



使用 Amazon DynamoDB 建模資料

# AWS 規定指引



# AWS 規定指引: 使用 Amazon DynamoDB 建模資料

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商標和商業外觀不得用於任何非 Amazon 的產品或服務，也不能以任何可能造成客戶混淆、任何貶低或使 Amazon 名譽受損的方式使用 Amazon 的商標和商業外觀。所有其他非 Amazon 擁有的商標均為其各自擁有者的財產，這些擁有者可能隸屬於 Amazon，或與 Amazon 有合作關係，或由 Amazon 贊助。

# Table of Contents

簡介 .....	1
處理流程 .....	2
拉齊矩陣 .....	2
處理步驟 .....	4
步驟 1. 識別使用案例和邏輯資料模型 .....	4
目標 .....	4
處理 .....	4
工具和資源 .....	4
拉西 .....	5
輸出 .....	5
步驟 2. 建立初步成本估算 .....	5
目標 .....	5
處理 .....	5
工具和資源 .....	6
拉西 .....	6
輸出 .....	6
步驟 3. 識別您的資料存取模式 .....	6
目標 .....	6
處理 .....	6
工具和資源 .....	7
拉西 .....	7
輸出 .....	7
範例 .....	8
步驟 4. 識別技術要求 .....	8
目標 .....	8
處理 .....	8
工具和資源 .....	8
拉西 .....	9
輸出 .....	9
步驟 5. 建立資料模型 .....	9
目標 .....	9
處理 .....	9
工具和資源 .....	10
拉西 .....	11

輸出 .....	11
範例 .....	11
步驟 6. 創建數據查詢 .....	12
目標 .....	12
處理 .....	12
工具和資源 .....	12
拉西 .....	12
輸出 .....	13
範例 .....	13
步驟 7. 驗證資料模型 .....	13
目標 .....	13
處理 .....	13
工具和資源 .....	13
拉西 .....	14
輸出 .....	14
步驟 8. 檢閱成本估算 .....	14
目標 .....	14
處理 .....	14
工具和資源 .....	14
拉西 .....	15
輸出 .....	15
步驟 9. 部署資料模型 .....	15
目標 .....	15
處理 .....	15
工具和資源 .....	15
拉西 .....	15
輸出 .....	16
範例 .....	16
範本 .....	18
業務需求評估範本 .....	18
技術要求評估範本 .....	21
訪問模式模板 .....	25
範本 .....	25
最佳實務 .....	28
分層數據建模 .....	29
步驟 1：識別使用案例和邏輯資料模型 .....	29

步驟 2：建立初步成本估算 .....	31
步驟 3：識別您的資料存取模式 .....	31
步驟 4：確定技術要求 .....	32
步驟 5：建立 DynamoDB 資料模型 .....	32
將元件儲存在資料表中 .....	33
GSI1 索引 .....	34
GSI2 索引 .....	35
步驟 6：建立資料查詢 .....	35
步驟 7：驗證資料模型 .....	39
步驟 8：檢視成本估算 .....	40
目標 .....	40
處理 .....	40
步驟 9：部署資料模型 .....	40
其他資源 .....	42
貢獻者 .....	44
文件歷史紀錄 .....	45
詞彙表 .....	46
# .....	46
A .....	46
B .....	49
C .....	50
D .....	52
E .....	55
F .....	57
G .....	58
H .....	59
I .....	60
L .....	62
M .....	62
O .....	65
P .....	67
Q .....	69
R .....	69
S .....	72
T .....	74
U .....	76

---

V .....	76
W .....	76
Z .....	77
.....	lxxviii

# 使用 Amazon DynamoDB 建模資料

## 流程、範本和最佳做法

Amazon Web Services ( AWS )

2023 年 12 月 ([文件歷史記錄](#))

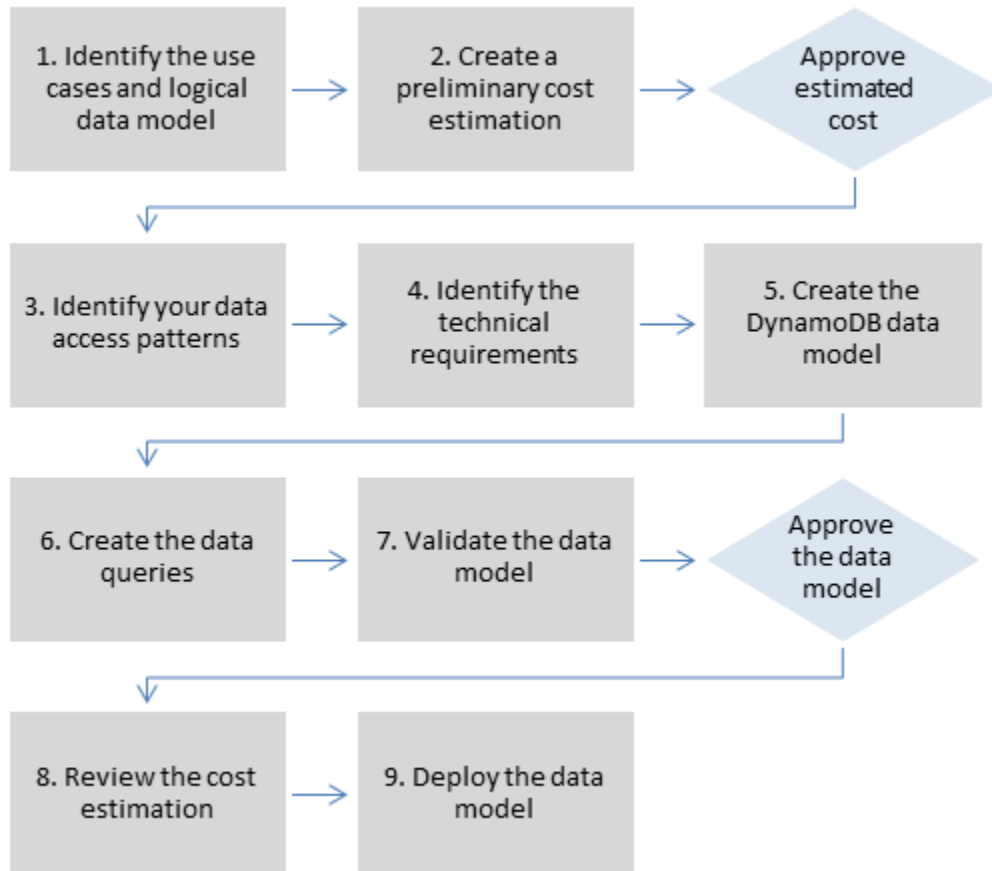
NoSQL 資料庫為建置現代應用程式提供彈性的結構描述。它們因其易於開發，功能和大規模性能而受到廣泛認可。Amazon DynamoDB 可為亞馬 Amazon Web Services (AWS) 雲端中的 NoSQL 資料庫提供快速且可預測的效能，並具備無縫的可擴展性。DynamoDB 是完全受控的資料庫服務，可協助您減輕操作和擴展分散式資料庫的管理負擔。您不必擔心硬體佈建、設定與組態、複寫、軟體修補或叢集擴展等問題。

NoSQL 架構設計需要與傳統的關聯式資料庫管理系統 (RDBMS) 設計不同的方法。RDBMS 數據模型側重於數據的結構及其與其他數據的關係。NoSQL 資料建模著重於存取模式，或應用程式將如何使用資料，因此它會以支援簡單查詢作業的方式儲存資料。對於 RDBMS，如 Microsoft SQL 服務器或 IBM Db2，您可以創建一個標準化的數據模型，而無需考慮太多關於訪問模式。您可以擴充資料模型，以便稍後支援您的模式和查詢。

本指南介紹使用 DynamoDB 的資料建模程序，可提供功能需求、效能和有效成本。本指南適用於計劃使用 DynamoDB 做為其執行應用程式的操作資料庫的資料庫工程師。AWS 專業服務使用建議的程序來協助企業公司針對不同的使用案例和工作負載進行 DynamoDB 資料建模。

# 資料建模流程

我們建議您在使用 Amazon DynamoDB 建立資料模型時執行下列程序。這些步驟將在[本指南後面](#)詳細討論。



## 拉齐矩阵

有些組織會使用職責指定矩陣 (也稱為 RACI 矩陣) 來描述特定專案或業務處理中涉及的各种角色。本指南提供建議的 RACI 矩陣，可協助您的組織針對 DynamoDB 資料建模程序識別適當的人員和正確的職責。對於過程中的每個步驟，它列出了利益相關者及其參與：

- R — 負責完成步驟
- A — 負責批准和簽署的工作
- C-諮詢為任務提供輸入
- 我 — 通知進度，但沒有直接參與任務



根據您組織和專案團隊的結構，以下 RACI 矩陣中的角色可由相同的利益相關者執行。在某些情況下，利益相關者對特定步驟負責和負責。例如，資料庫工程師可以負責建立和核准資料模型，因為這是他們的領域區域。

處理步驟	商業用戶	業務分析師	方案架構師	数据库工程	应用开发人	DevOps 工程師
1. 識別使用案例和邏輯資料模型	C	R/A	I	R		
2. 建立初步成本估算	C	A	I	R		
3. 識別您的資料存取模式	C	A	I	R		
4. 識別技術要求	C	C	A	R		
5. 建立資料模型	I	I	I	R/A		
6. 創建數據查詢	I	I	I	R/A	R	
7. 驗證資料模型	A	R	I	C		
8. 檢閱成本估算	C	A	I	R		
9. 部署動DynamoDB資料模型	I	I	C	C		R/A

# 數據建模過程步驟

本節詳細說明建議適用於 Amazon DynamoDB 的資料建模程序的每個步驟。

## 主題

- [步驟 1. 識別使用案例和邏輯資料模型](#)
- [步驟 2. 建立初步成本估算](#)
- [步驟 3. 識別您的資料存取模式](#)
- [步驟 4. 識別技術要求](#)
- [步驟 5. 建立資料模型](#)
- [步驟 6. 創建數據查詢](#)
- [步驟 7. 驗證資料模型](#)
- [步驟 8. 檢閱成本估算](#)
- [步驟 9. 部署資料模型](#)

## 步驟 1. 識別使用案例和邏輯資料模型

### 目標

- 收集需要 NoSQL 資料庫的業務需求和使用案例。
- 通過使用實體關係 ( ER ) 圖定義邏輯數據模型。

### 處理

- 業務分析師會訪問業務用戶，以確定用例和預期結果。
- 數據庫工程師創建概念數據模型。
- 數據庫工程師創建邏輯數據模型。
- 數據庫工程師收集有關項目大小，數據量和預期的讀寫吞吐量的信息。

### 工具和資源

- 業務需求評估 ( 見 [模板](#) )

- 訪問模式矩陣 ( 見 [模板](#) )
- 建立圖表的首選工具

## 拉西

商業用戶	業務分析師	方案架構師	数据库工程	应用开发人	DevOps 工程師
C	R/A	I	R		

## 輸出

- 記錄的使用案例和業務需求
- 邏輯數據模型 ( ER 圖 )

## 步驟 2. 建立初步成本估算

### 目標

- 為 DynamoDB 開發初步成本估算。

### 處理

- 資料庫工程師使用 [DynamoDB 定價](#) 頁面上提供的可用資訊和範例建立初始成本分析。
  - 建立隨需容量的成本估算 (請參閱 [範例](#))。
  - 建立已佈建容量的成本預估 (請參閱 [範例](#))。
    - 對於佈建的容量模型，請從計算器取得預估成本，並針對預留容量套用 discount。
  - 比較兩個容量模型的估計成本。
  - 創建所有環境 ( 開發，產品，質量保證 ) 的估計。
- 業務分析師會審核並核准或拒絕初步成本估算。

## 工具和資源

- [AWS 價格計算器](#)

## 拉西

商業用戶	業務分析師	方案架構師	数据库工程	应用开发人	DevOps 工程師
C	A	I	R		

## 輸出

- 初步成本估算

## 步驟 3。識別您的資料存取模式

訪問模式或查詢模式定義了用戶和系統如何訪問數據以滿足業務需求。

## 目標

- 記錄資料存取模式。

## 處理

- 數據庫工程師和業務分析師訪問最終用戶，以確定如何使用數據訪問模式矩陣模板查詢數據。
  - 對於新的應用程序，他們審查有關活動和目標的用戶故事。他們記錄使用案例並分析使用案例所需的存取模式。
  - 對於現有的應用程序，他們分析查詢日誌，以了解人們當前如何使用該系統並識別關鍵的訪問模式。
- 數據庫工程師識別訪問模式的以下屬性：
  - 資料大小：瞭解一次儲存和要求多少資料，有助於判斷最有效的資料分割方式 (請參閱[部落格文章](#))。

- 資料形狀：NoSQL 資料庫會組織資料，而非在資料處理時重新改造資料 (如 RDBMS 系統)，如此其在資料庫中的狀態就會與將受到查詢的項目相對應。此為提升速度與可擴展性的關鍵因素。
- 資料速度：DynamoDB 會隨著增加程序查詢可用的實體分割區數與有效地在這些分割區間散佈資料來擴展。事先瞭解尖峰查詢負載可能有助於判斷如何分割資料以充分利用 I/O 容量。
- 企業用戶優先考慮訪問或查詢模式。
  - 優先級查詢通常是最常用或最相關的查詢。識別需要較低回應延遲的查詢也很重要。

## 工具和資源

- 訪問模式矩陣 ( 見 [模板](#) )
- [選擇正確的分割區金鑰](#) (AWS 資料庫部落格)
- [適用於動態支援的 NoSQL 設計](#) (文件集)

## 拉西

商業用戶	業務分析師	方案架構師	数据库工程	应用开发人	DevOps 工程師
C	A	I	R		

## 輸出

- 資料存取模式矩陣

## 範例

訪問模式	優先順序	讀取或寫入	Description	類型 (單一項目、多個項目或全部)	關鍵屬性	篩選條件	結果排序
創建用戶概況	高	寫入	使用者建立新的設定檔	單一項目	使用者名稱	N/A	N/A
更新用戶概況	中	寫入	使用者更新其設定檔	單一項目	使用者名稱	用戶名 = 當前用戶	N/A

## 步驟 4. 識別技術要求

### 目標

- 收集 DynamoDB 資料庫的技術需求。

### 處理

- 商業分析師會訪問業務用戶和 DevOps 團隊，使用評估問卷收集技術要求。

### 工具和資源

- 技術需求評估 ( 見 [問卷範例](#) )

## 拉西

商業用戶	業務分析師	方案架構師	数据库工程	应用开发人	DevOps 工程師
C	C	A	R		

## 輸出

- 技術要求文件

## 步驟 5. 建立資料模型

### 目標

- 建立動 DynamoDB 資料模型。

### 處理

- 數據庫工程師確定每個用例需要多少個表。我們建議在 DynamoDB 應用程式中維護盡可能少的表格。
- 根據最常見的存取模式，識別可以是兩種類型之一的主索引鍵：具有可識別資料的分割索引鍵的主索引鍵，或具有分割索引鍵和排序索引鍵的主索引鍵。排序索引鍵是用來分組和組織資料的次要索引鍵，以便在分割區內有效地查詢資料。您可以使用排序索引鍵來定義資料中的階層關係，以便在階層的任何層級進行查詢 (請參閱[部落格文章](#))。
  - 分割區索引鍵設計
    - 定義分割區索引鍵並評估其分佈。
    - 識別[寫入分片](#)以平均分配工作負載的需求。
  - 排序索引鍵設計
    - 識別排序索引鍵。
    - 識別複合排序索引鍵的需求。
    - 識別版本控制的需求。
- 根據存取模式，識別次要索引以滿足查詢需求。

- 識別[本機次要索引 \(LSI\)](#) 的需求。這些索引鍵與基底資料表具有相同的分割索引鍵，但排序索引鍵不同。
  - 對於具有 LSI 的資料表，每個分割區索引鍵值有 10 GB 的大小限制。只要任何一個分區索引鍵值的總大小不超過 10 GB，含 LSI 的表格就可以儲存任意數量的項目。
- 識別[全域次要索引 \(GSI\)](#) 的需求。這些索引具有分割索引鍵和排序索引鍵，可能與基底資料表上的索引鍵不同 (請參閱[部落格文章](#))。
- 定義索引投影。考慮投影少量的屬性，以將寫入到索引的項目大小減至最小。在此步驟中，您應該決定是否要使用下列項目：
  - [稀疏索引](#)
  - [具體化聚總查詢](#)
  - [GSI 超載重載](#)
  - [GSI 分片](#)
  - [使用 GSI 的最終一致複本](#)
- 數據庫工程師確定數據是否將包括大型項目。如果是這樣，他們[通過使用壓縮或在 Amazon 簡單存儲服務 \( Amazon S3 \) 中存儲數據來設計解決方案](#)。
- 資料庫工程師決定是否需要時間序列資料。如果是這樣，它們會使用[時間序列設計模式](#)來建模資料。
- 數據庫工程師確定 ER 模型是否包括 many-to-many 關係。如果是這樣，他們使用[鄰接表設計模式](#)來建模數據。

## 工具和資源

- [適用於 Amazon DynamoDB 的 NoSQL 工作台](#) — 提供資料建模、資料視覺化以及查詢開發和測試功能，協助您設計 DynamoDB 資料庫
- [適用於動態支援的 NoSQL 設計 \(文件集\)](#)
- [選擇正確的分割區金鑰 \(AWS 資料庫部落格\)](#)
- 在 DynamoDB [中使用次要索引的最佳做法 \(Dynamo DB 文件\)](#)
- [如何設計 Amazon DynamoDB 全球次要索引 \(AWS 資料庫部落格\)](#)



## 拉西

商業用戶	業務分析師	方案架構師	数据库工程	应用开发人	DevOps 工程師
			R/A		

## 輸出

- 滿足您的存取模式和需求的 DynamoDB 表格結構描述

## 範例

下面的屏幕截圖顯示了 NoSQL 的工作台。

Primary Key		Attributes					
Partition Key: pk	Sort Key: sk						
P1	B1	GS11-PK	GS11-SK	name	desc		
		B1	P1	The Tiki Bundle	Everything you need for an island theme party.		
P4	B2	GS11-PK	GS11-SK	name	desc		
		B2	P4	Tiki Bar Set	Be the Mai Tai master with your very own Tiki Bar.		
P2	B1	name	desc	qty	GS11-PK	GS11-SK	location
		Tiki Torch	Bamboo tiki torch, 4 ft	6	B1	P2	W1-A9-S10-B52
	name	desc	qty	GS11-PK	GS11-SK	location	
	Tiki Torch	Bamboo tiki torch, 4 ft	2	B2	P2	W1-A9-S10-B52	
	name	desc	qty	location	reorderAt	GS13-SK	
	Tiki Torch	Bamboo tiki torch, 4 ft	656	W1-A9-S10-B52	100	/GardenOutdoor/OutdoorDecor/Lighting/LanternsT	
B1	name	desc	qty	GS11-PK	GS11-SK	location	
	Tiki Statue - Pele	Tiki of the Hawaiian Fire Goddess Pele, 5 ft.	1	B1	P3	W1-A15-S6-B27	

## 步驟 6. 創建數據查詢

### 目標

- 創建主查詢來驗證數據模型。

### 處理

- 資料庫工程師在AWS區域或他們的電腦 (DynamoDB 本機) 上手動建立 DynamoDB 表格。
- 資料庫工程師將範例資料新增至 DynamoDB 資料表。
- [資料庫工程師使用適用於 Amazon DynamoDB 的 NoSQL 工作台或適用於 Java 或 Python 的AWS 開發套件來建立方面，以建立範例查詢 \(請參閱部落格文章\)。](#)

方面就像是 DynamoDB 資料表的檢視。

- 資料庫工程師和雲端開發人員使用偏好語言的 AWS Command Line Interface (AWS CLI) 或 AWS SDK 來建立範例查詢。

### 工具和資源

- 作用中AWS帳戶，以取得 DynamoDB 主控台的存取權
- [DynamoDB 本機](#) (選用)，如果您想要在電腦上建置資料庫，而不存取 DynamoDB Web 服務
- [適用於 Amazon DynamoDB 的 NoSQL 工作台](#) (下載和說明文件)
- [AWS SDK](#) 在您選擇的語言 ( JavaScript , Python , PHP , 網絡 , 紅寶石 , Java , 圍棋 , Node.js , C++ 和 SAP 阿巴 )

### 拉西

商業用戶	業務分析師	方案架構師	数据库工程	应用开发人	DevOps 工程師
I	I	I	R/A	R	

## 輸出

- [查詢動態資料表的程式碼](#)

## 範例

- [使用適用於 Java 的AWS開發套件的動態支援範例](#)
- [Python 的例子](#)
- [JavaScript例子](#)

## 步驟 7. 驗證資料模型

### 目標

- 確保數據模型將滿足您的需求。

### 處理

- 資料庫工程師會將範例資料填入 DynamoDB 表格。
- 資料庫工程師會執程式碼以查詢 DynamoDB 資料表。
- 數據庫工程師收集查詢結果。
- 資料庫工程師會收集查詢效能指標。
- 商務使用者會驗證查詢結果是否滿足業務需求。
- 業務分析師驗證技術要求。

### 工具和資源

- 作用中AWS帳戶，以取得 DynamoDB 主控台的存取權
- [DynamoDB 本機](#) (選用)，如果您想要在電腦上建置資料庫，而不存取 DynamoDB Web 服務
- AWS以您選擇的語言提供 [SDK](#)

## 拉西

商業用戶	業務分析師	方案架構師	数据库工程	应用开发人	DevOps 工程師
A	R	I	C		

## 輸出

- 核准的資料模型

## 步驟 8. 檢閱成本估算

### 目標

- 定義容量模型並估算 DynamoDB 成本，以從[步驟 2](#) 開始精簡成本估算。
- 獲得業務分析師和利益相關者的最終財務批准。

### 處理

- 數據庫工程師識別數據量估計。
- 數據庫工程師確定數據傳輸要求。
- 資料庫工程師定義所需的讀取和寫入容量單位。
- 商業分析師決定[隨需和佈建的容量模型](#)。
- 資料庫工程師識別 [DynamoDB auto](#) 擴展的需求。
- 資料庫工程師在每月成本簡易計算器工具中輸入參數。
- 數據庫工程師向業務利益相關者提供最終價格估算。
- 業務分析師和利益相關者批准或拒絕解決方案。

### 工具和資源

- [AWS 價格計算器](#)

## 拉西

商業用戶	業務分析師	方案架構師	数据库工程	应用开发人	DevOps 工程師
C	A	I	R		

## 輸出

- 容量模型
- 修訂成本估算

## 步驟 9. 部署資料模型

### 目標

- 將 DynamoDB 表 (一或多個表格) 部署到 AWS 區域

### 處理

- DevOps 架構師會為 DynamoDB 資料表 (或資料表) 建立 AWS CloudFormation 範本或其他基礎結構做為程式碼 (IaC) 工具。AWS CloudFormation 提供佈建和設定資料表和相關資源的自動化方式。

### 工具和資源

- [AWS CloudFormation](#)

## 拉西

商業用戶	業務分析師	方案架構師	数据库工程	应用开发人	DevOps 工程師
I	I	C	C		R/A

## 輸出

- AWS CloudFormation 範本

## 範例

```
mySecondDDBTable:
  Type: AWS::DynamoDB::
  Table DependsOn: "myFirstDDBTable"
  Properties:
  AttributeDefinitions:
    - AttributeName: "ArtistId"
      AttributeType: "S"
    - AttributeName: "Concert"
      AttributeType: "S"
    - AttributeName: "TicketSales"
      AttributeType: "S"
  KeySchema:
    - AttributeName: "ArtistId"
      KeyType: "HASH"
    - AttributeName: "Concert"
      KeyType: "RANGE"
  ProvisionedThroughput:
    ReadCapacityUnits:
      Ref: "ReadCapacityUnits"
    WriteCapacityUnits:
      Ref: "WriteCapacityUnits"
  GlobalSecondaryIndexes:
    - IndexName: "myGSI"
      KeySchema:
        - AttributeName: "TicketSales"
          KeyType: "HASH"
      Projection:
        ProjectionType: "KEYS_ONLY"
      ProvisionedThroughput:
        ReadCapacityUnits:
          Ref: "ReadCapacityUnits"
        WriteCapacityUnits:
          Ref: "WriteCapacityUnits"
  Tags:
    - Key: mykey
```

Value: myvalue

# 範本

本節提供的範本是根據網站上[使用 Amazon DynamoDB 建模玩家資料](#)的模型。AWS

## Note

本節中的表格使用 MM 作為百萬的縮寫，K 作為千的縮寫。

## 主題

- [業務需求評估範本](#)
- [技術要求評估範本](#)
- [訪問模式模板](#)

## 業務需求評估範本

提供使用案例的說明：

### 描述

想像一下，您正在構建一個在線多人遊戲。在您的遊戲中，由 50 名玩家組成的團隊加入一個遊戲階段來進行遊戲，這通常需要大約 30 分鐘的時間才能進行遊戲。在遊戲過程中，您必須更新特定玩家的記錄，以指出玩家已經玩過的時間、他們的統計數據，或者他們是否贏得比賽。用戶希望查看他們玩過的早期遊戲，以查看遊戲的獲勝者或觀看每個遊戲的動作的重播。

提供有關您的使用者的資訊：

使用者	Description	預期人數
玩家	線上玩家。	1 毫米
開發團隊	內部團隊，將使用遊戲統計數據來提高遊戲體驗。	100



提供有關資料來源以及如何擷取資料的資訊：

來源	Description	使用者
在線遊戲	遊戲玩家將創建配置文件並開始新遊戲。	玩家
遊戲應用	遊戲應用程序將自動收集有關遊戲的統計信息，例如開始和結束時間，玩家數量，每個玩家的位置以及遊戲的地圖。	

提供有關如何使用數據的信息：

消費者	Description	使用者
在線遊戲	遊戲玩家將查看配置文件並查看其遊戲統計信息。	玩家
資料分析	遊戲開發團隊將提取遊戲統計數據以進行數據分析並改善用戶體驗。資料將從資料存放區匯出並匯入 Amazon S3，以透過 Spark 應用程式支援分析。	開發團隊

提供實體清單及其識別方式：

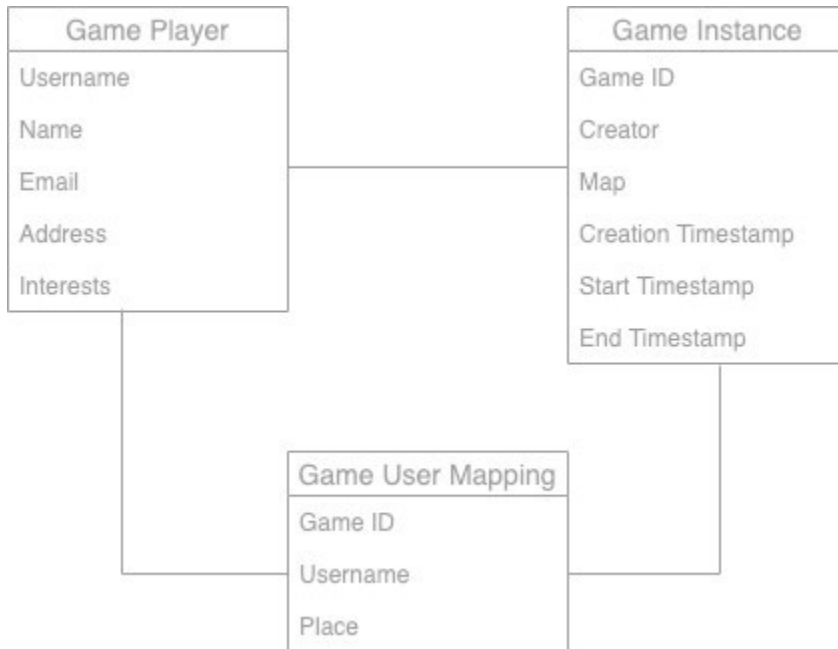
實體名稱	Description	識別符
玩家	儲存每個使用者 (玩家) 的身分識別、地址、人口統計資料、興趣等資訊。	使用者名稱
遊戲實例	提供有關每個遊戲的信息，包括創作者，開始，結束和地圖 YPLAYED。	遊戲識別碼

遊戲使用者對應

代表用戶和遊戲之間的 many-to-many 關係。

遊戲 ID 和使用者名稱

為實體建立 ER 模型：



提供實體的高階統計資料：

實體名稱	估計記錄數	記錄大小	備註
玩家	1 毫米	小於 1 KB	遊戲平台擁有大約 1 MM 用戶。
遊戲實例	6 毫米 (10 萬/天 * 60 天)	小於 1 KB	平均而言，每天有 10 萬場比賽。我們需要存儲最近的 60 天。
遊戲使用者對應	三百毫米 (6 公釐遊戲 * 50 位玩家)	小於 1 KB	平均而言，每個遊戲都有 50 名玩家，我們需要存儲信息。

## 技術要求評估範本

提供有關資料擷取類型的資訊：

資料擷取類型	Y/N	Description	Frequency (頻率)
應用程式存	Y		
API 閘道	Y		
資料串流	N		
Batch 處理	N		
ETL	N		
資料匯入	N		
時間序列	N		


提供有關資料使用類型的資訊：

資料消耗類型	Y/N	Description	Frequency (頻率)
應用程式存			
API 閘道			
資料匯出			
資料分析			
資料彙總			
報告			
搜尋			
資料串流			

## ETL

提供資料量估算：

實體名稱	估計記錄數	記錄大小	資料量
玩家	1 毫米	小於 1 KB	大約 1 GB ( 1 毫米 * 1 KB )
遊戲實例	6 毫米 (100 克/天 * 60 天)	小於 1 KB	約 6 GB ( 6 毫米 * 1 KB )
遊戲使用者對應	三百毫米 (6 公釐遊戲 * 50 位玩 家)	小於 1 KB	約三百 GB ( 300 毫米 * 1 千克 )

 Note

資料保留期限為 60 天。60 天之後，資料必須存放在 Amazon S3 以進行分析，方法是使用 [DynamoDB 存留時間 \(TTL\)](#) 將資料自動從 DynamoDB 移出到 Amazon S3。

回答以下有關時間模式的問題：

- 用戶可以使用該應用程序的時間範圍（例如，工作日的 24/7 或上午 9 點到下午 5 點）？
- 白天的使用量是否有高峰？多少個小時？應用程式使用量的百分比是多少？

指定寫入輸送量需求：

實體名稱	寫入/日	小時/天	寫入次數/秒
玩家	1 萬次更新	18	< 1
遊戲實例	30 萬	18	< 5

遊戲使用者對應	1,800,000,000	18	~ 27.777
---------	---------------	----	----------

### 備註

玩家寫入操作：1% 的使用者每天都會更新他們的設定檔，因此我們預計 1,000,000 名使用者可獲得 10,000 次更新。

遊戲實例寫入操作：每天 10 萬遊戲。對於每個遊戲，我們至少有 3 個寫入操作-在創建，開始和結束-所以總共有 300,000 個寫操作。

遊戲用戶映射寫入操作：每場遊戲每天有 50 個玩家，每天 100,000 場比賽。平均遊戲持續時間為 30 分鐘，玩家位置每 5 秒更新一次。我們估計每位玩家平均可獲得 360 次更新，因此總計為  $10 \text{ 萬} * 50 * 360 = 1,800,000,000$  次寫入操作。

指定讀取輸送量需求：

實體名稱	讀取/天	小時/天	讀取數/秒
玩家	200,000	18	3
遊戲實例	5,000,000	18	
遊戲使用者對應	1,800,000,000	18	~ 27.777

### 備註

玩家讀取操作：20% 的使用者開始遊戲，所以  $1 \text{ 萬} * 0.2 = 20 \text{ 萬}$ 。

遊戲實例讀取操作：每天 10 萬遊戲。對於每個遊戲，我們每個玩家至少有 1 次讀取操作，每場比賽有 50 個玩家，所以總共有 5,000,000 次讀取操作。

遊戲用戶映射讀取操作：100,000 場比賽/天，50 名玩家。平均遊戲持續時間為 30 分鐘，玩家位置每 5 秒更新一次。我們估計每位玩家平均會有 360 次更新，而且每個更新都需要讀取作業，因此總計為  $100,000 \text{ 個} * 50 * 360 = 1,800,000,000$  次讀取作業。

指定資料存取延遲需求：

操作	百分位數	最大延遲
讀取	30 毫秒	一百毫秒
寫入	10 毫秒	50 毫秒

## 指定資料可用性需求：

要求	Y/N	指標	備註
高可用性	Y	99.9%	
RTO	Y	1 小時	恢復時間目標
RPO	Y	1 小時	復原點目標
災難復原	N		
區域內資料複製	N		
跨區域資料複製	N	3 秒延遲	哪個AWS 區域？

## 指定安全性需求：

要求	Y/N	備註
敏感資料存放區	N	受保護的健康信息 ( PHI ) ，支付卡行業 ( PCI ) 信息，個人身份信息 ( PII ) ？
靜態加密	Y	
傳輸中加密	Y	
用戶端加密	N	
任何專有或第三方供應商加密庫	N	

資料存取記錄	N
資料存取稽核	N

## 訪問模式模板

使用下列欄位來收集並記錄使用案例存取模式的相關資訊：

欄位	描述
訪問模式	提供存取模式的名稱。
Description	提供存取模式的更詳細描述。
優先順序	定義存取模式的優先順序 (高、中或低)。這會定義應用程式最相關的存取模式。
讀取或寫入	它是讀取訪問還是寫訪問模式？
類型	模式是否存取單一項目、多個項目或所有項目？
過濾器	訪問模式是否需要任何過濾器？
排序	結果是否需要任何排序？

## 範本

訪問模式	Description	優先順序	讀取或寫入	類型 ( 單一項目， 多 項目，或 全部)	關鍵屬性	篩選條件	結果排序
建立使用者設定	用戶創建一個新的	高	寫入	單一項目	使用者名稱	無	無

	配置文件。							
更新使用者設定	使用者更新其設定檔。	中性	寫入	單一項目	使用者名稱	用戶名 = 當前用戶	無	
取得使用者設定	用戶審查他們的個人資料	高	讀取	單一項目	使用者名稱	用戶名 = 當前用戶	無	
建立遊戲	用戶創建一個新的遊戲。	高	寫入	單一項目	游戏 ID	無	無	
尋找開放式遊戲	使用者搜尋開放式遊戲。搜尋結果會依開始時間戳記以遞減順序排序。	高	讀取	多個項目		GameStatus = 開啟	開始時間戳記子代	
透過地圖尋找開放式遊戲	用戶通過使用按開始時間戳記降序排序的特定地圖來搜索打開的遊戲  順序。	中性	讀取	多個項目		GameStatus = 開啟 與地圖 = XYZ	開始時間戳記子代	
查看遊戲	用戶評論遊戲的詳細信息。	高	讀取	單一項目	游戏 ID	無	無	



檢視遊戲中的使用者	用戶獲取遊戲中所有用戶的列表。	中性	讀取	多個項目		遊戲 ID = XYZ	無	
加入使用者加入遊戲	使用者加入開啟的遊戲。	高	寫入	單一項目	遊戲 ID 和用戶名	GameStatus = 開啟	無	
開始遊戲	用戶開始一個新的遊戲。	高	寫入	單一項目	游戏 ID	無	無	
為使用者更新遊戲	更新用戶在遊戲中的位置。	中性	寫入	單一項目	遊戲 ID 和用戶名	無	無	
更新遊戲	遊戲結束；更新統計資料。	中性	寫入	單一項目	游戏 ID	無	無	
尋找使用者過去的所有遊戲	列出使用者玩過的所有遊戲，並按照遊戲的開始時間戳記排序。	低	讀取	多個項目	使用者名稱和遊戲 ID	用戶名 = 當前用戶	開始時間戳	
匯出資料以進行資料分析	開發團隊將執行批次任務，將資料匯出到 Amazon S3。	低	讀取	全部	無	無	無	

# 最佳實務

請考慮使用下列 DynamoDB 設計最佳做法：

- [分割區索引鍵設計](#) – 使用高基數分割區索引鍵來均勻分配負載。
- [鄰接表設計模式](#)-使用此設計模式進行管理 one-to-many 和 many-to-many 關係。
- [稀疏索引](#) — 對全域次要索引 (GSI) 使用稀疏索引。在您建立 GSI 時，指定一個分割區索引鍵和 (選用) 一個排序索引鍵。只有在基本資料表中包含對應 GSI 分割區索引鍵的項目才會出現在稀疏索引中。這有助於保持 GSI 更小。
- [索引過載](#) – 使用相同的 GSI 對各種類型的項目編製索引。
- [GSI 寫入碎片](#) – 明智地進行碎片以跨分割區分佈資料，以實現高效、更快的查詢。
- [大型項目](#) – 僅將中繼資料儲存在表內，將 Blob 儲存在 Amazon S3 中，並將參考保留在 DynamoDB 中。將大型項目分解為多個項目，並使用排序索引鍵有效率地編製索引。

如需更多設計最佳實務，請參閱 [Amazon DynamoDB](#) 文件。

# 階層式資料建模範例

以下各節使用汽車公司範例來說明如何使用資料建模程序步驟在 DynamoDB 中設計多層元件管理系統。

## 主題

- [步驟 1：識別使用案例和邏輯資料模型](#)
- [步驟 2：建立初步成本估算](#)
- [步驟 3：識別您的資料存取模式](#)
- [步驟 4：確定技術要求](#)
- [步驟 5：建立 DynamoDB 資料模型](#)
- [步驟 6：建立資料查詢](#)
- [步驟 7：驗證資料模型](#)
- [步驟 8：檢視成本估算](#)
- [步驟 9：部署資料模型](#)

## 步驟 1：識別使用案例和邏輯資料模型

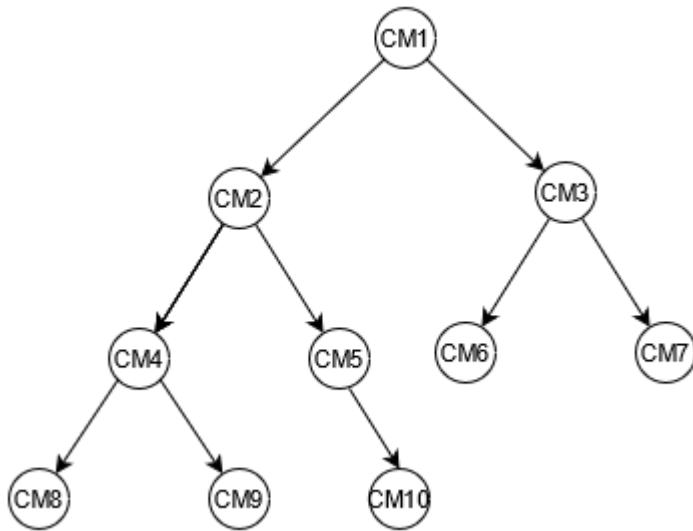
一家汽車公司希望建立一個交易部件管理系統，以存儲和搜索所有可用的汽車零件，並在不同的組件和零件之間建立關係。例如，一輛汽車包含多顆電池，每顆電池包含多個高層級模組，每個模組包含多個電芯，每個電芯包含多個低階元件。

一般來說，為了建置階層關係模型，[Amazon Neptune](#) 等圖形資料庫是更好的選擇。然而，在某些情況下，Amazon DynamoDB 因其靈活性、安全性、效能和可擴展性而成為階層式資料模型的更佳替代方案。

例如，您可能會建置一個系統，其中 80-90% 的查詢是交易，DynamoDB 非常適合該系統。在此範例中，另外 10-20% 的查詢是關聯式的，其中像 Neptune 這樣的圖形資料庫更適合。在這種情況下，在架構中包含一個額外的數據庫以僅滿足 10-20% 的查詢可能會增加成本。它還增加了維護多個系統和同步數據的操作負擔。相反，您可以在 DynamoDB 中對該 10-20% 的關聯式查詢建模。

繪製汽車元件的範例樹狀目錄可以協助您映射它們之間的關係。下列圖表顯示具有四個層級的相依性圖形。CM1 是範例汽車本身的頂層元件。它具有兩個子元件，用於兩個範例電池即 CM2 和 CM3。每個電池都具有兩個子元件，即模組。CM2 具有模組 CM4 和 CM5，而 CM3 具有模組 CM6 和 CM7。每

個模組都具有幾個子元件，即電芯。CM4 模組具有兩個電芯，即 CM8 和 CM9。CM5 具有一個電芯即 CM10。CM6 和 CM7 還沒有任何關聯的電芯。



本指南將使用此樹狀目錄及其元件識別符作為參考。頂端元件將稱為父項，子元件將稱為子項。例如，頂端元件 CM1 是 CM2 和 CM3 的父項。CM2 是 CM4 和 CM5 的父項。這描繪了父項和子項關係。

從樹狀目錄中，您可以看到元件的完整相依性圖形。例如，CM8 依賴於 CM4，CM4 依賴於 CM2，CM2 依賴於 CM1。此樹狀目錄將完整的相依性圖形定義為路徑。路徑描述了以下兩個項目：

- 相依性圖形
- 樹狀目錄中的位置

填寫業務需求的模板：

提供有關您的使用者的資訊：

使用者	Description
僱員	需要汽車及其組件信息的汽車公司的內部員工

提供有關資料來源以及如何擷取資料的資訊：

來源	Description	使用者
----	-------------	-----

管理系統	該系統將存儲與可用汽車零件及其與其他組件和零件之間的關係相關的所有數據。	僱員
------	--------------------------------------	----

提供有關如何使用數據的信息：

消費者	Description	使用者
管理系統	擷取父元件 ID 的所有直接子元件。	僱員
管理系統	擷取元件 ID 之所有子元件的遞迴清單。	僱員
管理系統	請參閱元件的祖先。	僱員

## 步驟 2：建立初步成本估算

計算應用程式所有環境的成本估算是非常重要的，因此您可以檢查解決方案是否在財務上是可行的。最佳做法是進行高層次的估計，並在開發和部署之前獲得業務分析師的批准。

- 資料庫工程師使用 [DynamoDB 定價](#) 頁面上提供的可用資訊和範例建立初始成本分析。
  - 建立隨需容量的成本估算 (請參閱[範例](#))。
  - 建立已佈建容量的成本預估 (請參閱[範例](#))。
    - 對於佈建的容量模型，請從計算器取得預估成本，並針對預留容量套用 discount。
  - 比較兩個容量模型的估計成本。
  - 創建所有環境 (開發，產品，質量保證) 的估計。
- 業務分析師會審核並核准或拒絕初步成本估算。

使用這些參考值，您可以建立預估價格以供核准。若要建立預算，您可以使用 [DynamoDB 定價頁面](#) 和 [AWS 定價計算器](#)。

## 步驟 3：識別您的資料存取模式

此範例使用案例具有下列存取模式，用於管理不同汽車零組件之間的關係。

訪問模式	優先順序	讀取或寫入	Description (描述)	類型	篩選條件	結果排序
直系子女	高	讀取	擷取父元 件 ID 的所 有直接子元 件。	多個	Component ID	N/A
所有子元件	高	讀取	擷取元件 ID 之所有 子元件的遞 迴清單。	多個	Component ID	N/A
祖先	高	讀取	擷取元件的 祖先。	多個	Component ID	N/A

## 步驟 4：確定技術要求

這個範例沒有任何特定的技術需求，這些需求不在本範例的範圍內。在實際情況下，最佳做法是完成此步驟，並在繼續開發和部署之前驗證所有技術需求是否符合。您可以使用[範例問卷](#)來完成業務案例中的此步驟。此外，我們建議您驗證 [DynamoDB 服務配額](#)，以確保您設計的解決方案沒有硬性限制。

## 步驟 5：建立 DynamoDB 資料模型

定義基底資料表和全域次要索引 (GSI) 的分割區索引鍵：

- 遵循關鍵設計最佳實務，在此範例中用ComponentId作基底資料表的分割索引鍵。因為它是獨一無二的，所ComponentId以可以提供粒度。DynamoDB 使用分割區索引鍵的雜湊值來決定實際儲存資料的分割區。唯一元件 ID 會產生不同的雜湊值，可加速分佈資料表內的資料。您可以使用ComponentId分區索引鍵來查詢基底資料表。
- 若要尋找元件的直接子系，請建立 GSI，其中ParentId是分割索引鍵，並且ComponentId是排序索引鍵。您可以使用ParentId做為分割區索引鍵來查詢此 GSI。
- 若要尋找元件的所有遞迴子項，請建立 GSI，其中 GraphId 是分割區索引鍵，Path 是排序索引鍵。您可以使用 GraphId 作為分割區索引鍵並在排序索引鍵上使用 BEGINS\_WITH(Path, "\$path") 運算子來查詢此 GSI。

	分割區索引鍵	排序鍵	映射屬性
基底資料表	ComponentId		ParentId, GraphId, Path
GS1	ParentId	ComponentId	
GS2	GraphId	Path	ComponentId

## 將元件儲存在資料表中

下一步是將每個元件儲存在 DynamoDB 基本資料表中。從範例樹中插入所有元件之後，您會取得下列基底資料表。

ComponentId	ParentId	GraphId	路徑
CM1		CM1#1	CM1
CM2	CM1	CM1#1	CM1 CM2
CM3	CM1	CM1#1	CM1 CM3
CM4	CM2	CM1#1	CM1 CM2 CM4
CM5	CM2	CM1#1	CM1 CM2 CM5
CM6	CM3	CM1#1	CM1 CM3 CM6

CM7	CM3	CM1#1	CM1 CM3 CM7
CM8	CM4	CM1#1	CM1 CM2 CM4 CM8
CM9	CM4	CM1#1	CM1 CM2 CM4 CM9
CM10	CM5	CM1#1	CM1 CM2 CM5 CM10

## GS11 索引

若要檢查元件的所有直接子系，您可以建立用ParentId作分割索引鍵和ComponentId排序索引鍵的索引。下列樞紐分析表呈現 GS11 索引。您可以使用此索引透過父元件 ID 擷取所有直屬子系元件。例如，您可以了解汽車 (CM1) 中有多少顆可用電池，或模組 (CM4) 中有哪些電芯可用。

ParentId	ComponentId
CM1	CM2
	CM3
	CM4
CM2	CM5
	CM6
CM3	CM7
	CM8
CM4	CM9



CM5

CM10

## GS12 索引

下列樞紐分析表呈現 GS12 索引。它使用 GraphId 作為分割區索引鍵和使用 Path 作為排序索引鍵進行設定。使用 GraphId 和 sort 鍵 (Path) 上的begins\_with操作，您可以在樹中找到組件的完整歷程。

GraphId	路徑	ComponentId
CM1#1	CM1	CM1
	CM1 CM2	CM2
	CM1 CM3	CM3
	CM1 CM2 CM4	CM4
	CM1 CM2 CM5	CM5
	CM1 CM2 CM4 CM8	CM8
	CM1 CM2 CM4 CM9	CM9
	CM1 CM2 CM5 CM10	CM10
	CM1 CM3 CM6	CM6
	CM1 CM3 CM7	CM7

## 步驟 6：建立資料查詢

定義存取模式並設計資料模型後，您可以在 DynamoDB 資料庫中查詢階層式資料。為了節省成本並協助確保效能的最佳作法，下列範例僅使用不含查詢作業的查詢作業Scan。

- 尋找元件的祖先。

若要尋找 CM8 元件的上階 (父、祖父、曾祖父等)，請使用 ComponentId = "CM8" 查詢基本資料表。此查詢會傳回下列記錄。

若要減少結果資料的大小，您可以使用投射表達式以僅傳回 Path 屬性。

ComponentId	ParentId	GraphId	路徑
CM8	CM4	CM1#1	CM1 CM2 CM4 CM8

路徑

CM1|CM2|CM4|CM8

現在，使用管道 (「|」) 拆分路徑，並採取第一個 N-1 組件以獲取祖先。

查詢結果：CM8 的上階是 CM1、CM2、CM4。

- 尋找元件的直接子系。

若要取得 CM2 元件的所有直接子元件或單層下游元件，請使用查詢 GSI1。ParentId = "CM2" 此查詢會傳回下列記錄。

ParentId	ComponentId
CM2	CM4
	CM5

- 使用頂層元件尋找所有下游子元件。

若要取得頂層元件 CM1 的所有子元件或下游元件，請使用和查詢 GSI2begins\_with("Path", "CM1|"), GraphId = "CM1#1" 然後使用投影運算式與 ComponentId 它將傳回與該樹狀目錄相關的所有元件。

此範例具有單一樹狀目錄，其中 CM1 為頂端元件。實際上，您在相同表格中可能具有數百萬個頂層元件。

GraphId	ComponentId
	CM2

CM1#1	CM3
	CM4
	CM5
	CM8
	CM9
	CM10
	CM6
	CM7

- 使用中層元件尋找所有下游子元件。

若要以遞迴方式取得元件 CM2 的所有子元件或下游元件，您有兩個選項。您可以用遞迴方式逐層查詢，也可以查詢 GSI2 索引。

- 以遞迴方式逐層查詢 GSI1，直到到達最後一層子元件。

1. 使用 ParentId = "CM2" 查詢 GSI1。它將傳回下列記錄。

ParentId	ComponentId
CM2	CM4
	CM5

2. 再次使用 ParentId = "CM4" 查詢 GSI1。它將傳回下列記錄。

ParentId	ComponentId
CM4	CM8
	CM9

3. 再次使用 ParentId = "CM5" 查詢 GSI1。它將傳回下列記錄。

繼續循環：查詢每個 ComponentId，直到到達最後一層。使用 ParentId = "<ComponentId>" 的查詢未傳回任何結果時，先前的結果來自樹狀目錄的最後一層。

ParentId	ComponentId
CM5	CM10

#### 4. 合併所有結果。

結果 = [CM4，平方厘米 5] + [CM8，平方厘米 9] + [CM10]

= [立方厘米 4, 厘米 5, 平方厘米 8, 平方厘米 9, 厘米 10]

- 查詢 GSI2，它儲存頂層元件 (汽車或 CM1) 的階層樹狀目錄。
  - 首先，尋找頂層元件或頂級上階和 CM2 的 Path。為此，使用 ComponentId = "CM2" 查詢基本資料表，以尋找階層樹狀目錄中該元件的路徑。選取 GraphId 和 Path 屬性。此查詢會傳回下列記錄。

GraphId	路徑
CM1#1	CM1 CM2

- 使用來查詢 GSI2。GraphId = "CM1#1" AND BEGINS\_WITH("Path", "CM1|CM2|") 此查詢會傳回下列結果。

GraphId	路徑	ComponentId
CM1#1	CM1 CM2 CM4	CM4
	CM1 CM2 CM5	CM5
	CM1 CM2 CM4 CM8	CM8
	CM1 CM2 CM4 CM9	CM9
	CM1 CM2 CM5 CM10	CM10

- 選取 ComponentId 屬性，以傳回 CM2 的所有子元件。

## 步驟 7：驗證資料模型

在此步驟中，商務使用者會驗證查詢結果，並檢查他們是否滿足業務需求。您可以使用下表根據使用者的需求來檢查存取模式。

問題	基表/GSI	查詢
作為使用者，我想要擷取父元件 ID 的所有直屬子元件。	GSI1	<pre>ParentId = "&lt;ComponentId&gt;"</pre> <p>(尋找元件的直屬子系。)</p>
作為使用者，我想要擷取元件 ID 的所有子元件的遞迴清單。	GSI1 或 GSI2	<pre>GSI1 : ParentId = "&lt;ComponentId&gt;"</pre> <p>或</p> <pre>GSI2 : GraphId = "&lt;TopLevelComponentId&gt;#N" AND BEGINS_WITH("Path", "&lt;PATH_OF_Component&gt;")</pre> <p>(使用頂層元件尋找所有下層子元件。使用中層元件尋找所有下層子元件。)</p>
作為使用者，我想要查看元件的上階。	基本資料表	<pre>ComponentId = "&lt;ComponentId&gt;"</pre> <p>，然後選取「路徑」屬性。</p> <p>(尋找元件的上階。)</p>

您也可以使用任何程式設計語言實作指令碼 (測試)，以直接查詢 DynamoDB，並將結果與預期的結果進行比較。

## 步驟 8：檢視成本估算

再次檢閱並細化成本估算。此外，與業務利益相關者進行驗證並獲得批准以進行下一步是一個很好的做法。

### 目標

- 定義容量模型，並估算 DynamoDB 成本，從[步驟 2](#) 開始精簡成本估算。
- 獲得業務分析師和利益相關者的最終財務批准。

### 處理

- 數據庫工程師識別數據量估計。
- 數據庫工程師確定數據傳輸要求。
- 資料庫工程師定義所需的讀取和寫入容量單位。
- 商業分析師決定[隨需和佈建的容量模型](#)。
- 資料庫工程師識別 [DynamoDB auto 調整規模](#) 的需求。
- 資料庫工程師在中輸入參數AWS Pricing Calculator。
- 數據庫工程師向業務利益相關者提供最終價格估算。
- 業務分析師和利益相關者批准或拒絕解決方案。

## 步驟 9：部署資料模型

對於這個特定的例子，模型的部署是使用 [NoSQL 工作台](#)，現代數據庫開發和操作的應用程序完成。使用此工具，您可以選擇建立資料模型、上傳資料，以及將資料直接部署到AWS 帳戶。如果你想實現這個例子，你可以使用下面的AWS CloudFormation模板，這是由 NoSQL 工作台生成。

```
AWSTemplateFormatVersion: 2010-09-09
Resources:
  Components:
    Type: 'AWS::DynamoDB::Table'
    Properties:
      KeySchema:
        - AttributeName: ComponentId
          KeyType: HASH
```

```
AttributeDefinitions:
  - AttributeName: ComponentId
    AttributeType: S
  - AttributeName: ParentId
    AttributeType: S
  - AttributeName: GraphId
    AttributeType: S
  - AttributeName: Path
    AttributeType: S
GlobalSecondaryIndexes:
  - IndexName: GS1
    KeySchema:
      - AttributeName: ParentId
        KeyType: HASH
      - AttributeName: ComponentId
        KeyType: RANGE
    Projection:
      ProjectionType: KEYS_ONLY
  - IndexName: GSI2
    KeySchema:
      - AttributeName: GraphId
        KeyType: HASH
      - AttributeName: Path
        KeyType: RANGE
    Projection:
      ProjectionType: INCLUDE
      NonKeyAttributes:
        - ComponentId
BillingMode: PAY_PER_REQUEST
TableName: Components
```

## 其他資源

### 有關 DynamoDB 資訊

- [DynamoDB](#)
- [說明文件](#)
- [適用於 DynamoDB 的 NoSQL 設計](#)
- [寫入分片](#)
- [本機次要索引 \(LSI\)](#)
- [全域次要索引 \(GSI\)](#)
- [超載 GSI](#)
- [GSI 分片](#)
- [使用 GSI 建立最終一致的複本](#)
- [稀疏索引](#)
- [具體化聚總查詢](#)
- [時間序列設計模式](#)
- [邻接表设计模式](#)
- [隨需和佈建的容量模型](#)
- [auto 調整](#)
- [DynamoDB 存留時間 \(TTL\)](#)
- [使用 DynamoDB \(實驗室\) 建模玩家資料](#)

### AWS 服務

- [AWS CloudFormation](#)
- [Amazon Simple Storage Service \(Amazon S3\)](#)

### 工具

- [AWS Pricing Calculator](#)
- [適用於 DynamoDB 資料庫的 NoSQL 工作台](#)
- [DynamoDB Local](#)



- [DynamoDB 和開發套件 AWS](#)

### 最佳實務

- [使用 DynamoDB 進行設計和架構的最佳實務 \(DynamoDB 文件\)](#)
- [使用次要索引的最佳做法 \(DynamoDB 文件\)](#)
- [儲存大型項目和屬性的最佳做法 \(DynamoDB 文件\)](#)
- [選擇正確的 DynamoDB 分割區金鑰 \(AWS 資料庫部落格\)](#)
- [如何設計 Amazon DynamoDB 全球次要索引 \(資料庫部落格\) AWS](#)
- [適用於 Amazon DynamoDB 的 NoSQL 工作台 \( 中型網站 \) 中有哪些方面](#)

### AWS 一般資源

- [AWS 規定指導網站](#)
- [AWS 文件](#)
- [AWS 一般參考](#)

## 貢獻者

本指南的貢獻者包括：

- 卡米洛·岡薩雷斯，高級數據架構師，AWS
- 莫努爾·阿爾馬蒙，高級大數據建築師 AWS
- 聖地亞哥塞古拉，專業服務顧問，AWS
- 撒西甚·庫馬爾，雲應用程序架構師，AWS

# 文件歷史紀錄

下表描述了本指南的重大變更。如果您想收到有關未來更新的通知，可以訂閱 [RSS 摘要](#)。

變更	描述	日期
<a href="#">已新增「最佳作法」區段和階層式資料模型的範例。</a>	<a href="#">我們新增 DynamoDB 最佳做法的摘要，以及設計和驗證階層式模型的 step-by-step 範例。</a>	2023 年 12 月 5 日
<a href="#">初次出版</a>	—	2020 年 10 月 26 日

# 《AWS 方案指引》詞彙表

以下是由《AWS 方案指引》提供的策略、指南和模式中常用的術語。若要建議項目，請使用詞彙表末尾的提供意見回饋連結。

## 數字

### 7 R

將應用程式移至雲端的七種常見遷移策略。這些策略以 Gartner 在 2011 年確定的 5 R 為基礎，包括以下內容：

- **重構/重新架構** – 充分利用雲端原生功能來移動應用程式並修改其架構，以提高敏捷性、效能和可擴展性。這通常涉及移植作業系統和資料庫。範例：將您的內部部署 Oracle 資料庫遷移至 Amazon Aurora PostgreSQL 相容版本。
- **平台轉換 (隨即重塑)** – 將應用程式移至雲端，並引入一定程度的優化以利用雲端功能。範例：將您的內部部署 Oracle 資料庫遷移至 AWS 雲端中的 Amazon Relational Database Service (Amazon RDS) for Oracle。
- **重新購買 (捨棄再購買)** – 切換至不同的產品，通常從傳統授權移至 SaaS 模型。範例：將您的客戶關係管理 (CRM) 系統遷移至 Salesforce.com。
- **主機轉換 (隨即轉移)** – 將應用程式移至雲端，而不進行任何變更以利用雲端功能。範例：將您的內部部署 Oracle 資料庫遷移至 AWS 雲端中 EC2 執行個體上的 Oracle。
- **重新放置 (虛擬機器監視器等級隨即轉移)** – 將基礎設施移至雲端，無需購買新硬體、重寫應用程式或修改現有操作。此遷移案例特定於 VMware Cloud on AWS，它支援內部部署環境與 AWS 之間的虛擬機器 (VM) 相容性和工作負載可移植性。在將基礎設施遷移至 VMware Cloud on AWS 時，您可以使用內部部署資料中心的 VMware Cloud Foundation 技術。範例：將託管 Oracle 資料庫的虛擬機器監視器重新放置到 VMware Cloud on AWS。
- **保留 (重新檢視)** – 將應用程式保留在來源環境中。其中可能包括需要重要重構的應用程式，且您希望將該工作延遲到以後，以及您想要保留的舊版應用程式，因為沒有業務理由來進行遷移。
- **淘汰** – 解除委任或移除來源環境中不再需要的應用程式。

## A

### ABAC

請參閱以[屬性為基礎的存取控制](#)。

## 抽象的服務

請參閱[受管理服務](#)。

## 酸

請參閱[原子性、一致性、隔離性、耐用性](#)。

## 主動-主動式遷移

一種資料庫遷移方法，其中來源和目標資料庫保持同步 (透過使用雙向複寫工具或雙重寫入操作)，且兩個資料庫都在遷移期間處理來自連接應用程式的交易。此方法支援小型、受控制批次的遷移，而不需要一次性切換。它比[主動-被動遷移](#)更具彈性，但需要更多的工作。

## 主動-被動式遷移

一種資料庫遷移方法，其中來源和目標資料庫保持同步，但只有來源資料庫處理來自連接應用程式的交易，同時將資料複寫至目標資料庫。目標資料庫在遷移期間不接受任何交易。

## 聚合函數

在一組資料列上運作，並計算群組的單一傳回值的 SQL 函數。彙總函式的範例包括SUM和MAX。

## AI

請參閱[人工智慧](#)。

## 艾奧運

請參閱[人工智慧作業](#)。

## 匿名化

永久刪除資料集中個人資訊的程序。匿名化可以幫助保護個人隱私。匿名資料不再被視為個人資料。

## 反模式

一種經常使用的解決方案，用於解決方案的生產力適得其反，效果不佳或效果低於替代方案。

## 應用控制

一種安全性方法，只允許使用核准的應用程式，以協助保護系統免受惡意軟體的攻擊。

## 應用程式組合

有關組織使用的每個應用程式的詳細資訊的集合，包括建置和維護應用程式的成本及其商業價值。此資訊是[產品組合探索和分析程序](#)的關鍵，有助於識別要遷移、現代化和優化的應用程式並排定其優先順序。

## 人工智慧 (AI)

電腦科學領域，致力於使用運算技術來執行通常與人類相關的認知功能，例如學習、解決問題和識別模式。如需詳細資訊，請參閱[什麼是人工智慧？](#)

## 人工智慧操作 (AIOps)

使用機器學習技術解決操作問題、減少操作事件和人工干預以及提高服務品質的程序。如需有關如何在 AWS 遷移策略中使用 AIOps 的詳細資訊，請參閱[操作整合指南](#)。

## 非對稱加密

一種加密演算法，它使用一對金鑰：一個用於加密的公有金鑰和一個用於解密的私有金鑰。您可以共用公有金鑰，因為它不用於解密，但對私有金鑰存取應受到高度限制。

## 原子性、一致性、隔離性、持久性 (ACID)

一組軟體屬性，即使在出現錯誤、電源故障或其他問題的情況下，也能確保資料庫的資料有效性和操作可靠性。

## 屬性型存取控制 (ABAC)

根據使用者屬性 (例如部門、工作職責和團隊名稱) 建立精細許可的實務。如需詳細資訊，請參閱 AWS Identity and Access Management (IAM) 文件中的[適用於 AWS 的 ABAC](#)。

## 授權資料來源

儲存資料主要版本的位置，被認為是最可靠的資訊來源。您可以將授權資料來源中的資料複製到其他位置，以便處理或修改資料，例如匿名化、編輯或將其化名化。

## 可用區域

AWS 區域內一個有所區別的位置，隔離了其他可用區域的故障，並對同區域內的其他可用區域提供低成本、低延遲的網路連線。

## AWS 雲端採用架構 (AWS CAF)

AWS 的指導方針和最佳實務架構，可協助組織制定高效且有效的計畫以成功移至雲端。AWS CAF 將指引分為六個焦點區域 (稱為層面)：業務、人員、控管、平台、安全和操作。業務、人員和控管層面著重於業務技能和程序；平台、安全和操作層面著重於技術技能和程序。例如，人員層面針對處理人力資源 (HR)、人員配備功能和人員管理的利害關係人。對於此層面，AWS CAF 為人員發展、培訓和通訊提供指引，以協助組織為成功採用雲端做好準備。如需詳細資訊，請參閱[AWS CAF 網站](#)和[AWS CAF 白皮書](#)。

## AWS Workload Qualification Framework (AWS WQF)

一種評估資料庫遷移工作負載、建議遷移策略並提供工作預估的工具。AWSWQF 隨附於 AWS Schema Conversion Tool (AWS SCT)。它會分析資料庫結構描述和程式碼物件、應用程式程式碼、相依性和效能特性，並提供評估報告。

## B

### BCP

請參閱[業務連續性規劃](#)。

### 行為圖

資源行為的統一互動式檢視，以及一段時間後的互動。您可以將行為圖與 Amazon Detective 搭配使用來檢查失敗的登入嘗試、可疑的 API 呼叫和類似動作。如需詳細資訊，請參閱偵測文件中的[行為圖中的資料](#)。

### 大端序系統

首先儲存最高有效位元組的系統。另請參閱 [「位元順序」](#)。

### 二進制分類

預測二進制結果的過程 (兩個可能的類別之一)。例如，ML 模型可能需要預測諸如「此電子郵件是否是垃圾郵件？」等問題或「產品是書還是汽車？」

### Bloom 篩選條件

一種機率性、記憶體高效的資料結構，用於測試元素是否為集的成員。

### 分支

程式碼儲存庫包含的區域。儲存庫中建立的第一個分支是主要分支。您可以從現有分支建立新分支，然後在新分支中開發功能或修正錯誤。您建立用來建立功能的分支通常稱為功能分支。當準備好發佈功能時，可以將功能分支合併回主要分支。如需詳細資訊，請參閱[關於分支](#) (GitHub 文件)。

### 防碎玻璃訪問

在特殊情況下，並透過核准的程序，使用者可以快速取得他AWS 帳戶們通常沒有存取權限的存取權。如需詳細資訊，請參閱 AWS Well-Architected 指南中的[實作防破玻璃程序](#)指標。

### 棕地策略

環境中的現有基礎設施。對系統架構採用棕地策略時，可以根據目前系統和基礎設施的限制來設計架構。如果正在擴展現有基礎設施，則可能會混合棕地和[綠地](#)策略。

## 緩衝快取

儲存最常存取資料的記憶體區域。

## 業務能力

業務如何創造價值 (例如, 銷售、客戶服務或營銷)。業務能力可驅動微服務架構和開發決策。如需詳細資訊, 請參閱在 [AWS 上執行容器化微服務](#) 白皮書的 [圍繞業務能力進行組織](#) 部分。

## 業務連續性規劃 (BCP)

一種解決破壞性事件 (如大規模遷移) 對營運的潛在影響並使業務能夠快速恢復營運的計畫。

# C

## 咖啡

請參閱 [AWS 雲端採用架構](#)。

## CCoE

請參閱 [雲端卓越中心](#)。

## CDC

請參閱 [變更資料擷取](#)。

## 變更資料擷取 (CDC)

追蹤對資料來源 (例如資料庫表格) 的變更並記錄有關變更的中繼資料的程序。您可以將 CDC 用於各種用途, 例如稽核或複寫目標系統中的變更以保持同步。

## 混沌工程

故意引入故障或破壞性事件來測試系統的彈性。您可以使用 [AWS Fault Injection Service\(AWS FIS\)](#) 執行實驗來 stress 您的AWS工作負載並評估其回應。

## CI/CD

請參閱 [持續整合和持續交付](#)。

## 分類

有助於產生預測的分類程序。用於分類問題的 ML 模型可預測離散值。離散值永遠彼此不同。例如, 模型可能需要評估影像中是否有汽車。



## 用戶端加密

在目標 AWS 服務接收資料之前，在本機對資料進行加密。

## 雲端卓越中心 (CCoE)

一個多學科團隊，可推動整個組織的雲端採用工作，包括開發雲端最佳實務、調動資源、制定遷移時間表以及領導組織進行大規模轉型。如需詳細資訊，請參閱 AWS 雲端企業策略部落格上的 [CCoE 文章](#)。

## 雲端運算

通常用於遠端資料儲存和 IoT 裝置管理的雲端技術。雲計算通常連接到 [邊緣計算](#) 技術。

## 雲端運作模式

在 IT 組織中，這是用來建置、成熟和最佳化一或多個雲端環境的作業模型。如需詳細資訊，請參閱 [建立您的雲端作業模型](#)。

## 採用雲端階段

組織遷移至 AWS 雲端時通常會經歷以下四個階段：

- 專案 – 執行一些與雲端相關的專案以進行概念驗證和學習用途
- 基礎 – 進行基礎投資以擴展雲端採用 (例如，建立登陸區域、定義 CCoE、建立營運模型)
- 遷移 – 遷移個別應用程式
- 重塑 – 優化產品和服務，並在雲端中創新

這些階段由 Stephen Orban 在 AWS 雲端企業策略部落格上的部落格文章 [邁向雲端優先之旅和採用階段](#) 中定義。如需有關其與 AWS 遷移策略如何相關的資訊，請參閱 [遷移準備程度指南](#)。

## CMDB

請參閱 [組態管理資料庫](#)。

## 程式碼儲存庫

透過版本控制程序來儲存及更新原始程式碼和其他資產 (例如文件、範例和指令碼) 的位置。常見的雲儲存庫包括 GitHub 或 AWS CodeCommit。程式碼的每個版本都稱為分支。在微服務結構中，每個儲存庫都專用於單個功能。單一 CI/CD 管道可以使用多個儲存庫。

## 冷快取

一種緩衝快取，它是空的、未填充的，或者包含過時或不相關的資料。這會影響效能，因為資料庫執行個體必須從主記憶體或磁碟讀取，這比從緩衝快取讀取更慢。

## 冷資料

很少存取且通常是歷史資料。查詢此類資料時，通常可以接受慢速查詢。將此資料移至效能較低且成本較低的儲存層或類別可降低成本。

## 電腦視覺

機器使用的 AI 領域，可以準確識別圖像中的人，地點和事物，在人類水平或以上。它通常使用深度學習模型構建，可以自動從單個圖像或一系列圖像中提取，分析，分類和理解有用的信息。

## 組態管理資料庫 (CMDB)

儲存和管理有關資料庫及其 IT 環境的資訊的儲存庫，同時包括硬體和軟體元件及其組態。您通常在遷移的產品組合探索和分析階段使用 CMDB 中的資料。

## 一致性套件

AWS Config 規則和補救措施的集合，您可以將其組合起來以自訂合規和安全檢查。使用 YAML 範本，您可以在 AWS 帳戶和區域中將一致性套件部署為單一實體，或者跨組織部署。如需詳細資訊，請參閱 AWS Config 文件中的[一致性套件](#)。

## 持續整合和持續交付 (CI/CD)

自動化軟體發程序的來源、建置、測試、暫存和生產階段的程序。CI/CD 通常被描述為管道。CI/CD 可協助您將程序自動化、提升生產力、改善程式碼品質以及加快交付速度。如需詳細資訊，請參閱[持續交付的優點](#)。CD 也可表示持續部署。如需詳細資訊，請參閱[持續交付與持續部署](#)。

# D

## 靜態資料

網路中靜止的資料，例如儲存中的資料。

## 資料分類

根據重要性和敏感性來識別和分類網路資料的程序。它是所有網路安全風險管理策略的關鍵組成部分，因為它可以協助您確定適當的資料保護和保留控制。資料分類是 AWS Well-Architected Framework 中安全支柱的一個組成部分。如需詳細資訊，請參閱[資料分類](#)。

## 資料漂移

生產資料與用來訓練 ML 模型的資料之間有意義的變化，或輸入資料隨著時間的推移有意義的變化。資料漂移可降低 ML 模型預測中的整體品質、準確性和公平性。

## 傳輸中的資料

在您的網路中主動移動的資料，例如在網路資源之間移動。

## 資料最小化

僅收集和處理絕對必要的數據的原則。在中執行資料最小化AWS 雲端可降低隱私權風險、成本和分析碳足跡。

## 資料周長

您AWS環境中的一組預防性護欄，可協助確保只有受信任的身分正在存取來自預期網路的受信任資源。若要取得更多資訊，請參閱 [〈在上建置資料周長〉](#) AWS。

## 資料預先處理

將原始資料轉換成 ML 模型可輕鬆剖析的格式。預處理資料可能意味著移除某些欄或列，並解決遺失、不一致或重複的值。

## 數據來源

在整個生命週期中追蹤資料來源和歷史記錄的程序，例如資料的產生、傳輸和儲存方式。

## 資料主體

正在收集和處理資料的個人。

## 資料倉儲

支援商業智慧 (例如分析) 的資料管理系統。資料倉儲通常包含大量歷史資料，通常用於查詢和分析。

## 資料庫定義語言 (DDL)

用於建立或修改資料庫中資料表和物件之結構的陳述式或命令。

## 資料庫處理語言 (DML)

用於修改 (插入、更新和刪除) 資料庫中資訊的陳述式或命令。

## DDL

請參閱[資料庫定義語言](#)。

## 深度整體

結合多個深度學習模型進行預測。可以使用深度整體來獲得更準確的預測或估計預測中的不確定性。

## 深度學習

一個機器學習子領域，它使用多層人工神經網路來識別感興趣的輸入資料與目標變數之間的對應關係。

### defense-in-depth

這是一種資訊安全方法，其中一系列的安全機制和控制項會在整個電腦網路中精心分層，以保護網路和其中資料的機密性、完整性和可用性。在 AWS 上採用此策略時，可以在 AWS Organizations 結構的不同層上新增多個控制，以協助保護資源。例如，一 defense-in-depth 種方法可能會結合多因素驗證、網路分段和加密。

### 委派的管理員

在 AWS Organizations 中，相容的服務可以註冊 AWS 成員帳戶，用於管理組織的帳戶並管理該服務的許可。此帳戶稱為該服務的委派管理員。如需詳細資訊和相容服務清單，請參閱 AWS Organizations 文件中的[可搭配 AWS Organizations 運作的服務](#)。

### 部署

在目標環境中提供應用程式、新功能或程式碼修正的程序。部署涉及在程式碼庫中實作變更，然後在應用程式環境中建置和執行該程式碼庫。

### 開發環境

請參閱[環境](#)。

### 偵測性控制

一種安全控制，用於在事件發生後偵測、記錄和提醒。這些控制是第二道防線，提醒您注意繞過現有預防性控制的安全事件。如需詳細資訊，請參閱在 AWS 上實作安全控制中的[偵測性控制](#)。

### 發展價值流映射

用於識別限制並排定優先順序，對軟體開發生命週期中的速度和品質產生不利影響的程序。DVSM 擴展了最初為精益生產實踐而設計的價值流映射流程。它著重於創造和通過軟件開發過程中移動價值所需的步驟和團隊。

### 數字雙胞胎

真實世界系統的虛擬表現法，例如建築物、工廠、工業設備或生產線。數位雙胞胎支援預測性維護、遠端監控和生產最佳化。

### 維度表

在 [star 結構描述](#) 中，較小的資料表包含事實資料表中定量資料的相關資料屬性。維度表格屬性通常是文字欄位或離散數字，其行為類似於文字。這些屬性通常用於查詢限制、篩選和結果集標籤。

## 災難

防止工作負載或系統在其主要部署位置達成其業務目標的事件。這些事件可能是自然災害、技術故障或人為行為造成的結果，例如意外設定錯誤或惡意軟體攻擊。

### 災難復原 (DR)

您使用的策略和程序，將因災難造成的停機時間和資料遺失降到最低。如需詳細資訊，請參閱 AWS Well-Architected [的架構中的雲端中的工作負載的災難復原](#)[AWS：雲端復原](#)。

### DML

請參閱[資料庫操作語言](#)。

### 領域驅動的設計

一種開發複雜軟體系統的方法，它會將其元件與每個元件所服務的不斷發展的領域或核心業務目標相關聯。Eric Evans 在其著作 *Domain-Driven Design: Tackling Complexity in the Heart of Software* (Boston: Addison-Wesley Professional, 2003) 中介紹了這一概念。如需有關如何將領域驅動的設計與 strangler fig 模式搭配使用的資訊，請參閱[使用容器和 Amazon API Gateway 逐步現代化舊版 Microsoft ASP.NET \(ASMX\) Web 服務](#)。

### 博士

請參閱[災難復原](#)。

### 漂移檢測

追蹤基線組態的偏差。例如，您可以用 AWS CloudFormation 來[偵測系統資源中的漂移](#)，也可以用 AWS Control Tower 來[偵測 landing zone 中可能會影響法規遵循治理要求的變更](#)。

### DVSM

請參閱[開發價值流映射](#)。

## E

### EDA

請參閱[探索性資料分析](#)。

### 邊緣運算

提升 IoT 網路邊緣智慧型裝置運算能力的技術。與[雲計算](#)相比，邊緣計算可以減少通信延遲並縮短響應時間。

## 加密

一種計算過程，將純文本數據（這是人類可讀的）轉換為密文。

### 加密金鑰

由加密演算法產生的隨機位元的加密字串。金鑰長度可能有所不同，每個金鑰的設計都是不可預測且唯一的。

### 端序

位元組在電腦記憶體中的儲存順序。大端序系統首先儲存最高有效位元組。小端序系統首先儲存最低有效位元組。

### 端點

請參閱[服務端點](#)。

### 端點服務

您可以在虛擬私有雲端 (VPC) 中託管以與其他使用者共用的服務。您可以使用 AWS PrivateLink 建立端點服務並向其他 AWS 帳戶或 AWS Identity and Access Management (IAM) 主體授予許可。這些帳戶或主體可以透過建立介面 VPC 端點私下連接至您的端點服務。如需詳細資訊，請參閱 Amazon Virtual Private Cloud (Amazon VPC) 文件中的[建立端點服務](#)。

### 信封加密

使用另一個加密金鑰對某個加密金鑰進行加密的程序。如需詳細資訊，請參閱 AWS Key Management Service (AWS KMS) 文件中的[封套加密](#)。

### 環境

執行中應用程式的執行個體。以下是雲端運算中常見的環境類型：

- 開發環境 – 執行中應用程式的執行個體，只有負責維護應用程式的核心團隊才能使用。開發環境用來測試變更，然後再將開發環境提升到較高的環境。此類型的環境有時稱為測試環境。
- 較低的環境 – 應用程式的所有開發環境，例如用於初始建置和測試的開發環境。
- 生產環境 – 最終使用者可以存取的執行中應用程式的執行個體。在 CI/CD 管道中，生產環境是最後一個部署環境。
- 較高的環境 – 核心開發團隊以外的使用者可存取的所有環境。這可能包括生產環境、生產前環境以及用於使用者接受度測試的環境。

## epic

在敏捷方法中，有助於組織工作並排定工作優先順序的功能類別。epic 提供要求和實作任務的高層級描述。例如，AWS CAF 安全 Epic 包括身分和存取管理、偵測性控制、基礎設施安全、資料保護和事件回應。如需有關 AWS 遷移策略中的 Epic 的詳細資訊，請參閱[計畫實作指南](#)。

## 探索性資料分析 (EDA)

分析資料集以了解其主要特性的過程。您收集或彙總資料，然後執行初步調查以尋找模式、偵測異常並檢查假設。透過計算摘要統計並建立資料可視化來執行 EDA。

# F

## 事實表

[星型架構](#)中的中央表格。它存儲有關業務運營的定量數據。事實資料表通常包含兩種類型的資料欄：包含計量的資料欄，以及包含維度表格外部索引鍵的資料欄。

## 快速失敗

一種使用頻繁和增量測試來減少開發生命週期的理念。這是敏捷方法的關鍵組成部分。

## 故障隔離邊界

在中AWS 雲端，可用區域、AWS 區域控制平面或資料平面等界限，可限制故障的影響，並協助改善工作負載的彈性。如需詳細資訊，請參閱[AWS錯誤隔離邊界](#)。

## 功能分支

請參閱[分支](#)。

## 特徵

用來進行預測的輸入資料。例如，在製造環境中，特徵可能是定期從製造生產線擷取的影像。

## 功能重要性

特徵對於模型的預測有多重要。這通常表示為可以透過各種技術來計算的數值得分，例如 Shapley Additive Explanations (SHAP) 和積分梯度。有關詳情，請參閱[機器學習模型可解釋性：AWS](#)。

## 特徵轉換

優化 ML 程序的資料，包括使用其他來源豐富資料、調整值、或從單一資料欄位擷取多組資訊。這可讓 ML 模型從資料中受益。例如，如果將「2021-05-27 00:15:37」日期劃分為「2021」、「五月」、「週四」和「15」，則可以協助學習演算法學習與不同資料元件相關聯的細微模式。



## FGAC

請參閱[精細的存取控制](#)。

### 精細的存取控制 (FGAC)

使用多個條件來允許或拒絕訪問請求。

### 閃切遷移

一種資料庫移轉方法，透過[變更資料擷取使用連續資料](#)複寫，在最短的時間內移轉資料，而不是使用階段化方法。目標是將停機時間降至最低。

## G

### 地理阻塞

請參閱[地理限制](#)。

### 地理限制 (地理封鎖)

在 Amazon 中 CloudFront，防止特定國家/地區的使用者存取內容分發的選項。您可以使用允許清單或封鎖清單來指定核准和禁止的國家/地區。如需詳細資訊，請參閱 CloudFront 文件[中的限制內容的地理分佈](#)。

### Gitflow 工作流程

這是一種方法，其中較低和較高環境在原始碼儲存庫中使用不同分支。Gitflow 工作流程被認為是遺留的，[基於主幹的工作流程是現代的首選方法](#)。

### 綠地策略

新環境中缺乏現有基礎設施。對系統架構採用綠地策略時，可以選擇所有新技術，而不會限制與現有基礎設施的相容性，也稱為[棕地](#)。如果正在擴展現有基礎設施，則可能會混合棕地和綠地策略。

### 防護機制

有助於跨組織單位 (OU) 來管控資源、政策和合規的高層級規則。預防性防護機制會強制執行政策，以確保符合合規標準。透過使用服務控制政策和 IAM 許可界限來將其實作。偵測性防護機制可偵測政策違規和合規問題，並產生提醒以便修正。它們是通過使用 AWS Config，Amazon AWS Security Hub GuardDuty，AWS Trusted Advisor 亞馬遜檢查 Amazon Inspector 和自定義 AWS Lambda 檢查來實現的。



# H

## 公頃

查看 [高可用性](#)。

## 異質資料庫遷移

將來源資料庫遷移至使用不同資料庫引擎的目標資料庫 (例如, Oracle 至 Amazon Aurora)。異質遷移通常是重新架構工作的一部分, 而轉換結構描述可能是一項複雜任務。 [AWS 提供有助於結構描述轉換的 AWS SCT](#)。

## 高可用性 (HA)

工作負載在遇到挑戰或災難時持續運作的能力, 無需干預。HA 系統的設計可自動容錯移轉、持續提供高品質的效能, 以及處理不同的負載和故障, 並將效能影響降到最低。

## 歷史學家現代化

一種用於現代化和升級操作技術 (OT) 系統的方法, 以更好地滿足製造業的需求。歷史學家是一種類型的數據庫, 用於收集和存儲工廠中的各種來源的數據。

## 異質資料庫遷移

將您的來源資料庫遷移至共用相同資料庫引擎的目標資料庫 (例如, Microsoft SQL Server 至 Amazon RDS for SQL Server)。同質遷移通常是主機轉換或平台轉換工作的一部分。您可以使用原生資料庫公用程式來遷移結構描述。

## 熱數據

經常存取的資料, 例如即時資料或最近的轉譯資料。此資料通常需要高效能的儲存層或類別, 才能提供快速的查詢回應。

## 修補程序

緊急修正生產環境中的關鍵問題。由於其緊迫性, hotfix 通常是在典型的 DevOps 發行工作流程之外進行。

## 超級護理期間

在切換後, 遷移團隊在雲端管理和監控遷移的應用程式以解決任何問題的時段。通常, 此期間的長度為 1-4 天。在超級護理期間結束時, 遷移團隊通常會將應用程式的責任轉移給雲端營運團隊。

## |

## IaC

查看[基礎結構即程式碼](#)。

## 身分型政策

附接至一個或多個 IAM 主體的政策，可在 AWS 雲端環境內部定義其許可。

## 閒置應用程式

90 天期間 CPU 和記憶體平均使用率在 5% 至 20% 之間的應用程式。在遷移專案中，通常會淘汰這些應用程式或將其保留在內部部署。

## IIoT

請參閱[工業物聯網](#)。

## 不可變基礎設施

為生產工作負載部署新基礎結構的模型，而不是更新、修補或修改現有基礎結構。[不可變的基礎架構本質上比可變基礎架構更加一致、可靠且可預測](#)。如需詳細資訊，請參閱 Well-Architected 的架構中的[使用不可變基礎結構AWS構進行部署](#)最佳作法。

## 傳入 (輸入) VPC

AWS 多帳戶架構中的 VPC，可接受、檢查和路由來自應用程式外部的網路連線。[AWS 安全參考架構](#)建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

## 增量遷移

一種切換策略，您可以在其中將應用程式分成小部分遷移，而不是執行單一、完整的切換。例如，您最初可能只將一些微服務或使用者移至新系統。確認所有項目都正常運作之後，您可以逐步移動其他微服務或使用者，直到可以解除委任舊式系統。此策略可降低與大型遷移關聯的風險。

## 基礎設施

應用程式環境中包含的所有資源和資產。

## 基礎設施即程式碼 (IaC)

透過一組組態檔案來佈建和管理應用程式基礎設施的程序。IaC 旨在協助您集中管理基礎設施，標準化資源並快速擴展，以便新環境可重複、可靠且一致。

## 工業物聯網 (IIoT)

在製造業、能源、汽車、醫療保健、生命科學和農業等產業領域使用網際網路連線的感測器和裝置。如需詳細資訊，請參閱[建立工業物聯網 \(IIoT\) 數位轉型策略](#)。

## 檢查 VPC

AWS 多帳戶架構中的集中式 VPC，可管理 VPC (在相同或不同 AWS 區域中)、網際網路和內部部署網路之間的網路流量的檢查。[AWS 安全參考架構](#)建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

## 物聯網 (IoT)

具有內嵌式感測器或處理器的相連實體物體網路，其透過網際網路或本地通訊網路與其他裝置和系統進行通訊。如需詳細資訊，請參閱[什麼是 IoT?](#)

## 可解釋性

機器學習模型的一個特徵，描述了人類能夠理解模型的預測如何依賴於其輸入的程度。如需詳細資訊，請參閱[AWS 的機器學習模型可解釋性](#)。

## IoT

請參閱[物聯網](#)。

## IT 資訊庫 (ITIL)

一組用於交付 IT 服務並使這些服務與業務需求保持一致的最佳實務。ITIL 為 ITSM 提供了基礎。

## IT 服務管理 (ITSM)

與組織的設計、實作、管理和支援 IT 服務關聯的活動。如需有關將雲端操作與 ITSM 工具整合的資訊，請參閱[操作整合指南](#)。

## ITIL

請參閱[IT 資訊庫](#)。

## ITSM

請參閱[IT 服務管理](#)。

## L

### 以標籤為基礎的存取控制 (LBAC)

強制存取控制 (MAC) 的實作，其中每個使用者和資料本身都明確指派一個安全性標籤值。使用者安全性標籤與資料安全性標籤之間的交集決定了使用者可以看到哪些列與欄。

### 登陸區域

登陸區域是一個可擴展且安全的、架構良好的多帳戶 AWS 環境。這是一個起點，您的組織可以從此起點快速啟動和部署工作負載與應用程式，並對其安全和基礎設施環境充滿信心。如需有關登陸區域的詳細資訊，請參閱[設定安全且可擴展的多帳戶 AWS 環境](#)。

### 大型遷移

遷移 300 部或更多伺服器。

### LBAC

請參閱以[標示為基礎的存取控制](#)。

### 最低權限

授予執行任務所需之最低許可的安全最佳實務。如需詳細資訊，請參閱 IAM 文件中的[套用最低權限許可](#)。

### 隨即轉移

見 [7 盧比](#)

### 小端序系統

首先儲存最低有效位元組的系統。另請參閱 [「位元順序」](#)。

### 較低的環境

請參閱[環境](#)。

## M

### 機器學習 (ML)

一種使用演算法和技術進行模式識別和學習的人工智慧。機器學習會進行分析並從記錄的資料 (例如物聯網 (IoT) 資料) 中學習，以根據模式產生統計模型。如需詳細資訊，請參閱[機器學習](#)。

## 主要分支

請參閱[分支](#)。

## 受管理服務

AWS 服務用於AWS操作基礎架構層、作業系統和平台，並且您可以存取端點以儲存和擷取資料。Amazon Simple Storage Service (Amazon S3) 和 Amazon DynamoDB 是受管服務的範例。這些也被稱為抽象的服務。

## MAP

請參閱 [Migration Acceleration Program](#)。

## 機制

一個完整的過程，您可以在其中創建工具，推動工具的採用，然後檢查結果以進行調整。機制是一個循環，它加強和改善自己，因為它運行。如需詳細資訊，請參閱 AWS Well-Architected 的架構中[建置機制](#)。

## 成員帳戶

管理帳戶之外的所有 AWS 帳戶，屬於 AWS Organizations 中組織的一部分。一個帳戶一次只能是一個組織的成員。

## 微服務

一種小型的獨立服務，它可透過定義明確的 API 進行通訊，通常由小型獨立團隊擁有。例如，保險系統可能包含對應至業務能力 (例如銷售或行銷) 或子領域 (例如購買、索賠或分析) 的微服務。微服務的優點包括靈活性、彈性擴展、輕鬆部署、可重複使用的程式碼和適應力。如需詳細資訊，請參閱[使用 AWS 無伺服器服務整合微服務](#)。

## 微服務架構

一種使用獨立元件來建置應用程式的方法，這些元件會以微服務形式執行每個應用程式程序。這些微服務會使用輕量型 API，透過明確定義的介面進行通訊。此架構中的每個微服務都可以進行更新、部署和擴展，以滿足應用程式特定功能的需求。如需詳細資訊，請參閱[在 AWS 上實作微服務](#)。

## Migration Acceleration Program (MAP)

一個提供諮詢支援、培訓和服務的 AWS 計畫，以協助組織為移至雲端建置強大的營運基礎，並協助抵消遷移的初始成本。MAP 包括用於有條不紊地執行舊式遷移的遷移方法以及一組用於自動化和加速常見遷移案例的工具。

## 大規模遷移

將大部分應用程式組合依波次移至雲端的程序，在每個波次中，都會以更快的速度移動更多應用程式。此階段使用從早期階段學到的最佳實務和經驗教訓來實作團隊、工具和流程的遷移工廠，以透過自動化和敏捷交付簡化工作負載的遷移。這是 [AWS 遷移策略](#) 的第三階段。

### 遷移工廠

可透過自動化、敏捷的方法簡化工作負載遷移的跨職能團隊。移轉工廠團隊通常包括營運、業務分析師和擁有者、移轉工程師、開發人員和 DevOps 專業人員。20% 至 50% 之間的企業應用程式組合包含可透過工廠方法優化的重複模式。如需詳細資訊，請參閱此內容集中的[遷移工廠的討論](#)和[雲端遷移工廠指南](#)。

### 遷移中繼資料

有關完成遷移所需的應用程式和伺服器的資訊。每種遷移模式都需要一組不同的遷移中繼資料。遷移中繼資料的範例包括目標子網路、安全群組和 AWS 帳戶。

### 遷移模式

可重複的遷移任務，詳細描述遷移策略、遷移目的地以及所使用的遷移應用程式或服務。範例：使用 AWS Application Migration Service 將遷移重新託管至 Amazon EC2。

### 遷移組合評定 (MPA)

一種線上工具，提供用於驗證遷移至 AWS 雲端的業務案例的資訊。MPA 提供詳細的組合評定 (伺服器適當規模、定價、總體擁有成本比較、遷移成本分析) 以及遷移規劃 (應用程式資料分析和資料收集、應用程式分組、遷移優先順序，以及波次規劃)。 [MPA 工具](#) (需要登入) 免費提供給所有 AWS 顧問和 APN 合作夥伴顧問。

### 遷移準備程度評定 (MRA)

使用 AWS CAF 深入了解組織的雲端準備狀態、識別優缺點並制定動作計畫來彌補已識別差距的程序。如需詳細資訊，請參閱[遷移準備程度指南](#)。MRA 是 [AWS 遷移策略](#) 的第一階段。

### 遷移策略

用於將工作負載遷移至 AWS 雲端的方法。如需詳細資訊，請參閱本詞彙表中的 [7 Rs](#) 項目，並參閱[動員您的組織以加速大規模移轉](#)。

### ML

請參閱[機器學習](#)。

### MPA

請參閱[移轉組合評估](#)。

## 現代化

將過時的 (舊版或單一) 應用程式及其基礎架構轉換為雲端中靈活、富有彈性且高度可用的系統，以降低成本、提高效率並充分利用創新。如需詳細資訊，請參閱 [AWS 雲端中應用程式現代化策略](#)。

### 現代化準備程度評定

這項評估可協助判斷組織應用程式的現代化準備程度；識別優點、風險和相依性；並確定組織能夠在多大程度上支援這些應用程式的未來狀態。評定的結果就是目標架構的藍圖、詳細說明現代化程序的開發階段和里程碑的路線圖、以及解決已發現的差距之行動計畫。如需詳細資訊，請參閱 [評估 AWS 雲端中應用程式的現代化準備情況](#)。

### 單一應用程式 (單一)

透過緊密結合的程序作為單一服務執行的應用程式。單一應用程式有幾個缺點。如果一個應用程式功能遇到需求激增，則必須擴展整個架構。當程式碼庫增長時，新增或改進單一應用程式的功能也會變得更加複雜。若要解決這些問題，可以使用微服務架構。如需詳細資訊，請參閱 [將單一體系分解為微服務](#)。

### 多類別分類

一個有助於產生多類別預測的過程 (預測兩個以上的結果之一)。例如，機器學習模型可能會詢問「此產品是書籍、汽車還是電話？」或者「這個客戶對哪種產品類別最感興趣？」

### 可變的基礎

更新和修改生產工作負載現有基礎結構的模型。為了提高一致性，可靠性和可預測性，AWS Well-Architected 框架建議使用 [不可變的基礎結構](#) 作為最佳實踐。

## O

### OAC

請參閱 [原始存取控制](#)。

### OAI

請參閱 [原始存取身分](#)。

### OCM

請參閱 [組織變更管理](#)。

## 離線遷移

一種遷移方法，可在遷移過程中刪除來源工作負載。此方法涉及延長停機時間，通常用於小型非關鍵工作負載。

## OI

請參閱[作業整合](#)。

## OLA

請參閱[作業層級協定](#)。

## 線上遷移

一種遷移方法，無需離線即可將來源工作負載複製到目標系統。連接至工作負載的應用程式可在遷移期間繼續運作。此方法涉及零至最短停機時間，通常用於關鍵的生產工作負載。

## 操作水準協議 (OLA)

一份協議，闡明 IT 職能群組承諾向彼此提供的內容，以支援服務水準協議 (SLA)。

## 操作準備程度檢討 (ORR)

問題和相關最佳做法的檢查清單，可協助您瞭解、評估、預防或減少事件和可能的故障範圍。如需詳細資訊，請參閱 AWS Well-Architected 的架構中的[作業準備檢閱 \(ORR\)](#)。

## 操作整合 (OI)

在雲端中將操作現代化的程序，其中包括準備程度規劃、自動化和整合。如需詳細資訊，請參閱[操作整合指南](#)。

## 組織追蹤

由 AWS CloudTrail 建立的追蹤，它會記錄 AWS Organizations 中某個組織的所有 AWS 帳戶 的所有事件。在屬於組織的每個 AWS 帳戶 中建立此追蹤，它會跟蹤每個帳戶中的活動。如需詳細資訊，請參閱 [CloudTrail 文件中的為組織建立追蹤](#)。

## 組織變更管理 (OCM)

用於從人員、文化和領導力層面管理重大、顛覆性業務轉型的架構。OCM 透過加速變更採用、解決過渡問題，以及推動文化和組織變更，協助組織為新系統和策略做好準備，並轉移至新系統和策略。在 AWS 遷移策略中，此架構稱為人員加速，因為雲端採用專案所需的變更速度。如需詳細資訊，請參閱 [OCM 指南](#)。



## 原始存取控制 (OAC)

在中 CloudFront，限制存取權限以保護 Amazon Simple Storage Service (Amazon S3) 內容的增強選項。OAC 支援 AWS 區域中的所有 S3 儲存貯體、具有 AWS KMS (SS-KMS) 的伺服器端加密以及對 S3 儲存貯體的動態 PUT 和 DELETE 請求。

## 原始存取身分 (OAI)

在中 CloudFront，用於限制存取以保護 Amazon S3 內容的選項。當您使用 OAI 時，CloudFront 會建立 Amazon S3 可用來進行驗證的主體。經驗證的主體只能透過特定散發存取 S3 儲存 CloudFront 貯體中的內容。另請參閱 [OAC](#)，它可提供更精細且增強的存取控制。

## ORR

請參閱 [作業整備檢閱](#)。

## 傳出 (輸出) VPC

AWS 多帳戶架構中的 VPC，它可處理從應用程式內部啟動的網路連線。[AWS 安全參考架構](#)建議您使用傳入、傳出和檢查 VPC 來設定網路帳戶，以保護應用程式與更廣泛的網際網路之間的雙向介面。

## P

### 許可界限

附接至 IAM 主體的 IAM 管理政策，可設定使用者或角色擁有的最大許可。如需詳細資訊，請參閱 IAM 文件中的 [許可界限](#)。

### 個人識別資訊 (PII)

直接查看或與其他相關數據配對時，可用於合理推斷個人身份的信息。PII 的範例包括姓名、地址和聯絡資訊。

### PII

請參閱 [個人識別資訊](#)。

### 手冊

一組預先定義的步驟，可擷取與遷移關聯的工作，例如在雲端中提供核心操作功能。手冊可以採用指令碼、自動化執行手冊或操作現代化環境所需的程序或步驟摘要的形式。

## 政策

可以定義權限 (請參閱以[身分識別為基礎的策略](#))、指定存取條件 (請參閱以[資源為基礎的策略](#)) 或定義組織中所有帳戶的最大權限的物件 AWS Organizations (請參閱[服務控制策略](#))。

## 混合持久性

根據資料存取模式和其他需求，獨立選擇微服務的資料儲存技術。如果您的微服務具有相同的資料儲存技術，則其可能會遇到實作挑戰或效能不佳。如果微服務使用最適合其需求的資料儲存，則可以更輕鬆地實作並達到更好的效能和可擴展性。如需詳細資訊，請參閱[在微服務中啟用資料持久性](#)。

## 組合評定

探索、分析應用程式組合並排定其優先順序以規劃遷移的程序。如需詳細資訊，請參閱[評估遷移準備程度](#)。

## 述詞

傳回 true 或的查詢條件 false，通常位於子 WHERE 句中。

## 謂詞下推

一種資料庫查詢最佳化技術，可在傳輸前篩選查詢中的資料。這樣可減少必須從關聯式資料庫擷取和處理的資料量，並改善查詢效能。

## 預防性控制

旨在防止事件發生的安全控制。這些控制是第一道防線，可協助防止對網路的未經授權存取或不必要變更。如需詳細資訊，請參閱在 AWS 上實作安全控制中的[預防性控制](#)。

## 委託人

AWS 中的實體，可以執行動作和存取資源。此實體通常是 AWS 帳戶、IAM 角色或使用者的根使用者。如需詳細資訊，請參閱 IAM 文件中[角色術語和概念](#)中的主體。

## 隱私設計

一種系統工程方法，在整個工程過程中將隱私權納入考量。

## 私有託管區域

一種容器，它包含有關您希望 Amazon Route 53 如何回應一個或多個 VPC 內的域及其子域之 DNS 查詢的資訊。如需詳細資訊，請參閱 Route 53 文件中的[使用私有託管區域](#)。

## 主動控制

旨在防止部署不合規資源的[安全控制](#)。這些控制項會在資源佈建之前進行掃描。如果資源不符合控制項，則不會佈建該資源。如需詳細資訊，請參閱AWS Control Tower文件中的[控制項參考指南](#)，並參閱實作安全性[控制中的主動控制](#)AWS。

## 生產環境

請參閱[環境](#)。

## 化名化

以預留位置值取代資料集中的個人識別碼的程序。假名化可以幫助保護個人隱私。假名化數據仍被認為是個人數據。

## Q

### 查詢計劃

一系列步驟，如指示，用來存取 SQL 關聯式資料庫系統中的資料。

### 查詢計劃迴歸

在資料庫服務優化工具選擇的計畫比對資料庫環境進行指定的變更之前的計畫不太理想時。這可能因為對統計資料、限制條件、環境設定、查詢參數繫結的變更以及資料庫引擎的更新所導致。

## R

### 拉齊矩陣

請參閱[負責任，負責，諮詢，通知 \( RAC I \)](#)。

### 勒索軟體

一種惡意軟體，旨在阻止對計算機系統或資料的存取，直到付款為止。

### 拉西矩陣

請參閱[負責任，負責，諮詢，通知 \( RAC I \)](#)。

### RCAC

請參閱[列與欄存取控制](#)。

## 僅供讀取複本

用於唯讀用途的資料庫複本。您可以將查詢路由至僅供讀取複本以減少主資料庫的負載。

## 重新建築師

見 [7 盧比](#)

## 復原點目標 (RPO)

自上次資料復原點以來可接受的時間上限。這決定了最後一個恢復點和服務中斷之間可接受的數據丟失。

## 復原時間目標 (RTO)

服務中斷與恢復服務之間的最大可接受延遲。

## 重構

見 [7 盧比](#)

## 區域

地理區域中 AWS 資源的集合。每個 AWS 區域 都是獨立的且獨立於其他區域，以提供容錯能力、穩定性和恢復能力。如需詳細資訊，請參閱 AWS 一般參考 中的[管理 AWS 區域](#)。

## 迴歸

預測數值的 ML 技術。例如，為了解決「這房子會賣什麼價格？」的問題 ML 模型可以使用線性迴歸模型，根據已知的房屋事實 (例如，平方英尺) 來預測房屋的銷售價格。

## 重新主持

見 [7 盧比](#)

## 版本

在部署程序中，它是將變更提升至生產環境的動作。

## 重新定位

見 [7 盧比](#)

## 再平台

見 [7 盧比](#)

## 買回

見 [7 盧比](#)

## 資源型政策

附接至資源的政策，例如 Amazon S3 儲存貯體、端點或加密金鑰。這種類型的政策會指定允許存取哪些主體、支援的動作以及必須滿足的任何其他條件。

## 負責者、當責者、事先諮詢者和事後告知者 (RACI) 矩陣

定義移轉活動和雲端作業所有相關方的角色和職責的矩陣。矩陣名稱衍生自矩陣中定義的責任類型：負責 (R)、負責 (A)、諮詢 (C)，以及通知 (I)。支撐 (S) 類型是可選的。如果您包含支援，則該矩陣稱為 RASCI 矩陣，如果您將其排除，則稱為 RACI 矩陣。

## 回應性控制

一種安全控制，旨在驅動不良事件或偏離安全基準的補救措施。如需詳細資訊，請參閱在 AWS 上實作安全控制中的 [回應性控制](#)。

## 保留

見 [7 盧比](#)

## 退休

見 [7 盧比](#)

## 旋轉

定期更新 [密碼](#) 以使攻擊者更難以存取認證的程序。

## 資料列與資料行存取控制 (RCAC)

使用已定義存取規則的基本、彈性 SQL 運算式。RCAC 由資料列權限和資料行遮罩所組成。

## RPO

請參閱 [復原點目標](#)。

## RTO

請參閱 [復原時間目標](#)。

## 執行手冊

執行特定任務所需的一組手動或自動程序。這些通常是為了簡化重複性操作或錯誤率較高的程序而建置。

# S

## SAML 2.0

許多身份提供者 ( IdPs ) 使用的開放標準。此功能可啟用聯合單一登入 (SSO) ，因此使用者可以登入 AWS Management Console 或呼叫 AWS API 操作，而不必為組織中的每個人都建立一個 IAM 使用者。如需有關以 SAML 2.0 為基礎的聯合詳細資訊，請參閱 IAM 文件中的[關於以 SAML 2.0 為基礎的聯合](#)。

## SCP

請參閱[服務控制策略](#)。

## 秘密

您以加密形式儲存的機密或受限制資訊，例如密碼或使用者認證。AWS Secrets Manager 它由秘密值及其中繼資料組成。密碼值可以是二進位、單一字串或多個字串。如需詳細資訊，請參閱[秘密管理員說明文件](#)中的秘密。

## 安全控制

一種技術或管理防護機制，它可預防、偵測或降低威脅行為者利用安全漏洞的能力。安全性控制有四種主要類型：[預防性](#)、[偵測](#)、[回應式](#)和[主動式](#)。

## 安全強化

減少受攻擊面以使其更能抵抗攻擊的過程。這可能包括一些動作，例如移除不再需要的資源、實作授予最低權限的安全最佳實務、或停用組態檔案中不必要的功能。

## 安全資訊與事件管理 (SIEM) 系統

結合安全資訊管理 (SIM) 和安全事件管理 (SEM) 系統的工具與服務。SIEM 系統會收集、監控和分析來自伺服器、網路、裝置和其他來源的資料，以偵測威脅和安全漏洞，並產生提醒。

## 安全回應自動化

預先定義且程式化的動作，其設計用來自動回應或修復安全性事件。這些自動化作業可做為[偵探](#)或[回應式](#)安全控制項，協助您實作 AWS 安全性最佳實務。自動回應動作的範例包括修改 VPC 安全群組、修補 Amazon EC2 執行個體或輪換登入資料。

## 伺服器端加密

由接收資料的 AWS 服務 對其目的地的資料進行加密。

## 服務控制政策 (SCP)

為 AWS Organizations 中的組織的所有帳戶提供集中控制許可的政策。SCP 會定義防護機制或設定管理員可委派給使用者或角色的動作限制。您可以使用 SCP 作為允許清單或拒絕清單，以指定允許或禁止哪些服務或動作。如需詳細資訊，請參閱 AWS Organizations 文件中的[服務控制政策](#)。

## 服務端點

AWS 服務的進入點 URL。您可以使用端點，透過程式設計方式連接至目標服務。如需詳細資訊，請參閱 AWS 一般參考中的[AWS 服務端點](#)。

## 服務水準協議 (SLA)

一份協議，闡明 IT 團隊承諾向客戶提供的服務，例如服務正常執行時間和效能。

## 服務等級指示器 (SLI)

對服務效能層面的測量，例如錯誤率、可用性或輸送量。

## 服務層級目標 (SLO)

代表服務狀況的目標測量結果，由[服務層次指示器](#)測量。

## 共同責任模式

描述您與 AWS 共同承擔責任以確保雲端安全和合規的模型。AWS 負責雲端的安全，而您負責雲端中的安全。如需詳細資訊，請參閱[共同責任模式](#)。

## 遲

請參閱[安全性資訊和事件管理系統](#)。

## 單一故障點 (SPF)

應用程式的單一重要元件發生故障，可能會中斷系統。

## SLA

請參閱[服務等級協議](#)。

## SLI

請參閱[服務層級指示器](#)。

## SLO

請參閱[服務等級目標](#)。

## split-and-seed 模型

擴展和加速現代化專案的模式。定義新功能和產品版本時，核心團隊會進行拆分以建立新的產品團隊。這有助於擴展組織的能力和服務，提高開發人員生產力，並支援快速創新。如需詳細資訊，請參閱 [AWS 雲端](#)

## 抽

請參閱 [單一故障點](#)。

## 星型綱要

使用一個大型事實資料表來儲存交易或測量資料，並使用一或多個較小的維度表格來儲存資料屬性的資料庫組織結構。這種結構是專為在 [數據倉庫](#) 中使用或用於商業智能目的。

## Strangler Fig 模式

一種現代化單一系統的方法，它會逐步重寫和取代系統功能，直到舊式系統停止使用為止。此模式源自無花果藤，它長成一棵馴化樹並最終戰勝且取代了其宿主。該模式由 [Martin Fowler 引入](#)，作為重寫單一系統時管理風險的方式。如需有關如何套用此模式的範例，請參閱 [使用容器和 Amazon API Gateway 逐步現代化舊版 Microsoft ASP.NET \(ASMX\) Web 服務](#)。

## 子網

您 VPC 中的 IP 地址範圍。子網必須位於單一可用區域。

## 對稱加密

使用相同金鑰來加密及解密資料的加密演算法。

## 合成測試

以模擬使用者互動以偵測潛在問題或監控效能的方式測試系統。您可以使用 [Amazon CloudWatch Synthetics](#) 來創建這些測試。

# T

## 標籤

作為組織 AWS 資源的中繼資料的索引鍵值配對。標籤可協助您管理、識別、組織、搜尋及篩選資源。如需詳細資訊，請參閱 [標記您的 AWS 資源](#)。

## 目標變數

您嘗試在受監督的 ML 中預測的值。這也被稱為結果變數。例如，在製造設定中，目標變數可能是產品瑕疵。



## 任務清單

用於透過執行手冊追蹤進度的工具。任務清單包含執行手冊的概觀以及要完成的一般任務清單。對於每個一般任務，它包括所需的預估時間量、擁有者和進度。

## 測試環境

請參閱[環境](#)。

## 訓練

為 ML 模型提供資料以供學習。訓練資料必須包含正確答案。學習演算法會在訓練資料中尋找將輸入資料屬性映射至目標的模式 (您想要預測的答案)。它會輸出擷取這些模式的 ML 模型。可以使用 ML 模型，來預測您不知道的目標新資料。

## 傳輸閘道

可以用於互連 VPC 和內部部署網路的網路傳輸中樞。如需詳細資訊，請參閱 AWS Transit Gateway 文件中的[傳輸閘道是什麼](#)。

## 主幹型工作流程

這是一種方法，開發人員可在功能分支中本地建置和測試功能，然後將這些變更合併到主要分支中。然後，主要分支會依序建置到開發環境、生產前環境和生產環境中。

## 受信任的存取權

准許您指定的服務在 AWS Organizations 中的組織中以及其帳戶中代表您執行任務。受信任的服務會在需要該角色時，在每個帳戶中建立服務連結角色，以便為您執行管理工作。如需詳細資訊，請參閱 AWS Organizations 文件中的[搭配使用 AWS Organizations 和其他 AWS 服務](#)。

## 調校

變更訓練程序的各個層面，以提高 ML 模型的準確性。例如，可以透過產生標籤集、新增標籤、然後在不同的設定下多次重複這些步驟來訓練 ML 模型，以優化模型。

## 雙比薩團隊

一個小 DevOps 團隊，你可以餵兩個比薩餅。雙披薩團隊規模可確保軟體開發中的最佳協作。

## U

### 不確定性

這是一個概念，指的是不精確、不完整或未知的資訊，其可能會破壞預測性 ML 模型的可靠性。有兩種類型的不確定性：認知不確定性是由有限的、不完整的資料引起的，而隨機不確定性是由資料中固有的噪聲和隨機性引起的。如需詳細資訊，請參閱[量化深度學習系統的不確定性](#)指南。

### 無差別的任務

也稱為繁重工作，是創建和操作應用程序所必需的工作，但不能為最終用戶提供直接價值或提供競爭優勢。無差異化作業的範例包括採購、維護和容量規劃。

### 較高的環境

請參閱[環境](#)。

## V

### 清空

一種資料庫維護操作，涉及增量更新後的清理工作，以回收儲存並提升效能。

### 版本控制

追蹤變更的程序和工具，例如儲存庫中原始程式碼的變更。

### VPC 對等互連

兩個 VPC 之間的連線，可讓您使用私有 IP 地址路由流量。如需詳細資訊，請參閱 Amazon VPC 文件中的[什麼是 VPC 對等互連](#)。

### 漏洞

會危及系統安全性的軟體或硬體瑕疵。

## W

### 暖快取

包含經常存取的目前相關資料的緩衝快取。資料庫執行個體可以從緩衝快取讀取，這比從主記憶體或磁碟讀取更快。

## 溫暖的數據

不常存取的資料。查詢此類資料時，通常可以接受中度緩慢的查詢。

## 視窗功能

一種 SQL 函數，可對以某種方式與當前記錄相關的一組行執行計算。視窗函數對於處理工作非常有用，例如計算移動平均值或根據目前列的相對位置存取列的值。

## 工作負載

提供商業價值的資源和程式碼集合，例如面向客戶的應用程式或後端流程。

## 工作串流

遷移專案中負責一組特定任務的功能群組。每個工作串流都是獨立的，但支援專案中的其他工作串流。例如，組合工作串流負責排定應用程式、波次規劃和收集遷移中繼資料的優先順序。組合工作串流將這些資產交付至遷移工作串流，然後再遷移伺服器 and 應用程式。

## 蠕蟲

看到 [寫一次，多讀](#)。

## WQF

請參閱 [AWS 工作負載資格架構](#)。

## 寫一次，多讀 ( WORM )

一種儲存模型，可單次寫入資料並防止資料遭到刪除或修改。授權用戶可以根據需要多次讀取數據，但無法更改數據。這種數據存儲基礎設施被認為是 [不可變](#) 的。

## Z

### 零日漏洞

一種利用 [零時差漏洞](#) 的攻擊，通常是惡意軟件。

### 零時差漏洞

生產系統中未緩解的瑕疵或弱點。威脅參與者可以利用這種類型的漏洞攻擊系統。由於攻擊，開發人員經常意識到該漏洞。

### 殭屍應用程式

CPU 和記憶體平均使用率低於 5% 的應用程式。在遷移專案中，通常會淘汰這些應用程式。

本文為英文版的機器翻譯版本，如內容有任何歧義或不一致之處，概以英文版為準。