歡迎學習“模組 11：緩存內容”。

模組概覽

本模組包含以下章節：

1. 架構需求
2. 緩存概覽
3. 邊緣緩存
4. 緩存 Web 會話
5. 緩存資料庫

該模組還包括一個指導實驗，您將在實驗中學習如何使用 Amazon CloudFront 資料流動態內容。

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：

- 確定如何通過緩存內容提升應用程式性能並減少延遲
- 確定如何設計通過邊緣網站實現分配和分散式拒絕服務 (DDoS) 保護的架構
- 創建使用 Amazon CloudFront 來緩存內容的架構
- 識別會話管理與緩存的關係
- 描述如何設計使用 Amazon ElastiCache 的架構

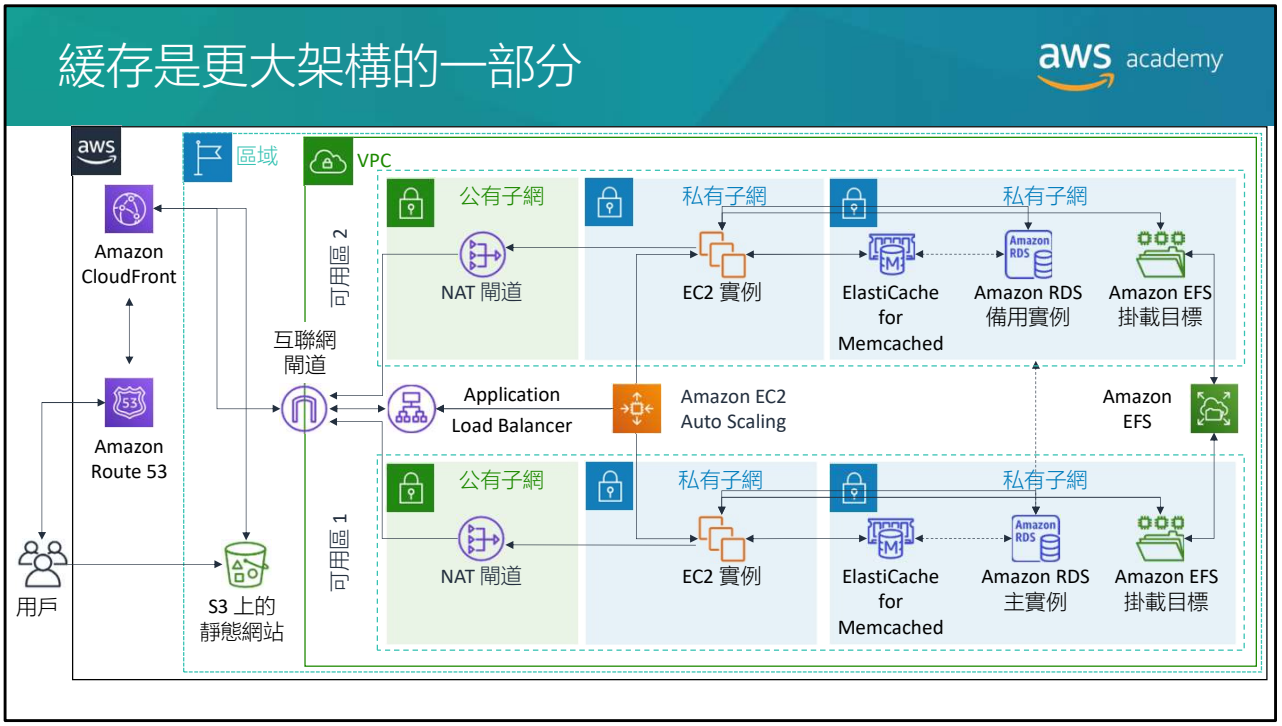
學完本模組後，您應該能夠：

- 確定如何通過緩存內容提升應用程式性能並減少延遲
- 確定如何設計通過邊緣網站實現分配和分散式拒絕服務 (DDoS) 保護的架構
- 創建使用 Amazon CloudFront 來緩存內容的架構
- 識別會話管理與緩存的關係
- 描述如何設計使用 Amazon ElastiCache 的架構

模組 11：緩存內容

第 1 節：架構需求

介紹第 1 節：架構需求。



在本模組中，您將學習如何在網路環境中實施緩存。本模組將介紹架構圖的最後兩個組件（Amazon ElastiCache 和 Amazon CloudFront），已在圖中展示。本模組還將介紹使用 Amazon DynamoDB 進行資料庫緩存。

咖啡館業務要求

咖啡館基礎設施的容量經常因同等量級的請求而超載。這不僅效率低下，還增加了成本和延遲。

咖啡館基礎設施的容量經常因對靜態內容（如功能表項目圖像）同等量級的請求而超載。這種情況增加了成本和延遲。Sofía 和 Nikhil 希望識別和緩存經常訪問的靜態內容，以減少延遲並改善客戶體驗。每次客戶載入功能表時，都會從緩存中獲取。但是，當功能表項目更改時，請求將路由到資料庫，並更新緩存。

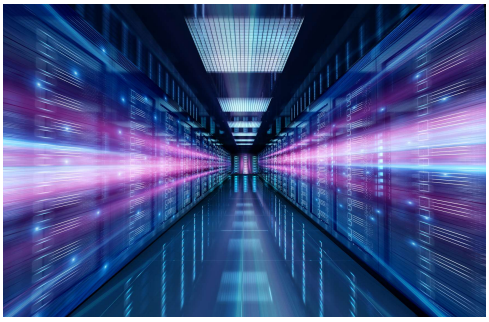
此外，當地名人開始通過視頻推薦支援咖啡館。Sofía 和 Nikhil 必須使用邊緣緩存來資料流資料，以根據載入網站的使用者所在的地理位置提供適當且具有區域吸引力的內容。

模組 11：緩存內容

第 2 節：緩存概覽

介紹第 2 節：緩存概覽。

緩存：犧牲容量換取速度



- 緩存是高速資料存儲層
- 緩存能存儲資料子集
- 緩存能提高資料檢索性能
- 緩存能減少對底層速度較慢的存儲層的訪問需求

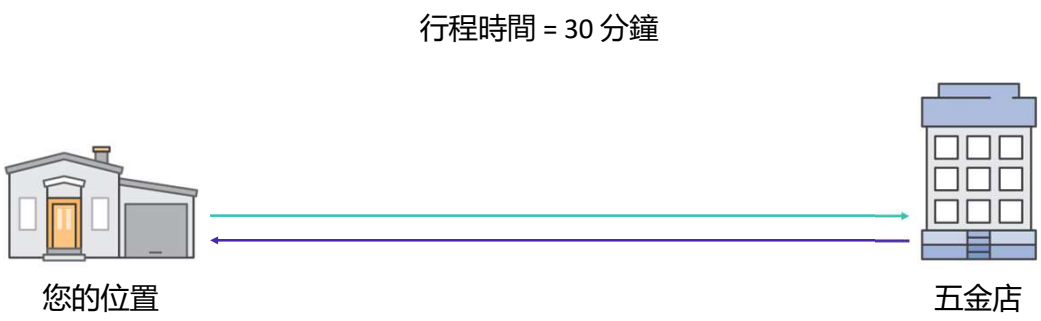
無論您的應用程式是用於提供最新新聞、前十位元的排行榜、產品目錄還是銷售活動門票，它的速度都很重要。應用程式成功與否受內容分發速度的影響。當某人需要資料時，無論是網頁數據還是推動業務決策的報告資料，如果此等資料已經緩存，您便能更快地交付數據。

在計算中，*緩存*是高速資料存儲層。與通常以完整且持久的形式存儲資料的資料庫不同，緩存會臨時存儲資料子集。緩存的主要目的是減少對底層速度較慢的存儲層的訪問需求，以此來提高資料檢索性能。與訪問資料主存儲位置的請求相比，未來對緩存資料的請求的處理速度更快。

緩存犧牲了容量來換取速度。通過緩存，您可以高效地重新使用之前檢索或計算的資料。

緩存中的資料通常存儲在快速訪問硬體中，例如隨機訪問記憶體 (RAM)。

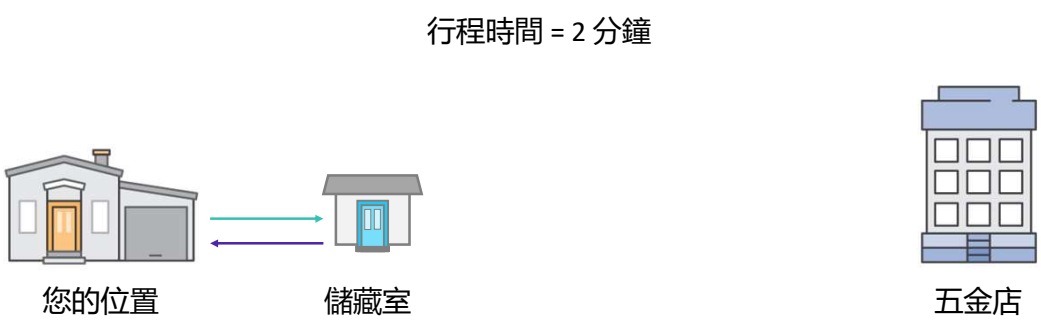
緩存示例 (1/2)



為了說明緩存如何提高性能，不妨考慮前往五金店購物的例子。

如果商店在數英里之外，那麼每次需要購物時前往商店都需要花費很大的精力。

緩存示例 (2/2)



您可以將日常使用的物品存放在靠近您家的儲藏室內。因此，獲取這些物品所需的時間比前往五金店所需的時間少。

但您仍可以前往商店更新物品。

在本例中，儲藏室類似於緩存。

為什麼要緩存?

需要借助速度慢且成本高的查詢才能獲取的資料

相對靜態且訪問頻繁的資料，例如社交媒體網站的使用者資料

可能在一段時間內過時的資訊，如公開交易的股票價格

在決定要緩存的資料時，請考慮以下因素：

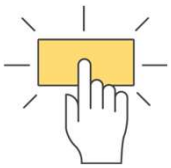
速度和費用 – 耗時的資料庫查詢和經常使用的複雜查詢通常會在應用程式中造成瓶頸。通常，如果要使用速度慢且成本高的查詢來獲取資料，則可以選擇緩存資料。例如，在多個表上執行聯接的查詢比簡單的單個表查詢要慢且成本更高。但即使資料所需的查詢相對快速且簡單，這種資料也可能是緩存的候選項，具體取決於其他因素。

資料和訪問模式 – 在確定要緩存的內容時，還需瞭解資料本身及其訪問模式。例如，對返回的搜索結果本質上動態變化程度異常高的網頁進行緩存並無意義。為了使緩存更有意義，資料應相對靜態且訪問頻繁，例如社交媒體網站上的個人資料。相反，如果緩存沒有任何速度或成本優勢，那麼您也無需緩存資料。例如，緩存返回搜索結果的網頁沒有意義，因為這些查詢和結果幾乎都是唯一的。

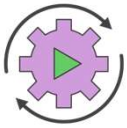
過時 – 根據定義，緩存的資料是過時的資料。即使緩存資料在特定環境中是未過時的，也應始終將其視為過時數據。在確定資料是否需要緩存時，還必須確定應用程式對過時資料的容忍度。您的應用程式可能會在某種情況下容忍過時資料，但其他情況下不行。例如，假設某應用程式在網站上提供公開交易的股票價格。對於該應用程式，如果有免責聲明，表示價格可能會延遲長達 *n* 分鐘，則短暫延遲是可以接受的。但是，當應用程式必須將股票價格提供給進行買賣的經紀人時，您需要使用即時資料。

緩存的優勢

提高應用程式速度



降低回應延遲



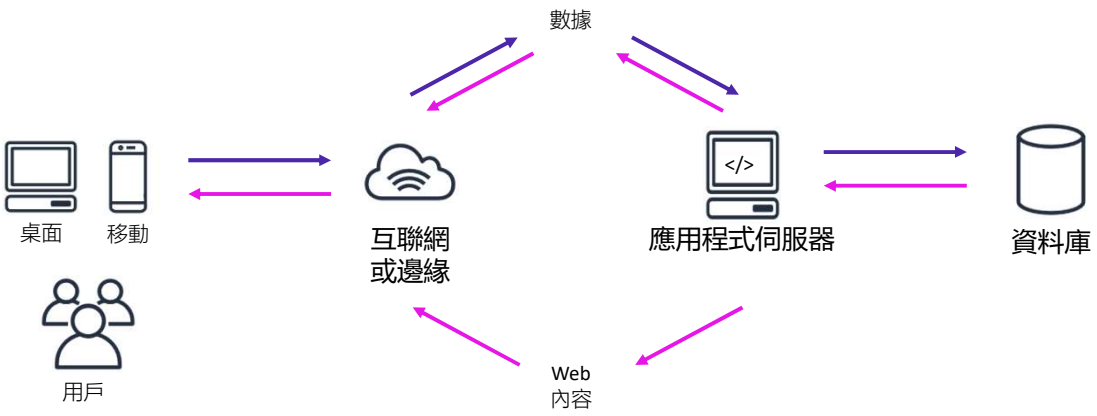
縮短資料庫存取時間

緩存通過將資料存儲在記憶體中，實現以高輸送量、低延遲的方式訪問常用的應用程式資料。

緩存可以：

- 提高應用程式的速度。
- 降低使用者在應用程式中遇到的回應延遲。
- 縮短應用程式處理時間和讀取密集型工作負載（如社交網路、遊戲、媒體共用和問答門戶）的資料庫存取時間。緩存為寫入密集型應用程式帶來的優勢通常並不明顯。但即使是寫入密集型應用程式，其讀/寫比率通常也大於 1，這意味著讀取緩存仍然是有益的。

在整個資料傳輸過程中緩存



這個簡單的 Web 應用程式架構展示了資料流程出和流向使用者的情況。您可以在每個層使用緩存來提高應用程式的整體性能和可用性。這些層包括作業系統、網路層（如內容分發網路 (CDN) 和網域名稱系統 (DNS)）、Web 應用程式和資料庫。

在這種架構中，您可以使用緩存來：

- 加快從網站檢索資訊
- 存儲功能變數名稱到 IP 位址的映射
- 加快從 Web 伺服器或應用程式伺服器檢索 Web 內容
- 提升應用程式性能並加快資料存取速度
- 降低與資料庫查詢請求相關的延遲

在本模組中，您將瞭解不同的 AWS 服務如何支援在不同的層進行緩存。

第 2 節要點

- 緩存通過將資料存儲在記憶體中，實現以高輸送量、低延遲的方式訪問常用的應用程式資料
- 在決定要緩存哪些資料時，需考慮速度和費用、資料和訪問模式以及應用程式對過時資料的容忍度
- 緩存可以在各個技術層中應用和使用，包括作業系統、網路層、Web 應用程式和資料庫

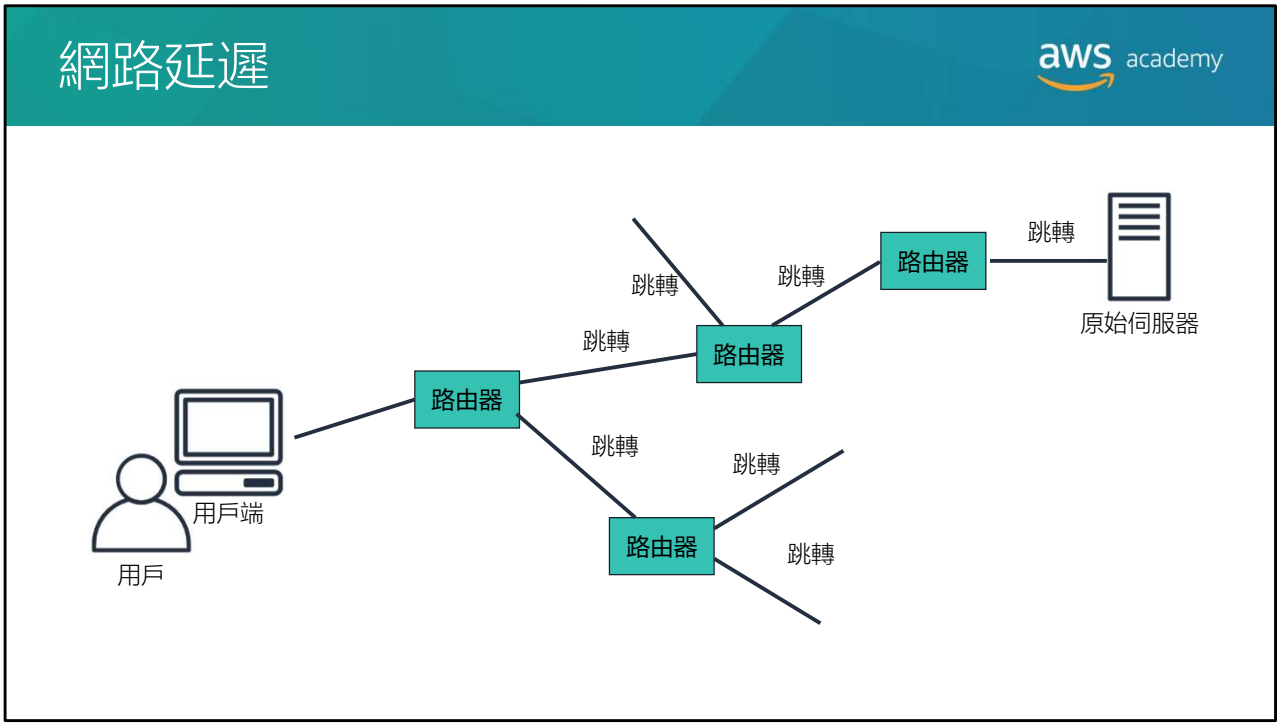
本模組中這節內容的要點包括：

- 緩存通過將資料存儲在記憶體中，實現以高輸送量、低延遲的方式訪問常用的應用程式資料
- 在決定要緩存哪些資料時，需考慮速度和費用、資料和訪問模式以及應用程式對過時資料的容忍度
- 緩存可以在各個技術層中應用和使用，包括作業系統、網路層、Web 應用程式和資料庫

模組 11：緩存內容

第 3 節：邊緣緩存

介紹第 3 節：邊緣緩存。



當某人流覽您的網站或使用您的應用程式時，其請求將通過許多不同的網路路由到您的原始伺服器。原始伺服器（也稱為*源*）存儲物件的原始最終版本（例如，Web 物件、圖像和媒體檔）。網路跳轉次數和請求必須經過的距離會顯著影響網站的性能和回應速度。

此外，網路延遲還取決於原始伺服器的地理位置。如果您的 Web 流量分散在不同的地理位置，那麼在全球範圍內複製整個基礎設施有時並不可行（或經濟高效）。在這種情況下，內容分發網路 (CDN) 可能非常有用。

內容分發網路 (CDN)

- 緩存伺服器的全球分散式系統
- 緩存經常請求的檔（靜態內容）的副本
- 從附近的緩存邊緣或接入點提供所請求內容的本機複本
- 提高應用程式性能和擴展性

內容分發網路 (CDN) 是緩存伺服器的全球分散式系統。CDN 會緩存應用程式原始伺服器上託管的常見請求檔的副本。這些檔可以包含靜態內容，例如 HTML、CSS、JavaScript、圖像和視頻檔。CDN 從緩存邊緣或接入點 (PoP) 分發所請求內容的本機複本，從而以最快的速度向請求者分發內容。

有關使用 CDN 進行緩存的更多資訊，請參閱內容分發網路 (CDN) 緩存。

Amazon CloudFront

- 為 Amazon 全球 CDN
- 針對具有多層緩存（默認）和廣泛靈活性的所有分發使用案例進行了優化
- 為您的架構提供一層額外的安全保護
- 支持 WebSocket 和 HTTP 或 HTTPS 方法

Amazon CloudFront 是一項全球 CDN 服務，可加速向使用者分發內容。此類內容可能是靜態和動態內容、使用 HTTP 或 HTTPS 的媒體檔以及流媒體視頻（包括視頻點播和直播）。 與其他 AWS 服務一樣，CloudFront 是一種按使用量付費的自助服務，不需要長期承諾，也無最低消費。

預設情況下，CDN 提供多層緩存。區域性邊緣緩存可在對象尚未緩存在邊緣時減少延遲並降低原始伺服器上的負載。此外，CDN 還提供多種媒體資料流選項，包括預先錄製的檔和直播活動。它可以為全球受眾提供 4K 傳輸所需的持續高輸送量。

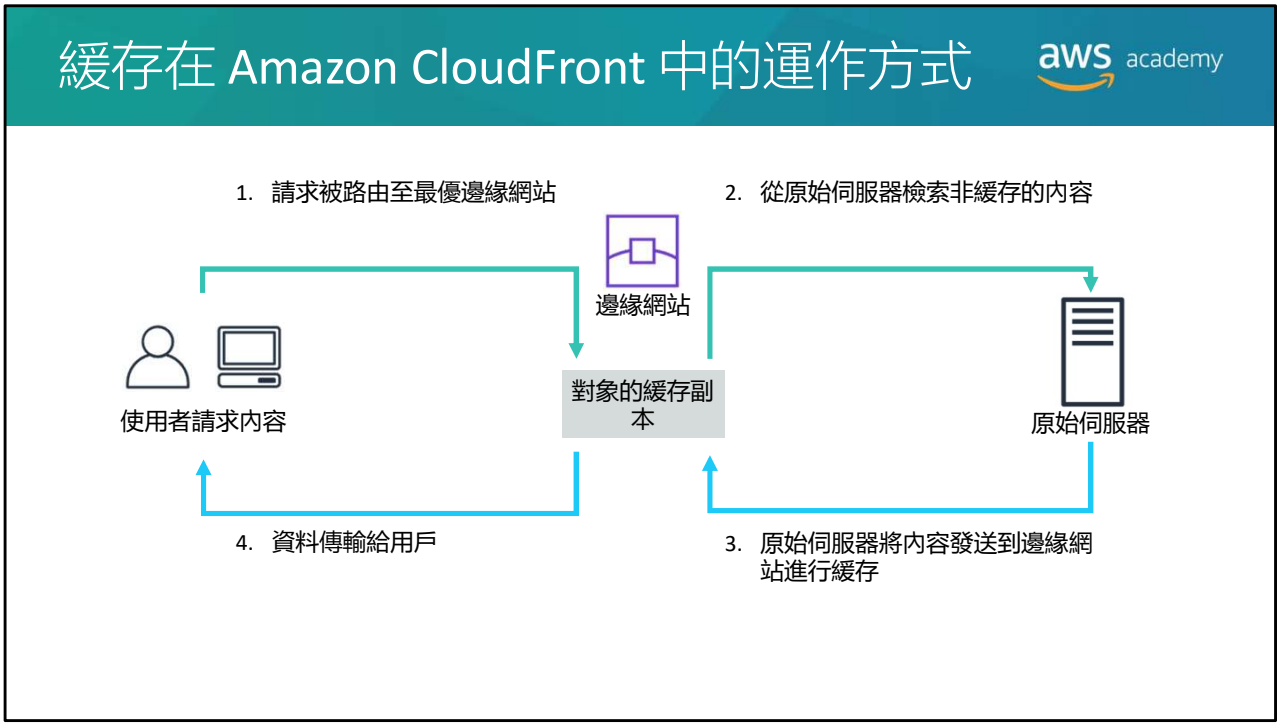
CloudFront 提供網路級別和應用程式級別保護。您的流量和應用程式可從 AWS Shield Standard 等各種內置保護功能受益，無需額外付費。您還可以使用可配置的功能（例如 AWS Certificate Manager (ACM)）來創建和管理自訂 SSL 證書而無需支付額外費用。CloudFront 支持安全通訊端層/傳輸層安全性 (SSL/TLS) 協議。

CloudFront 支援使用 WebSocket 協議進行即時雙向通信。這種持久連接使用戶端和伺服器可以互相發送即時資料，避免了重複打開連接的開銷。這對聊天、協作、遊戲和金融貿易等通信應用程式尤其有用。CloudFront 還支持 HTTP 方法（DELETE、GET、HEAD、OPTIONS、PATCH POST、PUT），這些方法可提高動態網站的性能。這些網站具有 Web 表單、評論和登錄框、加入購物車按鈕以及其他上傳使用者資料的功能。因此，您可以使用單個功能變數名稱通過 CloudFront 交付整個網站，從而加快網站各部分的下載和上傳速度。

該 Amazon.com 網頁示例展示了靜態和動態內容如何構成動態 Web 應用程式。該 Web 應用程式通過 HTTPS 協定提供，用於對使用者頁面請求和從 Web 伺服器返回的頁面進行加密。您可以使用 CDN 或邊緣緩存來緩存靜態內容。此類內容可能包括 Web 物件（例如，HTML 文檔、CSS 樣式表或 JavaScript 檔）、影像檔和視頻檔。

您無法緩存動態生成的內容或使用者生成的資料。但是，您可以將 CloudFront 配置為從自訂源上運行的應用程式提供這些資訊。例如，它可能是 EC2 實例或 Web 伺服器。

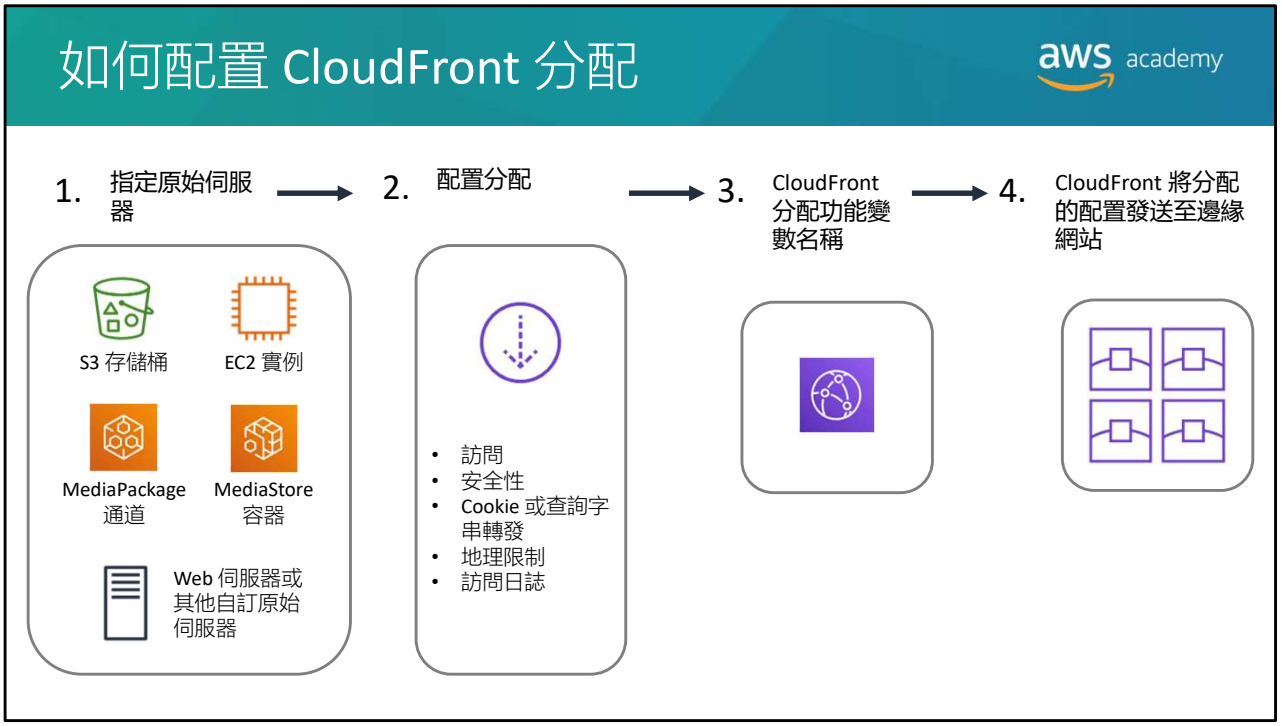
此外，您可以將 CloudFront 配置為要求檢視器使用 HTTPS 請求您的物件，以便在 CloudFront 與檢視器通信時加密連接。您也可以將 CloudFront 配置為使用 HTTPS 從源獲取物件，以便在 CloudFront 與源通信時加密連接。



Amazon CloudFront 通過全球資料中心網路（稱作*邊緣站點*）分發內容。

當使用者請求通過 CloudFront 提供的內容時，DNS 會將請求路由到最能滿足該請求需要的邊緣網站。通常，它是最近的邊緣網站，具有最低的延遲。CloudFront 在緩存中檢查請求內容（第 1 步）。如果內容在緩存中，CloudFront 會立即將其分發給用戶（第 4 步）。內容當前可能不在緩存中。如果不在，CloudFront 會將請求轉發到您標識為內容最終版本來源的原始伺服器（第 2 步）。原始伺服器將內容發回邊緣網站（第 3 步），然後 CloudFront 將內容轉發給使用者（第 4 步）。它還將內容添加到邊緣網站的緩存中，以備下次有人請求這些內容。

如果物件熱度下降，各個邊緣網站可以移除這些物件，從而為更熱門的內容騰出空間。對於熱度下降的內容，CloudFront 使用*區域性邊緣緩存*。區域性邊緣緩存是在全球範圍內部署且靠近您的受眾的 CloudFront 網站。它們位於原始伺服器和全球邊緣網站之間，直接向受眾提供內容。區域性邊緣緩存比單個邊緣網站的緩存空間更大，因此物件在區域性邊緣緩存中保留的時間更長。這種安排有助於讓更多的內容更靠近受眾。它減少了 CloudFront 需要返回原始伺服器的次數，為受眾提升了整體性能。



如果您想使用 CloudFront 來分配內容，需要創建*分配*。

1. 您可以指定託管檔的原始伺服器。您的原始伺服器可以是 S3 存儲桶、AWS Elemental MediaPackage 通道、AWS Elemental MediaStore 容器或自訂原始伺服器。例如，自訂原始伺服器可以是 EC2 實例或您自己的 Web 伺服器。

2. 然後，指定有關如何跟蹤和管理內容分發的詳細資訊。例如，您可以指定是希望檔對所有人都可用還是僅供特定用戶使用。您還可以指定是否希望 CloudFront 執行以下功能：創建顯示使用者活動的訪問日誌、將 Cookie 或查詢字串轉發到原始伺服器或要求用戶使用 HTTPS 訪問您的內容。

3. CloudFront 將功能變數名稱分配到您的新分配。

4. CloudFront 會將您的分配配置發送至所有的邊緣網站，但不會發送內容。

如何使內容過期

- 生存時間 (TTL) –
  - 固定時間（有效期）
  - 由您設置
  - 從 CloudFront 到原始伺服器的 GET 請求使用 If-Modified-Since 標頭
- 更改對象名稱 –
  - Header-v1.jpg 改為 Header-v2.jpg
  - 新名稱強制立即刷新
- 使對象失效 –
  - 不得已的手段：低效且成本高昂

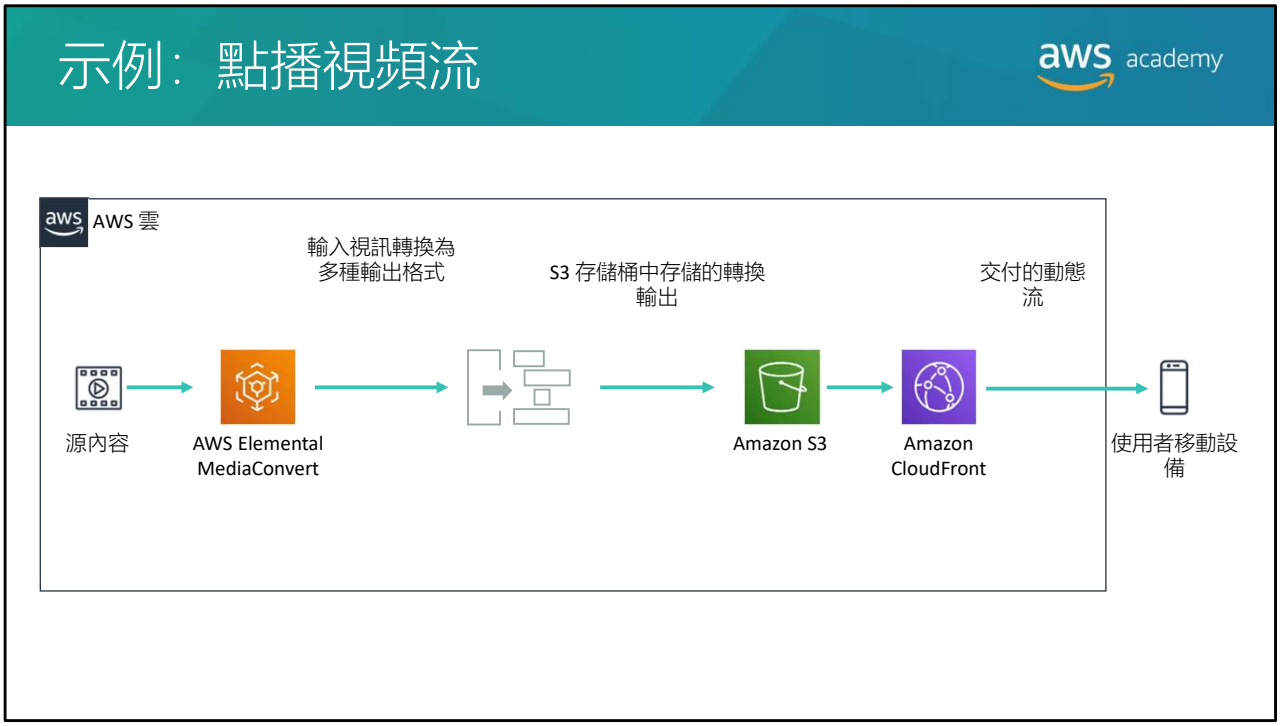
您可以通過三種方式使緩存內容過期：

- 生存時間 (TTL) – 使用這種方法，在 CloudFront 將另一個請求轉發到您的原始伺服器之前，您可以控制檔在 CloudFront 緩存中保留的時間。減少持續時間讓您可以提供動態內容。增加持續時間意味著您的用戶將獲得更高的性能，因為直接從邊緣緩存提供檔的可能性更大。較長的持續時間還會減輕原始伺服器的負載。如果您將特定原始伺服器的 TTL 設置為 *0*，CloudFront 仍會從該原始伺服器緩存內容。然後它將使用 If-Modified-Since 標頭髮出 GET 請求。因此，如果緩存的內容在原始伺服器中未發生更改，原始伺服器可以發出信號，指示 CloudFront 可以繼續使用緩存內容。如果不需要立即替換，TTL 是一種很好的方法。

- 更改物件名稱 – 此方法需要更多的工作，但可以立即替換。儘管您*可以*在 CloudFront 分配中更新現有物件，並使用相同的物件名稱，但不建議這樣做。僅當請求對象時（而不是當您在原始伺服器中放入新的物件或更新的物件時），CloudFront 才會將對象分配到邊緣網站。例如，您可以使用具有相同名稱的更新版本來更新原始伺服器中的現有物件。在這種情況下，在兩個列出的事件發生之後，邊緣網站才會從您的原始伺服器獲取新版本。

- 使物件失效 – 這種方法不是一個好的解決方案，因為系統必須強制與所有邊緣網站交互。您應該謹慎使用此方法，並且只能用於個別物件。

有關如何使緩存內容過期的詳細資訊，請參閱管理內容在邊緣緩存中保留的時長（有效期）。



正如您所學到的，您可以使用 CloudFront 來分發流媒體視頻，包括視頻點播和直播流。

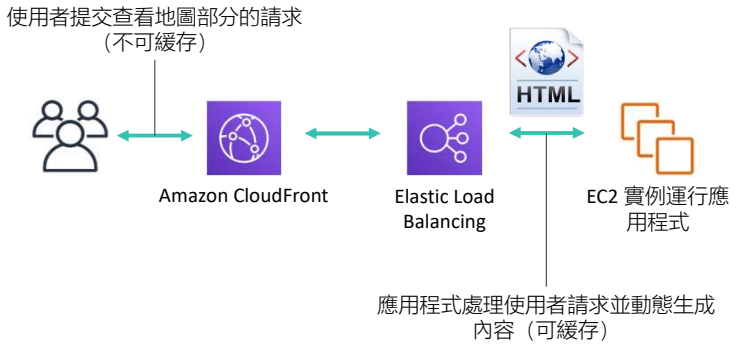
對於點播視頻流，必須先使用編碼器對視頻內容進行格式設置和打包，然後 CloudFront 才能分配視頻內容。例如，可以使用 AWS Elemental MediaConvert 和 Amazon Elastic Transcoder 編碼器。打包過程會創建*分段*，這些分段是靜態檔，其中包含音訊、視頻和字幕內容。它還會生成清單檔，這些檔描述了要播放哪些分段，以及播放它們的具體順序。打包格式包括基於 HTTP 的動態自我調整流（DASH 或 MPEG-DASH）、Apple HTTP Live Streaming (HLS)、Microsoft Smooth Streaming 以及通用媒體應用程式格式 (CMAF)。

將視訊轉換為輸出格式後，您將轉換後的內容託管在 S3 存儲桶，也即您的原始伺服器中。然後，您可以使用 CloudFront 將分段檔分發給世界各地的用戶。

有關使用 CloudFront 進行視頻資料流的更多資訊，請參閱使用 CloudFront 進行視頻點播和視頻直播。

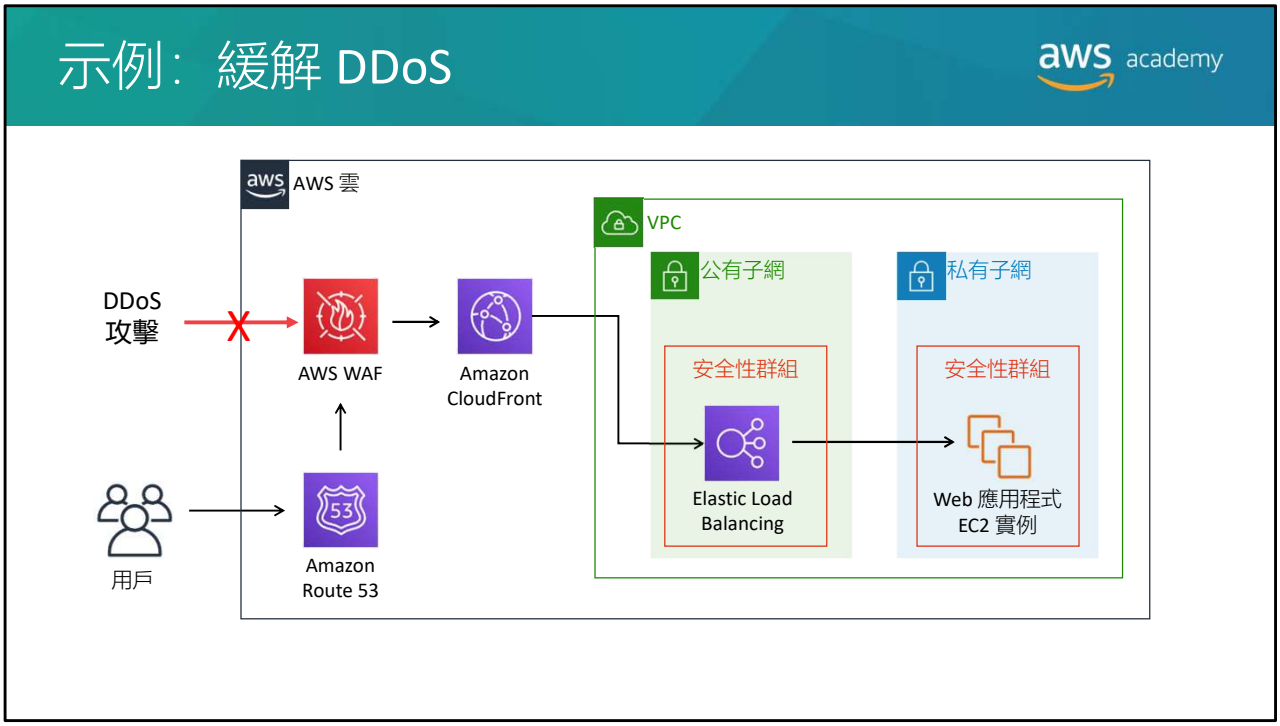
示例：動態生成的內容

使用案例：地圖圖塊
問題：需要更快的資料庫回應速度



通常，您只緩存靜態內容。但是，您可以讓內容*看起來*是靜態的（因為它的 URL），但在首次需要它時，它是動態構建的。當內容可重複使用時，這種動態構建可能很有用，但創建成本可能很高，而且可能不經常更改。

地圖圖塊是比較典型的例子。在此類例子中，您可以緩存人們經常查看的地點（如主要城市），而不是偏遠地區之類的地方。生成所有可能的圖塊組合非常昂貴和浪費，因為大多數圖塊幾乎永遠都不需要。每個圖塊 URL 的路徑部分都可以包含生成圖塊所需的參數。如果圖塊已存在於特定的 CloudFront 邊緣網站，則直接提供圖塊。否則，將生成圖塊並返回邊緣網站，以滿足將來的請求。



您可以使用 CloudFront 提高在 AWS 上運行的應用程式在遭受分散式拒絕服務 (DDoS) 攻擊後的恢復能力。DDoS 攻擊是蓄意為之，意在使您的網站或應用程式無法供使用者使用，例如，通過網路流量使其泛洪。為了實現這一目標，攻擊者使用多個來源編排針對目標的攻擊。這些來源可能包括受惡意軟體感染的電腦、路由器、物聯網 (IoT) 設備和其他終端節點的分散式組。

以下示例顯示了一個彈性架構，該架構有助於防止或緩解 DDoS 攻擊。

DNS 服務（如 Amazon Route 53）可以有效地將用戶的請求連接到 CloudFront 分配。然後，CloudFront 分配會將動態內容請求代理到託管應用程式終端節點的基礎設施。對通過 CloudFront 路由的 Route 53 DNS 請求和後續應用程式流量進行內聯檢查。Route 53 和 CloudFront 都內置了始終開啟的監控、異常檢測和針對常見基礎設施 DDoS 攻擊的緩解措施。

常見的基礎設施攻擊包括同步/確認 (SYN/ACK) 泛洪、使用者資料包通訊協定 (UDP) 泛洪和反射攻擊。當超過 SYN 泛洪攻擊閾值時，系統會啟動 SYN Cookie 以避免丟棄來自合法用戶端的連接。確定性資料包篩選將丟棄格式錯誤的 TCP 資料包和無效的 DNS 請求，並僅在流量適用于服務時才允許流量通過。啟發式異常檢測可評估流量的類型、來源和組成等屬性。在多個維度對流量進行評分，只有最可疑的流量才會被丟棄。

該方法使您能夠在保護應用程式可用性的同時避免誤報。Route 53 還可以抵禦 DNS 查詢泛洪。DNS 查詢泛洪實際是可持續數小時並嘗試耗盡 DNS 伺服器資源的 DNS 請求。Route 53 使用隨機分片和 Anycast 條帶化將 DNS 流量分佈到邊緣網站，並保護服務的可用性。

AWS WAF 是 Web 應用程式防火牆。它允許您監控轉發到 Amazon API Gateway API、Amazon CloudFront 或 Application Load Balancer 的 HTTP 和 HTTPS 請求。AWS WAF 還可讓您控制對內容的訪問。例如，您可以指定條件，例如請求來自的 IP 位址或查詢字串的值。基於這些條件，API Gateway、CloudFront 或 Application Load Balancer 以請求的內容或 HTTP 403 狀態碼（禁止）進行回應。您還可以將 CloudFront 配置為在請求被阻止時返回自訂錯誤頁面。

如需深入瞭解如何提高在 AWS 上運行的應用程式抵禦 DDoS 攻擊的彈性，請參閱以下資源：

- 實現 DDoS 彈性的 AWS 最佳實踐 AWS 白皮書
- How to Help Protect Dynamic Web Applications Against DDoS Attacks by Using Amazon CloudFront and Amazon Route 53 AWS 安全性博客文章

第 3 節要點

aws academy

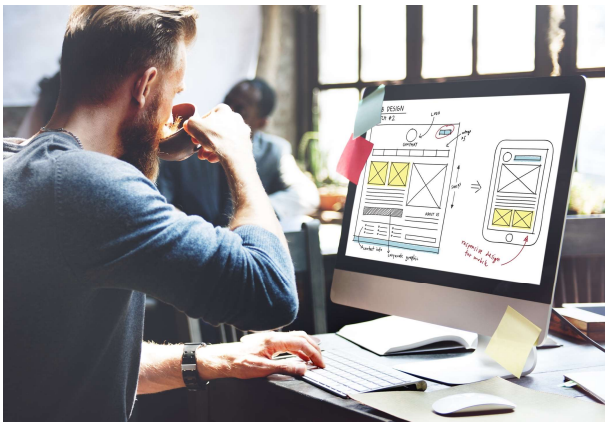
- Amazon CloudFront 是一項 全球 CDN服務，可加速向使用者分發內容（包括靜態內容和視頻），無最低使用承諾。
- CloudFront 使用由邊緣網站和區域邊緣緩存 組成的全球網路向使用者分發內容。
- 要使用 CloudFront 分發內容，請指定源伺服器 並配置 CloudFront 分配。CloudFront 將分配功能變數名稱，並將您的分配配置發送至其所有邊緣網站。
- 您可以使用 Amazon CloudFront 提高在 AWS 上運行的應用程式在遭受 DDoS 攻擊後的恢復能力 。

本模組中這節內容的要點包括：

- Amazon CloudFront 是一項全球 CDN 服務，可加速向使用者分發內容（包括靜態內容和視頻），無最低使用承諾。
- CloudFront 使用由邊緣網站和區域邊緣緩存組成的全球網路向使用者分發內容。
- 要使用 CloudFront 分發內容，請指定原始伺服器並配置 CloudFront 分配。CloudFront 將分配功能變數名稱，並將您的分配配置發送至其所有邊緣網站。
- 您可以使用 Amazon CloudFront 提高在 AWS 上運行的應用程式在遭受 DDoS 攻擊後的恢復能力。

模組 11 – 指導實驗：使用 Amazon CloudFront 流式處理動態內容

aws academy



您現在將完成“模組 11 – 指導實驗：使用 Amazon CloudFront 流式處理動態內容”。

指導實驗：場景

在此實驗中，您將使用 Amazon Elastic Transcoder 將源視訊轉換為多位元速率。然後，使用 Amazon CloudFront 通過 Apple HTTP Live Streaming (HLS) 協定將動態多位元速率流傳輸到連接的設備。

Amazon Elastic Transcoder

Amazon CloudFront

在此實驗中，您將使用 Amazon Elastic Transcoder 將源視訊轉換為多位元速率。然後，使用 Amazon CloudFront 通過 Apple HTTP Live Streaming (HLS) 協定將動態多位元速率流傳輸到連接的設備。此流可以在任何支援 HLS 協定的流覽器上播放。

Apple HLS 可通過普通 Web 伺服器動態調整影片播放品質，以匹配有線或無線網路的適用速度。這一過程通過創建不同品質的流實現。每個流再分解為不同的塊，按順序傳輸到用戶端設備。在用戶端，您可以選擇不同位元速率的流，以使資料流會話能夠適應不同的網路速度。

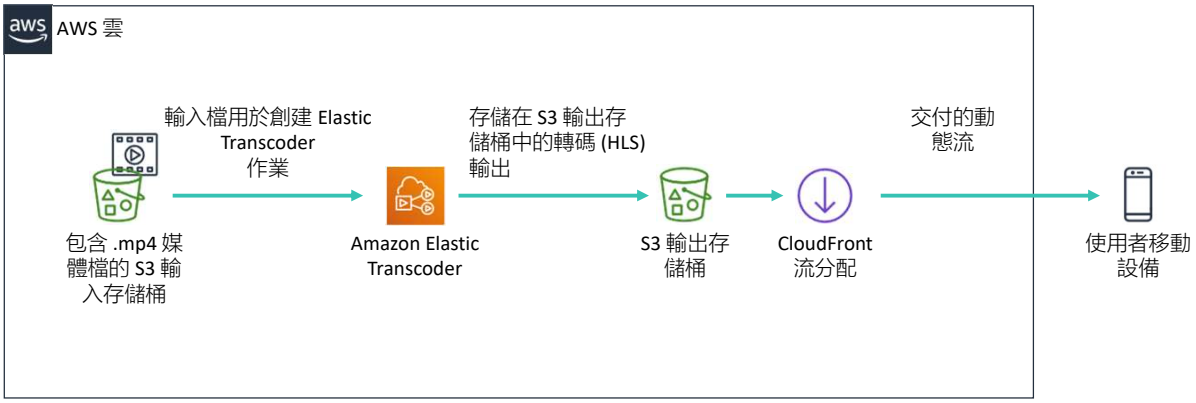
指導實驗：任務

1. 創建 Amazon CloudFront 分配
2. 創建 Amazon Elastic Transcoder 管道
3. 測試動態（多位元速率）流的播放

在本指導實驗中，您將完成以下任務：

1. 創建 Amazon CloudFront 分配
2. 創建 Amazon Elastic Transcoder 管道
3. 測試動態（多位元速率）流的播放

指導實驗：最終產品



該圖總結了您完成實驗後將會構建的內容。

大約 30 分鐘

現在可以開始指導實驗了。

指導實驗總結：要點

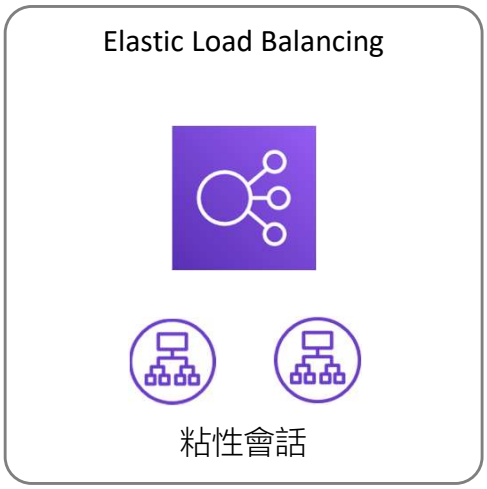
完成這個指導實驗之後，您的講師可能會帶您討論此指導實驗的要點。

模組 11：緩存內容

第 4 節：緩存 Web 會話

介紹第 4 節：緩存 Web 會話。

會話管理：粘性會話



使負載等化器能夠將請求路由到管理用戶會話的特定伺服器的功能。

- 使用用戶端 Cookie
- 經濟高效
- 加快會話檢索速度
- 存在缺點 –
  - 實例失敗時會話丟失
  - 限制可擴展性：負載分配不均衡和延遲增加

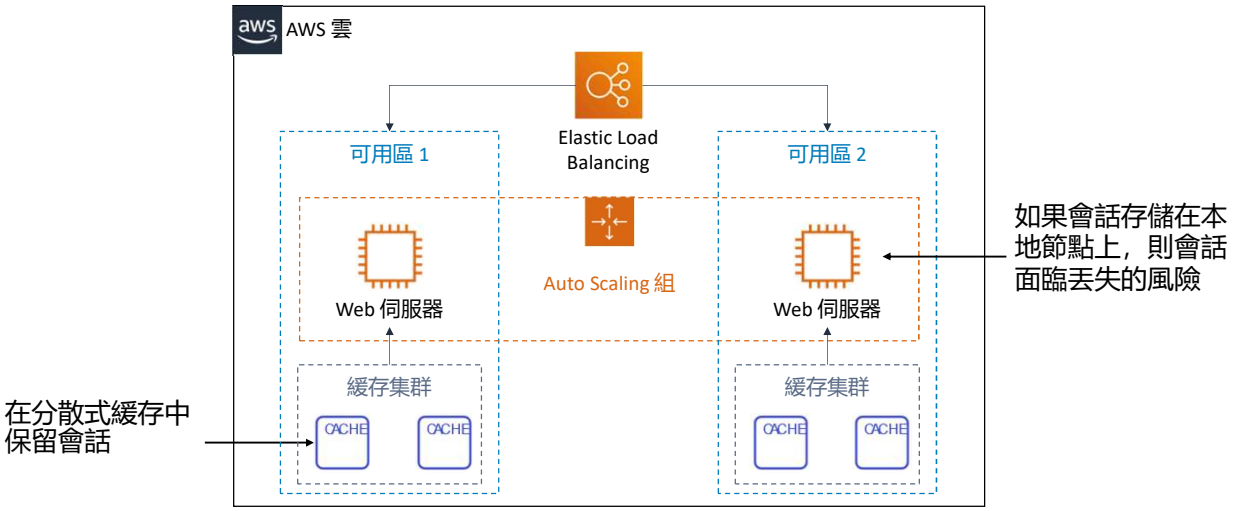
當使用者或服務與 Web 應用程式交互時，它會發送 HTTP 請求，應用程式返回回應。這樣的一系列事務稱為*會話*。每個請求都獨立於之前的事務。因此，會話用於管理用戶身份驗證，並在使用者與應用程式交互時存儲使用者資料。例如，通過使用會話，使用者不需要為他們向伺服器發出的每個請求發送憑證。

您可以通過多種方式管理使用者會話。（預設情況下，負載等化器會將每個請求單獨路由到負載最小的註冊實例。）要使用粘性會話，用戶端必須支援 Cookie。

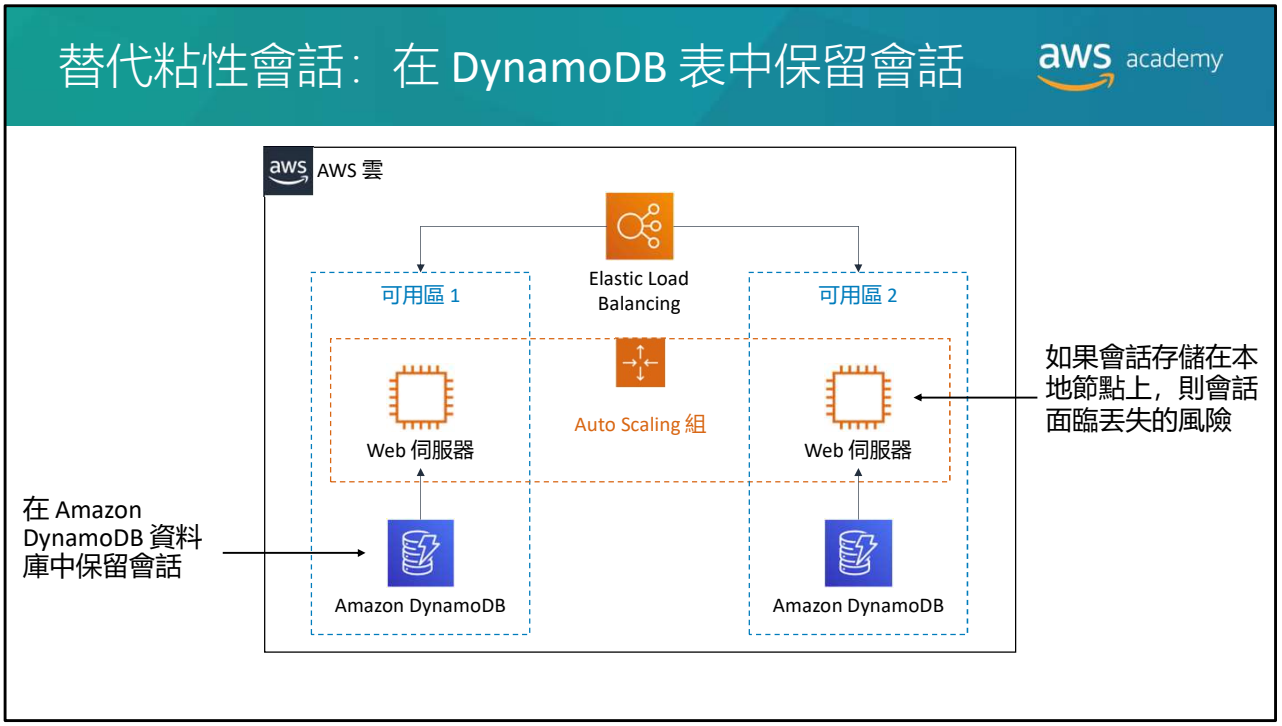
粘性會話具有成本效益，因為會話存儲在運行應用程式的 Web 伺服器上。因此，粘性會話可以消除網路延遲並加快這些會話的檢索速度。但是，如果實例發生故障，您可能會丟失存儲在該實例上的會話。

粘性會話的另一個缺點是它們可能會限制應用程式的可擴展性。使用粘性會話，負載等化器無法在每次收到來自用戶端的請求時真正均衡負載。粘性會話會強制負載等化器將所有請求發送到創建會話狀態的原始伺服器。如果該伺服器超載，那麼接收如此多的請求可能會導致伺服器之間的負載不均衡，並影響用戶回應時間。

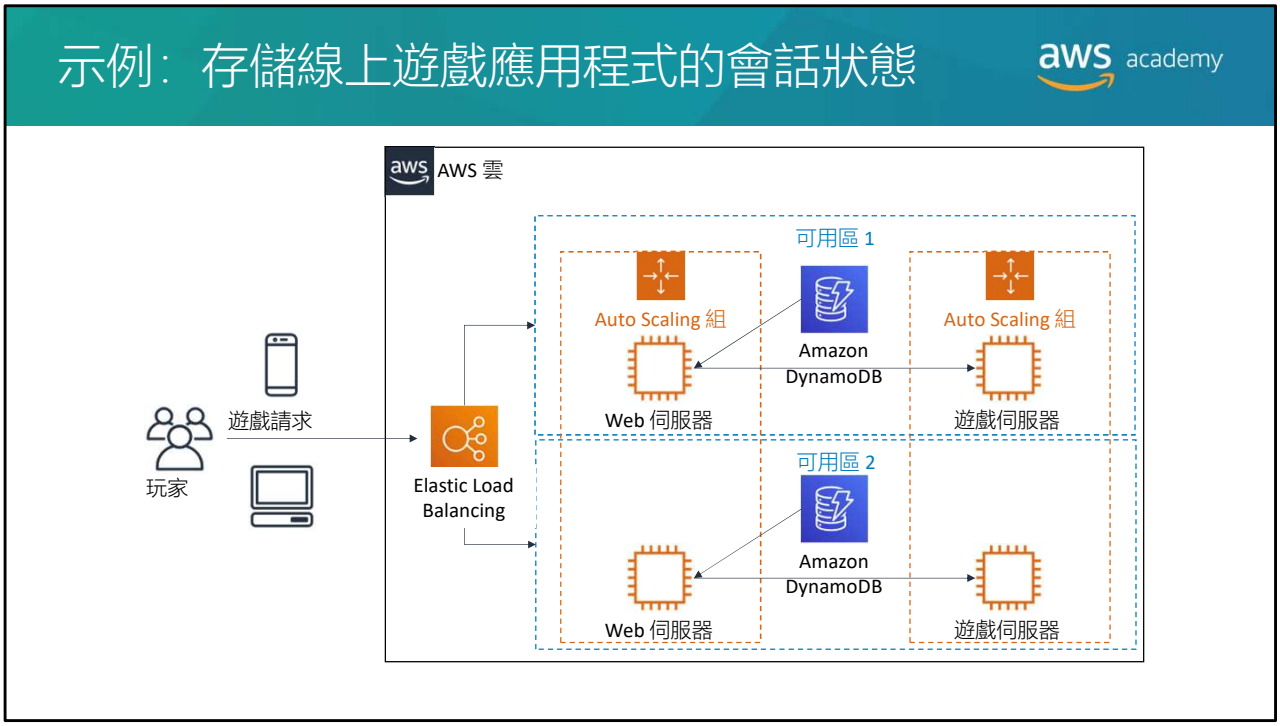
替代粘性會話：在分散式緩存中保留會話



您可以在架構中指定一個層，在實例外部以可擴展和可靠的方式存儲這些會話，而不是使用粘性會話。 一種選擇是將會話資料保存在分散式緩存中，如該架構圖所示。當 Web 伺服器數量發生變化以適應負載，並且您不希望冒丟失會話的風險時，在動態環境中實施此架構是有意義的。



另一種選擇是將會話資料保存在 Amazon DynamoDB 資料庫中，如該圖所示。通過 Amazon DynamoDB，您可以設置擴縮策略，並根據需要擴展和縮減 DynamoDB 容量。



該架構可能支援線上遊戲應用程式，在這種應用程式中，必須要快速檢索會話。隨著玩家收集物品、擊敗敵人、獲得金幣、解鎖關卡以及完成成就，遊戲資料將持續更新。必須將每個會話事件寫入資料庫層，以免丟失。遊戲製作者將會話歷史記錄和其他面向時間的資料存儲在 DynamoDB 中，以便按玩家、日期和時間快速查找。

每個 DynamoDB 資料庫表都與吞吐容量關聯。您可以指定每秒 1000 次寫入，DynamoDB 會在後臺擴縮資料庫。隨著需求的變化，您可以更新容量，然後，Amazon DynamoDB 會根據需要重新分配資源。這種彈性對遊戲開發商頗有助益：如果您的遊戲越來越受歡迎，您的玩家數量可能會突然從幾千名擴展到數百萬名。如果需要，您可以快速輕鬆地縮小規模。

DynamoDB 可以在任何規模上保持可預測的低延遲性能，如果您的遊戲發展到數百萬對延遲敏感的客戶，這一點至關重要。您不會花時間優化 DynamoDB 的性能。

要查看包含 DynamoDB 的類似遊戲架構，請參閱此 AWS 大資料博客文章。

第 4 節要點

- 會話用於管理使用者身份驗證，並在使用者與應用程式交互時存儲使用者資料。
- 您可以使用粘性會話管理會話，這是 Elastic Load Balancing 負載等化器的一項功能。粘性會話將請求路由到管理用戶會話的特定伺服器。
- 您還可以通過在 Web 伺服器實例外部保留會話資料來管理會話，例如，保留在分散式緩存或 DynamoDB 表中。

本模組中這節內容的要點包括：

- 會話用於管理用戶身份驗證，並在使用者與應用程式交互時存儲使用者資料。
- 您可以使用粘性會話管理會話，這是 Elastic Load Balancing 負載等化器的一項功能。粘性會話將請求路由到管理用戶會話的特定伺服器。
- 您還可以通過在 Web 伺服器實例外部保留會話資料來管理會話，例如，保留在分散式緩存或 DynamoDB 表中。

模組 11：緩存內容

第 5 節：緩存資料庫

介紹第 5 節：緩存資料庫。

應該在什麼時候緩存資料庫？

您很擔憂對客戶的回應時間。

大量的請求讓您的資料庫難以招架。

您希望降低資料庫成本。

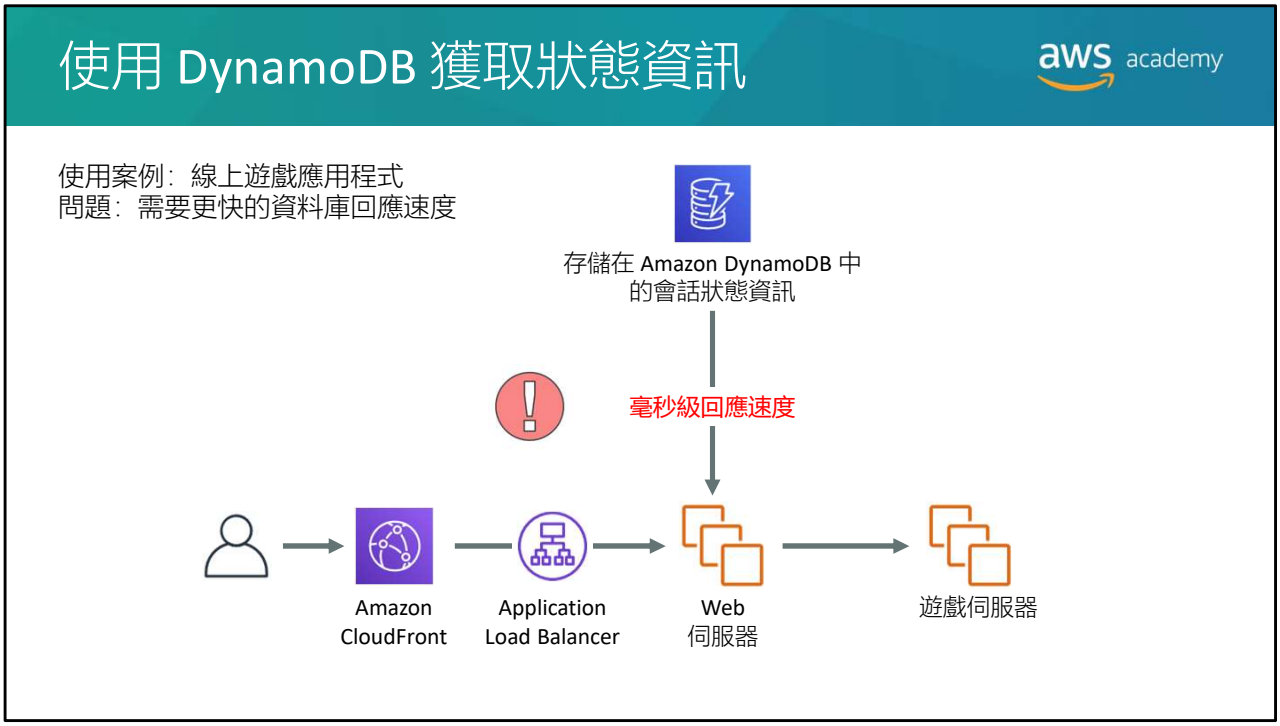
正如您在本模組前面所學到的，耗時的資料庫查詢和複雜的查詢會在應用程式中造成瓶頸。

在以下情況下，應考慮緩存資料庫：

- 您擔心對客戶的回應時間。您可能需要加速延遲敏感型工作負載。緩存有助於提高輸送量並減少資料檢索延遲，從而提高應用程式的性能。
- 大量的請求讓您的資料庫難以招架。您可能擁有大量流量，因而無法獲得該等工作負載所需的輸送量。在資料庫旁邊放置緩存層可以提高輸送量並實現更高的性能。
- 希望降低資料庫成本。無論資料是分佈在基於磁片的 NoSQL 資料庫中還是在關聯式資料庫中縱向擴展，通過擴展來實現高讀取可能成本高昂。可能需要多個資料庫唯讀副本才能匹配單個記憶體中緩存節點每秒的請求數。

資料庫緩存可消除主要資料庫面臨的不必要壓力（通常是頻繁訪問的讀取資料），以此為其提供助益。緩存本身可存在于資料庫、應用程式等許多區域中，也可作為獨立層存在。

使用 DynamoDB 獲取狀態資訊



假設有一個線上遊戲應用程式架構的簡化版本，您在 DynamoDB 中存儲會話狀態資訊。在某些情況下，您可能會發現毫秒級的回應速度對於您的應用程式來說不夠快。資料庫緩存可以說明解決這個問題。

Amazon DynamoDB Accelerator

適用于 DynamoDB 的完全託管、高度可用的記憶體中緩存

- 極致的性能（微秒級回應時間）
- 高度可擴展
- 完全託管
- 與 DynamoDB 集成
- 靈活
- 安全

Amazon DynamoDB Accelerator (DAX) 是適用于 DynamoDB 的完全託管且高度可用的記憶體中緩存。即使每秒數百萬個請求，它仍可提供高達 10 倍的性能提升（從毫秒到微秒）。DAX 將承擔向 DynamoDB 表添加記憶體中加速所需的所有繁重工作。您無需管理緩存失效、資料填充或集群管理。

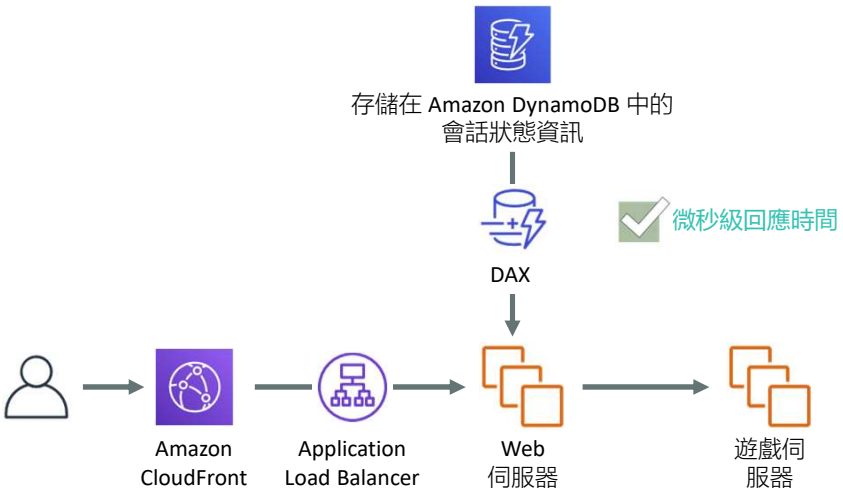
DAX 具有以下優勢：

- **極致的性能** – DynamoDB 提供一致的個位數毫秒延遲。當 DynamoDB 和 DAX 一起使用時，對於每秒數百萬個請求的讀取密集型工作負載，可實現微秒級的回應速度。
- **高度可擴展** – DAX 具有按需擴展功能。您可以先使用三節點 DAX 集群，並根據需要添加容量，擴展到十節點集群。
- **完全託管** – 和 DynamoDB 一樣，DAX 為完全託管。DAX 負責處理管理任務，包括預置、設置和配置、軟體修補以及在擴展操作期間通過節點複製資料。DAX 可自動處理常見的管理任務，如故障檢測、故障恢復和軟體修補。
- **與 DynamoDB 集成** – DAX 的 API 與 DynamoDB 相容，無需更改任何功能型應用程式碼。預置 DAX 集群，使用 DAX 用戶端開發套件 (SDK) 指向 DAX 集群中的現有 DynamoDB API 調用。由 DAX 負責處理其餘操作。
- **靈活** – 您可以為多個 DynamoDB 表預置一個 DAX 集群，或者為單個 DynamoDB 表預置多個 DAX 集群，也可以兩種預置方法結合使用。
- **安全** – DAX 與多項 AWS 服務完全集成，增強了安全性。您可以使用 AWS Identity and Access Management (IAM) 為每個用戶分配唯一的安全憑證，並控制每個使用者對服務和資源的訪問。使用 Amazon CloudWatch，您可以全面瞭解資源使用率、應用程式性能和運行狀況。通過與 AWS CloudTrail 集成，您可以輕鬆記錄和審計集群配置的更改。DAX 支援 Amazon Virtual Private Cloud (Amazon VPC)，可從現有應用程式安全輕鬆地訪問。標記可讓您進一步瞭解 DAX 集群，幫助您進行相應管理。

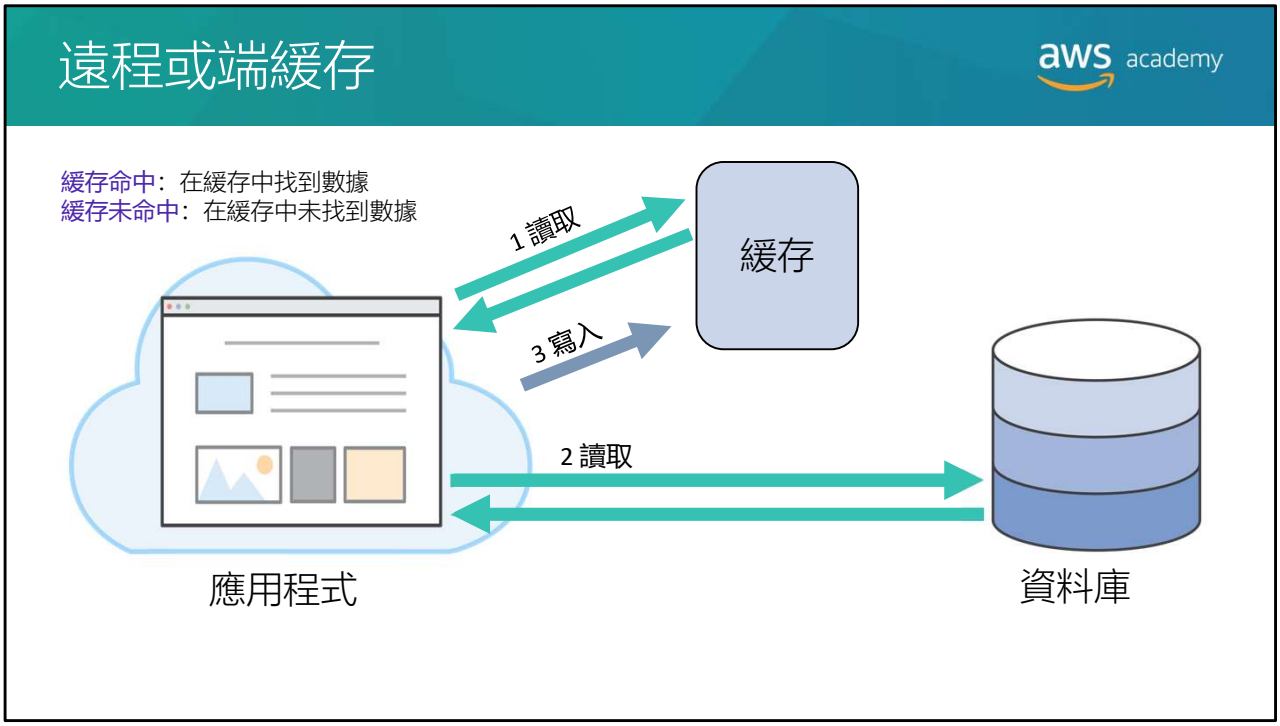
對緩存資料的檢索減少了現有 DynamoDB 表的讀取負載。因此，它可能會減少預置讀取容量，降低總體運營成本。

有關 DAX 的更多資訊，請參閱 AWS 資料庫博客文章。

將 DynamoDB 與 DAX 結合使用以加快回應速度



在遊戲示例中，通過將 DAX 添加到架構中，無需在遊戲代碼中進行任何重大更改，即可而獲得加速。通過這種方式，可以簡化對架構的部署。您必須要做的一點是使用指向 DAX 的新終端節點重新初始化 DynamoDB 用戶端。無需對代碼的其餘部分進行任何更改。DAX 會處理緩存失效和資料填充，無需您進行任何干預。當您舉辦可能導致玩家數量激增的活動時，緩存有助於加快響應速度。此類活動的一個例子是季節性可下載內容 (DLC) 產品或發佈新的補丁。



DAX 是透明的緩存。實施資料庫緩存部署的另一種方法是使用遠端緩存或端緩存。端緩存不直接連接到資料庫，而是與資料庫相鄰使用。端緩存通常基於鍵值 NoSQL 存儲（如 Redis 或 Memcached）構建。其中的每個緩存節點每秒可提供數十萬到百萬個請求。

端緩存通常用於讀取密集型工作負載。其工作方式如下：

1. 對於給定的鍵值對，應用程式首先嘗試從緩存中讀取資料。如果緩存包含相應資料（稱為*緩存命中*），則返回相關值。
2. 如果在緩存中找不到所需的鍵值對（稱為*緩存未命中*），應用程式會從底層資料庫中獲取資料。
3. 當應用程式再次需要資料時，資料必須存在，這一點非常重要。為了確保這一點，從資料庫獲取的鍵值對隨後將寫入緩存。

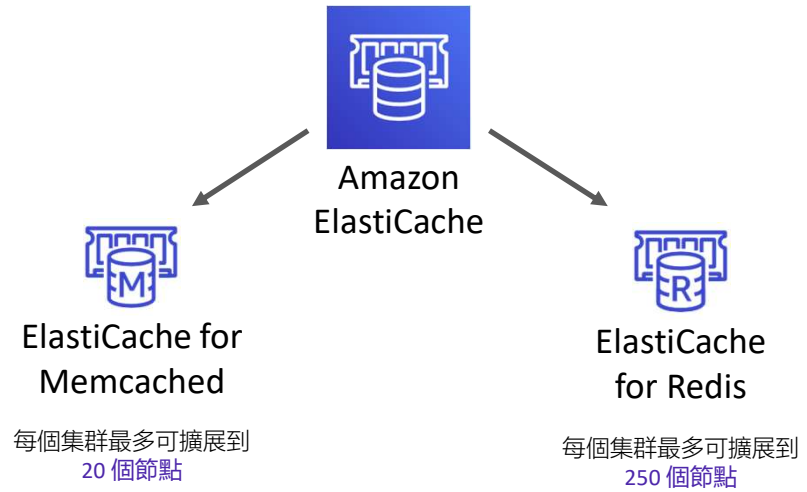
Amazon ElastiCache

ElastiCache 為 Web 應用程式提供雲中記憶體資料存儲。

- 使用記憶體中資料存儲和緩存
- 提供高性能
- 完全託管
- 可擴展
- 支持 Redis 和 Memcached

Amazon ElastiCache 是可以用作記憶體中資料存儲的端緩存，支持要求最嚴苛且需要亞毫秒級回應速度的應用程式。借助 ElastiCache，您無需執行硬體預置、軟體修補、設置、配置、監控、故障恢復和備份等管理任務。ElastiCache 會持續監控您的集群以保證工作負載正常運行，使您可以專注于價值更高的應用程式開發工作。Amazon ElastiCache 支援橫向擴展、縮減及縱向擴展，可滿足不斷變化的應用程式需求。通過分片實現寫入擴展和記憶體擴展。副本提供讀取擴展。ElastiCache 支援兩種開源的記憶體中資料庫：Redis 和 Memcached。

Redis 和 Memcached



ElastiCache for Memcached 最多可以擴展到每個集群 20 個節點。相比之下，ElastiCache for Redis 可以擴展到 250 個節點，從而提高資料訪問性能。ElastiCache 支持 Amazon VPC，使您能夠將集群隔離在您為節點選擇的 IP 範圍內。

ElastiCache 在高度可靠的基礎設施上運行，其可靠性與其他 AWS 服務使用的基礎設施相同。ElastiCache for Redis 可通過具有自動容錯移轉功能的多可用區部署實現高可用性。對於 Memcached 工作負載，資料將在集群中的所有節點之間進行分區。因此，當需求增長時，您可以橫向擴展以更好地處理更多資料。

Memcached 與 Redis 比較

特徵	Memcached	Redis
亞毫秒級延遲	是	是
能夠橫向擴展以進行寫入和存儲	是	否
多執行緒性能	是	否
高級資料結構	否	是
對資料集進行排序和排名	否	是
發佈/訂閱消息收發服務	否	是
具有自動容錯移轉功能的多可用區部署	否	是
持久性	否	是

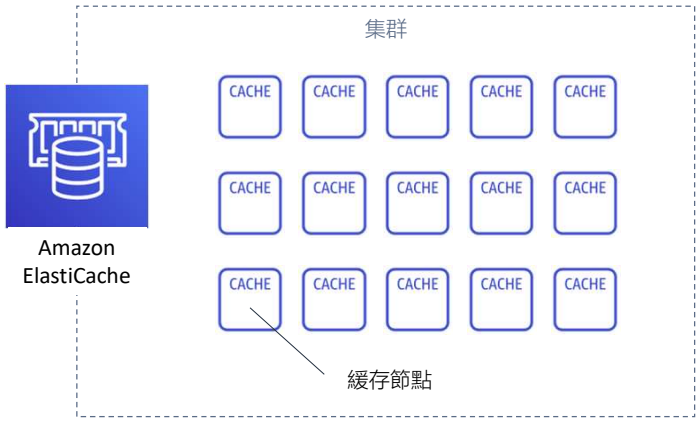
Memcached 和 Redis 引擎是簡單的緩存，可用於減輕資料庫負擔。每種引擎都具有某些優勢。此表比較了 Memcached 和 Redis 之間的一些關鍵特徵。

- **亞毫秒級延遲** – 兩種引擎都提供亞毫秒級回應速度。相較基於磁片的資料庫，將資料存儲在記憶體中可以更快地讀取資料。
- **橫向擴縮能力** – Memcached 使您能夠隨著系統需求的增加和減少而擴展和縮減，從而添加和刪除節點。
- **多執行緒性能** – 由於 Memcached 是多主題的，它可以使用多個處理內核，這意味著您可以通過縱向擴展計算容量來處理更多操作。
- **高級資料結構** – Redis 支援複雜的資料類型，如字串、雜湊、清單、集、排序集和點陣圖。
- **對資料集進行排序或排名** – 您可以使用 Redis 對記憶體中資料集進行排序或排名。例如，您可以使用 Redis 排序集來實施遊戲排行榜，用於保留按排名排序的玩家列表。
- **發佈/訂閱消息** – Redis 支援具有模式匹配功能的發佈/訂閱消息服務，可將其用於高性能聊天室、即時評論流、社交媒體源和伺服器間通信。

- **具有自動容錯移轉功能的多可用區部署** – ElastiCache for Redis 通過具有自動容錯移轉功能的多可用區部署提供高可用性，以防主節點發生故障。

- **持久性** – Redis 使您能夠持續保留金鑰存儲。相比之下，Memcached 引擎不支援持久性。例如，節點出現故障並被新的空節點替換；或者終止節點或縮小節點。在這種情況下，您將丟失存儲在緩存記憶體中的資料。

ElastiCache 組件

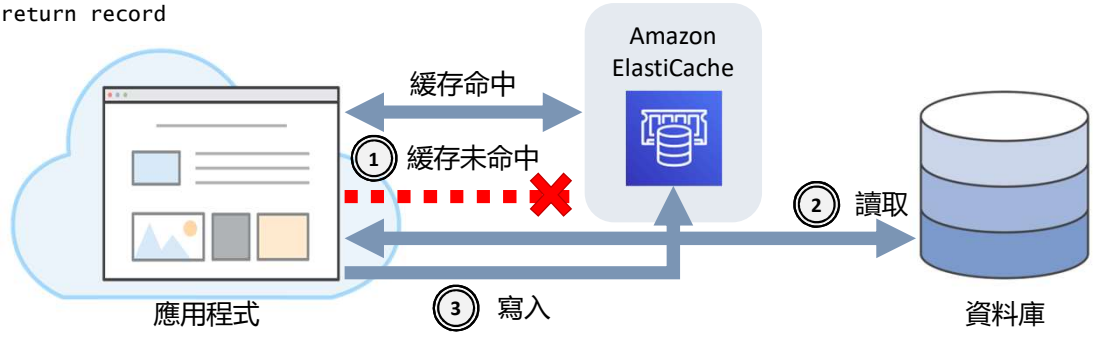


- 節點是 ElastiCache 部署的最小資料塊
- 每個節點都有自己的 DNS 名稱和埠
- 集群是一個或多個節點的邏輯分組

緩存節點*是 ElastiCache 部署的最小構建塊。*它是固定大小、與網路連接的安全 RAM 區塊。每個節點都運行創建或最後一次修改集群或複製組時選擇的引擎。每個節點都有自己的 DNS 名稱和埠。它可獨立於其他節點存在，也可以與其他節點組成分組（也稱為*集群*）。

緩存策略：延遲載入

```
def get_user(user_id):
    # Check the cache
    record = cache.get(user_id)
    if record is None:
        # Run a DB query
        record = db.query("select * from users where id = ?", user_id)
        # Populate the cache
        cache.set(user_id, record)
    return record
```



您可以使用 ElastiCache 部署兩種緩存策略。

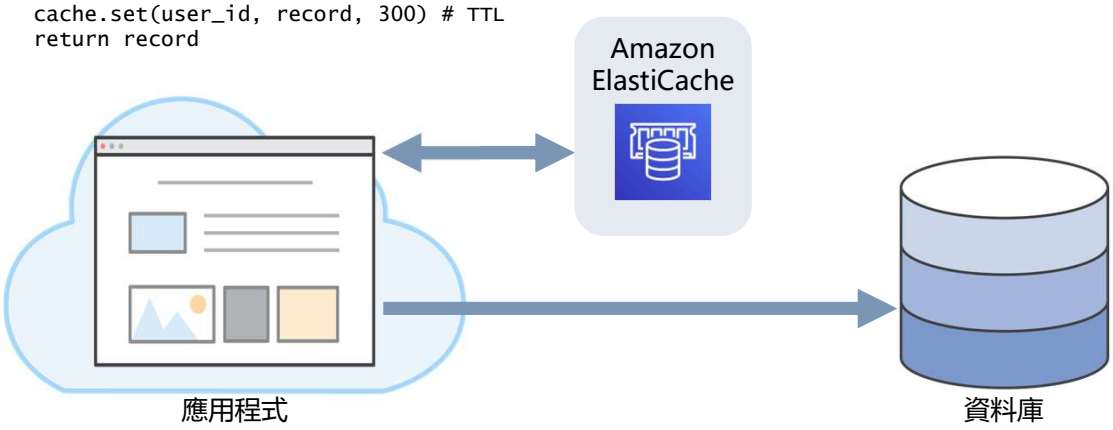
第一種策略是*延遲加載*，這是一種緩存策略，僅在必要時才將資料載入到緩存中。在此情況下，當應用程式請求資料時，它會先向 ElastiCache 緩存發出請求。如果資料存在於緩存中且為最新狀態，則發生緩存命中，ElastiCache 會將資料返回給您的應用程式。否則（如果緩存未命中），您的應用程式會從資料存儲中請求資料，然後將資料返回給應用程式。然後，您的應用程式會將資料寫入緩存，以便在下次請求時更快地檢索到。

當您擁有經常讀取但很少寫入的資料時，請使用延遲載入。例如，在典型的 Web 或移動應用程式中，使用者個人資料很少發生變化，但需要在整個應用程式中訪問。一個人每年可能只對自己的個人資料進行幾次更新。但是，根據使用者的不同，個人資料每天可能會被訪問數十次或數百次。

延遲載入的優點是只緩存請求的資料。由於大部分數據從未被請求，所以延遲載入避免了向緩存中填入不必要的資料。但是，緩存未命中會帶來不利影響。每次出現緩存未命中都會造成三次往返，這可能會導致資料在到達應用程式時出現明顯延遲。另外，如果僅在存在緩存未命中時將資料寫入緩存，則緩存中的資料可能會過時。當資料庫中的資料發生更改時，延遲載入不會向緩存提供更新。

緩存策略：直寫

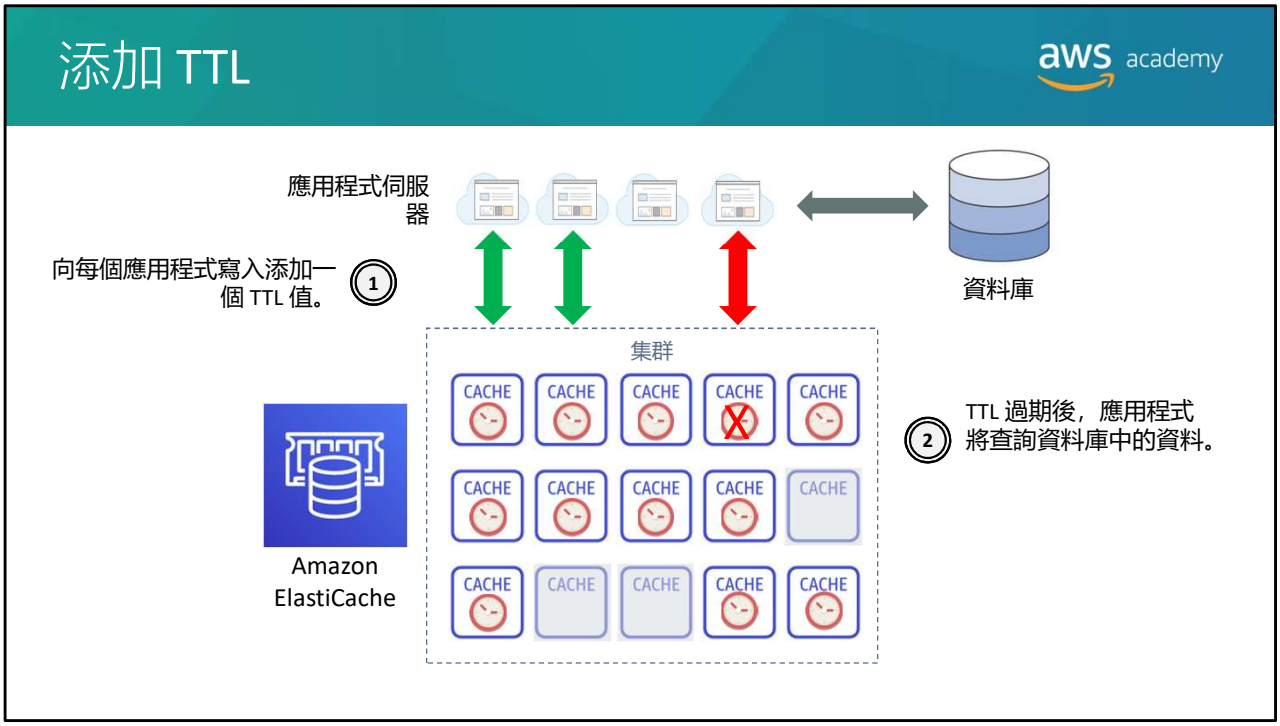
```
def save_user(user_id, values):
    # Save to DB
    record = db.query("update users…where id = ?", user_id, values)
    # Push into cache
    cache.set(user_id, record, 300) # TTL
    return record
```



第二種緩存策略是*直寫策略*。如果將資料寫入資料庫中，則會在緩存中添加資料或更新資料。當您具有必須即時更新的資料時，請使用直寫緩存策略。這種方法具有主動性：如果您知道哪些資料會被訪問到，就可以避免不必要的緩存未命中情況。一個很好的例子是：任何類型的聚合都適合採用這種方法，例如前 100 名遊戲排行榜、最熱門的 10 大新聞報導或推薦內容。由於此類資料通常由特定的應用程式或後臺作業代碼更新，可直接更新緩存。

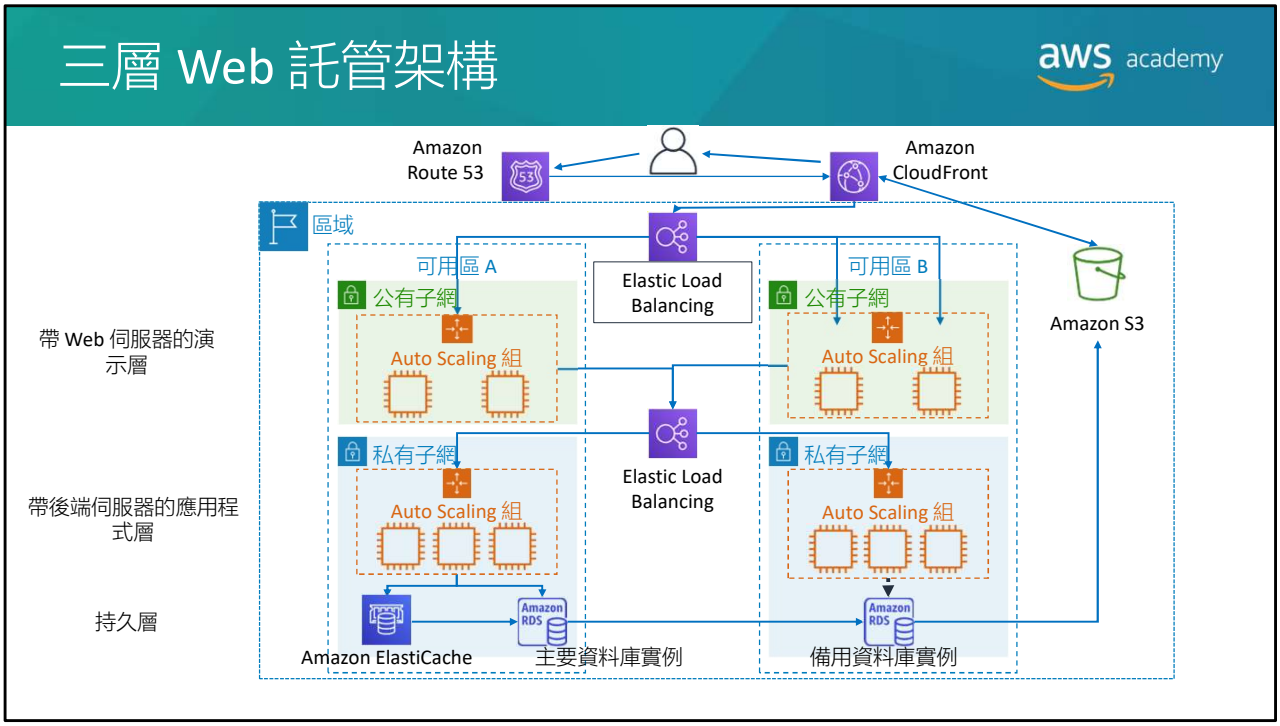
這種方法的優點是，它增加了應用程式在查找該值時在緩存中找到該值的可能性。缺點是，您可能緩存的是不需要的資料，因此這種方法可能會增加成本。

實際上，這兩種方法可結合使用，因此瞭解資料更改的頻率並使用適當的 TTL 對您來說非常重要。



延遲載入允許資料過時。而直寫可確保資料始終是最新狀態，但這種方法可能會使用不必要的資料填充緩存。

通過向每次寫入添加 TTL 值，您可以獲得每種策略的優點，並避免因數據造成的緩存混亂。TTL 是一個整數值或鍵，它指定了鍵過期之前的秒數或毫秒數，具體取決於記憶體引擎。當應用程式嘗試讀取過期鍵時，則視為在緩存中找不到數據。因此，系統會查詢資料庫並更新緩存。這樣可以防止資料過於陳舊，並且需要從資料庫中偶爾刷新緩存中的值。



您可能希望使用 Amazon ElastiCache 的一種場景是在傳統的三層 Web 託管架構中。在這種場景下，您希望在私有子網中維護私有後端伺服器的同時運行公共 Web 應用程式。您可以為 Web 伺服器創建一個可訪問互聯網的公有子網。同時，您可以將後端基礎設施放置在無法訪問互聯網的私有子網中。後端基礎設施的資料庫層可能包括 Amazon Relational Database Service (Amazon RDS) 資料庫實例和提供記憶體中層的 ElastiCache 集群。

在此 Web 託管架構圖中：

- *Amazon Route 53* 使您能夠將頂層網域名（例如 *example.com*）DNS 名稱映射到負載等化器 DNS 名稱。
- *Amazon CloudFront* 提供邊緣緩存，支援大量內容。
- 負載均衡器在展示層的 Auto Scaling 組中跨 Web 伺服器分配流量。
- 另一個負載等化器在應用程式層的 Auto Scaling 組中跨後端應用程式伺服器分配流量。
- *Amazon ElastiCache* 為應用程式提供了記憶體中資料緩存，從資料庫層中刪除負載。

第 5 節要點

- 資料庫緩存可消除主要資料庫面臨的不必要壓力（通常是頻繁訪問的讀取資料），以此為其提供助益
- DAX 是適用于 DynamoDB 的完全託管且高度可用的記憶體中緩存，可實現高達 10 倍的性能提升 – 從毫秒到微秒
- Amazon ElastiCache 是可以用作記憶體中資料存儲的端緩存，支持要求最嚴苛且需要亞毫秒級回應速度的應用程式

本模組中這節內容的要點包括：

- 資料庫緩存可消除主要資料庫面臨的不必要壓力（通常是頻繁訪問的讀取資料），以此為其提供助益
- DAX 是適用于 DynamoDB 的完全託管且高度可用的記憶體中緩存，可實現高達 10 倍的性能提升 – 從毫秒到微秒
- Amazon ElastiCache 是可以用作記憶體中資料存儲的端緩存，支持要求最嚴苛且需要亞毫秒級回應速度的應用程式

模組 11：緩存內容

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：

- 確定如何通過緩存內容提升應用程式性能並減少延遲
- 創建使用 Amazon CloudFront 來緩存內容的架構
- 確定如何設計通過邊緣網站實現分配和分散式拒絕服務 (DDoS) 保護的架構
- 識別會話管理與緩存的關係
- 描述如何設計使用 Amazon ElastiCache 的架構

總體來說，您在本模組中學習了如何：

- 確定如何通過緩存內容提升應用程式性能並減少延遲
- 創建使用 Amazon CloudFront 來緩存內容的架構
- 確定如何設計通過邊緣網站實現分配和分散式拒絕服務 (DDoS) 保護的架構
- 識別會話管理與緩存的關係
- 描述如何設計使用 Amazon ElastiCache 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

某家公司正在開發使用無狀態 Web 伺服器的高度可用的 Web 應用程式。哪些服務適合存儲會話狀態資料？（請選擇兩項。）

A. Amazon CloudWatch
B. Amazon DynamoDB
C. Elastic Load Balancing
D. Amazon ElastiCache
E. AWS Storage Gateway

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 B (Amazon DynamoDB) 和 D (Amazon ElastiCache)： DynamoDB 和 ElastiCache 均可提供鍵值對高性能存儲。CloudWatch 和 Elastic Load Balancing 都不是存儲服務。AWS Storage Gateway 雖然是存儲服務，不過它是一種混合存儲服務，本地應用程式可以借助它來使用雲存儲。

感謝您完成本模組的學習。