歡迎學習“模組 9：實施彈性、高可用性和監控。”

本模組包含以下章節：

1. 架構需求
2. 擴縮計算資源
3. 擴縮數據庫
4. 設計高度可用的環境
5. 監控

本模組還包括：

- 演示如何為 Amazon EC2 Auto Scaling 創建目標跟蹤和步進擴縮策略
- 演示如何使用 Application Load Balancer 部署高度可用的 Web 應用程式
- 演示 Amazon Route 53
- 指導實驗：創建高度可用的環境
- 挑戰實驗：為咖啡館創建可擴展且高度可用的環境

最後，您需要完成一個知識測驗，以測試您對本模組中涵蓋的關鍵概念的理解程度。

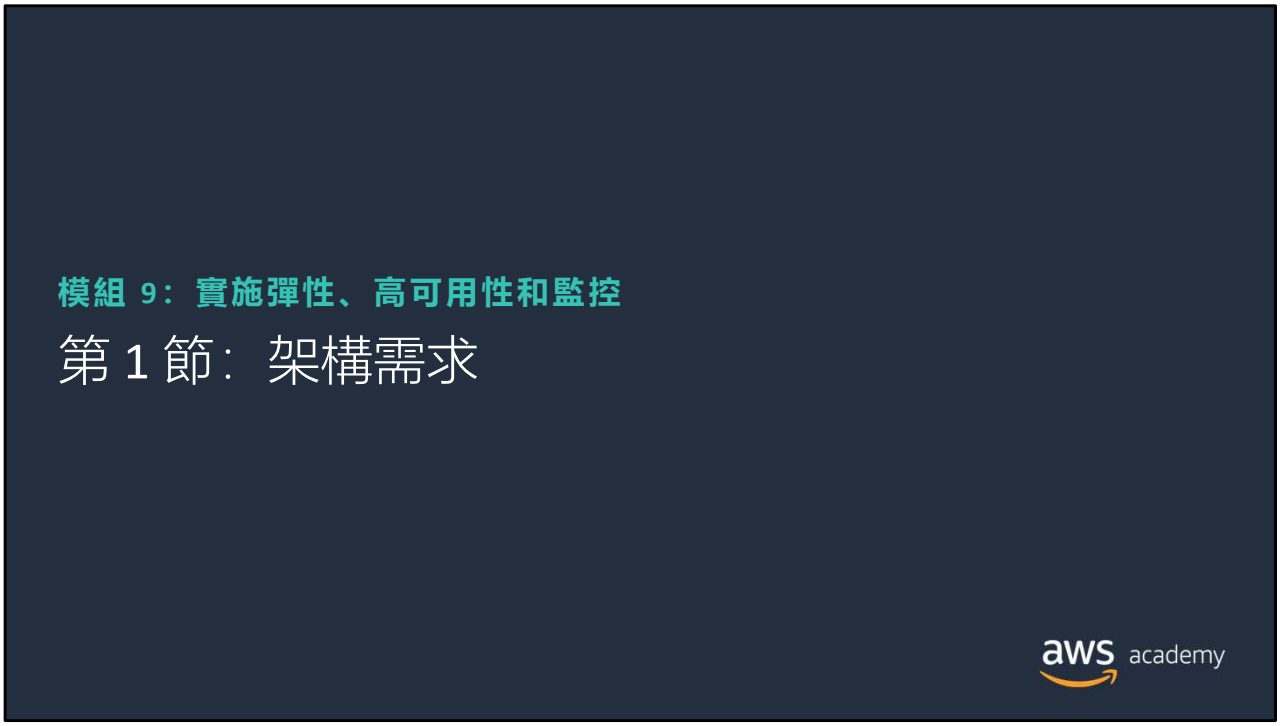
模組目標

學完本模組後，您應該能夠：

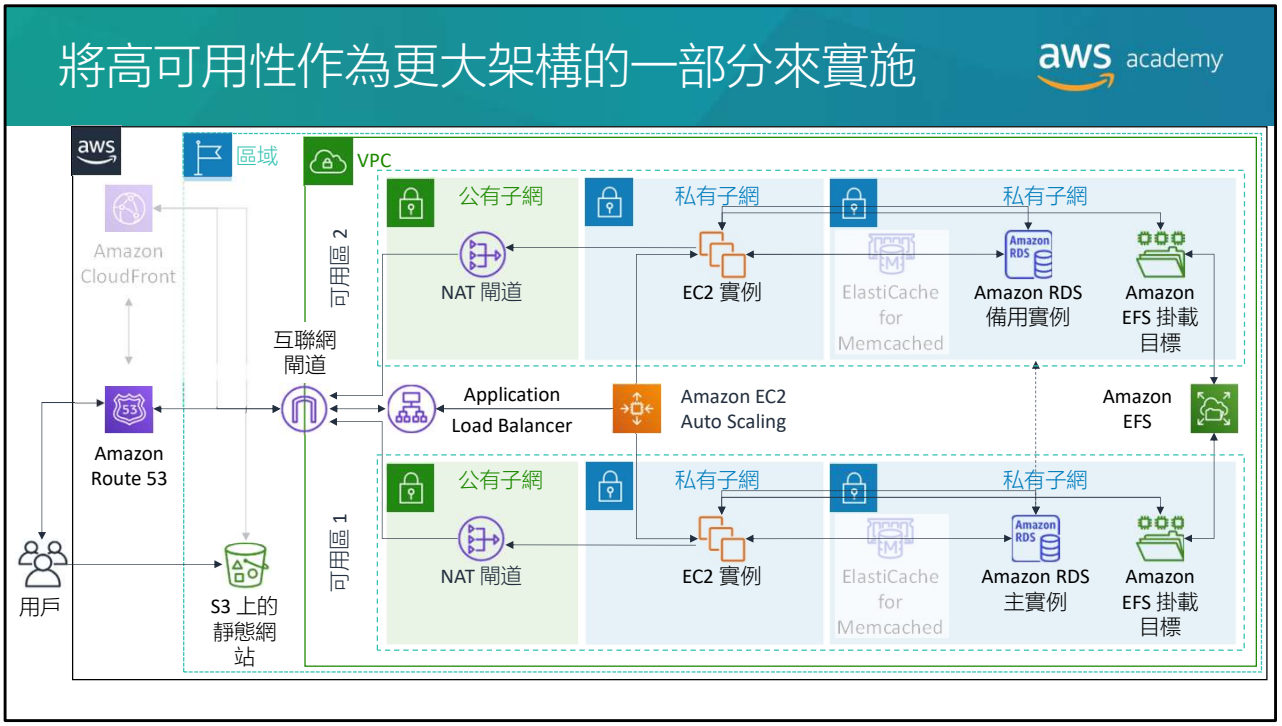
- 在架構中使用 Amazon EC2 Auto Scaling 來提高彈性
- 說明如何擴縮資料庫資源
- 部署 Application Load Balancer 以創建高度可用的環境
- 使用 Amazon Route 53 進行網域名稱系統 (DNS) 容錯移轉
- 創建高度可用的環境
- 設計能夠使用 Amazon CloudWatch 監控資源並做出相應反應的架構

學完本模組後，您應該能夠：

- 在架構中使用 Amazon EC2 Auto Scaling 來提高彈性
- 說明如何擴縮資料庫資源
- 部署 Application Load Balancer 以創建高度可用的環境
- 使用 Amazon Route 53 進行網域名稱系統 (DNS) 容錯移轉
- 創建高度可用的環境
- 設計能夠使用 Amazon CloudWatch 監控資源並做出相應反應的架構



介紹第 1 節：架構需求。



在本模組中，您將學習如何實施富有彈性且回應迅速的反應式架構。討論使此架構可擴縮且高度可用的元件，例如第二個可用區、Application Load Balancer、Amazon EC2 Auto Scaling 和 Amazon Route 53。

咖啡館業務要求

咖啡館將在著名的電視美食節目中亮相。當節目播出時，架構必須處理大幅增加的容量。

咖啡館很快將在著名的電視美食節目中亮相。當節目播出時，Sofía 和 Nikhil 預計咖啡館 Web 伺服器的用戶數量將出現短暫的激增，甚至可能達到數萬人。目前，咖啡館的 Web 伺服器部署在一個可用區，他們擔心伺服器無法應對預期的流量增長。他們希望確保客戶在訪問網站時擁有出色的體驗，不會遇到任何問題，例如下單延遲或延誤。

為了確保提供這種體驗，網站必須迅速回應，通過擴縮來滿足不斷變化的客戶需求，並做到高度可用。它還必須包含負載均衡。此架構必須跨多個應用程式伺服器分發客戶訂單請求，以便應對需求的增加，而不是讓單個伺服器超載。

反應式架構

彈性且可擴縮

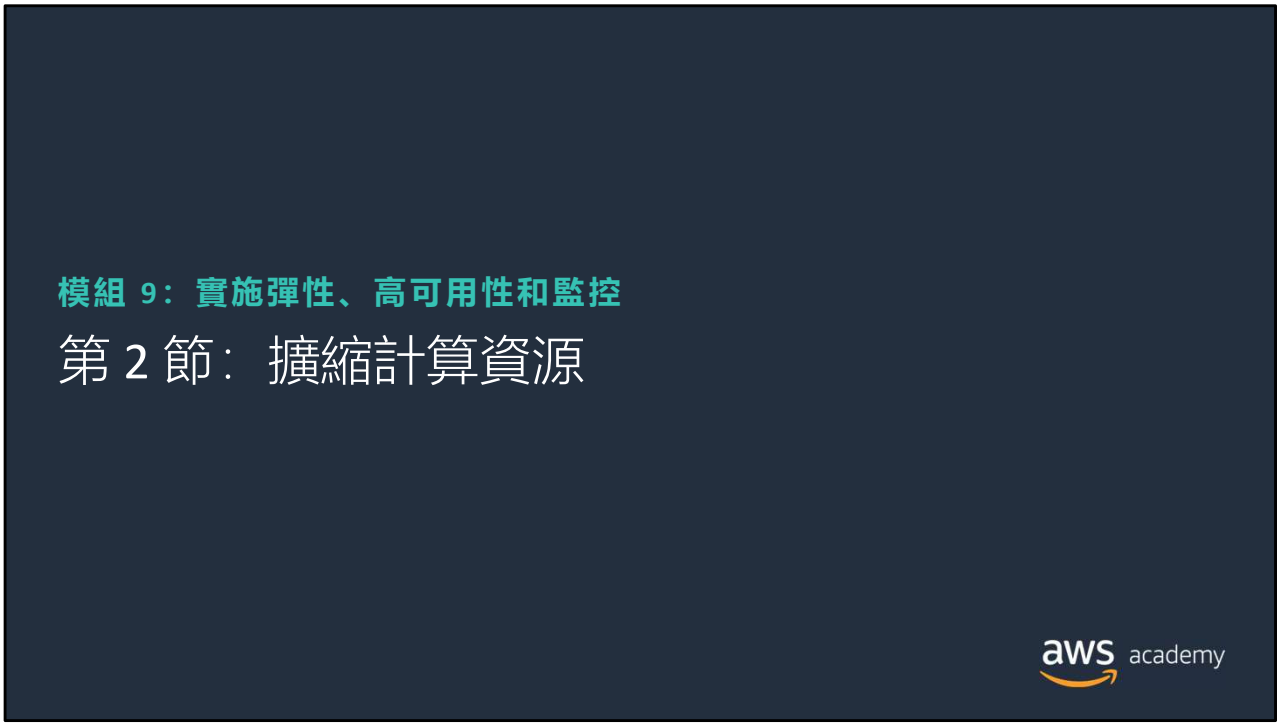
富有彈性

回應迅速

消息驅動

現代應用程式必須能夠處理大量資料，不會停機，回應時間達到亞秒級。為了滿足這些需求，您可以實施一個具有彈性、回應迅速、消息驅動的反應式系統。精心設計的反應式架構可以為您節省資金並為用戶提供更好的體驗。

在本模組中，您將學習如何在 AWS 上構建富有彈性且回應迅速的反應式架構。您將瞭解消息驅動元件，以便在後面的模組中學習構建解耦架構。



介紹第 2 節：擴縮計算資源。

什麼是彈性?

aws academy

彈性基礎設施可以隨著容量需求的變化而擴展和收縮。

示例:

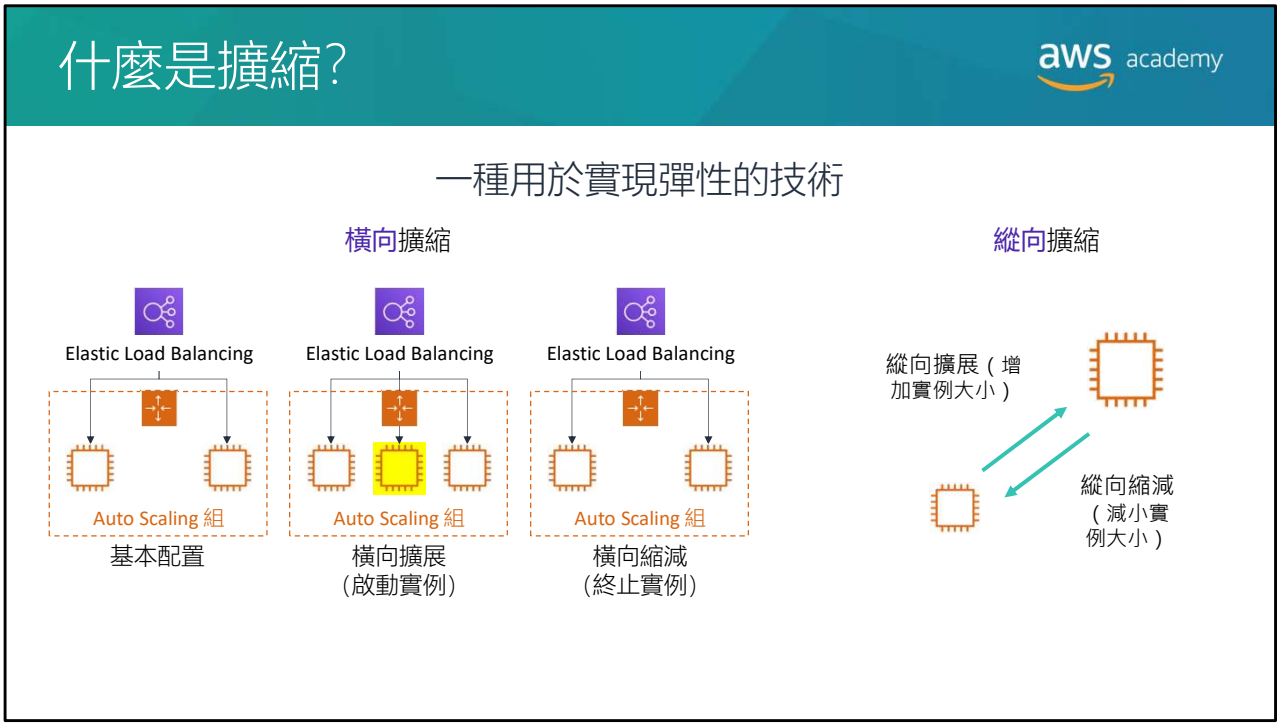
- 在流量激增時增加 Web 伺服器的數量
- 在流量下降時降低資料庫的寫入容量
- 處理整個架構中需求的日常波動

反應式架構的一個特徵是彈性。*彈性*意味著，基礎設施可隨著容量需求的變化而擴展和收縮。您可以在需要時獲取資源，在不需要時釋放資源。

彈性使您能夠:

- 當應用程式流量激增時，增加 Web 伺服器的數量
- 在流量下降時降低資料庫的寫入容量
- 處理整個架構中需求的日常波動

在咖啡館的例子中，彈性很重要，因為在電視節目播出之後，網站的流量可能會立即增加。流量可能會在一周後下降到正常水準，或者可能在節假日期間再次增加。



*擴縮*是一種用於實現彈性的技術。擴縮是指增加或減少應用程式計算容量的能力。

擴縮有兩種類型:

- *橫向擴縮*是指添加或刪除資源。例如，您可能需要向存儲陣列添加更多硬碟驅動器，或添加更多伺服器來支援應用程式。添加資源稱為*擴展*，而終止資源稱為*縮減*。橫向擴縮是構建利用雲計算彈性的互聯網級應用程式的有效方式。

- *縱向擴縮*是指增大或減小單個資源的規格。例如，您可以升級伺服器，使其具有更大的硬碟驅動器或更快的 CPU。借助 Amazon Elastic Compute Cloud (Amazon EC2),，您可以停止實例並將其大小調整為具有更多 RAM、CPU、I/O 或聯網功能的實例類型。縱向擴縮最終可能達到極限，而且有時不是一種具有成本效益或高度可用的方法。但它很容易實施，對於許多使用案例來說可能已經足夠，尤其是在短期內。

Amazon EC2 Auto Scaling

- 根據指定條件 啟動或終止實例
- 指定後，使用負載等化器自動注冊新實例
- 可以跨可用區啟動

在雲中，可以自動處理擴縮。Amazon EC2 Auto Scaling 有助於保持應用程式的可用性，允許您根據您定義的策略、計畫和運行狀況檢查自動添加或刪除 EC2 實例。如果您指定了擴縮策略，Amazon EC2 Auto Scaling 可以在應用程式需求增加或降低時啟動或終止實例。

Amazon EC2 Auto Scaling 與 Elastic Load Balancing 集成 – 它會自動向負載等化器註冊新實例，以便在實例之間分配傳入的流量。

Amazon EC2 Auto Scaling 允許您在一個區域中構建跨多個可用區的高可用性架構。稍後您將在本模組中瞭解有關高可用性的更多資訊。如果一個可用區運行狀況不佳或無法使用，Amazon EC2 Auto Scaling 會在未受影響的可用區中啟動新實例。當運行狀況不佳的可用區恢復到正常運行狀態時，Amazon EC2 Auto Scaling 會自動將這些應用程式實例重新平均分配到所有指定的可用區中。

Amazon EC2 Auto Scaling 提供了多種擴縮調整方式來滿足您的應用程式需求：

- 計劃擴縮 – 使用計畫擴縮，擴縮操作將按照日期和時間的函數自動執行。如果您確切地知道應在何時增加或減少組中的實例數量，那麼該功能對於可預測的工作負載會非常有用。例如，假設您每週的 Web 應用程式流量在星期三開始增加，星期四仍然保持較高水準，然後在星期五開始減少。您可以根據 Web 應用程式的可預測流量模式來規劃擴縮操作。要實施計畫擴縮，您可以創建計劃操作。

- 動態按需擴縮 – 此方法是更高級的資源擴縮方式。它允許您定義用於控制擴縮過程的參數。例如，您有一個當前在兩個 EC2 實例上運行的 Web 應用程式。您希望在應用程式負載變化時將 Auto Scaling 組的 CPU 利用率保持在接近 50%。在根據條件變化進行擴縮，但卻不知道條件何時改變時，可以使用這種方法。動態擴縮為您提供了額外的容量來應對流量高峰，而無需維護過多的空閒資源。您可以將 Auto Scaling 組配置為自動擴縮來滿足這一需求。

- 預測性擴縮 – 您可以將 Amazon EC2 Auto Scaling 與 AWS Auto Scaling 配合使用來實施預測性擴縮，使用此方式時，容量可根據預測的需求進行擴縮。預測性擴縮使用從您的 Amazon EC2 實際使用情況中收集的資料，而這些資料通過 AWS 的觀察得出的數十億個資料點得到進一步豐富。然後，AWS 使用經過良好訓練的機器學習模型來預測您的預期流量（和 Amazon EC2 使用情況），包括每日和每週模式。該模型至少需要 1 天的歷史資料才能開始進行預測。每 24 小時重新評估一次，以創建接下來 48 小時的預測。預測過程中會產生可以驅動一組或多組自動擴縮的 EC2 實例的擴縮計畫。

動態擴縮和預測性擴縮可結合使用，以便更快地擴縮基礎設施。

最後，您還可以手動添加或刪除 EC2 實例。對於手動擴縮，您只指定 Auto Scaling 組的最大容量、最小容量或所需容量的變化。

動態擴展策略類型

- 簡單擴縮 – 單次擴縮調整
  - 示例使用案例：新工作負載、突增的工作負載
- 步進擴縮 – 根據警報違規數量調整
  - 示例使用案例：可預測的工作負載
- 目標跟蹤擴縮 – 特定指標的目標值
  - 示例使用案例：橫向可擴縮應用程式，比如負載均衡應用程式和批資料處理應用程式

動態擴縮意味著調整應用程式的容量以滿足不斷變化的需求，從而優化可用性、性能和成本。擴縮策略類型決定了如何執行擴縮操作：

- 對於**步進擴縮**和**簡單擴縮**策略，您要為觸發擴縮過程的 CloudWatch 警報選擇擴縮指標和閾值。您還要定義在指定數量的評估期內違反閾值時應如何擴縮 Auto Scaling 組。主要區別在於，步進擴縮允許根據一組擴縮調整值（稱為*步進調整*，具體因警報觸發情況而異）來增加或減少 Auto Scaling 組的當前容量。

- **目標跟蹤擴縮**策略根據特定指標的目標值，增加或減少當前組容量。這種類型的擴縮類似於您家中負責控制溫度的恒溫器： 您只需選擇一個溫度，恒溫器會替您完成所有其他工作。使用目標跟蹤擴縮策略時，您可以選擇擴縮指標並設置目標值。Amazon EC2 Auto Scaling 會創建和管理 CloudWatch 警報，這些警報會根據指標和目標值觸發擴縮策略並計算擴縮調整值。擴縮策略根據需要增加或減少容量，將指標保持在指定的目標值或接近指定的目標值。除了將指標保持在接近目標值以外，目標跟蹤擴縮策略還會針對由於負載模式變化而造成的指標變化進行調整。

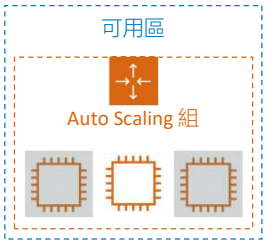
要瞭解有關目標跟蹤擴展擴縮的更多資訊，請參閱以下資源：
- Amazon EC2 Auto Scaling 的目標跟蹤擴縮策略
- Set it and Forget it Auto Scaling Target Tracking Policies
- AWS re: Invent 2017: Auto Scaling Prime Time: Target Tracking Hits the Bullseye at Netflix

Auto Scaling 組

Auto Scaling 組定義以下內容：

- 最小容量
- 最大容量
- 所需容量*

容量?



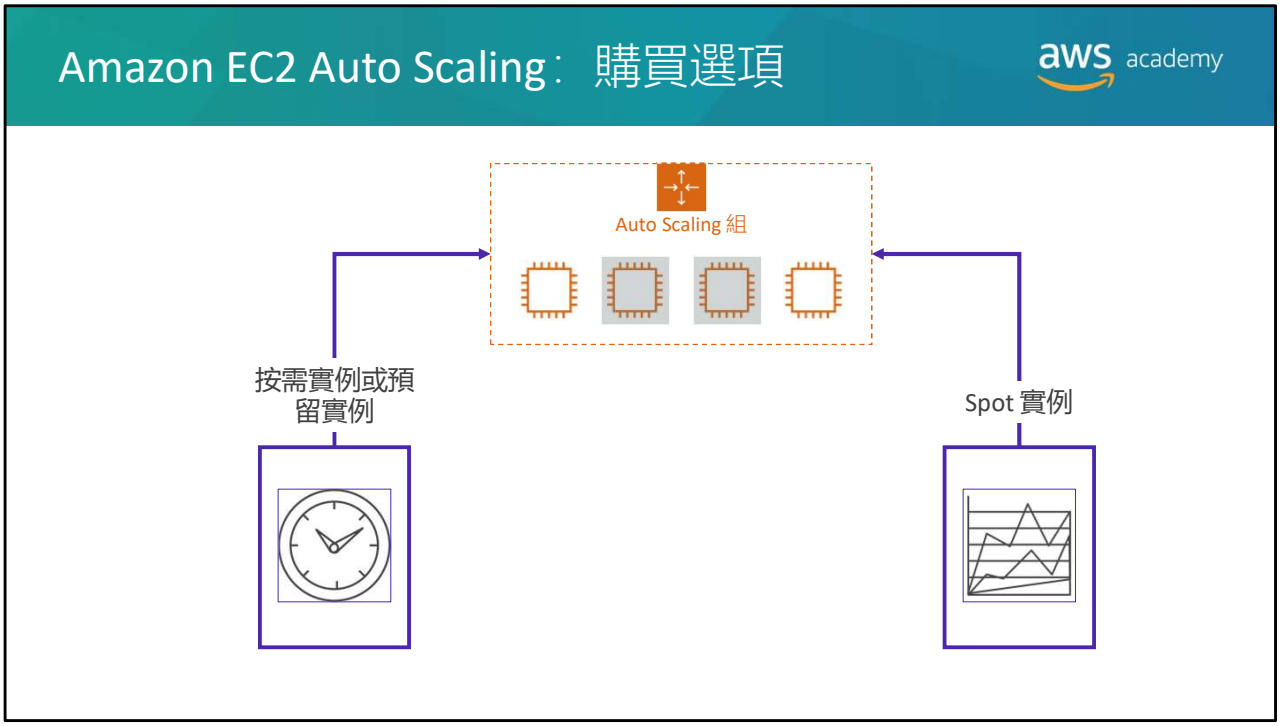
*所需容量反映了正在運行並會隨對事件的回應而波動的實例數量。

Amazon EC2 Auto Scaling 可幫助您確保擁有適量的 Amazon EC2 實例來處理您的應用程式負載。

您可以創建稱為 *Auto Scaling* 的 Amazon EC2 實例集合。您可以指定每個 Auto Scaling 組內的*最小*實例數量，而 Amazon EC2 Auto Scaling 會確保您的組始終不低於此數量。您可以指定每個 Auto Scaling 組內的*最大*實例數量，而 Amazon EC2 Auto Scaling 會確保您的組始終不高於此數量。如果您在創建組時或創建之後指定*所需*容量，則 Amazon EC2 Auto Scaling 會確保您的組內實例始終保持此數量。

請注意，所需容量是基於觸發器的設置，可能會隨著諸如違反閾值之類的事件而波動。它反映了當時正在運行的實例數量，並且永遠不能低於最小值或高於最大值。擴縮策略的作用在於充當您的自動代表，就如何調整所需容量做出決策。然後，Amazon EC2 Auto Scaling 會回應所需容量配置的更改。

首先，設置所需容量以告知 Auto Scaling 組在特定時間運行的實例數量。在 Amazon EC2 Auto Scaling 擴縮之前，當前正在運行的實例數量可能與所需的值不同。



Amazon EC2 Auto Scaling 使您能夠根據不斷變化的條件橫向擴展和縮減基礎設施。在配置 Auto Scaling 組時，可以指定其使用的 EC2 實例類型。您還可以指定應由按需實例、預留實例和 Spot 實例滿足所需容量的百分比。然後，Amazon EC2 Auto Scaling 將根據這些首選項預置能夠達到預期容量的價格最低的實例組合。

您可以僅使用一種實例類型。但最好使用多個實例類型以避免從容量不足的實例池中啟動實例。如果 Auto Scaling 組對 Spot 實例的請求無法在一個 Spot 實例池中得到滿足，它將繼續在其他 Spot 實例池中嘗試，而不是啟動按需實例。

有關購買選項的更多資訊，請參閱具有多個實例類型和購買選項的 Auto Scaling 組。

自動擴縮注意事項

- 多種類型的自動擴展
- 簡單、分步或目標跟蹤擴展
- 多個指標（不僅僅是 CPU）
- 何時橫向擴展和縮減
- 使用生命週期掛鉤

在使用 Amazon EC2 Auto Scaling 擴縮架構時，需要考慮以下事項：

- 多種自動擴縮類型 – 您可能需要實施計畫擴縮、動態擴縮和預測性擴縮的組合。
- 動態擴縮策略類型 – 簡單擴縮策略可根據一個擴縮調整值來增加或減少當前組容量。步進擴縮策略根據一組擴縮調整值（稱為*步進調整*，因警報違反情況而異）來增加或減少當前組容量。目標跟蹤擴縮策略根據特定指標的目標值，增加或減少當前組容量。
- 多個指標 – 某些架構必須根據兩個或更多指標（不僅僅是 CPU）進行擴縮。AWS 建議您使用目標跟蹤擴縮策略按某個指標進行擴縮，例如平均 CPU 利用率或 Application Load Balancer 的 RequestCountPerTarget 指標。可通過在容量增加時減少並在容量減少時增加的指標按比例橫向擴展或縮減使用目標跟蹤的實例數。此類指標有助於確保 Amazon EC2 Auto Scaling 嚴格遵循應用程式的需求曲線。
- 何時橫向擴展和縮減 – 橫向擴展要早而快，橫向縮減要緩而慢。
- 使用生命週期掛鉤 – *生命週期掛鉤使您能夠在 Auto Scaling 組啟動或終止實例時通過*暫停實例來執行自訂操作。當實例暫停時，它將保持等候狀態，直到您使用 complete-lifecycle-action 命令或 CompleteLifecycleAction 操作完成生命週期操作，或者超時時段結束（默認為 1 小時）。

演示：為 Amazon EC2 Auto Scaling 創建擴縮策略

現在，講師可能會選擇演示如何為 Amazon EC2 Auto Scaling 創建目標跟蹤和步進擴縮策略。

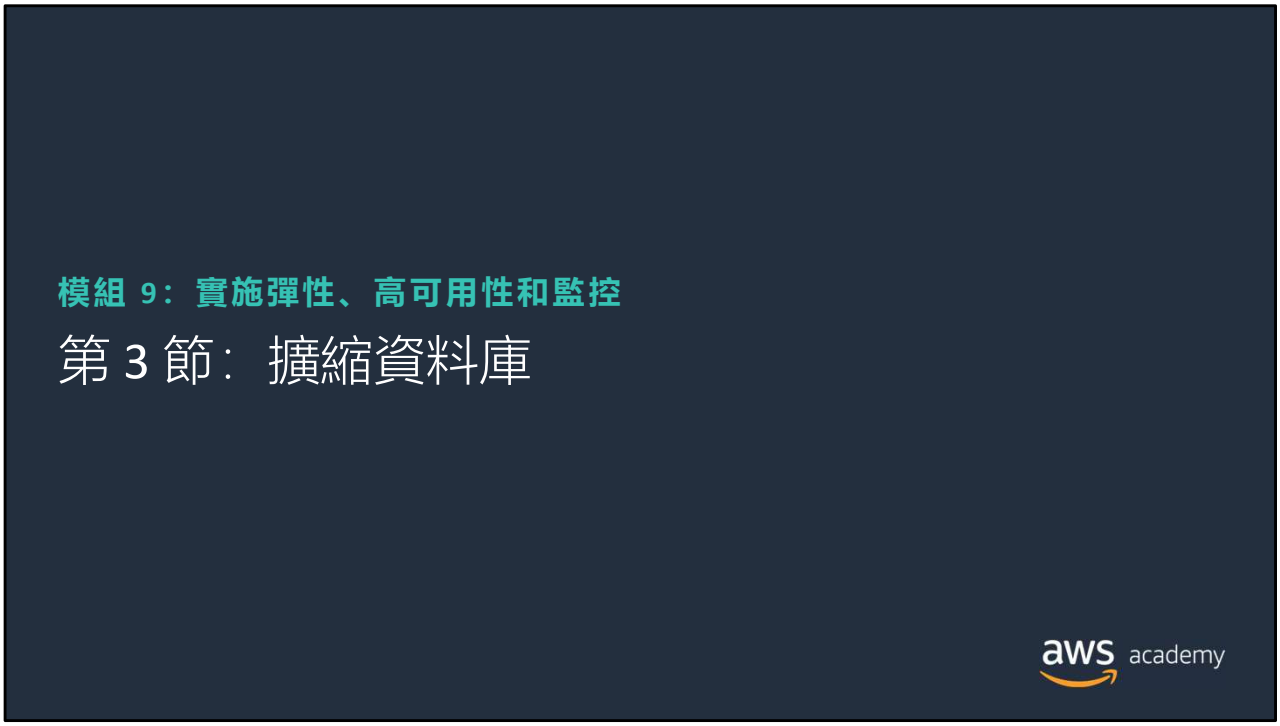
第 2 節要點

aws academy

- 彈性基礎設施可以隨著容量需求的變化而擴展和收縮
- Amazon EC2 Auto Scaling 會根據您定義的策略、計畫和運行狀況檢查自動添加或刪除 EC2 實例
- Amazon EC2 Auto Scaling 提供了多種擴縮選項，以最好地滿足您的應用程式需求
- 在配置 Auto Scaling 組時，您可以指定 EC2 實例類型及其使用的定價模型組合

本模組中這節內容的要點包括：

- 彈性基礎設施可以隨著容量需求的變化而擴展和收縮
- Amazon EC2 Auto Scaling 會根據您定義的策略、計畫和運行狀況檢查自動添加或刪除 EC2 實例
- Amazon EC2 Auto Scaling 提供了多種擴縮選項，以最好地滿足您的應用程式需求
- 在配置 Auto Scaling 組時，您可以指定 EC2 實例類型及其使用的定價模型組合

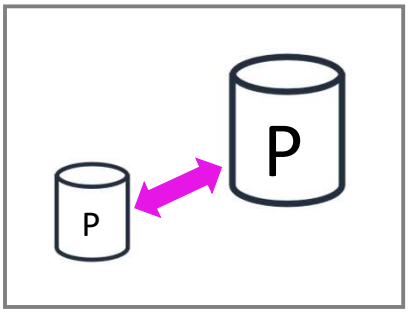


介紹第 3 節：擴縮資料庫。

在上一節中，您學習了如何擴縮 EC2 實例。除此之外，您還可以擴縮資料庫實例。在本節中，您將學習如何擴縮關聯式資料庫和非關聯式資料庫。

使用 Amazon RDS 進行縱向擴縮：按鈕式擴縮

- 縱向擴展或縮減資料庫實例
- 從 micro 到 24xlarge 以及介於兩者之間的所有實例
- 以最短的停機時間縱向擴縮

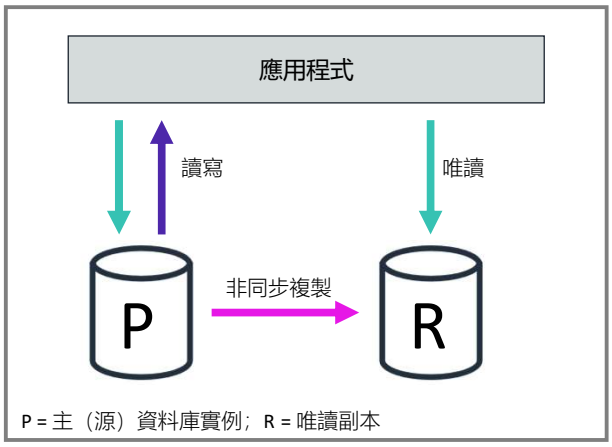


首先，考慮擴縮關聯式資料庫。作為一項託管服務，Amazon Relational Database Service (Amazon RDS) 可以擴縮您的關聯式資料庫，使其能夠滿足應用程式不斷增長的需求。

您可以通過更改 Amazon RDS 實例類來縱向擴縮 Amazon RDS 資料庫 (DB) 實例。如果您決定擴展或縮減資料庫實例的可用計算資源，注意在修改資料庫實例類時，資料庫將暫時不可用。這段不可用的時間通常只持續幾分鐘。除非您指定應立即應用修改，否則它僅在資料庫實例維護期間發生。

當您擴展或縮減資料庫實例時，存儲大小保持不變，且不受更改影響。您可以單獨修改資料庫實例以增加分配的存儲空間或通過更改存儲類型來提高性能，例如，從通用固態硬碟 (SSD) 更改為預置的每秒輸入/輸出操作數 (IOPS) SSD。您還可以使用 Amazon RDS Storage Autoscaling 來自動擴展存儲容量，以適應不斷增長的資料庫工作負載，而不是手動預置存儲。

使用 Amazon RDS 進行橫向擴縮：唯讀副本



- 橫向擴縮以處理多讀少寫工作負載
- 最多 5 個唯讀副本，最多 15 個 Aurora 副本
- 複制是非同步的
- 適用於 Amazon RDS for MySQL、MariaDB、 PostgreSQL 和 Oracle

除了縱向擴縮，您還可以橫向擴縮資料庫。Amazon RDS 使用 MariaDB、MySQL、Oracle 和 PostgreSQL 資料庫引擎的內置複製功能，從來源資料庫實例創建一個特殊類型的資料庫實例（稱為*只讀副本*）。對來源資料庫實例的更新將非同步複製到唯讀副本。您可以將應用程式發出的讀取查詢路由到唯讀副本，以減輕來源資料庫實例上的負載。

為了提高多讀少寫資料庫工作負載的性能，可以使用唯讀副本來橫向擴縮來源資料庫實例。您可以為給定來源資料庫實例創建一個或多個副本（最多 5 個），利用多個資料副本滿足大量應用程式讀取流量需求，從而增加總讀取輸送量。

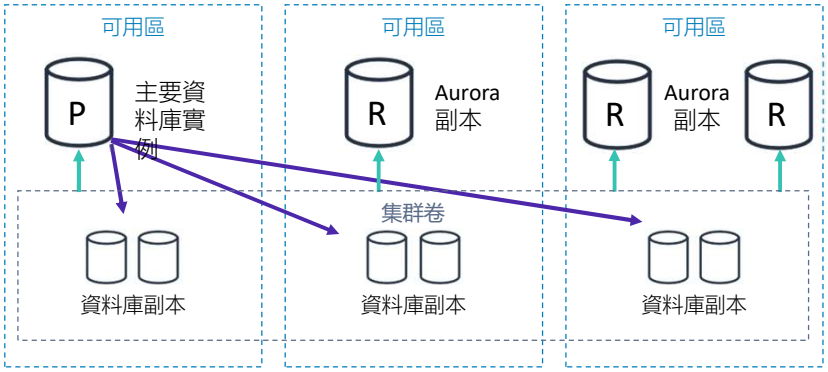
如果發生災難，您可以將唯讀副本提升為獨立資料庫實例來提高資料庫的可用性。

有關唯讀副本的更多資訊，請參閱使用唯讀副本。

有關如何擴展 RDS 資料庫實例的更多資訊，請參閱以下資源：
- 修改 Amazon RDS 資料庫實例（AWS 文檔）
- Scaling Your Amazon RDS Instance Vertically and Horizontally AWS 資料庫博客文章

使用 Amazon Aurora 進行擴縮

每個 Aurora 資料庫集群最多可包含 15 個 Aurora 副本



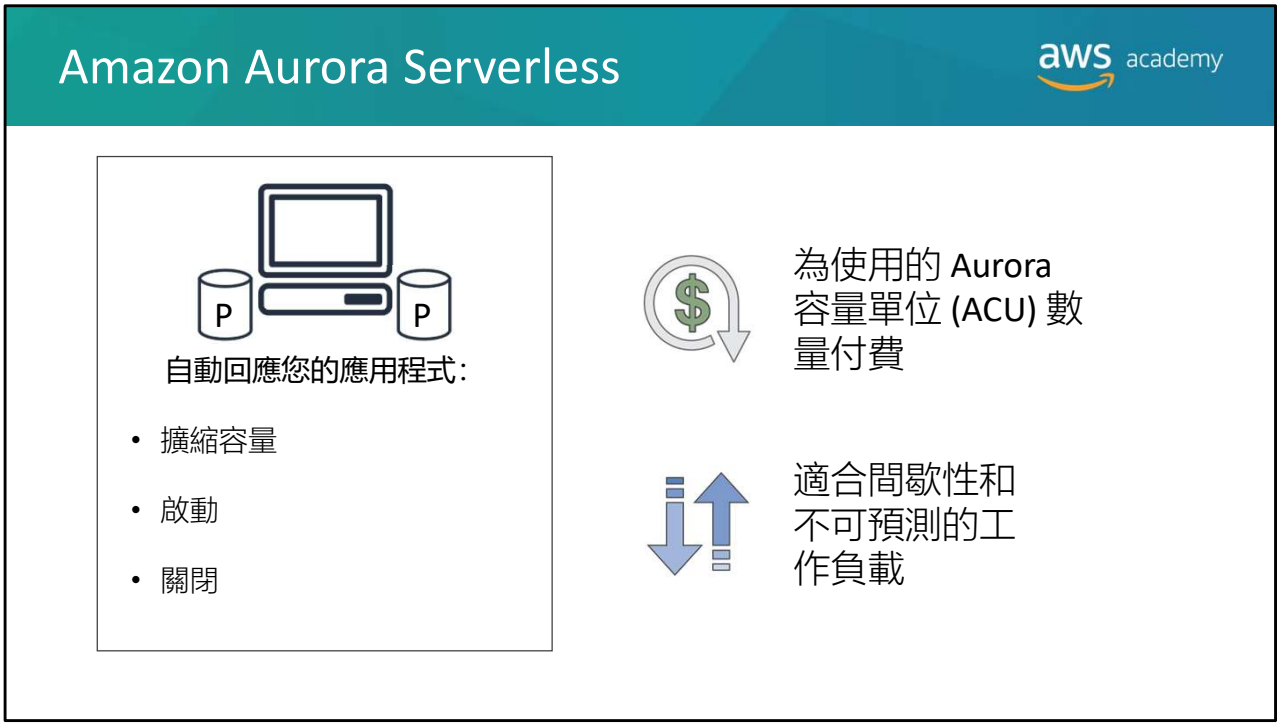
Amazon Aurora 通過采用專門為資料庫工作負載構建的 SSD 支援的虛擬化存儲層，進一步擴展了唯讀副本的好處。Amazon Aurora 是針對雲構建的相容 MySQL 和 PostgreSQL 的關聯式資料庫引擎。Amazon RDS 可用來管理您的 Amazon Aurora 資料庫，處理預置、修補、備份、恢復、故障檢測和修復等任務。Amazon Aurora 採用一種具有容錯能力並能自我修復的分散式存儲系統，這一系統可以將每個資料庫實例自動擴展到高達 64TB。 它提供高性能和可用性、時間點恢復、向 Amazon Simple Storage Service (Amazon S3) 的持續備份，以及跨三個可用區 (AZ) 的複製。

Amazon Aurora *資料庫集群*由一個或多個資料庫實例以及一個管理這些資料庫實例資料的集群卷組成。

Aurora 資料庫集群由兩類資料庫實例組成：

- **主要資料庫實例** – 支援讀取和寫入操作，並執行對集群卷的所有資料修改。每個 Aurora 資料庫集群都有一個主要資料庫實例。
- **Aurora 副本** – 連接到與主要資料庫實例相同的存儲卷，並且僅支援讀取操作。除了主要資料庫實例外，每個 Aurora 資料庫集群最多可包含 15 個 Aurora 副本。

您可以選擇資料庫實例類大小並添加 Aurora 副本以增加讀取輸送量。如果您的工作負載發生變化，您可以修改資料庫實例類大小並更改 Aurora 副本的數量。此模型適合可預測的資料庫工作負載，因為您可以根據預期工作負載手動調整容量。



但是，在某些環境中，工作負載可能是間歇性和不可預測的。也許存在可能僅持續幾分鐘或幾小時的大量工作負載的時間段以及少量活動或甚至無活動的長時間段。例如，一些進行間歇性銷售活動的零售網站、根據需要生成報告的報告資料庫、開發和測試環境，以及具有不確定需求的新應用程式。在這些情況及很多其他情況下，可能很難在正確的時間配置正確的容量。在您為未使用的容量付費時，這還可能會產生更高的成本。

Aurora Serverless 是 Amazon Aurora 的一種按需自動擴縮配置。Aurora Serverless 使您的資料庫能夠根據應用程式的需求自動啟動、關閉以及擴展或縮減容量。它讓您可以在雲中運行資料庫，而無需管理任何資料庫實例。Aurora Serverless 可用於不頻繁、間歇性或不可預測的工作負載。

您可以創建資料庫終端節點，而無需指定資料庫實例類大小。您可以指定 Aurora 容量單位 (ACU)。每個 ACU 都是處理和記憶體容量的組合。資料庫存儲自動從 10 吉位元組 (GiB) 擴展到 64 百萬位元組 (TiB)，這與標準 Aurora 資料庫集群中的存儲相同。數據庫終端節點連接到代理佇列，後者將工作負載路由到資源佇列。Aurora Serverless 根據最小和最大容量規格自動擴縮資源。

您需要在資料庫處於活動狀態期間按照每秒使用的資料庫容量付費，並在標準配置和無伺服器配置之間遷移。您需要為使用的 ACU 數量付費。

有關 Aurora Serverless 的更多資訊，請參閱 Aurora Serverless 的工作原理。

橫向擴縮：資料庫分片

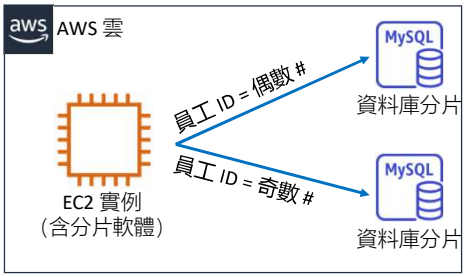
如果沒有分片，所有資料都將駐留在一個分區中。

- 示例：一個資料庫中的員工 ID

通過分片，數據將被拆分為大塊 （分片）。

- 示例：偶數員工 ID 在一個資料庫中，奇數員工 ID 在另一資料庫中

在許多情況下，分片可提高寫入性能。

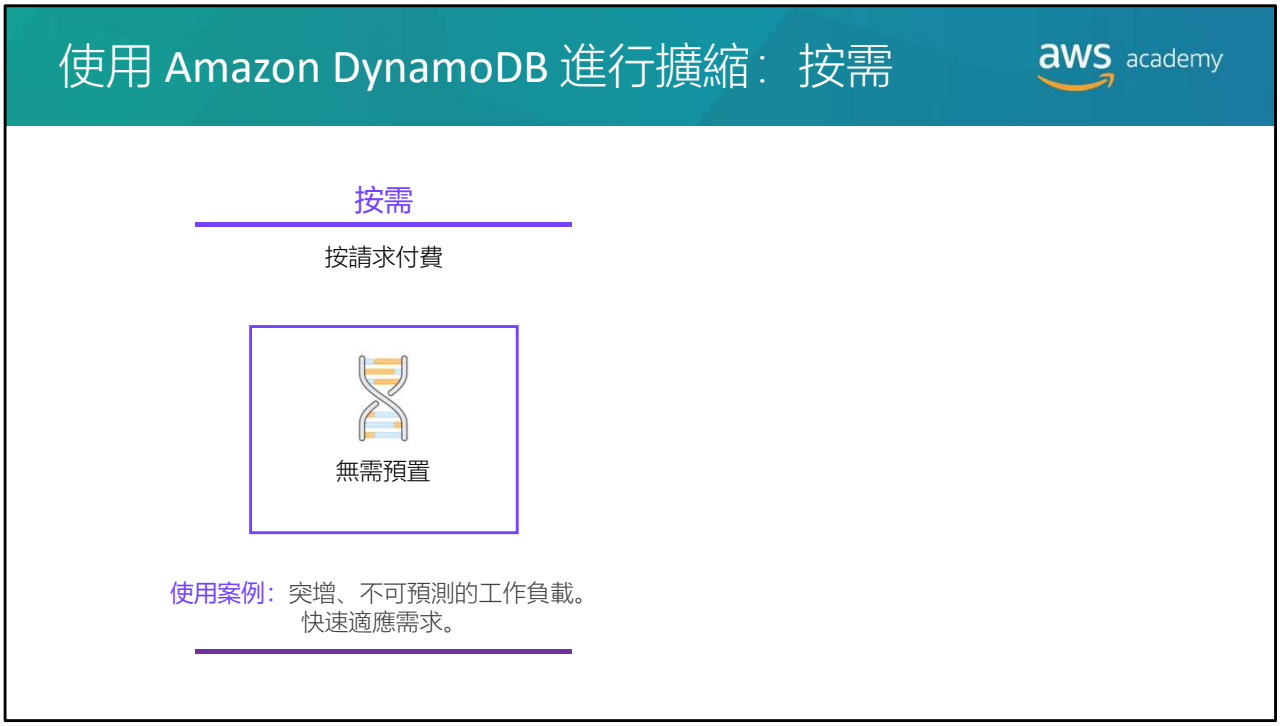


分片，也稱為*橫向分區*，是關聯式資料庫中常用的擴展方法，可提高寫入性能。

分片是一種技術，它將資料拆分成較小的子集，並將它們分佈在多個物理上分離的資料庫伺服器上。每台伺服器均稱為資料庫*分片*。資料庫分片通常具有相同類型的硬體、資料庫引擎和資料結構，可產生相似的性能水準。但是，它們彼此互不知曉，這是將分片與其他擴展方法（例如資料庫集群或複製）區分開來的關鍵特徵。

分片資料庫架構提供了可擴縮性和容錯能力。通過分片，可以根據需要在多個資料庫伺服器之間拆分數據。此外，如果一個資料庫分片存在硬體問題或經歷容錯移轉，其他分片將不會受到影響，因為單點故障或減速在物理上是隔離的。這種方法的缺點是，由於資料分佈在多個分片，必須專門設計資料映射和路由邏輯，以便從多個分片讀取或聯接資料。與非分片資料庫相比，這些類型的查詢可能會產生更高的延遲。

有關使用 Amazon RDS 進行分片的更多資訊，請閱讀這篇 AWS 資料庫博客文章。



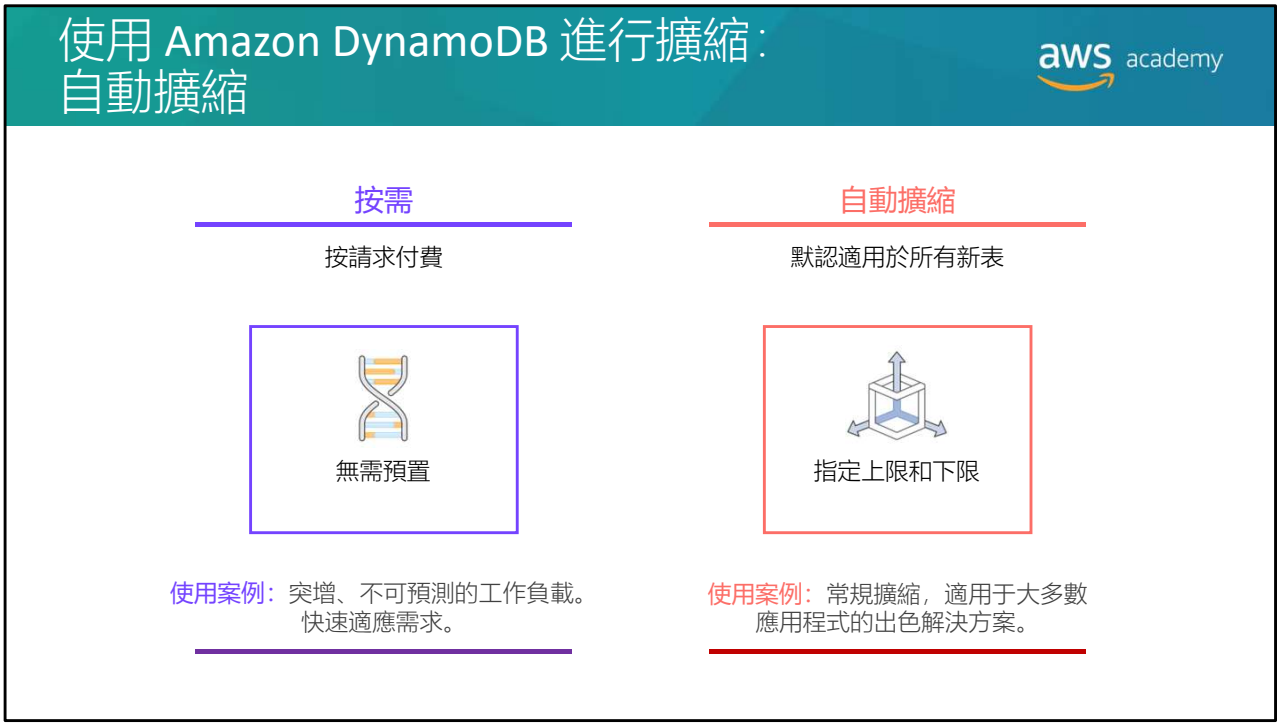
對於需要非關聯式資料庫的不可預測的工作負載，Amazon DynamoDB On-Demand 是 DynamoDB 的靈活計費選項。它可以每秒處理數千個請求，而無需進行容量規劃。它採用按請求付費的定價模式，而不是預置的定價模式。

DynamoDB On-Demand 可以觀察到流量水準的任何增長或擴展。如果流量水準達到新峰值，DynamoDB 會快速調整以適應工作負載要求。如果您的工作負載難以預測，或在短時間內突增，該功能非常有用。

您可以將表從預置容量更改為按需容量，每天可以更改一次。您可以根據需要隨時將按需容量更改為預置容量。

要瞭解有關 Amazon DynamoDB On-Demand 的更多資訊，請閱讀這篇 AWS 新聞博客文章。

使用 Amazon DynamoDB 進行擴縮：自動擴縮

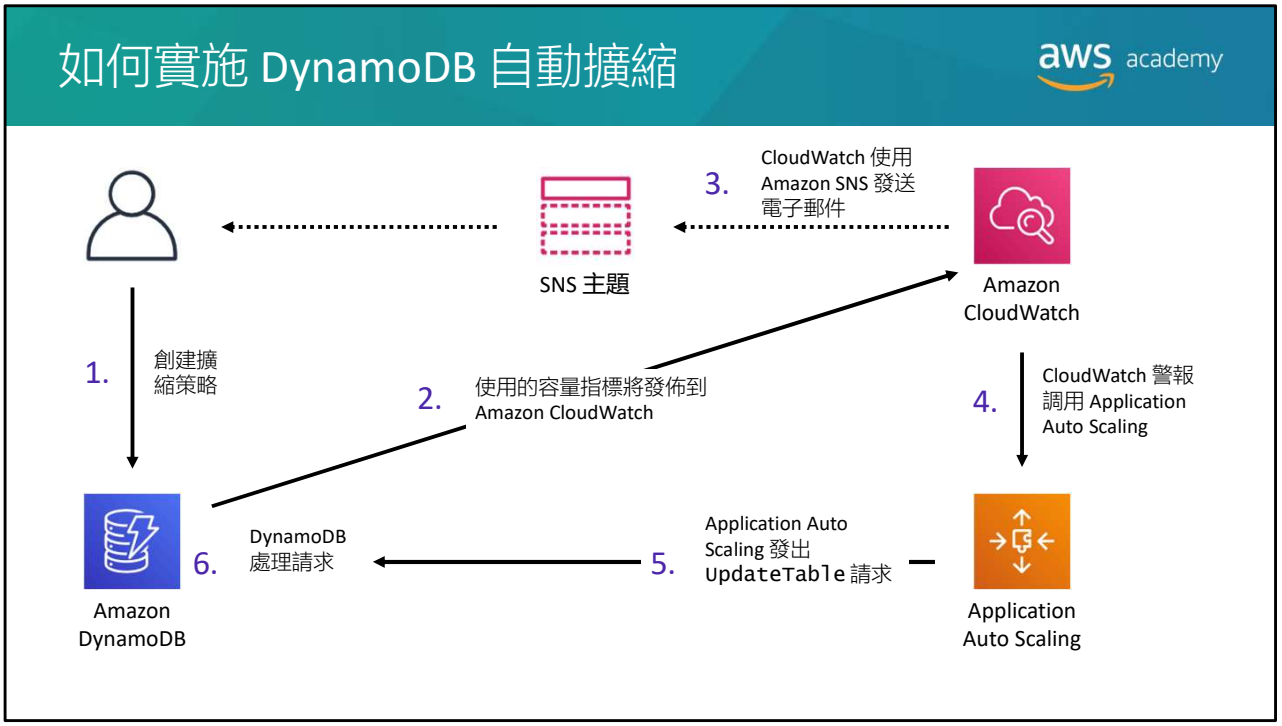


Amazon DynamoDB 還具有一項默認啟用的名為 *Auto Scaling* 的功能。Amazon DynamoDB Auto Scaling 可以根據動態變化的請求量自動調整讀取和寫入吞吐容量，而無需停機。借助 DynamoDB Auto Scaling，您只需設置所需的輸送量利用率目標以及最小和最大限制 – DynamoDB Auto Scaling 會完成其餘工作。

DynamoDB Auto Scaling 與 Amazon CloudWatch 配合使用，可持續監控實際輸送量消耗。當實際利用率偏離目標時，它會自動擴展或縮減容量。

除了您已為 DynamoDB 和 CloudWatch 警報支付的費用外，使用 DynamoDB Auto Scaling 不會產生任何額外費用。

有關 DynamoDB Auto Scaling 的更多資訊，請參閱使用 DynamoDB Auto Scaling 自動管理吞吐容量。



Amazon DynamoDB Auto Scaling 使用 *Application Auto Scaling* 服務代您動態調整預置的吞吐容量，以回應實際的流量模式。此服務使表或全域二級索引 (GSI) 能夠增加預置的讀取和寫入容量，從而不受限制地應對流量的突增。如果工作負載減少，Application Auto Scaling 會降低輸送量，這樣您就無需為未使用的預置容量付費。

要實施 DynamoDB Auto Scaling，請執行以下步驟：

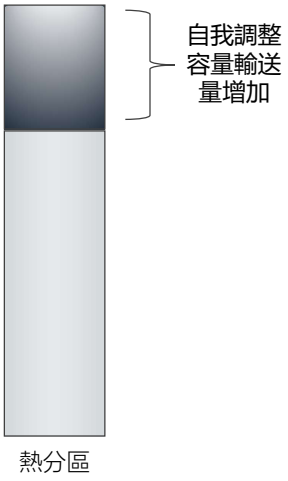
1. 為 DynamoDB 表或 GSI *創建*擴縮策略 。擴縮策略可指定要擴縮讀取容量還是寫入容量（或二者），並為表或索引指定最小的和最大的預置容量單位設置。
2. DynamoDB 將使用的容量指標發佈到 Amazon CloudWatch。
3. 如果表使用的容量在特定時段內超出目標利用率（或低於目標利用率），則 Amazon CloudWatch 會觸發警報。您可以使用 Amazon Simple Notification Service (Amazon SNS) 在 Amazon CloudWatch 控制台中查看警報並接收通知。
4. CloudWatch 警報調用 Application Auto Scaling 來評估您的擴縮策略。
5. Application Auto Scaling 向 DynamoDB 發出 *UpdateTable* 請求以調整表的預置輸送量。
6. DynamoDB 將處理 *UpdateTable* 請求，動態增加（或減少）表的預置吞吐容量，使它接近目標利用率。

有關如何實施 DynamoDB Auto Scaling 的更多資訊，請參閱使用 DynamoDB Auto Scaling 自動管理吞吐容量。

擴縮吞吐容量：DynamoDB 自我調整容量

- 無限制啟用對熱分區的讀寫操作
- 自動增加分區的輸送量以接收更多流量*
- 針對每個 DynamoDB 表自動啟用

*流量不能超過表的總預置容量或分區的最大容量。



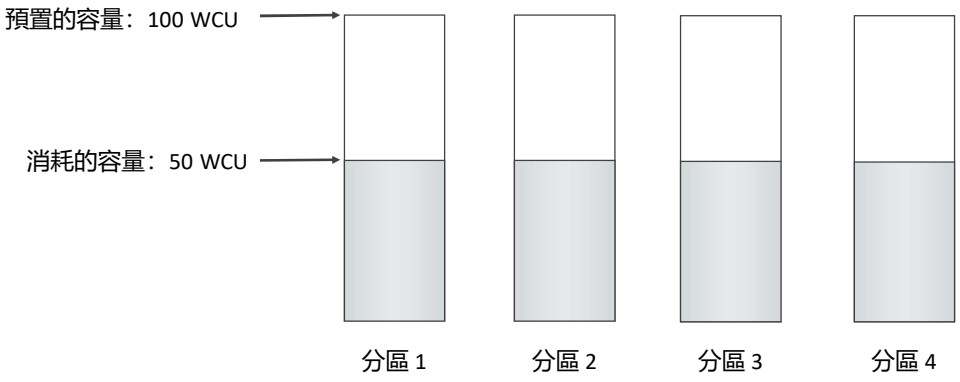
讀寫活動有時無法在各個分區之間平均分配。當資料訪問不平衡時，某個分區可以接收高於其他分區的讀取和寫入流量，這稱為*熱分區*。在極端情況下，如果單個分區接收到超過 3000 個讀取容量單位 (RCU) 或 1000 個寫入容量單位 (WCU)，則會發生限制。

為了更好地適應不均勻的訪問模式，DynamoDB 自我調整容量讓您的應用程式可以不斷對熱分區進行讀寫而不受限制。但是，流量不能超過表的總預置容量或分區的最大容量。自我調整容量的工作原理是，自動增加分區的吞吐容量來接收更多流量。

系統會自動為每個 DynamoDB 表啟用自我調整容量，因此您無需顯式啟用或禁用它。

自我調整容量示例 (1/3)

具有自我調整容量的示例表
預置的總容量 = 400 WCU
消耗的總容量 = 200 WCU

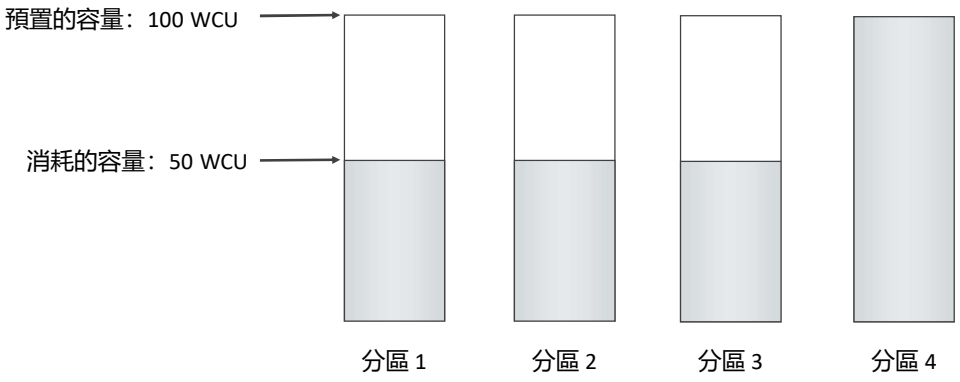


上圖展示了自我調整容量的工作原理。

示例表預置了 400 個 WCU，這些容量單位均勻分佈在 4 個分區中，每個分區最多支援每秒 100 個 WCU。

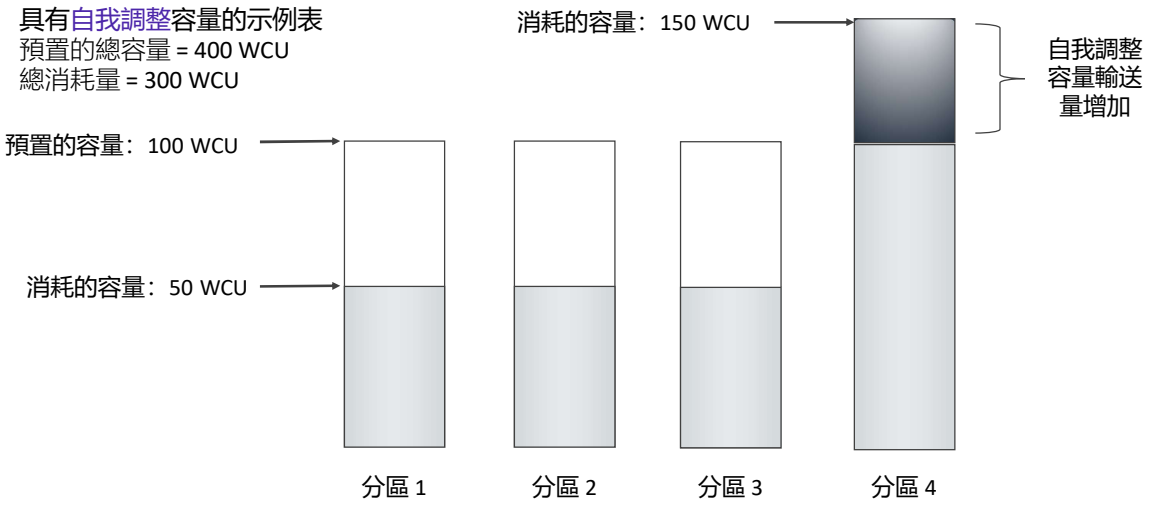
自我調整容量示例 (2/3)

具有自我調整容量的示例表
預置的總容量 = 400 WCU
消耗的總容量 = 250 WCU



分區 1、2 和 3 每秒接收的寫入流量為 50 個 WCU。分區 4 每秒接收 150 個 WCU。此熱分區可以在接收寫入流量的同時仍具有未利用的突增容量；但是，它最終會限制每秒超過 100 個 WCU 的流量。

自我調整容量示例 (3/3)



DynamoDB 自我調整容量通過增加分區 4 的容量來做出回應，因此分區 4 可以承受每秒 150 個 WCU 的更高工作負載，而不會受到限制。

有關 DynamoDB 自我調整容量的更多資訊，請參閱瞭解 DynamoDB 自我調整容量。

自我調整容量不能修復熱鍵和熱分區

分區鍵值	統一性
使用者 ID，應用程式有許多使用者	好
狀態碼，只有幾個可用的狀態碼	差
項目創建日期，四捨五入到最近的時間段（例如，日、小時或分鐘）	差
設備 ID，每個設備以相對相似的間隔訪問資料設備	好
設備 ID，即使跟蹤許多設備，其中一台也比所有其他設備更常用	差

表的主鍵的分區鍵部分決定了存儲表資料的邏輯分區。這反過來會影響底層物理分區。表的預置 I/O 容量在這些物理分區之間平均分配。因此，如果分區鍵設計無法均勻分發 I/O 請求，則可能會創建熱分區，從而導致預置的 I/O 容量使用受到限制且效率低下。

表的預置輸送量的最佳使用情況不僅取決於各個專案的工作負載模式，還取決於分區鍵設計。這既不意味著您必須訪問所有分區鍵值才能達到高效的輸送量級別，也不意味著訪問的分區鍵值的百分比必須很高。而是*意味著工作負載訪問的分區鍵值越不同，這些請求在分區空間中的分佈就越廣*。一般來說，隨著訪問的分區鍵值占分區鍵值總數的比率增大，您使用預置輸送量的效率就越高。

該表顯示了一些常見分區鍵 schema 的預置吞吐效率對比。

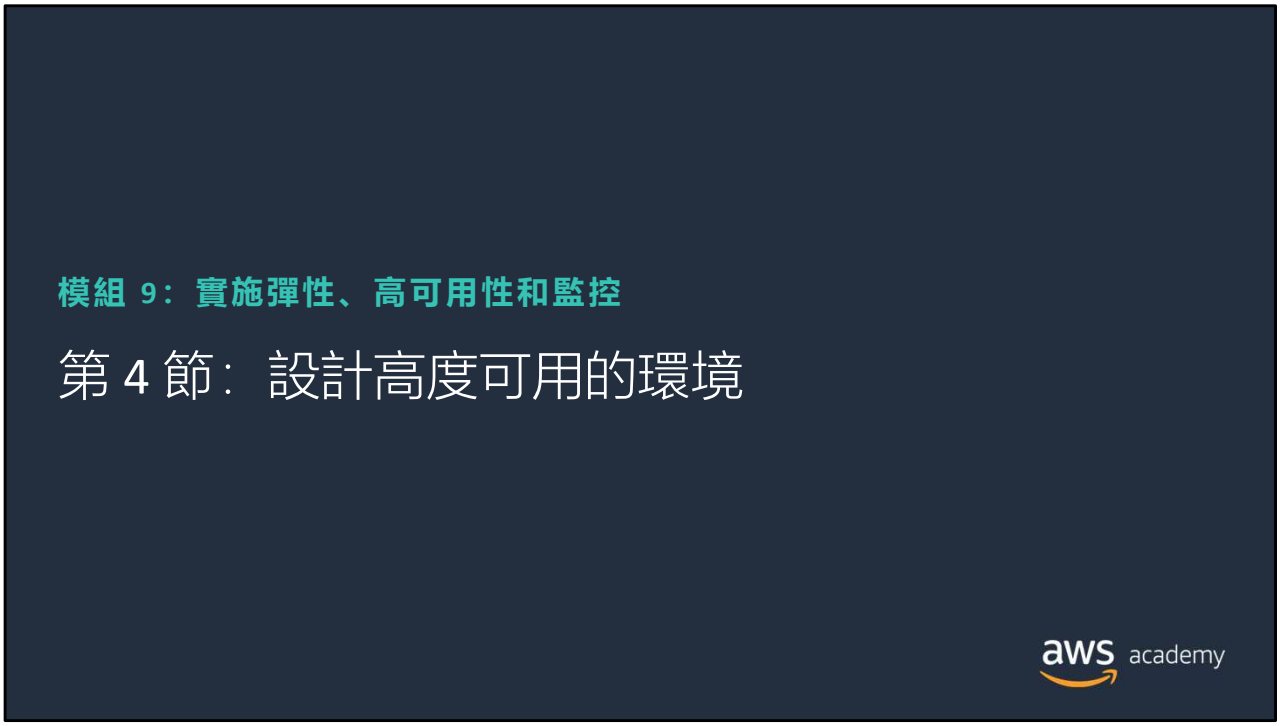
有關設計分區鍵的更多資訊，請參閱設計分區鍵以均勻分發工作負載。

第 3 節要點

- 您可以使用按鈕式擴縮來縱向擴縮 RDS 資料庫實例的計算容量
- 您可以使用唯讀副本或分區來橫向擴縮 RDS 資料庫實例
- 通過 Amazon Aurora，您可以選擇資料庫實例類的大小和 Aurora 副本的數量（最多 15 個）
- Aurora Serverless 根據最小和最大容量規格自動擴縮資源
- Amazon DynamoDB On-Demand 提供按請求付費的定價模式
- DynamoDB Auto Scaling 使用 Amazon Application Auto Scaling 動態調整預置的吞吐容量
- DynamoDB 自我調整容量的工作方式是自動增加分區的吞吐容量，從而接收更多的流量

本模組中這節內容的要點包括：

- 您可以使用按鈕式擴縮來縱向擴縮 RDS 資料庫實例的計算容量
- 您可以使用唯讀副本或分區來橫向擴縮 RDS 資料庫實例
- 通過 Amazon Aurora，您可以選擇資料庫實例類的大小和 Aurora 副本的數量（最多 15 個）
- Aurora Serverless 根據最小和最大容量規格自動擴縮資源
- Amazon DynamoDB On-Demand 提供按請求付費的定價模式
- DynamoDB Auto Scaling 使用 Amazon Application Auto Scaling 動態調整預置的吞吐容量
- DynamoDB 自我調整容量的工作方式是自動增加分區的吞吐容量，從而接收更多的流量



介紹第 4 節：設計高度可用的環境。

高度可用的系統

- 可以承受某種程度的降級，同時保持可用
- 最大限度地減少停機時間
- 需要極少的人為干預
- 在可接受的性能下降時間內從故障中恢復或轉移到輔助源

正常執行時間百分比	每年最長停機時間	每天等效停機時間
90%	36.5 天	2.4 小時
99%	3.65 天	14 分鐘
99.9%	8.76 小時	86 秒
99.99%	52.6 分鐘	8.6 秒
99.999%	5.25 分鐘	0.86 秒

最佳實踐是在設計和構建解決方案時，避免單點故障。要遵循此最佳實踐，您希望將架構設計為高度可用。

高度可用的*系統可以承受某種程度的降級，同時仍然保持可用。*在高度可用的系統中，停機時間盡可能減至最少，只需極少的人為干預。

高度可用的系統可在反應式架構中實現彈性。當受到負載（更多的服務請求）、攻擊或元件故障壓力時，彈性工作負載可以恢復。彈性工作負載可以在可接受的性能下降時間內從故障中恢復或轉移到輔助源。

Elastic Load Balancing

一種托管的負載均衡服務，可在多個 EC2 實例、容器、IP 位址和 Lambda 函數之間分配傳入的應用程式流量。

- 可以面向外部或面向內部
- 每個負載等化器都會收到一個 DNS 名稱
- 發現並響應運行狀況不佳的實例

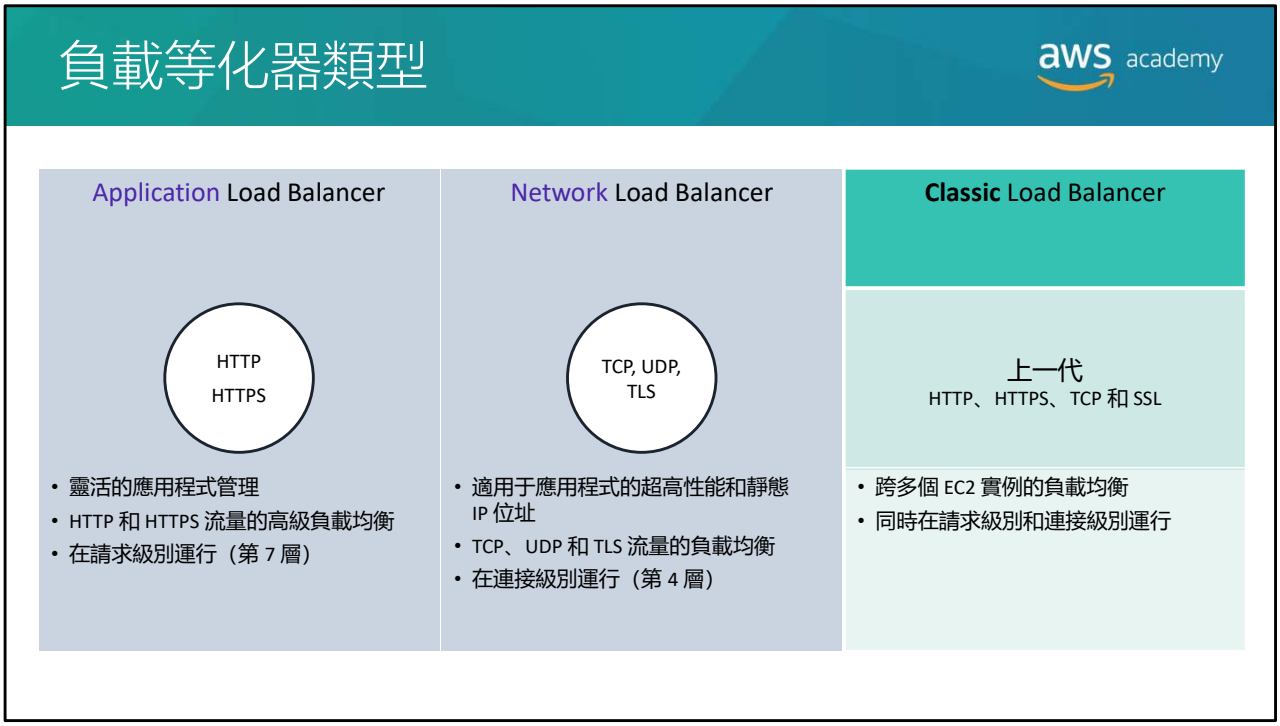
Elastic Load Balancing 是創建高度可用的架構的關鍵元件。

ELB 可在多個目標（如 EC2 實例、容器、IP 位址和 Lambda 函數）之間自動分發傳入的應用程式流量。它可以在單個可用區中或跨多個可用區處理不同的應用程式流量負載。

ELB 提供三種負載等化器。負載等化器可以面向外部並分發入站公共流量。它們還可以面向內部並分發私有流量。

每種負載等化器都會收到一個預設的網域名稱系統 (DNS) 名稱。

最後，ELB 自動將傳入流量分發到多個可用區中的多個目標，並且僅將流量發送到正常運行的目標。為了查明 EC2 實例的可用性，負載等化器會定期發送 ping、嘗試連接或發送請求來測試 EC2 實例。這些測試稱為*運行狀況檢查*。每個註冊的 EC2 實例必須使用 HTTP 狀態碼 200 來回應運行狀況檢查的目標，這樣才能被負載等化器視為運行狀況良好。



ELB 提供三種負載等化器，它們都具有高可用性、自動擴縮功能和強大的安全性，可讓您的應用程式實現容錯。

- Application Load Balancer *在應用程式級別（開放系統互相連線 (OSI) 模型第 7 層）運行。*它根據請求的內容將流量路由到目標 – EC2 實例、容器、IP 位址和 Lambda 函數。它非常適合 HTTP 和安全 HTTP (HTTPS) 流量的高級負載均衡。Application Load Balancer 提供面向現代化應用程式架構交付的高級請求路由，包括微服務和基於容器的應用程式。Application Load Balancer 通過確保始終使用最新的安全通訊端層/傳輸層安全性 (SSL/TLS) 密碼和協定，簡化並提高應用程式的安全性。

- Network Load Balancer *在網路傳輸級別（OSI 模型第 4 層）運行，將連接路由到 EC2 實例、容器和 IP 位址等目標。*它是傳輸控制協議 (TCP) 和使用者資料包通訊協定 (UDP) 流量實現負載均衡的理想之選。Network Load Balancer 能夠在保持超低延遲的同時，每秒處理數百萬個請求。Network Load Balancer 針對處理突發和不穩定的網路流量模式進行了優化。

- Classic Load Balancer *可以在應用程式級別和網路傳輸級別運行，在多個 EC2 實例之間提供基本的負載均衡。*Classic Load Balancer 支援對使用 HTTP、HTTPS、TCP 和 SSL 的應用程式實施負載均衡。Classic Load Balancer 是一種較舊的實施方案。如果可能，AWS 建議您使用專用的 Application Load Balancer 或 Network Load Balancer。

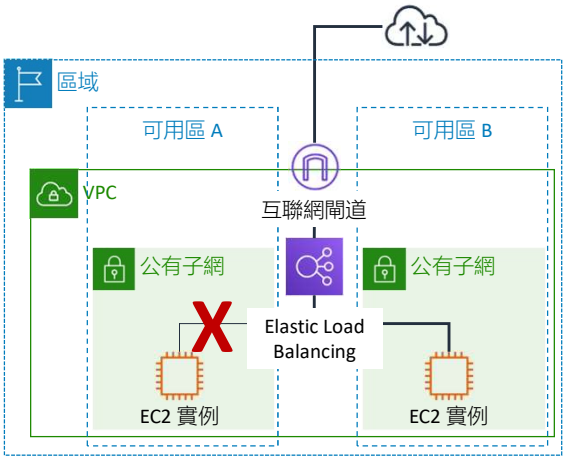
使用 VPC 對等連接，您可以從另一 VPC 訪問內部負載等化器（包括 Classic Load Balancer、Application Load Balancer 和 Network Load Balancer）。VPC 對等連接可用於本地或跨帳戶 VPC 的區域內和區域間連接。

要詳細瞭解三種負載等化器之間的差異，請參閱 Elastic Load Balancing 功能頁面上的*產品比較*。

實現高可用性

從每個 AWS 區域兩個可用區開始。

如果一個可用區中的資源無法訪問，應用程式不會出現故障。



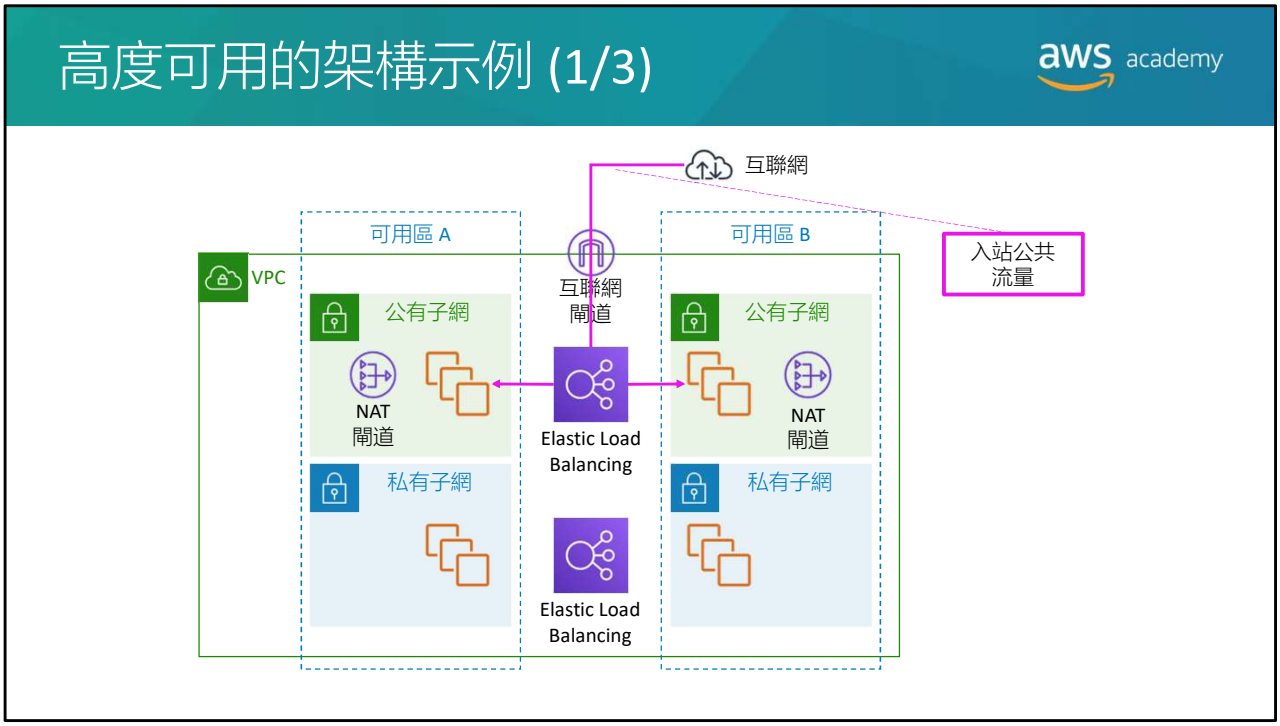
要構建高度可用的應用程式，最佳實踐是在多個可用區中啟動資源，然後使用負載等化器在這些資源之間分配流量。在多個可用區中運行您的應用程式可以在資料中心發生故障時實現更高的可用性。

在此基本模式中，兩台 Web 伺服器在不同可用區中的 EC2 實例上運行。這些實例被置於 Elastic Load Balancing 負載等化器之後，由該負載等化器在實例之間分配流量。如果一台伺服器不可用，負載等化器將配置為停止將流量分配到運行狀況不佳的實例，並開始將流量路由到運行狀況良好的實例。這樣，如果其中一個可用區中的資料中心出現故障，則應用程式仍然可用。

大多數應用程式可以設計為每個 AWS 區域支援兩個可用區。但如果您使用的是僅支援主或輔助容錯移轉的資料來源，那麼使用多個可用區對應用程式而言可能並無益處。由於可用區在物理空間上是分散的，因此在一個 AWS 區域中的三個或更多可用區內複製資源並不會帶來太多好處。

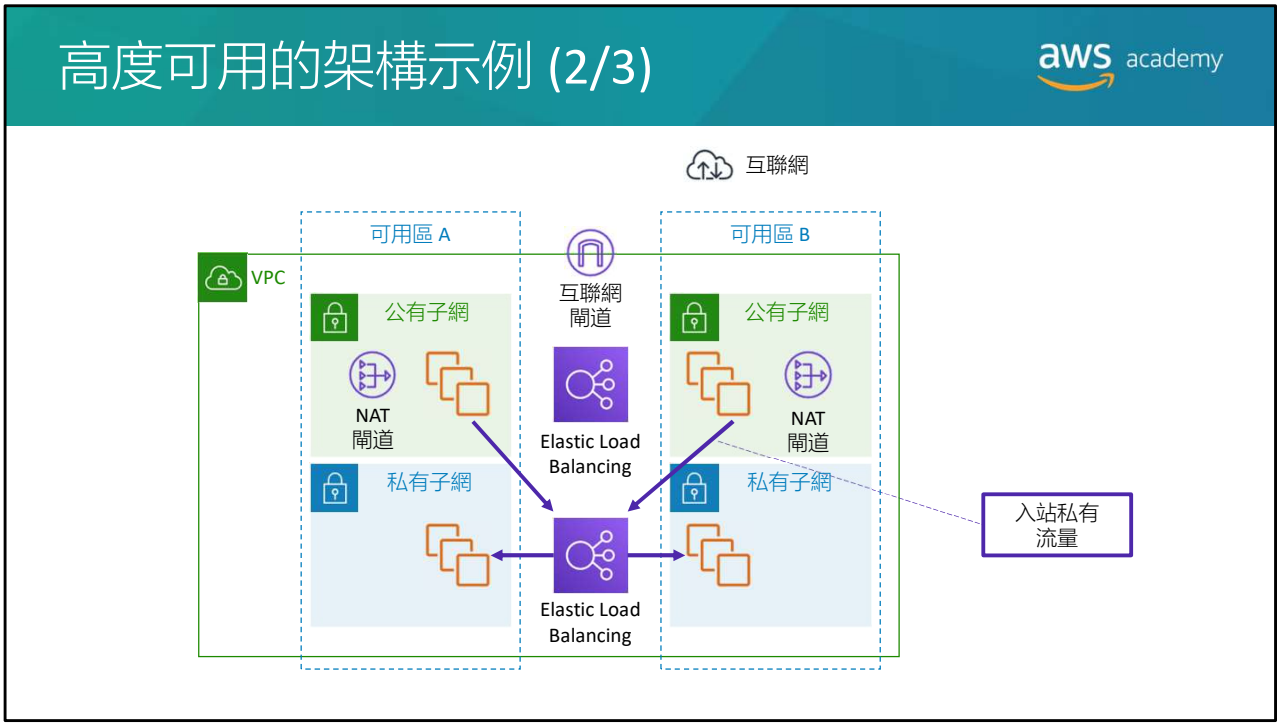
如果您使用大量 Amazon EC2 Spot 實例或使用超出主動或被動範圍的資料來源（例如 Amazon DynamoDB），那麼使用兩個以上的可用區可能會頗有助益。

高度可用的架構示例 (1/3)



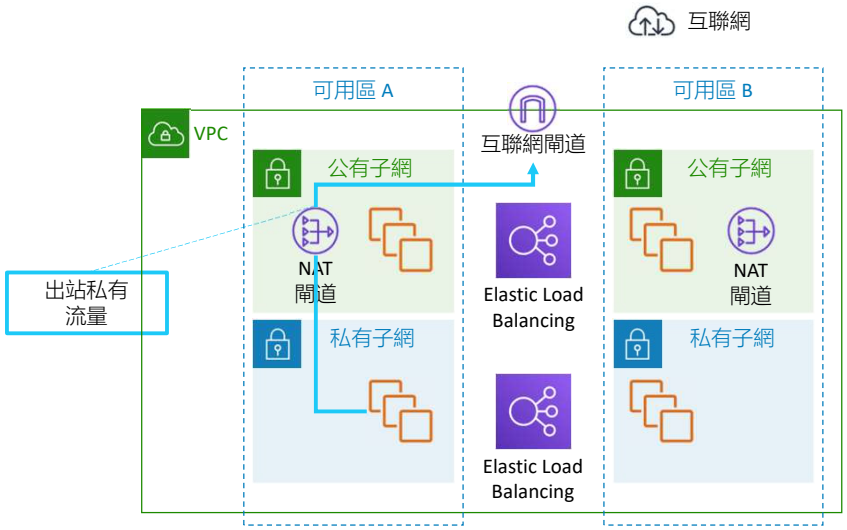
與您剛才考慮的架構一樣，EC2 實例被置於負載等化器之後，由該負載等化器在實例之間分配入站公共流量。如果其中一台伺服器不可用，負載等化器將配置為停止向運行狀況不佳的實例分配流量，然後開始將流量路由到運行狀況良好的實例。

高度可用的架構示例 (2/3)



您可以在架構中加入第二個負載等化器，將入站流量從公有子網中的實例路由到私有子網中的實例。

高度可用的架構示例 (3/3)

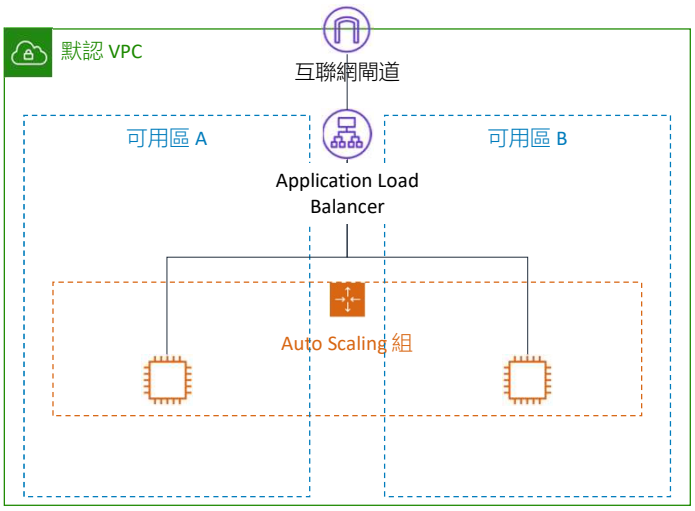


如果您在多個可用區中擁有資源並且它們共用一個 NAT 閘道，則當該 NAT 閘道的可用區不可用時，其他可用區中的資源將無法訪問互聯網。最佳實踐是在兩個可用區中使用 NAT 閘道以確保高可用性。

演示：
創建高度可用的 Web 應用程式

現在，講師可能會選擇演示如何通過跨多個可用區在 Application Load Balancer 之後部署 Web 伺服器來創建高度可用的 Web 應用程式。

演示架構



本演示展示了如何：

1. 創建並啟動 EC2 Web 伺服器
2. 通過 EC2 Web 伺服器創建 Amazon 系統映射 (AMI)
3. 創建和配置 Application Load Balancer
4. 創建和配置 Auto Scaling 組並將其部署到兩個可用區
5. 測試 Application Load Balancer

Amazon Route 53

Amazon Route 53 是一種高度可用且可擴展的雲 DNS 服務。

- 將功能變數名稱轉換為 IP 位址
- 將用戶請求連接到在 AWS 內部和外部運行的基礎設施
- 可以配置為將流量路由到運行狀況良好的終端節點，或監控應用程式及其終端節點的運行狀況
- 提供功能變數名稱註冊
- 有多個路由選項

您還可以使用 Amazon Route 53 在網路架構中實施多區域高可用性和容錯能力。

Amazon Route 53 是一種高度可用且可擴展的雲 DNS 服務。該服務旨在提供一種可靠且經濟高效的方式，以將使用者路由到互聯網應用程式。它將名稱（如 *example.com*）轉換為電腦相互連接所用的數位 IP 位址（如 *192.0.2.1*）。

Route 53 能有效地將用戶請求連接到在 AWS 中運行的基礎設施上，例如 EC2 實例、ELB 負載等化器或 S3 存儲桶。您也可以使用 Route 53 將用戶路由到 AWS 之外的基礎設施上。

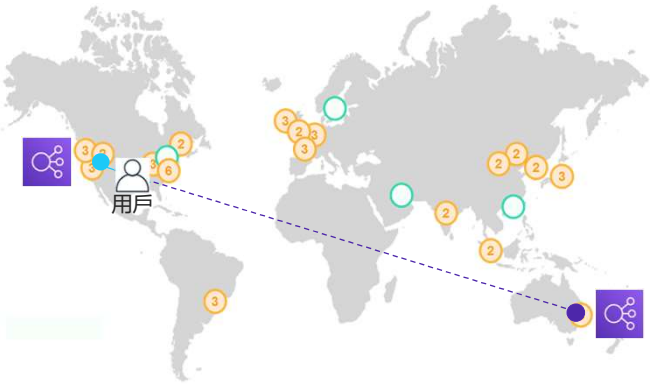
您可以使用 Route 53 配置 DNS 運行狀況檢查以將流量路由到運行狀況良好的終端節點，或者獨立監控應用程式及其終端節點的運行狀況。

Route 53 還提供功能變數名稱註冊功能。您可以購買和管理功能變數名稱（如 *example.com*），而 Amazon Route 53 將為您的域自動配置 DNS 設置。

Amazon Route 53 提供了各種路由選項，這些選項可與 DNS 容錯移轉相結合，以實現低延遲容錯架構。有關 Amazon Route 53 路由選項的詳細資訊，請參閱選擇路由策略。

Amazon Route 53 支持的路由

- 簡單路由
- 加權輪詢路由
- 基于延遲的路由
- 地理位置路由
- 地理位置臨近度路由
- 故障轉移路由
- 多值應答路由



Amazon Route 53 支持多種類型的路由策略，這些策略可確定 Amazon Route 53 如何回應查詢：

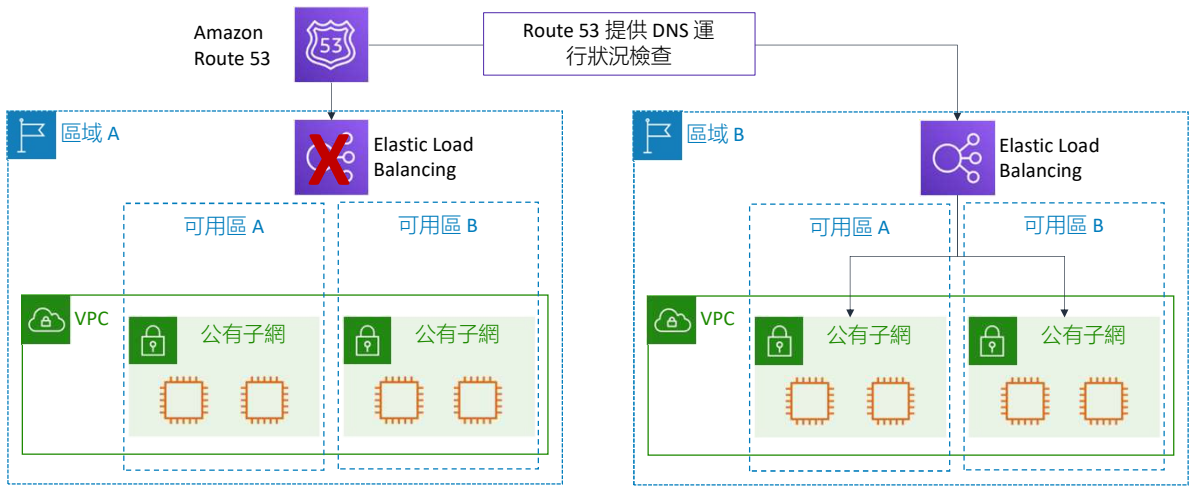
- *簡單路由（輪詢）* – 可在所有參與伺服器之間盡可能均勻地分配請求數。
- *加權輪詢路由* – 允許您為資源記錄集分配權重，以便指定提供不同回應的頻率。您可以使用此功能進行 A/B 測試，將一小部分流量發送到您已更改軟體的伺服器。例如，假設您向一個 DNS 名稱關聯了兩個記錄集：一個權重為 3，另一個權重為 1。在本例中，75% 的時間內 Amazon Route 53 將返回權重為 3 的記錄集，25% 的時間內 Amazon Route 53 將返回權重為 1 的記錄集。權重可以是 0 到 255 之間的任意數字。
- *延遲路由 (LBR)* – 如果您的資源位於多個 AWS 區域，並且您想要將流量路由到提供最佳延遲的區域，則可以使用該策略。延遲路由的工作原理是將您的客戶路由到 AWS 終端節點（如 EC2 實例、彈性 IP 位址 或負載等化器），以便根據運行應用程式的不同 AWS 區域的實際性能測量結果提供最快的體驗。
- *地理位置路由* – 允許您根據使用者的地理位置（DNS 查詢的來源）選擇提供流量的資源。使用地理位置路由時，您可以對您的內容進行當地語系化，並使用使用者的語言顯示您的部分或全部網站。您也可以使用地理位置路由將內容分配限制為僅分配至具有分配許可權的位置。另一種可能的用途是以可預測、易於管理的方式在終端節點間均衡負載，以便每個使用者位置一致地路由到同一終端節點。

- *地理位置臨近度路由* – 如果您使用的是 Route 53 流量，則允許您根據使用者與資源之間的物理距離來路由流量。您還可以通過指定正偏差或負偏差來為每個資源路由更多或更少的流量。 創建流量策略時，您可以為每個終端節點指定 AWS 區域（如果您正在使用 AWS 資源）或緯度和經度。

- *容錯移轉路由（DNS 容錯移轉）* – 如果您想配置主動-被動容錯移轉，則可以使用該策略。Route 53 可以幫助檢測網站是否中斷，並將使用者重定向到應用程式正常運行的備用位置。啟用此功能後，Route 53 運行狀況檢查代理將監控應用程式的每個網站或終端節點，以確定其可用性。您可以利用此功能來提高面向客戶的應用程式的可用性。

- *多值應答路由* – 如果您想將流量大致隨機路由到多個資源（如 Web 伺服器），則使用此功能。您可以為每個資源創建一條多值應答記錄。您還可以選擇將 Route 53 運行狀況檢查與每條記錄相關聯。例如，假設您管理著一項使用 12 台 Web 伺服器的 HTTP Web 服務，且每台 Web 伺服器都有自己的 IP 地址。沒有一台 Web 伺服器可以處理所有流量。但如果您創建了十幾條多值應答記錄，則 Route 53 可回應最多具有 8 條運行狀況良好記錄的 DNS 查詢，從而能夠回應每個 DNS 查詢。Route 53 可為不同的 DNS 解析程式提供不同的應答。如果解析程式緩存回應後 Web 伺服器變得不可用，用戶端軟體可以嘗試回應中的其他 IP 位址。

多區域高可用性和 DNS



通過 *DNS 容錯移轉路由*， Route 53 可以幫助檢測網站是否中斷，並將使用者重定向到應用程式正常運行的備用位置。啟用此功能後，Route 53 運行狀況檢查代理將監控應用程式的每個網站或終端節點，以確定其可用性。您可以利用此功能來提高面向客戶的應用程式的可用性。

演示：
Amazon Route 53

現在，講師可能會選擇播放視頻，演示如何使用 Route 53 進行簡單路由、容錯移轉路由和地理位置路由。

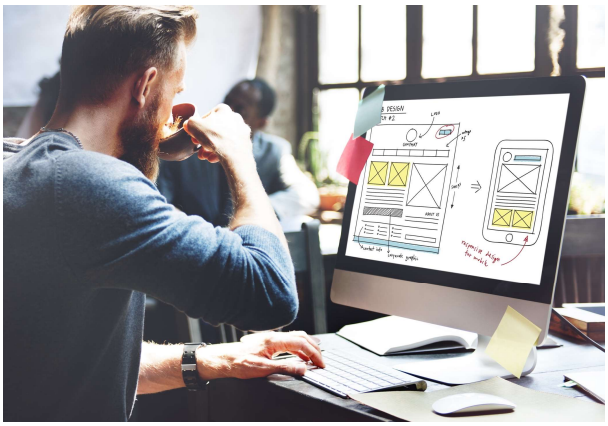
第 4 節要點

- 您可以將網路架構設計為高度可用，避免單點故障
- Route 53 提供了各種路由選項，可與 DNS 容錯移轉結合使用，以實現低延遲容錯架構

本模組中這節內容的要點包括：

- 您可以將網路架構設計為高度可用，避免單點故障
- Route 53 提供了多種路由選項，這些選項可與 DNS 容錯移轉相結合，以實現多種低延遲容錯架構

模組 9 – 指導實驗：創建高度可用的環境



現在，您要完成“模組 9 – 指導實驗：創建高度可用的環境”。

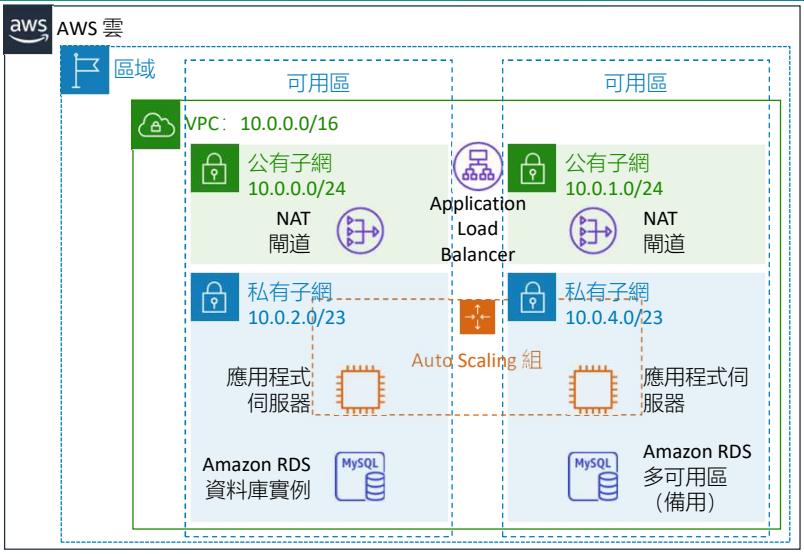
指導實驗：任務

1. 檢查提供的 VPC
2. 創建 Application Load Balancer
3. 創建 Auto Scaling 組
4. 測試應用程式是否實現了高可用性

在本指導實驗中，您將完成以下任務：

1. 檢查提供的 VPC
2. 創建 Application Load Balancer
3. 創建 Auto Scaling 組
4. 測試應用程式是否實現了高可用性

指導實驗：最終產品

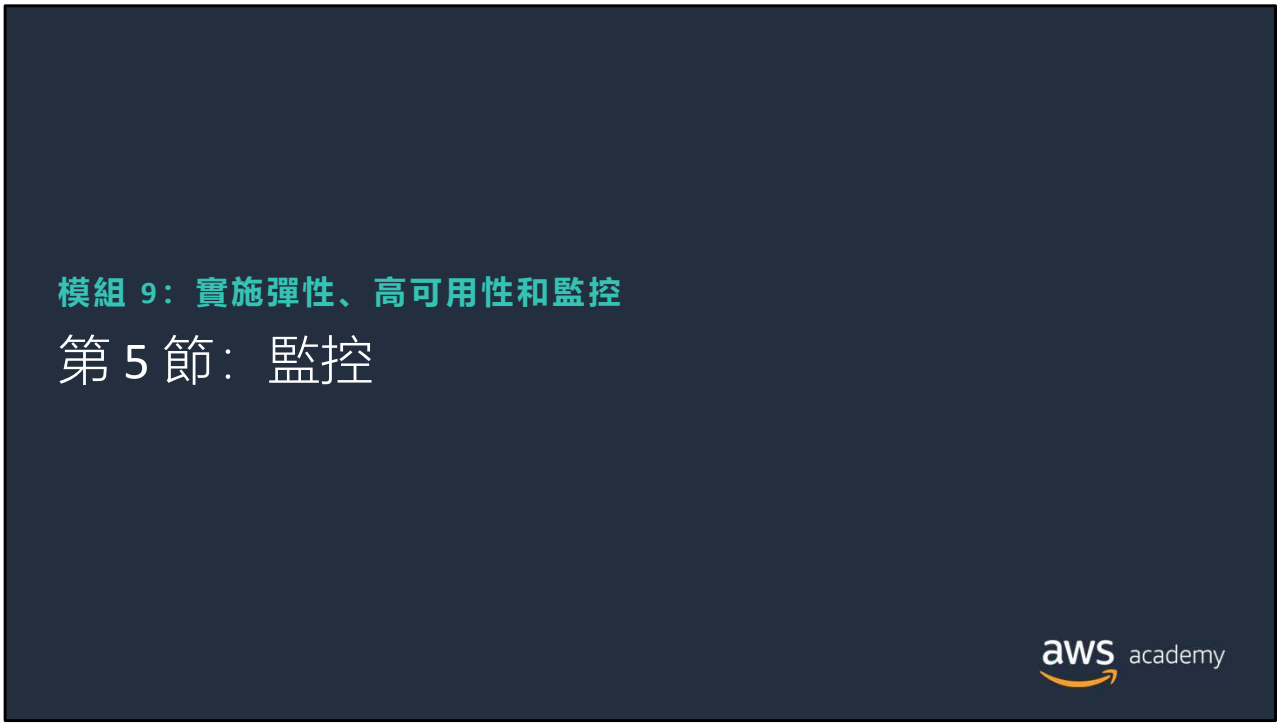


該圖總結了您要在實驗中構建的內容。

現在可以開始指導實驗了。

指導實驗總結：要點

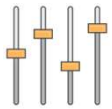
完成這個指導實驗之後，講師可能會帶您討論此指導實驗的要點。



介紹第 5 節：監控。

監控使用情況、操作和性能

運行狀況



資源利用率

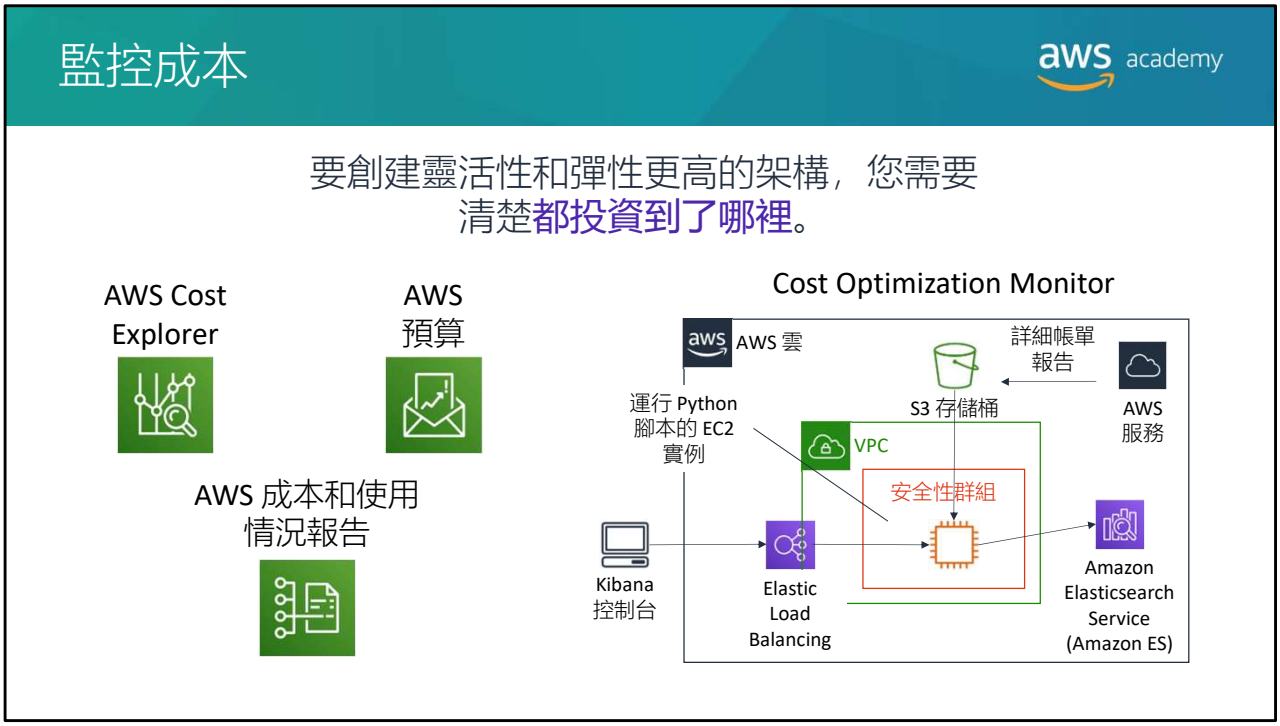
應用程式性能

安全審計

監控是反應式架構的重要組成部分。

監控可以幫助您：

- 跟蹤資源的運行和執行情況
- 跟蹤資源利用率和應用程式性能，以確保您的基礎設施能夠滿足需求
- 確定為您的 AWS 資源設置哪些許可權以實現所需的安全目標

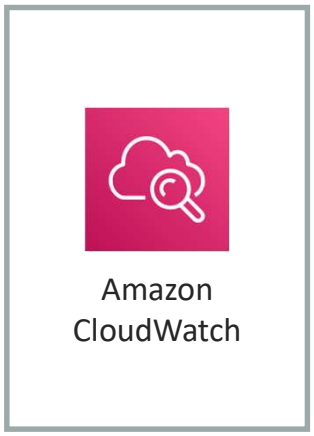


監控還有助於您瞭解和管理 AWS 基礎設施的成本。AWS 提供多種監控和報告工具，其中包括：

- AWS Cost Explorer 可幫助您按天或按月直觀地查看、理解和管理 AWS 成本和使用情況。您可以使用它查看過去長達 13 個月的資料，從而瞭解您在某段時間內使用 AWS 資源的模式。
- AWS 預算使您能夠設置自訂預算，以便在成本或用量超過（或預計會超過）您的預算數量時向您發出警報。
- AWS 成本和使用情況報告包含最全面的一組 AWS 成本和使用情況資料，其中包括有關 AWS 服務、定價和預留的中繼資料。
- Cost Optimization Monitor 是一種解決方案架構，可自動處理詳細的帳單報告，以提供可在自訂控制台中進行搜索、分析和實現視覺化的精細指標。該解決方案可以讓您深入瞭解服務使用情況和成本，您可以按週期、帳戶、資源或標籤進行細分。

要瞭解有關如何監控 AWS 基礎設施成本的更多資訊，請參閱 AWS 成本管理服務。

Amazon CloudWatch



- 收集並跟蹤資源和應用程式的指標
- 幫助您關聯、視覺化和分析指標和日誌
- 支持您創建警報並檢測異常行為
- 可以發送通知或更改您正在監控的資源

Amazon CloudWatch 是一種面向開發運維工程師、開發人員、網站可靠性工程師和 IT 經理的監控和可觀測性服務。CloudWatch 為您提供相關資料和可行見解，以監控應用程式、回應系統範圍的性能變化、優化資源利用率，並在統一視圖中查看運營狀況。

您可以使用 CloudWatch 收集和跟蹤指標，這些指標是您用於衡量資源和應用程式的變數。您可以創建用於監控指標並發送通知的 CloudWatch 警報。此外，當超過閾值時，CloudWatch 可以自動更改您正在監控的資源。

例如，您可以監控 CPU 使用情況以及 EC2 實例的磁片讀取和寫入情況。然後，您可以使用這些資料來確定是否應啟動其他實例來處理增加的負載。您還可以使用這些資料停止未完全利用的實例以節省資金。

除了監控 AWS 隨附的內置指標外，您還可以監控自己的自訂指標。您可以使用 CloudWatch 全面瞭解資源使用率、應用程式性能和運行狀況。

有關 CloudWatch 的更多資訊，請參閱什麼是 Amazon CloudWatch?

CloudWatch 如何回應


指標


日誌


警報


事件


規則


目標

您可以使用多個 CloudWatch 元件來監控資源和應用程式，並回應事件。

CloudWatch 指標


指標

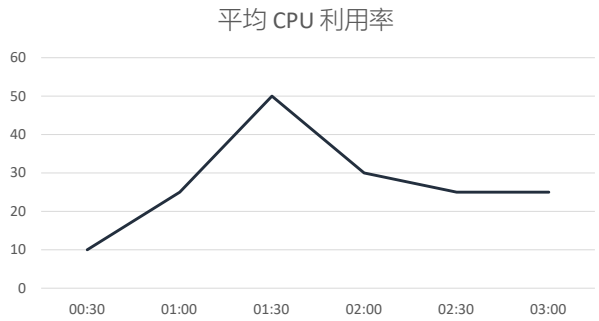

日誌


警報


事件


規則


目標

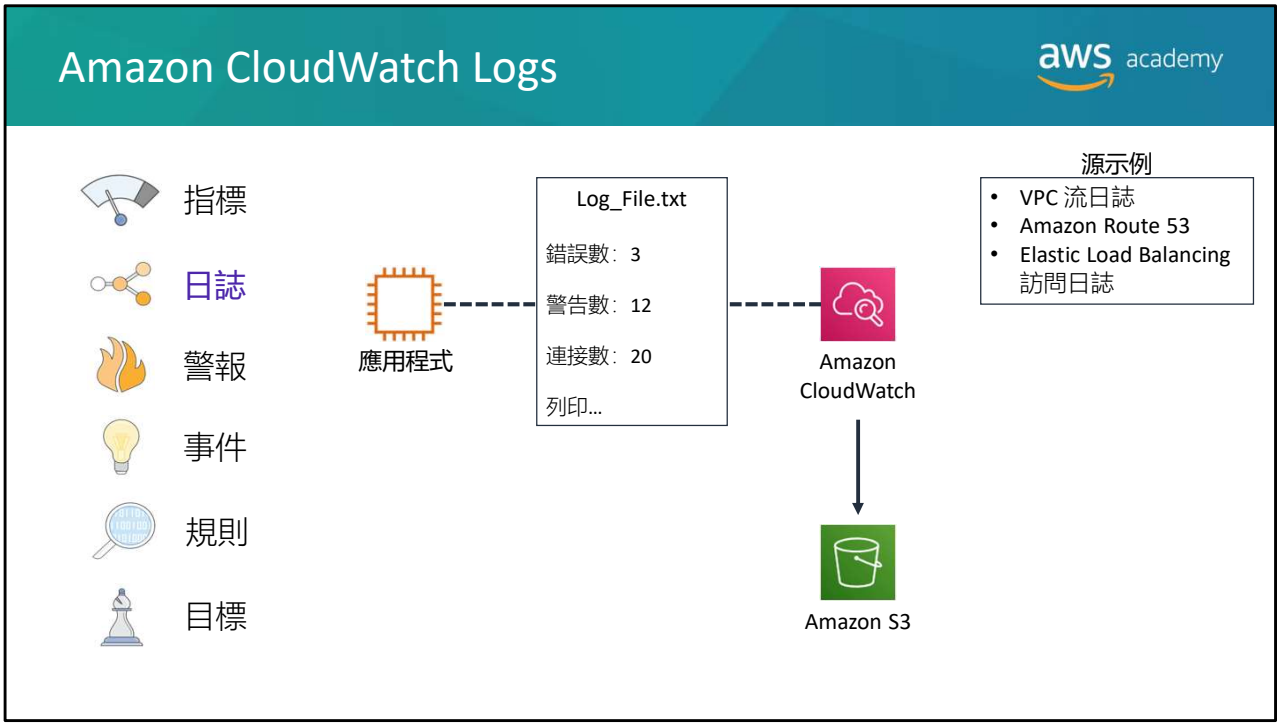


指標資料將保留 15 個月

*指標*是與系統性能有關的資料。預設情況下，許多 AWS 服務可提供資源指標，例如 EC2 實例、Amazon Elastic Block Store (Amazon EBS) 卷和 Amazon RDS 資料庫實例。此外，您還可以為某些資源啟用詳細監控（例如 EC2 實例），或發佈您自己的應用程式指標。CloudWatch 可以載入您帳戶中的所有指標（包括 AWS 資源指標和您提供的應用程式指標），用於搜索、繪圖和發送警報。

指標資料的保留期限為 15 個月，這使您能夠查看最新資料和歷史資料。

有關指標的更多資訊，請參閱使用 Amazon CloudWatch 指標。



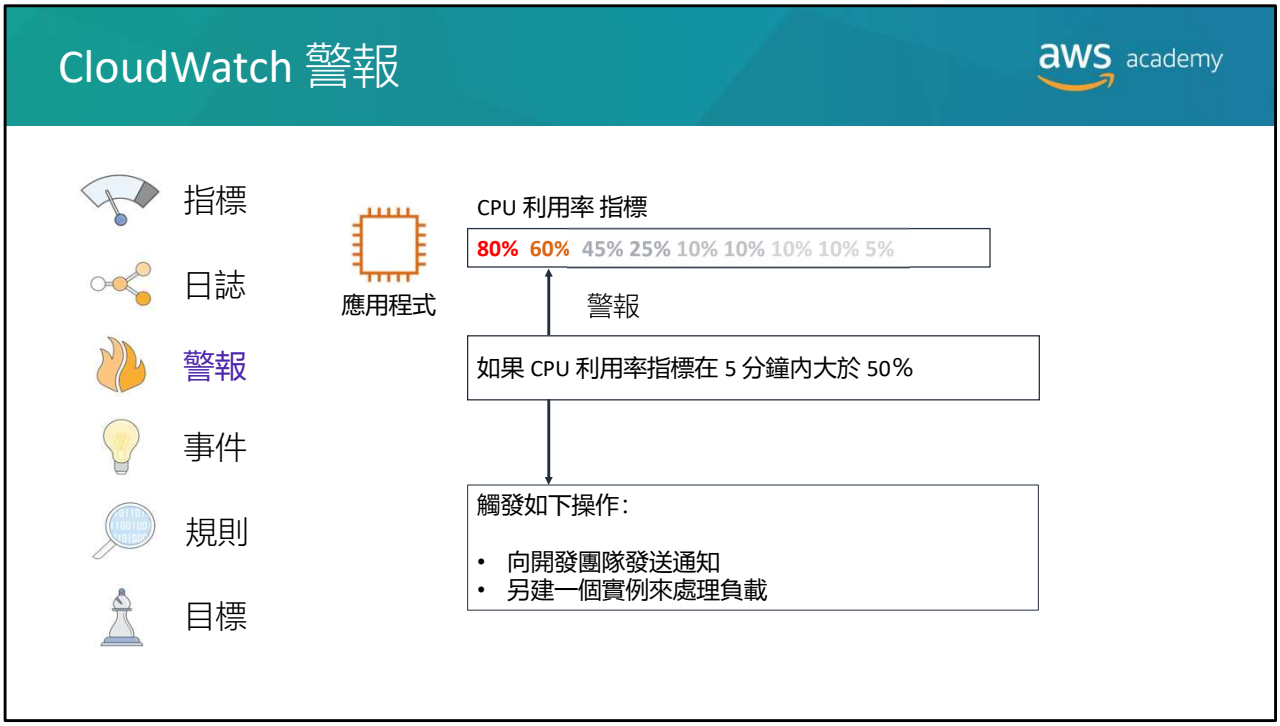
您可以使用 Amazon CloudWatch Logs 監控、存儲和訪問來自 EC2 實例、AWS CloudTrail、Route 53 和其他 AWS 服務等來源的日誌檔。

例如，您可以使用 CloudWatch Logs 通過日誌資料監控應用程式和系統。CloudWatch Logs 可以跟蹤應用程式日誌中發生的錯誤數。當錯誤率超過您指定的閾值時，它將發送通知。

此外，您可以使用 CloudWatch Logs Insights 在幾秒鐘內分析日誌。它為您提供快速的互動式查詢和視覺化效果。您可以使用折線圖或堆疊面積圖直觀呈現查詢結果，並將這些查詢添加到 CloudWatch 控制台。

有關 CloudWatch Logs 的更多資訊，請參閱以下資源：

- Amazon CloudWatch Logs
- Amazon CloudWatch Logs Insights



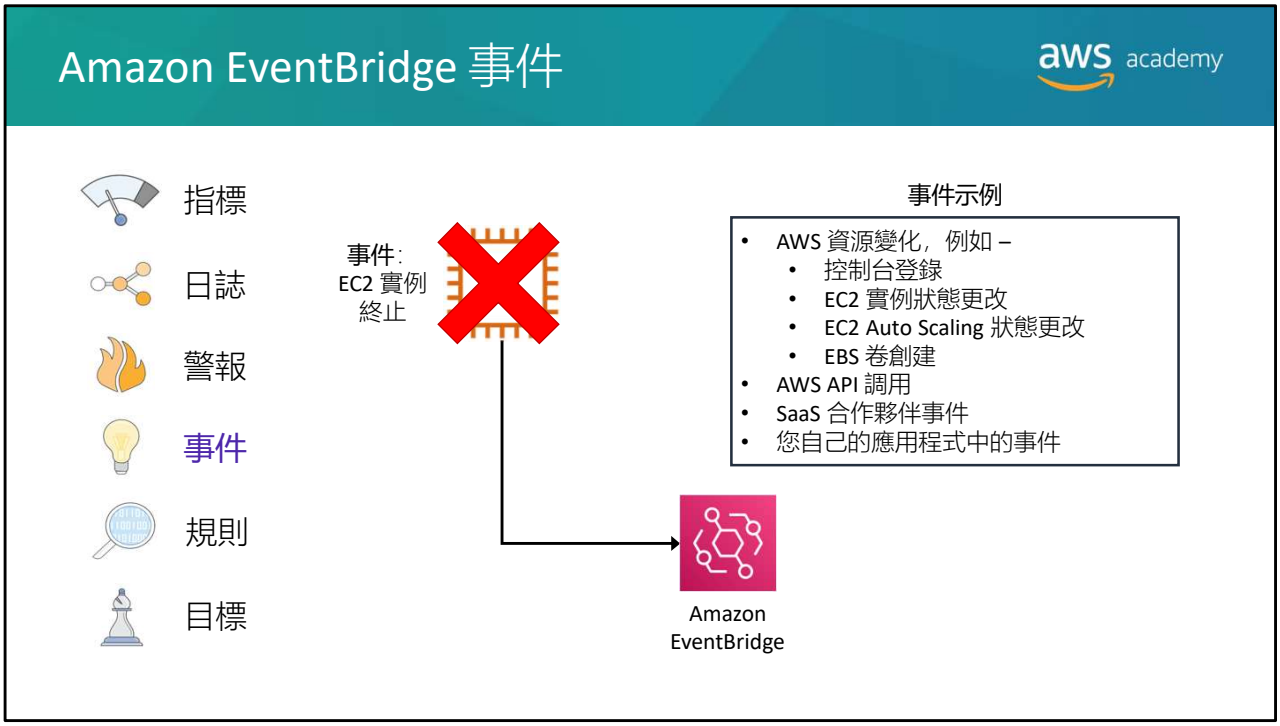
您可以使用警報代您自動發起操作。警報*會在指定時間段內監控單個指標*。它根據相對於某個閾值的指標值在一段時間執行一項或多項指定操作。操作指的是發送至 Amazon SNS 主題或 Auto Scaling 策略的通知。您還可以將警報添加到控制台。

警報僅在出現持續的狀態更改時才會調用操作。CloudWatch 警報不會僅因操作處於某個狀態而調用它們。狀態必須已改變並已維持了指定的若干個時間段。

在幻燈片所示的示例中，當 CPUUtilization 指標（也即當前正用於實例的已分配 EC2 計算單元的百分比）大於 50% 並持續 5 分鐘時將觸發警報。警報會觸發操作，例如運行 Auto Scaling 策略或向開發團隊發送通知。

未觸發警報時也可以執行操作。

有關 CloudWatch 警報的更多資訊，請參閱使用 Amazon CloudWatch 警報。

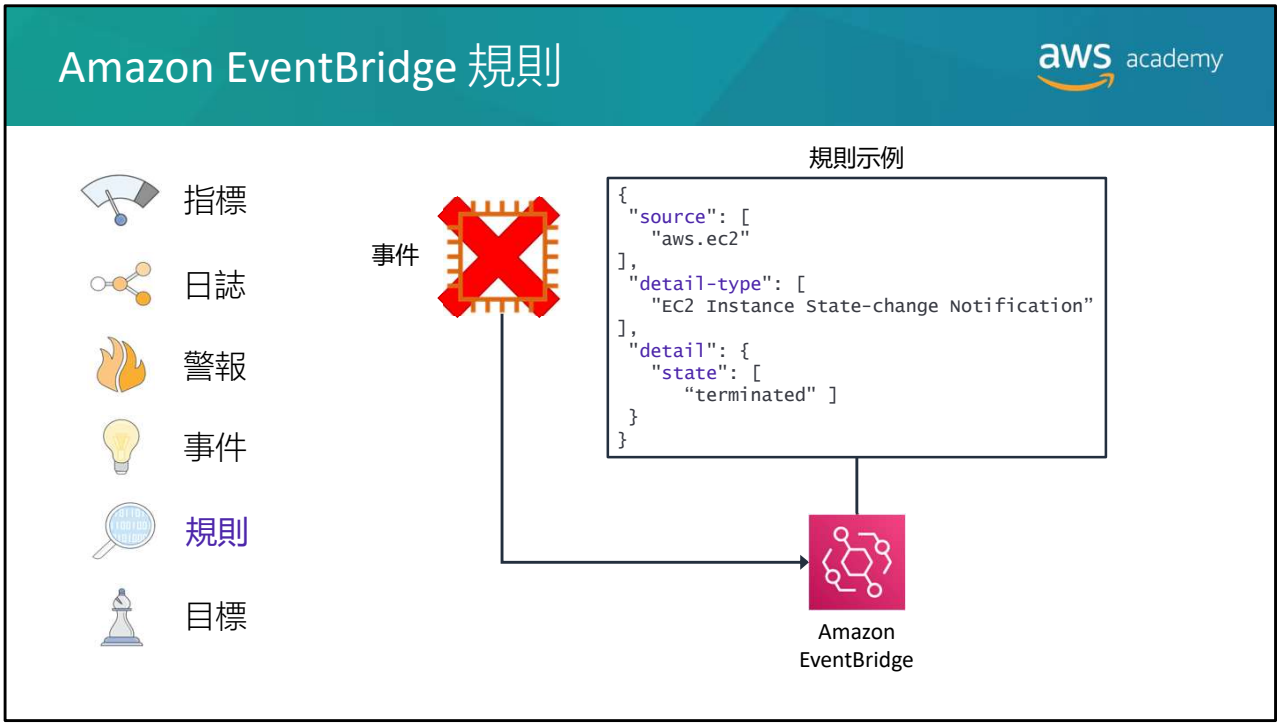


Amazon EventBridge（前身為 Amazon CloudWatch Events）可從您自己的應用程式、軟體即服務 (SaaS) 應用程式和 AWS 服務中提取即時資料流。然後，它將資料路由到目標（如 AWS Lambda）。

事件*表示環境發生的變化*。這可以是 AWS 環境、SaaS 合作夥伴服務或應用程式，或您自己的自訂應用程式或服務之一。例如，Amazon EC2 會在 EC2 實例的狀態從*等待中*更改為*運行中*時生成事件，Amazon EC2 Auto Scaling 會在啟動或終止實例時生成事件。 AWS CloudTrail 在您進行 API 調用時發佈事件。您還可以設置定期生成的計畫事件。

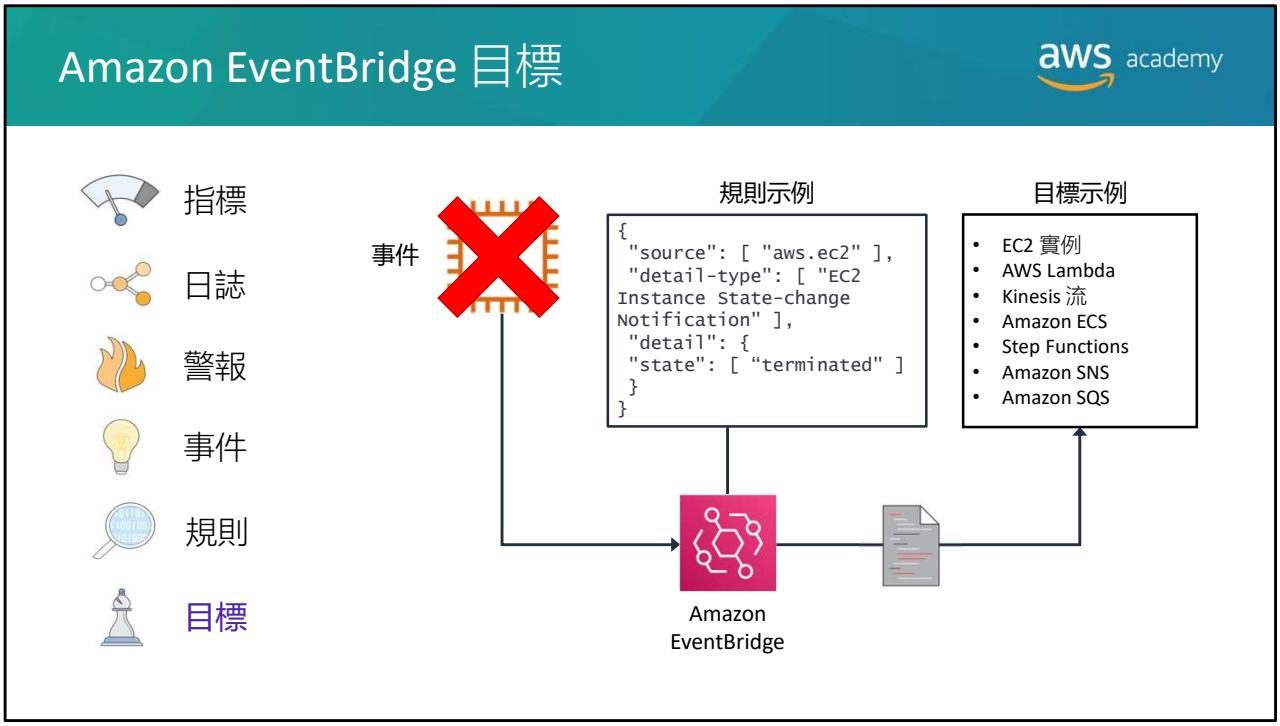
現有的 CloudWatch Events 用戶可以在新的 EventBridge 控制台和 CloudWatch Events 控制台中訪問其現有預設匯流排、規則和事件。EventBridge 使用相同的 CloudWatch Events API，因此現有 CloudWatch Events API 的所有使用方式均保持不變。

有關 EventBridge 的更多資訊，請參閱什麼是 Amazon EventBridge?



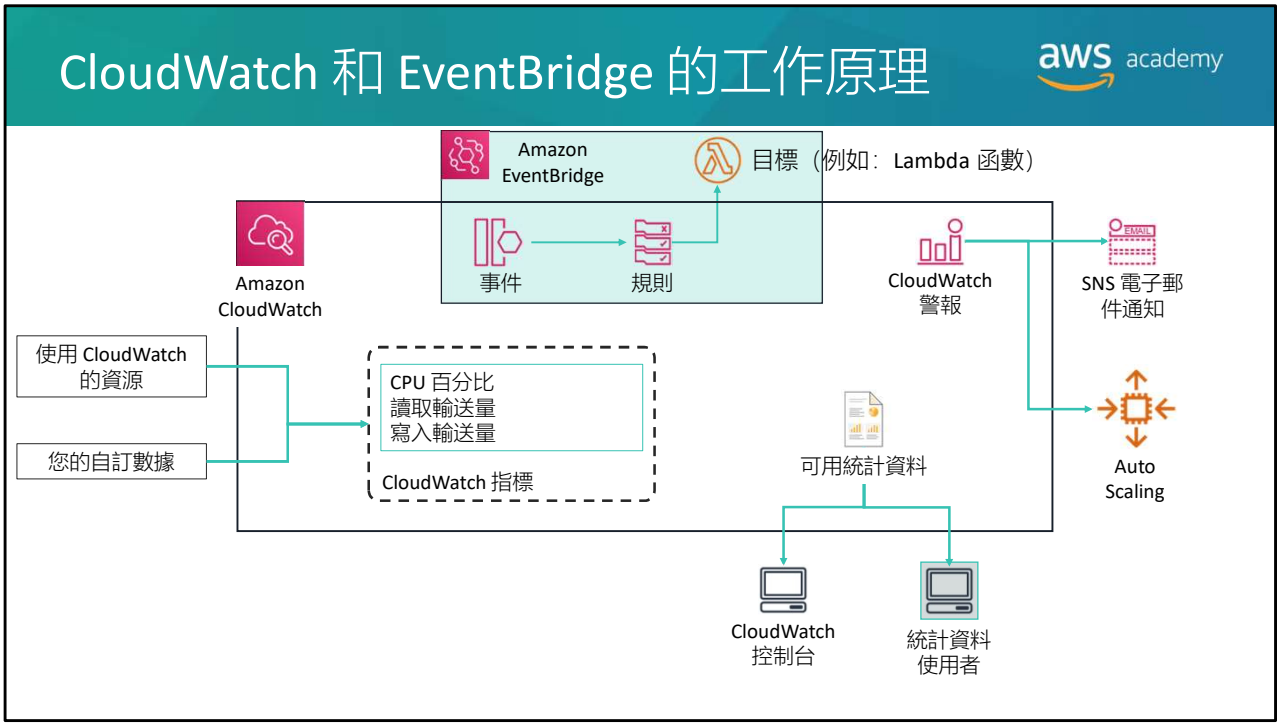
您可以設置路由規則來確定資料發送目的地，以構建能夠即時回應所有資料來源的應用程式架構。

規則*會匹配傳入事件並將它們路由到目標以進行處理。*單個規則可以路由到多個目標，所有目標都是並行處理的。規則並不是按特定順序進行處理的。因此，組織的不同部門能夠查找和處理他們感興趣的事件。規則可以自訂發送到目標的 JavaScript 物件標記法 (JSON) 消息，方法是僅傳遞某些部分或使用常量進行覆蓋。



目標*負責處理事件*。目標包括 EC2 實例、Lambda 函數、Amazon Kinesis Streams、Amazon Elastic Container Service (Amazon ECS) 任務、AWS Step Functions 狀態機、SNS 主題、Amazon Simple Queue Service (Amazon SQS) 佇列和內置目標。目標接收 JSON 格式的事件。

創建規則時，將其與特定*事件匯流排*關聯，並且該規則僅與該事件匯流排接收的事件匹配。



此架構在宏觀層面展示了 CloudWatch 和 EventBridge 的工作方式。

CloudWatch 充當指標存儲庫。AWS 服務（如 Amazon EC2）會將指標放在存儲庫中，您可以根據這些指標檢索統計資訊。如果您將自己的自訂指標放在存儲庫中，則還可以檢索有關這些指標的統計資料。

您可以在 CloudWatch 控制台中使用指標計算統計資料，然後以圖形化的方式顯示資料。

您可以在 EventBridge 中創建匹配傳入事件並將其路由到目標進行處理的規則。

在滿足特定條件時，您可以配置警報操作以停止、啟動或終止 EC2 實例。此外，您還可以創建警報代您啟動 Amazon EC2 Auto Scaling 和 Amazon SNS 操作。

第 5 節要點

aws academy

- AWS Cost Explorer、AWS 預算、AWS 成本和使用情況報告和 Cost Optimization Monitor 可以幫助您瞭解和管理 AWS 基礎設施的成本。
- CloudWatch 以日誌、指標和事件的形式收集監控和運營資料。它使用自動化控制台對資料進行視覺化，以便您統一查看在 AWS 和本地運行的 AWS 資源、應用程式和服務。
- EventBridge 是一種無伺服器事件匯流排服務，通過它可以將應用程式與來自各種來源的資料連接起來。EventBridge 可傳輸來自您自己的應用程式、SaaS 應用程式和 AWS 服務的即時資料流，然後將這些資料路由到目標。

本模組中這節內容的要點包括：

- AWS Cost Explorer、AWS 預算、AWS 成本和使用情況報告和 Cost Optimization Monitor 可以說明您瞭解和管理 AWS 基礎設施的成本。
- CloudWatch 以日誌、指標和事件的形式收集監控和運營資料。它使用自動化控制台對資料進行視覺化，以便您統一查看在 AWS 和本地運行的 AWS 資源、應用程式和服務。
- EventBridge 是一項無伺服器事件匯流排服務，通過它可以輕鬆將應用程式與來自各種來源的資料相連。EventBridge 從您自己的應用程式、SaaS 應用程式和 AWS 服務中提取即時資料流。然後將這些資料路由到目標。

模組 9 – 挑戰實驗：為咖啡館創建可擴展且高度可用的環境

您現在要完成“模組 9 – 挑戰實驗：為咖啡館創建可擴展且高度可用的環境”。

業務需求：可擴縮且高度可用的環境

- 咖啡館很快將在著名的電視美食節目中亮相。
- Sonía 和 Nikhil 想要確保咖啡館的網站能夠應對預期的流量增長。

咖啡館很快將在著名的電視美食節目中亮相。當節目播出時，Sofía 和 Nikhil 預計咖啡館 Web 伺服器的用戶數量將出現短暫的激增，甚至可能達到數萬人。目前，咖啡館的 Web 伺服器部署在一個可用區，他們擔心伺服器無法應對預期的流量增長。他們希望確保客戶在訪問網站時擁有出色的體驗，不會遇到任何問題，例如下單延遲或延誤。

為了確保提供這種體驗，網站必須迅速回應，通過擴縮來滿足不斷變化的客戶需求，並做到高度可用。它還必須包含負載均衡。此架構必須跨多個應用程式伺服器分發客戶訂單請求，以便應對需求的增加，而不是讓單個伺服器超載。

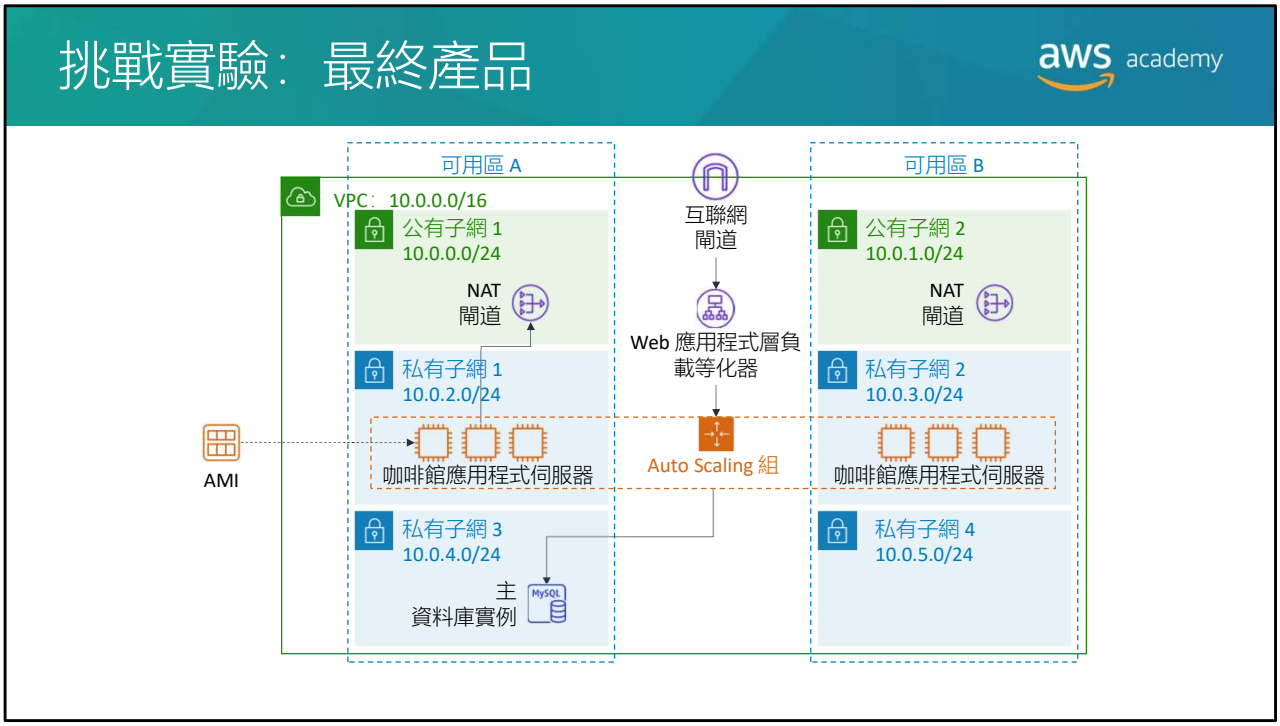
挑戰實驗：任務

1. 為第二個可用區創建 NAT 閘道
2. 在公有子網中創建堡壘主機實例
3. 創建啟動範本
4. 創建 Auto Scaling 組
5. 創建負載等化器
6. 測試 Web 應用程式
7. 測試負載下的自動擴縮

在本挑戰實驗中，您將完成以下任務：

1. 為第二個可用區創建 NAT 閘道
2. 在公有子網中創建堡壘主機實例
3. 創建啟動範本
4. 創建 Auto Scaling 組
5. 創建負載等化器
6. 測試 Web 應用程式
7. 測試負載下的自動擴縮

挑戰實驗：最終產品



該圖總結了您完成實驗後將會構建的內容。

現在可以開始挑戰實驗了。

挑戰實驗總結：要點

完成這個挑戰實驗之後，您的講師可能會帶您討論此挑戰實驗的要點。

模組 9：實施彈性、高可用性和監控

模組總結

現在來回顧下本模組，並對知識測驗和對實踐認證考試問題的討論進行總結。

模組總結

aws academy

總體來說，您在本模組中學習了如何：

- 在架構中使用 Amazon EC2 Auto Scaling 來提高彈性
- 說明如何擴縮資料庫資源
- 部署 Application Load Balancer 以創建高度可用的環境
- 使用 Amazon Route 53 進行 DNS 容錯移轉
- 創建高度可用的環境
- 設計能夠使用 Amazon CloudWatch 監控資源並做出相應反應的架構

總體來說，您在本模組中學習了如何：

- 在架構中使用 Amazon EC2 Auto Scaling 來提高彈性
- 說明如何擴縮資料庫資源
- 部署 Application Load Balancer 以創建高度可用的環境
- 使用 Amazon Route 53 進行 DNS 容錯移轉
- 創建高度可用的環境
- 設計能夠使用 Amazon CloudWatch 監控資源並做出相應反應的架構

完成知識測驗

現在可以完成本模組的知識測驗。

樣題

aws academy

Web 應用程式使客戶能夠將訂單上傳到 S3 存儲桶。生成的 Amazon S3 事件會觸發一個 Lambda 函數，該函數會將消息插入 SQS 佇列。單個 EC2 實例會從佇列中讀取消息，然後處理消息並將其存儲在按唯一訂單 ID 分區的 DynamoDB 表中。預計下個月的流量將增加 10 倍，解決方案架構師正在審查架構是否存在可能的擴縮問題。

哪個元件最有可能需要重新架構才能擴展以容納新的流量？

A. Lambda 函數
B. SQS 佇列
C. EC2 實例
D. DynamoDB 表

請查看答案選項，並根據之前突出顯示的關鍵字排除錯誤選項。

***正確答案是 C：** EC2 實例。* 單個 EC2 實例無法擴縮，是架構中的單點故障。更好的解決方案是將 EC2 實例放在兩個可用區的 Auto Scaling 組中，使其從佇列中讀取消息。其他答案都是可配置為擴縮或自動擴縮的託管服務。

其他資源

- Set it and Forget it: Auto Scaling Target Tracking Policies
- Amazon Elastic Load Balancer 簡介 – 應用
- 使用 ELB Elastic Load Balancer 配置 Auto Scaling 組
- 什麼是 Application Load Balancer?

如果您想瞭解有關本模組所涵蓋主題的更多資訊，下面這些其他資源可能會有所幫助：

- Set it and Forget it: Auto Scaling Target Tracking Policies
- Amazon Elastic Load Balancer 簡介 – 應用
- 使用 ELB Elastic Load Balancer 配置 Auto Scaling 組
- 什麼是 Application Load Balancer?

感謝您完成本模組的學習。