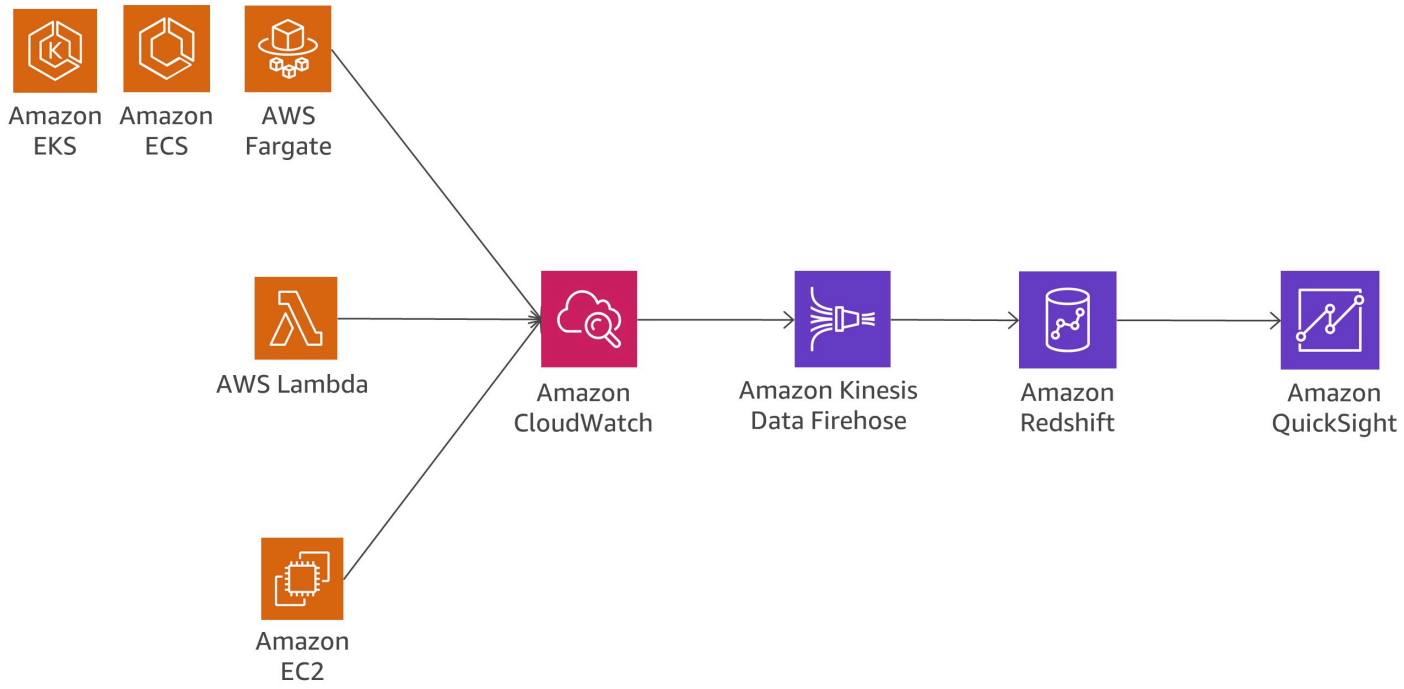
使用 Amazon OpenSearch Service 和 Kibana 來進行日誌分析

分析日誌檔的另一個選項，是使用 Amazon Redshift 搭配 Amazon QuickSight。

您可輕鬆地將 QuickSight 連線至 AWS 資料服務，包括 Amazon Redshift、Amazon RDS、Amazon Aurora、Amazon EMR、DynamoDB、Amazon S3 與 Amazon Kinesis。

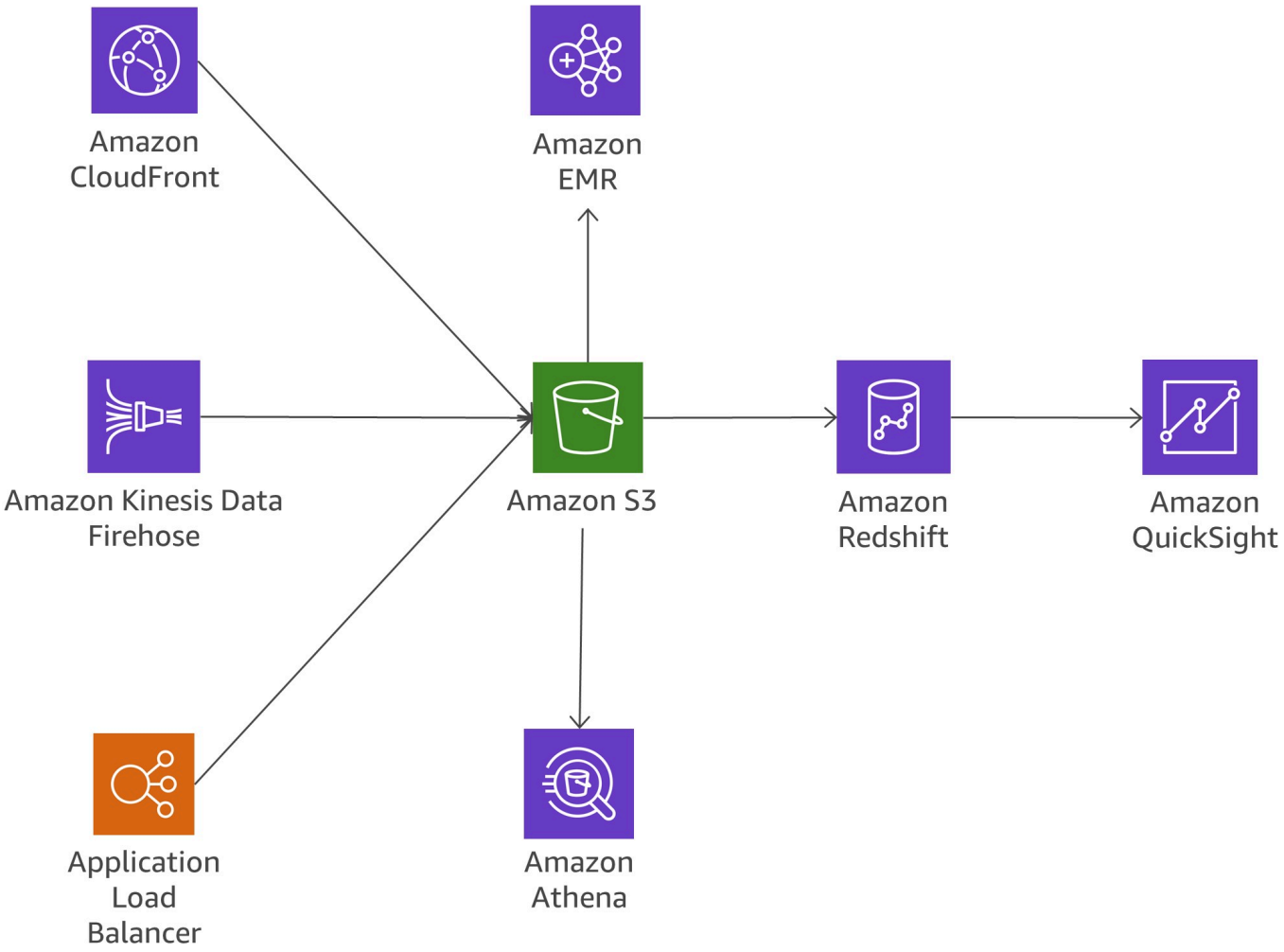
CloudWatch Logs 可做為日誌資料的集中存放，除了僅存放資料外，亦可將日誌記錄串流至 Amazon Kinesis Data Firehose。

下圖顯示案例，從不同的來源，使用 CloudWatch Logs 和 Kinesis Data Firehose，將日誌記錄串流至 Amazon Redshift。Amazon QuickSight 會使用 Amazon Redshift 中所儲存的資料，來進行分析、報告和視覺化。



使用 Amazon Redshift 和 Amazon QuickSight 進行日誌分析

下圖顯示 Amazon S3 上的日誌分析情境。如果日誌存放於 Amazon S3 儲存貯體中，日誌資料即可載入不同的 AWS 資料服務 (例如 Amazon Redshift 或 Amazon EMR)，來分析日誌串流所儲存的資料，並找出異常。



Amazon S3 上的日誌分析

繁瑣性

透過將整合型應用程式拆分為較小型的微型服務，通訊開銷會增加，因為微型服務必須彼此互相通訊。在許多實作中，會使用 REST over HTTP，因為這是輕量型通訊協定，但訊息量大時卻可能會導致問題發生。在某些情況下，您可能會考慮將往返傳送大量訊息的服務整合。若您發現自己處於合併更多服務只是為了減少繁瑣的狀態中，則您應審視自己的問題網域和網域模型。

通訊協定

在本白皮書先前的章節 the section called “非同步通訊與輕量型傳訊” 中，對可能使用的不同通訊協定進行說明。使用 HTTP 等簡單的通訊協定，對微型服務而言很常見。服務所交換的訊息，可以用不同

的方式編碼，例如，使用人類可讀的 JSON 或 YAML 等格式，或使用有效率的二進位格式，例如 Avro 或協定緩衝區。

快取

若要減少微型服務架構的延遲和繁瑣性，快取是一項絕佳的方法。取決於實際的使用案例和瓶頸，有可能使用好幾個快取層。在 AWS 上執行的許多微型服務的應用程式，會使用 ElastiCache，透過在本機建立結果的快取，來減少對其他微型服務發出的呼叫量。API Gateway 提供內建的快取層，以減少後端伺服器上的負載。此外，快取對減少資料持久層的負載，也非常有用。所有快取機制的挑戰，是在良好的快取命中率和資料的時效及一致性之間，找到適當的平衡。

稽核

在可能包含數百個分散式服務的微型服務架構中，另一項挑戰是確保所有服務上使用者動作的能見度，及可在組織層級取得所有服務的清楚整體視圖。為了協助強制執行安全政策，請務必對資源的存取和造成系統變更的活動進行稽核。

必須在個別服務層級和更廣泛系統上執行的服務中，進行變更的追蹤。一般來說，微型服務架構中會頻繁發生變更，所以稽核變更更顯重要。本節會檢視 AWS 中的重要服務與功能，可協助對您的微型服務架構進行稽核。

稽核軌跡

AWS CloudTrail 是一項實用的工具，可用來追蹤微型服務中的變更，因為此服務可記錄於 AWS 雲端中發出的所有 API 呼叫，並以即時的速度，將這些呼叫傳送到 CloudWatch Logs，或是在幾分鐘內傳送到 Amazon S3。

所有的使用者動作和自動化系統動作變成可搜尋，也可加以分析，找出未預期的行為，公司政策違規或進行除錯。記錄的資訊包含時間戳記、使用者和帳戶資訊、呼叫的服務、要求的服務動作、呼叫者的 IP 地址，以及請求參數和回應項目。

CloudTrail 可針對同一個帳戶定義多個軌跡，其可讓不同的利害關係人 (例如安全管理員、軟體開發人員或 IT 稽核員) 建立和管理自己的軌跡。如果微型服務團隊擁有不同的 AWS 帳戶，可將軌跡彙總到單一 S3 儲存貯體中。

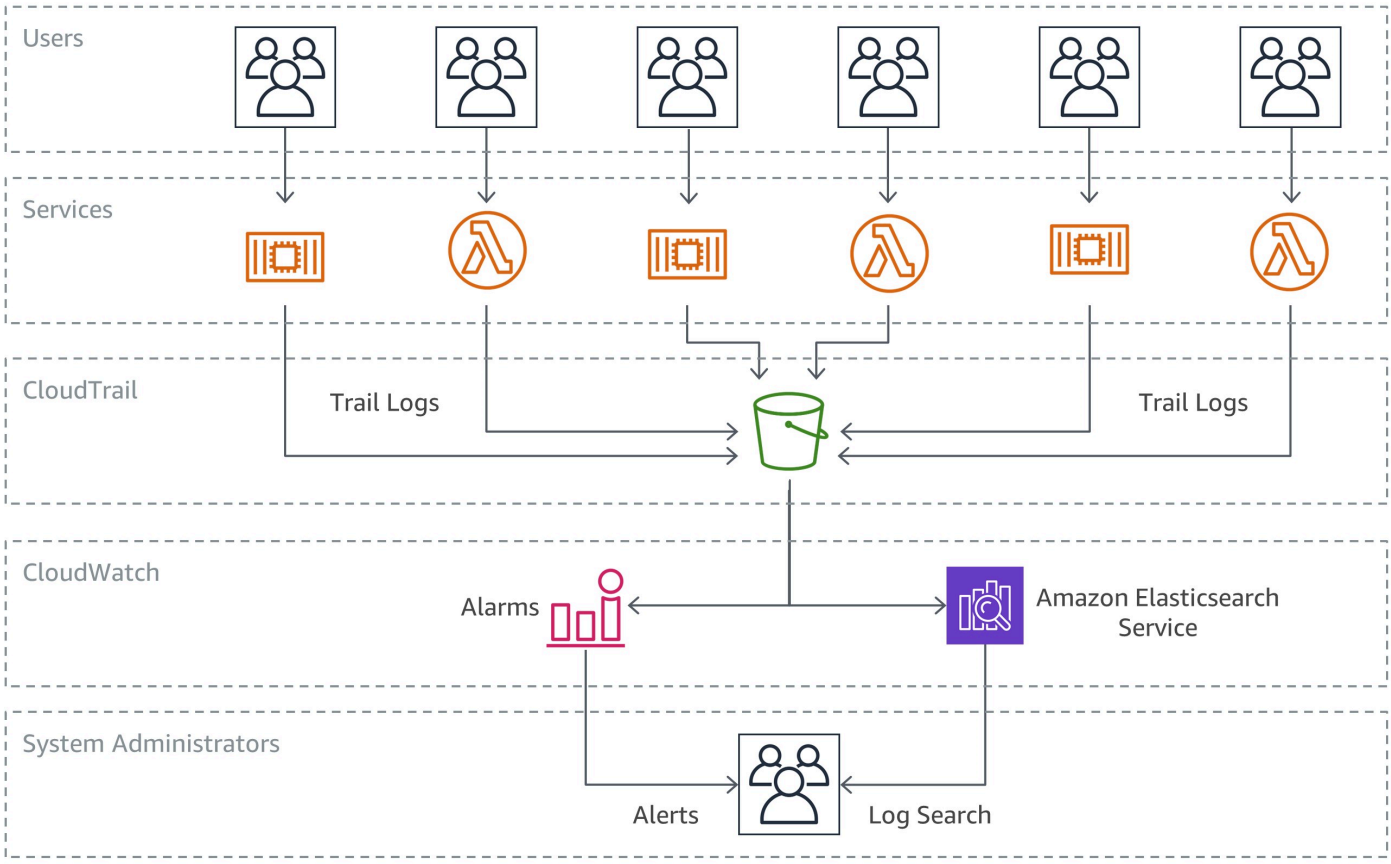
將稽核軌跡儲存於 CloudWatch 中的優點，是即時擷取稽核軌跡資料，並將資訊轉傳到 Amazon OpenSearch Service 以進行搜尋和視覺化呈現。您可設定 CloudTrail，以記錄到 Amazon S3 和 CloudWatch Logs。

事件與即時動作

系統架構中的某些變更必須回應快速，且必須執行矯正情況的動作，或必須遵循特定的管理程序才能授權必須起始的變更。Amazon CloudWatch Events 與 CloudTrail 的整合，可針對所有 AWS 服務上變異的 API 呼叫產生事件。您也可以定義自訂事件，或者根據固定的排程來產生事件。

當事件觸發而且符合所定義的規則時，可立即通知您組織中預先定義的人員組，則其可採取適當動作。如果可以自動執行所需的動作，則規則可以自動觸發內建的工作流程，或叫用 Lambda 函數來解決問題。

下圖顯示 CloudTrail 與 CloudWatch Events 搭配運作的環境，以滿足微型服務架構中的稽核與修復要求。CloudTrail 正在追蹤的所有微型服務和稽核軌跡，會存放在 Amazon S3 儲存貯體中。CloudWatch Events 在營運變更時會查覺到。CloudWatch Events 會回應這些操作變更並視需要進行修正動作，透過傳送訊息以回應環境、啟用功能、執行變更和擷取狀態資訊。CloudWatch Events 在 CloudTrail 之上運作，且會在您的架構出現特定變更時觸發警示。



稽核與修復

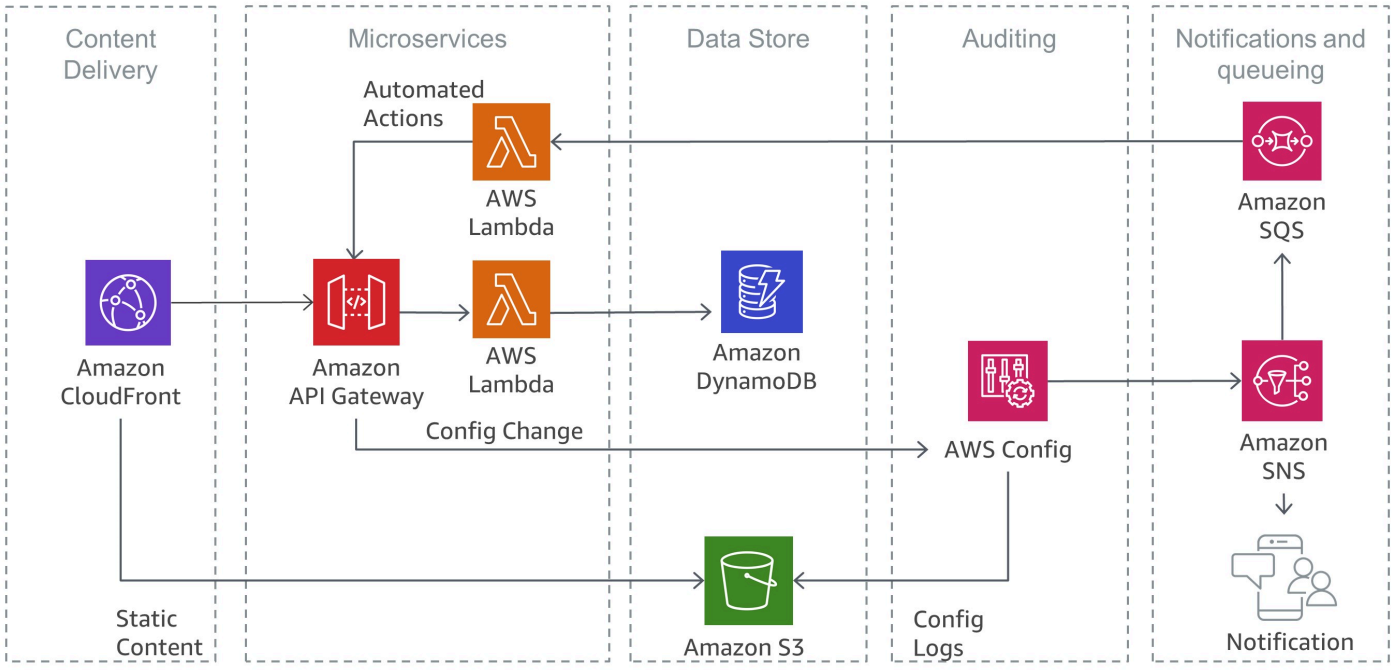
資源清單與變更管理

如要在靈活的開發環境中，對快速變動的基礎設施組態保持控制，使用更高度自動化的受管方法，來稽核和控制您的架構至關重要。

雖然 CloudTrail 和 CloudWatch Events 是追蹤和回應整個微型服務的基礎設施變更的重要基礎，AWS Config 規則可讓公司使用特定的規則，來定義安全政策，以自動偵測、追蹤和警示政策的違規。

下一個範例示範如何可針對您微型服務架構中的未合規組態變更，進行偵測、通知和自動回應。開發團隊的成員已針對微型服務變更了 API Gateway，來允許端點接受入站 HTTP 流量，而不是僅允許 HTTPS 請求。

由於之前組織已將這種情況列為安全性合規問題，因此 AWS Config 規則早已監控這個情況。此規則會識別視為安全違規的變更和執行兩種動作：針對 Amazon S3 儲存貯體中偵測到的變更建立日誌 (適用於稽核)，以及建立 SNS 通知。在我們的案例中，Amazon SNS 有兩種用途：傳送電子郵件給指定的群組，以通知安全違規的相關事宜，及新增訊息到 SQS 佇列。接著會擷取訊息，並透過變更 API Gateway 組態，來還原合規狀態。



使用 AWS Config 偵測安全違規

結論

微型服務架構是一種分散式的設計方法，旨在克服傳統整合型架構的限制。微型服務有助於擴展應用程式和組織，同時縮短週期時間。不過，隨之而來的數個挑戰，也可能增加額外的架構複雜性和運作負擔。

AWS 提供龐大的受管服務產品組合，可協助產品團隊建置微型服務架構，將架構和操作複雜性降到最低。本白皮書會說明相關的 AWS 服務，和如何使用 AWS 服務，建置原生的典型模式，例如服務探索或事件來源。

資源

- AWS 架構中心
- AWS 白皮書
- AWS 每月架構
- AWS 架構部落格
- This Is My Architecture 影片
- AWS Answers
- AWS 文件

文件歷史記錄和貢獻者

文件歷史記錄

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	整合 Amazon EventBridge、AWS OpenTelemetry、AMP、AMG、容器洞察、小幅度文字變更。	2021 年 11 月 9 日
小幅度更新	調整過的頁面配置	2021 年 4 月 30 日
小幅度更新	小幅度的文字變更。	2019 年 8 月 1 日
白皮書已更新	整合 Amazon EKS、AWS Fargate、Amazon MQ、AWS PrivateLink、AWS App Mesh、AWS Cloud Map	2019 年 6 月 1 日
白皮書已更新	整合 AWS Step Functions，AWS X-Ray 與 ECS 事件資料流。	2017 年 9 月 1 日
初次出版	已發佈了實作 AWS 上的微型服務	2016 年 12 月 1 日

> **Note**
>
> 若要訂閱 RSS 更新，您必須為正在使用的瀏覽器啟用 RSS 外掛程式。

作者群

協力完成本文件的個人與組織如下：

- Sascha Möllering，AWS 解決方案架構
- Christian Müller，AWS 解決方案架構
- Matthias Jung，AWS 解決方案架構
- Peter Dalbhanjan，AWS 解決方案架構
- Peter Chapman，AWS 解決方案架構
- Christoph Kassen，AWS 解決方案架構
- Umair Ishaq，AWS 解決方案架構
- Rajiv Kumar，AWS 解決方案架構

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。