aws

開發人員指南

Amazon Keyspaces (適用於 Apache Cassandra)

Amazon Keyspaces (適用於 Apache Cassandra): 開發人員指南

Table of Contents

什麼是 Amazon Keyspaces（阿帕奇卡桑德拉）？

Amazon Keyspaces（對於阿帕奇卡桑德拉）是一個可擴展的，高可用性和託管的 Apache 卡桑德拉兼容數據庫服務。使用 Amazon Keyspaces space，您不必佈建、修補或管理伺服器，也不必安裝、維護或操作軟體。

Amazon Keyspaces 是無伺服器的，因此您只需為使用的資源付費，服務會自動上下擴展表格以回應應用程式流量。您可以建置每秒處理數千個請求的應用程式，並具有幾乎無限制的輸送量和儲存體。

> **Note**
>
> 阿帕奇卡桑德拉是一個開源的，寬列的數據存儲，旨在處理大量的數據。如需詳細資訊，請參閱阿帕奇卡桑德拉。

Amazon Keyspaces 可讓您輕鬆移轉、執行和擴展. AWS 雲端只要在 AWS 管理主控台按幾下或幾行程式碼，就可以在 Amazon Keyspaces 中建立金鑰空間和表格，而無需部署任何基礎設施或安裝軟體。

使用 Amazon Keyspaces，您可以使用目前使用的相同 Cassandra 應 AWS 用程式程式碼和開發人員工具來執行現有的 Cassandra 工作負載。

如需可用 AWS 區域 和端點的清單，請參閱 Amazon Keyspaces 的服務端點。

我們建議您先閱讀以下章節：

主題

- Amazon Keyspaces：它是如何工作的
- Amazon Keyspaces 使用案例
- 什麼是卡桑德拉查詢語言（CQL）？

Amazon Keyspaces：它是如何工作的

Amazon Keyspaces 刪除管理卡桑德拉的管理開銷。要理解為什麼，從 Cassandra 架構開始，然後將其與 Amazon Keyspaces 進行比較是有幫助的。

主題

- 高級架構：阿帕奇卡桑德拉與 Amazon Keyspaces

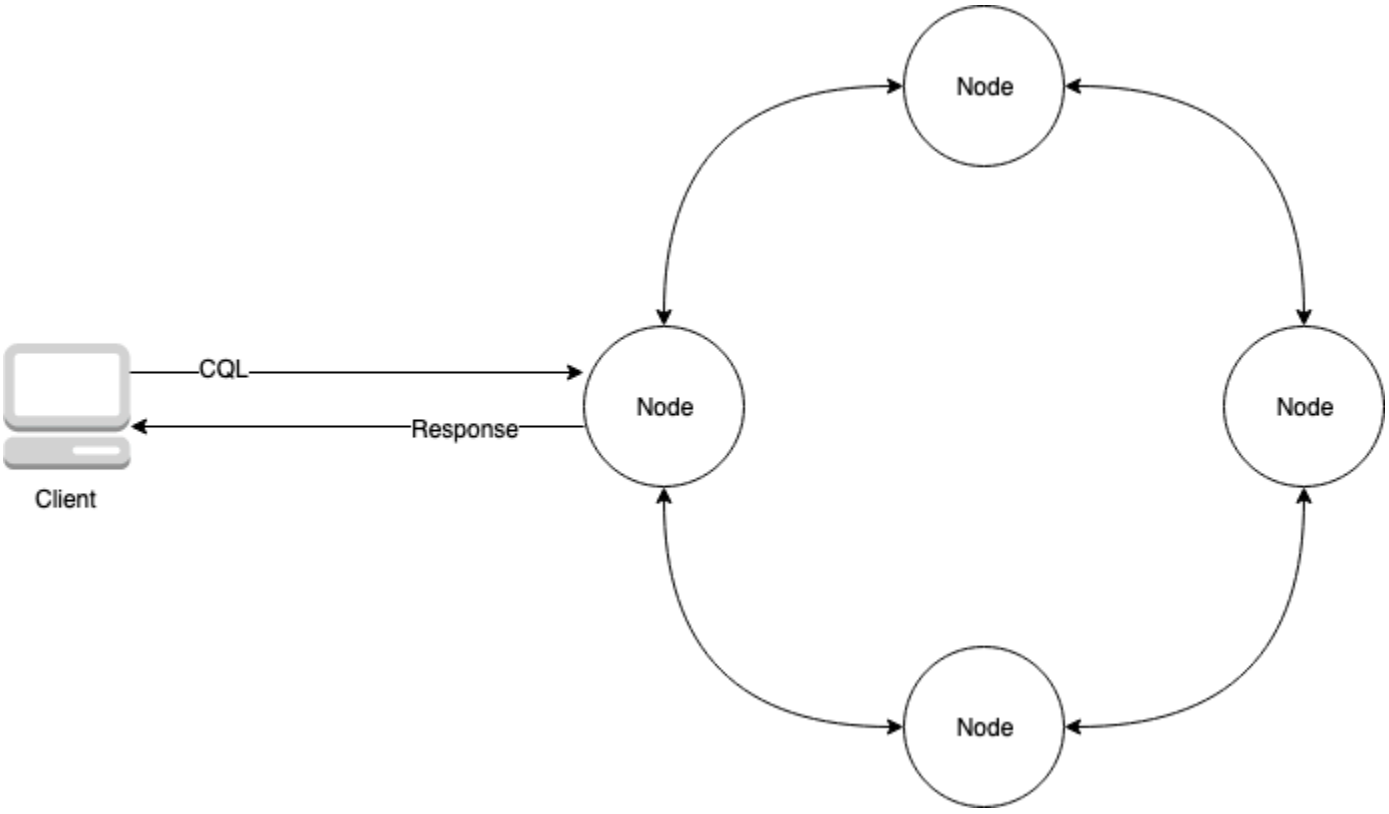
- 卡桑德拉數據模型
- 從應用程式存取 Amazon Keyspaces

高級架構：阿帕奇卡桑德拉與 Amazon Keyspaces

傳統的 Apache 卡桑德拉部署在由一個或多個節點組成的集群。您負責管理每個節點，並在叢集擴展時新增和移除節點。

客戶端程序通過連接到節點之一並發出卡桑德拉查詢語言（CQL）語句訪問卡桑德拉。CQL 類似於 SQL，在關係數據庫中使用的流行語言。即使卡桑德拉是不是一個關係數據庫，CQL 提供了查詢和操作卡桑德拉數據熟悉的接口。

下圖顯示了一個簡單的 Apache 卡桑德拉集群，由四個節點。



生產 Cassandra 部署可能包括數百個節點，在一個或多個物理數據中心的數百台物理計算機上運行。除了安裝、維護和操作軟體之外，還需要佈建、修補和管理伺服器的應用程式開發人員，這可能會造成操作負擔。

使用 Amazon Keyspaces (適用於 Apache Cassandra)，您不需要佈建、修補或管理伺服器，因此您可以專注於建置更好的應用程式。Amazon Keyspaces 為讀取和寫入提供兩種輸送量容量模式：隨需和

佈建。您可以選擇表格的輸送量容量模式，根據工作負載的可預測性和變化性，將讀取和寫入的價格最佳化。

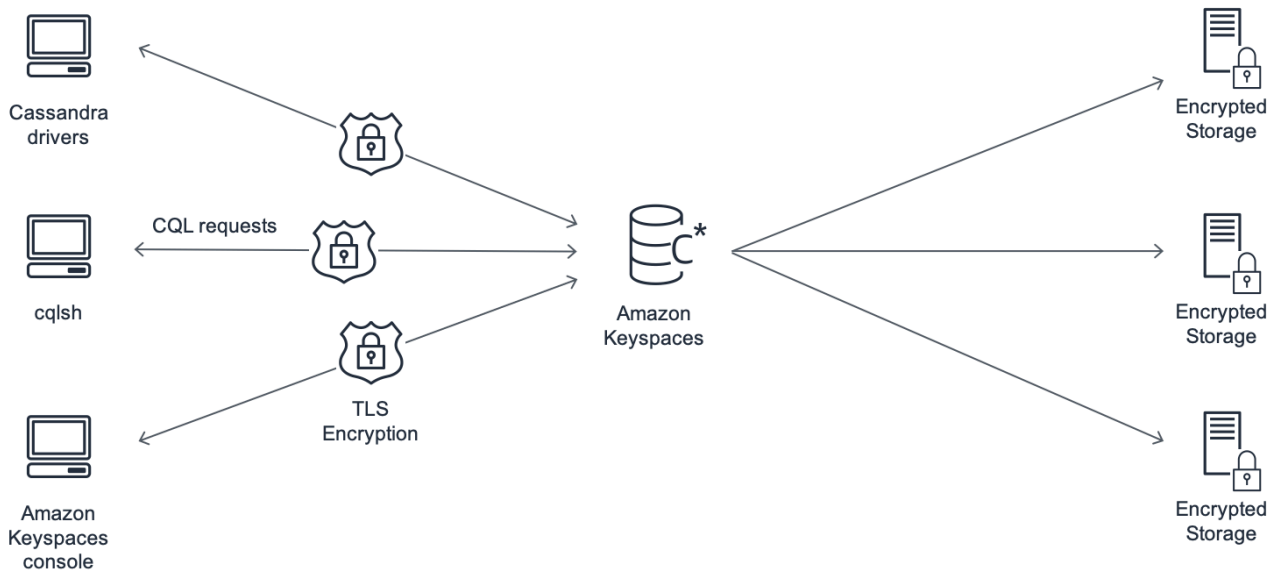
使用隨選模式時，您只需支付應用程式實際執行的讀取和寫入費用。您不需要事先指定表格的輸送量容量。Amazon Keyspaces 在上升或下降時幾乎可以立即容納您的應用程式流量，使其成為流量不可預測的應用程式的理想選擇。

如果您擁有可預測的應用程式流量，並且可以預先預測表格的容量需求，佈建容量模式可協助您將輸送量價格最佳化。使用佈建的容量模式，您可以指定預期應用程式執行的每秒讀取和寫入次數。您可以啟用自動擴展來自動增加和減少表格的佈建容量。

當您深入瞭解工作負載的流量模式時，您可以每天變更一次資料表的容量模式，或者如果您預期會有大量流量 (例如預期發生的重大事件將會導致大量資料表流量)。如需有關讀取和寫入容量佈建的詳細資訊，請參閱the section called “讀/寫容量模式”。

Amazon Keyspaces (適用於 Apache Cassandra) 會在多個可用區域中儲存三份資料副本，以確保持久性和高可用性。此外，您還可以從資料中心和網路架構中獲益，該架構專為滿足對安全性最敏感的組織的需求而打造。當您建立新的 Amazon Keyspaces 表格時，靜態加密會自動啟用，且所有用戶端連線都需要傳輸層安全性 (TLS)。其他 AWS 安全功能包括監控和虛擬私有雲端 (VPC) 端點。AWS Identity and Access Management如需所有可用安全性功能的概觀，請參閱安全性。

下圖顯示了 Amazon Keyspaces 的體系結構。



用戶端程式透過連線到預先確定的端點 (主機名稱和連接埠號碼) 並發出 CQL 陳述式來存取 Amazon Keyspaces。如需可用端點的清單，請參閱the section called “服務端點”。

卡桑德拉數據模型

如何為商業案例建立資料模型，對於從 Amazon Keyspace 達到最佳效能至關重要。不良的資料模型可能會大幅降低效能。

即使 CQL 看起來類似於 SQL，卡桑德拉和關係數據庫的後端是非常不同的，必須以不同的方式處理。以下是一些需要考慮的更重要的問題：

儲存

您可以在表中可視化您的 Cassandra 數據，每行代表一條記錄，每列都是該記錄中的字段。

資料表設計：先查詢

CQL 中沒有 `JOIN` s。因此，您應該根據資料的形式設計資料表格，以及在業務使用案例中存取資料的方式。這可能會導致重複資料的反正規化。您應該專門針對特定的存取模式設計每個表格。

分區

您的數據存儲在磁盤上的分區中。資料儲存在分割區的數目，以及在分割區之間的分佈方式取決於您的分割區索引鍵。如何定義分區索引鍵可能會對查詢的效能產生重大影響。如需最佳實務做法，請參閱「the section called “分割區索引鍵設計”」。

主索引鍵

在卡桑德拉，數據被存儲為鍵-值對。為此，每個 Cassandra 表必須有一個主鍵，這是表中每一行的關鍵。主索引鍵是必要的資料分割索引鍵和選擇性叢集資料行的複合。包含主鍵的數據在表中的所有記錄中必須是唯一的。

- 分割索引鍵 — 需要主索引鍵的分割區索引鍵部分，並決定資料儲存在叢集中的哪個分割區。分區鍵可以是單個列，也可以是由兩個或多個列組成的複合值。如果單一資料行分割索引鍵會導致單一分割區或極少數分割區擁有大部分資料，並因此承載大部分磁碟 I/O 作業，您可以使用複合分割區索引鍵。
- 叢集資料欄 — 主索引鍵的選用叢集資料行部分決定了資料在每個分割區內的叢集和排序方式。如果您在主索引鍵中包含叢集資料行，叢集資料行可以有一或多個資料行。如果叢集資料行中有多個資料行，排序順序會由叢集資料行中的列出順序 (從左到右) 決定。

如需有關 NoSQL 設計和 Amazon Keyspaces 的詳細資訊，請參閱。the section called “NoSQL 設計”如需 Amazon Keyspaces 和資料建模的詳細資訊，請參閱the section called “建立資料模型”。

從應用程式存取 Amazon Keyspaces

Amazon Keyspaces（對於阿帕奇卡桑德拉）實現了 Apache 卡桑德拉查詢語言（CQL）API，所以你可以使用你已經使用 CQL 和卡桑德拉驅動程序。更新應用程式就像將 Cassandra 驅動程式或`cqlsh`組態更新為指向 Amazon Keyspaces 服務端點一樣簡單。如需所需認證的詳細資訊，請參閱the section called “身分 AWS 驗證的 IAM 登入資”。

> **Note**
>
> 為了協助您開始使用，您可以在 Amazon Keyspaces 程式 end-to-end 碼範例儲存庫中使用各種 Cassandra 用戶端驅動程式，找到連接至 Amazon Keyspaces 的程式碼範例程式碼範例。GitHub

考慮下面的 Python 程序，它連接到一個卡桑德拉集群和查詢表。

```
from cassandra.cluster import Cluster
#TLS/SSL configuration goes here

ksp = 'MyKeyspace'
tbl = 'WeatherData'

cluster = Cluster(['NNN.NNN.NNN.NNN'], port=NNNN)
session = cluster.connect(ksp)

session.execute('USE ' + ksp)

rows = session.execute('SELECT * FROM ' +  tbl)
for row in rows:
    print(row)
```

要對 Amazon Keyspaces 運行相同的程序，您需要：

- 新增叢集端點和連接埠：例如，可以將主機取代為服務端點，例如`cassandra.us-east-2.amazonaws.com`，連接埠號碼為：`9142`。
- 新增 TLS/SSL 組態：如需有關新增 TLS/SSL 組態以連接到 Amazon Keyspaces，使用卡桑德拉用戶端 Python 驅動程式的詳細資訊，請參閱。使用卡桑德拉 Python 客戶端驅動程序以編程方式訪問 Amazon Keyspaces

Amazon Keyspaces 使用案例

以下是您可以使用 Amazon Keyspaces 的一些方法：

- 建置需要低延遲的應用程式 — 針對需要延 single-digit-millisecond遲的應用程式高速處理資料，例如工業設備維護、交易監控、車隊管理和路線最佳化。
- 使用開放原始碼技術建置應用程式 — AWS 使用開放原始碼卡桑德拉 API 和可用於多種程式設計語言的驅動程式，例如 Java、Python、Ruby、Microsoft NET、Node.js、PHP、C++、Perl 和 Go 來建置應用程式。如需程式碼範例，請參閱「資料庫和工具」。
- 將您的 Cassandra 工作負載移至雲端 — 自行管理 Cassandra 資料表既耗時又昂貴。使用 Amazon Keyspaces，您可以在中設定、保護和擴展 Cassandra 表， AWS 雲端 而無需管理基礎設施。如需詳細資訊，請參閱 無伺服器資源管理。

什麼是卡桑德拉查詢語言（CQL）？

卡桑德拉查詢語言（CQL）是與阿帕奇卡桑德拉通信的主要語言。Amazon Keyspaces（對於阿帕奇卡桑德拉）與 CQL 3.x 的 API 兼容（向後兼容版本 2.x）。

若要執行 CQL 查詢，您可以執行下列其中一項作業：

- 使用上的 CQL 編輯器。 AWS Management Console
- 使用 AWS CloudShell 和 cq lsh 擴展。
- 在`cqlsh`用戶端上執行它們。
- 使用 Apache 2.0 許可的卡桑德拉客戶端驅動程序以編程方式運行它們。

此外，您 Amazon Keyspaces 用 AWS SDK 和. AWS Command Line Interface

如需使用這些方法存取 Amazon Keyspaces 的詳細資訊，請參閱訪問 Amazon Keyspaces（阿帕奇卡桑德拉）。

如需 CQL 的詳細資訊，請參閱Amazon Keyspaces 的 CQL 語言參考（阿帕奇卡桑德拉）。

如何 Amazon Keyspaces（阿帕奇卡桑德拉）比較阿帕奇卡桑德拉？

若要建立與 Amazon Keyspaces 的連線，您可以在 Amazon 虛擬私有雲端中使用公共AWS 服務端點或使用界面 VPC 端點 (AWS PrivateLink) 來使用私有端點。視使用的端點而定，Amazon Keyspaces 可以透過下列其中一種方式向用戶端顯示。

AWS 服務端點連線

這是透過任何公用端點建立的連線。在這種情況下，Amazon Keyspaces 顯示為九節點阿帕奇卡桑德拉 3.11.2 集群到客戶端。

介面 VPC 端點連線

這是使用介面 VPC 端點建立的私人連線。在這種情況下，Amazon Keyspaces 顯示為三節點阿帕奇卡桑德拉 3.11.2 集群到客戶端。

Amazon Keyspaces 與連線類型和用戶端可見的節點數量無關，提供幾乎無限的輸送量和儲存。為此，Amazon Keyspaces space 會將節點對應到負載平衡器，將查詢路由到眾多基礎儲存分區之一。如需連線的相關資訊，請參閱the section called “他們如何工作”。

Amazon Keyspaces 將數據存儲在分區中。分割區是資料表的儲存空間配置，由固態硬碟 (SSD) 支援。Amazon Keyspaces 會在多個可用區域自動複寫您的資料，以提高耐 AWS 區域 用性和高可用性。隨著輸送量或儲存需求的增長，Amazon Keyspaces 會為您處理分割區管理，並自動佈建所需的其他分割區。如需詳細資訊，請參閱 the section called “儲存”。

Amazon Keyspaces 支援所有常用的 Cassandra 資料平面作業，例如建立金鑰空間和表格、讀取資料和寫入資料。Amazon Keyspaces 是無伺服器的，因此您不必佈建、修補或管理伺服器。您也不必安裝、維護或操作軟體。因此，在 Amazon Keyspaces 中，您不需要使用 Cassandra 控制平面 API 操作來管理叢集和節點設定。

Amazon Keyspaces 會自動設定複寫係數和一致性等設定，為您提供高可用性、耐用性和效能。single-digit-millisecond 為了獲得更多彈性和低延遲的本機讀取，Amazon Keyspaces 提供多區域複寫。

主題

- 功能差異：Amazon Keyspaces 與阿帕奇卡桑德拉

- Amazon Keyspaces 中支持的卡桑德拉 API，操作，函數和數據類型
- 亞馬遜密鑰空間中支持的 Apache 卡桑德拉一致性級別

功能差異：Amazon Keyspaces 與阿帕奇卡桑德拉

以下是 Amazon Keyspaces 間和 Apache 卡桑德拉之間的功能差異。

主題

- 阿帕奇卡桑德拉 API，操作和數據類型
- 異步創建和刪除密鑰空間和表
- 身份驗證和授權
- 批次
- 叢集組態
- 連線
- IN關鍵字
- CQL 查詢輸送量調整
- FROZEN集合
- 輕量型交易
- 負載平衡
- 分頁
- 磁碟分割程式
- 準備好陳述
- 範圍刪除
- 系統表
- 時間戳記

阿帕奇卡桑德拉 API，操作和數據類型

Amazon Keyspaces 支援所有常用的 Cassandra 資料平面作業，例如建立金鑰空間和表格、讀取資料和寫入資料。若要查看目前支援的項目，請參閱Amazon Keyspaces 中支持的卡桑德拉 API，操作，函數和數據類型。

異步創建和刪除密鑰空間和表

Amazon Keyspaces 會以非同步方式執行資料定義語言 (DDL) 作業，例如建立和刪除金鑰空間和表格。若要瞭解如何監視資源的建立狀態，請參閱the section called “創建密鑰空間”和the section called “建立資料表”。如需 CQL 語言參考中的 DDL 陳述式清單，請參閱。the section called “DDL 陳述式”

身份驗證和授權

Amazon Keyspaces（對於阿帕奇卡桑德拉）使用 AWS Identity and Access Management （IAM）的用戶身份驗證和授權，並支持等效的授權政策作為 Apache 卡桑德拉。因此，Amazon Keyspaces 不支持 Apache 卡桑德拉的安全配置命令。

批次

Amazon Keyspaces 支援未記錄的批次命令，批次中最多可提供 30 個命令。批次中只允許使用無條件INSERTUPDATE、或DELETE指令。不支援記錄的批次。

叢集組態

Amazon Keyspaces 是無伺服器的，因此沒有要設定的叢集、主機或 Java 虛擬機器 (JVM)。Cassandra 的壓縮，壓縮，緩存，垃圾收集和綻放過濾的設置不適用於 Amazon Keyspaces，如果指定將被忽略。

連線

您可以使用現有的 Cassandra 驅動程序與 Amazon Keyspaces 進行通信，但您需要以不同的方式配置驅動程序。Amazon Keyspaces 每秒每個 TCP 連線最多支援 3,000 個 CQL 查詢，但驅動程式可建立的連線數目沒有限制。

大多數開源卡桑德拉驅動程序建立了一個連接池卡桑德拉，並通過該連接池進行負載平衡查詢。Amazon Keyspaces 會向驅動程式公開 9 個對等 IP 位址，而大多數驅動程式的預設行為是建立與每個對等 IP 位址的單一連線。因此，使用預設設定之驅動程式的最大 CQL 查詢輸送量為每秒 27,000 個 CQL 查詢。

若要增加此數目，我們建議您增加驅動程式在其連線集區中維護的每個 IP 位址的連線數目。例如，將每個 IP 位址的最大連線數設定為 2，將驅動程式的最大輸送量加倍至每秒 54,000 個 CQL 查詢。

最佳做法是，我們建議將驅動程式設定為每個連線每秒使用 500 個 CQL 查詢，以允許額外負荷並改善散佈。在這個案例中，規劃每秒 18,000 個 CQL 查詢需要 36 個連線。針對 9 個端點的 4 個連線設定

驅動程式，可提供 36 個連線，每秒執行 500 個要求。如需有關連線最佳做法的更多資訊，請參閱the section called “連線”。

使用 VPC 端點連線時，可用的端點可能較少。這表示您必須增加驅動程式組態中的連線數目。如需 VPC 連線最佳做法的詳細資訊，請參閱the section called “VPC 端連線”。

IN關鍵字

Amazon Keyspaces 支持SELECT語句中的IN關鍵字。 IN與不支UPDATE援DELETE。在SELECT陳述式中使用IN關鍵字時，會依照索引鍵在陳述SELECT式中呈現的順序傳回查詢結果。在卡桑德拉，結果按字母順序排序。

使用時 ORDER BY，不支持禁用分頁的完全重新排序，並且結果在頁面中排序。IN關鍵字不支援切片查詢。 TOKENSIN關鍵字不支援。Amazon Keyspaces 會透過建立子查詢來處理IN關鍵字的查詢。每個子查詢都會計為每秒 TCP 連線 3,000 CQL 查詢的連線。如需詳細資訊，請參閱 the section called “INSELECT聲明”。

CQL 查詢輸送量調整

Amazon Keyspaces 每秒每個 TCP 連線最多支援 3,000 個 CQL 查詢，但驅動程式可建立的連線數目沒有限制。

大多數開源卡桑德拉驅動程序建立了一個連接池卡桑德拉，並通過該連接池進行負載平衡查詢。Amazon Keyspaces 會向驅動程式公開 9 個對等 IP 位址，而大多數驅動程式的預設行為是建立與每個對等 IP 位址的單一連線。因此，使用預設設定的驅動程式的最大 CQL 查詢輸送量將是每秒 27,000 個 CQL 查詢。

若要增加此數目，我們建議您增加驅動程式在其連線集區中維護的每個 IP 位址的連線數目。例如，將每個 IP 位址的最大連線數設定為 2 會將驅動程式的最大輸送量加倍至每秒 54,000 個 CQL 查詢。

如需有關連線最佳做法的更多資訊，請參閱the section called “連線”。

使用 VPC 端點連線時，可用端點較少。這表示您必須增加驅動程式組態中的連線數目。如需 VPC 端點連線最佳做法的詳細資訊，請參閱the section called “VPC 端連線”。

FROZEN集合

Cassandra 中的FROZEN關鍵字序列化集合數據類型的多個組件成一個單一的不可變值，該值被視為一個。BLOB INSERT和UPDATE陳述式會覆寫整個集合。

預設情況下，Amazon Keyspaces 最多支援五個層級的凍結集合。如需詳細資訊，請參閱 the section called “Amazon Keyspaces 服務配額”。

Amazon Keyspaces 不支援在條件或陳述式中使用整個凍結集合的不等UPDATE式SELECT比較。集合和凍結集合的行為在 Amazon Keyspaces 中是相同的。

當您使用具有用戶端時間戳記的凍結集合時，如果寫入作業的時間戳記與未過期或標記的現有資料行的時間戳記相同，Amazon Keyspace 不會執行比較。相反，它可以讓服務器確定最新的作家，最新的作家獲勝。

如需凍結集合的詳細資訊，請參閱the section called “集合類型”。

輕量型交易

Amazon Keyspaces（對於 Apache 卡桑德拉）完全支持比較和設置功能，和DELETE命令INSERTUPDATE，這是所謂的輕量級交易（LWT）在 Apache 卡桑德拉。作為無服務器產品，Amazon Keyspaces（適用於 Apache Cassandra）在任何規模下提供一致的性能，包括輕量級交易。使用 Amazon Keyspaces 時，使用輕量型交易不會造成效能損失。

負載平衡

`system.peers`表格項目對應於 Amazon Keyspaces 負載平衡器。為了獲得最佳結果，我們建議您使用循環配置資源負載平衡政策，並調整每個 IP 的連線數目，以符合應用程式的需求。

分頁

Amazon Keyspaces 會根據其讀取來處理請求的資料列數，而不是結果集中傳回的資料列數來分頁結果。因此，某些頁面所包含的列數可能比您在篩選查詢的 PAGE SIZE 中指定的列少。此外，Amazon Keyspaces 會在讀取 1 MB 的資料後自動分頁結果，為客戶提供一致的 10 毫秒讀取效能。如需詳細資訊，請參閱 the section called “分頁結果”。

磁碟分割程式

Amazon Keyspaces 中的默認分區程序是卡桑德拉兼容。`Murmur3Partitioner`此外，您可以選擇使用 Amazon Keyspaces `DefaultPartitioner` 或卡桑德拉`RandomPartitioner`兼容。

使用 Amazon Keyspaces，您可以安全地更改帳戶的分區程序，而無需重新加載 Amazon 密 Keyspaces 數據。設定變更完成後 (大約需要 10 分鐘)，用戶端會在下次連線時自動看到新的磁碟分割程式設定。如需詳細資訊，請參閱 the section called “使用磁碟分割程式”。

準備好陳述

Amazon Keyspaces 支援使用準備好的陳述式進行資料操作語言 (DML) 操作，例如讀取和寫入資料。Amazon Keyspaces 目前不支援針對資料定義語言 (DDL) 作業 (例如建立資料表和金鑰空間) 使用準備好的陳述式。DDL 操作必須在準備好的語句之外運行。

範圍刪除

Amazon Keyspaces 支持刪除範圍內的行。範圍是分區內連續的一組行。您可以使用 WHERE 子句在 DELETE 作業中指定範圍。您可以將範圍指定為整個分割區。

此外，您可以使用關聯式運算子 (例如 '>'、'<')，或包括分割索引鍵並省略一或多個叢集資料行，將範圍指定為分割區內連續資料列的子集。使用 Amazon Keyspaces，您可以在單一作業中刪除範圍內多達 1,000 個資料列。此外，範圍刪除是原子的，但不是孤立的。

系統表

Amazon Keyspaces 填充由 Apache 2.0 開源卡桑德拉驅動程序所需的系統表。用戶端可見的系統資料表包含已驗證使用者專屬的資訊。系統表由 Amazon Keyspaces 完全控制，並且是唯讀的。

系統資料表的唯讀存取權是必要的，您可以使用 IAM 存取政策進行控制。如需詳細資訊，請參閱 the section called “使用政策管理存取權”。您必須根據您是否透過 Cassandra 驅動程式和開發人員工具使用 AWS SDK 或 Cassandra 查詢語言 (CQL) API 呼叫，為系統資料表定義基於標籤的存取控制原則。若要進一步瞭解系統表格的標籤式存取控制，請參閱the section called “ 基於標籤的 Amazon Keyspaces 資源訪問”。

如果您使用 Amazon VPC 端點存取 Amazon Keyspaces，您會在`system.peers`表格中看到 Amazon Keyspaces 有權查看的每個 Amazon VPC 端點的項目。因此，您的 Cassandra 驅動程序可能會在表中發出有關控制節點本身的警告消息。`system.peers`您可以放心地忽略此警告。

時間戳記

在 Amazon Keyspaces 間中，與 Apache Cassandra 中的默認時間戳兼容的單元格級時間戳是一種選擇加入功能。

只有在開啟資料表的用戶端時間戳記時，才`WRITETIME`能使用`USING TIMESTAMP`子句和函數。若要進一步了解 Amazon Keyspaces 間中的用戶端時間戳記，請參閱。用戶端時間時間

Amazon Keyspaces 中支持的卡桑德拉 API，操作，函數和數據類型

Amazon Keyspaces（對於阿帕奇卡桑德拉）與卡桑德拉查詢語言（CQL）3.11 API 兼容（向後兼容版本 2.x）。

Amazon Keyspaces 支援所有常用的 Cassandra 資料平面作業，例如建立金鑰空間和表格、讀取資料和寫入資料。

以下各節列出支援的功能。

主題

卡桑德拉 API 支持

API 操作	支援
CREATE KEYSPACE	是
ALTER KEYSPACE	是
DROP KEYSPACE	是
CREATE TABLE	是
ALTER TABLE	是
DROP TABLE	是
CREATE INDEX	否
DROP INDEX	否
UNLOGGED BATCH	是

API 操作	支援
LOGGED BATCH	否
SELECT	是
INSERT	是
DELETE	是
UPDATE	是
USE	是
CREATE TYPE	否
ALTER TYPE	否
DROP TYPE	否
CREATE TRIGGER	否
DROP TRIGGER	否
CREATE FUNCTION	否
DROP FUNCTION	否
CREATE AGGREGATE	否
DROP AGGREGATE	否
CREATE MATERIALIZED VIEW	否
ALTER MATERIALIZED VIEW	否
DROP MATERIALIZED VIEW	否
TRUNCATE	否

卡桑德拉控制平面 API 支持

由於 Amazon Keyspaces 是受管理的，因此不需要用於管理叢集和節點設定的 Cassandra 控制平面 API 操作。因此，以下卡桑德拉功能不適用。

功能	原因
持久寫入切換	所有寫入都經久耐用
讀取修復設定	不適用
GC 寬限秒	不適用
綻放濾鏡設定	不適用
壓實設定	不適用
Compression settings (壓縮設定)	不適用
快取設定	不適用
安全性設定	由 IAM 取代

卡桑德拉數據平面 API 支持

功能	支援
對於選擇和插入語句的 JSON 支持	是
靜態列	是
生存時間 (TTL)	是

卡桑德拉函數支持

如需支援函數的詳細資訊，請參閱the section called “內建函數”。

函數	支援
Aggregate 函式	否
Blob轉換	是
Cast	是
Datetime 函式	是
時間轉換功能	是
TimeUuid 函式	是
Token	是
User defined functions (UDF)	否
Uuid	是

卡桑德拉數據類型支持

資料類型	支援	注意
ascii	是	
bigint	是	
blob	是	
boolean	是	
counter	是	
date	是	
decimal	是	
double	是	

資料類型	支援	注意
float	是	
frozen	是	
inet	是	
int	是	
list	是	
map	是	
set	是	
smallint	是	
text	是	
time	是	
timestamp	是	
timeuuid	是	
tinyint	是	
tuple	是	
user-defined types (UDT)	否	若要使用通訊協定緩衝區重構 UDT，請參閱 Amazon Keyspaces 通訊協定緩衝區。
uuid	是	
varchar	是	
varint	是	

亞馬遜密鑰空間中支持的 Apache 卡桑德拉一致性級別

本節中的主題描述了哪些 Apache 卡桑德拉一致性級別支持讀取和寫入操作亞馬遜密鑰空間（Apache 卡桑德拉）。

主題

寫入一致性層級

Amazon 密鑰空間會跨多個可用區域複寫所有寫入操作三次，以提高耐用性和高可用性。寫入會持久儲存，然後才能使用LOCAL_QUORUM一致性水平。對於每 1 KB 寫入，您需針對使用佈建容量模式的表格支付 1 個寫入容量單位 (WCU) 的費用，或針對使用隨選模式的表格支付 1 個寫入請求單位 (WRU) 的費用。

您可以使用cqlsh將當前會話中所有查詢的一致性設置為LOCAL_QUORUM使用下面的代碼。

```
CONSISTENCY LOCAL_QUORUM;
```

要以編程方式配置一致性級別，可以使用適當的 Cassandra 客戶端驅動程序設置一致性。例如，4.x 版 Java 驅動程式可讓您在app config文件，如下圖所示。

```
basic.request.consistency = LOCAL_QUORUM
```

如果您使用的是 3.x 版 Java Cassandra 驅動程序，則可以通過添加來指定會話的一致性級別.withQueryOptions(new QueryOptions().setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)如下列程式碼範例所示。

```
Session session = Cluster.builder()
                      .addContactPoint(endPoint)
                      .withPort(portNumber)
                      .withAuthProvider(new SigV4AuthProvider("us-east-2"))
                      .withSSL()
                      .withQueryOptions(new
 QueryOptions().setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
```

```
                        .build()
                        .connect();
```

若要設定特定寫入作業的一致性層級，您可以在呼叫時定義一致性`QueryBuilder.insertInto`用一個`setConsistencyLevel`當您使用 Java 驅動程序時的參數。

讀取一致性等級

Amazon 金鑰空間支援三種讀取一致性層級：`ONE`,`LOCAL_ONE`，以及`LOCAL_QUORUM`。在一個`LOCAL_QUORUM`閱讀時，Amazon 密鑰空間返回反映所有先前成功寫入操作的最新更新的響應。使用一致性級別`ONE`或者`LOCAL_ONE`可以改善讀取要求的效能和可用性，但回應可能不會反映最近完成寫入的結果。

對於每個 4 KB 讀取使用`ONE`或者`LOCAL_ONE`一致性：使用佈建容量模式的表格需支付 0.5 個讀取容量單位 (RCU) 的費用，或針對使用隨選模式的表格支付 0.5 個讀取請求單位 (RRU) 的費用。對於每個 4 KB 讀取使用`LOCAL_QUORUM`一致性：使用佈建容量模式的表格需支付 1 個讀取容量單位 (RCU) 費用，或針對使用隨選模式的表格支付 1 個讀取請求單位 (RRU) 的費用。

根據每個 4 KB 讀取的每個表格的讀取一致性和讀取容量輸送量模式計費

一致性等級	佈建	隨需
`ONE`	0.5 RCU	0.5 RUS
`LOCAL_ONE`	0.5 RCU	0.5 RUS
`LOCAL_QUORUM`	1 RCU	1 RRU

若要為讀取作業指定不同的一致性，請呼叫`QueryBuilder.select`用一個`setConsistencyLevel`當您使用 Java 驅動程序時的參數。

不支援的一致性

Amazon 密鑰空間不支援下列一致性層級，並會導致例外狀況。

不支援的一致性

阿帕奇·卡桑德拉	Amazon Keyspaces
`EACH_QUORUM`	不支援

阿帕奇·卡桑德拉	Amazon Keyspaces
QUORUM	不支援
ALL	不支援
TWO	不支援
THREE	不支援
ANY	不支援
SERIAL	不支援
LOCAL_SERIAL	不支援

訪問 Amazon Keyspaces（阿帕奇卡桑德拉）

您可以使用控制台，以編程方式通過運行`cqlsh`客戶端 AWS CloudShell， AWS SDK 或使用 Apache 2.0 許可卡桑德拉驅動程序訪問 Amazon Keyspaces。Amazon Keyspaces 支持與 Apache 卡桑德拉 3.11.2 兼容的驅動程序和客戶端。在存取 Amazon Keyspaces 之前，您必須先完成設定， AWS Identity and Access Management 然後授與 IAM 身分存取權限給 Amazon Keyspaces。

設定 AWS Identity and Access Management

註冊一個 AWS 帳戶

如果您沒有 AWS 帳戶，請完成以下步驟來建立一個。

若要註冊成為 AWS 帳戶

1. 開啟 https://portal.aws.amazon.com/billing/signup。
2. 請遵循線上指示進行。

   部分註冊程序需接收來電，並在電話鍵盤輸入驗證碼。

   當您註冊一個時 AWS 帳戶，將創建AWS 帳戶根使用者一個。根使用者有權存取該帳戶中的所有 AWS 服務 和資源。安全性最佳做法是將管理存取權指派給使用者，並僅使用 root 使用者來執行需要 root 使用者存取權的工作。

AWS 註冊過程完成後，會向您發送確認電子郵件。您可以隨時登錄 https://aws.amazon.com/ 並選擇我的帳戶，以檢視您目前的帳戶活動並管理帳戶。

建立具有管理權限的使用者

註冊後，請保護您的 AWS 帳戶 AWS 帳戶根使用者 AWS IAM Identity Center、啟用和建立系統管理使用者，這樣您就不會將 root 使用者用於日常工作。

保護您的 AWS 帳戶根使用者

1. 選擇 Root 使用者並輸入您的 AWS 帳戶 電子郵件地址，以帳戶擁有者身分登入。AWS Management Console在下一頁中，輸入您的密碼。

   如需使用根使用者登入的說明，請參閱 AWS 登入 使用者指南中的以根使用者身分登入。

2. 若要在您的根使用者帳戶上啟用多重要素驗證 (MFA)。

   如需指示，請參閱《IAM 使用者指南》中的為 AWS 帳戶 根使用者啟用虛擬 MFA 裝置 (主控台)。

建立具有管理權限的使用者

1. 啟用 IAM Identity Center。

   如需指示，請參閱 AWS IAM Identity Center 使用者指南中的啟用 AWS IAM Identity Center。

2. 在 IAM 身分中心中，將管理存取權授予使用者。

   若要取得有關使用 IAM Identity Center 目錄 做為身分識別來源的自學課程，請參閱《使用指南》IAM Identity Center 目錄中的「以預設值設定使用AWS IAM Identity Center 者存取」。

以具有管理權限的使用者身分登入

- 若要使用您的 IAM Identity Center 使用者簽署，請使用建立 IAM Identity Center 使用者時傳送至您電子郵件地址的簽署 URL。

  如需使用 IAM 身分中心使用者登入的說明，請參閱使用AWS 登入 者指南中的登入 AWS 存取入口網站。

指派存取權給其他使用者

1. 在 IAM 身分中心中，建立遵循套用最低權限許可的最佳做法的權限集。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「建立權限集」。

2. 將使用者指派給群組，然後將單一登入存取權指派給群組。

   如需指示，請參閱《AWS IAM Identity Center 使用指南》中的「新增群組」。

設置 Amazon Keyspaces

對 Amazon Keyspaces 資源的訪問是使用 IAM 進行管理。您可以使用 IAM 將政策附加到 IAM 使用者、角色和聯合身分，以授與讀取和寫入權限給 Amazon Keyspaces 中的特定資源。

若要開始授予 IAM 身分許可，您可以使用 Amazon Keyspaces 的其中一個 AWS 受管政策：

- AmazonKeyspacesFullAccess— 此政策授予存取 Amazon Keyspaces 中所有資源的權限，並可完全存取所有功能。
- AmazonKeyspacesReadOnlyAccess_v2 — 此政策授予 Amazon 金 Keyspaces 的唯讀許可。

如需受管理策略中定義之處理行動的詳細說明，請參閱the section called “AWS 受管政策”。

若要限制 IAM 身分可執行的動作範圍，或限制身分可存取的資源，您可以建立使用AmazonKeyspacesFullAccess受管政策做為範本的自訂政策，並移除不需要的所有許可。您還可以限制對特定密鑰空間或表格的訪問。如需有關如何限制動作或限制對 Amazon Keyspaces 中特定資源的存取權限的詳細資訊，請參閱the section called “Amazon Keyspaces 如何與 IAM 一起工作”。

若要在建立 AWS 帳戶 並建立授予 IAM 身分存取 Amazon 金 Keyspaces 的政策之後存取 Amazon 金 Keyspaces，請繼續執行下列其中一個部分：

- 使用主控台
- 使用 AWS CloudShell
- 編程連接

使用控制台訪問 Amazon Keyspaces

您可以在https://console.aws.amazon.com/keyspaces/home訪問 Amazon Keyspaces 的控制台。如需有關 AWS Management Console 存取的詳細資訊，請參閱《IAM 使用者指南》 AWS Management Console中的《控制 IAM 使用者存取權限》。

您可以使用控制台在 Amazon Keyspaces 中執行以下操作：

- 建立、刪除和管理金鑰空間和資料表。
- 在表格的「監控」索引標籤上監視重要的表格指標：
  - 計費資料表大小 (位元組)
  - 容量指標
- 使用 CQL 編輯器執行查詢，例如插入、更新和刪除資料。
- 變更帳戶的磁碟分割程式組態。
- 在儀表板上檢視帳戶的效能和錯誤指標。

若要了解如何建立 Amazon Keyspaces 間和表格，並使用範例應用程式資料進行設定，請參閱。開始使用 Amazon Keyspaces（阿帕奇卡桑德拉）

使用 AWS CloudShell 訪問 Amazon Keyspaces

AWS CloudShell 是一個以瀏覽器為基礎的預先驗證殼層，您可以直接從. AWS Management Console 您可以使用偏好的殼層 (Bash PowerShell 或 Z 殼層) 針對 AWS 服務執行 AWS CLI 命令。要使用 Amazon Keyspacescqlsh，您必須安裝. cqlsh-expansion 如需cqlsh-expansion安裝指示，請參閱the section called “使用 cqlsh-expansion”。

您可以AWS CloudShell 從啟動 AWS Management Console，並且您用來登入主控台的 AWS 認證會自動在新的 shell 工作階段中使用。這種使用 AWS CloudShell 者的預先驗證可讓您在使用cqlsh或 AWS CLI 版本 2 (在殼層的運算環境中預先安裝) 與 AWS 服務 (例如 Amazon Keyspaces) 進行互動時略過設定登入資料。

取得的 IAM 許可 AWS CloudShell

管理員可以使用提供的存取管理資源將許可授予 IAM 使用者 AWS Identity and Access Management，以便他們可以存取 AWS CloudShell 和使用環境的功能。

系統管理員授與使用者存取權的最快方法是透過 AWS 受管理的原則。AWS 受管政策是由 AWS建立並管理的獨立政策。的下列 AWS 受管政策 CloudShell 可附加至 IAM 身分：

- AWSCloudShellFullAccess：授予使用權限 AWS CloudShell 以完全訪問所有功能。

如果您想要限制 IAM 使用者可以執行的動作範圍 AWS CloudShell，您可以建立使用AWSCloudShellFullAccess受管政策做為範本的自訂政策。如需有關限制中使用者可使用的動作的詳細資訊 CloudShell，請參閱《使用AWS CloudShell 者指南》中的「使用 IAM 政策管理 AWS CloudShell 存取和使用」。

> **Note**
>
> 您的 IAM 身分還需要一項政策，以授予撥打 Amazon Keyspaces 的權限。

您可以使用 AWS 受管政策授予 IAM 身分存取 Amazon Keyspaces，或從受管政策開始做為範本開始，然後移除不需要的許可。您也可以限制特定金鑰空間和資料表的存取權，以建立自訂原則。Amazon Keyspaces 的下列受管政策可附加至 IAM 身分：

- AmazonKeyspacesFullAccess— 此政策授予使用完全存取所有功能的 Amazon Keyspaces 的權限。

如需受管理策略中定義之處理行動的詳細說明，請參閱the section called “AWS 受管政策”。

如需有關如何限制動作或限制對 Amazon Keyspaces 中特定資源的存取權限的詳細資訊，請參閱the section called “Amazon Keyspaces 如何與 IAM 一起工作”。

使用與 Amazon Keyspaces 交互 AWS CloudShell

AWS CloudShell 從啟動之後 AWS Management Console，您可以使用cqlsh或命令列界面立即開始與 Amazon Keyspaces 互動。如果尚未安裝cqlsh-expansion，請參閱以the section called “使用 cqlsh-expansion”取得詳細步驟。

> **Note**
>
> 使用 cqlsh-expansion in 時 AWS CloudShell，您不需要在撥打電話之前設定認證，因為您已在 shell 中進行驗證。

Connect 到 Amazon Keyspaces 間並創建一個新的密鑰空間。然後從系統表中讀取以確認密鑰空間是使用 AWS CloudShell

1. 從中 AWS Management Console，您可以選擇 CloudShell 導覽列上的下列可用選項來啟動：
   - 選擇圖 CloudShell 示。
   - 開始在搜索框中輸入「cloudshell」，然後選擇該 CloudShell選項。
2. 您可以使用以下命令建立到 Amazon Keyspaces 的連接。請務必使用適用於您所在地區的正確端點取代*###### 1.amazonaws.com#*

   ```
   cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
   ```

   如果連線成功，您應該會看到類似下列範例的輸出。

   ```
   Connected to Amazon Keyspaces at cassandra.us-east-1.amazonaws.com:9142
   [cqlsh 6.1.0 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
   Use HELP for help.
   cqlsh current consistency level is ONE.
   cqlsh>
   ```

3. 使用名mykeyspace稱創建一個新的密鑰空間。您可以使用以下命令來執行此操作。

```
CREATE KEYSPACE mykeyspace WITH REPLICATION = {'class': 'SingleRegionStrategy'};
```

4. 要確認是否創建了密鑰空間，您可以使用以下命令從系統表中讀取。

```
SELECT * FROM system_schema_mcs.keyspaces WHERE keyspace_name = 'mykeyspace';
```

如果呼叫成功，命令列會顯示類似下列輸出的服務回應：

```
 keyspace_name  | durable_writes | replication
----------------+----------------
+-------------------------------------------------------------------------------------------
 mykeyspace     |           True | {'class':
 'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}

(1 rows)
```

以編程方式連接到 Amazon Keyspaces

本主題概述以程式設計方式連接到 Amazon Keyspaces 所需的步驟。它會引導您完成建立 IAM 登入資料，並列出可用的 AWS 服務端點。最後一節說明如何使用 cqlsh 連接到 Amazon Keyspaces。有關使用不同 Apache 卡桑德拉驅動 step-by-step 程序連接到 Amazon Keyspaces 的教程，請參閱。the section called “使用卡桑德拉客戶端驅動程序”如需說明如何從 Amazon VPC 端點連接到 Amazon Keyspaces 的 step-by-step 教學課程，請參閱。the section called “連線至 VPC 端點”

> **Note**
>
> 為了協助您開始使用，您可以在 Amazon Keyspaces 程式 end-to-end 碼範例儲存庫中使用各種 Cassandra 用戶端驅動程式，找到連接至 Amazon Keyspaces 的程式碼範例程式碼範例。GitHub

Amazon Keyspaces 支持與 Apache 卡桑德拉 3.11.2 兼容的驅動程序和客戶端。它假設您已完成中的 AWS 設定指示訪問 Amazon Keyspaces。

如果您已有 AWS 帳戶，請參閱下列主題，以了解如何以程式設計方式使用 cqlsh 存取 Amazon Keyspaces：

主題

- 創建憑據以編程方式訪問 Amazon Keyspaces
- 亞馬遜 Keyspaces 的服務端點
- 用cqlsh於連接到 Amazon Keyspaces
- 使用 AWS CLI
- 使用 API
- 使用 Amazon Keyspaces 與 SDK AWS
- 使用 Cassandra 客戶端驅動程序以編程方式訪問 Amazon Keyspaces
- 教程：從 Amazon 彈性 Kubernetes 服務連接到亞馬遜 Keyspaces

創建憑據以編程方式訪問 Amazon Keyspaces

若要提供使用者和應用程式以程式設計方式存取 Amazon Keyspace 資源的登入資料，您可以執行下列其中一項作業：

- 創建與 Cassandra 用於身份驗證和訪問管理的傳統用戶名和密碼類似的服務特定憑據。 AWS 服務特定登入資料與特定 AWS Identity and Access Management (IAM) 使用者相關聯，而且只能用於為其建立的服務。如需詳細資訊，請參閱 IAM 使用者指南中的將 IAM 與 Amazon Keyspaces 搭配使用 (適用於 Apache 卡桑德拉)。

  > **Warning**
  >
  > IAM 使用者擁有長期登入資料，這會帶來安全風險。為了減輕此風險，我們建議您僅向這些使用者提供執行工作所需的權限，並在不再需要這些使用者時移除這些使用者。

- 為了增強安全性，我們建議您建立適用於所有 AWS 服務的 IAM 身分，並使用臨時登入資料。適用於 Cassandra 客戶端驅動程序的 Amazon 密 Keyspaces Sigv4 身份驗證插件使您可以使用 IAM 訪問密鑰而不是用戶名和密碼對 Amazon 密鑰 Keyspaces 進行身份驗證呼叫。若要進一步了解 Amazon Keyspaces Sigv4 外掛程式如何讓 IAM 使用者、角色和聯合身分在 Amazon Keyspaces API 請求中進行身份驗證，請參閱AWS 簽名版本 4 程序 (Sigv4)。

  您可以從以下位置下載 Sigv4 外掛程式。

  - 爪哇:https://github.com/aws/aws-sigv4-auth-cassandra-java-driver-plugin.
  - Node.js:https://github.com/aws/aws-sigv4-auth-cassandra-nodejs-driver-plugin.
  - Python: https://github.com/aws/aws-sigv4-auth-cassandra-python-driver-plugin.
  - 去：https://github.com/aws/aws-sigv4-auth-cassandra-gocql-driver-plugin。

如需示範如何使用 Sigv4 驗證外掛程式建立連線的程式碼範例，請參閱the section called “使用卡桑德拉客戶端驅動程序”。

主題

- 產生服務特定認證
- 如何創建和配置 Amazon Keyspaces 的 AWS 憑據

產生服務特定認證

服務特定憑據類似於 Cassandra 用於身份驗證和訪問管理的傳統用戶名和密碼。服務特定登入資料可讓 IAM 使用者存取特定 AWS 服務。這些長期憑證無法用於存取其他 AWS 服務。它們與特定 IAM 使用者相關聯，其他 IAM 使用者無法使用。

> **Important**
>
> 服務特定登入資料是與特定 IAM 使用者相關聯的長期登入資料，只能用於為其建立的服務。若要授予 IAM 角色或聯合身分使用臨時登入資料存取所有 AWS 資源的許可，您應該使用 Amazon Keyspaces 的 Sigv4 身份AWS 驗證外掛程式進行身份驗證。

使用下列其中一個程序來產生服務特定認證。

使用主控台產生服務特定認證

使用主控台產生服務特定認證

1. 登入 AWS Management Console 並開啟 AWS Identity and Access Management 主控台，位於https://console.aws.amazon.com/iam/home。
2. 在瀏覽窗格中，選擇 [使用者]，然後選擇您先前建立且具有 Amazon Keyspaces 許可 (附加政策) 的使用者。
3. 選擇安全登入資料。在 Amazon Keyspaces 的登入資料下，選擇產生登入資料以產生服務特定的登入資料。

   您的服務特定憑證現在可供使用。這是您唯一可以下載或查看密碼的時間。您稍後無法進行復原。不過，您可以隨時重設密碼。請將使用者和密碼儲存在安全的位置，因為稍後需要用到。

使用產生服務特定認證 AWS CLI

若要使用 AWS CLI

在產生服務特定認證之前，您需要下載、安裝及設定 AWS Command Line Interface (AWS CLI)：

1. 下載網 AWS CLI 址為 http://aws.amazon.com/cli.

   > **Note**
   >
   > 在視窗上 AWS CLI 運行, macOS 系統, 或 Linux.

2. 請遵循《AWS Command Line Interface 使用者指南》中有關安裝 AWS CLI 和設定 AWS CLI 的說明進行操作。
3. 使用 AWS CLI，執行下列命令為使用者產生服務特定的登入資料`alice`，以便她可以存取 Amazon Keyspaces。

```
aws iam create-service-specific-credential \
    --user-name alice \
    --service-name cassandra.amazonaws.com
```

輸出看起來如下。

```
{
    "ServiceSpecificCredential": {
        "CreateDate": "2019-10-09T16:12:04Z",
        "ServiceName": "cassandra.amazonaws.com",
        "ServiceUserName": "alice-at-111122223333",
        "ServicePassword": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "ServiceSpecificCredentialId": "ACCAYFI33SINPGJEBYESF",
        "UserName": "alice",
        "Status": "Active"
    }
}
```

在輸出中，請注意`ServiceUserName`和的值`ServicePassword`。將這些值儲存在安全的位置，因為稍後會需要這些值。

> **Important**
>
> 這是您唯一可以`ServicePassword`使用的時間。

如何創建和配置 Amazon Keyspaces 的 AWS 憑據

若要使用 AWS CLI、 AWS SDK 或使用 Cassandra 用戶端驅動程式和 Sigv4 外掛程式以程式設計方式存取 Amazon 金 Keyspaces，您需要具有存取金鑰的 IAM 使用者或角色。當您以程式設計 AWS 方式使用時，您會提供 AWS 存取金鑰， AWS 以便在程式設計呼叫中驗證您的身分識別。您的存取金鑰由存取金鑰識別碼 (例如 AKIAIOSFODNN7EXAMPLE) 和秘密存取金鑰 (例如，密碼) 所bPxRfi組成。AKIAIOSFODNN7EXAMPLE 本主題將逐步引導您完成此程序中的必要步驟。

安全性最佳實務建議您建立具有有限許可的 IAM 使用者，而是將 IAM 角色與執行特定工作所需的許可建立關聯。然後，IAM 使用者可以暫時擔任 IAM 角色來執行必要的工作。例如，您帳戶中使用 Amazon Keyspaces 主控台的 IAM 使用者可以切換到角色，以暫時使用主控台中角色的許可。使用者放棄其原始許可並取得指派給該角色的許可。當使用者退出角色時，將恢復其原始許可。使用者用來承擔角色的認證是暫時性的。相反，IAM 使用者擁有長期登入資料，如果他們沒有假設角色直接指派給他們的權限，就會帶來安全風險。為了減輕此風險，我們建議您僅向這些使用者提供執行工作所需的權限，並在不再需要這些使用者時移除這些使用者。如需有關角色的詳細資訊，請參閱 IAM 使用者指南中的角色常見案例：使用者、應用程式和服務。

主題

- 所需的憑據 AWS CLI, 該 AWS SDK, 或 Amazon Keyspaces Sigv4 插件卡桑德拉客戶端驅動程序
- 建立 IAM 使用者，以程式設計方式存取帳戶中的 Amazon Keyspaces AWS
- 為 IAM 使用者建立新的存取金鑰
- 如何管理 IAM 使用者的存取金鑰
- 使用臨時登入資料，透過 IAM 角色和 Sigv4 外掛程式連線到 Amazon Keyspaces

所需的憑據 AWS CLI, 該 AWS SDK, 或 Amazon Keyspaces Sigv4 插件卡桑德拉客戶端驅動程序

驗證 IAM 使用者或角色時需要下列登入資料：

`AWS_ACCESS_KEY_ID`

指定與 IAM 使用者或角色相關聯的 AWS 存取金鑰。

需要存取金鑰`aws_access_key_id`才能以程式設計方式連接到 Amazon 金 Keyspaces。

`AWS_SECRET_ACCESS_KEY`

指定與存取金鑰相關聯的私密金鑰。這基本上是存取金鑰的「密碼」。

需要以編程方式連接到 Amazon Keyspaces。`aws_secret_access_key`

`AWS_SESSION_TOKEN`— 可選

指定當您直接從 AWS Security Token Service 作業擷取的暫時安全憑證時需要的工作階段字符值。如需詳細資訊，請參閱 the section called “使用臨時登入資料連接到 Amazon Keyspaces”。

如果您要與 IAM 使用者連線，`aws_session_token`則不需要。

建立 IAM 使用者，以程式設計方式存取帳戶中的 Amazon Keyspaces AWS

若要透過 AWS SDK 或 Sigv4 外掛程式以程式設計方 AWS CLI式存取 Amazon Keyspaces 的登入資料，您需要先建立 IAM 使用者或角色。以下步驟顯示建立 IAM 使用者和設定該 IAM 使用者以程式設計方式存取 Amazon Keyspaces 的程序：

1. 在 AWS Management Console、、、適用於視窗的 AWS CLI工具中建立使用者 PowerShell，或使用 AWS API 作業。如果您在中建立使用者 AWS Management Console，則會自動建立認證。
2. 如果以程式設計方式建立使用者，則必須在其他步驟中為該使用者建立存取金鑰 (存取金鑰 ID 和秘密存取金鑰)。
3. 授予用戶訪問 Amazon Keyspaces 的權限。

如需建立使用者所需許可的相關資訊，請參閱存取 IAM 資源所需的權限。

建立 IAM 使用者 (主控台)

您可以使用建 AWS Management Console 立 IAM 使用者。

建立具有程式設計存取權的 IAM 使用者 (主控台)

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇 Users (使用者)，然後選擇 Add users (新增使用者)。
3. 為新使用者輸入使用者名稱。這是的登入名稱 AWS。

> **Note**
>
> 使用者名稱可以是長達 64 個字母、數字以及這些字元的組合：加號 (+)、等號 (=)、逗號 (,)、句號 (.)、@ 符號、底線 (_) 以及連字號 (-)。名稱在帳戶中必須是唯一的。它們無法透過大小寫進行區分。例如，您不可以建立兩個名為 TESTUSER 和 testuser 的使用者。

4. 選取 [存取金鑰-程式設計存取]，為新使用者建立存取金鑰。進入最終頁面時，您可以查看或下載訪問密鑰。

   選擇下一步：許可。

5. 在 [設定權限] 頁面上，選擇 [直接連結現有原則]，將權限指派給新使用者。

   此選項會顯示您帳戶中可用的 AWS 受管理政策和客戶管理策略清單。您可以在搜尋欄位`keyspaces`中輸入，以僅顯示與 Amazon Keyspaces 相關的政策。

   對於 Amazon Keyspaces，可用的受管政策為`AmazonKeyspacesFullAccess`和`AmazonKeyspacesReadOnlyAccess`。如需每個策略的詳細資訊，請參閱the section called “AWS 受管政策”。

   為了測試目的並遵循連線教學課程，請選取新 IAM 使用者的`AmazonKeyspacesReadOnlyAccess`政策。附註：最佳作法是，我們建議您遵循最低權限原則，並建立自訂原則，以限制對特定資源的存取，而且只允許執行必要動作。如需 IAM 政策的詳細資訊，並檢視 Amazon Keyspaces 的範例政策，請參閱the section called “Amazon Keyspaces 基於身份的政策”。建立自訂權限原則之後，請將原則附加至角色，然後讓使用者暫時擔任適當的角色。

   選擇下一步：標籤。

6. 在「新增標記 (選用)」頁面上，您可以為使用者新增標籤，或選擇「下一步：檢閱」。
7. 在「複查」頁面上，您可以看到目前為止所做的所有選擇。當您準備好繼續時，請選擇 [建立使用者]。
8. 若要檢視使用者的存取金鑰 (存取金鑰 ID 和私密存取金鑰)，請選擇密碼和存取金鑰旁的 Show (顯示)。若要儲存取金鑰，請選擇 Download .csv (下載 .csv)，然後將檔案儲存到安全的位置。

> **Important**
>
> 這是您查看或下載秘密訪問密鑰的唯一機會，您需要這些信息才能使用 Sigv4 插件。請將使用者的新存取金鑰 ID 和私密存取金鑰存放在安全處。在此步驟之後，您將無法再存取該私密金鑰。

建立 IAM 使用者 (AWS CLI)

您可以使用 AWS CLI 建立 IAM 使用者。

若要建立具有程式設計存取權的 IAM 使用者 ()AWS CLI

1. 使用下面的 AWS CLI 代碼創建一個用戶。

   - aws iam create-user

2. 為使用者提供程式設計存取權限。這需要訪問密鑰，可以通過以下方式生成。

   - AWS CLI: aws iam create-access-key
   - 視窗工具 PowerShell:New-IAMAccessKey
   - IAM API：CreateAccessKey

   > **Important**
   >
   > 這是您查看或下載秘密訪問密鑰的唯一機會，您需要這些信息才能使用 Sigv4 插件。請將使用者的新存取金鑰 ID 和私密存取金鑰存放在安全處。在此步驟之後，您將無法再存取該私密金鑰。

3. 將`AmazonKeyspacesReadOnlyAccess`原則附加至定義使用者權限的使用者。附註：最佳作法是，建議您將使用者新增至群組並將原則附加至群組，而不是直接附加至使用者，以管理使用者權限。

   - AWS CLI: aws iam attach-user-policy

為 IAM 使用者建立新的存取金鑰

如果您已有 IAM 使用者，則可以隨時建立新的存取金鑰。如需金鑰管理的詳細資訊，例如如何輪換存取金鑰，請參閱管理 IAM 使用者的存取金鑰。

建立 IAM 使用者 (主控台) 的存取金鑰

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇使用者 。
3. 選擇您要建立其存取金鑰的使用者名稱。
4. 在使用者的 [摘要] 頁面上，選擇 [安全性認證] 索引標籤。
5. 在 Access keys (存取金鑰) 區段中，選擇 Create access key (建立存取金鑰)。

   若要查看新的存取金鑰，請選擇 Show (顯示)。您的憑證看起來如下：

   - 存取金鑰 ID：AKIAIOSFODNN7EXAMPLE
   - 秘密訪問密鑰：密鑰鑰匙/K7M登/CYEXAMPLEKEY bPxRfi

   > **Note**
   >
   > 在關閉此對話方塊後，您將無法再次存取該私密存取金鑰。

6. 若要下載金鑰對，請選擇 Download .csv file (下載 .csv 檔案)。請將金鑰存放在安全位置。
7. 下載 .csv 檔案後，請選擇 Close (關閉)。

當您建立存取金鑰時，在預設情況下，該金鑰對是作用中的，且您可以立即使用該金鑰對。

如何管理 IAM 使用者的存取金鑰

最佳做法是，建議您不要將存取金鑰直接嵌入程式碼中。 AWS SDK 和 AWS 命令行工具使您可以將訪問密鑰放在已知位置，以便您不必將它們保留在代碼中。將存取金鑰放入以下其中一個位置：

- 環境變數 — 在多租戶系統上，選擇使用者環境變數，而非系統環境變數。
- CLI 憑證檔案 – 當您執行命令 `aws configure` 時，會更新 `credentials` 和 `config` 檔案。該`credentials`文件位`~/.aws/credentials`於 Linux，macOS 或 Unix，或在視窗`C:\Users\`*`USERNAME`*`\.aws\credentials`上。此檔案可包含 `default` 描述檔和任何具名描述檔的憑證詳細資訊。

- CLI 組態檔 – 當您執行命令 `aws configure` 時，會更新 `credentials` 和 `config` 檔案。該`config`文件位`~/.aws/config`於 Linux，macOS 或 Unix，或在視窗`C:\Users\USERNAME\.aws\config`上。此檔案包含預設描述檔和任何具名描述檔的組態設定。

將存取金鑰儲存為環境變數是. the section called “適用於 Java 4.x 的驗證外掛程式” 用戶端會使用預設認證提供者鏈結來搜尋認證，而儲存為環境變數的存取金鑰會先於所有其他位置，例如組態檔案。如需詳細資訊，請參閱組態設定和優先順序。

下列範例說明如何為預設使用者設定環境變數。

Linux, macOS, or Unix

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
$ export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of security token>
```

設定環境變數會變更使用的數值，直到 Shell 工作階段結束或直到您將該變數設為其他數值。您可以在 Shell 的啟動指令碼中設定變數，讓它們跨未來的工作階段持續生效。

Windows Command Prompt

```
C:\> setx AWS_ACCESS_KEY_ID AKIAIOSFODNN7EXAMPLE
C:\> setx AWS_SECRET_ACCESS_KEY wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
C:\> setx AWS_SESSION_TOKEN AQoDYXdzEJr...<remainder of security token>
```

使用 `set` 設定環境變數會變更使用的數值，直到目前命令提示工作階段結束或直到您將該變數設為其他數值。使用 `setx` 設定環境變數時，將會變更在目前命令提示工作階段及您在執行命令後建立的所有命令提示工作階段中使用的數值。不會影響您執行命令當時已執行的其他命令 Shell。

PowerShell

```
PS C:\> $Env:AWS_ACCESS_KEY_ID="AKIAIOSFODNN7EXAMPLE"
PS C:\> $Env:AWS_SECRET_ACCESS_KEY="wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
PS C:\> $Env:AWS_SESSION_TOKEN="AQoDYXdzEJr...<remainder of security token>"
```

如果您在 PowerShell 提示中設定環境變數 (如前面範例所示)，則只會儲存目前工作階段持續時間的值。若要讓環境變數設定在所有 PowerShell 和 [命令提示字元] 工作階段中持續存放，請使用 [控制台] 中的系統應用程式來儲存。或者，您可以將變量添加到您的個 PowerShell 人資料中，為所有 future 的 PowerShell 會話設置該變量。如需有關儲存環境變數或在工作階段中保留環境變數的詳細資訊，請參閱PowerShell 文件。

使用臨時登入資料，透過 IAM 角色和 Sigv4 外掛程式連線到 Amazon Keyspaces

為了增強安全性，您可以使用臨時憑據與 Sigv4 插件進行身份驗證。在許多情況下，您不必像 IAM 使用者一樣需要永遠不會過期的長期存取金鑰。相反地，您可以建立 IAM 角色並產生臨時安全登入資料。臨時安全登入資料包含存取金鑰 ID 和私密存取金鑰，但其中也包含指出登入資料何時到期的安全符記。若要進一步了解如何使用 IAM 角色而非長期存取金鑰，請參閱切換至 IAM 角色 (AWS API)。

若要開始使用臨時登入資料，您首先需要建立 IAM 角色。

建立 IAM 角色，以授予 Amazon 金 Keyspaces 的唯讀存取權

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在導覽窗格中，選擇角色，然後選擇建立角色。
3. 在 [建立角色] 頁面的 [選取信任實體的類型] 下，選擇 [AWS 服務]。在 [選擇使用案例] 下方，選擇 [Amazon EC2]，然後選擇 [下一步]。
4. 在「新增許可」頁面的「權限政策」下，從政策清單中選擇「Amazon Keyspaces 唯讀存取」，然後選擇「下一步」。
5. 在 [名稱、檢閱和建立] 頁面上，輸入角色的名稱，然後檢閱 [選取信任的實體] 和 [新增權限] 區段。您也可以在此頁面上新增角色的選擇性標籤。完成後，請選取 [建立角色]。請記住此名稱，因為啟動 Amazon EC2 執行個體時會需要這個名稱。

若要在程式碼中使用臨時安全登入資料，您可以透過程式設計方式呼叫類似`AssumeRole`的 AWS Security Token Service API，並從您在上一步中建立的 IAM 角色擷取產生的認證和工作階段權杖。然後，您可以使用這些值作為後續呼叫的認證 AWS。下列範例顯示如何使用臨時安全性登入資料的虛擬程式碼：

```
assumeRoleResult = AssumeRole(role-arn);
tempCredentials = new SessionAWSCredentials(
   assumeRoleResult.AccessKeyId,
   assumeRoleResult.SecretAccessKey,
   assumeRoleResult.SessionToken);
cassandraRequest = CreateAmazoncassandraClient(tempCredentials);
```

如需使用 Python 驅動程式實作臨時登入資料以存取 Amazon Keyspaces 的範例，請參閱???。

如需如何呼叫 `AssumeRole`、`GetFederationToken` 以及其他 API 操作的詳細資訊，請參閱 AWS Security Token Service API 參考。有關從結果取得臨時安全憑證和工作階段權杖的詳細資訊，請參閱

您正在使用的軟體開發套件的說明文件。您可以在主要文件頁面的 AWS SDK 和工具組區段中找到所有 SDK 的AWS 文件。

亞馬遜 Keyspaces 的服務端點

主題

網路連接埠和協定

您可以通過運行`cqlsh`客戶端，使用 Apache 2.0 許可卡桑德拉驅動程序或使用AWS CLI和 SDK 以編程方式訪問亞馬遜 Keyspaces。AWS

下表顯示不同存取機制的連接埠和通訊協定。

程序化訪問	連線埠	通訊協定
CQLSH	9142	TLS
卡桑德拉驅動程序	9142	TLS
AWS CLI	443	HTTPS
AWS SDK	443	HTTPS

對於 TLS 連接，亞馬遜 Keyspaces 使用星空 CA 對服務器進行身份驗證。如需詳細資訊，請參閱本章中的the section called “如何手動設定 TLS 的`cqlsh`連線”或驅動程式開始之前—the section called “使用卡桑德拉客戶端驅動程序”節。

全球端點

亞馬遜 Keyspaces 在下面AWS 區域是可用的。此表格顯示每個區域的可用服務端點。

區域名稱	區域	端點	通訊協定
美國東部 (俄亥俄)	us-east-2	cassandra.us-east-2.amazonaws.com	HTTPS 和 TLS
美國東部 (維吉尼亞北部)	us-east-1	cassandra.us-east-1.amazonaws.com cassandra-fips.us-east-1.amazonaws.com	HTTPS 和 TLS TLS
美國西部 (加州北部)	us-west-1	cassandra.us-west-1.amazonaws.com	HTTPS 和 TLS
美國西部 (奧勒岡)	us-west-2	cassandra.us-west-2.amazonaws.com cassandra-fips.us-west-2.amazonaws.com	HTTPS 和 TLS TLS
亞太區域 (香港)	ap-east-1	cassandra.ap-east-1.amazonaws.com	HTTPS 和 TLS
亞太區域 (孟買)	ap-south-1	cassandra.ap-south-1.amazonaws.com	HTTPS 和 TLS
亞太區域 (首爾)	ap-northeast-2	cassandra.ap-northeast-2.amazonaws.com	HTTPS 和 TLS
亞太區域 (新加坡)	ap-southeast-1	cassandra.ap-southeast-1.amazonaws.com	HTTPS 和 TLS
亞太區域 (雪梨)	ap-southeast-2	cassandra.ap-southeast-2.amazonaws.com	HTTPS 和 TLS
亞太區域 (東京)	ap-northeast-1	cassandra.ap-northeast-1.amazonaws.com	HTTPS 和 TLS

區域名稱	區域	端點	通訊協定
加拿大 (中部)	ca-central-1	cassandra.ca-central-1.amazonaws.com	HTTPS 和 TLS
歐洲 (法蘭克福)	eu-central-1	cassandra.eu-central-1.amazonaws.com	HTTPS 和 TLS
歐洲 (愛爾蘭)	eu-west-1	cassandra.eu-west-1.amazonaws.com	HTTPS 和 TLS
歐洲 (倫敦)	eu-west-2	cassandra.eu-west-2.amazonaws.com	HTTPS 和 TLS
歐洲 (巴黎)	eu-west-3	cassandra.eu-west-3.amazonaws.com	HTTPS 和 TLS
歐洲 (斯德哥爾摩)	eu-north-1	cassandra.eu-north-1.amazonaws.com	HTTPS 和 TLS
中東 (巴林)	me-south-1	cassandra.me-south-1.amazonaws.com	HTTPS 和 TLS
南美洲 (聖保羅)	sa-east-1	cassandra.sa-east-1.amazonaws.com	HTTPS 和 TLS
AWS GovCloud (美國東部)	us-gov-east-1	cassandra.us-gov-east-1.amazonaws.com	HTTPS 和 TLS
AWS GovCloud (美國西部)	us-gov-west-1	cassandra.us-gov-west-1.amazonaws.com	HTTPS 和 TLS

AWS GovCloud (US) RegionFIPS 端點

中可用的 AWS GovCloud (US) Region FIPS 端點。如需詳細資訊，請參閱AWS GovCloud (US)使用者指南中的 Amazon Keyspaces。

區域名稱	區域	FIPS 端點	通訊協定
AWS GovCloud (美國東部)	us-gov-east-1	cassandra.us-gov-east-1.amazonaws.com	HTTPS 和 TLS
AWS GovCloud (美國西部)	us-gov-west-1	cassandra.us-gov-west-1.amazonaws.com	HTTPS 和 TLS

中國區域端點

下列 Amazon Keyspaces 端點可在中AWS國區域使用。

若要存取這些端點，您必須註冊一組獨立的中國區域專屬帳戶認證。如需詳細資訊，請參閱中國註冊、帳戶和憑證。

區域名稱	區域	端點	通訊協定
中國 (北京)	cn-north-1	卡桑德拉. CN-北阿馬遜. COM	HTTPS 和 TLS
中國 (寧夏)	cn-northwest-1	卡桑德拉 .cn-西北部-1. 亞馬遜.	HTTPS 和 TLS

用`cqlsh`於連接到 Amazon Keyspaces

要使用連接到 Amazon Keyspacescqlsh，您可以使用`cqlsh-expansion`. 這是一個包含常見的 Apache 卡桑德拉工具一樣，`cqlsh`並預先配置為 Amazon Keyspaces 助手，同時保持與 Apache 卡桑德拉完全兼容的工具包。該`cqlsh-expansion`集成了 Sigv4 身份驗證插件，並允許您使用 IAM 訪問密鑰而不是用戶名和密碼進行連接。您只需要安裝`cqlsh`腳本來建立連接，而不是完整的 Apache 卡桑

德拉分佈，因為 Amazon Keyspaces 是無服務器的。此輕量型安裝套件cqlsh-expansion包含可在任何支援 Python 的平台上安裝的傳統cqlsh指令碼。

如需有關的一般資訊cqlsh，請參閱cqlsh：CQL 殼層。

主題

- 使用連cqlsh-expansion接到 Amazon Keyspaces
- 如何手動設定 TLS 的cqlsh連線

使用連**cqlsh-expansion**接到 Amazon Keyspaces

安裝和配置 **cqlsh-expansion**

1. 若要安裝 cqlsh-expansion Python 套件，您可以執行pip命令。這將使用 pip install 以及包含依賴關係列表的文件在您的計算機上安裝cqlsh-expansion腳本。--user flag告訴使pip用 Python 用戶安裝目錄為您的平台。在基於 Unix 的系統上，這應該是~/.local/目錄。

   你需要 Python 3 來安裝cqlsh-expansion，找出你的 Python 版本，使用Python --version. 要安裝，您可以運行以下命令。

   ```
   python3 -m pip install --user cqlsh-expansion
   ```

   輸出看起來應該類似於這個。

   ```
   Collecting cqlsh-expansion
     Downloading cqlsh_expansion-0.9.6-py3-none-any.whl (153 kB)
        ######################################## 153.7/153.7 KB 3.3 MB/s eta 0:00:00
   Collecting cassandra-driver
     Downloading cassandra_driver-3.28.0-cp310-cp310-
   manylinux_2_17_x86_64.manylinux2014_x86_64.whl (19.1 MB)
        ######################################## 19.1/19.1 MB 44.5 MB/s eta 0:00:00
   Requirement already satisfied: six>=1.12.0 in /usr/lib/python3/dist-packages (from
    cqlsh-expansion) (1.16.0)
   Collecting boto3
     Downloading boto3-1.29.2-py3-none-any.whl (135 kB)
        ######################################## 135.8/135.8 KB 17.2 MB/s eta 0:00:00
   Collecting cassandra-sigv4>=4.0.2
     Downloading cassandra_sigv4-4.0.2-py2.py3-none-any.whl (9.8 kB)
   Collecting botocore<1.33.0,>=1.32.2
     Downloading botocore-1.32.2-py3-none-any.whl (11.4 MB)
   ```

```
     ######################################## 11.4/11.4 MB 60.9 MB/s eta 0:00:00
Collecting s3transfer<0.8.0,>=0.7.0
  Downloading s3transfer-0.7.0-py3-none-any.whl (79 kB)
     ######################################## 79.8/79.8 KB 13.1 MB/s eta 0:00:00
Collecting jmespath<2.0.0,>=0.7.1
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Collecting geomet<0.3,>=0.1
  Downloading geomet-0.2.1.post1-py3-none-any.whl (18 kB)
Collecting python-dateutil<3.0.0,>=2.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     ######################################## 247.7/247.7 KB 33.1 MB/s eta 0:00:00
Requirement already satisfied: urllib3<2.1,>=1.25.4 in /usr/lib/python3/dist-
packages (from botocore<1.33.0,>=1.32.2->boto3->cqlsh-expansion) (1.26.5)
Requirement already satisfied: click in /usr/lib/python3/dist-packages (from
 geomet<0.3,>=0.1->cassandra-driver->cqlsh-expansion) (8.0.3)
Installing collected packages: python-dateutil, jmespath, geomet, cassandra-driver,
 botocore, s3transfer, boto3, cassandra-sigv4, cqlsh-expansion
  WARNING: The script geomet is installed in '/home/ubuntu/.local/bin' which is not
 on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this
 warning, use --no-warn-script-location.
  WARNING: The scripts cqlsh, cqlsh-expansion and cqlsh-expansion.init are
 installed in '/home/ubuntu/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this
 warning, use --no-warn-script-location.
Successfully installed boto3-1.29.2 botocore-1.32.2 cassandra-driver-3.28.0
 cassandra-sigv4-4.0.2 cqlsh-expansion-0.9.6 geomet-0.2.1.post1 jmespath-1.0.1
 python-dateutil-2.8.2 s3transfer-0.7.0
```

如果安裝目錄不在中PATH，您需要依照作業系統的指示來新增安裝目錄。下面是一個例子為

```
export PATH="$PATH:/home/ubuntu/.local/bin"
```

要確認該軟件包已安裝，您可以運行以下命令。

```
cqlsh-expansion --version
```

輸出應該是這樣的。

```
cqlsh 6.1.0
```

2. 若要設定cqlsh-expansion，您可以執行安裝後指令碼來自動完成下列步驟：

1. 如果目.cassandra錄尚未存在，請在使用者主目錄中建立該目錄。
2. 將預先設定的cqlshrc組態檔複製到.cassandra目錄中。
3. 將星空數位憑證複製到.cassandra目錄中。Amazon Keyspaces 使用此憑證來設定與傳輸層安全性 (TLS) 的安全連線。傳輸中的加密功能會在資料往返 Amazon Keyspaces 時加密，藉此提供額外的資料保護層。

要首先查看腳本，您可以在 Github 存儲庫中訪問它post_install.py。

要使用該腳本，您可以運行以下命令。

```
cqlsh-expansion.init
```

> **Note**
>
> 當您解除安裝cqlsh-expansion使用時，不會移除安裝後指令碼所建立的目錄和檔案pip uninstall，而且必須手動刪除。

使用連接到 Amazon Keyspaces **cqlsh-expansion**

1. 配置您的 AWS 區域 並將其添加為用戶環境變量。

   若要將預設 Region 新增為 Unix 系統上的環境變數，您可以執行下列命令。在此範例中，我們使用美國東部 (維吉尼亞北部)。

   ```
   export AWS_DEFAULT_REGION=us-east-1
   ```

   如需如何設定環境變數 (包括其他平台) 的相關資訊，請參閱如何設定環境變數。
2. 尋找您的服務端點。

   為您的區域選擇適合的服務端點。若要檢閱 Amazon Keyspaces 的可用端點，請參閱the section called “服務端點”。在此範例中，我們使用端點cassandra.us-east-1.amazonaws.com。
3. 設定驗證方法。

   建議使用 IAM 存取金鑰 (IAM 使用者、角色和聯合身分) 連線，以增強安全性。

   您必須先完成下列步驟，才能使用 IAM 存取金鑰連線：

a. 建立 IAM 使用者，或遵循最佳實務並建立 IAM 使用者可以承擔的 IAM 角色。如需如何建立 IAM 存取金鑰的詳細資訊，請參閱the section called “身分 AWS 驗證的 IAM 登入資”。

b. 建立 IAM 政策，以授予角色 (或 IAM 使用者) 至少對 Amazon 金 Keyspaces 的唯讀存取權限。如需 IAM 使用者或角色連線至 Amazon Keyspaces 所需許可的詳細資訊，請參閱the section called “訪問 Amazon Keyspaces 表”。

c. 將 IAM 使用者的存取金鑰新增至使用者的環境變數，如下列範例所示。

```
export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
```

如需如何設定環境變數 (包括其他平台) 的相關資訊，請參閱如何設定環境變數。

> **Note**
>
> 如果從 Amazon EC2 執行個體連線，您還需要在安全群組中設定輸出規則，以允許從執行個體傳輸到 Amazon Keyspaces 的流量。如需如何檢視和編輯 EC2 輸出規則的詳細資訊，請參閱 Amazon EC2 使用者指南中的向安全群組新增規則。

4. 使用`cqlsh-expansion`和 SIGv4 身份驗證 Connect 到 Amazon Keyspaces。

要使用連接到 Amazon Keyspaces`cqlsh-expansion`，您可以使用以下命令。確保將服務端點替換為您區域的正確端點。

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 --ssl
```

如果連線成功，您應該會看到類似下列範例的輸出。

```
Connected to Amazon Keyspaces at cassandra.us-east-1.amazonaws.com:9142
[cqlsh 6.1.0 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
cqlsh current consistency level is ONE.
cqlsh>
```

如果遇到連線錯誤，請參閱以the section called “Cqlsh 連線錯誤”取得疑難排解資訊。

- 使用服務特定登入資料 Connect 到 Amazon Keyspaces。

要與 Cassandra 用於身份驗證的傳統用戶名和密碼組合連接，您必須首先為 Amazon 密 Keyspaces 創建特定服務的憑據，如中所述。the section called “服務特定認證”您還必須授予該使用者存取 Amazon Keyspaces 的權限，如需詳細資訊，請參閱the section called “訪問 Amazon Keyspaces 表”。

為使用者建立服務特定認證和權限之後，您必須更新cqlshrc檔案，通常位於使用者目錄路徑~/.cassandra/中。在該cqlshrc文件中，轉到卡桑德拉[authentication]部分，並註釋掉[auth_provider]使用下的 SIGv4 模塊和類「;」字符，如下面的例子。

```
[auth_provider]

; module = cassandra_sigv4.auth

; classname = SigV4AuthProvider
```

更新cqlshrc檔案後，您可以使用下列命令使用服務特定登入資料連線到 Amazon Keyspaces。

```
cqlsh-expansion cassandra.us-east-1.amazonaws.com 9142 -u myUserName -
p myPassword --ssl
```

清除

- 若要移除cqlsh-expansion套件，您可以使用pip uninstall指令。

  ```
  pip3 uninstall cqlsh-expansion
  ```

  此命pip3 uninstall令不會移除安裝後指令碼所建立的目錄和相關檔案。若要移除安裝後指令碼所建立的資料夾和檔案，您可以刪除目.cassandra錄。

如何手動設定 TLS 的cqlsh連線

Amazon Keyspaces 僅接受使用傳輸層安全性 (TLS) 的安全連線。您可以使用自動為您下載憑證並安裝預先設定cqlshrc組態檔的cqlsh-expansion公用程式。如需詳細資訊，請參閱本頁the section called “使用 cqlsh-expansion”上的。

如果您想要手動下載憑證並設定連線，可以使用下列步驟進行。

1. 使用下列指令下載 Starfield 數位憑證，並儲存在`sf-class2-root.crt`本機或主目錄中。

```
curl https://certs.secureserver.net/repository/sf-class2-root.crt -O
```

> **Note**
>
> 您也可以使用 Amazon 數位憑證連線到 Amazon Keyspaces，如果您的用戶端成功連線至 Amazon Keyspaces，則可以繼續這麼做。Starfield 憑證為使用舊版憑證授權單位的用戶端提供額外的向後相容性。

2. 例如，在卡桑德拉主目錄中打開`cqlshrc`配置文件，`${HOME}/.cassandra/cqlshrc`並添加以下幾行。

```
[connection]
port = 9142
factory = cqlshlib.ssl.ssl_transport_factory

[ssl]
validate = true
certfile =  path_to_file/sf-class2-root.crt
```

使用 AWS CLI

您可以使用 AWS Command Line Interface (AWS CLI) 從命令列控制多項 AWS 服務，並透過指令碼將服務自動化。透過 Amazon Keyspaces，您可以使AWS CLI用資料定義語言 (DDL) 作業，例如建立資料表。此外，您可以使用基礎結構做為程式碼服務和工具，例如AWS CloudFormation和 Terraform。

在您AWS CLI搭配 Amazon Keyspaces 使用之前，必須先取得存取金鑰 ID 及私密存取金鑰 ID 及私密存取金鑰。如需詳細資訊，請參閱the section called “身分 AWS 驗證的 IAM 登入資”。

如需中 Amazon Keyspaces 所有可用命令的完整清單AWS CLI，請參閱命AWS CLI令參考》。

主題

- 下載和設定 AWS CLI
- AWS CLI搭配 Amazon Keyspaces 使用

下載和設定 AWS CLI

可在以下AWS CLI位置取得https://aws.amazon.com/cli。它可在 Windows、macOS，或 Linux 上執行。下載之後AWS CLI，請依照下列步驟進行安裝和設定：

1. 前往AWS Command Line Interface使用者指南
2. 按照安裝AWS CLI和配置的說明進行操作AWS CLI

AWS CLI搭配 Amazon Keyspaces 使用

命令列格式包含 Amazon Keyspaces 操作名稱，隨後接著該操作的參數。AWS CLI 支援適用於參數數值的速記語法以及 JSON。下列 Amazon Keyspaces 範例使用AWS CLI簡寫語法。如需詳細資訊，請參閱搭配AWS CLI 使用簡寫語法。

以下命令創建具有名稱目錄的密鑰空間。

```
aws keyspaces create-keyspace --keyspace-name 'catalog'
```

命令會傳回輸出中資源名稱 (ARN)。

```
{
    "resourceArn": "arn:aws:cassandra:us-east-1:111222333444:/keyspace/catalog/"
}
```

要確認密鑰空間目錄存在，您可以使用下列命令。

```
aws keyspaces get-keyspace --keyspace-name 'catalog'
```

命令輸出會傳回下列值。

```
{
    "keyspaceName": "catalog",
    "resourceArn": "arn:aws:cassandra:us-east-1:111222333444:/keyspace/catalog/"
}
```

下面的命令創建一個名為 book_獎項表。資料表的資料分割索引鍵由資料行組成`year`，`award`叢集索引鍵由資料行組成`rank`，`category`而且兩個叢集資料行都使用遞增排序順序。(為確保易讀性，本節的長命令以分行顯示。)

```
aws keyspaces create-table --keyspace-name 'catalog' --table-name 'book_awards'
```

```
            --schema-definition 'allColumns=[{name=year,type=int},
{name=award,type=text},{name=rank,type=int},
            {name=category,type=text}, {name=author,type=text},
{name=book_title,type=text},{name=publisher,type=text}],
            partitionKeys=[{name=year},
{name=award}],clusteringKeys=[{name=category,orderBy=ASC},{name=rank,orderBy=ASC}]'
```

此命令會產生下列輸出。

```
{
    "resourceArn": "arn:aws:cassandra:us-east-1:111222333444:/keyspace/catalog/table/
book_awards"
}
```

要確認表的元數據和屬性，可以使用下列命令。

```
aws keyspaces get-table --keyspace-name 'catalog' --table-name 'book_awards'
```

此命令會傳回下列輸出。

```
{
    "keyspaceName": "catalog",
    "tableName": "book_awards",
    "resourceArn": "arn:aws:cassandra:us-east-1:111222333444:/keyspace/catalog/table/
book_awards",
    "creationTimestamp": 1645564368.628,
    "status": "ACTIVE",
    "schemaDefinition": {
        "allColumns": [
            {
                "name": "year",
                "type": "int"
            },
            {
                "name": "award",
                "type": "text"
            },
            {
                "name": "category",
                "type": "text"
            },
            {
```

```
                "name": "rank",
                "type": "int"
            },
            {
                "name": "author",
                "type": "text"
            },
            {
                "name": "book_title",
                "type": "text"
            },
            {
                "name": "publisher",
                "type": "text"
            }
        ],
        "partitionKeys": [
            {
                "name": "year"
            },
            {
                "name": "award"
            }
        ],
        "clusteringKeys": [
            {
                "name": "category",
                "orderBy": "ASC"
            },
            {
                "name": "rank",
                "orderBy": "ASC"
            }
        ],
        "staticColumns": []
    },
    "capacitySpecification": {
        "throughputMode": "PAY_PER_REQUEST",
        "lastUpdateToPayPerRequestTimestamp": 1645564368.628
    },
    "encryptionSpecification": {
        "type": "AWS_OWNED_KMS_KEY"
    },
    "pointInTimeRecovery": {
```

```
        "status": "DISABLED"
    },
    "ttl": {
        "status": "ENABLED"
    },
    "defaultTimeToLive": 0,
    "comment": {
        "message": ""
    }
}
```

使用複雜結構定義建立資料表時，從 JSON 檔案載入資料表的結構定義會很有幫助。以下是範例。從 schema_definition.zip 下載結構定義範例 JSON 檔案並擷取`schema_definition.json`，並記下檔案的路徑。在此範例中，結構定義 JSON 檔位於目前的目錄。如需不同的檔案路徑選項，請參閱如何從檔案載入參數。

```
aws keyspaces create-table --keyspace-name 'catalog'
            --table-name 'book_awards' --schema-definition 'file://
schema_definition.json'
```

下列範例說明如何使用其他選項建立名稱為 myTable 的簡單資料表。請注意，這些命令會分成單獨的列，以提高可讀性。此命令顯示如何建立資料表和：

- 設定資料表的容量模式
- 啟用表格的 Point-in-time 復原
- 將資料表的預設存留時間 (TTL) 值設定為一年
- 為表添加兩個標籤

```
aws keyspaces create-table --keyspace-name 'catalog' --table-name 'myTable'
            --schema-definition 'allColumns=[{name=id,type=int},{name=name,type=text},
{name=date,type=timestamp}],partitionKeys=[{name=id}]'
            --capacity-specification
 'throughputMode=PROVISIONED,readCapacityUnits=5,writeCapacityUnits=5'
            --point-in-time-recovery 'status=ENABLED'
            --default-time-to-live '31536000'
            --tags 'key=env,value=test' 'key=dpt,value=sec'
```

此範例顯示如何建立新資料表，該資料表使用客戶管理的金鑰進行加密，並啟用 TTL 以允許您設定欄和資料列的到期日。若要執行此範例，您必須使用自己的金鑰取代客戶受管AWS KMS金鑰的資源 ARN，並確保 Amazon 金 Keyspaces 可存取該金鑰。

```
aws keyspaces create-table --keyspace-name 'catalog' --table-name 'myTable'
            --schema-definition 'allColumns=[{name=id,type=int},{name=name,type=text},
{name=date,type=timestamp}],partitionKeys=[{name=id}]'
            --encryption-specification
 'type=CUSTOMER_MANAGED_KMS_KEY,kmsKeyIdentifier=arn:aws:kms:us-
east-1:111222333444:key/11111111-2222-3333-4444-555555555555'
            --ttl 'status=ENABLED'
```

使用 API

您可以使用 AWS SDK 和 AWS Command Line Interface (AWS CLI) 與亞馬遜密鑰空間互動工作。您可以使用 API 進行資料語言定義 (DDL) 作業，例如建立金鑰空間或資料表。此外，您可以使用基礎結構作為代碼（IaC）服務和工具，如AWS CloudFormation和 Terraform。

在您可以AWS CLI與 Amazon 密鑰空間一起使用之前，您必須獲得訪問密鑰 ID 和秘密訪問密鑰。如需詳細資訊，請參閱the section called “身分 AWS 驗證的 IAM 登入資”。

如需 API 中可用於 Amazon 密鑰空間的所有操作的完整清單，請參閱 Amazon 密鑰空間 API 參考。

使用 Amazon Keyspaces 與 SDK AWS

AWS 軟件開發套件（SDK）可用於許多流行的編程語言。每個 SDK 都提供 API、程式碼範例和說明文件，讓開發人員能夠更輕鬆地以偏好的語言建置應用程式。

SDK 文件	代碼範例
AWS SDK for C++	AWS SDK for C++ 程式碼範例
AWS CLI	AWS CLI 程式碼範例
AWS SDK for Go	AWS SDK for Go 程式碼範例
AWS SDK for Java	AWS SDK for Java 程式碼範例
AWS SDK for JavaScript	AWS SDK for JavaScript 程式碼範例
適用於 Kotlin 的 AWS SDK	適用於 Kotlin 的 AWS SDK 程式碼範例

SDK 文件	代碼範例
AWS SDK for .NET	AWS SDK for .NET 程式碼範例
AWS SDK for PHP	AWS SDK for PHP 程式碼範例
AWS Tools for PowerShell	PowerShell 程式碼範例工具
AWS SDK for Python (Boto3)	AWS SDK for Python (Boto3) 程式碼範例
AWS SDK for Ruby	AWS SDK for Ruby 程式碼範例
適用於 Rust 的 AWS SDK	適用於 Rust 的 AWS SDK 程式碼範例
適用於 SAP ABAP 的 AWS SDK	適用於 SAP ABAP 的 AWS SDK 程式碼範例
適用於 Swift 的 AWS SDK	適用於 Swift 的 AWS SDK 程式碼範例

可用性範例

找不到所需的內容嗎？ 請使用本頁面底部的提供意見回饋連結申請程式碼範例。

使用 Cassandra 客戶端驅動程序以編程方式訪問 Amazon Keyspaces

您可以使用許多第三方的開源卡桑德拉驅動程序連接到 Amazon Keyspaces。Amazon Keyspaces 與支持阿帕奇卡桑德拉 3.11.2 版本卡桑德拉驅動程序兼容。以下是我們測試並建議搭配 Amazon Keyspaces 使用的驅動程式和最新版本：

- `Java v3.3`
- `Java v4.17`
- `Python Cassandra-driver 3.29.1`
- `Node.js cassandra driver -v 4.7.2`
- `GO using GOCQL v1.6`
- `.NET CassandraCSharpDriver -v 3.20.1`

有關卡桑德拉驅動程序的更多信息，請參閱 Apache 卡桑德拉客戶端驅動程序。

> **Note**
>
> 為了協助您開始使用，您可以檢視和下載程式 end-to-end 碼範例，以便透過熱門驅動程式建立 Amazon Keyspaces 的連線。請參閱上 GitHub的 Amazon Keyspaces 示例。

本章中的教學課程包括簡單的 CQL 查詢，以確認已成功建立與 Amazon Keyspaces 的連線。要了解如何在連接到 Amazon Keyspaces 端點後使用密鑰空間和表格，請參閱。語言參考如需說明如何從 Amazon VPC 端點連接到 Amazon Keyspaces 的 step-by-step 教學課程，請參閱。the section called "連線至 VPC 端點"

主題

- 使用卡桑德拉 Java 客戶端驅動程序以編程方式訪問 Amazon Keyspaces
- 使用卡桑德拉 Python 客戶端驅動程序以編程方式訪問 Amazon Keyspaces
- 使用卡桑德拉 Node.js 客戶端驅動程序以編程方式訪問 Amazon Keyspaces
- 使用卡桑德拉 .NET 核心客戶端驅動程序以編程方式訪問 Amazon Keyspaces
- 使用 Cassandra Go 客戶端驅動程序以編程方式訪問 Amazon Keyspaces
- 使用卡桑德拉 Perl 客戶端驅動程序以編程方式訪問 Amazon Keyspaces

使用卡桑德拉 Java 客戶端驅動程序以編程方式訪問 Amazon Keyspaces

本節介紹如何使用 Java 客戶端驅動程序連接到 Amazon Keyspaces。

> **Note**
>
> Java 17 和 DataStax Java 驅動程式 4.17 目前僅提供測試版支援。如需詳細資訊，請參閱 https://docs.datastax.com/en/developer/java-driver/4.17/upgrade_guide/。

若要提供使用者和應用程式以程式設計方式存取 Amazon Keyspace 資源的登入資料，您可以執行下列其中一項作業：

- 建立與特定 AWS Identity and Access Management (IAM) 使用者相關聯的服務特定登入資料。
- 為了增強安全性，我們建議為跨所有 AWS 服務使用的 IAM 身分建立 IAM 存取金鑰。適用於 Cassandra 客戶端驅動程序的 Amazon 密 Keyspaces Sigv4 身份驗證插件使您可以使用 IAM 訪問密

鑰而不是用戶名和密碼對 Amazon 密鑰 Keyspaces 進行身份驗證呼叫。如需詳細資訊，請參閱 the section called “身分 AWS 驗證的 IAM 登入資”。

> **Note**
>
> 有關如何在 Spring Boot 中使用 Amazon Keyspaces 的示例，請參閱https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/java/datastax-v4/spring。

主題

- 開始之前
- S tep-by-step 教程連接到 Amazon Keyspaces 使用 DataStax Java 驅動程序的 Apache 卡桑德拉使用特定於服務的憑據
- S tep-by-step 教程連接到 Amazon Keyspaces 使用 4.x DataStax Java 驅動程序阿帕奇卡桑德拉和 SIGv4 身份驗證插件
- Connect 到 Amazon Keyspaces 使用 3.x DataStax Java 驅動程序阿帕奇卡桑德拉和 Sigv4 身份驗證插件

開始之前

若要連接到 Amazon Keyspaces，您必須先完成下列任務，才能開始。

1. Amazon Keyspaces 需要使用傳輸層安全性 (TLS) 來協助保護與用戶端的連線安全。

   a. 使用下列指令下載 Starfield 數位憑證，並儲存在`sf-class2-root.crt`本機或主目錄中。

   ```
   curl https://certs.secureserver.net/repository/sf-class2-root.crt -O
   ```

   > **Note**
   >
   > 您也可以使用 Amazon 數位憑證連線到 Amazon Keyspaces，如果您的用戶端成功連線至 Amazon Keyspaces，則可以繼續這麼做。Starfield 憑證為使用舊版憑證授權單位的用戶端提供額外的向後相容性。

   b. 將星空數位憑證轉換為信任庫檔案。

   ```
   openssl x509 -outform der -in sf-class2-root.crt -out temp_file.der
   ```

```
keytool -import -alias cassandra -keystore cassandra_truststore.jks -file
 temp_file.der
```

在此步驟中，您需要為金鑰庫建立密碼並信任此憑證。互動式指令看起來像這樣。

```
Enter keystore password:
Re-enter new password:
Owner: OU=Starfield Class 2 Certification Authority, O="Starfield Technologies,
 Inc.", C=US
Issuer: OU=Starfield Class 2 Certification Authority, O="Starfield
 Technologies, Inc.", C=US
Serial number: 0
Valid from: Tue Jun 29 17:39:16 UTC 2004 until: Thu Jun 29 17:39:16 UTC 2034
Certificate fingerprints:
  MD5:  32:4A:4B:BB:C8:63:69:9B:BE:74:9A:C6:DD:1D:46:24
  SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
  SHA256:
 14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB
Signature algorithm name: SHA1withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3
Extensions:
#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: BF 5F B7 D1 CE DD 1F 86   F4 5B 55 AC DC D7 10 C2  ._.......[U.....
0010: 0E A9 88 E7                                        ....
]
[OU=Starfield Class 2 Certification Authority, O="Starfield Technologies,
 Inc.", C=US]
SerialNumber: [    00]
]
#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
  CA:true
  PathLen:2147483647
]
#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: BF 5F B7 D1 CE DD 1F 86   F4 5B 55 AC DC D7 10 C2  ._.......[U.....
0010: 0E A9 88 E7                                        ....
]
```

```
]
Trust this certificate? [no]:  y
```

2. 在 JVM 引數中附加信任庫檔案：

```
-Djavax.net.ssl.trustStore=path_to_file/cassandra_truststore.jks
-Djavax.net.ssl.trustStorePassword=my_password
```

S tep-by-step 教程連接到 Amazon Keyspaces 使用 DataStax Java 驅動程序的 Apache 卡桑德拉使用特定於服務的憑據

以下 step-by-step 教程將引導您使用特定服務的憑據使用 Cassandra 的 Java 驅動程序連接到 Amazon Keyspaces。具體來說，你會使用 4.0 版本的 DataStax Java 驅動程序的阿帕奇卡桑德拉。

主題

- 步驟 1：事前準備
- 步驟 2：設定驅動程式
- 步驟 3：執行範例應用程式

步驟 1：事前準備

要遵循本教程，您需要生成特定於服務的憑據，並為 Apache 卡桑德拉 DataStax Java 驅動程序添加到您的 Java 項目。

- 透過完成中的步驟，為您的 Amazon Keyspaces IAM 使用者產生服務特定的登入資料。the section called “服務特定認證”如果您偏好使用 IAM 存取金鑰進行身份驗證，請參閱the section called “適用於 Java 4.x 的驗證外掛程式”。
- 阿帕奇卡桑德拉 DataStax Java 驅動程序添加到您的 Java 項目。確保您使用的是支持阿帕奇卡桑德拉 3.11.2 的驅動程序版本。有關更多信息，請參閱 DataStax Java 驅動程序阿帕奇卡桑德拉文檔。

步驟 2：設定驅動程式

您可以通過為您的應用程序創建一個配置文件指定 DataStax Java 卡桑德拉驅動程序的設置。此組態檔案會覆寫預設設定，並告知驅動程式使用連接埠 9142 連接至 Amazon Keyspaces 服務端點。如需可用服務端點的清單，請參閱the section called “服務端點”。

建立組態檔案並將檔案儲存在應用程式的資源資料夾中，例如。`src/main/resources/application.conf`開啟`application.conf`並新增下列組態設定。

1. 驗證提供者 — 使用PlainTextAuthProvider類別建立驗證提供者。*ServiceUser#*#和*ServicePassword*應符合您依照中產生服務特定認證的步驟產生服務特定認證時所取得的使用者名稱和密碼。

   > **Note**
   >
   > 您可以通過使用 Apache Cassandra 的 DataStax Java 驅動程序的身份驗證插件來使用短期憑據，而不是在驅動程序配置文件中硬編碼憑據。若要深入了解，請遵循的指示the section called “適用於 Java 4.x 的驗證外掛程式”。

2. 本機資料中心 — 將值設定local-datacenter為您要連線的區域。例如，如果應用程式正在連線到cassandra.us-east-2.amazonaws.com，則將本機資料中心設定為us-east-2。對於所有可用的 AWS 區域，請參閱???。設定slow-replica-avoidance = false為針對較少節點的負載平衡。

3. SSL/TLS — EngineFactory 透過使用單行指定類別的組態檔案中新增區段來初始化 SSL。class = DefaultSslEngineFactory提供信任庫檔案的路徑和您先前建立的密碼。Amazon Keyspaces 不支持hostname-validation對等，因此將此選項設置為 false。

```
datastax-java-driver {

    basic.contact-points = [ "cassandra.us-east-2.amazonaws.com:9142"]
    advanced.auth-provider{
        class = PlainTextAuthProvider
        username = "ServiceUserName"
        password = "ServicePassword"
    }
    basic.load-balancing-policy {
        local-datacenter = "us-east-2"
        slow-replica-avoidance = false
    }

    advanced.ssl-engine-factory {
        class = DefaultSslEngineFactory
        truststore-path = "./src/main/resources/cassandra_truststore.jks"
        truststore-password = "my_password"
        hostname-validation = false
      }
}
```

> **Note**
>
> 您也可以直接在應用程式程式碼中新增信任存放區路徑，或者將信任存放區的路徑新增至 JVM 引數，而不是在組態檔案中加入信任存放區的路徑。

步驟 3：執行範例應用程式

此程式碼範例顯示一個簡單的命令列應用程式，透過使用我們之前建立的組態檔建立連線至 Amazon Keyspaces 的連線集區。它確認連接是通過運行一個簡單的查詢建立。

```
package <your package>;
// add the following imports to your project
import com.datastax.oss.driver.api.core.CqlSession;
import com.datastax.oss.driver.api.core.config.DriverConfigLoader;
import com.datastax.oss.driver.api.core.cql.ResultSet;
import com.datastax.oss.driver.api.core.cql.Row;

public class App
{

    public static void main( String[] args )
    {
        //Use DriverConfigLoader to load your configuration file
        DriverConfigLoader loader =
 DriverConfigLoader.fromClasspath("application.conf");
        try (CqlSession session = CqlSession.builder()
                .withConfigLoader(loader)
                .build()) {

            ResultSet rs = session.execute("select * from system_schema.keyspaces");
            Row row = rs.one();
            System.out.println(row.getString("keyspace_name"));
        }
    }
}
```

> **Note**
>
> 使用try塊來建立連接，以確保它始終關閉。如果您不使用try塊，請記住關閉連接以避免資源洩漏。

S tep-by-step 教程連接到 Amazon Keyspaces 使用 4.x DataStax Java 驅動程序阿帕奇卡桑德拉和 SIGv4 身份驗證插件

下一節介紹如何使用 Sigv4 身份驗證插件的開源 4.x DataStax Java 驅動程序阿帕奇卡桑德拉訪問 Amazon Keyspaces（阿帕奇卡桑德拉）。該插件可從GitHub存儲庫中獲得。

Sigv4 身份驗證外掛程式可讓您在連線至 Amazon Keyspaces 時，為使用者或角色使用 IAM 登入資料。此插件不需要用戶名和密碼，而是使用訪問密鑰簽署 API 請求。如需詳細資訊，請參閱 the section called “身分 AWS 驗證的 IAM 登入資”。

步驟 1：事前準備

若要遵循本教學課程，您需要完成下列工作。

- 如果您尚未這麼做，請依照中的步驟為您的 IAM 使用者或角色建立登入資料the section called “身分 AWS 驗證的 IAM 登入資”。本教學課程假設存取金鑰儲存為環境變數。如需詳細資訊，請參閱 the section called “如何管理存取金鑰”。
- 阿帕奇卡桑德拉 DataStax Java 驅動程序添加到您的 Java 項目。確保您使用的是支持阿帕奇卡桑德拉 3.11.2 的驅動程序版本。有關更多信息，請參閱 DataStax Java 驅動程序阿帕奇卡桑德拉文檔。
- 將身份驗證插件添加到您的應用程序。身份驗證插件支持的 DataStax Java 驅動程序的 Apache 卡桑德拉 4.x 版。如果您使用的是 Apache Maven 或可以使用 Maven 依賴關係的構建系統，請將以下依賴項添加到您的`pom.xml`文件中。

> **Important**
>
> 將插件的版本替換為最新版本，如GitHub 存儲庫中所示。

```
<dependency>
        <groupId>software.aws.mcs</groupId>
        <artifactId>aws-sigv4-auth-cassandra-java-driver-plugin</artifactId>
        <version>4.0.9</version>
</dependency>
```

步驟 2：設定驅動程式

您可以通過為您的應用程序創建一個配置文件指定 DataStax Java 卡桑德拉驅動程序的設置。此組態檔案會覆寫預設設定，並告知驅動程式使用連接埠 9142 連接至 Amazon Keyspaces 服務端點。如需可用服務端點的清單，請參閱the section called “服務端點”。

建立組態檔案並將檔案儲存在應用程式的資源資料夾中，例如。src/main/resources/application.conf開啟application.conf並新增下列組態設定。

1. 驗證提供者 — advanced.auth-provider.class 將設定為的新執行個體software.aws.mcs.auth.SigV4AuthProvider。Sigv4 AuthProvider 是由插件提供用於執行 Sigv4 身份驗證的身份驗證處理程序。
2. 本機資料中心 — 將值設定local-datacenter為您要連線的區域。例如，如果應用程式正在連線到cassandra.us-east-2.amazonaws.com，則將本機資料中心設定為us-east-2。對於所有可用的 AWS 區域，請參閱???。設定slow-replica-avoidance = false為對所有可用節點的負載平衡。
3. Idempotence — 設置應用程序的默認值，以true配置驅動程序始終重試失敗的讀/寫/idempotence準備/執行請求。這是分散式應用程式的最佳作法，可透過重試失敗的要求來協助處理暫時性失敗。
4. SSL/TLS — EngineFactory 透過使用單行指定類別的組態檔案中新增區段來初始化 SSL。class = DefaultSslEngineFactory提供信任庫檔案的路徑和您先前建立的密碼。Amazon Keyspaces 不支持hostname-validation對等，因此將此選項設置為 false。
5. 連線 — 透過設定，為每個端點建立至少 3 個本機連線local.size = 3。這是幫助您的應用程序處理開銷和流量突發的最佳實踐。如需如何根據預期的流量模式計算應用程式每個端點需要多少本機連線的詳細資訊，請參閱the section called “如何設定連線”。
6. 重試政策 — Amazon Keyspaces 重試政策AmazonKeyspacesExponentialRetryPolicy是 Cassandra 驅動程DefaultRetryPolicy序附帶的替代方案。兩個重試原則之間的主要差異在於，您可以設定重試嘗試的次數，AmazonKeyspacesExponentialRetryPolicy以符合您的需求。依預設，的重試次數設定AmazonKeyspacesExponentialRetryPolicy為 3。此外，Amazon Keyspaces 重試政策不會返回通NoHostAvailableException用。相反，Amazon Keyspaces 重試政策會傳回服務傳回的原始例外狀況。如需實作重試政策的更多程式碼範例，請參閱 Github 上的 Amazon Keyspaces 重試政策。
7. 準備好的陳述式 — 設prepare-on-all-nodes定為 false 以最佳化網路使用。

```
datastax-java-driver {
    basic {
        contact-points = [ "cassandra.us-east-2.amazonaws.com:9142"]
        request {
            timeout = 2 seconds
            consistency = LOCAL_QUORUM
            page-size = 1024
            default-idempotence = true
```

```
        }
        load-balancing-policy {
            local-datacenter = "us-east-2"
            class = DefaultLoadBalancingPolicy
            slow-replica-avoidance = false
        }
    }
    advanced {
        auth-provider {
            class = software.aws.mcs.auth.SigV4AuthProvider
            aws-region = us-east-2
        }
        ssl-engine-factory {
            class = DefaultSslEngineFactory
            truststore-path = "./src/main/resources/cassandra_truststore.jks"
            truststore-password = "my_password"
            hostname-validation = false
        }
        connection {
      connect-timeout = 5 seconds
      max-requests-per-connection = 512
      pool {
                local.size = 3
      }
        }
       retry-policy {
           class =  com.aws.ssa.keyspaces.retry.AmazonKeyspacesExponentialRetryPolicy
     max-attempts = 3
     min-wait = 10 mills
     max-wait = 100 mills
        }
        prepared-statements {
     prepare-on-all-nodes = false
        }
    }
}
```

> **Note**
>
> 您也可以直接在應用程式程式碼中新增信任存放區路徑，或者將信任存放區的路徑新增至 JVM 引數，而不是在組態檔案中加入信任存放區的路徑。

步驟 3：運行應用程序

此程式碼範例顯示一個簡單的命令列應用程式，透過使用我們之前建立的組態檔建立連線至 Amazon Keyspaces 的連線集區。它確認連接是通過運行一個簡單的查詢建立。

```
package <your package>;
// add the following imports to your project
import com.datastax.oss.driver.api.core.CqlSession;
import com.datastax.oss.driver.api.core.config.DriverConfigLoader;
import com.datastax.oss.driver.api.core.cql.ResultSet;
import com.datastax.oss.driver.api.core.cql.Row;

public class App
{

    public static void main( String[] args )
    {
        //Use DriverConfigLoader to load your configuration file
        DriverConfigLoader loader =
 DriverConfigLoader.fromClasspath("application.conf");
        try (CqlSession session = CqlSession.builder()
                .withConfigLoader(loader)
                .build()) {

            ResultSet rs = session.execute("select * from system_schema.keyspaces");
            Row row = rs.one();
            System.out.println(row.getString("keyspace_name"));
        }
    }
}
```

> **Note**
>
> 使用try塊來建立連接，以確保它始終關閉。如果您不使用try塊，請記住關閉連接以避免資源洩漏。

Connect 到 Amazon Keyspaces 使用 3.x DataStax Java 驅動程序阿帕奇卡桑德拉和 Sigv4 身份驗證插件

以下部分介紹如何使用 Sigv4 身份驗證插件為阿帕奇卡桑德拉的 3.x 開源 DataStax Java 驅動程序訪問 Amazon Keyspaces。該插件可從GitHub 存儲庫中獲得。

Sigv4 身份驗證外掛程式可讓您在連線至 Amazon Keyspaces 時，針對使用者和角色使用 IAM 登入資料。此插件不需要用戶名和密碼，而是使用訪問密鑰簽署 API 請求。如需詳細資訊，請參閱 the section called “身分 AWS 驗證的 IAM 登入資”。

步驟 1：事前準備

若要執行此程式碼範例，您必須先完成下列工作。

- 按照中的步驟為您的 IAM 使用者或角色建立登入資料the section called “身分 AWS 驗證的 IAM 登入資”。本教學課程假設存取金鑰儲存為環境變數。如需詳細資訊，請參閱 the section called “如何管理存取金鑰”。
- 請依照下列步驟下載 Starfield 數位憑證、將其轉換為信任庫檔案，並將 JVM 引數中的 TrustStore 檔案附加至您的應用程式。the section called “開始之前”
- 阿帕奇卡桑德拉 DataStax Java 驅動程序添加到您的 Java 項目。確保您使用的是支持阿帕奇卡桑德拉 3.11.2 的驅動程序版本。有關更多信息，請參閱 DataStax Java 驅動程序阿帕奇卡桑德拉文檔。
- 將身份驗證插件添加到您的應用程序。身份驗證插件支持的 DataStax Java 驅動程序的 Apache 卡桑德拉 3.x 版。如果您使用的是 Apache Maven 或可以使用 Maven 依賴關係的構建系統，請將以下依賴項添加到您的`pom.xml`文件中。將插件的版本替換為最新版本，如GitHub 存儲庫中所示。

```
<dependency>
        <groupId>software.aws.mcs</groupId>
        <artifactId>aws-sigv4-auth-cassandra-java-driver-plugin_3</artifactId>
        <version>3.0.3</version>
</dependency>
```

步驟 2：運行應用程序

此程式碼範例顯示一個簡單的命令列應用程式，可建立連線至 Amazon Keyspaces 的連線集區。它確認連接是通過運行一個簡單的查詢建立。

```
package <your package>;
// add the following imports to your project

import software.aws.mcs.auth.SigV4AuthProvider;
import com.datastax.driver.core.Cluster;
import com.datastax.driver.core.ResultSet;
import com.datastax.driver.core.Row;
import com.datastax.driver.core.Session;
```

```
public class App
{

    public static void main( String[] args )
    {
        String endPoint = "cassandra.us-east-2.amazonaws.com";
        int portNumber = 9142;
        Session session = Cluster.builder()
                                 .addContactPoint(endPoint)
                                 .withPort(portNumber)
                                 .withAuthProvider(new SigV4AuthProvider("us-east-2"))

                                 .withSSL()
                                 .build()
                                 .connect();

        ResultSet rs = session.execute("select * from system_schema.keyspaces");
        Row row = rs.one();
        System.out.println(row.getString("keyspace_name"));
    }
}
```

使用注意事項：

如需可用端點的清單，請參閱the section called “服務端點”。

如需將 Java 驅動程式與 Amazon Keyspaces 搭配使用時的有用 Java 驅動程式政策、範例和最佳實務，請參閱下列儲存庫：https://github.com/aws-samples/amazon-keyspaces-java-driver-helpers。

使用卡桑德拉 Python 客戶端驅動程序以編程方式訪問 Amazon Keyspaces

在本節中，我們將向您展示如何使用 Python 客戶端驅動程序連接到 Amazon Keyspaces。若要提供使用者和應用程式以程式設計方式存取 Amazon Keyspace 資源的登入資料，您可以執行下列其中一項作業：

- 建立與特定 AWS Identity and Access Management (IAM) 使用者相關聯的服務特定登入資料。
- 為了增強安全性，我們建議您為 IAM 使用者或跨所有 AWS 服務使用的角色建立 IAM 存取金鑰。適用於 Cassandra 客戶端驅動程序的 Amazon 密 Keyspaces Sigv4 身份驗證插件使您可以使用 IAM 訪問密鑰而不是用戶名和密碼對 Amazon 密鑰 Keyspaces 進行身份驗證呼叫。如需詳細資訊，請參閱 the section called “身分 AWS 驗證的 IAM 登入資”。

主題

開始之前

您必須先完成下列工作，才能開始。

Amazon Keyspaces 需要使用傳輸層安全性 (TLS) 來協助保護與用戶端的連線安全。要使用 TLS 連接到 Amazon Keyspaces，您需要下載 Amazon 數字證書並配置 Python 驅動程序以使用 TLS。

使用下列指令下載 Starfield 數位憑證，並儲存在`sf-class2-root.crt`本機或主目錄中。

```
curl https://certs.secureserver.net/repository/sf-class2-root.crt -0
```

> **Note**
>
> 您也可以使用 Amazon 數位憑證連線到 Amazon Keyspaces，如果您的用戶端成功連線至 Amazon Keyspaces，則可以繼續這麼做。Starfield 憑證為使用舊版憑證授權單位的用戶端提供額外的向後相容性。


```
curl https://certs.secureserver.net/repository/sf-class2-root.crt -0
```


使用 Python 驅動程序的 Apache 卡桑德拉和特定於服務的憑據 Connect 到 Amazon Keyspaces

下列程式碼範例說明如何使用 Python 用戶端驅動程式和服務特定登入資料連線至 Amazon Keyspaces。

```
from cassandra.cluster import Cluster
from ssl import SSLContext, PROTOCOL_TLSv1_2 , CERT_REQUIRED
from cassandra.auth import PlainTextAuthProvider

ssl_context = SSLContext(PROTOCOL_TLSv1_2 )
ssl_context.load_verify_locations('path_to_file/sf-class2-root.crt')
```

```
ssl_context.verify_mode = CERT_REQUIRED
auth_provider = PlainTextAuthProvider(username='ServiceUserName',
 password='ServicePassword')
cluster = Cluster(['cassandra.us-east-2.amazonaws.com'], ssl_context=ssl_context,
 auth_provider=auth_provider, port=9142)
session = cluster.connect()
r = session.execute('select * from system_schema.keyspaces')
print(r.current_rows)
```

使用注意事項：

1. 取代"*path_to_file*/sf-class2-root.crt"為第一個步驟中儲存的憑證路徑。
2. *####ServiceUser#########ServicePassword########################* 產生服務特定認證
3. 如需可用端點的清單，請參閱the section called “服務端點”。

Connect 到 Amazon Keyspaces 使用 DataStax Python 驅動程序阿帕奇卡桑德拉和 SIGv4 身份驗證插件

下面的部分演示了如何使用 Sigv4 身份驗證插件的開源 DataStax Python 驅動程序阿帕奇卡桑德拉訪問 Amazon Keyspaces（阿帕奇卡桑德拉）。

如果您尚未這麼做，請依照中的步驟開始為您的 IAM 角色建立登入資料the section called “身分 AWS 驗證的 IAM 登入資”。本教學課程使用需要 IAM 角色的臨時登入資料。如需暫時登入資料的詳細資訊，請參閱the section called “使用臨時登入資料連接到 Amazon Keyspaces”。

然後，將 Python Sigv4 身份驗證插件從GitHub 存儲庫添加到您的環境中。

```
pip install cassandra-sigv4
```

下面的代碼示例演示了如何通過使用開源 DataStax Python 驅動程序卡桑德拉和 SIGv4 身份驗證插件連接到 Amazon Keyspaces。該插件取決 AWS SDK for Python (Boto3)。它用`boto3.session`來獲取臨時憑據。

```
from cassandra.cluster import Cluster
from ssl import SSLContext, PROTOCOL_TLSv1_2 , CERT_REQUIRED
from cassandra.auth import PlainTextAuthProvider
import boto3
from cassandra_sigv4.auth import SigV4AuthProvider
```

```
ssl_context = SSLContext(PROTOCOL_TLSv1_2)
ssl_context.load_verify_locations('path_to_file/sf-class2-root.crt')
ssl_context.verify_mode = CERT_REQUIRED

# use this if you want to use Boto to set the session parameters.
boto_session = boto3.Session(aws_access_key_id="AKIAIOSFODNN7EXAMPLE",
                             aws_secret_access_key="wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY",
                             aws_session_token="AQoDYXdzEJr...<remainder of token>",
                             region_name="us-east-2")
auth_provider = SigV4AuthProvider(boto_session)

# Use this instead of the above line if you want to use the Default Credentials and not
 bother with a session.
# auth_provider = SigV4AuthProvider()

cluster = Cluster(['cassandra.us-east-2.amazonaws.com'], ssl_context=ssl_context,
 auth_provider=auth_provider,
                  port=9142)
session = cluster.connect()
r = session.execute('select * from system_schema.keyspaces')
print(r.current_rows)
```

使用注意事項：

1. 取代"*path_to_file*/sf-class2-root.crt"為第一個步驟中儲存的憑證路徑。
2. *########aws_####aws_session_ token ##########.* Access Key Secret Access Key Session Token boto3.session 如需詳細資訊，請參閱中的認證AWS SDK for Python (Boto3)。
3. 如需可用端點的清單，請參閱the section called “服務端點”。

使用卡桑德拉 Node.js 客戶端驅動程序以編程方式訪問 Amazon Keyspaces

本節介紹如何使用 Node.js 客戶端驅動程序連接到 Amazon Keyspaces。若要提供使用者和應用程式以程式設計方式存取 Amazon Keyspace 資源的登入資料，您可以執行下列其中一項作業：

- 建立與特定 AWS Identity and Access Management (IAM) 使用者相關聯的服務特定登入資料。
- 為了增強安全性，我們建議您為 IAM 使用者或跨所有 AWS 服務使用的角色建立 IAM 存取金鑰。適用於 Cassandra 客戶端驅動程序的 Amazon 密 Keyspaces Sigv4 身份驗證插件使您可以使用 IAM

訪問密鑰而不是用戶名和密碼對 Amazon 密鑰 Keyspaces 進行身份驗證呼叫。如需詳細資訊，請參閱 the section called “身分 AWS 驗證的 IAM 登入資”。

主題

- 開始之前
- Connect 到 Amazon Keyspaces 使用 Node.js DataStax 驅動程序 Apache 卡桑德拉和特定服務的憑據
- 使用 DataStax Node.js 驅動程序的 Apache 卡桑德拉和 SIGv4 身份驗證插件 Connect 到 Amazon Keyspaces

開始之前

您必須先完成下列工作，才能開始。

Amazon Keyspaces 需要使用傳輸層安全性 (TLS) 來協助保護與用戶端的連線安全。要使用 TLS 連接到 Amazon Keyspaces，您需要下載 Amazon 數字證書並配置 Python 驅動程序以使用 TLS。

使用下列指令下載 Starfield 數位憑證，並儲存在`sf-class2-root.crt`本機或主目錄中。

```
curl https://certs.secureserver.net/repository/sf-class2-root.crt -O
```

> **Note**
>
> 您也可以使用 Amazon 數位憑證連線到 Amazon Keyspaces，如果您的用戶端成功連線至 Amazon Keyspaces，則可以繼續這麼做。Starfield 憑證為使用舊版憑證授權單位的用戶端提供額外的向後相容性。


```
curl https://certs.secureserver.net/repository/sf-class2-root.crt -O
```


Connect 到 Amazon Keyspaces 使用 Node.js DataStax 驅動程序 Apache 卡桑德拉和特定服務的憑據

將您的驅動程式設定為使用 Starfield 數位憑證進行 TLS，並使用服務特定憑證進行驗證。例如：

```
const cassandra = require('cassandra-driver');
const fs = require('fs');
```

```
const auth = new cassandra.auth.PlainTextAuthProvider('ServiceUserName',
 'ServicePassword');
const sslOptions1 = {
         ca: [
                    fs.readFileSync('path_to_file/sf-class2-root.crt', 'utf-8')],
                    host: 'cassandra.us-west-2.amazonaws.com',
                    rejectUnauthorized: true
        };
const client = new cassandra.Client({
                   contactPoints: ['cassandra.us-west-2.amazonaws.com'],
                   localDataCenter: 'us-west-2',
                   authProvider: auth,
                   sslOptions: sslOptions1,
                   protocolOptions: { port: 9142 }
        });
const query = 'SELECT * FROM system_schema.keyspaces';

client.execute(query)
                   .then( result => console.log('Row from Keyspaces %s',
 result.rows[0]))
                   .catch( e=> console.log(`${e}`));
```

使用注意事項：

1. 取代"*path_to_file*/sf-class2-root.crt"為第一個步驟中儲存的憑證路徑。
2. *####ServiceUser#########ServicePassword#########################* 產生服務特定認證
3. 如需可用端點的清單，請參閱the section called “服務端點”。

使用 DataStax Node.js 驅動程序的 Apache 卡桑德拉和 SIGv4 身份驗證插件 Connect 到 Amazon Keyspaces

下面的部分演示了如何使用 Sigv4 身份驗證插件的開源 DataStax Node.js 驅動程序阿帕奇卡桑德拉訪問 Amazon Keyspaces（阿帕奇卡桑德拉）。

如果您尚未這麼做，請依照中的步驟為您的 IAM 使用者或角色建立登入資料the section called “身分 AWS 驗證的 IAM 登入資”。

將 Node.js Sigv4 身份驗證插件添加到您的應用程序從GitHub 存儲庫中。該插件支持卡桑德拉的 DataStax Node.js 驅動程序的 4.x 版本，並取決於 Node.js 的 AWS SDK。它用`AWSCredentialsProvider`來獲取憑據。

```
$ npm install aws-sigv4-auth-cassandra-plugin --save
```

此程式碼範例顯示如何將的區域特定執行個體設定SigV4AuthProvider為驗證提供者。

```
const cassandra = require('cassandra-driver');
const fs = require('fs');
const sigV4 = require('aws-sigv4-auth-cassandra-plugin');

const auth = new sigV4.SigV4AuthProvider({
    region: 'us-west-2',
    accessKeyId:'AKIAIOSFODNN7EXAMPLE',
    secretAccessKey: 'wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY'});

const sslOptions1 = {
  ca: [
      fs.readFileSync('path_to_filecassandra/sf-class2-root.crt', 'utf-8')],
  host: 'cassandra.us-west-2.amazonaws.com',
  rejectUnauthorized: true
};


const client = new cassandra.Client({
  contactPoints: ['cassandra.us-west-2.amazonaws.com'],
  localDataCenter: 'us-west-2',
  authProvider: auth,
  sslOptions: sslOptions1,
  protocolOptions: { port: 9142 }
});


const query = 'SELECT * FROM system_schema.keyspaces';

client.execute(query).then(
    result => console.log('Row from Keyspaces %s', result.rows[0]))
    .catch( e=> console.log(`${e}`));
```

使用注意事項：

1. 取代"*path_to_file*/sf-class2-root.crt"為第一個步驟中儲存的憑證路徑。
2. 確保*##KeyId*和*#*碼與您使用取得的存取金鑰和秘密存取金鑰AccessKey相符AWSCredentialsProvider。如需詳細資訊，請參閱在 Node.js 的AWS SDK 中的 Node.js JavaScript 中設定認證。

3. 若要將存取金鑰儲存在程式碼之外，請參閱的最佳做法the section called “如何管理存取金鑰”。
4. 如需可用端點的清單，請參閱the section called “服務端點”。

使用卡桑德拉 .NET 核心客戶端驅動程序以編程方式訪問 Amazon Keyspaces

本節說明如何使用 .NET 核心用戶端驅動程式連接到 Amazon Keyspaces。設定步驟會根據您的環境和作業系統而有所不同，您可能必須進行相應的修改。Amazon Keyspaces 需要使用傳輸層安全性 (TLS) 來協助保護與用戶端的連線安全。若要使用 TLS 連線到 Amazon Keyspaces，您需要下載星空數位憑證，並將驅動程式設定為使用 TLS。

1. 下載 Starfield 證書並將其保存到本地目錄，並記下路徑。下面是一個例子使用 PowerShell.

```
$client = new-object System.Net.WebClient
$client.DownloadFile("https://certs.secureserver.net/repository/sf-class2-
root.crt","path_to_file\sf-class2-root.crt")
```

2. 安裝卡桑德拉克SharpDriver 通過核，使用核控制台。

```
PM> Install-Package CassandraCSharpDriver
```

3. 下列範例使用 .NET Core C# 主控台專案來連線到 Amazon Keyspaces 並執行查詢。

```
using Cassandra;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net.Security;
using System.Runtime.ConstrainedExecution;
using System.Security.Cryptography.X509Certificates;
using System.Text;
using System.Threading.Tasks;

namespace CSharpKeyspacesExample
{
    class Program
    {
        public Program(){}

        static void Main(string[] args)
        {
```

```
            X509Certificate2Collection certCollection = new
 X509Certificate2Collection();
            X509Certificate2 amazoncert = new X509Certificate2(@"path_to_file\sf-
class2-root.crt");
            var userName = "ServiceUserName";
            var pwd = "ServicePassword";
            certCollection.Add(amazoncert);

            var awsEndpoint =  "cassandra.us-east-2.amazonaws.com" ;

            var cluster = Cluster.Builder()
                     .AddContactPoints(awsEndpoint)
                     .WithPort(9142)
                     .WithAuthProvider(new PlainTextAuthProvider(userName, pwd))
                     .WithSSL(new
 SSLOptions().SetCertificateCollection(certCollection))
                     .Build();

            var session = cluster.Connect();
            var rs = session.Execute("SELECT * FROM system_schema.tables;");
            foreach (var row in rs)
            {
                var name = row.GetValue<String>("keyspace_name");
                Console.WriteLine(name);
            }
        }
    }
}
```

使用注意事項：

a. 取代"*path_to_file*/sf-class2-root.crt"為第一個步驟中儲存的憑證路徑。

b. *####ServiceUser#########ServicePassword#########################* 產生服務特定認證

c. 如需可用端點的清單，請參閱the section called “服務端點”。

使用 Cassandra Go 客戶端驅動程序以編程方式訪問 Amazon Keyspaces

本節介紹如何使用 Go 客戶端驅動程序連接到 Amazon Keyspaces。若要提供使用者和應用程式以程式設計方式存取 Amazon Keyspace 資源的登入資料，您可以執行下列其中一項作業：

- 建立與特定 AWS Identity and Access Management (IAM) 使用者相關聯的服務特定登入資料。

- 為了增強安全性，我們建議您為 IAM 使用者和所有 AWS 服務中使用的角色建立 IAM 存取金鑰。適用於 Cassandra 客戶端驅動程序的 Amazon 密 Keyspaces Sigv4 身份驗證插件使您可以使用 IAM 訪問密鑰而不是用戶名和密碼對 Amazon 密鑰 Keyspaces 進行身份驗證呼叫。如需詳細資訊，請參閱 the section called “身分 AWS 驗證的 IAM 登入資”。

主題

開始之前

您必須先完成下列工作，才能開始。

Amazon Keyspaces 需要使用傳輸層安全性 (TLS) 來協助保護與用戶端的連線安全。要使用 TLS 連接到 Amazon Keyspaces，您需要下載 Amazon 數字證書並配置 Python 驅動程序以使用 TLS。

使用下列指令下載 Starfield 數位憑證，並儲存在`sf-class2-root.crt`本機或主目錄中。

```
curl https://certs.secureserver.net/repository/sf-class2-root.crt -O
```

> **Note**
>
> 您也可以使用 Amazon 數位憑證連線到 Amazon Keyspaces，如果您的用戶端成功連線至 Amazon Keyspaces，則可以繼續這麼做。Starfield 憑證為使用舊版憑證授權單位的用戶端提供額外的向後相容性。


```
curl https://certs.secureserver.net/repository/sf-class2-root.crt -O
```


使用適用於 Apache 卡桑德拉和服務特定憑據的 Gocql 驅動程序 Connect 到 Amazon Keyspaces

1. 為您的應用程式建立目錄。

   ```
   mkdir ./gocqlexample
   ```

2. 導覽至新目錄。

```
cd gocqlexample
```

3. 為您的應用程式建立檔案。

```
touch cqlapp.go
```

4. 下載 Go 驅動程式。

```
go get github.com/gocql/gocql
```

5. 將下列範例程式碼新增至 cqlapp.go 檔案。

```
package main

import (
     "fmt"
     "github.com/gocql/gocql"
     "log"
)

func main() {

    // add the Amazon Keyspaces service endpoint
    cluster := gocql.NewCluster("cassandra.us-east-2.amazonaws.com")
    cluster.Port=9142
    // add your service specific credentials
    cluster.Authenticator = gocql.PasswordAuthenticator{
            Username: "ServiceUserName",
            Password: "ServicePassword"}
    // provide the path to the sf-class2-root.crt
    cluster.SslOpts = &gocql.SslOptions{
            CaPath: "path_to_file/sf-class2-root.crt",
            EnableHostVerification: false,
     }

     // Override default Consistency to LocalQuorum
     cluster.Consistency = gocql.LocalQuorum
     cluster.DisableInitialHostLookup = false

     session, err := cluster.CreateSession()
     if err != nil {
            fmt.Println("err>", err)
```

```
    }
    defer session.Close()

    // run a sample query from the system keyspace
    var text string
    iter := session.Query("SELECT keyspace_name FROM system_schema.tables;").Iter()
    for iter.Scan(&text) {
        fmt.Println("keyspace_name:", text)
    }
    if err := iter.Close(); err != nil {
        log.Fatal(err)
    }
    session.Close()
}
```

使用注意事項：

a. 取代"*path_to_file*/sf-class2-root.crt"為第一個步驟中儲存的憑證路徑。

b. *####ServiceUser#########ServicePassword#########################* 產生服務特定認證

c. 如需可用端點的清單，請參閱the section called “服務端點”。

6. 建置程式。

```
go build cqlapp.go
```

7. 執行程式。

```
./cqlapp
```

使用 Go 驅動程序的 Apache 卡桑德拉和 SIGv4 身份驗證插件 Connect 到 Amazon Keyspaces

下面的代碼示例演示了如何使用 Sigv4 身份驗證插件的開源 Go 驅動程序訪問 Amazon Keyspaces（適用於 Apache 卡桑德拉）。

如果您尚未這麼做，請依照中的步驟為您的 IAM 使用者或角色建立登入資料the section called “身分 AWS 驗證的 IAM 登入資”。

從GitHub存儲庫將 Go Sigv4 身份驗證插件添加到您的應用程序。該插件支持開源 Go 驅動程序卡桑德拉的 1.2.x 版本，並取決 SDK for Go。 AWS

```
$ go mod init
$ go get github.com/aws/aws-sigv4-auth-cassandra-gocql-driver-plugin
```

在此程式碼範例中，Amazon Keyspaces 端點由Cluster類別表示。它會使AwsAuthenticator用叢集的驗證器屬性來取得認證。

```
package main

import (
        "fmt"
        "github.com/aws/aws-sigv4-auth-cassandra-gocql-driver-plugin/sigv4"
        "github.com/gocql/gocql"
        "log"
)

func main() {
    // configuring the cluster options
    cluster := gocql.NewCluster("cassandra.us-west-2.amazonaws.com")
    cluster.Port=9142
    var auth sigv4.AwsAuthenticator = sigv4.NewAwsAuthenticator()
    auth.Region = "us-west-2"
    auth.AccessKeyId = "AKIAIOSFODNN7EXAMPLE"
    auth.SecretAccessKey = "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"

    cluster.Authenticator = auth

    cluster.SslOpts = &gocql.SslOptions{
            CaPath: "path_to_file/sf-class2-root.crt",
            EnableHostVerification: false,
    }
    cluster.Consistency = gocql.LocalQuorum
    cluster.DisableInitialHostLookup = false

    session, err := cluster.CreateSession()
    if err != nil {
     fmt.Println("err>", err)
     return
    }
    defer session.Close()

    // doing the query
    var text string
    iter := session.Query("SELECT keyspace_name FROM system_schema.tables;").Iter()
```

```
    for iter.Scan(&text) {
     fmt.Println("keyspace_name:", text)
    }
    if err := iter.Close(); err != nil {
     log.Fatal(err)
    }
}
```

使用注意事項：

1. 取代"*path_to_file*/sf-class2-root.crt"為第一個步驟中儲存的憑證路徑。
2. 確保 *AccessKeyID* 和密*SecretAccess#*與您使用獲得的訪問密鑰和秘密訪問密鑰匹配AwsAuthenticator。如需詳細資訊，請參閱在 AWS SDK for Go. AWS
3. 若要將存取金鑰儲存在程式碼之外，請參閱的最佳做法the section called “如何管理存取金鑰”。
4. 如需可用端點的清單，請參閱the section called “服務端點”。

使用卡桑德拉 Perl 客戶端驅動程序以編程方式訪問 Amazon Keyspaces

本節介紹如何通過使用 Perl 客戶端驅動程序連接到 Amazon Keyspaces。對於此代碼示例，我們使用 Perl 5。Amazon Keyspaces 需要使用傳輸層安全性 (TLS) 來協助保護與用戶端的連線安全。

> **Important**
>
> 為了建立安全連線，我們的程式碼範例使用 Starfield 數位憑證來驗證伺服器，然後再建立 TLS 連線。Perl 驅動程序不驗證服務器的 Amazon SSL 證書，這意味著您無法確認您正在連接到 Amazon Keyspaces。第二個步驟是設定驅動程式以在連線到 Amazon Keyspaces space 時使用 TLS，並確保用戶端和伺服器之間傳輸的資料已加密。

1. 從下載卡桑德拉 DBI 驅動程序https://metacpan.org/pod/DBD::Cassandra並將驅動程序安裝到您的 Perl 環境中。確切的步驟取決於環境。以下是一個常見的例子。

   ```
   cpanm DBD::Cassandra
   ```

2. 為您的應用程式建立檔案。

   ```
   touch cqlapp.pl
   ```

3. 將下列範例程式碼新增至 cqlapp.pl 檔案。

```
use DBI;
my $user = "ServiceUserName";
my $password = "ServicePassword";
my $db = DBI->connect("dbi:Cassandra:host=cassandra.us-
east-2.amazonaws.com;port=9142;tls=1;",
$user, $password);

my $rows = $db->selectall_arrayref("select * from system_schema.keyspaces");
print "Found the following Keyspaces...\n";
for my $row (@$rows) {
      print join(" ",@$row['keyspace_name']),"\n";
}

$db->disconnect;
```

Important

####ServiceUser#########ServicePassword######################### 產生服務特定認證

Note

如需可用端點的清單，請參閱the section called “服務端點”。

4. 執行應用程式。

```
perl cqlapp.pl
```

教程：從 Amazon 彈性 Kubernetes 服務連接到亞馬遜 Keyspaces

本教學將引導您完成設定 Amazon 彈性 Kubernetes 服務 (Amazon EKS) 叢集以託管使用 SIGv4 身份驗證連接至 Amazon Keyspaces 的容器化應用程式所需的步驟。

Amazon EKS 是一項受管服務，無需安裝、操作和維護自己的 Kubernetes 控制平面。Kubernetes 是一套開放原始碼系統，可將容器化應用程式的管理、擴展和部署工作自動化。

本教學提 step-by-step 供設定、建置容器化 Java 應用程式並將其部署到 Amazon EKS 的指導。在最後一個步驟中，您可以執行應用程式，將資料寫入 Amazon Keyspaces 資料表。

主題

教學課程事前準備

在開始使用教學課程之前，請先建立下列 AWS 資源

1. 開始本自學課程之前，請遵循中的 AWS 設定指示訪問 Amazon Keyspaces（阿帕奇卡桑德拉）。這些步驟包括註冊 AWS 和建立可存取 Amazon 金 Keyspaces 的 AWS Identity and Access Management (IAM) 主體。
2. 建立具有名稱的 Amazon 金 Keyspaces 間金鑰空間，aws並建立名稱的表格，您可以從本教學稍後在 Amazon EKS 中執行的容器化應用程式寫入user該資料表。您可以使用 AWS CLI 或使用來執行此操作cqlsh。

   AWS CLI

   ```
   aws keyspaces create-keyspace --keyspace-name 'aws'
   ```

   要確認創建密鑰空間，可以使用以下命令。

   ```
   aws keyspaces list-keyspaces
   ```

   要創建表，您可以使用下面的命令。

   ```
   aws keyspaces create-table --keyspace-name 'aws' --table-name 'user' --schema-
   definition 'allColumns=[
               {name=username,type=text}, {name=fname,type=text},
   {name=last_update_date,type=timestamp},{name=lname,type=text}],
   ```

```
            partitionKeys=[{name=username}]'
```

要確認您的表已創建，您可以使用以下命令。

```
aws keyspaces list-tables --keyspace-name 'aws'
```

如需詳細資訊，請參閱《AWS CLI 命令參考》中的建立金鑰空間和建立資料表。

cqlsh

```
CREATE KEYSPACE aws WITH replication = {'class': 'SimpleStrategy',
 'replication_factor': '3'}  AND durable_writes = true;
CREATE TABLE aws.user (
    username text PRIMARY KEY,
    fname text,
    last_update_date timestamp,
    lname text
);
```

要驗證您的表是否已創建，您可以使用下面的語句。

```
SELECT * FROM system_schema.tables WHERE keyspace_name = "aws";
```

您的表應該在此語句的輸出中列出。請注意，建立資料表之前可能會有延遲。如需詳細資訊，請參閱 the section called “CREATE TABLE”。

3. 使用 Fargate-Linux 節點類型建立 Amazon EKS 叢集。Fargate 是一種無伺服器運算引擎，可讓您部署 Kubernetes 網繭，而無需管理 Amazon EC2 執行個體。若要遵循本教學課程而不需更新所有範例命令中的叢集名稱，請依my-eks-cluster照 Amazon EKS 使用者指南中〈Amazon EKS 入門〉eksctl中的指示建立名稱的叢集。建立叢集時，請確認您的節點和兩個預設 Pod 正在執行且狀態良好。您可以使用以下命令執行此操作。

```
kubectl get pods -A -o wide
```

您應該看到類似於此輸出的內容。

```
NAMESPACE     NAME                       READY   STATUS    RESTARTS   AGE   IP
     NODE                                                 NOMINATED NODE   READINESS
 GATES
```

```
kube-system   coredns-1234567890-abcde   1/1     Running   0          18m
 192.0.2.0   fargate-ip-192-0-2-0.region-code.compute.internal   <none>
 <none>
kube-system   coredns-1234567890-12345   1/1     Running   0          18m
 192.0.2.1   fargate-ip-192-0-2-1.region-code.compute.internal   <none>
 <none>
```

4. 安裝 Docker. 如需如何在 Amazon EC2 執行個體上安裝 Docker 的指示，請參閱 Amazon 彈性容器登錄使用者指南中的安裝 Docker。

   Docker 可在多個不同的作業系統上使用，包括大部分的現代 Linux 發行版本，例如 Ubuntu，甚至是 macOS 和 Windows。如需如何在特定作業系統上安裝 Docker 的詳細資訊，請前往「Docker 安裝指南」。

5. 建立 Amazon ECR 儲存庫。Amazon ECR 是一種 AWS 受管容器映像登錄服務，您可以將其與偏好的 CLI 搭配使用，以推送、提取和管理 Docker 映像。如需 Amazon ECR 儲存庫的詳細資訊，請參閱 Amazon 彈性容器登錄使用者指南。您可以使用以下命令來創建名稱的存儲庫`my-ecr-repository`。

   ```
   aws ecr create-repository --repository-name my-ecr-repository
   ```

完成先決條件步驟後，請繼續執行the section called “步驟 1：設定 Amazon EKS 叢集”。

步驟 1：設定 Amazon EKS 叢集並設定 IAM 許可

設定 Amazon EKS 叢集並建立所需的 IAM 資源，以允許 Amazon EKS 服務帳戶連接到您的 Amazon Keyspaces 表格

1. 為 Amazon EKS 叢集建立開放式 ID Connect (OIDC) 提供者。若要針對服務帳戶使用 IAM 角色，則需要執行此動作。如需有關 OIDC 提供者及如何建立它們的詳細資訊，請參閱 Amazon EKS 使用者指南中的為叢集建立 IAM OIDC 提供者。

   a. 使用下列命令為您的叢集建立 IAM OIDC 身分提供者。此範例假設您的叢集名稱為`my-eks-cluster`。如果您有不同名稱的叢集，請記得在以 future 的所有命令中更新名稱。

      ```
      eksctl utils associate-iam-oidc-provider --cluster my-eks-cluster --approve
      ```

   b. 使用以下命令確認 OIDC 身分提供者已向 IAM 註冊。

      ```
      aws iam list-open-id-connect-providers --region aws-region
      ```

輸出看起來應該類似於這個。記下 OIDC 的 Amazon 資源名稱 (ARN)，當您為服務帳戶建立信任政策時，需要在下一個步驟中使用它。

```
{
    "OpenIDConnectProviderList": [
        ..
        {
            "Arn": "arn:aws:iam::111122223333:oidc-provider/oidc.eks.aws-
region.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
        }
    ]
}
```

2. 為 Amazon EKS 叢集建立服務帳戶。服務帳戶為在網繭中執行的處理程序提供識別。Pod 是您可用來部署容器化應用程式的最小且最簡單的 Kubernetes 物件。接下來，建立服務帳戶可以假設為取得資源許可的 IAM 角色。您可以從已設定為使用可承擔具有該服務存取權限的 IAM 角色的服務帳戶存取任 AWS 何服務。

   a. 為服務帳戶建立新的命名空間。命名空間有助於隔離針對本教學課程建立的叢集資源。您可以使用以下命令創建一個新的命名空間。

   ```
   kubectl create namespace my-eks-namespace
   ```

   b. 要使用自定義命名空間，您必須將其與 Fargate 配置文件相關聯。下面的代碼是這樣的一個例子。

   ```
   eksctl create fargateprofile \
       --cluster my-eks-cluster \
       --name my-fargate-profile \
       --namespace my-eks-namespace \
       --labels *=*
   ```

   c. 使用下列命令`my-eks-serviceaccount`在 Amazon EKS 叢集的命名空`my-eks-namespace`間中建立名稱的服務帳戶。

   ```
   cat >my-serviceaccount.yaml <<EOF
   apiVersion: v1
   kind: ServiceAccount
   metadata:
     name: my-eks-serviceaccount
     namespace: my-eks-namespace
   ```

```
EOF
kubectl apply -f my-serviceaccount.yaml
```

d. 執行下列命令以建立信任政策檔案，以指示 IAM 角色信任您的服務帳戶。主參與者可以擔任角色之前，需要此信任關係。您需要對文件進行以下編輯：

- 對於`Principal`，請輸入 IAM 傳回給`list-open-id-connect-providers`命令的 ARN。ARN 包含您的帳號和地區。
- 在`condition`陳述式中，取代 AWS 區域 和 OIDC 識別碼。
- 確認服務帳戶名稱和命名空間是否正確。

建立 IAM 角色時，您需要在下一個步驟中附加信任政策檔案。

```
cat >trust-relationship.json <<EOF
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.aws-region.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.aws-region.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:my-eks-
namespace:my-eks-serviceaccount",
                    "oidc.eks.aws-region.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
                }
            }
        }
    ]
}
EOF
```

可選：您也可以在`StringEquals`或`StringLike`條件中新增多個項目，以允許多個服務帳戶或命名空間擔任該角色。若要允許您的服務帳戶在不同 AWS 帳戶中擔任 IAM 角色，請參閱 Amazon EKS 使用者指南中的跨帳戶 IAM 許可。

3. 使用要假設的 Amazon EKS 服務帳戶名稱`my-iam-role`建立 IAM 角色。將在最後一個步驟中建立的信任原則檔案附加至角色。信任政策會指定 IAM 角色可信任的服務帳戶和 OIDC 提供者。

```
aws iam create-role --role-name my-iam-role --assume-role-policy-document file://
trust-relationship.json --description "EKS service account role"
```

4. 透過附加存取政策，將 IAM 角色許可指派給 Amazon Keyspaces。

   a. 附加存取政策以定義 IAM 角色可在特定 Amazon Keyspaces 資源上執行的動作。在本教程中，我們使用 AWS 受管政策`AmazonKeyspacesFullAccess`，因為我們的應用程序將數據寫入您的 Amazon Keyspaces 表。不過，最佳作法是建議您建立實作最低權限原則的自訂存取原則。如需詳細資訊，請參閱 the section called “Amazon Keyspaces 如何與 IAM 一起工作”。

```
aws iam attach-role-policy --role-name my-iam-role --policy-
arn=arn:aws:iam::aws:policy/AmazonKeyspacesFullAccess
```

   使用下列陳述式確認政策已成功附加至 IAM 角色。

```
aws iam list-attached-role-policies --role-name my-iam-role
```

   輸出應該是這樣的。

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "AmazonKeyspacesFullAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AmazonKeyspacesFullAccess"
        }
    ]
}
```

   b. 使用可承擔的 IAM 角色的 Amazon 資源名稱 (ARN) 為服務帳戶加上註解。確保使用您的帳戶 ID 更新角色 ARN。

```
kubectl annotate serviceaccount -n my-eks-namespace my-eks-serviceaccount
 eks.amazonaws.com/role-arn=arn:aws:iam::111122223333:role/my-iam-role
```

5. 確認 IAM 角色和服務帳戶設定正確。

a. 使用下列陳述式確認 IAM 角色的信任政策已正確設定。

```
aws iam get-role --role-name my-iam-role --query Role.AssumeRolePolicyDocument
```

輸出看起來應該類似於這個。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.aws-region.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "oidc.eks.aws-region/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
                    "oidc.eks.aws-region.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:my-eks-
namespace:my-eks-serviceaccount"
                }
            }
        }
    ]
}
```

b. 確認已使用 IAM 角色註釋 Amazon EKS 服務帳戶。

```
kubectl describe serviceaccount my-eks-serviceaccount -n my-eks-namespace
```

輸出看起來應該類似於這個。

```
Name: my-eks-serviceaccount
Namespace:my-eks-namespace
Labels: <none>
Annotations: eks.amazonaws.com/role-arn: arn:aws:iam::111122223333:role/my-iam-
role
Image pull secrets: <none>
```

```
Mountable secrets: <none>
Tokens: <none>
[...]
```

建立 Amazon EKS 服務帳戶、IAM 角色並設定必要的關係和許可後，請繼續執行the section called “步驟 2：設定應用程式”。

步驟 2：設定應用程式

在此步驟中，您可以使用 Sigv4 外掛程式建立連接到 Amazon Keyspaces 的應用程式。您可以從 Github 上的 Amazon Keyspaces 示例代碼存儲庫查看和下載示例 Java 應用程序。或者，您可以使用自己的應用程序進行操作，確保完成所有配置步驟。

設定您的應用程式並新增必要的相依性。

1. 您可以通過使用以下命令克隆 Github 存儲庫下載示例 Java 應用程序。

```
git clone https://github.com/aws-samples/amazon-keyspaces-examples.git
```

2. 下載 Github 倉庫後，解壓縮下載的文件並導航到該`application.conf`文件的`resources`目錄。

   a. 應用程式組

      在此步驟中，您可以設定 SIGv4 驗證外掛程式。您可以在您的應用程序中使用以下示例。如果您尚未這麼做，則需要產生 IAM 存取金鑰 (存取金鑰 ID 和秘密存取金鑰)，並將它們儲存在 AWS 設定檔或環境變數中。如需詳細說明，請參閱 the section called “驗證所需的認 AWS 證”。視需要更新 Amazon Keyspaces 的 AWS 區域和服務端點。如需更多服務端點，請參閱the section called “服務端點”。將信任庫位置、信任庫名稱和信任庫密碼取代為您自己的密碼。

```
datastax-java-driver {
  basic.contact-points = ["cassandra.aws-region.amazonaws.com:9142"]
  basic.load-balancing-policy.local-datacenter = "aws-region"
  advanced.auth-provider {
    class = software.aws.mcs.auth.SigV4AuthProvider
    aws-region = "aws-region"
  }
  advanced.ssl-engine-factory {
    class = DefaultSslEngineFactory
```

```
    truststore-path = "truststore_locationtruststore_name.jks"
    truststore-password = "truststore_password;"
  }
}
```

b. 添加 STS 模塊依賴關係。

這增加了使用返回應用WebIdentityTokenCredentialsProvider程序需要提供的 AWS 憑據的功能，以便服務帳戶可以擔任 IAM 角色。您可以根據以下示例執行此操作。

```
        <dependency>
            <groupId>com.amazonaws</groupId>
            <artifactId>aws-java-sdk-sts</artifactId>
            <version>1.11.717</version>
        </dependency>
```

c. 新增 Sigv4 相依性。

該軟件包實現了對 Amazon Keyspaces 進行身份驗證所需的 Sigv4 身份驗證插件

```
        <dependency>
            <groupId>software.aws.mcs</groupId>
            <artifactId>aws-sigv4-auth-cassandra-java-driver-plugin</
artifactId>
            <version>4.0.3</version>
        </dependency>
```

3. 新增記錄相依性。

如果沒有記錄檔，就無法進行連線問題疑難 在本教程中，我們使slf4j用日誌記錄框架，並用logback.xml於存儲日誌輸出。我們將記錄級別設置debug為以建立連接。您可以使用下列範例來新增相依性。

```
      <dependency>
          <groupId>org.slf4j</groupId>
          <artifactId>slf4j-api</artifactId>
          <version>2.0.5</version>
      </dependency>
```

您可以使用下面的代碼片段來配置日誌記錄。

```
<configuration>
```

```
    <appender name="STDOUT" class="ch.qos.logback.core.ConsoleAppender">

        <encoder>
            <pattern>%d{HH:mm:ss.SSS} [%thread] %-5level %logger{36} - %msg%n</
pattern>
        </encoder>
    </appender>

    <root level="debug">
        <appender-ref ref="STDOUT" />
    </rootv
</configuration>
```

> **Note**
>
> 調查連線失敗所需的debug層級。從應用程式成功連線到 Amazon Keyspaces 之後，您可以視需要將記錄層級變更warning為info或。

步驟 3：建立應用程式映像檔，並將 Docker 檔案上傳到您的 Amazon ECR 儲存庫

在此步驟中，您要編譯範例應用程式、建立 Docker 映像，然後將映像推送到 Amazon ECR 儲存庫。

建置您的應用程式、建立 Docker 映像，然後將其提交至 Amazon 彈性容器登錄

1. 為定義 AWS 區域. 用您自己的區域替換示例中的區域。

```
export CASSANDRA_HOST=cassandra.aws-region.amazonaws.com:9142
export CASSANDRA_DC=aws-region
```

2. 使用下面的命令編譯與阿帕奇 Maven 3.6.3 版本或更高版本的應用程序。

```
mvn clean install
```

這將創建一個包含在target目錄中的所有依賴關係的JAR文件。

3. 使用下列指令擷取下一個步驟所需的 ECR 存放庫 URI。請務必將區域更新為您一直使用的區域。

```
aws ecr describe-repositories --region aws-region
```

輸出應如下列範例所示。

```
"repositories": [
 {
 "repositoryArn": "arn:aws:ecr:aws-region:111122223333:repository/my-ecr-
repository",
 "registryId": "111122223333",
 "repositoryName": "my-ecr-repository",
 "repositoryUri": "111122223333.dkr.ecr.aws-region.amazonaws.com/my-ecr-
repository",
 "createdAt": "2023-11-02T03:46:34+00:00",
 "imageTagMutability": "MUTABLE",
 "imageScanningConfiguration": {
 "scanOnPush": false
 },
 "encryptionConfiguration": {
 "encryptionType": "AES256"
 }
 },
```

4. 從應用程式的根目錄中，使用上一個步驟的儲存庫 URI 建立 Docker 映像檔。視需要修改泊塢視窗檔案。在建置命令中，請務必取代您的帳戶 ID，並將其設定 AWS 區域 為 Amazon ECR 儲存庫所在my-ecr-repository的區域。

```
docker build -t 111122223333.dkr.ecr.aws-region.amazonaws.com/my-ecr-
repository:latest .
```

5. 擷取身份驗證權杖以將 Docker 映像推送至 Amazon ECR。您可以使用以下命令執行此操作。

```
aws ecr get-login-password --region aws-region | docker login --username AWS --
password-stdin 111122223333.dkr.ecr.aws-region.amazonaws.com
```

6. 首先，請檢查 Amazon ECR 儲存庫中的現有映像檔。您可以使用下列命令。

```
aws ecr describe-images --repository-name my-ecr-repository --region aws-region
```

然後，將 Docker 映像推送到回購。您可以使用下列命令。

```
docker push 111122223333.dkr.ecr.aws-region.amazonaws.com/my-ecr-repository:latest
```

步驟 4：將應用程序部署到 Amazon EKS 並將數據寫入 Amazon Keyspaces 表

在本教學的這個步驟中，您要為應用程式設定 Amazon EKS 部署，並確認應用程式正在執行且可以連線到 Amazon Keyspaces。

若要將應用程式部署到 Amazon EKS，您需要在名為deployment.yaml的檔案中設定所有相關設定。然後 Amazon EKS 會使用此檔案來部署應用程式。檔案中的中繼資料應包含下列資訊：

- 應用程式名稱應用程式的名稱。在本教程中，我們使用my-keyspaces-app.
- Kubernetes 命名空間是 Amazon EKS 叢集的命名空間。在本教程中，我們使用my-eks-namespace.
- Amazon EKS 服務帳戶名稱 Amazon EKS 服務帳戶的名稱。在本教程中，我們使用my-eks-serviceaccount.
- 影像名稱應用程式影像的名稱。在本教程中，我們使用my-keyspaces-app.
- 圖像 URI 來自 Amazon ECR 的碼頭圖像 URI。
- AWS 帳戶 ID 您的 AWS 帳戶 ID。
- IAM 角色 ARN 是為服務帳戶建立的 IAM 角色所建立的 ARN。在本教程中，我們使用my-iam-role.
- AWS 區域 您在其中創建了 Amazon EKS 集群的 Amazon EKS 集群。 AWS 區域

在此步驟中，您會部署並執行連線到 Amazon Keyspaces 並將資料寫入資料表的應用程式。

1. 設定 deployment.yaml 檔案。您需要替換以下值：

   - name
   - namespace
   - serviceAccountName
   - image
   - AWS_ROLE_ARN value
   - AWS 區域 在 CASSANDRA_HOST
   - AWS_REGION

   您可以使用下列檔案做為範例。

```
apiVersion: apps/v1
```

```
kind: Deployment
metadata:
  name: my-keyspaces-app
  namespace: my-eks-namespace
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-keyspaces-app
  template:
    metadata:
      labels:
        app: my-keyspaces-app
    spec:
      serviceAccountName: my-eks-serviceaccount
      containers:
      - name: my-keyspaces-app
        image: 111122223333.dkr.ecr.aws-region.amazonaws.com/my-ecr-
repository:latest
        ports:
        - containerPort: 8080
        env:
        - name: CASSANDRA_HOST
          value: "cassandra.aws-region.amazonaws.com:9142"
        - name: CASSANDRA_DC
          value: "aws-region"
        - name: AWS_WEB_IDENTITY_TOKEN_FILE
          value: /var/run/secrets/eks.amazonaws.com/serviceaccount/token
        - name: AWS_ROLE_ARN
          value: "arn:aws:iam::111122223333:role/my-iam-role"
        - name: AWS_REGION
          value: "aws-region"
```

2. 部署 deployment.yaml。

```
kubectl apply -f deployment.yaml
```

輸出應該是這樣的。

```
deployment.apps/my-keyspaces-app created
```

3. 檢查 Amazon EKS 叢集命名空間中網繭的狀態。

```
kubectl get pods -n my-eks-namespace
```

輸出看起來應該類似於這個例子。

```
NAME                    READY STATUS RESTARTS AGE
my-keyspaces-app-123abcde4f-g5hij 1/1 Running 0 75s
```

有關更多詳細信息，您可以使用以下命令。

```
kubectl describe pod my-keyspaces-app-123abcde4f-g5hij -n my-eks-namespace
```

```
Name:                 my-keyspaces-app-123abcde4f-g5hij
Namespace:            my-eks-namespace
Priority:             2000001000
Priority Class Name:  system-node-critical
Service Account:      my-eks-serviceaccount
Node:                 fargate-ip-192-168-102-209.ec2.internal/192.168.102.209
Start Time:           Thu, 23 Nov 2023 12:15:43 +0000
Labels:               app=my-keyspaces-app
                      eks.amazonaws.com/fargate-profile=my-fargate-profile
                      pod-template-hash=6c56fccc56
Annotations:          CapacityProvisioned: 0.25vCPU 0.5GB
                      Logging: LoggingDisabled: LOGGING_CONFIGMAP_NOT_FOUND
Status:               Running
IP:                   192.168.102.209
IPs:
  IP:           192.168.102.209
Controlled By:  ReplicaSet/my-keyspaces-app-6c56fccc56
Containers:
  my-keyspaces-app:
    Container ID:
 containerd://41ff7811d33ae4bc398755800abcdc132335d51d74f218ba81da0700a6f8c67b
    Image:          111122223333.dkr.ecr.aws-region.amazonaws.com/
my_eks_repository:latest
    Image ID:       111122223333.dkr.ecr.aws-region.amazonaws.com/
my_eks_repository@sha256:fd3c6430fc5251661efce99741c72c1b4b03061474940200d0524b84a951439c
    Port:           8080/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Thu, 23 Nov 2023 12:15:19 +0000
```

```
      Finished:     Thu, 23 Nov 2023 12:16:17 +0000
    Ready:          True
    Restart Count:  1
    Environment:
      CASSANDRA_HOST:               cassandra.aws-region.amazonaws.com:9142
      CASSANDRA_DC:                 aws-region
      AWS_WEB_IDENTITY_TOKEN_FILE:  /var/run/secrets/eks.amazonaws.com/
serviceaccount/token
      AWS_ROLE_ARN:                 arn:aws:iam::111122223333:role/my-iam-role
      AWS_REGION:                   aws-region
      AWS_STS_REGIONAL_ENDPOINTS:   regional
    Mounts:
      /var/run/secrets/eks.amazonaws.com/serviceaccount from aws-iam-token (ro)
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-fssbf (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  aws-iam-token:
    Type:                     Projected (a volume that contains injected data from
 multiple sources)
    TokenExpirationSeconds:  86400
  kube-api-access-fssbf:
    Type:                     Projected (a volume that contains injected data from
 multiple sources)
    TokenExpirationSeconds:  3607
    ConfigMapName:           kube-root-ca.crt
    ConfigMapOptional:       <nil>
    DownwardAPI:             true
QoS Class:                   BestEffort
Node-Selectors:              <none>
Tolerations:                 node.kubernetes.io/not-ready:NoExecute op=Exists for
 300s
                             node.kubernetes.io/unreachable:NoExecute op=Exists for
 300s
Events:
  Type     Reason           Age                From               Message
  ----     ------           ----               ----               -------
  Warning  LoggingDisabled  2m13s              fargate-scheduler  Disabled logging
 because aws-logging configmap was not found. configmap "aws-logging" not found
```

```
  Normal   Scheduled        89s                fargate-scheduler  Successfully
 assigned my-eks-namespace/my-keyspaces-app-6c56fccc56-mgs2m to fargate-
ip-192-168-102-209.ec2.internal
  Normal   Pulled           75s                kubelet
 Successfully pulled image "111122223333.dkr.ecr.aws-region.amazonaws.com/
my_eks_repository:latest" in 13.027s (13.027s including waiting)
  Normal   Pulling          54s (x2 over 88s)  kubelet            Pulling image
 "111122223333.dkr.ecr.aws-region.amazonaws.com/my_eks_repository:latest"
  Normal   Created          54s (x2 over 75s)  kubelet            Created container
 my-keyspaces-app
  Normal   Pulled           54s                kubelet
 Successfully pulled image "111122223333.dkr.ecr.aws-region.amazonaws.com/
my_eks_repository:latest" in 222ms (222ms including waiting)
  Normal   Started          53s (x2 over 75s)  kubelet            Started container
 my-keyspaces-app
```

4. 檢查 Pod 的日誌以確認您的應用程式正在執行，並且可以連線到 Amazon Keyspaces 表格。您可以使用以下命令執行此操作。請務必取代部署的名稱。

```
kubectl logs -f my-keyspaces-app-123abcde4f-g5hij -n my-eks-namespace
```

您應該能夠看到應用程式記錄項目，確認與 Amazon Keyspaces 的連線，如下列範例所示。

```
2:47:20.553 [s0-admin-0] DEBUG c.d.o.d.i.c.metadata.MetadataManager
 - [s0] Adding initial contact points [Node(endPoint=cassandra.aws-
region.amazonaws.com/1.222.333.44:9142, hostId=null, hashCode=e750d92)]
22:47:20.562 [s0-admin-1] DEBUG c.d.o.d.i.c.c.ControlConnection - [s0] Initializing
 with event types [SCHEMA_CHANGE, STATUS_CHANGE, TOPOLOGY_CHANGE]
22:47:20.564 [s0-admin-1] DEBUG c.d.o.d.i.core.context.EventBus - [s0] Registering
 com.datastax.oss.driver.internal.core.metadata.LoadBalancingPolicyWrapper$$Lambda
$812/0x0000000801105e88@769afb95 for class
 com.datastax.oss.driver.internal.core.metadata.NodeStateEvent
22:47:20.566 [s0-admin-1] DEBUG c.d.o.d.i.c.c.ControlConnection -
 [s0] Trying to establish a connection to Node(endPoint=cassandra.us-
east-1.amazonaws.com/1.222.333.44:9142, hostId=null, hashCode=e750d92)
```

5. 在 Amazon Keyspaces 資料表上執行下列 CQL 查詢，以確認已將一列資料寫入資料表：

```
SELECT * from aws.user;
```

您應該會看到下列輸出：

```
fname    | lname | username | last_update_date
---------+-------+----------+---------------------------------
random    | k     | test     | 2023-12-07 13:58:31.57+0000
```

步驟 5：（可選）清理

請依照下列步驟移除本教學課程中建立的所有資源。

移除此教學課程中建立的資源

1. 刪除您的部署。您可以使用以下命令來執行此操作。

```
kubectl delete deployment my-keyspaces-app -n my-eks-namespace
```

2. 刪除 Amazon EKS 叢集及其中包含的所有網繭。這也會刪除相關資源，例如服務帳戶和 OIDC 身分識別提供者。您可以使用以下命令來執行此操作。

```
eksctl delete cluster --name my-eks-cluster --region aws-region
```

3. 刪除具有 Amazon Keyspaces 存取權限的 Amazon EKS 服務帳戶所使用的 IAM 角色。首先，您必須移除附加至該角色的受管理原則。

```
aws iam detach-role-policy --role-name my-iam-role --policy-arn
 arn:aws:iam::aws:policy/AmazonKeyspacesFullAccess
```

然後，您可以使用以下命令刪除角色。

```
aws iam delete-role --role-name my-iam-role
```

如需詳細資訊，請參閱 IAM 使用者指南中的刪除 IAM 角色 (AWS CLI)。

4. 刪除 Amazon ECR 儲存庫，包括儲存在其中的所有映像。您可以使用以下命令執行此操作。

```
aws ecr delete-repository \
      --repository-name my-ecr-repository \
      --force \
      --region aws-region
```

請注意，要刪除包含映像檔的儲存庫，必須使用force旗標。要首先刪除圖像，可以使用以下命令執行此操作。

```
aws ecr batch-delete-image \
      --repository-name my-ecr-repository \
      --image-ids imageTag=latest \
      --region aws-region
```

如需詳細資訊，請參閱 Amazon 彈性容器登錄使用者指南中的刪除映像。

5. 刪除 Amazon 密鑰空間和表。刪除密鑰空間會自動刪除該密鑰空間中的所有表。您可以使用以下選項之一來執行此操作。

   AWS CLI

   ```
   aws keyspaces delete-keyspace --keyspace-name 'aws'
   ```

   要確認密鑰空間已刪除，可以使用以下命令。

   ```
   aws keyspaces list-keyspaces
   ```

   要首先刪除該表，您可以使用以下命令。

   ```
   aws keyspaces delete-table --keyspace-name 'aws' --table-name 'user'
   ```

   要確認您的表已刪除，可以使用以下命令。

   ```
   aws keyspaces list-tables --keyspace-name 'aws'
   ```

   如需詳細資訊，請參閱《AWS CLI 命令參考》中的刪除金鑰空間和刪除資料表。

   cqlsh

   ```
   DROP KEYSPACE IF EXISTS "aws";
   ```

   要驗證您的密鑰空間已刪除，可以使用以下語句。

   ```
   SELECT * FROM system_schema.keyspaces ;
   ```

您的密鑰空間不應該列在此語句的輸出中。請注意，在刪除密鑰空間之前可能會有延遲。如需詳細資訊，請參閱 the section called “刪除密鑰空間”。

要首先刪除該表，您可以使用以下命令。

```
DROP TABLE "aws.user"
```

要確認您的表已刪除，可以使用以下命令。

```
SELECT * FROM system_schema.tables WHERE keyspace_name = "aws";
```

您的表不應該在此語句的輸出中列出。請注意，刪除表格之前可能會有延遲。如需更多詳細資訊，請參閱 the section called “DROP TABLE”。

教學課程：使用介面 VPC 端點連接至 Amazon Keyspaces

本教學將逐步引導您如何設定和使用 Amazon Keyspaces 的介面 VPC 端點。

介面 VPC 端點可讓您在 Amazon VPC 中執行的虛擬私有雲 (VPC) 與 Amazon Keyspaces 間之間進行私有通訊。介面 VPC 端點由支援 AWS PrivateLink，這是一項 AWS 服務，可在 VPC 和 AWS 服務之間進行私有通訊。如需詳細資訊，請參閱 the section called “使用 介面 VPC 端點”。

主題

教學課程的先決條件

開始本自學課程之前，請遵循中的 AWS 設定指示訪問 Amazon Keyspaces（阿帕奇卡桑德拉）。這些步驟包括註冊 AWS 和建立可存取 Amazon 金 Keyspaces 的 AWS Identity and Access Management (IAM) 主體。請記下 IAM 使用者的名稱和存取金鑰，因為本教學稍後將需要這些金鑰。

使用名稱myKeyspace和至少一個表格建立金鑰空間，以便使用本教學課程稍後的 VPC 端點測試連線。您可以在中找到詳細說明開始使用。

完成先決條件步驟後，請繼續執行步驟 1：啟動 Amazon EC2 執行個體。

步驟 1：啟動 Amazon EC2 執行個體

在此步驟中，您會在預設的 Amazon VPC 中啟動 Amazon EC2 執行個體。然後，您可以為 Amazon Keyspaces 建立和使用 VPC 端點。

啟動 Amazon EC2 執行個體

1. 前往 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 選擇 Launch Instance (啟動執行個體) 並執行下列作業：

   從 EC2 主控台儀表板的 [啟動執行個體] 方塊中，選擇 Launch 執行個體，然後從顯示的選項中選擇 Launch 執行個體。

   在「名稱和標籤」下，對於「名稱」，輸入例證的描述性名稱。

   在應用程序和操作系統映像（Amazon 機器映像）下：

   - 選擇 [快速入門]，然後選擇 [Ubuntu]。這是您的執行個體運行所在的作業系統 (OS)。
   - 在 Amazon 機器映像 (AMI) 下，您可以使用標記為符合免費方案資格的預設映像。Amazon Machine Image (AMI) 是可做為執行個體範本的基本組態。

   在例證類型之下：

   - 從「執行個體類型」清單中，選擇 t2.micro 執行個體類型 (依預設為選取)。

   在金鑰配對 (login) 下，針對金鑰配對名稱，為此教學課程選擇下列其中一個選項：

- 如果您沒有 Amazon EC2 金鑰對，請選擇 Create a new key pair (建立新的金鑰對) 並依照指示進行。系統會要求您下載私密金鑰檔案 (.pem 檔案)。稍後當您登入 Amazon EC2 執行個體時，您將需要此檔案，因此請記下檔案路徑。
- 若您已擁有 Amazon EC2 金鑰對，請前往 Select a key pair (選取金鑰對)，然後從清單中選擇您的金鑰對。您必須已具備可用的私有金鑰檔案 (.pem 檔案) 才能登入您的 Amazon EC2 執行個體。

在「網路設定」 下：

- 選擇編輯。
- 選擇 Select an existing security group (選取現有的安全群組)。
- 在安全群組清單中，選擇 default (預設)。這是您 VPC 的預設安全群組。

繼續前往 「摘要」。

- 在「摘要」面板中檢閱執行個體設定的摘要。就緒後，選擇啟動執行個體。

3. 在新 Amazon EC2 執行個體的完成畫面上，選擇 Connect 到執行個體圖標。下一個畫面會顯示必要的資訊，以及連線至新執行個體的必要步驟。請注意下列資訊：

- 用於保護密鑰文件的示例命令
- 連接字符串
- 公用網路伺服器的 DNS 名稱

記下此頁面上的資訊之後，您可以繼續進行本教學課程 (步驟 2：設定您的 Amazon EC2 執行個體) 中的下一個步驟。

> **Note**
>
> Amazon EC2 執行個體需要幾分鐘的時間才會變成可用。在您繼續進行下一個步驟之前，請先確定 Instance State (執行個體狀態) 為 `running`，並已通過其所有 Status Checks (狀態檢查)。

步驟 2：設定您的 Amazon EC2 執行個體

當您的 Amazon EC2 執行個體可用時，您可以登入該執行個體並準備首次使用。

> **Note**
>
> 以下步驟假設您是從執行 Linux 的電腦連線到 Amazon EC2 執行個體。如需其他 Connect 方式，請參閱 Amazon EC2 使用者指南中的連線到 Linux 執行個體。

設定您的亞馬遜 EC2 執行個體

1. 您需要授權傳入 SSH 流量到您的 Amazon EC2 執行個體。若要這麼做，請建立新的 EC2 安全群組，然後將安全群組指派給 EC2 執行個體。

   a. 在導覽窗格中，選擇 Security Groups (安全群組)。

   b. 選擇 Create Security Group (建立安全群組)。在 Create Security Group (建立安全群組) 視窗中，執行下列動作：

      - 安全性群組名稱 — 輸入安全性群組的名稱。例如：`my-ssh-access`
      - 說明 — 輸入安全性群組的簡短描述。
      - VPC — 選擇您的預設 VPC。
      - 在「輸入規則」區段中，選擇「新增規則」，然後執行下列動作：
        - 類型 — 選擇 [SSH]。
        - 來源 — 選擇我的 IP。
        - 選擇新增規則。

      在頁面底部，確認組態設定，然後選擇建立安全性群組。

   c. 在導覽窗格中，選擇執行個體。

   d. 選擇您在 步驟 1：啟動 Amazon EC2 執行個體 中啟動的 Amazon EC2 執行個體。

   e. 選擇 [動作]，選擇 [安全性]，然後選擇 [變更安全性群組

   f. 在變更安全性群組中，選取您先前在此程序中建立的安全性群組 (例如，`my-ssh-access`)。此外，還應選擇現有的 `default` 安全群組。確認組態設定，然後選擇指派安全群組。

2. 使用以下命令來保護您的私鑰文件免受訪問。如果跳過此步驟，連線會失敗。

```
chmod 400 path_to_file/my-keypair.pem
```

3. 使用 ssh 命令登入您的 Amazon EC2 執行個體，如下列範例所示。

```
ssh -i path_to_file/my-keypair.pem ubuntu@public-dns-name
```

您必須指定私密金鑰檔案 (.pem 檔案) 和執行個體的公開 DNS 名稱。(請參閱 步驟 1：啟動 Amazon EC2 執行個體)。

登入 ID 為 ubuntu。不需要任何密碼。

如需允許連線至 Amazon EC2 執行個體的詳細資訊以及如需 AWS CLI 指示，請參閱 Amazon EC2 使用者指南中的授權 Linux 執行個體的入站流量。

4. 下載並安裝最新版本的 AWS Command Line Interface.

a. 安裝 unzip。

```
sudo apt install unzip
```

b. 使用下載zip檔案 AWS CLI。

```
curl "https://awscli.amazonaws.com/awscli-exe-linux-x86_64.zip" -o
 "awscliv2.zip"
```

c. 解壓縮檔案。

```
unzip awscliv2.zip
```

d. 安裝 AWS CLI.

```
sudo ./aws/install
```

e. 確認 AWS CLI 安裝的版本。

```
aws --version
```

輸出應如下所示：

```
aws-cli/2.9.19 Python/3.9.11 Linux/5.15.0-1028-aws exe/x86_64.ubuntu.22 prompt/
off
```

5. 設定您的 AWS 認證，如下列範例所示。出現提示時，請輸入您的 AWS 存取金鑰 ID、密鑰和預設區域名稱。

```
aws configure

AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Default region name [None]: us-east-1
Default output format [None]:
```

6. 您必須使用 Amazon Keyspaces 的cqlsh連線，以確認您的 VPC 端點已正確設定。如果您使用本機環境或中的 Amazon Keyspaces CQL 編輯器 AWS Management Console，連線會自動透過公有端點而不是 VPC 端點進行。若要在本教學課程中用cqlsh來測試您的 VPC 端點連線，請完成中用cqlsh於連接到 Amazon Keyspaces的設定指示。

您現在已準備好為 Amazon Keyspaces 建立 VPC 端點。

步驟 3：為 Amazon Keyspaces 創建 VPC 端點

在此步驟中，您 Amazon Keyspaces 用. AWS CLI若要使用 VPC 主控台建立 VPC 端點，您可以按照指南中的建立 VPC 端點指示進行操作。AWS PrivateLink 篩選服務名稱時，請輸入**Cassandra**。

若要使用建立 VPC 端點 AWS CLI

1. 開始之前，請確認您可以使用 Amazon 金鑰空間的公有端點與 Amazon 金鑰空間通訊。

```
aws keyspaces list-tables --keyspace-name 'myKeyspace'
```

輸出顯示指定 Keyspaces 間中包含的 Amazon 密鑰空間表的列表。如果您沒有任何表格，則清單為空白。

```
{
    "tables": [
        {
            "keyspaceName": "myKeyspace",
            "tableName": "myTable1",
```

```
            "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
catalog/table/myTable1"
        },
        {
            "keyspaceName": "myKeyspace",
            "tableName": "myTable2",
            "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
catalog/table/myTable2"
        }
    ]
}
```

2. 確認 Amazon Keyspaces 是在目前 AWS 區域中建立 VPC 端點的可用服務。(命令會以粗體文字顯示，後面接著輸出範例。)

```
aws ec2 describe-vpc-endpoint-services

{
    "ServiceNames": [
        "com.amazonaws.us-east-1.cassandra",
        "com.amazonaws.us-east-1.cassandra-fips"
    ]
}
```

在範例輸出中，Amazon Keyspaces 是可用的服務之一，因此您可以繼續為其建立 VPC 端點。

3. 確定您的 VPC 識別碼。

```
aws ec2 describe-vpcs

{
    "Vpcs": [
        {
            "VpcId": "vpc-a1234bcd",
            "InstanceTenancy": "default",
            "State": "available",
            "DhcpOptionsId": "dopt-8454b7e1",
            "CidrBlock": "111.31.0.0/16",
            "IsDefault": true
        }
    ]
}
```

在範例輸出中，VPC ID 為 vpc-a1234bcd。

4. 使用篩選器收集有關 VPC 子網路的詳細資訊。

```
aws ec2 describe-subnets --filters "Name=vpc-id,Values=vpc-a1234bcd"

{
     {
    "Subnets":[
       {
          "AvailabilityZone":"us-east-1a",
          "AvailabilityZoneId":"use2-az1",
          "AvailableIpAddressCount":4085,
          "CidrBlock":"111.31.0.0/20",
          "DefaultForAz":true,
          "MapPublicIpOnLaunch":true,
          "MapCustomerOwnedIpOnLaunch":false,
          "State":"available",
          "SubnetId":"subnet-920aacf9",
          "VpcId":"vpc-a1234bcd",
          "OwnerId":"111122223333",
          "AssignIpv6AddressOnCreation":false,
          "Ipv6CidrBlockAssociationSet":[

          ],
          "SubnetArn":"arn:aws:ec2:us-east-1:111122223333:subnet/subnet-920aacf9",
          "EnableDns64":false,
          "Ipv6Native":false,
          "PrivateDnsNameOptionsOnLaunch":{
             "HostnameType":"ip-name",
             "EnableResourceNameDnsARecord":false,
             "EnableResourceNameDnsAAAARecord":false
          }
       },
       {
          "AvailabilityZone":"us-east-1c",
          "AvailabilityZoneId":"use2-az3",
          "AvailableIpAddressCount":4085,
          "CidrBlock":"111.31.32.0/20",
          "DefaultForAz":true,
          "MapPublicIpOnLaunch":true,
          "MapCustomerOwnedIpOnLaunch":false,
          "State":"available",
```

```
            "SubnetId":"subnet-4c713600",
            "VpcId":"vpc-a1234bcd",
            "OwnerId":"111122223333",
            "AssignIpv6AddressOnCreation":false,
            "Ipv6CidrBlockAssociationSet":[

            ],
            "SubnetArn":"arn:aws:ec2:us-east-1:111122223333:subnet/subnet-4c713600",
            "EnableDns64":false,
            "Ipv6Native":false,
            "PrivateDnsNameOptionsOnLaunch":{
               "HostnameType":"ip-name",
               "EnableResourceNameDnsARecord":false,
               "EnableResourceNameDnsAAAARecord":false
            }
         },
         {
            "AvailabilityZone":"us-east-1b",
            "AvailabilityZoneId":"use2-az2",
            "AvailableIpAddressCount":4086,
            "CidrBlock":"111.31.16.0/20",
            "DefaultForAz":true,
            "MapPublicIpOnLaunch":true,


         }
    ]
}
```

在範例輸出中，有兩個可用的子網路 ID：subnet-920aacf9和subnet-4c713600。

5. 建立 VPC 端點。對於 --vpc-id 參數，請指定上一個步驟的 VPC ID。針對--subnet-id參數，指定上一個步驟中的子網路 ID。使用--vpc-endpoint-type參數將端點定義為介面。若要取得有關該指令的更多資訊，請參閱《指AWS CLI 令參考》create-vpc-endpoint中的。

```
aws ec2 create-vpc-endpoint --vpc-endpoint-type Interface --vpc-id vpc-a1234bcd
 --service-name com.amazonaws.us-east-1.cassandra --subnet-id subnet-920aacf9
 subnet-4c713600

{
    "VpcEndpoint": {
        "VpcEndpointId": "vpce-000ab1cdef23456789",
        "VpcEndpointType": "Interface",
```

```
        "VpcId": "vpc-a1234bcd",
        "ServiceName": "com.amazonaws.us-east-1.cassandra",
        "State": "pending",
        "RouteTableIds": [],
        "SubnetIds": [
            "subnet-920aacf9",
            "subnet-4c713600"
        ],
        "Groups": [
            {
                "GroupId": "sg-ac1b0e8d",
                "GroupName": "default"
            }
        ],
        "IpAddressType": "ipv4",
        "DnsOptions": {
            "DnsRecordIpType": "ipv4"
        },
        "PrivateDnsEnabled": true,
        "RequesterManaged": false,
        "NetworkInterfaceIds": [
            "eni-043c30c78196ad82e",
            "eni-06ce37e3fd878d9fa"
        ],
        "DnsEntries": [
            {
                "DnsName": "vpce-000ab1cdef23456789-m2b22rtz.cassandra.us-
east-1.vpce.amazonaws.com",
                "HostedZoneId": "Z7HUB22UULQXV"
            },
            {
                "DnsName": "vpce-000ab1cdef23456789-m2b22rtz-us-
east-1a.cassandra.us-east-1.vpce.amazonaws.com",
                "HostedZoneId": "Z7HUB22UULQXV"
            },
            {
                "DnsName": "vpce-000ab1cdef23456789-m2b22rtz-us-
east-1c.cassandra.us-east-1.vpce.amazonaws.com",
                "HostedZoneId": "Z7HUB22UULQXV"
            },
            {
                "DnsName": "vpce-000ab1cdef23456789-m2b22rtz-us-
east-1b.cassandra.us-east-1.vpce.amazonaws.com",
                "HostedZoneId": "Z7HUB22UULQXV"
```

```
            },
            {
                "DnsName": "vpce-000ab1cdef23456789-m2b22rtz-us-
east-1d.cassandra.us-east-1.vpce.amazonaws.com",
                "HostedZoneId": "Z7HUB22UULQXV"
            },
            {
                "DnsName": "cassandra.us-east-1.amazonaws.com",
                "HostedZoneId": "ZONEIDPENDING"
            }
        ],
        "CreationTimestamp": "2023-01-27T16:12:36.834000+00:00",
        "OwnerId": "111122223333"
    }
}

}
```

步驟 4：設定 VPC 端點連線的權限

此步驟中的程序示範如何設定將 VPC 端點與 Amazon Keyspaces 搭配使用的規則和許可。

設定新端點的輸入規則以允許 TCP 輸入流量

1. 在 Amazon VPC 主控台的左側面板上，選擇「端點」，然後選擇您在先前步驟中建立的端點。
2. 選擇 [安全性群組]，然後選擇與此端點相關聯的安全性群組。
3. 選擇入站規則，然後選擇編輯入站規則。
4. 添加與類型作為 CQLSH /卡桑德拉的入站規則。這會將連接埠範圍自動設定為 9142。
5. 若要儲存新的輸入規則，請選擇 [儲存規則]。

若要設定 IAM 使用者許可

1. 確認用來連線到 Amazon Keyspaces 的 IAM 使用者具有適當的許可。在 AWS Identity and Access Management (IAM) 中，您可以使用 AWS 受管政策授與 IAM 使用者`AmazonKeyspacesReadOnlyAccess`對 Amazon Keyspaces 的讀取存取權限。

   a. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
   b. 在 IAM 主控台儀表板上，選擇 Users (使用者)，然後從清單中選擇 IAM 使用者。

c. 在 Summary (摘要) 頁面上，選擇 Add permissions (新增許可)。

d. 選擇 Attach existing policies directly (直接連接現有政策)。

e. 從原則清單中選擇 [AmazonKeyspacesReadOnly存取權]，然後選擇 [下一步：複查]。

f. 選擇新增許可。

2. 確認您可以透過 VPC 端點存取 Amazon Keyspaces。

```
aws keyspaces list-tables --keyspace-name 'my_Keyspace'
```

如果你願意，你可以嘗試一些其他的 AWS CLI 命令 Amazon Keyspaces。如需詳細資訊，請參閱 AWS CLI 命令參考。

> **Note**
>
> IAM 使用者或角色存取 Amazon Keyspaces 所需的最低許可是系統表格的讀取許可，如下列政策所示。如需以原則為基礎之權限的詳細資訊，請參閱the section called “身分型政策範例”。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "cassandra:Select"
         ],
         "Resource":[
            "arn:aws:cassandra:us-east-1:555555555555:/keyspace/system*"
         ]
      }
   ]
}
```

3. 使用 VPC 授予 IAM 使用者對 Amazon EC2 執行個體的讀取存取權限。

將 Amazon Keyspaces 與 VPC 端點搭配使用時，您需要將存取 Amazon Keyspaces 唯讀許可的 IAM 使用者或角色授與 Amazon EC2 執行個體和 VPC，以收集端點和網路界面資料。Amazon Keyspaces 會將此資訊儲存在`system.peers`表格中，並使用它來管理連線。

> **Note**
>
> 受管政策`AmazonKeyspacesReadOnlyAccess_v2`和`AmazonKeyspacesFullAccess`包含允許 Amazon Keyspaces 存取 Amazon EC2 執行個體以讀取有關可用界面 VPC 端點的資訊的必要許可。

a. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。

b. 在 IAM 主控台儀表板上，選擇 [政策]。

c. 選擇 [建立原則]，然後選擇 [JSON] 索引標籤。

d. 複製下列原則，然後選擇「下一步：標籤」。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ListVPCEndpoints",
         "Effect":"Allow",
         "Action":[
            "ec2:DescribeNetworkInterfaces",
            "ec2:DescribeVpcEndpoints"
         ],
         "Resource": "*"
      }
   ]
}
```

e. 選擇下一步：複查，輸入策略`keyspacesVPCendpoint`的名稱，然後選擇建立策略。

f. 在 IAM 主控台儀表板上，選擇 Users (使用者)，然後從清單中選擇 IAM 使用者。

g. 在 Summary (摘要) 頁面上，選擇 Add permissions (新增許可)。

h. 選擇 Attach existing policies directly (直接連接現有政策)。

i. 從策略清單中，選擇金鑰空間格點，然後選擇下一步：複查。

j. 選擇新增許可。

4. 若要確認 Amazon Keyspaces 資料`system.peers`表是否已使用 VPC 資訊更新，請使用從 Amazon EC2 執行個體執行下列查詢。`cqlsh`如果您在步驟 2 中尚未在 Amazon EC2 執行個體`cqlsh`上安裝，請按照中的指示進行the section called “使用 cqlsh-expansion”。

```
SELECT peer FROM system.peers;
```

根據您所在地 AWS 區的 VPC 和子網路設定，輸出會傳回具有私有 IP 位址的節點。

```
peer
---------------
112.11.22.123
112.11.22.124
112.11.22.125
```

> **Note**
>
> 您必須使用 Amazon Keyspaces 的`cqlsh`連線，以確認您的 VPC 端點已正確設定。如果您使用本機環境或中的 Amazon Keyspaces CQL 編輯器 AWS Management Console，連線會自動透過公有端點而不是 VPC 端點進行。如果您看到九個 IP 地址，這些是 Amazon Keyspaces 為公用端點連接自動寫入`system.peers`表格的條目。

步驟 5：配置監視 CloudWatch

此步驟說明如何使用 Amazon CloudWatch 監控 VPC 端點與 Amazon Keyspaces 的連線。

AWS PrivateLink 將資料點發佈至介面端點的 CloudWatch 相關資料點。您可以使用指標來確認系統的運作符合預期。中的`AWS/PrivateLinkEndpoints`命名空間 CloudWatch 包括介面端點的度量。如需詳細資訊，請參閱CloudWatch 指AWS PrivateLink南 AWS PrivateLink中的量度。

使用 VPC 端點指標建立 CloudWatch 儀表板

1. 在開啟 CloudWatch 主控台https://console.aws.amazon.com/cloudwatch/。
2. 在導覽窗格中，選擇 Dashboards (儀表板)。然後選擇建立儀表板。輸入控制面板的名稱，然後選擇 [建立]。
3. 在「新增小工具」下，選擇「編號
4. 在「量度」下，選擇「AWS/ PrivateLink 端點」。
5. 選擇端點類型、服務名稱、VPC 端點識別碼、VPC 識別碼。

6. 選取量度NewConnections，ActiveConnections然後選擇「建立小工具」。
7. 儲存儀表板。

此ActiveConnections測量結果定義為端點在過去 1 分鐘內接收到的並行作用中連線數目。此NewConnections測量結果定義為過去 1 分鐘內透過端點建立的新連線數目。

如需有關建立面板的詳細資訊，請參閱CloudWatch 使用者指南中的建立儀表板。

步驟 6：(選擇性) 為應用程式設定連線集區大小的最佳作法

在本節中，我們概述如何根據應用程式的查詢輸送量需求判斷理想的連線集區大小。

Amazon Keyspaces 允許每個 TCP 連接每秒最多 3,000 個 CQL 查詢。因此，驅動程序可以使用 Amazon Keyspaces 建立的連接數量幾乎沒有限制。不過，建議您將連線集區大小與應用程式的需求相符，並在搭配 VPC 端點連線使用 Amazon Keyspaces space 時考量可用的端點。

您可以在用戶端驅動程式中設定連線集區大小。例如，根據 2 的本機集區大小和跨 3 個可用區域建立的 VPC 介面端點，驅動程式會建立 6 個連線以供查詢 (總共 7 個，其中包括控制連線)。使用這 6 個連線，您每秒最多可支援 18,000 個 CQL 查詢。

如果您的應用程式每秒需要支援 40,000 個 CQL 查詢，請從判斷所需連線集區大小所需的查詢數量倒退。若要支援每秒 40,000 個 CQL 查詢，您必須將本機集區大小設定為至少 5 個，每秒至少支援 45,000 個 CQL 查詢。

您可以使用AWS/Cassandra命名空間中的PerConnectionRequestRateExceeded CloudWatch測量結果來監督是否超過每秒每個連線的作業數目上限配額。該PerConnectionRequestRateExceeded指標顯示超出每個連線請求率配額的 Amazon Keyspaces 的請求數目。

此步驟中的程式碼範例顯示如何在使用介面 VPC 端點時估計和設定連線集區。

Java

您可以在 Java 驅動程式中設定每個集區的連線數目。有關 Java 客戶端驅動程序連接的完整實例，敬請參閱the section called “使用卡桑德拉 Java 客戶端驅動程序”。

啟動用戶端驅動程式時，會先建立管理工作的控制連線，例如結構描述和拓撲變更。然後創建其他連接。

在下列範例中，本機集區大小驅動程式組態指定為 2。如果 VPC 端點是跨 VPC 內的 3 個子網路建立的，則介面端點會產生 7 個NewConnections中 CloudWatch 的結果，如下列公式所示。

```
NewConnections = 3 (VPC subnet endpoints created across) * 2 (pool size) + 1
 ( control connection)
```

```
datastax-java-driver {

    basic.contact-points = [ "cassandra.us-east-1.amazonaws.com:9142"]
    advanced.auth-provider{
        class = PlainTextAuthProvider
           username = "ServiceUserName"
           password = "ServicePassword"
    }
    basic.load-balancing-policy {
        local-datacenter = "us-east-1"
        slow-replica-avoidance = false
    }

    advanced.ssl-engine-factory {
        class = DefaultSslEngineFactory
        truststore-path = "./src/main/resources/cassandra_truststore.jks"
        truststore-password = "my_password"
        hostname-validation = false
      }
    advanced.connection {
        pool.local.size = 2
          }
}
```

如果活動連接的數量與配置的池大小（跨子網絡的聚合）+ 1 個控制連接不匹配，則有些東西阻止了連接被創建。

Node.js

您可以在 Node.js 驅動程式中設定每個集區的連線數目。如需 Node.js 用戶端驅動程式連線的完整範例，請參閱the section called “使用卡桑德拉 Node.js 客戶端驅動程序”。

對於下列程式碼範例，本機集區大小驅動程式組態會指定為 1。如果 VPC 端點是跨 VPC 內的 4 個子網路建立的，則介面端點會導致 5 NewConnections in CloudWatch ，如下列公式所示。

```
NewConnections = 4 (VPC subnet endpoints created across) * 1 (pool size) + 1
 ( control connection)
```

```
const cassandra = require('cassandra-driver');
```

```
const fs = require('fs');
const types = cassandra.types;
const auth = new cassandra.auth.PlainTextAuthProvider('ServiceUserName',
 'ServicePassword');
const sslOptions1 = {
         ca: [
                    fs.readFileSync('/home/ec2-user/sf-class2-root.crt', 'utf-8')],
                    host: 'cassandra.us-east-1.amazonaws.com',
                    rejectUnauthorized: true
        };
const client = new cassandra.Client({
                   contactPoints: ['cassandra.us-east-1.amazonaws.com'],
                   localDataCenter: 'us-east-1',
                   pooling: { coreConnectionsPerHost: { [types.distance.local]:
 1 } },
                   consistency: types.consistencies.localQuorum,
                   queryOptions: { isIdempotent: true },
                   authProvider: auth,
                   sslOptions: sslOptions1,
                   protocolOptions: { port: 9142 }
        });
```

步驟 7：（可選）清理

如果您要刪除在此教學課程中建立的資源，請遵循下列程序。

刪除 Amazon Keyspaces 的 VPC 端點

1. 登入 Amazon EC2 執行個體。
2. 判斷用於 Amazon Keyspaces 的 VPC 端點識別碼。如果您省略`grep`參數，則會顯示所有服務的 VPC 端點資訊。

```
aws ec2 describe-vpc-endpoint-services | grep ServiceName | grep cassandra

{
    "VpcEndpoint": {
        "PolicyDocument": "{\"Version\":\"2008-10-17\",\"Statement\":[{\"Effect\":
\"Allow\",\"Principal\":\"*\",\"Action\":\"*\",\"Resource\":\"*\"}]}",
        "VpcId": "vpc-0bbc736e",
        "State": "available",
        "ServiceName": "com.amazonaws.us-east-1.cassandra",
```

```
        "RouteTableIds": [],
        "VpcEndpointId": "vpce-9b15e2f2",
        "CreationTimestamp": "2017-07-26T22:00:14Z"
    }
}
```

在範例輸出中，VPC 端點 ID 為 vpce-9b15e2f2。

3. 刪除 VPC 端點。

```
aws ec2 delete-vpc-endpoints --vpc-endpoint-ids vpce-9b15e2f2

{
    "Unsuccessful": []
}
```

空白陣列 [] 表示成功 (沒有未成功的請求)。

若要終止 Amazon EC2 執行個體

1. 在 https://console.aws.amazon.com/ec2/ 開啟 Amazon EC2 主控台。
2. 在導覽窗格中，選擇執行個體。
3. 選擇 Amazon EC2 執行個體。
4. 選擇動作，選擇「執行處理狀態」，然後選擇「終止」。
5. 在確認視窗中，請選擇 Yes, Terminate (是，終止)。

設定亞馬遜 Keyspaces 的跨帳戶存取權

您可以獨立建立和使用AWS 帳戶來隔離資源，並在不同的環境中使用，例如開發和生產。本主題將逐步引導您 VPC 用Amazon Virtual Private Cloud. 如需 IAM 的詳細資訊，請參閱《IAM 使用者指南使用者指南使用者指南使用者指南》中的使用個開發和生產帳戶的範例案

如需 Amazon Keyspaces 和私有 VPC 端點的詳細資訊，請參閱the section called “使用 介面 VPC 端點”。

主題

- 在共享 VPC 中設定 Keyspaces 存的跨帳戶存取權
- 設定 Amazon Keyspaces 的跨帳戶存取權

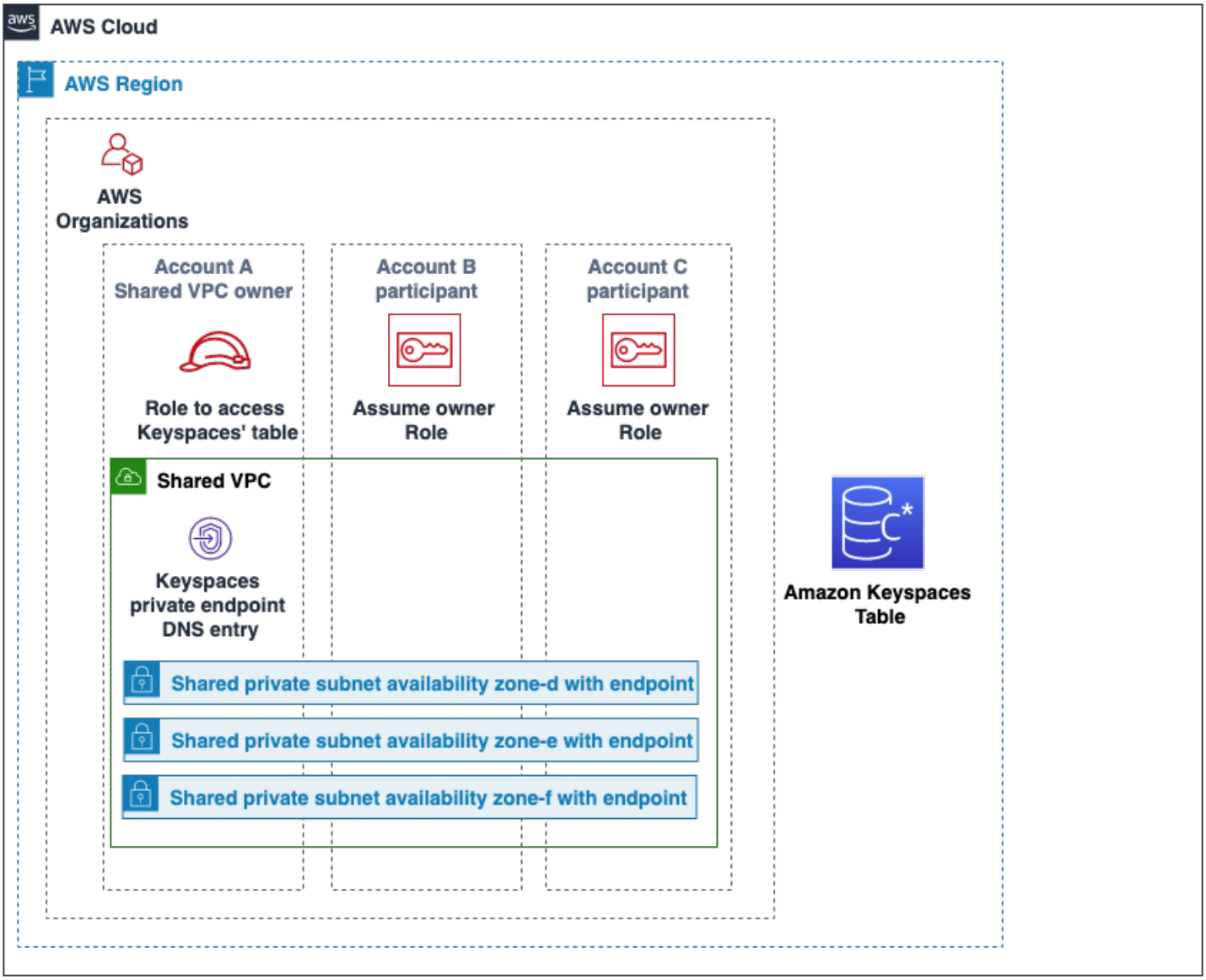
在共享 VPC 中設定 Keyspaces 存的跨帳戶存取權

您可以建立不同AWS 帳戶的資源與應用程式分開。例如，您可以為 Amazon Keyspaces 表格建立一個帳戶、在開發環境中為應用程式建立不同的帳戶，以及為生產環境中的應用程式建立另一個帳戶。本主題將引導您完成使用共用 VPC 中的介面 VPC 端點為 Amazon Keyspaces 設定跨帳戶存取所需的組態步驟。

如需如何為 Amazon Keyspaces 設定 VPC 端點的詳細步驟，請參閱the section called “步驟 3：為 Amazon Keyspaces 創建 VPC 端點”。

在此範例中，我們在共用 VPC 中使用下列三個帳戶：

- `Account A`— 此帳戶包含基礎設施，包括 VPC 端點、VPC 子網路和 Amazon Keyspaces 表。
- `Account B`— 此帳戶包含開發環境中的應用程式，該應用程式需要連線至中的 Amazon Keyspaces 表格`Account A`。
- `Account C`— 此帳戶包含生產環境中的應用程式，該應用程式需要連線至中的 Amazon Keyspaces 表格`Account A`。



Account A是包含Account B和Account C需要訪問的資源的帳戶，信任帳戶Account A也是如此。 Account B並且Account C是具有需要存取中資源的主體的帳戶Account A，以Account B及Account C是受信任的帳戶。信任帳戶會透過共用 IAM 角色，將許可授與受信任的帳戶。下列程序概述中的設定步驟Account A。

的組態**Account A**

1. 用AWS Resource Access Manager於建立子網路的資源共用，並與和共用私有Account B子網路Account C。

   Account B現在Account C可以在與其共享的子網路中查看和建立資源。

2. 建立由支援的 Amazon Keyspaces 私有 VPC 端點AWS PrivateLink。這會跨共用子網路和 Amazon Keyspaces 服務端點的 DNS 項目建立多個端點。
3. 創建一個亞馬遜密 Keyspaces 間和表。
4. 建立可完整存取 Amazon Keyspaces 表格的 IAM 角色、讀取 Amazon Keyspaces 系統表格的存取權限，以及能夠描述 Amazon EC2 VPC 資源，如下列政策範例所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "CrossAccountAccess",
            "Effect": "Allow",
            "Action": [
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeVpcEndpoints",
                "cassandra:*"
            ],
            "Resource": "*"
        }
    ]
}
```

5. 設定Account C可假設為受信任帳戶的Account B IAM 角色信任政策，如下列範例所示。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111111111111:root"
            },
            "Action": "sts:AssumeRole",
            "Condition": {}
        }
    ]
}
```

如需跨帳戶的詳細資訊，請參閱《IAM 使用者指南》中的跨帳戶政策。

Account B和中的組態Account C

1. 在Account B和中Account C，建立新角色並附加下列原則，以允許主參與者擔任在中建立的共用角色Account A。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

   允許主體假設共用角色是使用AWS Security Token Service (AWS STS) 的AssumeRole API 來實作。如需詳細資訊，請參閱《IAM 使用者指南》中的對您擁有的另一AWS 帳戶個使用者中的 IAM 使用者提供存取權。

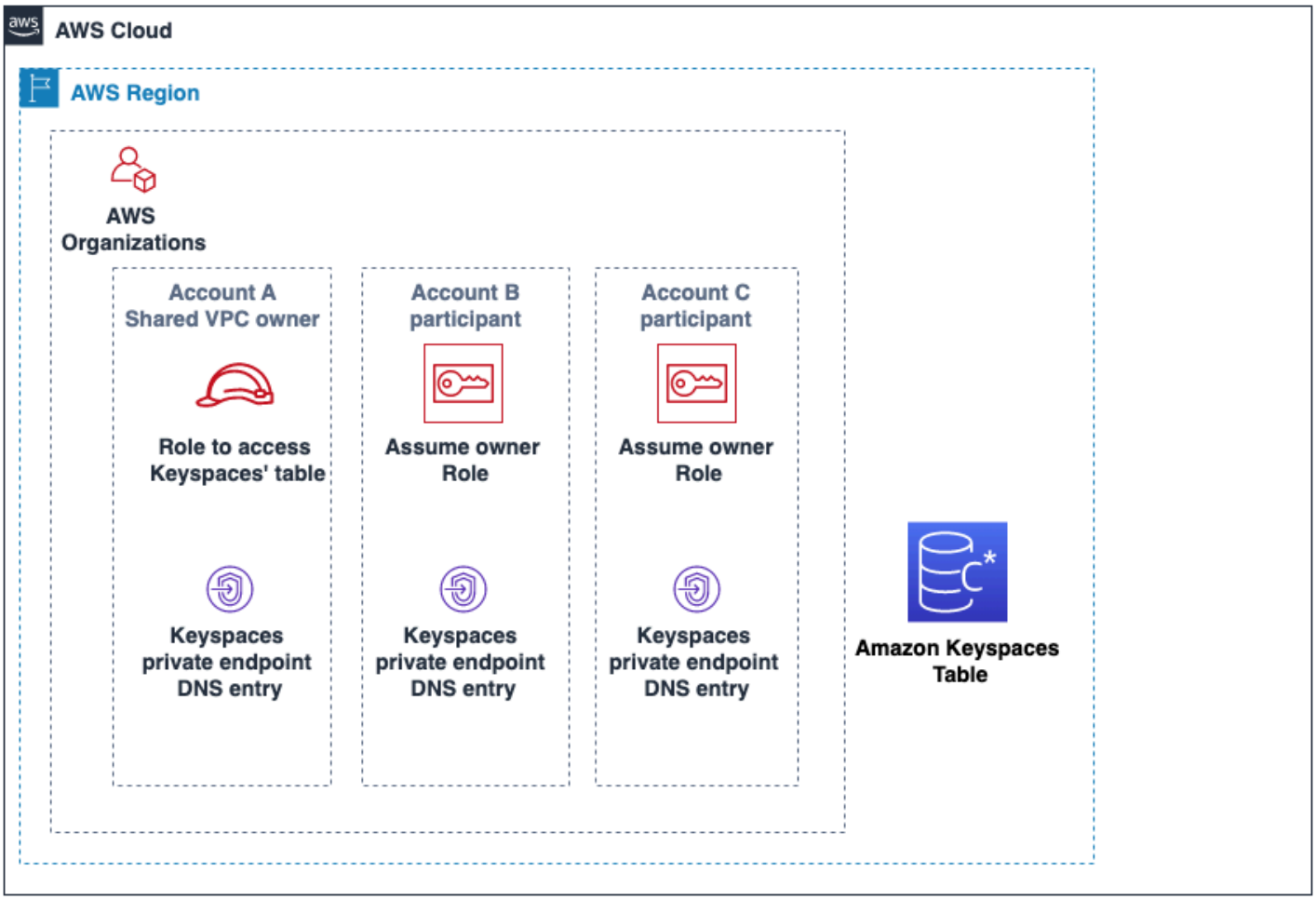
2. 在Account B和中Account C，您可以建立使用 SIGV4 身份驗證外掛程式的應用程式，該應用程式允許應用程式假設共用角色Account A透過共用 VPC 中的 VPC 端點連接至位於中的 Amazon Keyspaces 表。如需 SIGV4 外掛程式的詳細資訊，請參閱the section called “建立認證”。

設定 Amazon Keyspaces 的跨帳戶存取權

如果 Amazon Keyspaces 表和私有 VPC 端點屬於不同帳戶，但未共用 VPC，則應用程式仍然可以使用 VPC 端點連接跨帳戶。因為這些帳戶不共用 VPC 端點、Account AAccount B、且Account C需要它們自己的 VPC 端點。對於 Cassandra 客戶端驅動程序，亞馬遜 Keyspaces 看起來像單個節點而不是多節點集群。連線後，用戶端驅動程式會到達 DNS 伺服器，該伺服器會傳回帳戶 VPC 中的其中一個可用端點。

您也可以使用公有端點或在每個帳戶中部署私有 VPC 端點，在不使用共用 VPC 端點的情況下跨不同帳戶存取 Amazon Keyspaces 表。不使用共用 VPC 時，每個帳戶都需要自己的 VPC 端點。在此範例中Account AAccount B，並Account C要求其自己的 VPC 端點才能存取中的資料表Account A。在此組態中使用 VPC 端點時，Amazon Keyspaces 會顯示為 Cassandra 用戶端驅動程式的單一節點叢集，而不是多節點叢集。連線後，用戶端驅動程式會到達 DNS 伺服器，該伺服器會傳回帳戶 VPC

中的其中一個可用端點。但用戶端驅動程式無法存取system.peers資料表以探索其他端點。由於可用的主機較少，因此驅動程式所產生的連線較少。若要進行調整，請將驅動程式的連線集區設定增加三倍。



開始使用 Amazon Keyspaces（阿帕奇卡桑德拉）

本教程是給你的，如果你是新的 Apache 卡桑德拉和 Amazon Keyspaces（阿帕奇卡桑德拉）。在本教學中，您會安裝成功使用 Amazon Keyspaces 所需的所有程式和驅動程式。

如需使用不同 Cassandra 用戶端驅動程式以程式設計方式連接至 Amazon Keyspaces 的教學課程，請參 the section called “使用卡桑德拉客戶端驅動程序”

主題

教學課程的先決條件

開始本自學課程之前，請遵循中的 AWS 設定指示訪問 Amazon Keyspaces（阿帕奇卡桑德拉）。這些步驟包括註冊 AWS 和建立可存取 Amazon Keyspaces 的 AWS Identity and Access Management (IAM) 使用者。

此外，如果您正在使用`cqlsh`或 Apache 2.0 授權的 Cassandra 用戶端驅動程式來完成教學課程，請完成. 用`cqlsh`於連接到 Amazon Keyspaces

完成先決條件步驟後，請繼續執行教程步驟 1：在 Amazon 密鑰空間中創建 Keyspaces 間和表。

教程步驟 1：在 Amazon 密鑰空間中創建 Keyspaces 間和表

在本節中，您會建立金鑰空間，並使用主控台將資料表加入其中。

> **Note**
>
> 在開始之前，請確定您已設定所有教學課程必要條件。

主題

創建一個密鑰空間

索引鍵空間會將與一或多個應用程式相關的相關資料表分組。一個密鑰空間包含一個或多個表，並定義了它包含的所有表的複製策略。如需金鑰空間的相關資訊，請參閱下列主題：

- 使用密鑰空間：the section called “創建密鑰空間”
- 資料定義語言 (DDL) 陳述式：Keyspaces
- 配額 Amazon Keyspaces（阿帕奇卡桑德拉）

當您建立金鑰空間時，您必須指定金鑰空間名稱。

> **Note**
>
> 密鑰空間的複製策略必須是`SingleRegionStrategy`。 `SingleRegionStrategy`在其 AWS 區域中的三個可用區域複寫資料。

使用主控台

若要使用主控台建立金鑰空間

1. 登錄到 AWS Management Console，並打開 Amazon Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。
2. 在導覽窗格中，選擇 [Keyspaces]。
3. 選擇建立金鑰空間。
4. 在 [金鑰空間名稱] 方塊中**`myGSGKeyspace`**，輸入金鑰空間的名稱。

   名稱限制：

   - 不能為空。
   - 允許的字元：英數字元和底線 (_)。
   - 長度上限為 48 個字元。
5. 若要建立金鑰空間，請選擇 [建立金鑰空間]。
6. 驗證密鑰空間`myGSGKeyspace`是通過執行以下操作創建的：

   a. 在導覽窗格中，選擇 [Keyspaces]。

   b. 在密鑰空間列表`myGSGKeyspace`中找到您的密鑰空間。

使用定制列表

下列程序會使用 CQL 建立金鑰空間。

若要使用 CQL 建立金鑰空間

1. 開啟命令殼層，然後輸入下列內容：

   ```
   cqlsh
   ```

2. 使用以下 CQL 命令創建密鑰空間。

   ```
   CREATE KEYSPACE IF NOT EXISTS "myGSGKeyspace"
      WITH REPLICATION = {'class': 'SingleRegionStrategy'};
   ```

   SingleRegionStrategy使用三個複寫係數，並在其區域中跨三個 AWS 可用區域複寫資料。

   > **Note**
   >
   > Amazon Keyspaces 將所有輸入預設為小寫，除非您用引號括起來。在這種情況下，請注意"myGSGKeyspace"。

3. 確認您的密鑰空間已創建。

   ```
   SELECT * from system_schema.keyspaces ;
   ```

   您的密鑰空間應該被列出。

建立資料表

表格是您資料的組織和儲存位置。資料表的主索引鍵會決定資料表中資料分割的方式。主索引鍵是由必要的分割索引鍵和一或多個選擇性叢集資料行組成。構成主鍵的組合值在所有表的數據中必須是唯一的。如需有關表格的詳細資訊，請參閱下列主題：

- 使用表格：the section called “建立資料表”
- DDL 陳述式：資料表
- 表資源管理：無伺服器資源管理
- 監視表資源使用率：the section called “使用監控 CloudWatch”
- 配額 Amazon Keyspaces（阿帕奇卡桑德拉）

當您建立資料表時，請指定下列項目：

- 資料表的名稱。
- 資料表中每個資料行的名稱和資料類型。
- 資料表的主索引鍵。
  - 分割區索引鍵 — 必要
  - 叢集資料欄 — 選用

使用下列程序來建立含有指定資料行、資料類型、資料分割索引鍵和叢集資料行的資料表。

使用主控台

下列程序會`employees_tbl`使用這些資料行和資料類型建立資料表。

```
ID            text
name          text
region        text
division      text
project       text
role          text
pay_scale     int
vacation_hrs  float
manager_id    text
```

若要使用主控台建立資料表

1. 登錄到 AWS Management Console，並打開 Amazon Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。
2. 在導覽窗格中，選擇 [Keyspaces]。
3. 選擇`myGSGKeyspace`作為您要在其中創建此表的密鑰空間。
4. 選擇 建立資料表 。
5. 在 [表格名稱] 方塊中，輸入**`employees_tbl`**做為表格的名稱。

   名稱限制：

   - 不能為空。
   - 允許的字元：英數字元和底線 (_)。
   - 長度上限為 48 個字元。

6. 在「欄」段落中，針對要新增至此表格的每個資料欄重複下列步驟。

   新增下列欄和資料類型。

```
id            text
name          text
region        text
division      text
project       text
role          text
pay_scale     int
vacation_hrs  float
manager_id    text
```

   a. 名稱 — 輸入欄的名稱。

      名稱限制：

      - 不能為空。
      - 允許的字元：英數字元和底線 (_)。
      - 長度上限為 48 個字元。

   b. 類型 — 在資料類型清單中，選擇此欄的資料類型。

   c. 如果您要新增其他欄，請選擇 [新增欄]。

7. 在分割區索引鍵下選擇`id`作為分割區索引鍵。每個表都需要一個分區鍵。磁碟分割索引鍵可以由一或多個資料行組成。

8. 新增`division`為叢集資料行。叢集資料行是選擇性的，可決定每個磁碟分割內的排序順序。

   a. 若要新增叢集資料行，請選擇 [新增叢集資料行]。

   b. 在「欄」清單中，選擇「分割」。在「順序」清單中，選擇 ASC，依此欄中的值以遞增順序排序。(選擇「描述」做為遞減順序)。

9. 在 [表格設定] 區段中，選擇 [預設設定]。

10. 選擇 建立資料表 。

11. 確認您的資料表是否已建立。

    a. 在導覽窗格中，選擇 Tables (資料表)。

    b. 確認您的表格位於表格清單中。

    c. 選擇表格的名稱。

d. 確認所有資料欄和資料類型都正確無誤。

> **Note**
>
> 列出的順序可能不會與您將它們加入表格中的順序相同。

e. 在叢集資料行中，確認分割是以 true 識別。所有其他表列應該是假的。

使用定制列表

下列程序會使用 CQL 建立包含下列欄和資料類型的資料表。該id列是分區鍵。

```
id            text
name          text
region        text
division      text
project       text
role          text
pay_scale     int
vacation_hrs  float
manager_id    text
```

若要使用 CQL 建立資料表

1. 開啟命令外殼，然後輸入以下內容：

   cqlsh

2. 在 cqlsh prompt (cqlsh>) 中，指定要在其中建立資料表的金鑰空間。

   ```
   USE "myGSGKeyspace" ;
   ```

3. 在 keyspace 提示符（cqlsh:keyspace_name>）中，通過在命令窗口中輸入以下代碼來創建表格。

   ```
   CREATE TABLE IF NOT EXISTS "myGSGKeyspace".employees_tbl (
      id text,
      name text,
      region text,
      division text,
      project text,
   ```

```
    role text,
    pay_scale int,
    vacation_hrs float,
    manager_id text,
    PRIMARY KEY (id,division))
    WITH CLUSTERING ORDER BY (division ASC) ;
```

> **Note**
>
> ASC是預設的叢集順序。您也可以指DESC定遞減順序。

請注意，該id列是分區鍵。然後，division是按升序排序的聚類列（ASC）。

4. 確認您的資料表是否已建立。

```
SELECT * from system_schema.tables WHERE keyspace_name='myGSGKeyspace' ;
```

你的表應該被列出。

5. 驗證表格的結構。

```
SELECT * FROM system_schema.columns WHERE keyspace_name = 'myGSGKeyspace' AND
 table_name = 'employees_tbl' ;
```

確認所有資料行和資料類型都如預期般。列的順序可能與CREATE語句中的不同。

若要對資料表中的資料執行 CRUD (建立、讀取、更新和刪除) 作業，請繼續執行the section called “第 2 步：CRUD 操作”。

教學步驟 2：建立、讀取、更新和刪除資料 (CRUD)

在本節中，您可以使用主控台上的 CQL 編輯器對資料表中的資料執行 CRUD (建立、讀取、更新和刪除) 作業。您也可以使用執行命令cqlsh。

主題

- 教學課程：將資料插入並載入 Amazon Keyspaces 表格
- 教程：從 Amazon Keyspaces 表中讀取

- 教程：更新 Amazon Keyspaces 表中的數據
- 教學課程：刪除 Amazon Keyspaces 表格中的資料

教學課程：將資料插入並載入 Amazon Keyspaces 表格

若要在資料employees_tbl表中建立資料，請使用INSERT陳述式新增單一資料列。

1. 您必須先將目前 cqlsh 工作階段的寫入一致性設定為，才能使用 cqlsh 將資料寫入 Amazon Keyspaces 資料表。LOCAL_QUORUM如需有關支援一致性層級的詳細資訊，請參閱the section called “寫入一致性層級”。請注意，如果您在中使用 CQL 編輯器，則不需要執行此步驟。 AWS Management Console

```
CONSISTENCY LOCAL_QUORUM;
```

2. 若要插入單一記錄，請在 CQL 編輯器中執行下列命令。

```
INSERT INTO "myGSGKeyspace".employees_tbl (id, name, project, region, division,
 role, pay_scale, vacation_hrs, manager_id)
VALUES ('012-34-5678','Russ','NightFlight','US','Engineering','IC',3,12.5,
 '234-56-7890') ;
```

3. 執行下列命令，確認資料已正確新增至您的資料表。

```
SELECT * FROM "myGSGKeyspace".employees_tbl ;
```

若要使用 cqlsh 從檔案插入多筆記錄

1. 下載包含在下列封存檔 sampledata.zip 中的範例資料檔案 (employees.csv)。此 CSV (逗號分隔值) 檔案包含下列資料。請記住您儲存檔案的目標路徑。

ID	name	project	region	division	role	pay_scale	vacation_hrs	manager_id
123-45-6789	Bob	NightFlight	US	Engineering	Intern	1	0	234-56-7890
234-56-7890	Bob	NightFlight	US	Engineering	Manager	6	72	789-01-2345
345-67-8901	Sarah	Storm	US	Engineering	IC	4	108	234-56-7890
456-78-9012	Beth	NightFlight	US	Engineering	IC	7	100.5	234-56-7890
567-89-0123	Ahmed	NightFlight	US	Marketing	IC	4	88	678-90-1234
678-90-1234	Alan	Storm	US	Marketing	Manager	3	18.4	789-01-2345
789-01-2345	Roberta	All	US	Executive	CEO	15	184	None

2. 開啟命令外殼，然後輸入以下內容：

cqlsh

3. 在cqlsh提示符號 (cqlsh>) 中，指定一個密鑰空間。

```
USE "myGSGKeyspace" ;
```

4. 將寫入一致性設定為LOCAL_QUORUM。如需有關支援一致性層級的詳細資訊，請參閱the section called “寫入一致性層級”。

```
CONSISTENCY LOCAL_QUORUM;
```

5. 在密鑰空間提示符（cqlsh:keyspace_name>），運行以下查詢。

```
COPY employees_tbl
 (id,name,project,region,division,role,pay_scale,vacation_hrs,manager_id)
FROM 'path-to-the-csv-file/employees.csv' WITH delimiter=',' AND header=TRUE ;
```

6. 執行下列查詢，確認資料已正確新增至資料表。

```
SELECT * FROM employees_tbl ;
```

教程：從 Amazon Keyspaces 表中讀取

在此教學課程：將資料插入並載入 Amazon Keyspaces 表格區段中，您使用SELECT陳述式來確認您已成功將資料新增至資料表。在本節中，您可以調整使用SELECT以顯示特定欄，以及僅顯示符合特定準則的列。

SELECT聲明的一般形式如下。

```
SELECT column_list FROM table_name [WHERE condition [ALLOW FILTERING]] ;
```

主題

- 選取表格中的所有資料
- 選取資料欄的子集
- 選取列的子集

選取表格中的所有資料

SELECT語句的最簡單形式返回表中的所有數據。

> **Important**
>
> 在生產環境中，執行此命令通常不是最佳作法，該命令會傳回資料表中的所有資料。

若要選取表格的所有資料

- 執行下列查詢。

```
SELECT * FROM "myGSGKeyspace".employees_tbl ;
```

使用萬用字元 (*) 做為column_list選取所有欄。

選取資料欄的子集

若要查詢資料欄的子集

- 若只要擷取idname、和manager_id資料行，請執行下列查詢。

```
SELECT name, id, manager_id FROM "myGSGKeyspace".employees_tbl ;
```

輸出將僅包含在SELECT語句中列出的順序指定的列。

選取列的子集

查詢大型資料集時，您可能只想要符合特定條件的記錄。要做到這一點，你可以追加一個WHERE子句到我們的SELECT聲明的末尾。

若要查詢列的子集

- 若要僅擷取具有 ID 員工的記錄'234-56-7890'，請執行下列查詢。

```
SELECT * FROM "myGSGKeyspace".employees_tbl WHERE id='234-56-7890' ;
```

前面的SELECT語句只返回其中的id行234-56-7890。

了解條**WHERE**款

該WHERE子句用於過濾數據，並僅返回符合指定條件的數據。指定的條件可以是簡單條件或複合條件。

如何在**WHERE**子句中使用條件

- 一個簡單的條件-單列。

```
WHERE column_name=value
```

  如果符合下列任一條件，您可以在WHERE子句中使用簡單的條件：

  - 資料行是資料表主索引鍵中唯一的資料行。
  - 您可以ALLOW FILTERING在條WHERE款中的條件之後加入。

    請注意，使用ALLOW FILTERING可能會導致效能不一致，尤其是對於大型和多分區資料表而言。

- 複合條件 — 由連接的多個簡單條件AND。

```
WHERE column_name1=value1 AND column_name2=value2 AND column_name3=value3...
```

  如果符合下列任一WHERE條件，您可以在子句中使用複合條件：

  - WHERE子句中的列與表的主鍵中的列完全匹配，不多也不少。
  - 您可以在WHERE子句中的複合條件ALLOW FILTERING之後加入，如下列範例所示。

```
SELECT * FROM my_table WHERE col1=5 AND col2='Bob' ALLOW FILTERING ;
```

    請注意，使用ALLOW FILTERING可能會導致效能不一致，尤其是對於大型和多分區資料表而言。

嘗試一下

建立您自己的 CQL 查詢，以便從employees_tbl資料表中尋找下列項目：

- 尋找所nameproject有員工id的、和。
- 查找實習生正在進行的項目Bob（至少在輸出中包括他的姓名，項目和角色）。
- 進階：建立應用程式以尋找與實習生擁有相同經理Bob的所有員工。提示：這可能需要多個查詢。
- 進階：建立應用程式以尋找處理專案之所有員工的選取欄位NightFlight。提示：解決這個問題可能需要多個語句。

教程：更新 Amazon Keyspaces 表中的數據

若要更新資料employees_tbl表中的資料，請使用UPDATE陳述式。

UPDATE聲明的一般形式如下。

```
UPDATE table_name SET column_name=new_value WHERE primary_key=value ;
```

> **Tip**
>
> - 您可以使用逗號分隔的column_names和值清單來更新多個資料欄，如下列範例所示。
>
>   ```
>   UPDATE my_table SET col1='new_value_1', col2='new_value2' WHERE id='12345' ;
>   ```
>
> - 如果主索引鍵由多個資料行組成，則所有主索引鍵資料行及其值都必須包含在WHERE子句中。
> - 您不能更新主鍵中的任何列，因為這將改變記錄的主鍵。

更新單一儲存格

使用你的employees_tbl表，給與 ID 的員567-89-0123工加薪。

```
UPDATE "myGSGKeyspace".employees_tbl SET pay_scale=5 WHERE id='567-89-0123' AND
 division='Marketing' ;
```

驗證員工的薪酬表現在是5。

```
SELECT * FROM "myGSGKeyspace".employees_tbl WHERE id='567-89-0123' ;
```

嘗試一下

高級：您的公司聘請了 Bob 實習生。改變他的記錄，使他的角色是'IC'和他的薪水表2。

教學課程：刪除 Amazon Keyspaces 表格中的資料

若要刪除資料employees_tbl表中的資料，請使用DELETE陳述式。

您可以從列或分割區中刪除資料。刪除數據時要小心，因為刪除是不可逆的。

從資料表中刪除一列或所有資料列並不會刪除該表格。因此，您可以使用數據重新填充它。刪除資料表會刪除資料表及其中的所有資料。要再次使用該表，您必須重新創建它並向其中添加數據。刪除密鑰空間將刪除密鑰空間和其中的所有表。要使用密鑰空間和表格，您必須重新創建它們，然後用數據填充它們。

刪除儲存格

從列中刪除欄會從指定的儲存格中移除資料。當您使用SELECT陳述式顯示該資料行時，資料會顯示為 *null*，雖然 Null 值不會儲存在該位置。

刪除一個或多個特定列的一般語法如下。

```
DELETE column_name1[, column_name2...] FROM table_name WHERE condition ;
```

在你的employees_tbl桌子上，你可以看到 CEO 有"None"一個經理。首先，刪除該單元格，以便您不攜帶任何數據。

刪除特定儲存格

1. 執行下列DELETE查詢。

```
DELETE manager_id FROM "myGSGKeyspace".employees_tbl WHERE id='789-01-2345' AND
 division='Executive';
```

2. 確認刪除是否如預期般進行。

```
SELECT * FROM "myGSGKeyspace".employees_tbl WHERE id='789-01-2345' AND
 division='Executive';
```

刪除列

可能有一段時間需要刪除整行，例如當員工退休時。刪除列的一般語法如下。

```
DELETE FROM table_name WHERE condition ;
```

刪除列的步驟

1. 執行下列DELETE查詢。

```
DELETE FROM "myGSGKeyspace".employees_tbl WHERE id='456-78-9012' AND
 division='Engineering';
```

2. 確認刪除是否如預期般進行。

```
SELECT * FROM "myGSGKeyspace".employees_tbl WHERE id='456-78-9012' AND
 division='Engineering';
```

教程步驟 3：刪除 Amazon 密鑰空間中的表和 Keyspaces 間

若要避免針對不需要的資料表和資料收費，請刪除您未使用的所有資料表和金鑰空間。當您刪除資料表時，表格及其資料會被刪除，而您就會停止累計這些資料表的費用。但是，密鑰空間仍然存在。當您刪除一個密鑰空間時，密鑰空間及其所有表格都會被刪除，並停止為它們累積費用。

刪除資料表

您可以使用控制台或 CQL 刪除表。刪除表格時，表格及其所有資料都會被刪除。

使用主控台

下列程序會使用刪除資料表及其所有資料 AWS Management Console。

若要使用主控台刪除資料表

1. 登錄到 AWS Management Console，並打開 Amazon Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。
2. 在導覽窗格中，選擇 Tables (資料表)。
3. 選擇您要刪除之每個表格名稱左邊的方塊。
4. 選擇刪除。
5. 在「刪除表格」畫面上，於方塊**Delete**中輸入。然後，選擇刪除表格。
6. 若要確認表格已刪除，請在導覽窗格中選擇 [表格]，然後確認該`employees_tbl`表格不再列出。

使用定制列表

下列程序會使用 CQL 刪除資料表及其所有資料。

若要使用 CQL 刪除資料表

1. 開啟命令外殼，然後輸入以下內容：

   cqlsh

2. 在密鑰空間提示符（cqlsh:*keyspace_name*>）中輸入以下命令來刪除表格。

```
DROP TABLE IF EXISTS "myGSGKeyspace".employees_tbl ;
```

3. 確認您的資料表已刪除。

```
SELECT * FROM system_schema.tables WHERE keyspace_name = 'myGSGKeyspace' ;
```

   您的表格不應該被列出。

刪除密鑰空間

您可以使用 AWS Management Console 或 CQL 刪除密鑰空間。當您刪除一個密鑰空間時，密鑰空間及其所有表格和數據都將被刪除。

使用 AWS Management Console

下列程序會使用刪除索引鍵空間及其所有資料表和資料。 AWS Management Console

使用控制台刪除密鑰空間

1. 登錄到 AWS Management Console，並打開 Amazon Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。
2. 在導覽窗格中，選擇 [Keyspaces]。
3. 選擇您要刪除的每個密鑰空間名稱左側的方塊。
4. 選擇刪除。
5. 在 [刪除金鑰空間] 畫面上，於方塊**Delete**中輸入。然後，選擇刪除密鑰空間。
6. 若要確認是否已刪除金鑰空myGSGKeyspace間，請在導覽窗格中選擇「Keyspaces 間」，並確認不再列出該金鑰空間。因為您刪除了它的密鑰空間，因此也不應該列出 employees_tbl T ables 下的表格。

使用定制列表

下列程序會使用 CQL 刪除金鑰空間及其所有資料表和資料。

使用 CQL 刪除密鑰空間

1. 開啟命令外殼，然後輸入以下內容：

   cqlsh

2. 在密鑰空間提示符處輸入以下命令來刪除密鑰空間 () cqlsh:*keyspace_name*>。

   ```
   DROP KEYSPACE IF EXISTS "myGSGKeyspace" ;
   ```

3. 確認您的密鑰空間已刪除。

   ```
   SELECT * from system_schema.keyspaces ;
   ```

   您的密鑰空間不應該列出。請注意，由於這是非同步操作，因此在刪除密鑰空間之前可能會有延遲。

遷移到 Amazon Keyspaces

遷移到 Amazon Keyspaces（對於 Apache 卡桑德拉）為企業和組織提供了一系列令人信服的好處。以下是一些關鍵優勢，使 Amazon 密 Keyspaces 成為一個有吸引力的遷移選擇。

- 可擴展性 — Amazon Keyspaces 旨在處理大量工作負載並無縫擴展以適應不斷增長的資料量和流量。使用傳統的卡桑德拉，縮放不是按需執行，需要規劃 future 的高峰。使用 Amazon Keyspaces，您可以根據需求輕鬆擴展或縮減表格，確保應用程式能夠處理突然尖峰的流量，而不會影響效能。
- 效能 — Amazon Keyspaces 提供低延遲資料存取，讓應用程式能夠以優異的速度擷取和處理資料。它的分散式架構可確保讀取和寫入操作分佈在多個節點上，即使在高請求率下也能提供一致的 10 毫秒回應時間。
- 全受管 — Amazon Keyspaces 是由 AWS提供的全受管服務。這表示可以 AWS 處理資料庫管理的作業層面，包括佈建、組態、修補、備份和擴展。這可讓您更專注於開發應用程式，減少資料庫管理工作。
- 無伺服器架構 — Amazon Keyspaces 是無伺服器的。您只需為耗用的容量付費，無需預先佈建容量。您沒有要管理的伺服器，也不需要選擇執行個體。此 pay-per-request 模型提供了成本效益和最低的營運開銷，因為您只需支付消耗的資源費用，而無需佈建和監控容量。
- 結構描述的 NoSQL 彈性 — Amazon Keyspaces 遵循 NoSQL 資料模型，提供結構描述設計的彈性。使用 Amazon Keyspaces，您可以存放結構化、半結構化和非結構化資料，因此非常適合處理各種不斷發展的資料類型。此外，Amazon Keyspaces 會在寫入時執行結構描述驗證，以便集中演進資料模型。這種靈活性使開發週期更快，更容易適應不斷變化的業務需求。
- 高可用性和耐用性 — Amazon Keyspaces 可在多個可用區域內複寫資料 AWS 區域，確保高可用性和資料耐久性。它會自動處理複寫、容錯移轉和復原，將資料遺失或服務中斷的風險降到最低。Amazon Keyspaces 提供高達 99.999% 的可用性 SLA。為了獲得更高的彈性和低延遲的本機讀取，Amazon Keyspaces 提供多區域複寫。
- 安全性和合規性 — Amazon Keyspaces 與整合以 AWS Identity and Access Management 提供精細的存取控制。它提供靜態和傳輸中的加密，有助於提高數據的安全性。Amazon Keyspaces 也符合安全標準和隱私權法律，包括 HIPAA、PCI DSS 和 GDPR，讓您能夠符合法規要求。
- 與 AWS 生態系統集成 — 作為 AWS 生態系統的一部分，Amazon 密鑰空間與其他密鑰空間無縫集成 AWS 服務 AWS CloudFormation，例如 CloudWatch，Amazon 和 AWS CloudTrail. 此整合可讓您建置無伺服器架構、利用基礎架構即程式碼，以及建立即時資料驅動的應用程式。

遷移至 Amazon Keyspaces 的一般考量

- 將移轉分解為較小的元件。

  就原始資料大小而言，請考慮下列移轉單位及其潛在佔用空間。在一或多個階段移轉較少量的資料可能有助於簡化遷移作業。

  - 通過集群-一次遷移所有卡桑德拉數據。對於較小的叢集而言，這種方法可能很好。
  - 按密鑰空間或表格 — 將您的遷移分為密鑰空間或表格群組。此方法可協助您根據每個工作負載的需求分階段移轉資料。
  - 依資料 — 請考慮移轉特定使用者或產品群組的資料，以減少資料大小。
- 根據簡易性優先排定要移轉的資料。

  請考慮是否有可以更輕鬆地遷移的資料，例如在特定時間不會變更的資料、每晚批次工作的資料、離線時間未使用的資料，或來自內部應用程式的資料。

主題

指導從阿帕奇卡桑德拉數據遷移

對於從 Apache 卡桑德拉到 Amazon Keyspaces 的成功遷移，我們建議仔細規劃和可用選項的比較。本主題概述移轉程序的運作方式、可用的工具，以及如何評估不同的移轉策略，以選取最符合您需求的移轉策略。

主題

功能兼容性

考慮遷移之前，阿帕奇卡桑德拉和 Amazon Keyspaces 間之間的功能差異。Amazon Keyspaces 支援所有常用的 Cassandra 資料平面作業，例如建立金鑰空間和表格、讀取資料和寫入資料。然而，有一些卡桑德拉 API Amazon Keyspaces 不支持。如需支援 API 的詳細資訊，請參閱the section called “支

持的卡桑德拉 API，操作，函數和數據類型”。有關 Amazon 密鑰空間和 Apache 卡桑德拉之間的所有功能差異的概述，請參閱。the section called “與阿帕奇卡桑德拉功能差異”

若要將您使用的 Cassandra API 和結構描述與 Amazon Keyspaces 中受支援的功能進行比較，您可以在上執行 Amazon Keyspaces 工具組中提供的相容性指令碼。GitHub

如何使用相容性指令碼

1. 下載兼容性 Python 腳本，GitHub並將其移動到有權訪問您現有的 Apache 卡桑德拉集群的位置。
2. 相容性指令碼使用與CQLSH. 對於--host並--port輸入 IP 地址和端口，您用來連接和運行查詢到集群中的 Cassandra 節點之一。如果您的 Cassandra 群集使用身份驗證，則還需要提供-username和。-password要運行兼容性腳本，您可以使用以下命令。

```
python toolkit-compat-tool.py --host hostname or IP -u "username" -p "password" --
port native transport port
```

估計 Amazon Keyspaces 定價

本節提供了您需要從 Apache Cassandra 表收集的信息的概述，以計算 Amazon Keyspaces 的估計成本。每個表都需要不同的數據類型，需要支持不同的 CQL 查詢，並維護獨特的讀/寫流量。根據表格思考您的需求，與 Amazon Keyspaces space 表格層級資源隔離和讀取/寫入輸送量容量模式保持一致。使用 Amazon Keyspaces，您可以獨立為表格定義讀取/寫入容量和自動擴展政策。瞭解資料表需求可協助您根據功能、成本和移轉工作排定資料表的優先順序。

在遷移之前收集以下卡桑德拉表度量。此資訊有助於估算 Amazon Keyspaces 上工作負載的成本。

- 表格名稱 — 完整金鑰空間和表格名稱的名稱。
- 說明 — 表格的描述，例如表格的使用方式，或儲存在表格中的資料類型。
- 平均每秒讀取數 — 大型時間間隔內，對表格進行座標層次讀取的平均數目。
- 平均每秒寫入次數 — 大型時間間隔內對表格的平均座標層級寫入次數。
- 平均資料列大小 (位元組) — 平均資料列大小 (位元組)。
- 儲存大小 (以 GB 為單位) — 資料表的原始儲存大小。
- 讀取一致性劃分 — 使用最終一致性 (LOCAL_ONE或ONE) 與強一致性 () 的讀取百分比。LOCAL_QUORUM

此表格顯示規劃移轉時需要提取的資料表相關資訊範例。

資料表名稱	描述	平均每秒讀取	平均每秒寫入	平均資料列大小 (位元組	儲存大小 (以 GB 為單位)	讀取一致性細分
我的金鑰空間. 我的資料表	用於存儲購物車歷史記錄	10,000	5,000	2,200	2,000	100% LOCAL_ONE
我的密鑰空間. 我的表 2	用於存儲最新的配置文件信息	20,000	1,000	850	1,000	25% LOCAL_QUORUM 75% LOCAL_ONE

如何收集表格指標

本節提供有關如何從現有的 Cassandra 集群收集必要的表指標逐步說明。這些測量結果包括資料列大小、表格大小，以及每秒讀取/寫入要求 (RPS)。它們可讓您評估 Amazon Keyspaces 表的輸送量容量需求並估算定價。

如何收集卡桑德拉源表上的表指標

1. 確定行大小

   資料列大小對於判斷 Amazon Keyspaces 中的讀取容量和寫入容量使用率非常重要。下圖顯示了在卡桑德拉令牌範圍的典型數據分佈。

您可以使用上可用的行大小採樣器腳本GitHub來收集 Cassandra 集群中每個表的行大小指標。該腳本通過使用cqlsh和計算行大小的最小，最大值，平均值和awk標準偏差從 Apache Cassandra 出口表數據表數據表數據的表數據。行大小採樣器將參數傳遞給cqlsh，因此可以使用相同的參數來連接並從 Cassandra 集群中讀取。

下面的語句是這樣的一個例子。

```
./row-size-sampler.sh 10.22.33.44 9142 \\
   -u "username" -p "password" --ssl
```

如需有關如何在 Amazon Keyspaces 中計算列大小的詳細資訊，請參閱the section called “計算列大小”。

2. 確定表格大小

使用 Amazon Keyspaces，您無需提前佈建儲存空間。Amazon Keyspaces 會持續監控表格的可計費大小，以判斷儲存費用。儲存體按每月 GB 計費。Amazon Keyspaces 表格大小是以單一複本的原始大小 (未壓縮) 為基礎。若要監視 Amazon Keyspaces 中的表格大小，您可以使用指標BillableTableSizeInBytes，該指標會針對中的每個表格顯示。 AWS Management Console

若要估算 Amazon Keyspaces 表的可計費大小，您可以使用以下兩種方法之一：

- 使用平均列大小，並乘以數字或列。

  您可以通過將平均行大小乘以 Cassandra 源表中的行數來估計 Amazon Keyspaces 表的大小。使用上一節的資料列大小範例指令碼擷取平均資料列大小。要捕獲行計數，您可以使用類似的工具dsbulk count來確定源表中的總行數。

- 使用nodetool來收集表格中繼資料。

  Nodetool是在 Apache 卡桑德拉分佈提供了洞察到卡桑德拉過程的狀態，並返回表元數據提供的管理工具。您可以使用nodetool來取樣有關表格大小的中繼資料，並在 Amazon Keyspaces 中推斷表格大小。要使用的命令是nodetool tablestats。表格會傳回表格的大小和壓縮比率。資料表的大小會儲存tablelivespace為表格的大小，您可以將它除以compression ratio。然後通過節點數量將此大小值倍數。最後除以複製因子（通常為三個）。這是計算的完整公式，可用於評估表格大小。

  ```
  ((tablelivespace / compression ratio) * (total number of nodes))/ (replication
   factor)
  ```

讓我們假設您的卡桑德拉集群有 12 個節點。執行此指nodetool tablestats令會傳回一tablelivespace個值為 200 GB 且一個值為 0.5 compression ratio 的空間。密鑰空間具有三個複製係數。這就是此範例的計算方式。

```
(200 GB / 0.5) * (12 nodes)/ (replication factor of 3)
                        = 4,800 GB / 3
                        = 1,600 GB is the table size estimate for Amazon
 Keyspaces
```

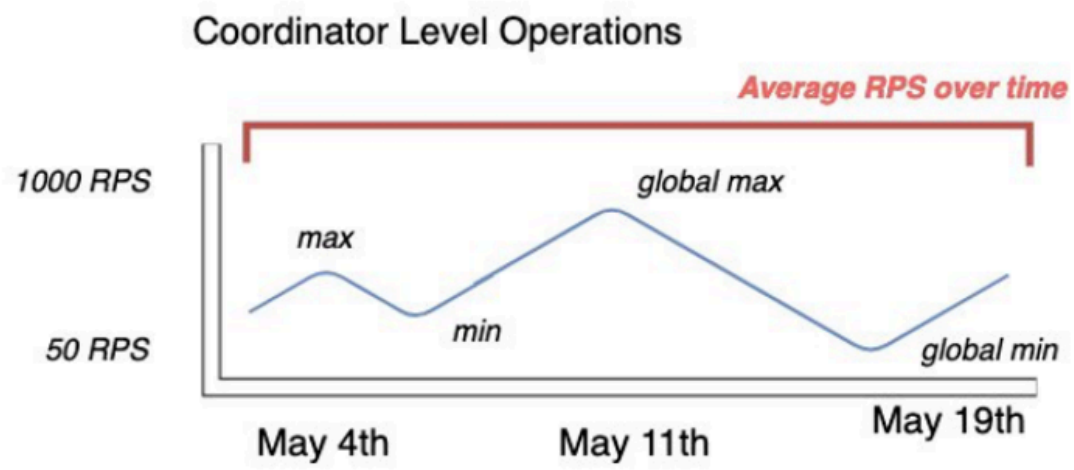
3. 擷取讀取和寫入的次數

若要確定 Amazon Keyspaces 表的容量和擴展需求，請在遷移之前擷取 Cassandra 表格的讀取和寫入請求率。

Amazon Keyspaces 是無伺服器的，您只需按使用量付費。一般而言，Amazon Keyspaces 中讀取/寫入輸送量的價格取決於請求的數量和大小。Amazon Keyspaces 有兩種容量模式：隨需和佈建容量模式。隨需容量模式是一種靈活的計費選項，每秒可處理數千個請求，而無需進行容量規劃。此選項提供讀取和寫入請求的 pay-per-request 定價，因此您只需按使用量付費。如果您選擇佈建的輸送量容量模式，您可以指定應用程式所需的每秒讀取和寫入次數。這有助於您管理 Amazon Keyspaces 的使用情況，使其保持在或低於定義的請求率，以優化價格並保持可預測性。佈建模式提供 auto 動擴充功能，可自動調整佈建的速率，以擴展或縮減規模，以提高營運效率。如需無伺服器資源管理的詳細資訊，請參閱無伺服器資源管理。

由於您分別在 Amazon Keyspaces 中佈建讀取和寫入輸送量容量，因此您需要單獨測量現有表格中讀取和寫入的請求率。

若要從現有 Cassandra 叢集收集最準確的使用率指標，請針對在單一資料中心的所有節點上彙總的資料表，擷取協調員層級讀取和寫入作業的平均每秒要求 (RPS)。擷取至少數週內的平均 RPS，將擷取流量模式中的高峰和山谷，如下圖所示。



你有兩個選項來確定你的卡桑德拉表的讀寫請求率。

- 使用現有的卡桑德拉監控

  您可以使用下表中顯示的度量來觀察讀取和寫入要求。請注意，指標名稱可能會根據您使用的監視工具而變更。

維度	卡桑德拉 JMX 公制
寫入	`org.apache.cassandra.metrics:type=ClientRequest,scope=Write,name=Latency#Count`
讀取	`org.apache.cassandra.metrics:type=ClientRequest,scope=Read,name=Latency#Count`

- 使用 `nodetool`

  使用`nodetool tablestats`和擷`nodetool info`取表格中的平均讀取和寫入作業。`tablestats`返回從節點起始的總讀取和寫入計數。 `nodetool info`在幾秒鐘內提供節點的正常運行時間。若要接收讀取和寫入的每秒平均值，請將讀取和寫入計數除以節點執行時間(以秒為單位)。然後，對於讀取，您將寫入的一致性級別廣告除以複製因子。這些計算以下列公式表示。

每秒平均讀取數的公式：

```
((number of reads * number of nodes in cluster) / read consistency quorum
 (2)) / uptime
```

平均每秒寫入量的公式：

```
((number of writes * number of nodes in cluster) / replication factor of 3) /
 uptime
```

假設我們有一個 12 節點叢集已經啟動了 4 週。 `nodetool info`返回 2,419,200 秒的正常運行時間，並`nodetool tablestats`返回 10 億次寫入和 20 億次讀取。此範例會產生以下計算。

```
((2 billion reads * 12 in cluster) / read consistency quorum (2)) / 2,419,200
 seconds
=  12 billion reads / 2,419,200 seconds
=  4,960 read request per second
                        ((1 billion writes * 12 in cluster) / replication
 factor of 3) / 2,419,200 seconds
=  4 billion writes / 2,419,200 seconds
=  1,653 write request per second
```

4. 決定表格的容量使用率

要估計平均容量利用率，請從平均請求率和 Cassandra 源表的平均行大小開始。

Amazon Keyspaces 使用讀取容量單位 (RCU) 和寫入容量單位 (WCU) 來測量表格讀取和寫入的佈建輸送量容量。針對此估算值，我們使用這些單位來計算移轉後新 Amazon Keyspaces 表格的讀取和寫入容量需求。本主題稍後我們將討論佈建和隨需容量模式之間的選擇如何影響計費。但是為了估計容量使用率，我們假設表格處於佈建模式。

對於大小不超過 4 KB 的資料列，一個 RCU 代表一`LOCAL_QUORUM`個`LOCAL_ONE`讀取要求或兩個讀取要求。如果您需要讀取大於 4 KB 的資料列，讀取作業會使用額外的 RCU。所需的 RCU 總數取決於資料列大小，以及您要使用`LOCAL_QUORUM`還是`LOCAL_ONE`讀取一致性。例如，讀取 8 KB 資料列需要 2 個使用讀取一致性的 RCU；如果您選擇`LOCAL_ONE`讀`LOCAL_QUORUM`取一致性，則需要 1 個 RCU。

一個 WCU 代表一個寫入大小不超過 1 KB 的資料列。所有寫入都使用LOCAL_QUORUM一致性，使用輕量型交易 (LWT) 無需額外付費。如果您需要寫入大於 1 KB 的資料列，則寫入作業會使用額外的 WCU。所需的 WCU 總數取決於資料列大小。例如，如果您的資料列大小為 2 KB，則需要 2 個 WCU 才能執行一個寫入要求。

以下公式可用於估計所需的 RCU 和 WCU。RCU 中的讀取容量可由每秒讀取數乘以每次讀取的資料列數乘以平均資料列大小除以 4KB，然後四捨五入至最接近的整數來決定。

WCU 中的寫入容量可透過將要求數目乘以平均資料列大小除以 1 KB，然後四捨五入至最接近的整數來決定。這是在下面的公式表示。

```
Read requests per second * ROUNDUP((Average Row Size)/4096 per unit) =  RCUs per
 second

Write requests per second * ROUNDUP(Average Row Size/1024 per unit) = WCUs per
 second
```

例如，如果您在卡桑德拉表上執行 4,960 個讀取請求，並且行大小為 2.5KB，則在 Amazon Keyspaces 中需要 4,960 個 RCU。如果您目前正在卡桑德拉表上每秒執行 1,653 個寫入請求，且資料列大小為 2.5KB，則在 Amazon Keyspaces 中每秒需要 4,959 個 WCU。此範例以下列公式表示。

```
4,960 read requests per second * ROUNDUP( 2.5KB /4KB bytes per unit)
= 4,960 read requests per second * 1 RCU
= 4,960 RCUs

1,653 write requests per second * ROUNDUP(2.5KB/1KB per unit)
= 1,653 requests per second * 3 WCUs
= 4,959 WCUs
```

使用eventual consistency可讓您在每個讀取要求上節省最多一半的輸送量容量。每個最終一致的讀取最多可能會消耗 8KB。您可以將先前的計算乘以 0.5 來計算最終一致讀取，如下列公式所示。

```
4,960 read requests per second * ROUNDUP( 2.5KB /4KB per unit) * .5
= 2,480 read request per second * 1 RCU
= 2,480 RCUs
```

5. 計算 Amazon Keyspaces 的每月定價估算

若要根據讀取/寫入容量輸送量估算表格的每月帳單，您可以使用不同的公式計算隨需和佈建模式的定價，並比較表格的選項。

佈建模式 — 讀取和寫入容量使用量會根據每秒的容量單位，按小時費率計費。首先，將該比率除以 0.7，代表預設的自動調度資源目標使用率為 70%。然後按 30 個日曆天，每天 24 小時和區域費率定價多次。此計算摘要在以下公式中。

```
(read capacity per second / .7) * 24 hours * 30 days * regional rate
                (write capacity per second / .7) * 24 hours * 30 days * regional
 rate
```

隨需模式 — 讀取和寫入容量按每個請求費率計費。首先，將請求率乘以 30 個日曆天，每天 24 小時。然後除以一百萬個請求單位。最後，乘以區域速率。此計算摘要在以下公式中。

```
((read capacity per second * 30 * 24 * 60 * 60) / 1 Million read request units) *
 regional rate
                ((write capacity per second * 30 * 24 * 60 * 60) / 1 Million write
 request units) * regional rate
```

選擇移轉策略

一般來說，從 Apache 卡桑德拉遷移到 Amazon Keyspaces 間時，您可以在三種不同的遷移策略之間進行選擇：

- 離線 — 此遷移涉及使用藍色/綠色風格的應用程序遷移部署將數據集從 Cassandra 複製到 Amazon Keyspaces。如果您的應用程式可以在移轉期間容許某些停機時間，此選項可以簡化遷移程序。如需離線移轉的詳細資訊，請參閱

  the section called “離線遷移”.
- 線上 — 這是初期測試樣式部署，通常包含直接寫入應用程式邏輯的雙重寫入。在移轉期間需要零停機時間的應用程式需要資料進行複製，而即時讀取和寫入則從一個資料來源切換到另一個資料來源。
- 混合式 — 這種方法允許以近乎即時的速度複寫變更，但應用程式負責切換讀取和寫入。

在詳細檢閱可用的移轉策略之後，您可以將選項放置在決策樹中，以便根據您的需求和可用資源來簡化程序。

離線遷移到 Amazon Keyspaces

當您可以負擔停機時間來執行遷移時，離線遷移是適合的。企業通常會有修補、大型版本或硬體升級或主要升級的停機時間的維護時間。離線遷移可以使用此窗口複製數據，並在應用程序流量從 Apache Cassandra 切換到 Amazon Keyspaces。離線遷移減少了對應用程序的修改，因為它不需要同時與 Cassandra 和 Amazon Keyspaces 進行通信。此外，在資料流程暫停的情況下，可以複製確切的狀態，而不會維持突變。

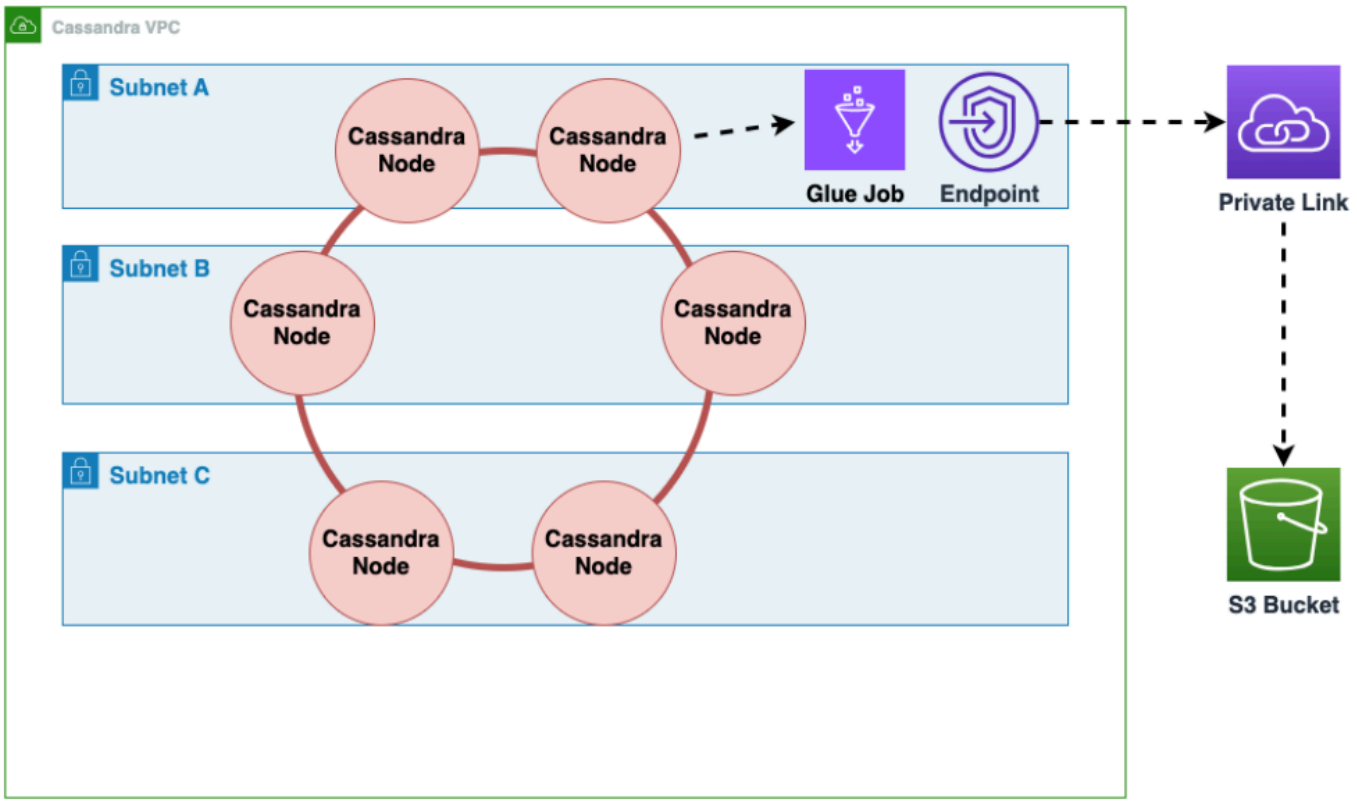
在此範例中，您將 Amazon Simple Storage Service (Amazon S3) 用作離線移轉期間資料的暫存區，以將停機時間降至最低。您可以使用星火卡桑德拉連接器和自動導入您在 Amazon S3 中的鑲木地板格式存儲到 Amazon Keyspaces 表中的數據。 AWS Glue以下部分將展示該過程的高級概述。您可以在 Github 上找到此程序的程式碼範例。

從 Apache 卡桑德拉離線遷移過程使用 Amazon Amazon S3 Keyspaces， AWS Glue 需要以下任務。 AWS Glue

1. 一種 ETL 任務，可擷取和轉換 CQL 資料，並將其存放在 Amazon S3 儲存貯體中。
2. 將資料從儲存貯體匯入 Amazon Keyspaces 的第二個任務。
3. 匯入增量資料的第三項工作。

如何從 Amazon 虛擬私有雲中的亞馬遜 Amazon EC2 上運行的卡桑德拉執行離線遷移到亞馬遜 Keyspaces

1. 首先，您使 AWS Glue 用木地板格式從 Cassandra 導出表數據，並將其保存到 Amazon S3 存儲桶。您需要使用 AWS Glue 連接到運行卡桑德拉的 Amazon EC2 實例所在的 VPC 的連接器來運行任務。 AWS Glue 然後，您可以使用 Amazon S3 私有端點將資料儲存到 Amazon S3 儲存貯體。下圖說明這些步驟。



2. 隨機排列 Amazon S3 儲存貯體中的資料，以改善資料隨機化。均勻匯入的資料允許在目標資料表中產生更多分散式流量。從包含大分割區 (超過 1000 列的分割區) 的 Cassandra 匯出資料時，需要執行此步驟，以避免將資料插入 Amazon 金鑰 Keyspaces 時的熱鍵模式。熱鍵問題導致 `WriteThrottleEvents` Amazon 密 Keyspaces 間，並導致加載時間增加。

3. 使用其他 AWS Glue 任務將資料從 Amazon S3 儲存貯體匯入 Amazon Keyspaces。Amazon S3 儲存貯體中的洗牌資料以實木複合地板格式存放。

將資料遷移到 Amazon Keyspaces 的工具

可以使用不同的工具將數據遷移到 Amazon Keyspaces

- 遷移工具
  - 對於大型移轉，請考慮使用擷取、轉換和載入 (ETL) 工具。您可以使 AWS Glue 用快速有效地執行資料轉換移轉。
  - 要了解如何使用 Apache 卡桑德拉星火連接器將數據寫入 Amazon Keyspaces，請參閱。與阿帕奇星火集成
  - 使用 cqlsh `COPY FROM` 指令快速開始將資料載入 Amazon Keyspaces。cqlsh 隨附於 Apache Cassandra 中，最適合載入小型資料集或測試資料。如需 step-by-step 指示，請參閱the section called “使用 cqlsh 載入資料”。
  - 您也可以使用 DataStax 批量裝載機 Apache 卡桑德拉使用命令將數據加載到 Amazon Keyspaces。`dsbulk`與 cqlsh 相比，DSBulk 提供了更強大的匯入功能，並且可從存放庫中取得。GitHub 如需 step-by-step 指示，請參閱the section called “使用 DSBulk 載入資料”。

主題

- 教學課程：使用 cqlsh 將資料載入 Amazon Keyspaces
- 教學課程：使用 DSBulk 將資料載入 Amazon Keyspaces

教學課程：使用 cqlsh 將資料載入 Amazon Keyspaces

本 step-by-step 教程指導您完成從 Apache 卡桑德拉數據遷移到 Amazon Keyspaces 使用命令。`cqlsh COPY`在此教學課程中，您將執行下列操作：

主題

- 必要條件

必要條件

您必須先完成下列工作，才能開始此自學課程。

1. 如果您尚未這樣做，請 AWS 帳戶 依照中的步驟註冊the section called “設定 AWS Identity and Access Management”。
2. 依照中的步驟建立服務特定認證。the section called “使用主控台產生服務特定認證”
3. 設置卡桑德拉查詢語言外殼 (cqlsh) 連接，並確認您可以按照中的步驟連接到 Amazon Keyspaces。the section called “使用 cqlsh”

步驟 1：建立來源 CSV 檔案和目標資料表

在本自學課程中，我們使用逗號分隔值 (CSV) 檔案，其名稱keyspaces_sample_table.csv做為資料移轉的來源檔案。提供的範例檔案包含名稱為資料表的幾列資料book_awards。

1. 建立來源檔案。您可以選擇以下其中一個選項：
   - 下載下列封存檔案 samplemigration.zip 中包含的範例 CSV 檔案 (keyspaces_sample_table.csv)。解壓縮封存並記下的路徑。keyspaces_sample_table.csv
   - 要使用存儲在 Apache 卡桑德拉數據庫您自己的數據填充 CSV 文件，可以通過使用cqlshCOPY TO語句填充源 CSV 文件，如下面的例子。

     ```
     cqlsh localhost 9042 -u "username" -p "password" --execute
      "COPY mykeyspace.mytable TO 'keyspaces_sample_table.csv' WITH HEADER=true"
     ```

     請確定您建立的 CSV 檔案符合下列需求：

     - 第一列包含欄名稱。

- 來源 CSV 檔案中的欄名稱與目標資料表中的欄名稱相符。
- 資料會以逗號分隔。
- 所有資料值都是有效的 Amazon Keyspaces 資料類型。請參閱 the section called “資料類型”。

2. 在 Amazon 密鑰空間中創建目標 Keyspaces 間和表。

   a. 使用 Connect 到 Amazon 密 Keyspacescqlsh，並將下列範例中的服務端點、使用者名稱和密碼取代為您自己的值。

   ```
   cqlsh cassandra.us-east-2.amazonaws.com 9142 -u "111122223333" -
   p "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY" --ssl
   ```

   b. 使用名稱catalog創建一個新的密鑰空間，如下面的例子。

   ```
   CREATE KEYSPACE catalog WITH REPLICATION = {'class': 'SingleRegionStrategy'};
   ```

   c. 當新的密鑰空間可用時，請使用下面的代碼來創建目標表book_awards。

   ```
   CREATE TABLE "catalog.book_awards" (
      year int,
      award text,
      rank int,
      category text,
      book_title text,
      author text,
      publisher text,
      PRIMARY KEY ((year, award), category, rank)
      );
   ```

   如果 Apache Cassandra 是您的原始數據源，創建具有匹配標題的 Amazon Keyspaces 目標表的簡單方法是從源表中生成CREATE TABLE語句，如下面的語句所示。

   ```
   cqlsh localhost 9042  -u "username" -p "password" --execute "DESCRIBE
    TABLE mykeyspace.mytable;"
   ```

   然後使用與 Cassandra 源表中的描述相匹配的列名和數據類型在 Amazon Keyspaces 中創建目標表。

步驟 2：準備資料

準備來源資料以進行有效傳輸需要兩個步驟。首先，您隨機化資料。在第二個步驟中，您將分析資料以確定適當的cqlsh參數值和所需的表格設定。

隨機化資料

命cqlsh COPY FROM令會以與 CSV 檔案中顯示的相同順序讀取和寫入資料。如果您使用指cqlsh COPY TO令建立來源檔案，則會在 CSV 中以索引鍵排序順序寫入資料。在內部，Amazon 金 Keyspaces 會使用分割區金鑰來分割資料。雖然 Amazon Keyspaces 具有內建邏輯，可協助對相同分區金鑰進行負載平衡請求，但是如果您隨機排序順序，載入資料會更快速且更有效率。這是因為您可以利用 Amazon Keyspaces 寫入不同分割區時所發生的內建負載平衡。

若要將寫入平均分散到分割區中，您必須隨機化來源檔案中的資料。您可以編寫一個應用程序來執行此操作或使用開源工具，例如 Shuf。Shuf 可以在 Linux 發行版、macOS 上（通過在自製軟件中安裝核心軟件）和視窗（通過使用 Linux 的視窗子系統（WSL））上免費獲得。需要一個額外的步驟來防止帶有列名的標題行在此步驟中混洗。

若要在保留標頭的同時隨機化來源檔案，請輸入下列程式碼。

```
tail -n +2 keyspaces_sample_table.csv | shuf -o keyspace.table.csv && (head
 -1 keyspaces_sample_table.csv && cat keyspace.table.csv ) > keyspace.table.csv1 &&
 mv keyspace.table.csv1 keyspace.table.csv
```

Shuf 會將資料重新寫入名為的新 CSV 檔案。keyspace.table.csv您現在可以刪除keyspaces_sample_table.csv檔案，不再需要它。

分析資料

透過分析資料來決定平均和最大資料列大小。

您這樣做的原因如下：

- 平均資料列大小有助於估計要傳輸的資料總量。
- 您需要平均資料列大小來佈建資料上傳所需的寫入容量。
- 您可以確保每行的大小小於 1 MB，這是 Amazon Keyspaces 中的最大行大小。

> **Note**
>
> 此配額指的是行大小，而不是分區大小。與 Apache 卡桑德拉分區不同，Amazon Keyspaces 分區的大小幾乎可以取消綁定。分割索引鍵和叢集資料行需要額外的中繼資料儲存空間，您必須將其新增至資料列的原始大小。如需詳細資訊，請參閱 the section called “計算列大小”。

下列程式碼使用 AWK 來分析 CSV 檔案，並列印平均和最大資料列大小。

```
awk -F, 'BEGIN {samp=10000;max=-1;}{if(NR>1){len=length($0);t+=len;avg=t/
NR;max=(len>max ? len : max)}}NR==samp{exit}END{printf("{lines: %d, average: %d bytes,
 max: %d bytes}\n",NR,avg,max);}' keyspace.table.csv
```

運行此代碼會導致以下輸出。

```
using 10,000 samples:
{lines: 10000, avg: 123 bytes, max: 225 bytes}
```

您可以在本教學課程的下一個步驟中使用平均資料列大小來佈建資料表的寫入容量。

步驟 3：設定表格的輸送量容量

本教學課程說明如何調整 cqlsh，以便在設定的時間範圍內載入資料。因為您知道事先執行多少讀取和寫入，因此請使用佈建的容量模式。完成資料傳輸後，您應該設定表格的容量模式，以符合應用程式的流量模式。若要進一步了解容量管理，請參閱無伺服器資源管理。

使用佈建的容量模式，您可以指定預先佈建至表格的讀取和寫入容量。寫入容量按小時計費，並以寫入容量單位 (WCU) 計量。每個 WCU 都有足夠的寫入容量，可支援每秒寫入 1 KB 的資料。載入資料時，寫入速率必須低於目標資料表上設定的最大 WCU (參數:`write_capacity_units`)。

根據預設，您最多可以在一個表格佈建 40,000 個 WCU，在帳戶中的所有表格中佈建最多 80,000 個 WCU。如果您需要額外容量，可以在 Service Quotas 主控台中要求增加配額。如需配額的詳細資訊，請參閱 配額。

計算插入所需的 WCU 的平均數

每秒插入 1 KB 的資料需要 1 個 WCU。如果您的 CSV 檔案有 36 萬個資料列，而您想要在 1 小時內載入所有資料，您必須每秒寫入 100 個資料列 (36 萬個資料列/60 分鐘/60 秒 = 每秒 100 個資料列)。如果每個資料列最多有 1 KB 的資料，若要每秒插入 100 個資料列，您必須將 100 個 WCU 佈建至資料

表。如果每一列都有 1.5 KB 的資料，您需要兩個 WCU 才能每秒插入一個資料列。因此，若要每秒插入 100 個資料列，您必須佈建 200 個 WCU。

若要判斷每秒需要插入一個資料列的 WCU 數目，請將平均資料列大小 (以位元組為單位) 除以 1024，然後四捨五入至最接近的整數。

例如，如果平均資料列大小為 3000 個位元組，則需要三個 WCU 才能每秒插入一個資料列。

```
ROUNDUP(3000 / 1024) = ROUNDUP(2.93) = 3 WCUs
```

計算資料載入時間和容量

現在您已知道 CSV 檔案中的平均大小和列數，您可以計算在指定時間內載入資料所需的 WCU 數量，以及使用不同的 WCU 設定將所有資料載入 CSV 檔案中所需的大約時間。

例如，如果檔案中的每一列都是 1 KB，而您的 CSV 檔案中有 1,000,000 個資料列，若要在 1 小時內載入資料，您必須在該小時內至少佈建 278 個 WCU 至資料表。

```
1,000,000 rows * 1 KBs = 1,000,000 KBs
1,000,000 KBs / 3600 seconds =277.8 KBs / second = 278 WCUs
```

設定佈建的容量設定

您可以在建立資料表時或使用 ALTER TABLE CQL 指令來設定資料表的寫入容量設定。以下是使用 ALTER TABLE CQL 陳述式變更表格佈建容量設定的語法。

```
ALTER TABLE mykeyspace.mytable WITH custom_properties={'capacity_mode':
{'throughput_mode': 'PROVISIONED', 'read_capacity_units': 100,
 'write_capacity_units': 278}} ;
```

如需完整的語言參考，請參閱the section called “ALTER TABLE”。

步驟 4：配cqlsh COPY FROM置設置

本節概述如何決定的參數值cqlsh COPY FROM。此命cqlsh COPY FROM令會讀取您之前準備好的 CSV 檔案，並使用 CQL 將資料插入 Amazon Keyspaces。該命令分割行，並在一組 Worker 之間分配INSERT操作。每個工作者都會與 Amazon Keyspaces 建立連線，並沿著此通道傳送INSERT請求。

該cqlsh COPY命令沒有內部邏輯，可以在其 Worker 之間平均分配工作。但是，您可以手動配置它，以確保工作均勻分佈。請先檢閱下列 cqlsh 索引鍵參數：

- DELIMI@@ TER — 如果您使用逗號以外的分隔符，則可以設置該參數，該參數默認為逗號。
- INGRATE — 每秒cqlsh COPY FROM嘗試處理的目標資料列數目。如果未設定，則預設值為100,000。
- NUMLSH 處理作業 — cqlsh 為工作建立的子工作者處理作業數目。COPY FROM此設定的最大值為16，預設值為num_cores - 1，其num_cores中是執行 cqlsh 之主機上的處理核心數目。
- MAXBATCHSI ZE — 批次大小決定在單一批次中插入目標資料表的最大列數。如果未設定，cqlsh會使用 20 個插入資料列的批次。
- 區塊大小 — 傳遞給子工作者的工作單位大小。默認情況下，它設置為 5,000。
- 最大嘗試次數 — 重試失敗的工作站區塊的次數上限。達到最大嘗試次數之後，失敗的記錄會寫入新的 CSV 檔案，您可以稍後在調查失敗後再次執行該檔案。

INGESTRATE根據您佈建至目標目的地表格的 WCU 數目來設定。cqlsh COPY FROM命令INGESTRATE的不是限制，而是目標平均值。這意味著它可以（並且通常會）超出您設置的數字。若要允許突發並確定有足夠的容量來處理資料載入要求，請設定INGESTRATE為表格寫入容量的 90%。

```
INGESTRATE = WCUs * .90
```

接下來，將NUMPROCESSES參數設定為等於少於系統核心數的 1。要了解系統的內核數量，可以運行以下代碼。

```
python -c "import multiprocessing; print(multiprocessing.cpu_count())"
```

在本教程中，我們使用以下值。

```
NUMPROCESSES = 4
```

每個程序都會建立一個工作者，每個工作者都會建立與 Amazon Keyspaces 的連線。Amazon Keyspaces 在每個連線上每秒最多可支援 3,000 個 CQL 請求。這表示您必須確定每個 Worker 每秒處理的要求少於 3,000 個。

與此同樣INGESTRATE，工作人員通常超過您設定的數字，並且不受時鐘秒數的限制。因此，若要解決突發問題，請將 cqlsh 參數設定為每個 Worker 以每秒處理 2,500 個要求為目標。若要計算分配給 Worker 的工作量，請使用下列準則。

- 除INGESTRATE以NUMPROCESSES。
- 如果INGESTRATE/NUMPROCESSES> 2,500，則降低INGESTRATE以使此公式成立。

```
INGESTRATE / NUMPROCESSES <= 2,500
```

在您設定設定以最佳化範例資料上傳之前，讓我們先檢閱cqlsh預設設定，並瞭解使用這些設定如何影響資料上傳程序。因為cqlsh COPY FROM使CHUNKSIZE用建立工作區塊 (INSERT陳述式) 來散發給 Worker，因此工作不會自動均勻分配。某些工作人員可能會閒置，具體取決於INGESTRATE設置。

若要在 Worker 之間平均分配工作，並使每個 Worker 保持每秒 2,500 個要求的最佳速率CHUNKSIZE，您必須設定MAXBATCHSIZE、並INGESTRATE變更輸入參數。若要最佳化資料載入期間的網路流量使用率，請選擇接近最大值 30 的值。MAXBATCHSIZE通過更改CHUNKSIZE為 100 和 MAXBATCHSIZE 25，10,000 行平均分佈在四個工人之間（萬/2500 = 4）。

下面的代碼示例說明了這一點。

```
INGESTRATE = 10,000
NUMPROCESSES = 4
CHUNKSIZE = 100
MAXBATCHSIZE. = 25
Work Distribution:
Connection 1 / Worker 1 : 2,500 Requests per second
Connection 2 / Worker 2 : 2,500 Requests per second
Connection 3 / Worker 3 : 2,500 Requests per second
Connection 4 / Worker 4 : 2,500 Requests per second
```

總而言之，在設定cqlsh COPY FROM參數時，請使用下列公式：

- 內容 = 寫入容量 _ 單位 * .90
- 程序數 = 核心數 -1 (預設值)
- 整理/數量進程 = 2,500（這必須是一個真實的陳述。）
- 最大批次大小 = 30 (預設為 20。 Amazon Keyspaces 接受多達 30 個批次。)
- 區塊大小 = (擷取/處理編號)/最大批次大小

現在您已經計算NUMPROCESSESINGESTRATE、、和CHUNKSIZE，就可以載入資料了。

步驟 5：執行命**cqlsh COPY FROM**令

若要執行cqlsh COPY FROM命令，請完成以下步驟。

1. 使用 cqlsh Connect 到 Amazon Keyspaces。
2. 使用以下代碼選擇密鑰空間。

```
USE catalog;
```

3. 將寫入一致性設定為LOCAL_QUORUM。為了確保資料耐久性，Amazon Keyspaces 不允許其他寫入一致性設定。請參閱下面的代碼。

```
CONSISTENCY LOCAL_QUORUM;
```

4. 使用下列程式碼範例準備您的cqlsh COPY FROM語法。

```
COPY book_awards FROM './keyspace.table.csv' WITH HEADER=true
AND INGESTRATE=calculated ingestrate
AND NUMPROCESSES=calculated numprocess
AND MAXBATCHSIZE=20
AND CHUNKSIZE=calculated chunksize;
```

5. 執行先前步驟中準備的陳述式。cqlsh 會回應您已設定的所有設定。

   a. 確保設置與您的輸入相匹配。請參閱以下範例。

```
Reading options from the command line: {'chunksize': '120', 'header': 'true',
 'ingestrate': '36000', 'numprocesses': '15', 'maxbatchsize': '20'}
Using 15 child processes
```

   b. 複查傳輸的資料列數目和目前的平均匯率，如下列範例所示。

```
Processed: 57834 rows; Rate: 6561 rows/s; Avg. rate: 31751 rows/s
```

   c. 當 cqlsh 完成上傳資料時，請檢閱資料載入統計資料的摘要 (讀取的檔案數目、執行階段和略過的資料列數目)，如下列範例所示。

```
15556824 rows imported from 1 files in 8 minutes and 8.321 seconds (0 skipped).
```

在本教學的最後一個步驟中，您已將資料上傳到 Amazon Keyspaces。

> **Important**
>
> 現在您已傳輸資料，請調整目標資料表的容量模式設定，以符合應用程式的常規流量模式。在您變更已佈建容量之前，您會按小時費率產生費用。

故障診斷

資料上傳完成之後，請檢查是否已略過資料列。若要這麼做，請瀏覽至來源 CSV 檔案的來源目錄，然後搜尋具有下列名稱的檔案。

```
import_yourcsvfilename.err.timestamp.csv
```

cqlsh 會將任何略過的資料列寫入具有該名稱的檔案中。如果檔案存在於您的來源目錄中，且其中包含資料，則這些資料列不會上傳到 Amazon Keyspaces。若要重試這些資料列，請先檢查上載期間發生的任何錯誤，然後相應地調整資料。若要重試這些資料列，您可以重新執行程序。

常見錯誤

資料列未載入的最常見原因是容量錯誤和剖析錯誤。

將數據上傳到 Amazon Keyspaces 時發生無效的請求錯誤

在下列範例中，來源資料表包含計數器資料行，這會導致從 cql COPY sh 命令中記錄批次呼叫。Amazon Keyspaces 不支援記錄的批次呼叫。

```
Failed to import 10 rows: InvalidRequest - Error from server: code=2200 [Invalid query]
 message="Only UNLOGGED Batches are supported at this time.",  will retry later,
 attempt 22 of 25
```

若要解決此錯誤，請使用 DSBulk 移轉資料。如需詳細資訊，請參閱 the section called "使用 DSBulk 載入資料"。

將資料上傳到 Amazon Keyspaces 時發生剖析器錯誤

下列範例顯示由於 a 而略過的資料列`ParseError`。

```
Failed to import 1 rows: ParseError - Invalid ... –
```

若要解決此錯誤，您需要確定要匯入的資料與 Amazon Keyspaces 中的資料表結構描述相符。檢閱匯入檔案是否有剖析錯誤。您可以嘗試使用`INSERT`語句使用單行數據來隔離錯誤。

將資料上傳至 Amazon Keyspaces 時發生容量錯誤

```
Failed to import 1 rows: WriteTimeout - Error from server: code=1100 [Coordinator node
 timed out waiting for replica nodes' responses]
```

```
 message="Operation timed out - received only 0 responses." info={'received_responses':
 0, 'required_responses': 2, 'write_type': 'SIMPLE', 'consistency':
 'LOCAL_QUORUM'}, will retry later, attempt 1 of 100
```

Amazon Keyspaces 會使用`ReadTimeout`和`WriteTimeout`例外狀況來指示寫入請求何時因輸送量容量不足而失敗。為了協助診斷容量不足的例外狀況，Amazon Keyspaces 會在 Amazon CloudWatch 中發佈`WriteThrottleEvents`和`ReadThrottledEvents`指標。如需詳細資訊，請參閱 the section called “使用監控 CloudWatch”。

將數據上傳到 Amazon Keyspaces 時出現 cqlsh 錯誤

若要協助疑難排解 cqlsh 錯誤，請使用旗標重新執行失敗的命令。`--debug`

使用不相容版本的 cqlsh 時，您會看到下列錯誤。

```
AttributeError: 'NoneType' object has no attribute 'is_up'
Failed to import 3 rows: AttributeError - 'NoneType' object has no attribute 'is_up',
 given up after 1 attempts
```

執行下列命令，確認已安裝正確版本的 cqlsh。

```
cqlsh --version
```

您應該看到類似下面的輸出內容。

```
cqlsh 5.0.1
```

如果您使用的是 Windows，請將的所有執行個體`cqlsh`取代為`cqlsh.bat`. 例如，若要在 Windows 中檢查 cqlsh 的版本，請執行下列命令。

```
cqlsh.bat --version
```

cqlsh 用戶端從伺服器接收到任何類型的連續三個錯誤之後，與 Amazon Keyspaces 的連線會失敗。cqlsh 用戶端失敗，並顯示下列訊息。

```
Failed to import 1 rows: NoHostAvailable - , will retry later, attempt 3 of 100
```

若要解決此錯誤，您需要確定要匯入的資料與 Amazon Keyspaces 中的資料表結構描述相符。檢閱匯入檔案是否有剖析錯誤。您可以使用 INSERT 陳述式來隔離錯誤，嘗試使用單一資料列的資料。

用戶端會自動嘗試重新建立連線。

教學課程：使用 DSBulk 將資料載入 Amazon Keyspaces

本 step-by-step 教程指導您完成從阿帕奇卡桑德拉遷移數據到 Amazon Keyspaces 使用 DataStax 批量加載器 (DSBulk) 可用. GitHub 在本教學中，您會完成下列步驟：

主題

必要條件

您必須先完成下列工作，才能開始此自學課程。

1. 如果您尚未這麼做，請依照下列步驟註冊 AWS 帳戶the section called “設定 AWS Identity and Access Management”。
2. 依照中的步驟建立認證the section called “身分 AWS 驗證的 IAM 登入資”。
3. 建立 JKS 信任存放區檔案。

   a. 使用下列指令下載 Starfield 數位憑證，並儲存在`sf-class2-root.crt`本機或主目錄中。

   ```
   curl https://certs.secureserver.net/repository/sf-class2-root.crt -O
   ```

   > **Note**
   >
   > 您也可以使用 Amazon 數位憑證連線到 Amazon Keyspaces，如果您的用戶端成功連線至 Amazon Keyspaces，則可以繼續這麼做。Starfield 憑證為使用舊版憑證授權單位的用戶端提供額外的向後相容性。

   b. 將星空數位憑證轉換為信任庫檔案。

   ```
   openssl x509 -outform der -in sf-class2-root.crt -out temp_file.der
   ```

```
keytool -import -alias cassandra -keystore cassandra_truststore.jks -file
 temp_file.der
```

在此步驟中，您需要為金鑰庫建立密碼並信任此憑證。互動式指令看起來像這樣。

```
Enter keystore password:
Re-enter new password:
Owner: OU=Starfield Class 2 Certification Authority, O="Starfield Technologies,
 Inc.", C=US
Issuer: OU=Starfield Class 2 Certification Authority, O="Starfield
 Technologies, Inc.", C=US
Serial number: 0
Valid from: Tue Jun 29 17:39:16 UTC 2004 until: Thu Jun 29 17:39:16 UTC 2034
Certificate fingerprints:
  MD5:  32:4A:4B:BB:C8:63:69:9B:BE:74:9A:C6:DD:1D:46:24
  SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
  SHA256:
 14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB
Signature algorithm name: SHA1withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3
Extensions:
#1: ObjectId: 2.5.29.35 Criticality=false
AuthorityKeyIdentifier [
KeyIdentifier [
0000: BF 5F B7 D1 CE DD 1F 86   F4 5B 55 AC DC D7 10 C2  ._.......[U.....
0010: 0E A9 88 E7                                        ....
]
[OU=Starfield Class 2 Certification Authority, O="Starfield Technologies,
 Inc.", C=US]
SerialNumber: [    00]
]
#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
  CA:true
  PathLen:2147483647
]
#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: BF 5F B7 D1 CE DD 1F 86   F4 5B 55 AC DC D7 10 C2  ._.......[U.....
0010: 0E A9 88 E7                                        ....
]
```

```
]
Trust this certificate? [no]:  y
```

4. 設置卡桑德拉查詢語言外殼 (cqlsh) 連接，並確認您可以按照中的步驟連接到 Amazon Keyspaces。the section called “使用 cqlsh”
5. 下載並安裝 DSBulk。

   a. 要下載 DSBulk，您可以使用以下代碼。

   ```
   curl -OL https://downloads.datastax.com/dsbulk/dsbulk-1.8.0.tar.gz
   ```

   b. 然後解壓縮 tar 文件並將 DSBulk 添加到您的，如以下示PATH例所示。

   ```
   tar -zxvf dsbulk-1.8.0.tar.gz
   # add the DSBulk directory to the path
   export PATH=$PATH:./dsbulk-1.8.0/bin
   ```

   c. 創建一個application.conf文件來存儲 DSBulk 使用的設置。您可以將下列範例儲存為./dsbulk_keyspaces.conf。如果您不在本地節點上，例如 DNS 名稱或 IP 地址，請localhost用本地 Cassandra 集群的聯繫點替換。記下檔案名稱和路徑，因為您稍後需要在dsbulk load命令中指定此名稱。

   ```
   datastax-java-driver {
     basic.contact-points = [ "localhost"]
     advanced.auth-provider {
           class = software.aws.mcs.auth.SigV4AuthProvider
           aws-region = us-east-1
     }
   }
   ```

   d. 若要啟用 Sigv4 支援，請從下列範例中下載著色jar檔案GitHub並將其放置在 DSBulk lib 資料夾中。

   ```
   curl -O -L https://github.com/aws/aws-sigv4-auth-cassandra-java-driver-plugin/
   releases/download/4.0.6-shaded-v2/aws-sigv4-auth-cassandra-java-driver-
   plugin-4.0.6-shaded.jar
   ```

步驟 1：建立來源 CSV 檔案和目標資料表

在本自學課程中，我們使用逗號分隔值 (CSV) 檔案，其名稱keyspaces_sample_table.csv做為資料移轉的來源檔案。提供的範例檔案包含名稱為資料表的幾列資料book_awards。

1. 建立來源檔案。您可以選擇以下其中一個選項：
   - 下載下列封存檔案 samplemigration.zip 中包含的範例 CSV 檔案 (keyspaces_sample_table.csv)。解壓縮封存並記下的路徑。keyspaces_sample_table.csv
   - 要使用存儲在 Apache 卡桑德拉數據庫您自己的數據填充 CSV 文件，可以通過使用如下面的dsbulk unload例子填充源 CSV 文件。

     ```
     dsbulk unload -k mykeyspace -t mytable -f ./my_application.conf
      > keyspaces_sample_table.csv
     ```

     請確定您建立的 CSV 檔案符合下列需求：
     - 第一列包含欄名稱。
     - 來源 CSV 檔案中的欄名稱與目標資料表中的欄名稱相符。
     - 資料會以逗號分隔。
     - 所有資料值都是有效的 Amazon Keyspaces 資料類型。請參閱the section called “資料類型”。
2. 在 Amazon 密鑰空間中創建目標 Keyspaces 間和表。
   a. 使用 Connect 到 Amazon 密 Keyspacescqlsh，並將下列範例中的服務端點、使用者名稱和密碼取代為您自己的值。

      ```
      cqlsh cassandra.us-east-2.amazonaws.com 9142 -u "111122223333" -
      p "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY" --ssl
      ```

   b. 使用名稱catalog創建一個新的密鑰空間，如下面的例子。

      ```
      CREATE KEYSPACE catalog WITH REPLICATION = {'class': 'SingleRegionStrategy'};
      ```

   c. 新的密鑰空間狀態為可用之後，請使用下列程式碼來建立目標資料表book_awards。若要進一步瞭解非同步資源建立以及如何檢查資源是否可用，請參閱the section called “創建密鑰空間”。

```
CREATE TABLE catalog.book_awards (
   year int,
   award text,
   rank int,
   category text,
   book_title text,
   author text,
   publisher text,
   PRIMARY KEY ((year, award), category, rank)
   );
```

如果 Apache Cassandra 是您的原始數據源，創建具有匹配標題 Amazon Keyspaces 目標表的簡單方法是從源表中生成的CREATE TABLE語句，如下面的語句。

```
cqlsh localhost 9042  -u "username" -p "password" --execute "DESCRIBE
 TABLE mykeyspace.mytable;"
```

然後使用與 Cassandra 源表中的描述相匹配的列名和數據類型在 Amazon Keyspaces 中創建目標表。

步驟 2：準備資料

準備來源資料以進行有效傳輸需要兩個步驟。首先，您隨機化資料。在第二個步驟中，您將分析資料以確定適當的dsbulk參數值和所需的表格設定。

隨機化資料

命dsbulk令會以與 CSV 檔案中顯示的相同順序讀取和寫入資料。如果您使用指dsbulk令建立來源檔案，則會在 CSV 中以索引鍵排序順序寫入資料。在內部，Amazon 金 Keyspaces 會使用分割區金鑰來分割資料。雖然 Amazon Keyspaces 具有內建邏輯，可協助對相同分區金鑰進行負載平衡請求，但是如果您隨機排序順序，載入資料會更快速且更有效率。這是因為您可以利用 Amazon Keyspaces 寫入不同分區時發生的內建負載平衡。

若要將寫入平均分散到分割區中，您必須隨機化來源檔案中的資料。您可以編寫一個應用程序來執行此操作或使用開源工具，例如 Shuf。Shuf 可以在 Linux 發行版、macOS 上（通過在自製軟件中安裝核心軟件）和視窗（通過使用 Linux 的視窗子系統（WSL））上免費獲得。需要一個額外的步驟來防止帶有列名的標題行在此步驟中混洗。

若要在保留標頭的同時隨機化來源檔案，請輸入下列程式碼。

```
tail -n +2 keyspaces_sample_table.csv | shuf -o keyspace.table.csv && (head
 -1 keyspaces_sample_table.csv && cat keyspace.table.csv ) > keyspace.table.csv1 &&
 mv keyspace.table.csv1 keyspace.table.csv
```

Shuf 會將資料重新寫入名為的新 CSV 檔案。keyspace.table.csv您現在可以刪除keyspaces_sample_table.csv檔案，不再需要它。

分析資料

透過分析資料來決定平均和最大資料列大小。

您這樣做的原因如下：

- 平均資料列大小有助於估計要傳輸的資料總量。
- 您需要平均資料列大小來佈建資料上傳所需的寫入容量。
- 您可以確保每行的大小小於 1 MB，這是 Amazon Keyspaces 中的最大行大小。

> **Note**
>
> 此配額指的是行大小，而不是分區大小。與 Apache 卡桑德拉分區不同，Amazon Keyspaces 分區的大小幾乎可以取消綁定。分割索引鍵和叢集資料行需要額外的中繼資料儲存空間，您必須將其新增至資料列的原始大小。如需詳細資訊，請參閱 the section called “計算列大小”。

下列程式碼使用 AWK 來分析 CSV 檔案，並列印平均和最大資料列大小。

```
awk -F, 'BEGIN {samp=10000;max=-1;}{if(NR>1){len=length($0);t+=len;avg=t/
NR;max=(len>max ? len : max)}}NR==samp{exit}END{printf("{lines: %d, average: %d bytes,
 max: %d bytes}\n",NR,avg,max);}' keyspace.table.csv
```

運行此代碼會導致以下輸出。

```
using 10,000 samples:
{lines: 10000, avg: 123 bytes, max: 225 bytes}
```

請確定您的最大資料列大小不超過 1 MB。如果是這樣，則必須分解該行或壓縮數據以使行大小低於 1 MB。在本教學課程的下一個步驟中，您會使用平均資料列大小來佈建資料表的寫入容量。

步驟 3：設定表格的輸送量容量

本教學課程說明如何調整 DSBulk，以便在設定的時間範圍內載入資料。因為您知道事先執行多少讀取和寫入，因此請使用佈建的容量模式。完成資料傳輸後，您應該設定表格的容量模式，以符合應用程式的流量模式。若要進一步了解容量管理，請參閱無伺服器資源管理。

使用佈建的容量模式，您可以指定預先佈建至表格的讀取和寫入容量。寫入容量按小時計費，並以寫入容量單位 (WCU) 計量。每個 WCU 都有足夠的寫入容量，可支援每秒寫入 1 KB 的資料。載入資料時，寫入速率必須低於目標資料表上設定的最大 WCU (參數:`write_capacity_units`)。

根據預設，您最多可以在一個表格佈建 40,000 個 WCU，在帳戶中的所有表格中佈建最多 80,000 個 WCU。如果您需要額外容量，可以在 Service Quotas 主控台中要求增加配額。如需配額的詳細資訊，請參閱 配額。

計算插入所需的 WCU 的平均數

每秒插入 1 KB 的資料需要 1 個 WCU。如果您的 CSV 檔案有 36 萬個資料列，而您想要在 1 小時內載入所有資料，您必須每秒寫入 100 個資料列 (36 萬列/60 分鐘/60 秒 = 每秒 100 個資料列)。如果每個資料列最多有 1 KB 的資料，若要每秒插入 100 個資料列，您必須將 100 個 WCU 佈建至資料表。如果每一列都有 1.5 KB 的資料，您需要兩個 WCU 才能每秒插入一個資料列。因此，若要每秒插入 100 個資料列，您必須佈建 200 個 WCU。

若要判斷每秒需要插入一個資料列的 WCU 數目，請將平均資料列大小 (以位元組為單位) 除以 1024，然後四捨五入至最接近的整數。

例如，如果平均資料列大小為 3000 個位元組，則需要三個 WCU 才能每秒插入一個資料列。

```
ROUNDUP(3000 / 1024) = ROUNDUP(2.93) = 3 WCUs
```

計算資料載入時間和容量

現在您已知道 CSV 檔案中的平均大小和列數，您可以計算在指定時間內載入資料所需的 WCU 數量，以及使用不同的 WCU 設定將所有資料載入 CSV 檔案中所需的大約時間。

例如，如果檔案中的每一列都是 1 KB，而您的 CSV 檔案中有 1,000,000 個資料列，若要在 1 小時內載入資料，您必須在該小時內至少佈建 278 個 WCU 至資料表。

```
1,000,000 rows * 1 KBs = 1,000,000 KBs
1,000,000 KBs / 3600 seconds =277.8 KBs / second = 278 WCUs
```

設定佈建的容量設定

您可以在建立資料表時或使用ALTER TABLE指令來設定資料表的寫入容量設定。以下是使用ALTER TABLE命令更改表格佈建容量設定的語法。

```
ALTER TABLE catalog.book_awards WITH custom_properties={'capacity_mode':
{'throughput_mode': 'PROVISIONED', 'read_capacity_units': 100, 'write_capacity_units':
 278}} ;
```

如需完整的語言參考資料，請參閱the section called “CREATE TABLE”和the section called “ALTER TABLE”。

步驟 4：配DSBulk置設置

本節概述設定 DSBulk 以將資料上傳到 Amazon Keyspaces 所需的步驟。您可以使用組態檔來設定 DSBulk。您可以直接從命令列指定組態檔案。

1. 為遷移到 Amazon Keyspaces 創建 DSBulk 配置文件，在此示例中，我們使用文件名。dsbulk_keyspaces.conf在 DSBulk 組態檔案中指定下列設定。

   a. *PlainTextAuthProvider*— 使用PlainTextAuthProvider類別建立驗證提供者。ServiceUserName且ServicePassword應該符合您在the section called “建立認證”產生服務特定認證時所取得的使用者名稱和密碼，方法是遵循中的步驟。

   b. *local-datacenter*— 將值設定local-datacenter為您 AWS 區域 要連線到的。例如，如果應用程式正在連線到cassandra.us-east-2.amazonaws.com，則將本機資料中心設定為us-east-2。對於所有可用的 AWS 區域，請參閱the section called “服務端點”。若要避免複本，請slow-replica-avoidance將設定為false。

   c. *SSLEngineFactory*— 若要設定 SSL/TLS，請在SSLEngineFactory組態檔案中新增區段以指定類別的單行來初始化。class = DefaultSslEngineFactory提供您先前建立的路徑cassandra_truststore.jks和密碼。

   d. *consistency*— 將一致性層級設定為LOCAL QUORUM。不支援其他寫入一致性層級，如需詳細資訊，請參閱the section called “支持的卡桑德拉一致性級別”。

   e. 您可以在 Java 驅動程式中設定每個集區的連線數目。在此範例中，設advanced.connection.pool.local.size定為 3。

   以下是完整的範例組態檔案。

```
datastax-java-driver {
basic.contact-points = [ "cassandra.us-east-2.amazonaws.com:9142"]
advanced.auth-provider {
```

```
    class = PlainTextAuthProvider
    username = "ServiceUserName"
    password = "ServicePassword"
}

basic.load-balancing-policy {
    local-datacenter = "us-east-2"
    slow-replica-avoidance = false
}

basic.request {
    consistency = LOCAL_QUORUM
    default-idempotence = true
}
advanced.ssl-engine-factory {
    class = DefaultSslEngineFactory
    truststore-path = "./cassandra_truststore.jks"
    truststore-password = "my_password"
    hostname-validation = false
  }
advanced.connection.pool.local.size = 3
}
```

2. 檢閱 DSBulk `load` 命令的參數。

   a. *`executor.maxPerSecond`*— load 命令每秒嘗試同時處理的最大列數。如果未設定，則停用此設定時為 -1。

      `executor.maxPerSecond`根據您佈建至目標目的地表格的 WCU 數目來設定。`load`命令`executor.maxPerSecond`的不是限制，而是目標平均值。這意味著它可以（並且通常會）超出您設置的數字。若要允許突發並確定有足夠的容量來處理資料載入要求，請設定`executor.maxPerSecond`為表格寫入容量的 90%。

      ```
      executor.maxPerSecond = WCUs * .90
      ```

      在本自學課程中，我們設`executor.maxPerSecond`定為 5。

      > **Note**
      >
      > 如果您使用的是 DSBulk 1.6.0 或更高版本，則可以改用`dsbulk.engine.maxConcurrentQueries`。

b. 設定 DSBulk `load` 命令的這些其他參數。

- *`batch-mode`*— 此參數告訴系統按分區鍵對操作進行分組。我們建議禁用批處理模式，因為它可能導致熱鍵情況和原因`WriteThrottleEvents`。
- *`driver.advanced.retry-policy-max-retries`*— 這會決定重試失敗查詢的次數。如果未設定，預設值為 10。您可以根據需要調整此值。
- *`driver.basic.request.timeout`*— 系統等待查詢傳回的時間 (以分鐘為單位)。如果未設定，預設值為「5 分鐘」。您可以根據需要調整此值。

步驟 5：執行 DSBulk **`load`** 命令

在本教學的最後一個步驟中，您將資料上傳到 Amazon Keyspaces。

若要執行 DSBulk `load` 命令，請完成以下步驟。

1. 運行以下代碼，將 CSV 文件中的數據上傳到 Amazon 密 Keyspaces 表。請務必更新您先前建立之應用程式組態檔案的路徑。

```
dsbulk load -f ./dsbulk_keyspaces.conf  --connector.csv.url keyspace.table.csv
 -header true --batch.mode DISABLED --executor.maxPerSecond 5 --
driver.basic.request.timeout "5 minutes" --driver.advanced.retry-policy.max-
retries 10 -k catalog -t book_awards
```

2. 輸出包括記錄檔的位置，其中詳細說明成功和失敗的作業。檔案儲存在下列目錄中。

```
Operation directory: /home/user_name/logs/UNLOAD_20210308-202317-801911
```

3. 記錄檔項目將包含量度，如下列範例所示。檢查以確保列數與 csv 檔案中的列數一致。

```
total | failed | rows/s | p50ms | p99ms | p999ms
   200 |      0 |    200 | 21.63 | 21.89 |  21.89
```

> **Important**
>
> 現在您已傳輸資料，請調整目標資料表的容量模式設定，以符合應用程式的常規流量模式。在您變更已佈建容量之前，您會按小時費率產生費用。如需更多詳細資訊，請參閱 the section called “讀/寫容量模式”。

使 AWS 用 SDK 的 Amazon Keyspaces 的代碼示例

下列程式碼範例說明如何搭配 AWS 軟體開發套件 (SDK) 使用 Amazon Keyspaces。

Actions 是大型程式的程式碼摘錄，必須在內容中執行。雖然動作會告訴您如何呼叫個別服務函數，但您可以在其相關情境和跨服務範例中查看內容中的動作。

Scenarios (案例) 是向您展示如何呼叫相同服務中的多個函數來完成特定任務的程式碼範例。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Keyspaces 與 SDK AWS。此主題也包含入門相關資訊和舊版 SDK 的詳細資訊。

開始使用

你好 Amazon Keyspaces

下列程式碼範例說明如何開始使用 Amazon Keyspaces。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
namespace KeyspacesActions;

public class HelloKeyspaces
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Keyspaces (for Apache
 Cassandra).
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
```

```
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonKeyspaces>()
                .AddTransient<KeyspacesWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<HelloKeyspaces>();

        var keyspacesClient =
 host.Services.GetRequiredService<IAmazonKeyspaces>();
        var keyspacesWrapper = new KeyspacesWrapper(keyspacesClient);

        Console.WriteLine("Hello, Amazon Keyspaces! Let's list your keyspaces:");
        await keyspacesWrapper.ListKeyspaces();
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListKeyspaces中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.KeyspaceSummary;
import software.amazon.awssdk.services.keyspaces.model.KeyspacesException;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesRequest;
import software.amazon.awssdk.services.keyspaces.model.ListKeyspacesResponse;
```

```
import java.util.List;

/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloKeyspaces {
    public static void main(String[] args) {
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
                .region(region)
                .build();

        listKeyspaces(keyClient);
    }

    public static void listKeyspaces(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest =
 ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();

            ListKeyspacesResponse response =
 keyClient.listKeyspaces(keyspacesRequest);
            List<KeyspaceSummary> keyspaces = response.keyspaces();
            for (KeyspaceSummary keyspace : keyspaces) {
                System.out.println("The name of the keyspace is " +
 keyspace.keyspaceName());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考ListKeyspaces中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/**
Before running this Kotlin code example, set up your development environment,
 including your credentials.

For more information, see the following documentation topic:

https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
*/

suspend fun main() {
    listKeyspaces()
}

suspend fun listKeyspaces() {
    val keyspacesRequest =
        ListKeyspacesRequest {
            maxResults = 10
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.listKeyspaces(keyspacesRequest)
        response.keyspaces?.forEach { keyspace ->
            println("The name of the keyspace is ${keyspace.keyspaceName}")
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListKeyspaces中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
import boto3


def hello_keyspaces(keyspaces_client):
    """
    Use the AWS SDK for Python (Boto3) to create an Amazon Keyspaces (for Apache
 Cassandra)
    client and list the keyspaces in your account.
    This example uses the default settings specified in your shared credentials
    and config files.

    :param keyspaces_client: A Boto3 Amazon Keyspaces Client object. This object
 wraps
                             the low-level Amazon Keyspaces service API.
    """
    print("Hello, Amazon Keyspaces! Let's list some of your keyspaces:\n")
    for ks in keyspaces_client.list_keyspaces(maxResults=5).get("keyspaces", []):
        print(ks["keyspaceName"])
        print(f"\t{ks['resourceArn']}")


if __name__ == "__main__":
    hello_keyspaces(boto3.client("keyspaces"))
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ListKeyspaces中的 Python (博托 3) API 參考。

程式碼範例

使 AWS 用 SDK 對 Amazon Keyspaces 的操作

下列程式碼範例示範如何使用 AWS SDK 執行個別 Amazon Keyspaces 動作。這些摘錄會呼叫 Amazon Keyspaces API，是來自必須在內容中執行的大型程式碼摘錄。每個範例都包含一個連結 GitHub，您可以在其中找到設定和執行程式碼的指示。

下列範例僅包含最常使用的動作。有關完整列表，請參閱 Amazon Keyspaces（阿帕奇卡桑德拉）API 參考。

範例

- 搭RestoreTable配 AWS 開發套件或 CLI 使用
- 搭UpdateTable配 AWS 開發套件或 CLI 使用

搭CreateKeyspace配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CreateKeyspace。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用密鑰空間和表

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/// <summary>
/// Create a new keyspace.
/// </summary>
/// <param name="keyspaceName">The name for the new keyspace.</param>
/// <returns>The Amazon Resource Name (ARN) of the new keyspace.</returns>
public async Task<string> CreateKeyspace(string keyspaceName)
{
    var response =
        await _amazonKeyspaces.CreateKeyspaceAsync(
            new CreateKeyspaceRequest { KeyspaceName = keyspaceName });
    return response.ResourceArn;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考CreateKeyspace中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   public static void createKeySpace(KeyspacesClient keyClient, String
keyspaceName) {
       try {
           CreateKeyspaceRequest keyspaceRequest =
CreateKeyspaceRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           CreateKeyspaceResponse response =
keyClient.createKeyspace(keyspaceRequest);
           System.out.println("The ARN of the KeySpace is " +
response.resourceArn());

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考CreateKeyspace中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK CreateKeyspace中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
```

```
        return cls(keyspaces_client)


    def create_keyspace(self, name):
        """
        Creates a keyspace.

        :param name: The name to give the keyspace.
        :return: The Amazon Resource Name (ARN) of the new keyspace.
        """
        try:
            response = self.keyspaces_client.create_keyspace(keyspaceName=name)
            self.ks_name = name
            self.ks_arn = response["resourceArn"]
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.ks_arn
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件CreateKeyspace中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Keyspaces 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭CreateTable配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用CreateTable。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用密鑰空間和表

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Create a new Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace where the table will be
created.</param>
    /// <param name="schema">The schema for the new table.</param>
    /// <param name="tableName">The name of the new table.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new table.</returns>
    public async Task<string> CreateTable(string keyspaceName, SchemaDefinition
schema, string tableName)
    {
        var request = new CreateTableRequest
        {
            KeyspaceName = keyspaceName,
            SchemaDefinition = schema,
            TableName = tableName,
            PointInTimeRecovery = new PointInTimeRecovery { Status =
PointInTimeRecoveryStatus.ENABLED }
        };

        var response = await _amazonKeyspaces.CreateTableAsync(request);
        return response.ResourceArn;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考CreateTable中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
       try {
           // Set the columns.
           ColumnDefinition defTitle = ColumnDefinition.builder()
                   .name("title")
                   .type("text")
                   .build();

           ColumnDefinition defYear = ColumnDefinition.builder()
                   .name("year")
                   .type("int")
                   .build();

           ColumnDefinition defReleaseDate = ColumnDefinition.builder()
                   .name("release_date")
                   .type("timestamp")
                   .build();

           ColumnDefinition defPlot = ColumnDefinition.builder()
                   .name("plot")
                   .type("text")
                   .build();

           List<ColumnDefinition> colList = new ArrayList<>();
           colList.add(defTitle);
           colList.add(defYear);
           colList.add(defReleaseDate);
           colList.add(defPlot);

           // Set the keys.
           PartitionKey yearKey = PartitionKey.builder()
```

```
                .name("year")
                .build();

        PartitionKey titleKey = PartitionKey.builder()
                .name("title")
                .build();

        List<PartitionKey> keyList = new ArrayList<>();
        keyList.add(yearKey);
        keyList.add(titleKey);

        SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                .partitionKeys(keyList)
                .allColumns(colList)
                .build();

        PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
                .status(PointInTimeRecoveryStatus.ENABLED)
                .build();

        CreateTableRequest tableRequest = CreateTableRequest.builder()
                .keyspaceName(keySpace)
                .tableName(tableName)
                .schemaDefinition(schemaDefinition)
                .pointInTimeRecovery(timeRecovery)
                .build();

        CreateTableResponse response = keyClient.createTable(tableRequest);
        System.out.println("The table ARN is " + response.resourceArn());

    } catch (KeyspacesException e) {
        System.err.println(e.awsErrorDetails().errorMessage());
        System.exit(1);
    }
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考CreateTable中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
```

```
    colList.add(defReleaseDate)
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)

    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK CreateTable中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def create_table(self, table_name):
        """
        Creates a table in the  keyspace.
        The table is created with a schema for storing movie data
        and has point-in-time recovery enabled.

        :param table_name: The name to give the table.
        :return: The ARN of the new table.
        """
        try:
            response = self.keyspaces_client.create_table(
```

```
                keyspaceName=self.ks_name,
                tableName=table_name,
                schemaDefinition={
                    "allColumns": [
                        {"name": "title", "type": "text"},
                        {"name": "year", "type": "int"},
                        {"name": "release_date", "type": "timestamp"},
                        {"name": "plot", "type": "text"},
                    ],
                    "partitionKeys": [{"name": "year"}, {"name": "title"}],
                },
                pointInTimeRecovery={"status": "ENABLED"},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["resourceArn"]
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件CreateTable中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Keyspaces 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteKeyspace配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteKeyspace。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用密鑰空間和表

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Delete an existing keyspace.
    /// </summary>
    /// <param name="keyspaceName"></param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.DeleteKeyspaceAsync(
            new DeleteKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DeleteKeyspace中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
        try {
```

```
            DeleteKeyspaceRequest deleteKeyspaceRequest =
 DeleteKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            keyClient.deleteKeyspace(deleteKeyspaceRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DeleteKeyspace中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteKeyspace中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def delete_keyspace(self):
        """
        Deletes the keyspace.
        """
        try:
            self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
            self.ks_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DeleteKeyspace中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Keyspaces 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭DeleteTable配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用DeleteTable。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用密鑰空間和表

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Delete an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTable(string keyspaceName, string tableName)
    {
        var response = await _amazonKeyspaces.DeleteTableAsync(
```

```
            new DeleteTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考DeleteTable中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public static void deleteTable(KeyspacesClient keyClient, String
keyspaceName, String tableName) {
        try {
            DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            keyClient.deleteTable(tableRequest);

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考DeleteTable中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK DeleteTable中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class KeyspaceWrapper:
```

```
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def delete_table(self):
        """
        Deletes the table from the keyspace.
        """
        try:
            self.keyspaces_client.delete_table(
                keyspaceName=self.ks_name, tableName=self.table_name
            )
            self.table_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件DeleteTable中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Keyspaces 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetKeyspace配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetKeyspace。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用密鑰空間和表

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/// <summary>
/// Get data about a keyspace.
/// </summary>
/// <param name="keyspaceName">The name of the keyspace.</param>
/// <returns>The Amazon Resource Name (ARN) of the keyspace.</returns>
public async Task<string> GetKeyspace(string keyspaceName)
{
    var response = await _amazonKeyspaces.GetKeyspaceAsync(
        new GetKeyspaceRequest { KeyspaceName = keyspaceName });
    return response.ResourceArn;
}
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetKeyspace中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
       try {
           GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           GetKeyspaceResponse response =
keyClient.getKeyspace(keyspaceRequest);
           String name = response.keyspaceName();
           System.out.println("The " + name + " KeySpace is ready");

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考GetKeyspace中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response: GetKeyspaceResponse =
 keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK GetKeyspace中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
```

```
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def exists_keyspace(self, name):
        """
        Checks whether a keyspace exists.

        :param name: The name of the keyspace to look up.
        :return: True when the keyspace exists. Otherwise, False.
        """
        try:
            response = self.keyspaces_client.get_keyspace(keyspaceName=name)
            self.ks_name = response["keyspaceName"]
            self.ks_arn = response["resourceArn"]
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Keyspace %s does not exist.", name)
                exists = False
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return exists
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件GetKeyspace中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Keyspaces 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭GetTable配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用GetTable。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用密鑰空間和表

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Get information about an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the Amazon Keyspaces table.</param>
    /// <returns>The response containing data about the table.</returns>
    public async Task<GetTableResponse> GetTable(string keyspaceName, string
tableName)
    {
        var response = await _amazonKeyspaces.GetTableAsync(
            new GetTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考GetTable中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
           throws InterruptedException {
       try {
           boolean tableStatus = false;
           String status;
           GetTableResponse response = null;
           GetTableRequest tableRequest = GetTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           while (!tableStatus) {
               response = keyClient.getTable(tableRequest);
               status = response.statusAsString();
               System.out.println(". The table status is " + status);

               if (status.compareTo("ACTIVE") == 0) {
                   tableStatus = true;
               }
               Thread.sleep(500);
           }

           List<ColumnDefinition> cols =
response.schemaDefinition().allColumns();
           for (ColumnDefinition def : cols) {
               System.out.println("The column name is " + def.name());
               System.out.println("The column type is " + def.type());
           }

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考GetTable中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? =
 response!!.schemaDefinition?.allColumns
```

```
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK GetTable中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def get_table(self, table_name):
```

```
        """
        Gets data about a table in the keyspace.

        :param table_name: The name of the table to look up.
        :return: Data about the table.
        """
        try:
            response = self.keyspaces_client.get_table(
                keyspaceName=self.ks_name, tableName=table_name
            )
            self.table_name = table_name
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Table %s does not exist.", table_name)
                self.table_name = None
                response = None
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件GetTable中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Keyspaces 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListKeyspaces配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListKeyspaces。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用密鑰空間和表

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Lists all keyspaces for the account.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task ListKeyspaces()
    {
        var paginator = _amazonKeyspaces.Paginators.ListKeyspaces(new
ListKeyspacesRequest());

        Console.WriteLine("{0, -30}\t{1}", "Keyspace name", "Keyspace ARN");
        Console.WriteLine(new string('-', Console.WindowWidth));
        await foreach (var keyspace in paginator.Keyspaces)
        {

Console.WriteLine($"{keyspace.KeyspaceName,-30}\t{keyspace.ResourceArn}");
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListKeyspaces中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest =
ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();

            ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
            listRes.stream()
                    .flatMap(r -> r.keyspaces().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考ListKeyspaces中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun listKeyspacesPaginator() {
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
```

```
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListKeyspaces中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def list_keyspaces(self, limit):
        """
        Lists the keyspaces in your account.

        :param limit: The maximum number of keyspaces to list.
        """
        try:
```

```
            ks_paginator = self.keyspaces_client.get_paginator("list_keyspaces")
            for page in ks_paginator.paginate(PaginationConfig={"MaxItems":
limit}):
                for ks in page["keyspaces"]:
                    print(ks["keyspaceName"])
                    print(f"\t{ks['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list keyspaces. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ListKeyspaces中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Keyspaces 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭ListTables配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用ListTables。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用密鑰空間和表

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Lists the Amazon Keyspaces tables in a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>A list of TableSummary objects.</returns>
    public async Task<List<TableSummary>> ListTables(string keyspaceName)
    {
        var response = await _amazonKeyspaces.ListTablesAsync(new
ListTablesRequest { KeyspaceName = keyspaceName });
        response.Tables.ForEach(table =>
        {

Console.WriteLine($"{table.KeyspaceName}\t{table.TableName}\t{table.ResourceArn}");
        });

        return response.Tables;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考ListTables中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public static void listTables(KeyspacesClient keyClient, String keyspaceName)
{
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
```

```
            listRes.stream()
                    .flatMap(r -> r.tables().stream())
                    .forEach(content -> System.out.println(" ARN: " +
 content.resourceArn() +
                            " Table name: " + content.tableName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考ListTables中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK ListTables中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def list_tables(self):
        """
        Lists the tables in the keyspace.
        """
        try:
            table_paginator = self.keyspaces_client.get_paginator("list_tables")
            for page in table_paginator.paginate(keyspaceName=self.ks_name):
                for table in page["tables"]:
                    print(table["tableName"])
                    print(f"\t{table['resourceArn']}")
        except ClientError as err:
```

```
            logger.error(
                "Couldn't list tables in keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件ListTables中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Keyspaces 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭RestoreTable配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用RestoreTable。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用密鑰空間和表

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Restores the specified table to the specified point in time.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to restore.</param>
```

```
    /// <param name="timestamp">The time to which the table will be restored.</
param>
    /// <returns>The Amazon Resource Name (ARN) of the restored table.</returns>
    public async Task<string> RestoreTable(string keyspaceName, string tableName,
 string restoredTableName, DateTime timestamp)
    {
        var request = new RestoreTableRequest
        {
            RestoreTimestamp = timestamp,
            SourceKeyspaceName = keyspaceName,
            SourceTableName = tableName,
            TargetKeyspaceName = keyspaceName,
            TargetTableName = restoredTableName
        };

        var response = await _amazonKeyspaces.RestoreTableAsync(request);
        return response.RestoredTableARN;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考RestoreTable中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    public static void restoreTable(KeyspacesClient keyClient, String
keyspaceName, ZonedDateTime utc) {
        try {
            Instant myTime = utc.toInstant();
            RestoreTableRequest restoreTableRequest =
RestoreTableRequest.builder()
                    .restoreTimestamp(myTime)
                    .sourceTableName("Movie")
                    .targetKeyspaceName(keyspaceName)
```

```
                    .targetTableName("MovieRestore")
                    .sourceKeyspaceName(keyspaceName)
                    .build();

            RestoreTableResponse response =
 keyClient.restoreTable(restoreTableRequest);
            System.out.println("The ARN of the restored table is " +
 response.restoredTableARN());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考RestoreTable中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
```

```
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK RestoreTable中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def restore_table(self, restore_timestamp):
```

```
        """
        Restores the table to a previous point in time. The table is restored
        to a new table in the same keyspace.

        :param restore_timestamp: The point in time to restore the table. This
time
                                  must be in UTC format.
        :return: The name of the restored table.
        """
        try:
            restored_table_name = f"{self.table_name}_restored"
            self.keyspaces_client.restore_table(
                sourceKeyspaceName=self.ks_name,
                sourceTableName=self.table_name,
                targetKeyspaceName=self.ks_name,
                targetTableName=restored_table_name,
                restoreTimestamp=restore_timestamp,
            )
        except ClientError as err:
            logger.error(
                "Couldn't restore table %s. Here's why: %s: %s",
                restore_timestamp,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return restored_table_name
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件RestoreTable中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Keyspaces 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

搭UpdateTable配 AWS 開發套件或 CLI 使用

下列程式碼範例會示範如何使用UpdateTable。

動作範例是大型程式的程式碼摘錄，必須在內容中執行。您可以在下列程式碼範例的內容中看到此動作：

- 開始使用密鑰空間和表

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
    /// <summary>
    /// Updates the movie table to add a boolean column named watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to change.</param>
    /// <returns>The Amazon Resource Name (ARN) of the updated table.</returns>
    public async Task<string> UpdateTable(string keyspaceName, string tableName)
    {
        var newColumn = new ColumnDefinition { Name = "watched", Type =
"boolean" };
        var request = new UpdateTableRequest
        {
            KeyspaceName = keyspaceName,
            TableName = tableName,
            AddColumns = new List<ColumnDefinition> { newColumn }
        };
        var response = await _amazonKeyspaces.UpdateTableAsync(request);
        return response.ResourceArn;
    }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for .NET API 參考UpdateTable中的。

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
   public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
       try {
           ColumnDefinition def = ColumnDefinition.builder()
                   .name("watched")
                   .type("boolean")
                   .build();

           UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                   .keyspaceName(keySpace)
                   .tableName(tableName)
                   .addColumns(def)
                   .build();

           keyClient.updateTable(tableRequest);

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 如需 API 詳細資訊，請參閱 AWS SDK for Java 2.x API 參考UpdateTable中的。

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.updateTable(tableRequest)
    }
}
```

- 有關 API 的詳細信息，請參閱 AWS SDK UpdateTable中的 Kotlin API 參考。

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""

    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def update_table(self):
        """
        Updates the schema of the table.

        This example updates a table of movie data by adding a new column
        that tracks whether the movie has been watched.
        """
        try:
            self.keyspaces_client.update_table(
                keyspaceName=self.ks_name,
                tableName=self.table_name,
                addColumns=[{"name": "watched", "type": "boolean"}],
            )
```

```
        except ClientError as err:
            logger.error(
                "Couldn't update table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

- 如需 API 的詳細資訊，請參閱AWS 開發套件UpdateTable中的 Python (博托 3) API 參考。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Keyspaces 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

使 AWS 用 SDK 的 Amazon Keyspaces 的情況

下列程式碼範例說明如何使用 AWS SDK 在 Amazon Keyspaces 中實作常見案例。這些案例說明如何透過在 Amazon Keyspaces 中呼叫多個函數來完成特定任務。每個案例都包含一個連結 GitHub，您可以在其中找到如何設定和執行程式碼的指示。

範例

- 使用 SDK 開始使用 Amazon 密 Keyspaces 間和表 AWS

使用 SDK 開始使用 Amazon 密 Keyspaces 間和表 AWS

下列程式碼範例示範如何：

- 創建一個密鑰空間和表。資料表結構定義會保留影片資料，並啟用 point-in-time 復原功能。
- 使用具有 Sigv4 驗證的安全 TLS 連線連線至金鑰空間。
- 查詢資料表。添加，檢索和更新短片數據。
- 更新表格。添加列以跟踪觀看的電影。
- 將資料表還原至先前的狀態並清理資源。

.NET

AWS SDK for .NET

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
global using System.Security.Cryptography.X509Certificates;
global using Amazon.Keyspaces;
global using Amazon.Keyspaces.Model;
global using KeyspacesActions;
global using KeyspacesScenario;
global using Microsoft.Extensions.Configuration;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;
global using Newtonsoft.Json;


namespace KeyspacesBasics;

/// <summary>
/// Amazon Keyspaces (for Apache Cassandra) scenario. Shows some of the basic
/// actions performed with Amazon Keyspaces.
/// </summary>
public class KeyspacesBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the Amazon service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
```

```
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonKeyspaces>()
            .AddTransient<KeyspacesWrapper>()
            .AddTransient<CassandraWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<KeyspacesBasics>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        var keyspacesWrapper =
host.Services.GetRequiredService<KeyspacesWrapper>();
        var uiMethods = new UiMethods();

        var keyspaceName = configuration["KeyspaceName"];
        var tableName = configuration["TableName"];

        bool success; // Used to track the results of some operations.

        uiMethods.DisplayOverview();
        uiMethods.PressEnter();

        // Create the keyspace.
        var keyspaceArn = await keyspacesWrapper.CreateKeyspace(keyspaceName);

        // Wait for the keyspace to be available. GetKeyspace results in a
        // resource not found error until it is ready for use.
        try
        {
            var getKeyspaceArn = "";
            Console.Write($"Created {keyspaceName}. Waiting for it to become
available. ");
            do
            {
```

```
                getKeyspaceArn = await
keyspacesWrapper.GetKeyspace(keyspaceName);
                Console.Write(". ");
            } while (getKeyspaceArn != keyspaceArn);
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine("Waiting for keyspace to be created.");
        }

        Console.WriteLine($"\nThe keyspace {keyspaceName} is ready for use.");

        uiMethods.PressEnter();

        // Create the table.
        // First define the schema.
        var allColumns = new List<ColumnDefinition>
        {
            new ColumnDefinition { Name = "title", Type = "text" },
            new ColumnDefinition { Name = "year", Type = "int" },
            new ColumnDefinition { Name = "release_date", Type = "timestamp" },
            new ColumnDefinition { Name = "plot", Type = "text" },
        };

        var partitionKeys = new List<PartitionKey>
        {
            new PartitionKey { Name = "year", },
            new PartitionKey { Name = "title" },
        };

        var tableSchema = new SchemaDefinition
        {
            AllColumns = allColumns,
            PartitionKeys = partitionKeys,
        };

        var tableArn = await keyspacesWrapper.CreateTable(keyspaceName,
tableSchema, tableName);

        // Wait for the table to be active.
        try
        {
            var resp = new GetTableResponse();
            Console.Write("Waiting for the new table to be active. ");
```

```
            do
            {
                try
                {
                    resp = await keyspacesWrapper.GetTable(keyspaceName,
tableName);
                    Console.Write(".");
                }
                catch (ResourceNotFoundException)
                {
                    Console.Write(".");
                }
            } while (resp.Status != TableStatus.ACTIVE);

            // Display the table's schema.
            Console.WriteLine($"\nTable {tableName} has been created in
{keyspaceName}");
            Console.WriteLine("Let's take a look at the schema.");
            uiMethods.DisplayTitle("All columns");
            resp.SchemaDefinition.AllColumns.ForEach(column =>
            {
                Console.WriteLine($"{column.Name,-40}\t{column.Type,-20}");
            });

            uiMethods.DisplayTitle("Cluster keys");
            resp.SchemaDefinition.ClusteringKeys.ForEach(clusterKey =>
            {

Console.WriteLine($"{clusterKey.Name,-40}\t{clusterKey.OrderBy,-20}");
            });

            uiMethods.DisplayTitle("Partition keys");
            resp.SchemaDefinition.PartitionKeys.ForEach(partitionKey =>
            {
                Console.WriteLine($"{partitionKey.Name}");
            });

            uiMethods.PressEnter();
        }
        catch (ResourceNotFoundException ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
        }
```

```
        // Access Apache Cassandra using the Cassandra drive for C#.
        var cassandraWrapper =
host.Services.GetRequiredService<CassandraWrapper>();
        var movieFilePath = configuration["MovieFile"];

        Console.WriteLine("Let's add some movies to the table we created.");
        var inserted = await cassandraWrapper.InsertIntoMovieTable(keyspaceName,
tableName, movieFilePath);

        uiMethods.PressEnter();

        Console.WriteLine("Added the following movies to the table:");
        var rows = await cassandraWrapper.GetMovies(keyspaceName, tableName);
        uiMethods.DisplayTitle("All Movies");

        foreach (var row in rows)
        {
            var title = row.GetValue<string>("title");
            var year = row.GetValue<int>("year");
            var plot = row.GetValue<string>("plot");
            var release_date = row.GetValue<DateTime>("release_date");
            Console.WriteLine($"{release_date}\t{title}\t{year}\n{plot}");
            Console.WriteLine(uiMethods.SepBar);
        }

        // Update the table schema
        uiMethods.DisplayTitle("Update table schema");
        Console.WriteLine("Now we will update the table to add a boolean field
called watched.");

        // First save the current time as a UTC Date so the original
        // table can be restored later.
        var timeChanged = DateTime.UtcNow;

        // Now update the schema.
        var resourceArn = await keyspacesWrapper.UpdateTable(keyspaceName,
tableName);
        uiMethods.PressEnter();

        Console.WriteLine("Now let's mark some of the movies as watched.");

        // Pick some files to mark as watched.
        var movieToWatch = rows[2].GetValue<string>("title");
        var watchedMovieYear = rows[2].GetValue<int>("year");
```

```
        var changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[6].GetValue<string>("title");
        watchedMovieYear = rows[6].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[9].GetValue<string>("title");
        watchedMovieYear = rows[9].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[10].GetValue<string>("title");
        watchedMovieYear = rows[10].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        movieToWatch = rows[13].GetValue<string>("title");
        watchedMovieYear = rows[13].GetValue<int>("year");
        changedRows = await cassandraWrapper.MarkMovieAsWatched(keyspaceName,
tableName, movieToWatch, watchedMovieYear);

        uiMethods.DisplayTitle("Watched movies");
        Console.WriteLine("These movies have been marked as watched:");
        rows = await cassandraWrapper.GetWatchedMovies(keyspaceName, tableName);
        foreach (var row in rows)
        {
            var title = row.GetValue<string>("title");
            var year = row.GetValue<int>("year");
            Console.WriteLine($"{title,-40}\t{year,8}");
        }
        uiMethods.PressEnter();

        Console.WriteLine("We can restore the table to its previous state but
that can take up to 20 minutes to complete.");
        string answer;
        do
        {
            Console.WriteLine("Do you want to restore the table? (y/n)");
            answer = Console.ReadLine();
        } while (answer.ToLower() != "y" && answer.ToLower() != "n");

        if (answer == "y")
```

```
        {
            var restoredTableName = $"{tableName}_restored";
            var restoredTableArn = await keyspacesWrapper.RestoreTable(
                keyspaceName,
                tableName,
                restoredTableName,
                timeChanged);
            // Loop and call GetTable until the table is gone. Once it has been
            // deleted completely, GetTable will raise a
ResourceNotFoundException.
            bool wasRestored = false;

            try
            {
                do
                {
                    var resp = await keyspacesWrapper.GetTable(keyspaceName,
restoredTableName);
                    wasRestored = (resp.Status == TableStatus.ACTIVE);
                } while (!wasRestored);
            }
            catch (ResourceNotFoundException)
            {
                // If the restored table raised an error, it isn't
                // ready yet.
                Console.Write(".");
            }
        }

        uiMethods.DisplayTitle("Clean up resources.");

        // Delete the table.
        success = await keyspacesWrapper.DeleteTable(keyspaceName, tableName);

        Console.WriteLine($"Table {tableName} successfully deleted from
{keyspaceName}.");
        Console.WriteLine("Waiting for the table to be removed completely. ");

        // Loop and call GetTable until the table is gone. Once it has been
        // deleted completely, GetTable will raise a ResourceNotFoundException.
        bool wasDeleted = false;

        try
        {
```

```
            do
            {
                var resp = await keyspacesWrapper.GetTable(keyspaceName,
 tableName);
            } while (!wasDeleted);
        }
        catch (ResourceNotFoundException ex)
        {
            wasDeleted = true;
            Console.WriteLine($"{ex.Message} indicates that the table has been
 deleted.");
        }

        // Delete the keyspace.
        success = await keyspacesWrapper.DeleteKeyspace(keyspaceName);
        Console.WriteLine("The keyspace has been deleted and the demo is now
 complete.");
    }
}
```

```
namespace KeyspacesActions;

/// <summary>
/// Performs Amazon Keyspaces (for Apache Cassandra) actions.
/// </summary>
public class KeyspacesWrapper
{
    private readonly IAmazonKeyspaces _amazonKeyspaces;

    /// <summary>
    /// Constructor for the KeyspaceWrapper.
    /// </summary>
    /// <param name="amazonKeyspaces">An Amazon Keyspaces client object.</param>
    public KeyspacesWrapper(IAmazonKeyspaces amazonKeyspaces)
    {
        _amazonKeyspaces = amazonKeyspaces;
    }

    /// <summary>
    /// Create a new keyspace.
    /// </summary>
```

```
    /// <param name="keyspaceName">The name for the new keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new keyspace.</returns>
    public async Task<string> CreateKeyspace(string keyspaceName)
    {
        var response =
            await _amazonKeyspaces.CreateKeyspaceAsync(
                new CreateKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
    }


    /// <summary>
    /// Create a new Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace where the table will be
created.</param>
    /// <param name="schema">The schema for the new table.</param>
    /// <param name="tableName">The name of the new table.</param>
    /// <returns>The Amazon Resource Name (ARN) of the new table.</returns>
    public async Task<string> CreateTable(string keyspaceName, SchemaDefinition
schema, string tableName)
    {
        var request = new CreateTableRequest
        {
            KeyspaceName = keyspaceName,
            SchemaDefinition = schema,
            TableName = tableName,
            PointInTimeRecovery = new PointInTimeRecovery { Status =
PointInTimeRecoveryStatus.ENABLED }
        };

        var response = await _amazonKeyspaces.CreateTableAsync(request);
        return response.ResourceArn;
    }


    /// <summary>
    /// Delete an existing keyspace.
    /// </summary>
    /// <param name="keyspaceName"></param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.DeleteKeyspaceAsync(
```

```
            new DeleteKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteTable(string keyspaceName, string tableName)
    {
        var response = await _amazonKeyspaces.DeleteTableAsync(
            new DeleteTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Get data about a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>The Amazon Resource Name (ARN) of the keyspace.</returns>
    public async Task<string> GetKeyspace(string keyspaceName)
    {
        var response = await _amazonKeyspaces.GetKeyspaceAsync(
            new GetKeyspaceRequest { KeyspaceName = keyspaceName });
        return response.ResourceArn;
    }


    /// <summary>
    /// Get information about an Amazon Keyspaces table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the Amazon Keyspaces table.</param>
    /// <returns>The response containing data about the table.</returns>
    public async Task<GetTableResponse> GetTable(string keyspaceName, string
tableName)
    {
        var response = await _amazonKeyspaces.GetTableAsync(
```

```
            new GetTableRequest { KeyspaceName = keyspaceName, TableName =
tableName });
        return response;
    }


    /// <summary>
    /// Lists all keyspaces for the account.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task ListKeyspaces()
    {
        var paginator = _amazonKeyspaces.Paginators.ListKeyspaces(new
ListKeyspacesRequest());

        Console.WriteLine("{0, -30}\t{1}", "Keyspace name", "Keyspace ARN");
        Console.WriteLine(new string('-', Console.WindowWidth));
        await foreach (var keyspace in paginator.Keyspaces)
        {

Console.WriteLine($"{keyspace.KeyspaceName,-30}\t{keyspace.ResourceArn}");
        }
    }


    /// <summary>
    /// Lists the Amazon Keyspaces tables in a keyspace.
    /// </summary>
    /// <param name="keyspaceName">The name of the keyspace.</param>
    /// <returns>A list of TableSummary objects.</returns>
    public async Task<List<TableSummary>> ListTables(string keyspaceName)
    {
        var response = await _amazonKeyspaces.ListTablesAsync(new
ListTablesRequest { KeyspaceName = keyspaceName });
        response.Tables.ForEach(table =>
        {

Console.WriteLine($"{table.KeyspaceName}\t{table.TableName}\t{table.ResourceArn}");
        });

        return response.Tables;
    }
```

```
    /// <summary>
    /// Restores the specified table to the specified point in time.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to restore.</param>
    /// <param name="timestamp">The time to which the table will be restored.</
param>
    /// <returns>The Amazon Resource Name (ARN) of the restored table.</returns>
    public async Task<string> RestoreTable(string keyspaceName, string tableName,
 string restoredTableName, DateTime timestamp)
    {
        var request = new RestoreTableRequest
        {
            RestoreTimestamp = timestamp,
            SourceKeyspaceName = keyspaceName,
            SourceTableName = tableName,
            TargetKeyspaceName = keyspaceName,
            TargetTableName = restoredTableName
        };

        var response = await _amazonKeyspaces.RestoreTableAsync(request);
        return response.RestoredTableARN;
    }


    /// <summary>
    /// Updates the movie table to add a boolean column named watched.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table to change.</param>
    /// <returns>The Amazon Resource Name (ARN) of the updated table.</returns>
    public async Task<string> UpdateTable(string keyspaceName, string tableName)
    {
        var newColumn = new ColumnDefinition { Name = "watched", Type =
 "boolean" };
        var request = new UpdateTableRequest
        {
            KeyspaceName = keyspaceName,
            TableName = tableName,
            AddColumns = new List<ColumnDefinition> { newColumn }
        };
        var response = await _amazonKeyspaces.UpdateTableAsync(request);
        return response.ResourceArn;
    }
```

```
}
```

```
using System.Net;
using Cassandra;

namespace KeyspacesScenario;

/// <summary>
/// Class to perform CRUD methods on an Amazon Keyspaces (for Apache Cassandra)
 database.
///
/// NOTE: This sample uses a plain text authenticator for example purposes only.
/// Recommended best practice is to use a SigV4 authentication plugin, if
 available.
/// </summary>
public class CassandraWrapper
{
    private readonly IConfiguration _configuration;
    private readonly string _localPathToFile;
    private const string _certLocation = "https://certs.secureserver.net/
repository/sf-class2-root.crt";
    private const string _certFileName = "sf-class2-root.crt";
    private readonly X509Certificate2Collection _certCollection;
    private X509Certificate2 _amazoncert;
    private Cluster _cluster;

    // User name and password for the service.
    private string _userName = null!;
    private string _pwd = null!;

    public CassandraWrapper()
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        _localPathToFile = Path.GetTempPath();
```

```
        // Get the Starfield digital certificate and save it locally.
        var client = new WebClient();
        client.DownloadFile(_certLocation, $"{_localPathToFile}/
{_certFileName}");

        //var httpClient = new HttpClient();
        //var httpResult = httpClient.Get(fileUrl);
        //using var resultStream = await httpResult.Content.ReadAsStreamAsync();
        //using var fileStream = File.Create(pathToSave);
        //resultStream.CopyTo(fileStream);

        _certCollection = new X509Certificate2Collection();
        _amazoncert = new X509Certificate2($"{_localPathToFile}/
{_certFileName}");

        // Get the user name and password stored in the configuration file.
        _userName = _configuration["UserName"]!;
        _pwd = _configuration["Password"]!;

        // For a list of Service Endpoints for Amazon Keyspaces, see:
        // https://docs.aws.amazon.com/keyspaces/latest/devguide/
programmatic.endpoints.html
        var awsEndpoint = _configuration["ServiceEndpoint"];

        _cluster = Cluster.Builder()
            .AddContactPoints(awsEndpoint)
            .WithPort(9142)
            .WithAuthProvider(new PlainTextAuthProvider(_userName, _pwd))
            .WithSSL(new SSLOptions().SetCertificateCollection(_certCollection))
            .WithQueryOptions(
                new QueryOptions()
                    .SetConsistencyLevel(ConsistencyLevel.LocalQuorum)
                    .SetSerialConsistencyLevel(ConsistencyLevel.LocalSerial))
            .Build();
    }

    /// <summary>
    /// Loads the contents of a JSON file into a list of movies to be
    /// added to the Apache Cassandra table.
    /// </summary>
    /// <param name="movieFileName">The full path to the JSON file.</param>
    /// <returns>A list of movie objects.</returns>
```

```
    public List<Movie> ImportMoviesFromJson(string movieFileName, int numToImport
= 0)
    {
        if (!File.Exists(movieFileName))
        {
            return null!;
        }

        using var sr = new StreamReader(movieFileName);
        string json = sr.ReadToEnd();

        var allMovies = JsonConvert.DeserializeObject<List<Movie>>(json);

        // If numToImport = 0, return all movies in the collection.
        if (numToImport == 0)
        {
            // Now return the entire list of movies.
            return allMovies;
        }
        else
        {
            // Now return the first numToImport entries.
            return allMovies.GetRange(0, numToImport);
        }
    }

    /// <summary>
    /// Insert movies into the movie table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="movieTableName">The Amazon Keyspaces table.</param>
    /// <param name="movieFilePath">The path to the resource file containing
    /// movie data to insert into the table.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> InsertIntoMovieTable(string keyspaceName, string
movieTableName, string movieFilePath, int numToImport = 20)
    {
        // Get some movie data from the movies.json file
        var movies = ImportMoviesFromJson(movieFilePath, numToImport);

        var session = _cluster.Connect(keyspaceName);

        string insertCql;
```

```
        RowSet rs;

        // Now we insert the numToImport movies into the table.
        foreach (var movie in movies)
        {
            // Escape single quote characters in the plot.
            insertCql = $"INSERT INTO {keyspaceName}.{movieTableName}
(title, year, release_date, plot) values($${movie.Title}$$, {movie.Year},
'{movie.Info.Release_Date.ToString("yyyy-MM-dd")}', $${movie.Info.Plot}$$)";
            rs = await session.ExecuteAsync(new SimpleStatement(insertCql));
        }

        return true;
    }

    /// <summary>
    /// Gets all of the movies in the movies table.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <returns>A list of row objects containing movie data.</returns>
    public async Task<List<Row>> GetMovies(string keyspaceName, string tableName)
    {
        var session = _cluster.Connect();
        RowSet rs;
        try
        {
            rs = await session.ExecuteAsync(new SimpleStatement($"SELECT * FROM
{keyspaceName}.{tableName}"));

            // Extract the row data from the returned RowSet.
            var rows = rs.GetRows().ToList();
            return rows;
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            return null!;
        }
    }

    /// <summary>
    /// Mark a movie in the movie table as watched.
    /// </summary>
```

```
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <param name="title">The title of the movie to mark as watched.</param>
    /// <param name="year">The year the movie was released.</param>
    /// <returns>A set of rows containing the changed data.</returns>
    public async Task<List<Row>> MarkMovieAsWatched(string keyspaceName, string
tableName, string title, int year)
    {
        var session = _cluster.Connect();
        string updateCql = $"UPDATE {keyspaceName}.{tableName} SET watched=true
WHERE title = $${title}$$ AND year = {year};";
        var rs = await session.ExecuteAsync(new SimpleStatement(updateCql));
        var rows = rs.GetRows().ToList();
        return rows;
    }

    /// <summary>
    /// Retrieve the movies in the movies table where watched is true.
    /// </summary>
    /// <param name="keyspaceName">The keyspace containing the table.</param>
    /// <param name="tableName">The name of the table.</param>
    /// <returns>A list of row objects containing information about movies
    /// where watched is true.</returns>
    public async Task<List<Row>> GetWatchedMovies(string keyspaceName, string
tableName)
    {
        var session = _cluster.Connect();
        RowSet rs;
        try
        {
            rs = await session.ExecuteAsync(new SimpleStatement($"SELECT
title, year, plot FROM {keyspaceName}.{tableName} WHERE watched = true ALLOW
FILTERING"));

            // Extract the row data from the returned RowSet.
            var rows = rs.GetRows().ToList();
            return rows;
        }
        catch (Exception ex)
        {
            Console.WriteLine(ex.Message);
            return null!;
        }
    }
```

```
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for .NET API 參考》中的下列主題。
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Java

適用於 Java 2.x 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/**
 * Before running this Java (v2) code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * Before running this Java code example, you must create a
 * Java keystore (JKS) file and place it in your project's resources folder.
```

```
 *
 * This file is a secure file format used to hold certificate information for
 * Java applications. This is required to make a connection to Amazon Keyspaces.
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html
 *
 * This Java example performs the following tasks:
 *
 * 1. Create a keyspace.
 * 2. Check for keyspace existence.
 * 3. List keyspaces using a paginator.
 * 4. Create a table with a simple movie data schema and enable point-in-time
 * recovery.
 * 5. Check for the table to be in an Active state.
 * 6. List all tables in the keyspace.
 * 7. Use a Cassandra driver to insert some records into the Movie table.
 * 8. Get all records from the Movie table.
 * 9. Get a specific Movie.
 * 10. Get a UTC timestamp for the current time.
 * 11. Update the table schema to add a ‘watched’ Boolean column.
 * 12. Update an item as watched.
 * 13. Query for items with watched = True.
 * 14. Restore the table back to the previous state using the timestamp.
 * 15. Check for completion of the restore action.
 * 16. Delete the table.
 * 17. Confirm that both tables are deleted.
 * 18. Delete the keyspace.
 */

public class ScenarioKeyspaces {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    /*
     * Usage:
     * fileName - The name of the JSON file that contains movie data. (Get this
 file
     * from the GitHub repo at resources/sample_file.)
     * keyspaceName - The name of the keyspace to create.
     */
    public static void main(String[] args) throws InterruptedException,
 IOException {
```

```
        String fileName = "<Replace with the JSON file that contains movie
data>";
        String keyspaceName = "<Replace with the name of the keyspace to
create>";
        String titleUpdate = "The Family";
        int yearUpdate = 2013;
        String tableName = "Movie";
        String tableNameRestore = "MovieRestore";
        Region region = Region.US_EAST_1;
        KeyspacesClient keyClient = KeyspacesClient.builder()
                .region(region)
                .build();

        DriverConfigLoader loader =
DriverConfigLoader.fromClasspath("application.conf");
        CqlSession session = CqlSession.builder()
                .withConfigLoader(loader)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Keyspaces example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("1. Create a keyspace.");
        createKeySpace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        Thread.sleep(5000);
        System.out.println("2. Check for keyspace existence.");
        checkKeyspaceExistence(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. List keyspaces using a paginator.");
        listKeyspacesPaginator(keyClient);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Create a table with a simple movie data schema and
enable point-in-time recovery.");
        createTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("5. Check for the table to be in an Active state.");
        Thread.sleep(6000);
        checkTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. List all tables in the keyspace.");
        listTables(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Use a Cassandra driver to insert some records into
the Movie table.");
        Thread.sleep(6000);
        loadData(session, fileName, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Get all records from the Movie table.");
        getMovieData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9. Get a specific Movie.");
        getSpecificMovie(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("10. Get a UTC timestamp for the current time.");
        ZonedDateTime utc = ZonedDateTime.now(ZoneOffset.UTC);
        System.out.println("DATETIME = " + Date.from(utc.toInstant()));
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("11. Update the table schema to add a watched Boolean
column.");
        updateTable(keyClient, keyspaceName, tableName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("12. Update an item as watched.");
        Thread.sleep(10000); // Wait 10 secs for the update.
```

```
        updateRecord(session, keyspaceName, titleUpdate, yearUpdate);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("13. Query for items with watched = True.");
        getWatchedData(session, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("14. Restore the table back to the previous state
using the timestamp.");
        System.out.println("Note that the restore operation can take up to 20
minutes.");
        restoreTable(keyClient, keyspaceName, utc);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("15. Check for completion of the restore action.");
        Thread.sleep(5000);
        checkRestoredTable(keyClient, keyspaceName, "MovieRestore");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("16. Delete both tables.");
        deleteTable(keyClient, keyspaceName, tableName);
        deleteTable(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("17. Confirm that both tables are deleted.");
        checkTableDelete(keyClient, keyspaceName, tableName);
        checkTableDelete(keyClient, keyspaceName, tableNameRestore);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("18. Delete the keyspace.");
        deleteKeyspace(keyClient, keyspaceName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The scenario has completed successfully.");
        System.out.println(DASHES);
    }
```

```
   public static void deleteKeyspace(KeyspacesClient keyClient, String
keyspaceName) {
       try {
           DeleteKeyspaceRequest deleteKeyspaceRequest =
DeleteKeyspaceRequest.builder()
                   .keyspaceName(keyspaceName)
                   .build();

           keyClient.deleteKeyspace(deleteKeyspaceRequest);

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void checkTableDelete(KeyspacesClient keyClient, String
keyspaceName, String tableName)
           throws InterruptedException {
       try {
           String status;
           GetTableResponse response;
           GetTableRequest tableRequest = GetTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           // Keep looping until table cannot be found and a
ResourceNotFoundException is
           // thrown.
           while (true) {
               response = keyClient.getTable(tableRequest);
               status = response.statusAsString();
               System.out.println(". The table status is " + status);
               Thread.sleep(500);
           }

       } catch (ResourceNotFoundException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
       }
       System.out.println("The table is deleted");
   }
```

```
   public static void deleteTable(KeyspacesClient keyClient, String
keyspaceName, String tableName) {
       try {
           DeleteTableRequest tableRequest = DeleteTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           keyClient.deleteTable(tableRequest);

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void checkRestoredTable(KeyspacesClient keyClient, String
keyspaceName, String tableName)
           throws InterruptedException {
       try {
           boolean tableStatus = false;
           String status;
           GetTableResponse response = null;
           GetTableRequest tableRequest = GetTableRequest.builder()
                   .keyspaceName(keyspaceName)
                   .tableName(tableName)
                   .build();

           while (!tableStatus) {
               response = keyClient.getTable(tableRequest);
               status = response.statusAsString();
               System.out.println("The table status is " + status);

               if (status.compareTo("ACTIVE") == 0) {
                   tableStatus = true;
               }
               Thread.sleep(500);
           }

           List<ColumnDefinition> cols =
response.schemaDefinition().allColumns();
           for (ColumnDefinition def : cols) {
               System.out.println("The column name is " + def.name());
               System.out.println("The column type is " + def.type());
```

```
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void restoreTable(KeyspacesClient keyClient, String
keyspaceName, ZonedDateTime utc) {
        try {
            Instant myTime = utc.toInstant();
            RestoreTableRequest restoreTableRequest =
RestoreTableRequest.builder()
                    .restoreTimestamp(myTime)
                    .sourceTableName("Movie")
                    .targetKeyspaceName(keyspaceName)
                    .targetTableName("MovieRestore")
                    .sourceKeyspaceName(keyspaceName)
                    .build();

            RestoreTableResponse response =
keyClient.restoreTable(restoreTableRequest);
            System.out.println("The ARN of the restored table is " +
response.restoredTableARN());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void getWatchedData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session
                .execute("SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE
watched = true ALLOW FILTERING;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }
```

```
   public static void updateRecord(CqlSession session, String keySpace, String
titleUpdate, int yearUpdate) {
       String sqlStatement = "UPDATE \"" + keySpace
               + "\".\"Movie\" SET watched=true WHERE title = :k0 AND year
= :k1;";
       BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
       builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
       PreparedStatement preparedStatement = session.prepare(sqlStatement);
       builder.addStatement(preparedStatement.boundStatementBuilder()
               .setString("k0", titleUpdate)
               .setInt("k1", yearUpdate)
               .build());

       BatchStatement batchStatement = builder.build();
       session.execute(batchStatement);
   }

   public static void updateTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
       try {
           ColumnDefinition def = ColumnDefinition.builder()
                   .name("watched")
                   .type("boolean")
                   .build();

           UpdateTableRequest tableRequest = UpdateTableRequest.builder()
                   .keyspaceName(keySpace)
                   .tableName(tableName)
                   .addColumns(def)
                   .build();

           keyClient.updateTable(tableRequest);

       } catch (KeyspacesException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void getSpecificMovie(CqlSession session, String keyspaceName)
{
       ResultSet resultSet = session.execute(
```

```
                "SELECT * FROM \"" + keyspaceName + "\".\"Movie\" WHERE title =
'The Family' ALLOW FILTERING ;");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    // Get records from the Movie table.
    public static void getMovieData(CqlSession session, String keyspaceName) {
        ResultSet resultSet = session.execute("SELECT * FROM \"" + keyspaceName +
"\".\"Movie\";");
        resultSet.forEach(item -> {
            System.out.println("The Movie title is " + item.getString("title"));
            System.out.println("The Movie year is " + item.getInt("year"));
            System.out.println("The plot is " + item.getString("plot"));
        });
    }

    // Load data into the table.
    public static void loadData(CqlSession session, String fileName, String
keySpace) throws IOException {
        String sqlStatement = "INSERT INTO \"" + keySpace + "\".\"Movie\" (title,
year, plot) values (:k0, :k1, :k2)";
        JsonParser parser = new JsonFactory().createParser(new File(fileName));
        com.fasterxml.jackson.databind.JsonNode rootNode = new
ObjectMapper().readTree(parser);
        Iterator<JsonNode> iter = rootNode.iterator();
        ObjectNode currentNode;
        int t = 0;
        while (iter.hasNext()) {

            // Add 20 movies to the table.
            if (t == 20)
                break;
            currentNode = (ObjectNode) iter.next();

            int year = currentNode.path("year").asInt();
            String title = currentNode.path("title").asText();
            String plot = currentNode.path("info").path("plot").toString();

            // Insert the data into the Amazon Keyspaces table.
```

```
            BatchStatementBuilder builder =
BatchStatement.builder(DefaultBatchType.UNLOGGED);
            builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM);
            PreparedStatement preparedStatement = session.prepare(sqlStatement);
            builder.addStatement(preparedStatement.boundStatementBuilder()
                    .setString("k0", title)
                    .setInt("k1", year)
                    .setString("k2", plot)
                    .build());

            BatchStatement batchStatement = builder.build();
            session.execute(batchStatement);
            t++;
        }

        System.out.println("You have added " + t + " records successfully!");
    }

    public static void listTables(KeyspacesClient keyClient, String keyspaceName)
{
        try {
            ListTablesRequest tablesRequest = ListTablesRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            ListTablesIterable listRes =
keyClient.listTablesPaginator(tablesRequest);
            listRes.stream()
                    .flatMap(r -> r.tables().stream())
                    .forEach(content -> System.out.println(" ARN: " +
content.resourceArn() +
                            " Table name: " + content.tableName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkTable(KeyspacesClient keyClient, String keyspaceName,
String tableName)
            throws InterruptedException {
        try {
            boolean tableStatus = false;
```

```
            String status;
            GetTableResponse response = null;
            GetTableRequest tableRequest = GetTableRequest.builder()
                    .keyspaceName(keyspaceName)
                    .tableName(tableName)
                    .build();

            while (!tableStatus) {
                response = keyClient.getTable(tableRequest);
                status = response.statusAsString();
                System.out.println(". The table status is " + status);

                if (status.compareTo("ACTIVE") == 0) {
                    tableStatus = true;
                }
                Thread.sleep(500);
            }

            List<ColumnDefinition> cols =
response.schemaDefinition().allColumns();
            for (ColumnDefinition def : cols) {
                System.out.println("The column name is " + def.name());
                System.out.println("The column type is " + def.type());
            }

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createTable(KeyspacesClient keyClient, String keySpace,
String tableName) {
        try {
            // Set the columns.
            ColumnDefinition defTitle = ColumnDefinition.builder()
                    .name("title")
                    .type("text")
                    .build();

            ColumnDefinition defYear = ColumnDefinition.builder()
                    .name("year")
                    .type("int")
                    .build();
```

```
        ColumnDefinition defReleaseDate = ColumnDefinition.builder()
                .name("release_date")
                .type("timestamp")
                .build();

        ColumnDefinition defPlot = ColumnDefinition.builder()
                .name("plot")
                .type("text")
                .build();

        List<ColumnDefinition> colList = new ArrayList<>();
        colList.add(defTitle);
        colList.add(defYear);
        colList.add(defReleaseDate);
        colList.add(defPlot);

        // Set the keys.
        PartitionKey yearKey = PartitionKey.builder()
                .name("year")
                .build();

        PartitionKey titleKey = PartitionKey.builder()
                .name("title")
                .build();

        List<PartitionKey> keyList = new ArrayList<>();
        keyList.add(yearKey);
        keyList.add(titleKey);

        SchemaDefinition schemaDefinition = SchemaDefinition.builder()
                .partitionKeys(keyList)
                .allColumns(colList)
                .build();

        PointInTimeRecovery timeRecovery = PointInTimeRecovery.builder()
                .status(PointInTimeRecoveryStatus.ENABLED)
                .build();

        CreateTableRequest tableRequest = CreateTableRequest.builder()
                .keyspaceName(keySpace)
                .tableName(tableName)
                .schemaDefinition(schemaDefinition)
                .pointInTimeRecovery(timeRecovery)
```

```
                    .build();

            CreateTableResponse response = keyClient.createTable(tableRequest);
            System.out.println("The table ARN is " + response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void listKeyspacesPaginator(KeyspacesClient keyClient) {
        try {
            ListKeyspacesRequest keyspacesRequest =
ListKeyspacesRequest.builder()
                    .maxResults(10)
                    .build();

            ListKeyspacesIterable listRes =
keyClient.listKeyspacesPaginator(keyspacesRequest);
            listRes.stream()
                    .flatMap(r -> r.keyspaces().stream())
                    .forEach(content -> System.out.println(" Name: " +
content.keyspaceName()));

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void checkKeyspaceExistence(KeyspacesClient keyClient, String
keyspaceName) {
        try {
            GetKeyspaceRequest keyspaceRequest = GetKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            GetKeyspaceResponse response =
keyClient.getKeyspace(keyspaceRequest);
            String name = response.keyspaceName();
            System.out.println("The " + name + " KeySpace is ready");

        } catch (KeyspacesException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void createKeySpace(KeyspacesClient keyClient, String
 keyspaceName) {
        try {
            CreateKeyspaceRequest keyspaceRequest =
 CreateKeyspaceRequest.builder()
                    .keyspaceName(keyspaceName)
                    .build();

            CreateKeyspaceResponse response =
 keyClient.createKeyspace(keyspaceRequest);
            System.out.println("The ARN of the KeySpace is " +
 response.resourceArn());

        } catch (KeyspacesException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS SDK for Java 2.x API 參考》中的下列主題。
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Kotlin

適用於 Kotlin 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation topic:

 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 This example uses a secure file format to hold certificate information for
 Kotlin applications. This is required to make a connection to Amazon Keyspaces.
 For more information, see the following documentation topic:

 https://docs.aws.amazon.com/keyspaces/latest/devguide/using_java_driver.html

 This Kotlin example performs the following tasks:

 1. Create a keyspace.
 2. Check for keyspace existence.
 3. List keyspaces using a paginator.
 4. Create a table with a simple movie data schema and enable point-in-time
 recovery.
 5. Check for the table to be in an Active state.
 6. List all tables in the keyspace.
 7. Use a Cassandra driver to insert some records into the Movie table.
 8. Get all records from the Movie table.
 9. Get a specific Movie.
 10. Get a UTC timestamp for the current time.
 11. Update the table schema to add a ‘watched’ Boolean column.
 12. Update an item as watched.
 13. Query for items with watched = True.
 14. Restore the table back to the previous state using the timestamp.
```

```
 15. Check for completion of the restore action.
 16. Delete the table.
 17. Confirm that both tables are deleted.
 18. Delete the keyspace.
 */

/*
   Usage:
     fileName - The name of the JSON file that contains movie data. (Get this
 file from the GitHub repo at resources/sample_file.)
     keyspaceName - The name of the keyspace to create.
 */
val DASHES: String = String(CharArray(80)).replace("\u0000", "-")

suspend fun main() {
    val fileName = "<Replace with the JSON file that contains movie data>"
    val keyspaceName = "<Replace with the name of the keyspace to create>"
    val titleUpdate = "The Family"
    val yearUpdate = 2013
    val tableName = "MovieKotlin"
    val tableNameRestore = "MovieRestore"

    val loader = DriverConfigLoader.fromClasspath("application.conf")
    val session =
        CqlSession
            .builder()
            .withConfigLoader(loader)
            .build()

    println(DASHES)
    println("Welcome to the Amazon Keyspaces example scenario.")
    println(DASHES)

    println(DASHES)
    println("1. Create a keyspace.")
    createKeySpace(keyspaceName)
    println(DASHES)

    println(DASHES)
    delay(5000)
    println("2. Check for keyspace existence.")
    checkKeyspaceExistence(keyspaceName)
    println(DASHES)
```

```
    println(DASHES)
    println("3. List keyspaces using a paginator.")
    listKeyspacesPaginator()
    println(DASHES)

    println(DASHES)
    println("4. Create a table with a simple movie data schema and enable point-
in-time recovery.")
    createTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("5. Check for the table to be in an Active state.")
    delay(6000)
    checkTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("6. List all tables in the keyspace.")
    listTables(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("7. Use a Cassandra driver to insert some records into the Movie
 table.")
    delay(6000)
    loadData(session, fileName, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("8. Get all records from the Movie table.")
    getMovieData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("9. Get a specific Movie.")
    getSpecificMovie(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("10. Get a UTC timestamp for the current time.")
    val utc = ZonedDateTime.now(ZoneOffset.UTC)
    println("DATETIME = ${Date.from(utc.toInstant())}")
    println(DASHES)
```

```
    println(DASHES)
    println("11. Update the table schema to add a watched Boolean column.")
    updateTable(keyspaceName, tableName)
    println(DASHES)

    println(DASHES)
    println("12. Update an item as watched.")
    delay(10000) // Wait 10 seconds for the update.
    updateRecord(session, keyspaceName, titleUpdate, yearUpdate)
    println(DASHES)

    println(DASHES)
    println("13. Query for items with watched = True.")
    getWatchedData(session, keyspaceName)
    println(DASHES)

    println(DASHES)
    println("14. Restore the table back to the previous state using the
timestamp.")
    println("Note that the restore operation can take up to 20 minutes.")
    restoreTable(keyspaceName, utc)
    println(DASHES)

    println(DASHES)
    println("15. Check for completion of the restore action.")
    delay(5000)
    checkRestoredTable(keyspaceName, "MovieRestore")
    println(DASHES)

    println(DASHES)
    println("16. Delete both tables.")
    deleteTable(keyspaceName, tableName)
    deleteTable(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("17. Confirm that both tables are deleted.")
    checkTableDelete(keyspaceName, tableName)
    checkTableDelete(keyspaceName, tableNameRestore)
    println(DASHES)

    println(DASHES)
    println("18. Delete the keyspace.")
```

```
    deleteKeyspace(keyspaceName)
    println(DASHES)

    println(DASHES)
    println("The scenario has completed successfully.")
    println(DASHES)
}

suspend fun deleteKeyspace(keyspaceNameVal: String?) {
    val deleteKeyspaceRequest =
        DeleteKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteKeyspace(deleteKeyspaceRequest)
    }
}

suspend fun checkTableDelete(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var status: String
    var response: GetTableResponse
    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    try {
        KeyspacesClient { region = "us-east-1" }.use { keyClient ->
            // Keep looping until the table cannot be found and a
 ResourceNotFoundException is thrown.
            while (true) {
                response = keyClient.getTable(tableRequest)
                status = response.status.toString()
                println(". The table status is $status")
                delay(500)
            }
        }
    } catch (e: ResourceNotFoundException) {
        println(e.message)
```

```
    }
    println("The table is deleted")
}

suspend fun deleteTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    val tableRequest =
        DeleteTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient.deleteTable(tableRequest)
    }
}

suspend fun checkRestoredTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println("The table status is $status")

            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
```

```
        val cols = response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun restoreTable(
    keyspaceName: String?,
    utc: ZonedDateTime,
) {
    // Create an aws.smithy.kotlin.runtime.time.Instant value.
    val timeStamp =
        aws.smithy.kotlin.runtime.time
            .Instant(utc.toInstant())
    val restoreTableRequest =
        RestoreTableRequest {
            restoreTimestamp = timeStamp
            sourceTableName = "MovieKotlin"
            targetKeyspaceName = keyspaceName
            targetTableName = "MovieRestore"
            sourceKeyspaceName = keyspaceName
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.restoreTable(restoreTableRequest)
        println("The ARN of the restored table is ${response.restoredTableArn}")
    }
}

fun getWatchedData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".
\"MovieKotlin\" WHERE watched = true ALLOW FILTERING;")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
```

```
    }
}

fun updateRecord(
    session: CqlSession,
    keySpace: String,
    titleUpdate: String?,
    yearUpdate: Int,
) {
    val sqlStatement =
        "UPDATE \"$keySpace\".\"MovieKotlin\" SET watched=true WHERE title = :k0
 AND year = :k1;"
    val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
    builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
    val preparedStatement = session.prepare(sqlStatement)
    builder.addStatement(
        preparedStatement
            .boundStatementBuilder()
            .setString("k0", titleUpdate)
            .setInt("k1", yearUpdate)
            .build(),
    )
    val batchStatement = builder.build()
    session.execute(batchStatement)
}

suspend fun updateTable(
    keySpace: String?,
    tableNameVal: String?,
) {
    val def =
        ColumnDefinition {
            name = "watched"
            type = "boolean"
        }

    val tableRequest =
        UpdateTableRequest {
            keyspaceName = keySpace
            tableName = tableNameVal
            addColumns = listOf(def)
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
```

```
        keyClient.updateTable(tableRequest)
    }
}

fun getSpecificMovie(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet =
        session.execute("SELECT * FROM \"$keyspaceName\".\"MovieKotlin\" WHERE
 title = 'The Family' ALLOW FILTERING ;")

    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Get records from the Movie table.
fun getMovieData(
    session: CqlSession,
    keyspaceName: String,
) {
    val resultSet = session.execute("SELECT * FROM \"$keyspaceName\".
\"MovieKotlin\";")
    resultSet.forEach { item: Row ->
        println("The Movie title is ${item.getString("title")}")
        println("The Movie year is ${item.getInt("year")}")
        println("The plot is ${item.getString("plot")}")
    }
}

// Load data into the table.
fun loadData(
    session: CqlSession,
    fileName: String,
    keySpace: String,
) {
    val sqlStatement =
        "INSERT INTO \"$keySpace\".\"MovieKotlin\" (title, year, plot) values
 (:k0, :k1, :k2)"
    val parser = JsonFactory().createParser(File(fileName))
    val rootNode = ObjectMapper().readTree<JsonNode>(parser)
```

```
    val iter: Iterator<JsonNode> = rootNode.iterator()
    var currentNode: ObjectNode

    var t = 0
    while (iter.hasNext()) {
        if (t == 50) {
            break
        }

        currentNode = iter.next() as ObjectNode
        val year = currentNode.path("year").asInt()
        val title = currentNode.path("title").asText()
        val info = currentNode.path("info").toString()

        // Insert the data into the Amazon Keyspaces table.
        val builder = BatchStatement.builder(DefaultBatchType.UNLOGGED)
        builder.setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
        val preparedStatement: PreparedStatement = session.prepare(sqlStatement)
        builder.addStatement(
            preparedStatement
                .boundStatementBuilder()
                .setString("k0", title)
                .setInt("k1", year)
                .setString("k2", info)
                .build(),
        )

        val batchStatement = builder.build()
        session.execute(batchStatement)
        t++
    }
}

suspend fun listTables(keyspaceNameVal: String?) {
    val tablesRequest =
        ListTablesRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listTablesPaginated(tablesRequest)
            .transform { it.tables?.forEach { obj -> emit(obj) } }
            .collect { obj ->
```

```
                println(" ARN: ${obj.resourceArn} Table name: ${obj.tableName}")
            }
    }
}

suspend fun checkTable(
    keyspaceNameVal: String?,
    tableNameVal: String?,
) {
    var tableStatus = false
    var status: String
    var response: GetTableResponse? = null

    val tableRequest =
        GetTableRequest {
            keyspaceName = keyspaceNameVal
            tableName = tableNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        while (!tableStatus) {
            response = keyClient.getTable(tableRequest)
            status = response!!.status.toString()
            println(". The table status is $status")
            if (status.compareTo("ACTIVE") == 0) {
                tableStatus = true
            }
            delay(500)
        }
        val cols: List<ColumnDefinition>? =
 response!!.schemaDefinition?.allColumns
        if (cols != null) {
            for (def in cols) {
                println("The column name is ${def.name}")
                println("The column type is ${def.type}")
            }
        }
    }
}

suspend fun createTable(
    keySpaceVal: String?,
    tableNameVal: String?,
) {
    // Set the columns.
```

```
    val defTitle =
        ColumnDefinition {
            name = "title"
            type = "text"
        }

    val defYear =
        ColumnDefinition {
            name = "year"
            type = "int"
        }

    val defReleaseDate =
        ColumnDefinition {
            name = "release_date"
            type = "timestamp"
        }

    val defPlot =
        ColumnDefinition {
            name = "plot"
            type = "text"
        }

    val colList = ArrayList<ColumnDefinition>()
    colList.add(defTitle)
    colList.add(defYear)
    colList.add(defReleaseDate)
    colList.add(defPlot)

    // Set the keys.
    val yearKey =
        PartitionKey {
            name = "year"
        }

    val titleKey =
        PartitionKey {
            name = "title"
        }

    val keyList = ArrayList<PartitionKey>()
    keyList.add(yearKey)
    keyList.add(titleKey)
```

```
    val schemaDefinitionOb =
        SchemaDefinition {
            partitionKeys = keyList
            allColumns = colList
        }

    val timeRecovery =
        PointInTimeRecovery {
            status = PointInTimeRecoveryStatus.Enabled
        }

    val tableRequest =
        CreateTableRequest {
            keyspaceName = keySpaceVal
            tableName = tableNameVal
            schemaDefinition = schemaDefinitionOb
            pointInTimeRecovery = timeRecovery
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createTable(tableRequest)
        println("The table ARN is ${response.resourceArn}")
    }
}

suspend fun listKeyspacesPaginator() {
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        keyClient
            .listKeyspacesPaginated(ListKeyspacesRequest {})
            .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
            .collect { obj ->
                println("Name: ${obj.keyspaceName}")
            }
    }
}

suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
    val keyspaceRequest =
        GetKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }
    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
```

```
        val response: GetKeyspaceResponse =
 keyClient.getKeyspace(keyspaceRequest)
        val name = response.keyspaceName
        println("The $name KeySpace is ready")
    }
}

suspend fun createKeySpace(keyspaceNameVal: String) {
    val keyspaceRequest =
        CreateKeyspaceRequest {
            keyspaceName = keyspaceNameVal
        }

    KeyspacesClient { region = "us-east-1" }.use { keyClient ->
        val response = keyClient.createKeyspace(keyspaceRequest)
        println("The ARN of the KeySpace is ${response.resourceArn}")
    }
}
```

- 如需 API 詳細資訊，請參閱《AWS 適用於 Kotlin 的 SDK API 參考》中的下列主題。
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

Python

適用於 Python (Boto3) 的 SDK

> **Note**
>
> 還有更多關於 GitHub。尋找完整範例，並了解如何在AWS 設定和執行程式碼範例儲存庫。

在命令提示中執行互動式案例。

```
class KeyspaceScenario:
    """Runs an interactive scenario that shows how to get started using Amazon
 Keyspaces."""

    def __init__(self, ks_wrapper):
        """
        :param ks_wrapper: An object that wraps Amazon Keyspace actions.
        """
        self.ks_wrapper = ks_wrapper

    @demo_func
    def create_keyspace(self):
        """
        1. Creates a keyspace.
        2. Lists up to 10 keyspaces in your account.
        """
        print("Let's create a keyspace.")
        ks_name = q.ask(
            "Enter a name for your new keyspace.\nThe name can contain only
 letters, "
            "numbers and underscores: ",
            q.non_empty,
        )
        if self.ks_wrapper.exists_keyspace(ks_name):
            print(f"A keyspace named {ks_name} exists.")
        else:
            ks_arn = self.ks_wrapper.create_keyspace(ks_name)
            ks_exists = False
            while not ks_exists:
                wait(3)
                ks_exists = self.ks_wrapper.exists_keyspace(ks_name)
```

```
            print(f"Created a new keyspace.\n\t{ks_arn}.")
        print("The first 10 keyspaces in your account are:\n")
        self.ks_wrapper.list_keyspaces(10)

    @demo_func
    def create_table(self):
        """
        1. Creates a table in the keyspace. The table is configured with a schema
to hold
           movie data and has point-in-time recovery enabled.
        2. Waits for the table to be in an active state.
        3. Displays schema information for the table.
        4. Lists tables in the keyspace.
        """
        print("Let's create a table for movies in your keyspace.")
        table_name = q.ask("Enter a name for your table: ", q.non_empty)
        table = self.ks_wrapper.get_table(table_name)
        if table is not None:
            print(
                f"A table named {table_name} already exists in keyspace "
                f"{self.ks_wrapper.ks_name}."
            )
        else:
            table_arn = self.ks_wrapper.create_table(table_name)
            print(f"Created table {table_name}:\n\t{table_arn}")
            table = {"status": None}
            print("Waiting for your table to be ready...")
            while table["status"] != "ACTIVE":
                wait(5)
                table = self.ks_wrapper.get_table(table_name)
        print(f"Your table is {table['status']}. Its schema is:")
        pp(table["schemaDefinition"])
        print("\nThe tables in your keyspace are:\n")
        self.ks_wrapper.list_tables()

    @demo_func
    def ensure_tls_cert(self):
        """
        Ensures you have a TLS certificate available to use to secure the
connection
        to the keyspace. This function downloads a default certificate or lets
you
        specify your own.
        """
```

```
        print("To connect to your keyspace, you must have a TLS certificate.")
        print("Checking for TLS certificate...")
        cert_path = os.path.join(
            os.path.dirname(__file__), QueryManager.DEFAULT_CERT_FILE
        )
        if not os.path.exists(cert_path):
            cert_choice = q.ask(
                f"Press enter to download a certificate from
{QueryManager.CERT_URL} "
                f"or enter the full path to the certificate you want to use: "
            )
            if cert_choice:
                cert_path = cert_choice
            else:
                cert = requests.get(QueryManager.CERT_URL).text
                with open(cert_path, "w") as cert_file:
                    cert_file.write(cert)
        else:
            q.ask(f"Certificate {cert_path} found. Press Enter to continue.")
        print(
            f"Certificate {cert_path} will be used to secure the connection to
your keyspace."
        )
        return cert_path

    @demo_func
    def query_table(self, qm, movie_file):
        """
        1. Adds movies to the table from a sample movie data file.
        2. Gets a list of movies from the table and lets you select one.
        3. Displays more information about the selected movie.
        """
        qm.add_movies(self.ks_wrapper.table_name, movie_file)
        movies = qm.get_movies(self.ks_wrapper.table_name)
        print(f"Added {len(movies)} movies to the table:")
        sel = q.choose("Pick one to learn more about it: ", [m.title for m in
movies])
        movie_choice = qm.get_movie(
            self.ks_wrapper.table_name, movies[sel].title, movies[sel].year
        )
        print(movie_choice.title)
        print(f"\tReleased: {movie_choice.release_date}")
        print(f"\tPlot: {movie_choice.plot}")
```

```
    @demo_func
    def update_and_restore_table(self, qm):
        """
        1. Updates the table by adding a column to track watched movies.
        2. Marks some of the movies as watched.
        3. Gets the list of watched movies from the table.
        4. Restores to a movies_restored table at a previous point in time.
        5. Gets the list of movies from the restored table.
        """
        print("Let's add a column to record which movies you've watched.")
        pre_update_timestamp = datetime.utcnow()
        print(
            f"Recorded the current UTC time of {pre_update_timestamp} so we can
restore the table later."
        )
        self.ks_wrapper.update_table()
        print("Waiting for your table to update...")
        table = {"status": "UPDATING"}
        while table["status"] != "ACTIVE":
            wait(5)
            table = self.ks_wrapper.get_table(self.ks_wrapper.table_name)
        print("Column 'watched' added to table.")
        q.ask(
            "Let's mark some of the movies as watched. Press Enter when you're
ready.\n"
        )
        movies = qm.get_movies(self.ks_wrapper.table_name)
        for movie in movies[:10]:
            qm.watched_movie(self.ks_wrapper.table_name, movie.title, movie.year)
            print(f"Marked {movie.title} as watched.")
        movies = qm.get_movies(self.ks_wrapper.table_name, watched=True)
        print("-" * 88)
        print("The watched movies in our table are:\n")
        for movie in movies:
            print(movie.title)
        print("-" * 88)
        if q.ask(
            "Do you want to restore the table to the way it was before all of
these\n"
            "updates? Keep in mind, this can take up to 20 minutes. (y/n) ",
            q.is_yesno,
        ):
            starting_table_name = self.ks_wrapper.table_name
```

```
            table_name_restored =
self.ks_wrapper.restore_table(pre_update_timestamp)
            table = {"status": "RESTORING"}
            while table["status"] != "ACTIVE":
                wait(10)
                table = self.ks_wrapper.get_table(table_name_restored)
            print(
                f"Restored {starting_table_name} to {table_name_restored} "
                f"at a point in time of {pre_update_timestamp}."
            )
            movies = qm.get_movies(table_name_restored)
            print("Now the movies in our table are:")
            for movie in movies:
                print(movie.title)

    def cleanup(self, cert_path):
        """
        1. Deletes the table and waits for it to be removed.
        2. Deletes the keyspace.

        :param cert_path: The path of the TLS certificate used in the demo. If
the
                          certificate was downloaded during the demo, it is
removed.
        """
        if q.ask(
            f"Do you want to delete your {self.ks_wrapper.table_name} table and "
            f"{self.ks_wrapper.ks_name} keyspace? (y/n) ",
            q.is_yesno,
        ):
            table_name = self.ks_wrapper.table_name
            self.ks_wrapper.delete_table()
            table = self.ks_wrapper.get_table(table_name)
            print("Waiting for the table to be deleted.")
            while table is not None:
                wait(5)
                table = self.ks_wrapper.get_table(table_name)
            print("Table deleted.")
            self.ks_wrapper.delete_keyspace()
            print(
                "Keyspace deleted. If you chose to restore your table during the
"
                "demo, the original table is also deleted."
            )
```

```
            if cert_path == os.path.join(
                os.path.dirname(__file__), QueryManager.DEFAULT_CERT_FILE
            ) and os.path.exists(cert_path):
                os.remove(cert_path)
                print("Removed certificate that was downloaded for this demo.")

    def run_scenario(self):
        logging.basicConfig(level=logging.INFO, format="%(levelname)s:
 %(message)s")

        print("-" * 88)
        print("Welcome to the Amazon Keyspaces (for Apache Cassandra) demo.")
        print("-" * 88)

        self.create_keyspace()
        self.create_table()
        cert_file_path = self.ensure_tls_cert()
        # Use a context manager to ensure the connection to the keyspace is
 closed.
        with QueryManager(
            cert_file_path, boto3.DEFAULT_SESSION, self.ks_wrapper.ks_name
        ) as qm:
            self.query_table(qm, "../../../resources/sample_files/movies.json")
            self.update_and_restore_table(qm)
        self.cleanup(cert_file_path)

        print("\nThanks for watching!")
        print("-" * 88)


if __name__ == "__main__":
    try:
        scenario = KeyspaceScenario(KeyspaceWrapper.from_client())
        scenario.run_scenario()
    except Exception:
        logging.exception("Something went wrong with the demo.")
```

定義包裝鍵空間和表格動作的類別。

```
class KeyspaceWrapper:
    """Encapsulates Amazon Keyspaces (for Apache Cassandra) keyspace and table
 actions."""
```

```
    def __init__(self, keyspaces_client):
        """
        :param keyspaces_client: A Boto3 Amazon Keyspaces client.
        """
        self.keyspaces_client = keyspaces_client
        self.ks_name = None
        self.ks_arn = None
        self.table_name = None

    @classmethod
    def from_client(cls):
        keyspaces_client = boto3.client("keyspaces")
        return cls(keyspaces_client)


    def create_keyspace(self, name):
        """
        Creates a keyspace.

        :param name: The name to give the keyspace.
        :return: The Amazon Resource Name (ARN) of the new keyspace.
        """
        try:
            response = self.keyspaces_client.create_keyspace(keyspaceName=name)
            self.ks_name = name
            self.ks_arn = response["resourceArn"]
        except ClientError as err:
            logger.error(
                "Couldn't create %s. Here's why: %s: %s",
                name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return self.ks_arn


    def exists_keyspace(self, name):
        """
        Checks whether a keyspace exists.

        :param name: The name of the keyspace to look up.
```

```
        :return: True when the keyspace exists. Otherwise, False.
        """
        try:
            response = self.keyspaces_client.get_keyspace(keyspaceName=name)
            self.ks_name = response["keyspaceName"]
            self.ks_arn = response["resourceArn"]
            exists = True
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Keyspace %s does not exist.", name)
                exists = False
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return exists


    def list_keyspaces(self, limit):
        """
        Lists the keyspaces in your account.

        :param limit: The maximum number of keyspaces to list.
        """
        try:
            ks_paginator = self.keyspaces_client.get_paginator("list_keyspaces")
            for page in ks_paginator.paginate(PaginationConfig={"MaxItems":
limit}):
                for ks in page["keyspaces"]:
                    print(ks["keyspaceName"])
                    print(f"\t{ks['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list keyspaces. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

```
    def create_table(self, table_name):
        """
        Creates a table in the  keyspace.
        The table is created with a schema for storing movie data
        and has point-in-time recovery enabled.

        :param table_name: The name to give the table.
        :return: The ARN of the new table.
        """
        try:
            response = self.keyspaces_client.create_table(
                keyspaceName=self.ks_name,
                tableName=table_name,
                schemaDefinition={
                    "allColumns": [
                        {"name": "title", "type": "text"},
                        {"name": "year", "type": "int"},
                        {"name": "release_date", "type": "timestamp"},
                        {"name": "plot", "type": "text"},
                    ],
                    "partitionKeys": [{"name": "year"}, {"name": "title"}],
                },
                pointInTimeRecovery={"status": "ENABLED"},
            )
        except ClientError as err:
            logger.error(
                "Couldn't create table %s. Here's why: %s: %s",
                table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return response["resourceArn"]


    def get_table(self, table_name):
        """
        Gets data about a table in the keyspace.

        :param table_name: The name of the table to look up.
        :return: Data about the table.
        """
        try:
```

```
            response = self.keyspaces_client.get_table(
                keyspaceName=self.ks_name, tableName=table_name
            )
            self.table_name = table_name
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                logger.info("Table %s does not exist.", table_name)
                self.table_name = None
                response = None
            else:
                logger.error(
                    "Couldn't verify %s exists. Here's why: %s: %s",
                    table_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return response


    def list_tables(self):
        """
        Lists the tables in the keyspace.
        """
        try:
            table_paginator = self.keyspaces_client.get_paginator("list_tables")
            for page in table_paginator.paginate(keyspaceName=self.ks_name):
                for table in page["tables"]:
                    print(table["tableName"])
                    print(f"\t{table['resourceArn']}")
        except ClientError as err:
            logger.error(
                "Couldn't list tables in keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def update_table(self):
        """
        Updates the schema of the table.
```

```
        This example updates a table of movie data by adding a new column
        that tracks whether the movie has been watched.
        """
        try:
            self.keyspaces_client.update_table(
                keyspaceName=self.ks_name,
                tableName=self.table_name,
                addColumns=[{"name": "watched", "type": "boolean"}],
            )
        except ClientError as err:
            logger.error(
                "Couldn't update table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def restore_table(self, restore_timestamp):
        """
        Restores the table to a previous point in time. The table is restored
        to a new table in the same keyspace.

        :param restore_timestamp: The point in time to restore the table. This
time
                                  must be in UTC format.
        :return: The name of the restored table.
        """
        try:
            restored_table_name = f"{self.table_name}_restored"
            self.keyspaces_client.restore_table(
                sourceKeyspaceName=self.ks_name,
                sourceTableName=self.table_name,
                targetKeyspaceName=self.ks_name,
                targetTableName=restored_table_name,
                restoreTimestamp=restore_timestamp,
            )
        except ClientError as err:
            logger.error(
                "Couldn't restore table %s. Here's why: %s: %s",
                restore_timestamp,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
```

```
            )
            raise
        else:
            return restored_table_name


    def delete_table(self):
        """
        Deletes the table from the keyspace.
        """
        try:
            self.keyspaces_client.delete_table(
                keyspaceName=self.ks_name, tableName=self.table_name
            )
            self.table_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete table %s. Here's why: %s: %s",
                self.table_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise


    def delete_keyspace(self):
        """
        Deletes the keyspace.
        """
        try:
            self.keyspaces_client.delete_keyspace(keyspaceName=self.ks_name)
            self.ks_name = None
        except ClientError as err:
            logger.error(
                "Couldn't delete keyspace %s. Here's why: %s: %s",
                self.ks_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
```

定義一個類，該類建立與密鑰空間的 TLS 連接，使用 Sigv4 進行身份驗證，並將 CQL 查詢發送到密鑰空間中的表。

```
class QueryManager:
    """
    Manages queries to an Amazon Keyspaces (for Apache Cassandra) keyspace.
    Queries are secured by TLS and authenticated by using the Signature V4
 (SigV4)
    AWS signing protocol. This is more secure than sending username and password
    with a plain-text authentication provider.

    This example downloads a default certificate to secure TLS, or lets you
 specify
    your own.

    This example uses a table of movie data to demonstrate basic queries.
    """

    DEFAULT_CERT_FILE = "sf-class2-root.crt"
    CERT_URL = f"https://certs.secureserver.net/repository/sf-class2-root.crt"

    def __init__(self, cert_file_path, boto_session, keyspace_name):
        """
        :param cert_file_path: The path and file name of the certificate used for
 TLS.
        :param boto_session: A Boto3 session. This is used to acquire your AWS
 credentials.
        :param keyspace_name: The name of the keyspace to connect.
        """
        self.cert_file_path = cert_file_path
        self.boto_session = boto_session
        self.ks_name = keyspace_name
        self.cluster = None
        self.session = None

    def __enter__(self):
        """
        Creates a session connection to the keyspace that is secured by TLS and
        authenticated by SigV4.
        """
        ssl_context = SSLContext(PROTOCOL_TLSv1_2)
```

```
        ssl_context.load_verify_locations(self.cert_file_path)
        ssl_context.verify_mode = CERT_REQUIRED
        auth_provider = SigV4AuthProvider(self.boto_session)
        contact_point = f"cassandra.
{self.boto_session.region_name}.amazonaws.com"
        exec_profile = ExecutionProfile(
            consistency_level=ConsistencyLevel.LOCAL_QUORUM,
            load_balancing_policy=DCAwareRoundRobinPolicy(),
        )
        self.cluster = Cluster(
            [contact_point],
            ssl_context=ssl_context,
            auth_provider=auth_provider,
            port=9142,
            execution_profiles={EXEC_PROFILE_DEFAULT: exec_profile},
            protocol_version=4,
        )
        self.cluster.__enter__()
        self.session = self.cluster.connect(self.ks_name)
        return self

    def __exit__(self, *args):
        """
        Exits the cluster. This shuts down all existing session connections.
        """
        self.cluster.__exit__(*args)

    def add_movies(self, table_name, movie_file_path):
        """
        Gets movies from a JSON file and adds them to a table in the keyspace.

        :param table_name: The name of the table.
        :param movie_file_path: The path and file name of a JSON file that
 contains movie data.
        """
        with open(movie_file_path, "r") as movie_file:
            movies = json.loads(movie_file.read())
        stmt = self.session.prepare(
            f"INSERT INTO {table_name} (year, title, release_date, plot) VALUES
 (?, ?, ?, ?);"
        )
        for movie in movies[:20]:
            self.session.execute(
                stmt,
```

```
                parameters=[
                    movie["year"],
                    movie["title"],
                    date.fromisoformat(movie["info"]
["release_date"].partition("T")[0]),
                    movie["info"]["plot"],
                ],
            )

    def get_movies(self, table_name, watched=None):
        """
        Gets the title and year of the full list of movies from the table.

        :param table_name: The name of the movie table.
        :param watched: When specified, the returned list of movies is filtered
 to
                        either movies that have been watched or movies that have
 not
                        been watched. Otherwise, all movies are returned.
        :return: A list of movies in the table.
        """
        if watched is None:
            stmt = SimpleStatement(f"SELECT title, year from {table_name}")
            params = None
        else:
            stmt = SimpleStatement(
                f"SELECT title, year from {table_name} WHERE watched = %s ALLOW
 FILTERING"
            )
            params = [watched]
        return self.session.execute(stmt, parameters=params).all()

    def get_movie(self, table_name, title, year):
        """
        Gets a single movie from the table, by title and year.

        :param table_name: The name of the movie table.
        :param title: The title of the movie.
        :param year: The year of the movie's release.
        :return: The requested movie.
        """
        return self.session.execute(
            SimpleStatement(
                f"SELECT * from {table_name} WHERE title = %s AND year = %s"
```

```
            ),
            parameters=[title, year],
        ).one()

    def watched_movie(self, table_name, title, year):
        """
        Updates a movie as having been watched.

        :param table_name: The name of the movie table.
        :param title: The title of the movie.
        :param year: The year of the movie's release.
        """
        self.session.execute(
            SimpleStatement(
                f"UPDATE {table_name} SET watched=true WHERE title = %s AND year
= %s"
            ),
            parameters=[title, year],
        )
```

- 如需 API 的詳細資訊，請參閱《適用於 Python (Boto3) 的AWS SDK API 參考資料》中的下列主題。
  - CreateKeyspace
  - CreateTable
  - DeleteKeyspace
  - DeleteTable
  - GetKeyspace
  - GetTable
  - ListKeyspaces
  - ListTables
  - RestoreTable
  - UpdateTable

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱使用 Amazon Keyspaces 與 SDK AWS。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

亞馬遜密鑰空間（阿帕奇卡桑德拉）庫和工具

本節提供有關亞馬遜密鑰空間（Apache 卡桑德拉）庫，代碼示例和工具的信息。

主題

資料庫與範例

您可以GitHub在和AWS範例存放庫中找到 Amazon 密鑰空間開放原始碼程式庫AWS和開發人員工具。

亞馬遜密鑰空間（阿帕奇卡桑德拉）開發工具包

此儲存庫為 Amazon 密鑰空間提供 Docker 映像檔，其中包含實用的開發人員工具。例如，它包含具有最佳實務的 CQLSHRC 檔案、cqlsh 的選用AWS驗證擴充，以及執行一般工作的協助工具。該工具包針對亞馬遜密鑰空間進行了優化，但也適用於 Apache 卡桑德拉集群。

https://github.com/aws-samples/amazon-keyspaces-toolkit.

亞馬遜密鑰空間（阿帕奇卡桑德拉）的例子

這個回購是我們的亞馬遜密鑰空間示例代碼的官方列表。回購按語言細分為多個部分（請參閱示例）。每種語言都有自己的範例子節。這些範例示範建置應用程式時可以使用的常見 Amazon Keyspaces 服務實作和模式。

https://github.com/aws-samples/amazon-keyspaces-examples/.

AWS簽名版本 4（SIGv4）身份驗證插件

這些外掛程式可讓您使用 AWS Identity and Access Management (IAM) 使用者和角色來管理 Amazon 金鑰空間的存取。

爪哇:https://github.com/aws/aws-sigv4-auth-cassandra-java-driver-plugin.

Node.js:https://github.com/aws/aws-sigv4-auth-cassandra-nodejs-driver-plugin.

Python: https://github.com/aws/aws-sigv4-auth-cassandra-python-driver-plugin.

去：https://github.com/aws/aws-sigv4-auth-cassandra-gocql-driver-plugin。

突出顯示的樣本和開發人員工具存

以下是亞馬遜密鑰空間（對於 Apache 卡桑德拉）的選擇有用的社區工具。

亞馬遜鍵空間協議緩衝區

您可以使用協議緩衝區（Protobuf）與亞馬遜密鑰空間來提供一個替代 Apache 卡桑德拉用戶定義類型（UDT）。Protobuf 是一種免費的開源跨平台數據格式，用於序列化結構化數據。您可以使用 CQL 資料類型和重構 UDT 來儲存 Protobuf BL0B 資料，同時保留應用程式和程式設計語言之間的結構化資料。

此儲存庫提供連線至 Amazon 金鑰空間、建立新資料表以及插入包含 Protobuf 訊息的資料列的程式碼範例。然後以強烈的一致性讀取該行。

https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/java/datastax-v4/protobuf-user-defined-types

AWS CloudFormation模板為亞馬遜密鑰空間創建亞馬遜CloudWatch儀表板（阿帕奇卡桑德拉）指標

此儲存庫提供可快速設定 Amazon 金鑰空間CloudWatch指標的AWS CloudFormation範本。使用此範本可讓您透過提供可部署的預先建置CloudWatch儀表板及常用指標，讓您更輕鬆地開始使用。

https://github.com/aws-samples/amazon-keyspaces-cloudwatch-cloudformation-templates.

使用亞馬遜密鑰空間（阿帕奇卡桑德拉）與 AWS Lambda

儲存庫包含示範如何從 Lambda 連接到亞馬遜金鑰空間的範例。以下是一些例子。

C＃/. https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/dotnet/datastax-v3/connection-lambda

爪哇:https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/java/datastax-v4/connection-lambda.

另一個 Lambda 範例說明如何從 Python Lambda 部署和使用亞馬遜金鑰空間，可從下列存放庫取得。

https://github.com/aws-samples/aws-keyspaces-lambda-python

使用亞馬遜密鑰空間（阿帕奇卡桑德拉）與春天

這是一個示例，向您展示如何使用亞馬遜密鑰空間與春季啟動。

https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/java/datastax-v4/spring

使用亞馬遜密鑰空間（阿帕奇卡桑德拉）與斯卡拉

這是一個示例，說明如何使用 Sigv4 身份驗證插件與 Scala 連接到亞馬遜密鑰空間。

https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/scala/datastax-v4/connection-sigv4

使用亞馬遜密鑰空間（阿帕奇卡桑德拉）與 AWS Glue

這是一個示例，演示瞭如何使用亞馬遜密鑰空間與AWS Glue.

https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/scala/datastax-v4/aws-glue

亞馬遜密鑰空間（阿帕奇卡桑德拉）卡桑德拉查詢語言（CQL）轉換器 AWS CloudFormation

此套件實作命令列工具，可將 Apache Cassandra 查詢語言 (CQL) 指令碼轉換為 AWS CloudFormation (CloudFormation) 範本，讓 Amazon 金鑰空間架構在堆疊中輕鬆管理。CloudFormation

https://github.com/aws/amazon-keyspaces-cql-to-cfn-converter.

亞馬遜密鑰空間（阿帕奇卡桑德拉）助手為 Java 的阿帕奇卡桑德拉驅動程序

此儲存庫包含將 DataStax Java 驅動程式與亞馬遜密鑰空間搭配使用時的驅動程式政策、範例和最佳實務 (適用於 Apache Cassandra)。

https://github.com/aws-samples/amazon-keyspaces-java-driver-helpers.

亞馬遜密鑰空間（阿帕奇卡桑德拉）活潑的壓縮演示

此儲存庫示範如何壓縮、儲存和讀取/寫入大型物件，以提高效能並降低輸送量和儲存成本。

https://github.com/aws-samples/amazon-keyspaces-compression-example.

亞馬遜密鑰空間（阿帕奇卡桑德拉）和亞馬遜 S3 編解碼器演示

自訂 Amazon S3 轉碼器支援透明、使用者可設定的 UUID 指標對應至 Amazon S3 物件。

https://github.com/aws-samples/amazon-keyspaces-large-object-s3-demo.

整合亞馬遜密鑰空間與阿帕奇星火

阿帕奇星火是用於大規模數據分析的開源引擎。Apache Spark 可讓您更有效率地對儲存在 Amazon 金鑰空間中的資料執行分析。您還可以使用 Amazon 密鑰空間為應用程序提供一致的，single-digit-millisecond從 Spark 讀取分析資料的存取權。開源的星火卡桑德拉連接器簡化了亞馬遜密鑰空間和星火之間的讀取和寫入數據。

Amazon 金鑰空間支援 Spark Cassandra 連接器，可使用完全受管的無伺服器資料庫服務，簡化在以火花為基礎的分析管道中執行 Cassandra 工作負載。使用 Amazon 金鑰空間，您不必擔心 Spark 會爭奪與資料表相同的基礎設施資源。Amazon 密鑰空間表會根據您的應用程式流量自動擴展和縮減。

下列教學將引導您完成使用 Spark Cassandra 連接器將資料讀取和寫入 Amazon 金鑰空間所需的步驟和最佳實務。本教程演示如何通過使用 Spark Cassandra 連接器從文件中加載數據並將其寫入亞馬遜密鑰空間表，將數據遷移到亞馬遜密鑰空間。然後，本教程演示了如何使用星火卡桑德拉連接器從亞馬遜密鑰空間讀回數據。您可以這樣做，在基於 Spark 的分析管道中運行卡桑德拉工作負載。

主題

使用星火卡桑德拉連接器建立連接到亞馬遜密鑰空間的先決條件

在您使用星火卡桑德拉連接器連接到亞馬遜密鑰空間之前，您需要確保您已安裝以下內容。亞馬遜密鑰空間與星火卡桑德拉連接器的兼容性已通過以下推薦版本進行了測試：

- 爪哇版本 8
- 斯卡拉
- 星火花
- 卡桑德拉連接器 2.5 及更高版本
- 卡桑德拉驅動程式 4.12

1. 要安裝 Scala，請按照以下說明進行操作https://www.scala-lang.org/download/scala2.html。
2. 要安裝星火 3.4.1，請按照這個例子。

```
curl -o spark-3.4.1-bin-hadoop3.tgz -k https://dlcdn.apache.org/spark/spark-3.4.1/
spark-3.4.1-bin-hadoop3.tgz

# now to untar
tar -zxvf spark-3.4.1-bin-hadoop3.tgz

# set this variable.
export SPARK_HOME=$PWD/spark-3.4.1-bin-hadoop3
```

第 1 步：配置亞馬遜密鑰空間與 Apache 卡桑德拉星火連接器集成

在此步驟中，您會確認帳戶的磁碟分割程式與 Apache Spark 連接器相容，並設定必要的 IAM 許可。下列最佳作法可協助您為表格佈建足夠的讀取/寫入容量。

1. 確認Murmur3Partitioner分區程序是您帳戶的默認分區程序。該分區是與星火卡桑德拉連接器兼容。如需有關磁碟分割程式以及如何變更它們的詳細資訊，請參閱the section called “使用磁碟分割程式”。
2. 使用界面 VPC 端點和 Apache Spark，為亞馬遜密鑰空間設置 IAM 許可。
   - 指派讀取/寫入存取權給使用者資料表，以及對系統資料表的讀取存取權，如下列 IAM 政策範例所示。
   - 使用 Spark 結束存取亞馬遜金鑰空間的用戶端，必須使用可用的介面 VPC 端點填入 System.peers 表格虛擬私人雲端端。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "cassandra:Select",
            "cassandra:Modify"
         ],
         "Resource":[
```

```
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/
mytable",
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
         ]
      },
      {
         "Sid":"ListVPCEndpoints",
         "Effect":"Allow",
         "Action":[
            "ec2:DescribeNetworkInterfaces",
            "ec2:DescribeVpcEndpoints"
         ],
         "Resource":"*"
      }
   ]
}
```

3. 請考慮下列最佳實務，為 Amazon 金鑰空間表格設定足夠的讀取/寫入輸送量容量，以支援來自 Spark Cassandra 連接器的流量。

   - 開始使用隨需容量來協助您測試案例。
   - 若要最佳化生產環境的表格輸送量成本，請針對來自連接器的流量使用速率限制器，並將您的表格設定為使用具有自動擴充的佈建容量。如需詳細資訊，請參閱the section called “透過 auto 擴充管理輸送量容量”。
   - 您可以使用 Cassandra 驅動程序附帶的固定速率限制器。有一些針對 Amazon 密鑰空間量身定制的速率限制器在AWS樣本回購。
   - 如需容量管理的詳細資訊，請參閱the section called “讀/寫容量模式”。

第 2 步：配置 Apache 卡桑德拉星火連接器

Apache Spark 是一個通用的計算平台，您可以使用不同的方式進行配置。若要設定 Spark 和 Spark Cassandra 連接器以與 Amazon 金鑰空間整合，我們建議您從下一節所述的最低組態設定開始，然後在稍後根據您的工作負載增加它們。

- 創建星火分區大小小於 8 MB。

  在火花，分區代表可以並行運行的數據的原子塊。當您將數據寫入亞馬遜密鑰空間與星火卡桑德拉連接器，較小的 Spark 分區，該任務將要寫入的記錄量越小。如果 Spark 任務遇到多個錯誤，則在指定的重試次數用盡後失敗。為了避免重播大型任務並重新處理大量數據，請將 Spark 分區的大小保持較小。

- 每個執行程序使用較少的並發寫入次數和大量的重試次數。

  由於操作超時，亞馬遜密鑰空間將容量不足的錯誤返回給 Cassandra 驅動程序。您無法通過更改配置的超時持續時間來解決由於容量不足引起的超時，因為 Spark Cassandra 連接器嘗試使用透明地重試請求`MultipleRetryPolicy`。為了確保重試不會壓倒驅動程序的連接池，請使用每個執行程序的並發寫入次數少，並且重試次數大。下面的代碼片段是這樣的一個例子。

  ```
  spark.cassandra.query.retry.count = 500
  spark.cassandra.output.concurrent.writes = 3
  ```

- 分解總吞吐量並將其分配到多個 Cassandra 會話中。
  - 卡桑德拉星火連接器為每個星火執行程序創建一個會話。請將此工作階段視為縮放單位，以判斷所需的輸送量和所需的連線數目。
  - 在定義每個執行程序的內核數量和每個任務的內核數量時，請從低位開始並根據需要增加。
  - 設定 Spark 工作失敗，以便在發生暫時性錯誤時進行處理。熟悉應用程式的流量特性和需求之後，我們建議您進行設定`spark.task.maxFailures`到一個有界值。
  - 例如，以下配置可以處理每個會話每個執行程序兩個並發任務：

    ```
    spark.executor.instances = configurable -> number of executors for the session.
    spark.executor.cores = 2 -> Number of cores per executor.
    spark.task.cpus = 1 -> Number of cores per task.
    spark.task.maxFailures = -1
    ```

- 關閉批次處理。
  - 建議您關閉批次處理，以改善隨機存取模式。下面的代碼片段是這樣的一個例子。

    ```
    spark.cassandra.output.batch.size.rows = 1 (Default = None)
    spark.cassandra.output.batch.grouping.key = none (Default = Partition)
    spark.cassandra.output.batch.grouping.buffer.size = 100 (Default = 1000)
    ```

- 套裝**SPARK_LOCAL_DIRS**到具有足夠空間的快速本地磁盤。
  - 默認情況下，Spark 將地圖輸出文件和彈性分佈式數據集（RDD）保存到`/tmp` 資料夾。根據您的 Spark 主機的配置，這可能會導致設備上沒有剩餘空間型式錯誤。
  - 若要設定`SPARK_LOCAL_DIRS`環境變量到名為的目錄`/example/spark-dir`，您可以使用以下命令。

    ```
    export SPARK_LOCAL_DIRS=/example/spark-dir
    ```

步驟 3：建立應用程式設定檔

要使用開源的星火 Cassandra 連接器與亞馬遜密鑰空間，你需要提供一個應用程序配置文件，其中包含與連接所需的設置DataStaxJava 驅動程式。您可以使用服務特定憑證或 Sigv4 外掛程式進行連線。

如果您尚未這樣做，則需要將 Starfield 數位憑證轉換為 TrustStore 檔案。您可以按照以下步驟進行操作：the section called “開始之前”從 Java 驅動程序連接教程。請記下 TrustStore 檔案路徑和密碼，因為在建立應用程式設定檔時需要此資訊。

使用 Sigv4 驗證連線

本節為您展示了一個例子`application.conf`連線時可以使用的檔案AWS憑證和 Sigv4 外掛程式。如果您尚未這樣做，則需要產生 IAM 存取金鑰 (存取金鑰 ID 和秘密存取金鑰)，並將其儲存在您的AWS配置文件或作為環境變量。如需詳細說明，請參閱the section called “驗證所需的認 AWS 證”。

在下列範例中，取代信任庫檔案的檔案路徑，並取代密碼。

```
datastax-java-driver {
        basic.contact-points = ["cassandra.us-east-1.amazonaws.com:9142"]
        basic.load-balancing-policy {
            class = DefaultLoadBalancingPolicy
            local-datacenter = us-east-1
            slow-replica-avoidance = false
        }
        basic.request {
              consistency = LOCAL_QUORUM
        }
        advanced {
                auth-provider = {
                   class = software.aws.mcs.auth.SigV4AuthProvider
                   aws-region = us-east-1
                 }
            ssl-engine-factory {
                class = DefaultSslEngineFactory
                truststore-path = "path_to_file/cassandra_truststore.jks"
                truststore-password = "password"
        hostname-validation=false
            }
   }
        advanced.connection.pool.local.size = 3
}
```

更新並將此配置文件另存為/home/user1/application.conf。下列範例會使用此路徑。

使用服務特定認證連線

本節為您展示了一個例子application.conf使用服務特定認證連線時可使用的檔案。如果您尚未這樣做，則需要為 Amazon 金鑰空間產生服務特定的登入資料。如需詳細說明，請參閱the section called "服務特定認證"。

在下列範例中，取代username和password使用您自己的憑據。此外，請取代信任庫檔案的檔案路徑，並取代密碼。

```
datastax-java-driver {
        basic.contact-points = ["cassandra.us-east-1.amazonaws.com:9142"]
        basic.load-balancing-policy {
            class = DefaultLoadBalancingPolicy
            local-datacenter = us-east-1
        }
        basic.request {
              consistency = LOCAL_QUORUM
        }
        advanced {
            auth-provider = {
            class = PlainTextAuthProvider
                    username = "username"
                    password = "password"
                    aws-region = "us-east-1"
            }
            ssl-engine-factory {
                class = DefaultSslEngineFactory
                truststore-path = "path_to_file/cassandra_truststore.jks"
                truststore-password = "password"
                hostname-validation=false
            }
            metadata = {
                schema {
                     token-map.enabled = true
                }
            }
        }
}
```

更新並將此配置文件另存為/home/user1/application.conf與代碼示例一起使用。

以固定費率連線

要強制每個 Spark 執行程序固定速率，您可以定義請求節流器。此要求節流程式會限制每秒要求的速率。火花卡桑德拉連接器部署每個執行程序卡桑德拉會話。使用下列公式可協助您針對資料表達成一致的輸送量。

```
max-request-per-second * numberOfExecutors = total throughput against a table
```

您可以將此範例新增至先前建立的應用程式設定檔中。

```
datastax-java-driver {
  advanced.throttler {
    class = RateLimitingRequestThrottler

    max-requests-per-second = 3000
    max-queue-size = 30000
    drain-interval = 1 millisecond
  }
}
```

第 4 步：準備源數據和亞馬遜密鑰空間中的目標表

在此步驟中，您會建立包含範例資料和 Amazon 金鑰空間表格的來源檔案。

1. 建立來源檔案。您可以選擇以下其中一個選項：
   - 在本教學課程中，您會使用名稱為逗號分隔值 (CSV) 檔案`keyspaces_sample_table.csv`做為資料移轉的來源檔案。提供的範例檔案包含名稱為資料表的幾列資料`book_awards`。
     - 下載 CSV 檔案範例 (`keyspaces_sample_table.csv`)，包含在下列封存檔案中samplemigration.zip。解壓縮歸檔並記下的路徑`keyspaces_sample_table.csv`。
   - 如果您想要跟隨自己的 CSV 檔案將資料寫入 Amazon 金鑰空間，請確定資料是隨機化的。直接從資料庫讀取或匯出至純資料檔的資料通常會依分割區和主索引鍵排序。將已訂購的資料匯入 Amazon 金鑰空間可能會導致將其寫入 Amazon 金鑰空間分割區的較小區段，進而導致流量分配不均勻。這可能會導致效能變慢，錯誤率較高。

     相較之下，隨機化資料有助於利用 Amazon Keyspace 的內建負載平衡功能，方法是將流量分散到分區之間更均勻。您可以使用各種工具來隨機化資料。如需使用開放原始碼工具的範例舒

夫，請參閱the section called “步驟 2：準備資料”在〈資料移轉自學課程〉中。下面是一個示例，示範如何將數據洗牌為DataFrame。

```
import org.apache.spark.sql.functions.randval
shuffledDF = dataframe.orderBy(rand())
```

2. 在亞馬遜密鑰空間中創建目標密鑰空間和表。

   a. 使用連接到亞馬遜密鑰空間cqlsh，並以您自己的值取代下列範例中的服務端點、使用者名稱和密碼。

   ```
   cqlsh cassandra.us-east-2.amazonaws.com 9142 -u "111122223333" -
   p "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY" --ssl
   ```

   b. 使用名稱創建一個新的密鑰空間catalog如下列範例所示。

   ```
   CREATE KEYSPACE catalog WITH REPLICATION = {'class': 'SingleRegionStrategy'};
   ```

   c. 在新的密鑰空間狀態為可用之後，請使用下面的代碼創建目標表book_awards。若要深入瞭解非同步資源建立，以及如何檢查資源是否可用，請參閱the section called “創建密鑰空間”。

   ```
   CREATE TABLE catalog.book_awards (
      year int,
      award text,
      rank int,
      category text,
      book_title text,
      author text,
      publisher text,
      PRIMARY KEY ((year, award), category, rank)
      );
   ```

第 5 步：使用 Apache 卡桑德拉星火連接器寫入和讀取亞馬遜密鑰空間數據

在此步驟中，您首先將範例檔案中的資料載入DataFrame與火花卡桑德拉連接器。接下來，您可以從DataFrame進入您的亞馬遜密鑰空間表。您也可以單獨使用此部分，例如，將資料遷移到 Amazon 金鑰空間表格中。最後，您將表中的數據讀入DataFrame使用火花卡桑德拉連接器。例如，您也可以單獨使用此部分，從 Amazon 密鑰空間表中讀取資料，以便使用 Apache Spark 執行資料分析。

1. 啟動星火外殼，如下面的例子。請注意，此範例使用的是 SIGv4 驗證。

```
./spark-shell --files application.conf --conf
 spark.cassandra.connection.config.profile.path=application.conf
 --packages software.aws.mcs:aws-sigv4-auth-cassandra-java-driver-
plugin:4.0.5,com.datastax.spark:spark-cassandra-connector_2.12:3.1.0 --conf
 spark.sql.extensions=com.datastax.spark.connector.CassandraSparkExtensions
```

2. 導入火花卡桑德拉連接器與下面的代碼。

```
import org.apache.spark.sql.cassandra._
```

3. 若要從 CSV 檔案讀取資料並將其儲存在DataFrame，您可以使用下列程式碼範例。

```
var df =
 spark.read.option("header","true").option("inferSchema","true").csv("keyspaces_sample_tabl
```

您可以使用以下命令顯示結果。

```
scala> df.show();
```

輸出看起來應該類似於這個。

```
+----------------+----+-----------+----+------------------+--------------------
+-------------+
|           award|year|   category|rank|            author|          book_title|
 publisher|
+----------------+----+-----------+----+------------------+--------------------
+-------------+
|Kwesi Manu Prize|2020|    Fiction|   1|        Akua Mansa|   Where did you go?|
SomePublisher|
|Kwesi Manu Prize|2020|    Fiction|   2|       John Stiles|           Yesterday|
Example Books|
|Kwesi Manu Prize|2020|    Fiction|   3|        Nikki Wolf|Moving to the Cha...|
 AnyPublisher|
|            Wolf|2020|Non-Fiction|   1|       Wang Xiulan|    History of Ideas|
Example Books|
|            Wolf|2020|Non-Fiction|   2|Ana Carolina Silva|       Science Today|
SomePublisher|
|            Wolf|2020|Non-Fiction|   3| Shirley Rodriguez|The Future of Sea...|
 AnyPublisher|
```

```
|     Richard Roe|2020|    Fiction|   1| Alejandro Rosalez|         Long Summer|
SomePublisher|
|     Richard Roe|2020|    Fiction|   2|       Arnav Desai|             The Key|
Example Books|
|     Richard Roe|2020|    Fiction|   3|     Mateo Jackson|    Inside the Whale|
 AnyPublisher|
+----------------+----+-----------+----+------------------+--------------------
+-------------+
```

您可以確認資料的結構描述DataFrame如下列範例所示。

```
scala> df.printSchema
```

輸出應該是這樣的。

```
root
|-- award: string (nullable = true)
|-- year: integer (nullable = true)
|-- category: string (nullable = true)
|-- rank: integer (nullable = true)
|-- author: string (nullable = true)
|-- book_title: string (nullable = true)
|-- publisher: string (nullable = true)
```

4. 使用以下命令將數據寫入DataFrame到亞馬遜密鑰空間表。

```
df.write.cassandraFormat("book_awards", "catalog").mode("APPEND").save()
```

5. 若要確認資料已儲存，您可以將其讀回資料框，如下列範例所示。

```
var newDf = spark.read.cassandraFormat("book_awards", "catalog").load()
```

然後，您可以顯示現在包含在數據框中的數據。

```
scala> newDf.show()
```

該命令的輸出應該是這樣的。

```
+--------------------+------------------+----------------+-----------+-------------
+----+----+
```

```
|          book_title|            author|           award|   category|
 publisher|rank|year|
+--------------------+------------------+----------------+-----------+-------------
+----+----+
|         Long Summer| Alejandro Rosalez|     Richard Roe|    Fiction|
SomePublisher|   1|2020|
|    History of Ideas|       Wang Xiulan|            Wolf|Non-Fiction|Example
 Books|   1|2020|
|   Where did you go?|        Akua Mansa|Kwesi Manu Prize|    Fiction|
SomePublisher|   1|2020|
|    Inside the Whale|     Mateo Jackson|     Richard Roe|    Fiction|
 AnyPublisher|   3|2020|
|           Yesterday|       John Stiles|Kwesi Manu Prize|    Fiction|Example
 Books|   2|2020|
|Moving to the Cha...|        Nikki Wolf|Kwesi Manu Prize|    Fiction|
 AnyPublisher|   3|2020|
|The Future of Sea...| Shirley Rodriguez|            Wolf|Non-Fiction|
 AnyPublisher|   3|2020|
|       Science Today|Ana Carolina Silva|            Wolf|Non-Fiction|
SomePublisher|   2|2020|
|             The Key|       Arnav Desai|     Richard Roe|    Fiction|Example
 Books|   2|2020|
+--------------------+------------------+----------------+-----------+-------------
+----+----+
```

使用星火卡桑德拉連接器與亞馬遜密鑰空間時的常見錯誤進行故障排除

如果您使用 Amazon 虛擬私有雲並連接到 Amazon 密鑰空間，則使用 Spark 連接器時遇到的最常見錯誤是由以下組態問題引起的。

- VPC 中使用的 IAM 使用者或角色缺少存取`system.peers`亞馬遜密鑰空間中的表。如需詳細資訊，請參閱the section called “使用介面 VPC 端點資訊填入`system.peers`表格項目”。
- IAM 使用者或角色缺乏對使用者表格所需的讀取/寫入許可，以及對 Amazon 金鑰空間中系統資料表的讀取存取權限。如需詳細資訊，請參閱the section called “步驟 1：配置亞馬遜密鑰空間”。
- 建立 SSL/TLS 連線時，Java 驅動程式組態不會停用主機名稱驗證。如需範例，請參閱the section called “步驟 2：設定驅動程式”。

如需詳細的連線疑難排解步驟，請the section called “VPC 端點連線錯誤”。

此外，您可以使用亞馬遜CloudWatch指標可協助您疑難排解 Amazon 金鑰空間中 Spark 卡桑德拉連接器組態的問題。要了解有關使用 Amazon 密鑰空間的更多信息CloudWatch，請參閱the section called “使用監控 CloudWatch”。

以下部分描述了最有用的指標，以觀察當你使用星火卡桑德拉連接器。

PerConnectionRequestRateExceeded

Amazon 密鑰空間每個連線的配額為每秒 3,000 個請求。每個星火執行程序建立與亞馬遜密鑰空間的連接。執行多次重試可能會耗盡每個連線的要求速率配額。如果超過此配額，亞馬遜密鑰空間會發出一個`PerConnectionRequestRateExceeded`公制CloudWatch。

如果你看到PerConnectionRequestRateExceeded事件與其他系統或用戶錯誤一起出現，Spark 可能在每個連接的請求分配數量之外運行多次重試。

如果你看到`PerConnectionRequestRateExceeded`沒有其他錯誤的事件，那麼您可能需要增加驅動程序設置中的連接數以允許更多輸送量，或者您可能需要增加 Spark 作業中的執行程序數量。

StoragePartitionThroughputCapacityExceeded

Amazon 金鑰空間的配額為每秒 1,000 個 WCU 或 WRU/每秒 3,000 個 RCU 或每個分割區的 RU 或 RRU。如果你看到`StoragePartitionThroughputCapacityExceeded` CloudWatch事件，它可能表明數據在加載時不是隨機分配的。如需如何隨機排列資料的範例，請參閱the section called “步驟 4：準備源數據和目標表”。

常見錯誤和警告

如果您使用的是亞馬遜虛擬私有雲並連接到亞馬遜密鑰空間，則 Cassandra 驅動程序可能會在中發出有關控制節點本身的警告消息`system.peers`表。如需詳細資訊，請參閱the section called “常見錯誤和警告”。您可以放心地忽略此警告。

Amazon Keyspaces 故障排除（阿帕奇卡桑德拉）

以下各節提供有關如何解決使用 Amazon Keyspaces (適用於 Apache Cassandra) 時可能遇到的常見組態問題的相關資訊。

如需 IAM 存取特定的疑難排解指引，請參閱the section called “疑難排解”。

如需有關安全性最佳做法的詳細資訊，請參閱the section called “安全最佳實務”。

主題

- 疑難排解 Amazon Keyspaces 中的一般錯誤
- 疑難排解 Amazon Keyspaces 中的連線
- 疑難排解 Amazon Keyspaces 中的容量管理
- 疑難排解 Amazon Keyspaces 中的資料定義語言

疑難排解 Amazon Keyspaces 中的一般錯誤

得到一般錯誤？ 以下是一些常見問題以及如何解決這些問題。

一般錯誤

您會收到以下頂級異常之一，由於許多不同的原因而可能發生。

- `NoNodeAvailableException`
- `NoHostAvailableException`
- `AllNodesFailedException`

這些異常是由客戶端驅動程序生成的，可能發生在您建立控制連接或執行讀/寫/準備/執行/批處理請求時。

當您建立控制項連線時發生錯誤時，表示您的應用程式中指定的所有聯絡點都無法連線。執行讀取/寫入/準備/執行查詢時發生錯誤時，表示該請求的所有重試都已用盡。當您使用預設重試原則時，會在不同的節點上嘗試每次重試。

如何將基礎錯誤與頂級 Java 驅動程序異常隔離

這些一般錯誤可能是由於連接問題或執行讀取/寫入/準備/執行操作時引起的。在分佈式系統中必須預期暫時性故障，並且應該通過重試請求來處理。遇到連線錯誤時，Java 驅動程式不會自動重試，因此建議您在應用程式中建立驅動程式連線時實作重試原則。如需連線最佳做法的詳細概觀，請參閱the section called “連線”。

默認情況下，Java 驅動程序設置`idempotence`為 false 的所有請求，這意味著 Java 驅動程序不會自動重試失敗的讀/寫/準備請求。`idempotence`要設置`true`並告訴驅動程序重試失敗的請求，您可以通過幾種不同的方式進行此操作。以下是一個例子，如何以編程方式為 Java 應用程序中的單個請求設置等效性。

```
Statement s = new SimpleStatement("SELECT * FROM my_table WHERE id = 1");
s.setIdempotent(true);
```

或者，您可以通過編程方式設置整個 Java 應用程序的默認等效果，如下面的例子所示。

```
// Make all statements idempotent by default:
cluster.getConfiguration().getQueryOptions().setDefaultIdempotence(true);
//Set the default idempotency to true in your Cassandra configuration
basic.request.default-idempotence = true
```

另一個建議是在應用程式層級建立重試原則。在這種情況下，應用程序需要 catch `NoNodeAvailableException` 並重試請求。我們建議 10 次重試，其指數輪詢從 10 毫秒開始，最多工作 100 毫秒，所有重試的總時間為 1 秒。

另一種選擇是在 Github 上建立 Java 驅動程序連接時應用 Amazon Keyspaces 指數重試政策。

使用預設重試原則時，請確認您已建立與多個節點的連線。您可以使用 Amazon Keyspaces 下面的查詢來做到這一點。

```
SELECT * FROM system.peers;
```

如果此查詢的回應為空，表示您正在使用 Amazon Keyspaces 的單一節點。如果您使用預設的重試原則，將不會有重試，因為預設重試永遠發生在不同的節點上。若要進一步瞭解如何透過 VPC 端點建立連線，請參閱the section called “VPC 端連線”。

有關說明如何使用 Datastax 4.x 卡桑德拉驅動 step-by-step 程序建立到 Amazon Keyspaces 的連接的教程，請參閱。the section called “適用於 Java 4.x 的驗證外掛程式”

疑難排解 Amazon Keyspaces 中的連線

連線時遇到問題嗎？ 以下是一些常見問題以及如何解決這些問題。

連線到 Amazon Keyspaces 端點時發生錯誤

連線失敗和連線錯誤可能會導致不同的錯誤訊息。以下部分涵蓋了最常見的情況。

主題

我無法使用 cqlsh 連接到 Amazon Keyspaces

您嘗試使用 cqlsh 連接到 Amazon Keyspaces 端點，並且連接失敗與. **`Connection error`**

如果您嘗試連線到 Amazon Keyspaces 資料表，且 cqlsh 尚未正確設定，則連線會失敗。以下部分提供當您嘗試使用 cqlsh 建立連線時導致連線錯誤的最常見設定問題的範例。

> **Note**
>
> 如果您嘗試從 VPC 連接到 Amazon Keyspaces，則需要其他許可。若要使用 VPC 端點成功設定連線，請遵循中的the section called “連線至 VPC 端點”步驟。

您正在嘗試使用 cqlsh 連接到 Amazon Keyspaces，但是出現連接錯誤。**`timed out`**

如果您沒有提供正確的端口，則可能是這種情況，導致以下錯誤。

```
#  cqlsh cassandra.us-east-1.amazonaws.com 9140 -u "USERNAME" -p "PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.199': error(None,
 "Tried connecting to [('3.234.248.199', 9140)]. Last error: timed out")})
```

若要解決這個問題，請確認您使用連接埠 9142 進行連線。

您正在嘗試使用 cqlsh 連接到 Amazon Keyspaces，但是出現錯誤。**`Name or service not known`**

如果您使用的端點拼寫錯誤或不存在，則可能是這種情況。在下列範例中，端點名稱拼錯。

```
#  cqlsh cassandra.us-east-1.amazon.com 9142 -u "USERNAME" -p "PASSWORD" --ssl
```

```
Traceback (most recent call last):
  File "/usr/bin/cqlsh.py", line 2458, in >module>
    main(*read_options(sys.argv[1:], os.environ))
  File "/usr/bin/cqlsh.py", line 2436, in main
    encoding=options.encoding)
  File "/usr/bin/cqlsh.py", line 484, in __init__
    load_balancing_policy=WhiteListRoundRobinPolicy([self.hostname]),
  File "/usr/share/cassandra/lib/cassandra-driver-internal-only-3.11.0-bb96859b.zip/
cassandra-driver-3.11.0-bb96859b/cassandra/policies.py", line 417, in __init__
socket.gaierror: [Errno -2] Name or service not known
```

若要在使用公用端點進行連線時解決此問題，請從中選取可用的端點the section called “服務端點”，然後確認端點名稱沒有任何錯誤。如果您使用 VPC 端點進行連線，請確認 cqlsh 組態中的 VPC 端點資訊是否正確。

您嘗試使用 cqlsh 連接到 Amazon Keyspaces，但收到錯誤。**OperationTimedOut**

Amazon Keyspaces 要求為連線啟用 SSL，以確保強大的安全性。如果您收到下列錯誤，SSL 參數可能會遺失。

```
# cqlsh cassandra.us-east-1.amazonaws.com -u "USERNAME" -p "PASSWORD"
Connection error: ('Unable to connect to any servers', {'3.234.248.192':
 OperationTimedOut('errors=Timed out creating connection (5 seconds),
 last_host=None',)})
#
```

若要解決這個問題，請將下列旗標新增至 cqlsh 連線命令。

```
--ssl
```

您正在嘗試使用 cqlsh 連接到 Amazon Keyspaces，並且收到錯誤。**SSL transport factory requires a valid certfile to be specified**

在此情況下，遺失 SSL/TLS 憑證的路徑，導致下列錯誤。

```
# cat .cassandra/cqlshrc
[connection]
port = 9142
factory = cqlshlib.ssl.ssl_transport_factory
#
```

```
# cqlsh cassandra.us-east-1.amazonaws.com -u "USERNAME" -p "PASSWORD" --ssl
Validation is enabled; SSL transport factory requires a valid certfile to be specified.
 Please provide path to the certfile in [ssl] section as 'certfile' option in /
root/.cassandra/cqlshrc (or use [certfiles] section) or set SSL_CERTFILE environment
 variable.
#
```

若要解決此問題，請將路徑新增至電腦上的憑證檔案。

```
certfile =  path_to_file/sf-class2-root.crt
```

您嘗試使用 cqlsh 連接到 Amazon Keyspaces，但收到錯誤。**No such file or directory**

如果您電腦上的憑證檔案路徑錯誤，導致下列錯誤，可能就是這種情況。

```
# cat .cassandra/cqlshrc
[connection]
port = 9142
factory = cqlshlib.ssl.ssl_transport_factory

[ssl]
validate = true
certfile = /root/wrong_path/sf-class2-root.crt
#



# cqlsh cassandra.us-east-1.amazonaws.com -u "USERNAME" -p "PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.192': IOError(2, 'No
 such file or directory')})
#
```

若要解決此問題，請確認電腦上憑證檔的路徑正確無誤。

您嘗試使用 cqlsh 連接到 Amazon Keyspaces，但收到錯誤。**[X509] PEM lib**

如果 SSL/TLS 憑證檔案無效，則可能`sf-class2-root.crt`是這種情況，導致下列錯誤。

```
# cqlsh cassandra.us-east-1.amazonaws.com -u "USERNAME" -p "PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.241':
 error(185090057, u"Tried connecting to [('3.234.248.241', 9142)]. Last error: [X509]
 PEM lib (_ssl.c:3063)")})
```

```
#
```

若要解決此問題，請使用下列命令下載 Starfield 數位憑證。儲存在`sf-class2-root.crt`本機或主目錄中。

```
curl https://certs.secureserver.net/repository/sf-class2-root.crt -O
```

您嘗試使用 cqlsh 連接到 Amazon Keyspaces，但收到 SSL 錯誤。**unknown**

如果 SSL/TLS 憑證檔案`sf-class2-root.crt`為空，則可能是這種情況，導致下列錯誤。

```
# cqlsh cassandra.us-east-1.amazonaws.com -u "USERNAME" -p "PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.220': error(0,
 u"Tried connecting to [('3.234.248.220', 9142)]. Last error: unknown error
 (_ssl.c:3063)")})
#
```

若要解決此問題，請使用下列命令下載 Starfield 數位憑證。儲存在`sf-class2-root.crt`本機或主目錄中。

```
curl https://certs.secureserver.net/repository/sf-class2-root.crt -O
```

您嘗試使用 cqlsh 連接到 Amazon Keyspaces，但收到錯誤。**SSL: CERTIFICATE_VERIFY_FAILED**

如果無法驗證 SSL/TLS 憑證檔案，則可能是這種情況，導致下列錯誤。

```
Connection error: ('Unable to connect to any servers', {'3.234.248.223':
 error(1, u"Tried connecting to [('3.234.248.223', 9142)]. Last error: [SSL:
 CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:727)")})
```

若要解決此問題，請使用下列命令再次下載憑證檔案。儲存在`sf-class2-root.crt`本機或主目錄中。

```
curl https://certs.secureserver.net/repository/sf-class2-root.crt -O
```

您嘗試使用 cqlsh 連接到 Amazon Keyspaces，但收到錯誤。**Last error: timed out**

如果您未在 Amazon Amazon EC2 安全群組中設定 Amazon Keyspaces 的輸出規則，則可能是這種情況，這會導致下列錯誤。

```
# cqlsh cassandra.us-east-1.amazonaws.com 9142 -u  "USERNAME" -p "PASSWORD" --ssl
```

```
Connection error: ('Unable to connect to any servers', {'3.234.248.206': error(None,
 "Tried connecting to [('3.234.248.206', 9142)]. Last error: timed out")})
#
```

若要確認此問題是由 Amazon EC2 執行個體的組態造成，而不是`cqlsh`，您可以嘗試使用 AWS CLI，例如使用下列命令連線至金鑰空間。

```
aws keyspaces list-tables --keyspace-name 'my_keyspace'
```

如果此命令也逾時，表示未正確設定 Amazon EC2 執行個體。

若要確認您有足夠的權限可以存取 Amazon Keyspaces，您可以使用 AWS CloudShell 來連線。`cqlsh`如果該連線已建立，您需要設定 Amazon EC2 執行個體。

若要解決此問題，請確認您的 Amazon EC2 執行個體具有允許流量傳輸至 Amazon Keyspaces 的輸出規則。如果不是這種情況，則需要為 EC2 執行個體建立新的安全群組，並新增允許傳出流量至 Amazon Keyspaces 資源的規則。若要更新輸出規則以允許流量傳送至 Amazon Keyspaces，請從類型下拉式功能表中選擇 CQLSH/CASSAN DRA。

使用輸出流量規則建立新的安全性群組後，您需要將其新增至執行個體。選取執行個體，然後選擇動作、安全性，然後選擇變更安全性群組。使用輸出規則新增安全性群組，但請確定預設群組也可以使用。

如需如何檢視和編輯 EC2 輸出規則的詳細資訊，請參閱 Amazon EC2 使用者指南中的向安全群組新增規則。

您嘗試使用 cqlsh 連接到 Amazon Keyspaces，但收到錯誤。**Unauthorized**

如果您在 IAM 使用者政策中缺少 Amazon Keyspaces 許可，則可能是這種情況，導致以下錯誤。

```
# cqlsh cassandra.us-east-1.amazonaws.com 9142 -u  "testuser-at-12345678910" -p
 "PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.241':
 AuthenticationFailed('Failed to authenticate to 3.234.248.241: Error from server:
 code=2100 [Unauthorized] message="User arn:aws:iam::12345678910:user/testuser has no
 permissions."',)})
#
```

若要解決此問題，請確保 IAM 使用者`testuser-at-12345678910`具有存取 Amazon Keyspaces 的許可。如需授與 Amazon Keyspaces 存取權的 IAM 政策範例，請參閱the section called “身分型政策範例”。

如需 IAM 存取特定的疑難排解指引，請參閱the section called “疑難排解”。

您嘗試使用 cqlsh 連接到 Amazon Keyspaces，但收到錯誤。**Bad credentials**

如果用戶名或密碼錯誤，則可能是這種情況，導致以下錯誤。

```
# cqlsh cassandra.us-east-1.amazonaws.com 9142 -u  "USERNAME" -p "PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.248':
 AuthenticationFailed('Failed to authenticate to 3.234.248.248: Error from server:
 code=0100 [Bad credentials] message="Provided username USERNAME and/or password are
 incorrect"',)})
#
```

```
########################################################
```

> **Important**
>
> 如果您在嘗試與 cqlsh 連線時持續看到錯誤，請使用--debug選項重新執行命令，並在連絡時包含詳細的輸出。 AWS Support

我無法使用 Cassandra 客戶端驅動程序連接到 Amazon Keyspaces

以下部分顯示了與 Cassandra 客戶端驅動程序連接時最常見的錯誤。

您嘗試使用 DataStax Java 驅動程序連接到 Amazon Keyspaces 表，但收到**NodeUnavailableException**錯誤信息。

如果嘗試要求的連線中斷，就會導致下列錯誤。

```
[com.datastax.oss.driver.api.core.NodeUnavailableException: No connection
 was available to Node(endPoint=vpce-22ff22f2f22222fff-aa1bb234.cassandra.us-
west-2.vpce.amazonaws.com/11.1.1111.222:9142, hostId=1a23456b-
c77d-8888-9d99-146cb22d6ef6, hashCode=123ca4567)]
```

若要解決此問題，請尋找活動訊號值，如果活動訊號值較高，則將其降低至 30 秒。

```
advanced.heartbeat.interval = 30 seconds
```

然後查找關聯的超時，並確保該值設置為至少 5 秒。

```
advanced.connection.init-query-timeout = 5 seconds
```

您嘗試使用驅動程式和 Sigv4 外掛程式連線到 Amazon Keyspaces 資料表，但收到錯誤**AttributeError**訊息。

如果未正確設定認證，則會導致下列錯誤。

```
cassandra.cluster.NoHostAvailable: ('Unable to connect to any servers',
 {'44.234.22.154:9142': AttributeError("'NoneType' object has no attribute
 'access_key'")})
```

若要解決此問題，請確認您在使用 Sigv4 外掛程式時傳遞與 IAM 使用者或角色相關聯的登入資料。Sigv4 外掛程式需要下列憑證。

- AWS_ACCESS_KEY_ID— 指定與 IAM 使用者或角色相關聯的 AWS 存取金鑰。
- AWS_SECRET_ACCESS_KEY— 指定與存取金鑰相關聯的秘密金鑰。這基本上是存取金鑰的「密碼」。

若要進一步瞭解存取金鑰和 Sigv4 外掛程式，請參閱the section called "身分 AWS 驗證的 IAM 登入資"。

您嘗試使用驅動程序連接到 Amazon Keyspaces 表，但收到**PartialCredentialsError**錯誤信息。

如果遺失，可能會導致下列錯誤。AWS_SECRET_ACCESS_KEY

```
cassandra.cluster.NoHostAvailable: ('Unable to connect to any servers',
 {'44.234.22.153:9142':
 PartialCredentialsError('Partial credentials found in config-file, missing:
 aws_secret_access_key')})
```

若要解決此問題，請確認您在使用 Sigv4 外掛程式AWS_SECRET_ACCESS_KEY時同時傳遞AWS_ACCESS_KEY_ID和。若要進一步瞭解存取金鑰和 Sigv4 外掛程式，請參閱the section called "身分 AWS 驗證的 IAM 登入資"。

您嘗試使用驅動程序連接到 Amazon Keyspaces 表，但收到**Invalid signature**錯誤信息。

如果您使用了錯誤的認證，則可能是這種情況，導致以下錯誤。

```
cassandra.cluster.NoHostAvailable: ('Unable to connect to any servers',
 {'44.234.22.134:9142':
 AuthenticationFailed('Failed to authenticate to 44.234.22.134:9142: Error from server:
 code=0100
 [Bad credentials] message="Authentication failure: Invalid signature"')})
```

若要解決此問題，請確認您傳遞的登入資料與您設定用來存取 Amazon Keyspaces 的 IAM 使用者或角色相關聯。若要進一步瞭解存取金鑰和 Sigv4 外掛程式，請參閱the section called "身分 AWS 驗證的 IAM 登入資"。

我的 VPC 端點連線無法正常運作

您嘗試使用 VPC 端點連接到 Amazon Keyspaces，但收到令牌映射錯誤，或者您遇到輸送量低的情況。

如果未正確配置 VPC 端點連接，則可能是這種情況。

若要解決這些問題，請確認下列組態詳細資料。要按照 step-by-step 教程進行操作，以了解如何通過界面 VPC 端點為 Amazon Keyspaces 配置連接，請參閱。the section called "連線至 VPC 端點"

1. 確認用於連線至 Amazon Keyspaces 的 IAM 實體具有對使用者資料表的讀取/寫入存取權，以及對系統表格的讀取存取權，如下列範例所示。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "cassandra:Select",
            "cassandra:Modify"
         ],
         "Resource":[
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/
mytable",
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
         ]
      }
   ]
}
```

2. 確認用於連接到 Amazon Keyspaces 的 IAM 實體具有存取 Amazon EC2 執行個體上 VPC 端點資訊所需的讀取許可，如下列範例所示。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ListVPCEndpoints",
         "Effect":"Allow",
         "Action":[
            "ec2:DescribeNetworkInterfaces",
            "ec2:DescribeVpcEndpoints"
         ],
         "Resource":"*"
      }
   ]
}
```

> **Note**
>
> 受管政策`AmazonKeyspacesReadOnlyAccess_v2`和`AmazonKeyspacesFullAccess`包含允許 Amazon Keyspaces 存取 Amazon EC2 執行個體以讀取有關可用界面 VPC 端點的資訊的必要許可。

如需 VPC 端點的詳細資訊，請參閱 the section called “使用 Amazon Keyspaces 的界面 VPC 端點”

3. 確認 Java 驅動程式的 SSL 組態將主機名稱驗證設定為 false，如此範例所示。

```
hostname-validation = false
```

如需驅動程式組態的詳細資訊，請參閱the section called “步驟 2：設定驅動程式”。

4. 若要確認已正確設定 VPC 端點，您可以從 VPC 中執行下列陳述式。

> **Note**
>
> 您無法使用本機開發人員環境或 Amazon Keyspaces CQL 編輯器來確認此組態，因為它們使用公有端點。

```
SELECT peer FROM system.peers;
```

根據您的 VPC 設定和 AWS 區域，輸出內容應類似於此範例，並在 2 到 6 個具有私有 IP 位址的節點之間傳回。

```
peer
---------------
 192.0.2.0.15
 192.0.2.0.24
 192.0.2.0.13
 192.0.2.0.7
 192.0.2.0.8

(5 rows)
```

我無法連接使用 **cassandra-stress**

您嘗試使用**cassandra-stress**命令連接到 Amazon Keyspaces，但收到**SSL context**錯誤訊息。

如果您嘗試連接到 Amazon Keyspaces，但沒有正確設置信任庫，則會發生這種情況。Amazon Keyspaces 需要使用傳輸層安全性 (TLS) 來協助保護與用戶端的連線安全。

在此情況下，您會看到下列錯誤。

```
Error creating the initializing the SSL Context
```

若要解決此問題，請依照本主題the section called “開始之前”所示的指示設定信任存放區。

設置信任庫後，您應該可以使用以下命令進行連接。

```
./cassandra-stress user profile=./profile.yaml n=100 "ops(insert=1,select=1)"
 cl=LOCAL_QUORUM -node "cassandra.eu-north-1.amazonaws.com" -port native=9142
```

```
 -transport ssl-alg="PKIX" truststore="./cassandra_truststore.jks" truststore-
password="trustStore_pw" -mode native cql3 user="user_name" password="password"
```

我無法使用 IAM 身分進行連線

您嘗試使用 IAM 身分連接到 Amazon Keyspaces 表，但收到**Unauthorized**錯誤訊息。

如果您嘗試使用 IAM 身分 (例如 IAM 使用者) 連線到 Amazon Keyspaces 表，而不實作政策並先授予使用者所需的許可，就會發生這種情況。

在此情況下，您會看到下列錯誤。

```
Connection error: ('Unable to connect to any servers', {'3.234.248.202':
 AuthenticationFailed('Failed to authenticate to 3.234.248.202:
Error from server: code=2100 [Unauthorized] message="User
 arn:aws:iam::1234567890123:user/testuser has no permissions."',)})
```

若要解決此問題，請驗證 IAM 使用者的許可。要與標準驅動程序連接，用戶必須至少具有對系統表的SELECT訪問權限，因為大多數驅動程序在建立連接時讀取系統密鑰空間/表。

如需授與 Amazon Keyspaces 系統和使用者資料表存取權的 IAM 政策範例，請參閱the section called “訪問 Amazon Keyspaces 表”。

若要檢閱 IAM 特定的疑難排解章節，請參閱the section called “疑難排解”。

我正在嘗試使用 cqlsh 導入數據，並且與我的 Amazon Keyspaces 表的連接丟失

您嘗試使用 cqlsh 將數據上傳到 Amazon Keyspaces，但收到連接錯誤。

cqlsh 用戶端從伺服器接收到任何類型的連續三個錯誤之後，與 Amazon Keyspaces 的連線會失敗。cqlsh 用戶端失敗，並顯示下列訊息。

```
Failed to import 1 rows: NoHostAvailable - , will retry later, attempt 3 of 100
```

若要解決此錯誤，您需要確定要匯入的資料與 Amazon Keyspaces 中的資料表結構描述相符。檢閱匯入檔案是否有剖析錯誤。您可以嘗試使用 INSERT 陳述式來隔離錯誤，使用單一資料列。

用戶端會自動嘗試重新建立連線。

疑難排解 Amazon Keyspaces 中的容量管理

無伺服器容量有問題嗎？ 以下是一些常見問題以及如何解決這些問題。

無伺服器容量錯誤

本節概述如何辨識與無伺服器容量管理相關的錯誤，以及如何解決這些錯誤。例如，當您的應用程式超過佈建的輸送量容量時，您可能會發現容量不足的事件。

因為 Apache Cassandra 是設計用於在節點叢集上執行的軟體，因此它沒有與無伺服器功能 (例如輸送量容量) 相關的例外狀況訊息。大多數驅動程序只了解在 Apache 卡桑德拉可用的錯誤代碼，所以 Amazon Keyspaces 使用相同的一組錯誤代碼來保持兼容性。

若要將 Cassandra 錯誤對應至基礎容量事件，您可以使用 Amazon CloudWatch 監控相關的 Amazon Keyspaces 指標。造成用戶端錯誤的容量不足事件可根據造成事件的資源，分為下列三個群組：

- 表格 — 如果您為表格選擇佈建容量模式，且應用程式超過佈建的輸送量，則可能會發現容量不足的錯誤。如需詳細資訊，請參閱 the section called “讀/寫容量模式”。
- 分割區 — 如果指定分割區的流量超過 3,000 個 RCU 或 1,000 個 WCU，您可能會遇到容量不足的事件。我們建議您將流量統一分配到分割區之間，做為最佳作法。如需詳細資訊，請參閱 the section called “建立資料模型”。
- 連線 — 如果您超過每個連線每秒作業數目上限的配額，可能會發生輸送量不足的情況。若要增加輸送量，您可以在設定與驅動程式的連線時增加預設連線數目。

  若要了解如何設定 Amazon Keyspaces 的連線，請參閱the section called “如何設定連線”。如需最佳化透過 VPC 端點連線的詳細資訊，請參閱the section called “VPC 端連線”。

若要判斷導致傳回用戶端錯誤的容量不足事件的資源，您可以在 Amazon Keyspaces 主控台中查看儀表板。根據預設，主控台會在表格的 [容量] 索引標籤的 [容量和相關 CloudWatch 測量結果] 區段中，提供最常見容量和流量相關測量結果的彙總檢視。

要使用 Amazon 創建自己的儀表板 CloudWatch，請查看以下 Amazon Keyspaces 指標。

- `PerConnectionRequestRateExceeded`— 超出每個連線要求率配額的 Amazon Keyspaces 的請求。每個用戶端連線至 Amazon Keyspaces，每秒最多可支援 3000 個 CQL 請求。您可以建立多個連線，每秒執行 3000 個以上的要求。
- `ReadThrottleEvents`— 對超出表格讀取容量的 Amazon Keyspaces 的請求。
- `StoragePartitionThroughputCapacityExceeded`— 對超出分割區輸送量容量的 Amazon Keyspaces 儲存分割區的請求。Amazon Keyspaces 儲存分割區每秒最多可支援 1000 個 WCU/WRU 和每秒 3000 個 RCU/RRU。若要緩解這些例外狀況，我們建議您檢閱資料模型，以便將讀取/寫入流量分配到更多分割區。
- `WriteThrottleEvents`— 對超出表格寫入容量的 Amazon Keyspaces 的請求。

若要進一步瞭解 CloudWatch，請參閱the section called “使用監控 CloudWatch”。如需 Amazon Keyspaces 的所有可用 CloudWatch 指標清單，請參閱the section called “指標與維度”。

> **Note**
>
> 若要開始使用顯示 Amazon Keyspace 所有常用指標的自訂儀表板，您可以使用範AWS 例儲存庫 GitHub 中提供的預先建置 CloudWatch 範本。

主題

- 我收到用戶端驅動程式的容量NoHostAvailable不足錯誤
- 我在資料匯入期間收到寫入逾時錯誤
- 我看不到密鑰空間或表的實際存儲大小

我收到用戶端驅動程式的容量**NoHostAvailable**不足錯誤

您正在看到**Read_Timeout**或表格的**Write_Timeout**例外狀況。

重複嘗試在容量不足的 Amazon Keyspaces 表格中寫入或讀取，可能會導致驅動程式特有的用戶端錯誤。

用 CloudWatch 於監視佈建和實際輸送量指標，以及表格的容量事件不足。例如，沒有足夠輸送量容量的讀取要求會失敗，並出現例Read_Timeout外狀況，並張貼至ReadThrottleEvents量度。沒有足夠輸送量容量的寫入要求會失敗，並發生Write_Timeout例外狀況，並張貼至WriteThrottleEvents量度。如需這些指標的詳細資訊，請參閱 the section called “指標與維度”。

若要解決這些問題，請考慮下列其中一個選項。

- 增加表格的佈建輸送量，這是應用程式可以消耗的最大輸送量容量。如需詳細資訊，請參閱 the section called “讀取容量單位和寫入容量單位”。
- 讓服務透過自動調整來代表您管理輸送量容量。如需詳細資訊，請參閱 the section called “透過 auto 擴充管理輸送量容量”。
- 為表格選擇隨需容量模式。如需詳細資訊，請參閱 the section called “隨需容量模式”。

如果您需要增加帳戶的預設容量配額，請參閱配額。

您看到與超出分割區容量相關的錯誤。

當您看到錯誤時，`StoragePartitionThroughputCapacityExceeded`分區容量暫時超出。這可能由調適性容量或隨需容量自動處理。我們建議您檢閱資料模型，將讀取/寫入流量分配到更多分割區，以減輕這些錯誤。Amazon Keyspaces 儲存分割區每秒最多可支援 1000 個 WCU/WRU 和每秒 3000 個 RCU/RRU。若要深入瞭解如何改善資料模型，以便在更多分割區之間分配讀取/寫入流量，請參閱the section called “建立資料模型”。

`Write_Timeout`例外狀況也可能是由於同一邏輯磁碟分割中包含靜態和非靜態資料的並行寫入作業的速率提高所造成。如果流量預期會執行多個並行寫入作業，其中包含相同邏輯分割區內的靜態和非靜態資料，建議您分別撰寫靜態和非靜態資料。分別寫入資料也有助於最佳化輸送量成本。

您看到與超出連線要求率相關的錯誤訊息。

您看到的原`PerConnectionRequestRateExceeded`因是下列其中一個原因。

- 您可能沒有為每個工作階段設定足夠的連線。
- 您獲得的連線可能比可用的對等少，因為您沒有正確設定 VPC 端點權限。如需 VPC 端點原則的詳細資訊，請參閱the section called “使用 Amazon Keyspaces 的界面 VPC 端點”。
- 如果您使用的是 4.x 驅動程式，請檢查是否已啟用主機名稱驗證。驅動程式預設會啟用 TLS 主機名稱驗證。此組態會導致 Amazon Keyspaces 顯示為驅動程式的單節點叢集。建議您關閉主機名稱驗證。

我們建議您遵循這些最佳做法，以確保最佳化連線和輸送量：

- 設定 CQL 查詢輸送量調整。

  Amazon Keyspaces 每秒每個 TCP 連線最多支援 3,000 個 CQL 查詢，但驅動程式可建立的連線數目沒有限制。

  大多數開源卡桑德拉驅動程序建立了一個連接池卡桑德拉，並通過該連接池進行負載平衡查詢。Amazon Keyspaces 會向驅動程式公開 9 個對等 IP 位址。大多數驅動程序的默認行為是建立到每個對等 IP 地址的單個連接。因此，使用預設設定的驅動程式的最大 CQL 查詢輸送量將是每秒 27,000 個 CQL 查詢。

  若要增加此數目，我們建議您增加驅動程式在其連線集區中維護的每個 IP 位址的連線數目。例如，將每個 IP 位址的最大連線數設定為 2 會將驅動程式的最大輸送量增加一倍，達到每秒 54,000 個 CQL 查詢。
- 最佳化您的單一節點連線。

默認情況下，大多數開源 Cassandra 驅動程序在建立會話時建立到`system.peers`表中公告的每個 IP 地址的一個或多個連接。不過，某些組態可能會導致驅動程式連線到單一 Amazon Keyspaces IP 位址。如果驅動程式嘗試驗對等節點（例如 DataStax Java 驅動程式）的 SSL 主機名稱驗證，或透過 VPC 端點連線時，就會發生這種情況。

若要取得與連線至多個 IP 位址的驅動程式相同的可用性和效能，建議您執行下列動作：

- 根據所需的用戶端輸送量，將每個 IP 的連線數增加到 9 或更高。
- 建立自訂重試原則，以確保針對相同節點執行重試。
- 如果您使用 VPC 端點，請授與用於連線到 Amazon Keyspaces 存取權限的 IAM 實體，以查詢您的 VPC 以取得端點和網路界面資訊。這可改善負載平衡並增加讀取/寫入輸送量 如需詳細資訊，請參閱 ???。

我在資料匯入期間收到寫入逾時錯誤

使用**`cqlshCOPY`**指令上傳資料時，您收到逾時錯誤訊息。

```
Failed to import 1 rows: WriteTimeout - Error from server: code=1100 [Coordinator node
 timed out waiting for replica nodes' responses]
 message="Operation timed out - received only 0 responses." info={'received_responses':
 0, 'required_responses': 2, 'write_type': 'SIMPLE', 'consistency':
 'LOCAL_QUORUM'}, will retry later, attempt 1 of 100
```

Amazon Keyspaces 會使用`ReadTimeout`和`WriteTimeout`例外狀況來指示寫入請求何時因輸送量容量不足而失敗。為了協助診斷容量不足的例外狀況，Amazon Keyspaces 會在 Amazon CloudWatch 發佈下列指標。

- `WriteThrottleEvents`
- `ReadThrottledEvents`
- `StoragePartitionThroughputCapacityExceeded`

若要解決資料載入期間容量不足的錯誤，請降低每個 Worker 的寫入速率或總擷取率，然後重試上傳資料列。如需詳細資訊，請參閱 the section called “步驟 4：配`cqlsh COPY FROM`置設置”。如需更強大的資料上傳選項，請考慮使用可從GitHub 存放庫取得的 DSBulk。如需 step-by-step指示，請參閱the section called “使用 DSBulk 載入資料”。

我看不到密鑰空間或表的實際存儲大小

您看不到密鑰空間或表的實際存儲大小。

若要深入瞭解資料表的儲存空間大小，請參閱the section called “在資料表層級評估您的成本”。您也可以透過開始計算資料表中的資料列大小來預估儲存體大小。有關計算列大小的詳細說明，請參閱the section called “計算列大小”。

疑難排解 Amazon Keyspaces 中的資料定義語言

建立資源時遇到問題嗎？ 以下是一些常見問題以及如何解決這些問題。

資料定義語言錯誤

Amazon Keyspaces 會非同步執行資料定義語言 (DDL) 作業，例如建立和刪除金鑰空間和表格。如果應用程式在準備就緒之前嘗試使用該資源，則作業會失敗。

您可以在中監視新密鑰空間和表格的建立狀態 AWS Management Console，這表示索引鍵空間或資料表何時處於擱置狀態或作用中狀態。您也可以透過程式設計方式查詢系統結構描述資料表來監視新金鑰空間或資料表的建立狀態。準備好可供使用時，鍵空間或資料表會在系統結構描述中顯示。

> **Note**
>
> 要使用優化密鑰空間的創建 AWS CloudFormation，您可以使用此實用程序將 CQL 腳本轉 CloudFormation 換為模板。該工具可從GitHub 存儲庫中獲得。

主題

- 我創建了一個新的密鑰空間，但我無法查看或訪問它
- 我創建了一個新的表，但我無法查看或訪問它
- 我正在嘗試使用 Amazon Keyspaces point-in-time 恢復（PITR）恢復表，但還原失敗
- 我正在嘗試使用 INSERT/UPDATE 來編輯自定義的存留時間（TTL）設置，但操作失敗
- 我正在嘗試將數據上傳到我的 Amazon Keyspaces 表，並且出現有關超過列數的錯誤
- 我試圖刪除我的 Amazon Keyspaces 表中的數據，並且刪除範圍失敗

我創建了一個新的密鑰空間，但我無法查看或訪問它

您正在嘗試訪問新密鑰空間的應用程序收到錯誤。

如果您嘗試訪問仍在異步創建的新創建的 Amazon 密 Keyspaces 間密鑰空間，則會收到錯誤信息。下面的錯誤是一個例子。

```
InvalidRequest: Error from server: code=2200 [Invalid query] message="unconfigured
 keyspace mykeyspace"
```

建議檢查新密鑰空間何時可以使用的設計模式是輪詢 Amazon Keyspaces 間系統架構表（system_schema_mcs.*）。

如需詳細資訊，請參閱 the section called “創建密鑰空間”。

我創建了一個新的表，但我無法查看或訪問它

您正在嘗試存取新資料表的應用程式收到錯誤。

如果您嘗試訪問仍在異步創建的新創建的 Amazon Keyspaces 表，則會收到錯誤信息。例如，嘗試查詢尚無法使用的資料表會失敗，並顯示錯`unconfigured table`誤。

```
InvalidRequest: Error from server: code=2200 [Invalid query] message="unconfigured
 table mykeyspace.mytable"
```

嘗試查看表`sync_table()`失敗，並顯示`KeyError`.

```
KeyError: 'mytable'
```

建議檢查新資料表何時可供使用的設計模式是輪詢 Amazon Keyspaces 系統架構資料表 (system_schema_mcs.*)。

這是正在建立之資料表的範例輸出。

```
user-at-123@cqlsh:system_schema_mcs> select table_name,status from
 system_schema_mcs.tables where keyspace_name='example_keyspace' and
 table_name='example_table';

table_name | status

------------+----------

example_table | CREATING
```

```
(1 rows)
```

這是作用中資料表的範例輸出。

```
user-at-123@cqlsh:system_schema_mcs> select table_name,status from
 system_schema_mcs.tables where keyspace_name='example_keyspace' and
 table_name='example_table';

table_name | status

------------+----------

example_table | ACTIVE

(1 rows)
```

如需詳細資訊，請參閱 the section called “建立資料表”。

我正在嘗試使用 Amazon Keyspaces point-in-time 恢復（PITR）恢復表，但還原失敗

如果您嘗試使用 point-in-time 復原 (PITR) 還原 Amazon Keyspaces 表格，並且看到還原程序開始但未成功完成，則可能尚未設定此特定表格的還原程序所需的所有必要許可。

除了使用者許可之外，Amazon Keyspaces space 可能需要許可，才能代表您的主體在還原程序期間執行動作。如果資料表使用客戶管理金鑰加密，或者您使用的是限制傳入流量的 IAM 政策，就會發生這種情況。

例如，如果您在 IAM 政策中使用條件金鑰將來源流量限制到特定端點或 IP 範圍，則還原作業會失敗。若要允許 Amazon 金 Keyspaces 代表您的主體執行表格還原作業，您必須在 IAM 政策中新增`aws:ViaAWSService`全域條件金鑰。

如需有關還原資料表之權限的詳細資訊，請參閱the section called “還原權限”。

我正在嘗試使用 INSERT/UPDATE 來編輯自定義的存留時間（TTL）設置，但操作失敗

如果您嘗試插入或更新自訂 TTL 值，作業可能會失敗，並顯示下列錯誤。

```
TTL is not yet supported.
```

若要使`INSERT`用或`UPDATE`操作來指定列或欄的自訂 TTL 值，您必須先為表格啟用 TTL。您可以使用`ttl`自訂內容為表格啟用 TTL。

如需為表格啟用自訂 TTL 設定的詳細資訊，請參閱the section called “如何啟用使用自訂屬性 (TTL) 啟用使用自訂屬性 (TTL)。”。

我正在嘗試將數據上傳到我的 Amazon Keyspaces 表，並且出現有關超過列數的錯誤

您正在上傳資料，且已超過可同時更新的欄數。

當您的資料表結構定義超過 350 KB 的大小上限時，就會發生這個錯誤。如需詳細資訊，請參閱 配額。

我試圖刪除我的 Amazon Keyspaces 表中的數據，並且刪除範圍失敗

您嘗試通過分區鍵刪除數據並收到範圍刪除錯誤。

當您嘗試在一次刪除作業中刪除超過 1,000 個資料列時，就會發生此錯誤。

```
Range delete requests are limited by the amount of items that can be deleted in a
 single range.
```

如需詳細資訊，請參閱 the section called “範圍刪除”。

若要刪除單一分割區中 1,000 個以上的資料列，請考慮下列選項。

- 依分割區刪除 — 如果大部分的分割區都在 1,000 列以下，您可以嘗試依分割區刪除資料。如果分割區包含超過 1,000 個資料列，請嘗試改用叢集資料行刪除。
- 透過叢集欄刪除 — 如果您的模型包含多個叢集資料行，您可以使用資料行階層來刪除多個資料列。叢集資料行是巢狀結構，您可以透過對頂層資料行進行操作來刪除許多資料列。
- 依個別資料列刪除 — 您可以逐一查看資料列，並依其完整的主索引鍵 (分割資料行和叢集資料行) 刪除每一列。
- 最佳做法是考慮跨分區分割資料列 — 在 Amazon Keyspace 中，我們建議您將輸送量分配到表格分割區之間。這樣可以在實體資源之間平均分配資料和存取，以提供最佳的輸送量。如需詳細資訊，請參閱 the section called “建立資料模型”。

當您計劃針對繁重的工作負載進行刪除作業時，也請考慮下列建議。

- 使用 Amazon Keyspaces，分割區可以包含幾乎無限制的列數。這使您可以擴展分區比 100 MB 的傳統卡桑德拉指導「更寬」。隨著時間的推移，時間序列或分類帳在一千兆字節的數據上增長並不罕見。

- 使用 Amazon Keyspaces 時，當您必須針對繁重的工作負載執行刪除操作時，不需要考慮壓縮策略或標記。您可以刪除任意數量的資料，而不會影響讀取效能。

Amazon Keyspaces 中的無服務器資源管理（阿帕奇卡桑德拉）

Amazon Keyspaces（阿帕奇卡桑德拉）是無服務器的。Amazon Keyspaces 不會透過叢集中的節點部署、管理和維護工作負載的儲存和運算資源，而是直接將儲存和讀取/寫入輸送量資源配置到表格。

本章提供 Amazon Keyspaces 中無伺服器資源管理的詳細資訊。要了解如何使用 Amazon 監控無伺服器資源 CloudWatch，請參閱the section called “使用監控 CloudWatch”。

主題

- Amazon Keyspaces 中的存儲
- Amazon 密Keyspaces 間中的讀取/寫入容量模
- 使用 Amazon Keyspaces 自動擴展自 auto 管理輸送量容量
- 在 Amazon Keyspaces 中有效使用爆發容量
- 如何估計 Amazon Keyspaces 中的容量消耗

Amazon Keyspaces 中的存儲

Amazon Keyspaces（對於 Apache 卡桑德拉）根據存儲在表中存儲的實際數據自動佈建存儲到表中。您不需要預先佈建資料表的儲存空間。Amazon Keyspaces 會在您的應用程式寫入、更新和刪除資料時自動擴展和縮減您的表格儲存。與傳統的 Apache Cassandra 集群不同，Amazon Keyspaces 不需要額外的存儲來支持低級別的系統操作，如壓縮。您只需為使用的儲存空間付費。

Amazon Keyspaces 預設會以三個複寫係數設定金鑰空間。您無法修改複製因子。Amazon Keyspaces 會在多個 AWS 可用區域中自動複寫表格資料三次，以提供高可用性。Amazon Keyspaces 儲存體的每 GB 價格已包含複寫。有關更多信息，請參閱 Amazon Keyspaces（Apache 卡桑德拉）定價。

Amazon Keyspaces 會持續監控表格的大小，以判斷儲存費用。如需 Amazon Keyspaces 如何計算資料可計費大小的詳細資訊，請參閱。the section called “計算列大小”

Amazon 密Keyspaces 間中的讀取/寫入容量模

Amazon Keyspaces 具有兩種讀取/寫入容量模式，用於處理表格上的讀取和寫入：

- 按需 (預設)

- 佈建

您選擇的讀取/寫入容量模式會控制讀取和寫入輸送量的費用，以及管理表格輸送量容量的方式。

主題

- 隨需容量模式
- 佈建輸送量容量模式
- 管理和檢視容量模式
- 變更容量模式時的考量

隨需容量模式

Amazon Keyspaces（適用於 Apache Cassandra）隨需容量模式是一種靈活的計費選項，能夠在無需容量規劃的情況下每秒服務數千個請求。此選項提供讀取和寫入請求的 pay-per-request 定價，因此您只需按使用量付費。

當您選擇隨需模式時，Amazon Keyspaces 可以立即將表格的輸送量容量擴展到先前達到的任何流量層級，然後在應用程式流量減少時縮減。如果工作負載的流量層級達到新的峰值，服務會快速適應以增加表格的輸送量容量。您可以為新表和現有表格啟用隨需容量模式。

如果符合以下任何一項，則按需模式是一個不錯的選擇：

- 您建立工作負載不明的新資料表。
- 您有無法預期的應用程式流量。
- 您偏好僅支付您實際用量的輕鬆付費方式。

若要開始使用隨選模式，您可以建立新資料表或更新現有資料表，以使用主控台或使用幾行 Cassandra 查詢語言 (CQL) 程式碼來使用隨選容量模式。如需詳細資訊，請參閱 the section called “資料表”。

主題

- 讀取請求單位與寫入請求單位
- 峰值流量與擴展屬性
- 隨需容量模式的初始輸送量

讀取請求單位與寫入請求單位

使用隨需容量模式表，您不需要指定預期應用程式預先使用多少讀取和寫入輸送量。Amazon Keyspaces 會針對您在表格上執行的讀取和寫入 (以讀取請求單位 (RRU) 和寫入請求單位 (RRU) 計費。

- 對於大小不超過 4 KB 的資料列，一個 RRU 代表一LOCAL_QUORUM個 LOCAL_ONE讀取要求或兩個讀取要求。如果您需要讀取大於 4 KB 的資料列，讀取作業會使用額外的 RRU。所需的 RRU 總數取決於資料列大小，以及您要使用LOCAL_QUORUM還是LOCAL_ONE讀取一致性。例如，讀取 8 KB 資料列需要使用讀取一致性的 2 個 RRU；如果您選擇LOCAL_ONE讀LOCAL_QUORUM取一致性，則需要 1 個 RRU。
- 一個 WRU 代表一個寫入大小不超過 1 KB 的資料列。所有寫入都使用LOCAL_QUORUM一致性，使用輕量型交易 (LWT) 無需額外付費。如果您需要寫入大於 1 KB 的資料列，則寫入作業會使用額外的 WRU。所需的 WRU 總數取決於資料列大小。例如，如果您的資料列大小為 2 KB，則需要 2 個 WRU 才能執行一個寫入要求。

如需有關支援一致性層級的資訊，請參閱the section called “支持的卡桑德拉一致性級別”。

峰值流量與擴展屬性

使用隨需容量模式的 Amazon Keyspaces 表會自動調整應用程式的流量。隨需容量模式會立即因應，最高達到資料表峰值流量的兩倍。例如，應用程式的流量模式可能會在每秒 5,000 到 10,000 次LOCAL_QUORUM讀取之間變化，其中每秒 10,000 次讀取為上一個流量峰值。

透過此模式，隨需容量模式可立即容納每秒高達 20,000 次讀取的持續流量。如果您的應用程式維持每秒 20,000 次讀取的流量，該峰值就會成為新的先前峰值，讓後續流量每秒最多可達 40,000 次讀取。

如果您在表格上需要先前一個峰值的兩倍以上，Amazon Keyspaces space 會隨著流量增加而自動分配更多容量。這有助於確保您的表格具有足夠的輸送量容量來處理其他要求。但是，如果您在 30 分鐘內超過前一個峰值的兩倍，則可能會發現輸送量容量不足錯誤。

例如，假設應用程式的流量模式在每秒 5,000 到 10,000 次強烈一致性讀取之間變化，其中每秒 20,000 次讀取是先前達到的流量峰值。在這種情況下，該服務建議您將流量增長時間保持至少 30 分鐘，然後再驅動高達每秒 40,000 次讀取。

若要瞭解如何估計資料表的讀取和寫入容量消耗，請參閱the section called “估計容量消耗”。

若要深入瞭解帳戶的預設配額，以及如何增加配額，請參閱配額。

隨需容量模式的初始輸送量

如果您建立啟用隨選容量模式的新表格，或是第一次將現有資料表切換為隨需容量模式，則表格會有下列先前的尖峰設定，即使該資料表先前未使用隨選容量模式提供流量服務：

- 具有隨需容量模式的新建立表格：先前的尖峰為 2,000 個 WRU 和 6,000 RRU。您可以立即開車到上一個峰值的兩倍。這樣做可讓新建立的隨選資料表提供高達 4,000 個 WRU 和 12,000 個 RRU。
- 現有資料表已切換至隨選容量模式：先前的尖峰是為表格佈建的前一個 WCU 和 RCU 的一半，或新建立的資料表具有隨選容量模式的設定 (以較高者為準)。

佈建輸送量容量模式

如果您選擇佈建的輸送量容量模式，您可以指定應用程式所需的每秒讀取和寫入次數。這有助於您管理 Amazon Keyspaces 的使用情況，使其保持在或低於定義的請求率，以優化價格並保持可預測性。若要進一步了解佈建輸送量的自動調整，請參閱the section called “透過 auto 擴充管理輸送量容量”。

如果符合下列任一條件，則佈建輸送量容量模式是一個不錯的選擇：

- 您有可預期的應用程式流量。
- 您執行流量一致或逐漸增加的應用程式。
- 您可以預測容量需求以優化價格。

讀取容量單位和寫入容量單位

對於佈建的輸送量容量模式表格，您可以根據讀取容量單位 (RCU) 和寫入容量單位 (WCU) 指定輸送量容量：

- 對於大小不超過 4 KB 的資料列，一LOCAL_QUORUM個 RCU 代表每秒一次LOCAL_ONE讀取，或每秒讀取兩次。如果您需要讀取大於 4 KB 的資料列，讀取作業會使用額外的 RCU。

  所需的 RCU 總數取決於資料列大小，以及您是LOCAL_QUORUM否要LOCAL_ONE讀取。例如，如果您的資料列大小為 8 KB，則需要 2 個 RCU 來維持每秒一次LOCAL_QUORUM讀取，如果您選擇LOCAL_ONE讀取，則需要 1 個 RCU。
- 一個 WCU 表示大小不超過 1 KB 的資料列每秒寫入一次。所有寫入都使用LOCAL_QUORUM一致性，使用輕量型交易 (LWT) 無需額外付費。如果您需要寫入大於 1 KB 的資料列，則寫入作業會使用額外的 WCU。

所需的 WCU 總數取決於資料列大小。例如，如果您的資料列大小為 2 KB，則需要 2 個 WCU 來維持每秒一個寫入要求。如需如何預估資料表讀取和寫入容量耗用量的詳細資訊，請參閱the section called “估計容量消耗”。

如果您的應用程式讀取或寫入較大的資料列 (最大為 1 MB 的 Amazon Keyspaces 最大資料列大小)，則會消耗更多容量單位。若要深入瞭解如何估計資料列大小，請參閱the section called “計算列大小”。例如，假設您建立包含 6 個 RCU 和 6 個 WCU 的已佈建表格。透過這些設定，您的應用程式可執行下列項目：

- 執行LOCAL_QUORUM讀取速度高達每秒 24 KB (4 KB × 6 個 RCU)。
- 執行LOCAL_ONE讀取速度高達每秒 48 KB (讀取輸送量的兩倍)。
- 每秒最多可寫入 6 KB (1 KB × 6 個 WCU)。

佈建輸送量是應用程式可從表格中消耗的最大輸送量容量。如果您的應用程式超出佈建的輸送量容量，您可能會發現容量不足錯誤。

例如，沒有足夠輸送量容量的讀取要求會失敗，並出現例Read_Timeout外狀況，並張貼至ReadThrottleEvents量度。沒有足夠輸送量容量的寫入要求會失敗，並發生Write_Timeout例外狀況，並張貼至WriteThrottleEvents量度。

您可以使用 Amazon CloudWatch 監控佈建和實際輸送量指標，以及容量不足事件。如需這些指標的詳細資訊，請參閱 the section called “指標與維度”。

> **Note**
>
> 由於容量不足而導致重複的錯誤可能導致客戶端驅動程序特定的異常，例如 DataStax Java 驅動程序失敗，並顯示NoHostAvailableException.

若要變更表格的輸送量容量設定，您可以使用 AWS Management Console 或使用 CQL 的ALTER TABLE陳述式，如需詳細資訊，請參閱the section called “ALTER TABLE”。

若要深入瞭解帳戶的預設配額，以及如何增加配額，請參閱配額。

管理和檢視容量模式

您可以在 Amazon 密鑰空間系統金鑰空間中查詢系統表格，以檢閱有關表格的容量模式資訊。您也可以查看表格是否使用隨需或佈建的輸送量容量模式。如果表格設定了佈建的輸送量容量模式，您可以看到為表格佈建的輸送量容量。

範例

```
SELECT * from system_schema_mcs.tables where keyspace_name = 'mykeyspace' and
 table_name = 'mytable';
```

以隨需容量模式設定的表格會傳回下列資訊。

```
                    {
                        'capacity_mode': {
                            'last_update_to_pay_per_request_timestamp':
'1579551547603',
                            'throughput_mode': 'PAY_PER_REQUEST'
                         }
                    }
```

以佈建輸送量容量模式設定的表格會傳回下列項目。

```
                    {
                        'capacity_mode': {
                            'last_update_to_pay_per_request_timestamp':
'1579048006000',
                            'read_capacity_units': '5000',
                            'throughput_mode': 'PROVISIONED',
                            'write_capacity_units': '6000'
                        }
                     }
```

該`last_update_to_pay_per_request_timestamp`值以毫秒為單位測量。

若要變更表格的佈建輸送量容量，請使用the section called “ALTER TABLE”。

變更容量模式時的考量

當您將表格從佈建的容量模式切換到隨需容量模式時，Amazon Keyspaces space 會對資料表和分區的結構進行一些變更。此程序需要幾分鐘的時間。在交換期間，您的表格會提供與先前佈建的 WCU 和 RCU 金額一致的輸送量。

當您從隨需容量模式切換回佈建容量模式時，您的表格會提供與表格設定為隨需容量模式時達到的先前峰值一致的輸送量。

> **Note**
>
> 您只能在 24 小時內將容量模式從佈建切換為隨需一次。

使用 Amazon Keyspaces 自動擴展自 auto 管理輸送量容量

許多資料庫工作負載本來就具週期性或難以事先預測。例如，假設有一個社交聯網應用程式，其中大部分使用者會在日間活動。資料庫必須能夠處理日間活動，但夜間則不需要同樣多的輸送量。

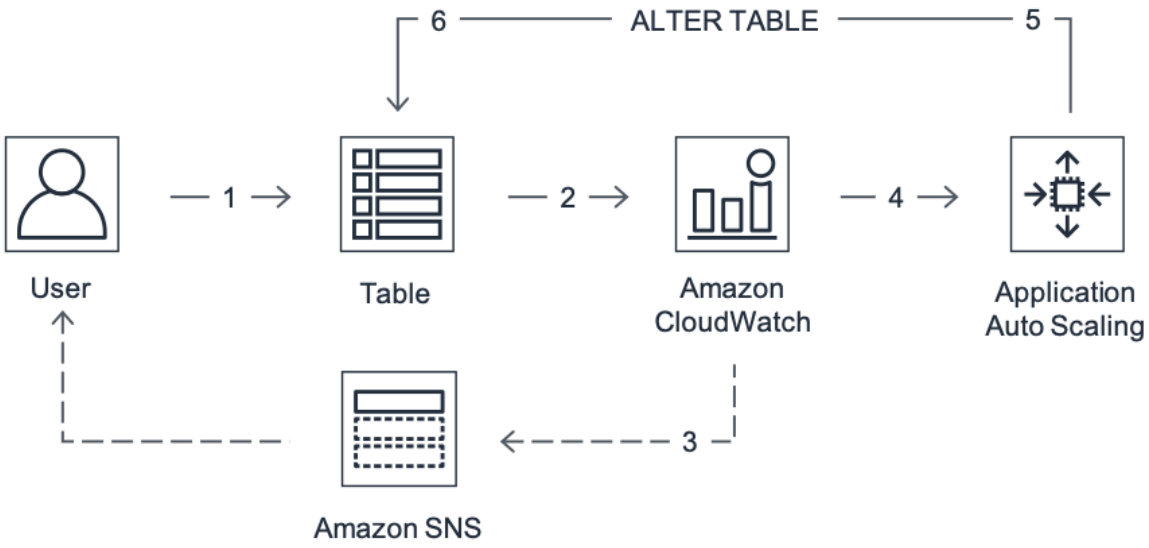
另一個範例則是正被快速採用的新行動遊戲應用程式。如果遊戲變得非常受歡迎，它可能會超過可用的數據庫資源，這將導致性能下降和不滿意的客戶。這類工作負載通常需要手動介入來擴展或縮減資料庫資源，以回應不斷改變的用量。

Amazon Keyspaces (適用於 Apache Cassandra) 可透過自動調整輸送量容量以回應實際應用程式流量，協助您有效地為可變工作負載佈建輸送量容量。Amazon Keyspaces 使用 Application Auto Scaling 服務代表您增加和減少表格的讀取和寫入容量。如需有關 Application Auto Scaling 的詳細資訊，請參閱應用程式自動調整規模

> **Note**
>
> 若要開始使用 Amazon Keyspaces 快速自動擴展，請參閱the section called “使用主控台”。若要使用卡桑德拉查詢語言 (CQL) 管理 Amazon Keyspaces 擴展政策，請參閱。the section called “使用定制列表”要了解如何使用 CLI 管理 Amazon Keyspaces 擴展政策，請參閱the section called “使用 CLI”。

Amazon Keyspaces 自動擴展如何工作

下圖提供 Amazon Keyspaces 自動擴展如何管理表格輸送量容量的高階概觀。



若要啟用資料表的自動調度資源，您可以建立資源調度政策。擴展政策指定是否要擴展讀取容量或寫入容量 (或兩者)，以及表格佈建的最小和最大容量單位設定。

擴展政策也會定義目標使用率。目標使用率是在某個時間點使用的容量單位與佈建容量單位的比率，以百分比表示。自動擴展使用目標追蹤演算法，向上或向下調整表格的佈建輸送量，以回應實際工作負載。這樣做是為了讓實際的容量使用率保持在目標使用率或接近目標使用率。

您可以將讀取和寫入容量的自動調整目標使用率值設定在 20% 到 90% 之間。預設目標使用率為 70%。如果流量快速變化，並且希望容量更快開始擴展，則可以將目標使用率設定為較低的百分比。如果應用程式流量變更緩慢，而且想要降低輸送量成本，您也可以將目標使用率設定為較高的速率。

如需有關擴展政策的詳細資訊，請參閱《應用 Ap plication Auto Scaling Scaling 使用者指南》中的 Application Auto Scaling 的目標追蹤

當您建立擴展政策時，Amazon Keyspaces 會代表您建立兩對 Amazon CloudWatch 警示。每個配對代表您已佈建和已耗用輸送量設定的上限和下限。當表格的實際使用率在持續一段時間內偏離目標使用率時，就會觸發這些 CloudWatch 警示。要了解有關 Amazon 的更多信息 CloudWatch，請參閱 Amazon CloudWatch 用戶指南。

觸發其中一個 CloudWatch 警示時，Amazon Simple Notification Service (Amazon SNS) 會傳送通知給您 (如果您已啟用它)。然後 CloudWatch 警示會叫 Application Auto Scaling，以評估您的資源調整政策。這反過來會向 Amazon Keyspaces 發出 Alter Table 請求，以適當地向上或向下調整表格的佈建容量。若要進一步了解有關 Amazon SNS 通知的資訊，請參閱設定 Amazon SNS 通知。

Amazon Keyspaces 會透過增加 (或減少) 表格的佈建輸送量容量來處理 Alter Table 請求，以便接近目標使用率。

> **Note**
>
> 只有當實際工作負載保持提高 (或壓抑) 持續數分鐘時，Amazon Keyspaces auto 擴展才會修改佈建的輸送量設定。 目標追蹤演算法會設法長期將目標使用率保持在或接近您選擇的值。資料表的內建高載容量可應付短期遽增的活動。

多區域表格 auto 縮放的運作方式

為了確保在佈建容量模式下，所有多區域表格中的所有表格複本始終有足夠 AWS 區域 的讀取和寫入容量，建議您設定 Amazon Keyspaces auto 擴展。

當您在具有 auto 擴展的佈建模式下使用多區域表格時，您無法停用單一表格複本的 auto 調整規模。但是您可以針對不同區域調整表格的讀取 auto 縮放設定。例如，您可以為複寫表格的每個區域指定不同的讀取容量和讀取 auto 調整設定。

您為指定區域中的表格複本所設定的讀取 auto 動調整比例設定會覆寫表格的一般 auto 調整比例設定。但是，寫入容量必須在所有表格複本之間保持同步，以確保有足夠的容量可以在所有區域中複寫寫入。

Amazon Keyspaces auto 擴展會 AWS 區域 根據該區域的使用情況獨立更新每個表格的佈建容量。因此，當 auto 擴展作用中時，多區域表格中每個區域的佈建容量可能會有所不同。

您可以使用 Amazon Keyspaces 主控台、API 或 CQL 來設定多區域表格及其複本的 auto 擴展設定。AWS CLI如需有關如何建立和更新多區域表格 auto 調整比例設定的詳細資訊，請參閱the section called “如何使用多區域複寫”。

> **Note**
>
> 如果您對多區域表使用 auto 擴展，則必須始終使用 Amazon Keyspaces API 操作來設定 auto 擴展設定。如果您直接使用應用程式 auto to Scaling API 操作來設定自動調整設定，則無法指定多區域表格 AWS 區域 的功能。這可能會導致不支援的組態。

使用須知

在開始使用 Amazon Keyspaces 自動擴展之前，您應該注意以下事項：

- 根據您的擴展政策，Amazon Keyspaces 自動擴展可視需要增加讀取容量或寫入容量。所有 Amazon Keyspaces 配額仍然有效，如中配額所述。
- Amazon Keyspaces 自動擴展不會阻止您手動修改佈建的輸送量設定。這些手動調整不會影響任何附加至資源調整政策的現有 CloudWatch 警示。
- 如果您使用主控台建立具有佈建輸送量容量的表格，則預設會啟用 Amazon Keyspaces 自動擴展。您可以隨時修改自動縮放設定。如需詳細資訊，請參閱 the section called “使用主控台”。
- 如果您使 AWS CloudFormation 用建立擴展政策，則應該從中管理擴展政策， AWS CloudFormation 以便堆疊與堆疊範本同步。如果您從 Amazon Keyspaces 變更擴展政策，當堆疊重設時，它們將被 AWS CloudFormation 堆疊範本中的原始值覆寫。
- 如果您使 CloudTrail 用監控 Amazon Keyspaces 自動擴展，您可能會在其組態驗證程序中看到應用程式 Auto Scaling 所發出的呼叫提醒。您可以使用包含`application-autoscaling.amazonaws.com`這些驗證檢查的`invokedBy`欄位來篩選掉這些警示。

使用主控台管理 Amazon Keyspaces 自動擴展政策

您可以使用主控台為新表和現有表格啟用 Amazon Keyspaces 自動擴展。您也可以使用主控台修改自動縮放設定或停用自動調整比例。

> **Note**
>
> 如需設定縮放和向外延展冷卻時間等進階功能，請使用 CQL 或 AWS Command Line Interface (AWS CLI) 以程式設計方式管理 Amazon Keyspaces 擴展政策。如需詳細資訊，請參閱 使用卡桑德拉查詢語言（CQL）管理 Amazon Keyspaces auto 擴展 或 使用 CLI 管理 Amazon Keyspaces 擴展政策 。

主題

- 開始之前:為 Amazon Keyspaces 授予使用者許可自動擴展
- 在啟用 Amazon Keyspaces 自動擴展的情況下創建新表
- 在現有表上啟用 Amazon Keyspaces 自動擴展
- 修改或停用 Amazon Keyspaces 自動擴展設定
- 在主控台上查看 Amazon Keyspaces 自動擴展活動

開始之前:為 Amazon Keyspaces 授予使用者許可自動擴展

若要開始使用，請確認使用者具有建立和管理自動資源調整設定的適當權限。在 AWS Identity and Access Management (IAM) 中，管理 Amazon Keyspaces 擴展政策`AmazonKeyspacesFullAccess`需要 AWS 受管政策。

> **Important**
>
> `application-autoscaling:*`需要權限才能停用資料表的自動調整比例。您必須先關閉表格的 auto 縮放功能，才能刪除該表格。

若要為 Amazon 金鑰空間主控台存取權和 Amazon 金鑰空間自動擴展設定 IAM 使用者，請新增下列政策。

若要附加**`AmazonKeyspacesFullAccess`**原則

1. 登入 AWS Management Console 並開啟身分與存取權管理主控台，網址為 https://console.aws.amazon.com/iam/。
2. 在 IAM 主控台儀表板上，選擇 Users (使用者)，然後從清單中選擇 IAM 使用者。
3. 在 Summary (摘要) 頁面上，選擇 Add permissions (新增許可)。
4. 選擇 Attach existing policies directly (直接連接現有政策)。
5. 從原則清單中選擇 AmazonKeyspacesFullAccess，然後選擇 [下一步：複查]。
6. 選擇新增許可。

在啟用 Amazon Keyspaces 自動擴展的情況下創建新表

> **Note**
>
> Amazon Keyspaces 自動擴展需要存在代表您執行自動擴展動作的服務連結角色 (`AWSServiceRoleForApplicationAutoScaling_CassandraTable`)。系統會自動建立此角色。如需詳細資訊，請參閱 the section called “使用服務連結角色”。

建立啟用自動調整比例的新表格

1. 登錄到 AWS Management Console，然後打開 Amazon Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。
2. 在導覽窗格中，選擇 Tables (資料表)，然後選擇 Create table (建立資料表)。
3. 在「表格詳細資訊」段落的「建立表格」頁面中，選取索引鍵空間並輸入新表格的名稱。
4. 在「欄」區段中，建立資料表的結構定義。
5. 在 [主索引鍵] 區段中，定義資料表的主索引鍵，並選取選用的叢集資料行。
6. 在 [表格設定] 區段中，選擇 [自訂設定]。
7. 繼續參閱讀/寫入容量設定。
8. 對於容量模式，選擇已佈建。
9. 在 [讀取容量] 區段中，確認已選取 [自動擴充]。

   在此步驟中，您可以選取表格的最小和最大讀取容量單位，以及目標使用率。

   - 最小容量單位 — 輸入表格應隨時可支援的最小輸送量層次值。此值必須介於 1 和帳戶每秒輸送量上限配額之間 (預設為 40,000)。
   - 最大容量單位 — 輸入您要為表格佈建的最大輸送量量。此值必須介於 1 和帳戶每秒輸送量上限配額之間 (預設為 40,000)。
   - 目標使用率 — 輸入介於 20% 到 90% 之間的目標使用率。當流量超過定義的目標使用率時，容量會自動擴充。當流量低於定義的目標時，它會再次自動縮小。

   > **Note**
   >
   > 若要深入瞭解帳戶的預設配額，以及如何增加配額，請參閱配額。

10. 在 [寫入容量] 區段中，選擇與上一步驟中定義的讀取容量相同的設定，或手動設定容量值。
11. 選擇 建立資料表 。您的表格是使用指定的自動縮放參數建立的。

在現有表上啟用 Amazon Keyspaces 自動擴展

> **Note**
>
> Amazon Keyspaces 自動擴展需要存在代表您執行自動擴展動作的服務連結角色 (`AWSServiceRoleForApplicationAutoScaling_CassandraTable`)。系統會自動建立此角色。如需詳細資訊，請參閱 the section called “使用服務連結角色”。

為現有表啟用 Amazon Keyspaces 自動擴展

1. 登錄到 AWS Management Console，然後打開 Amazon Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。
2. 選擇您要使用的表格，然後前往容量索引標籤。
3. 在 [容量設定] 區段中，選擇 [編輯]。
4. 在容量模式下，確定表格使用已佈建容量模式。
5. 選取 [自動調整] 並參閱步驟 6 中的在啟用 Amazon Keyspaces 自動擴展的情況下創建新表以編輯讀取和寫入容量。
6. 定義自動縮放設定後，請選擇「儲存」。

修改或停用 Amazon Keyspaces 自動擴展設定

您可以使用修 AWS Management Console 改您的 Amazon Keyspaces 自動擴展設定。若要執行此操作，請選擇您要編輯的表格，然後前往 [容量] 索引標籤。在 [容量設定] 區段中，選擇 [編輯]。您現在可以修改 [讀取容量] 或 [寫入容量] 區段中的設定。如需這些設定的詳細資訊，請參閱 在啟用 Amazon Keyspaces 自動擴展的情況下創建新表。

若要停用 Amazon Keyspaces 自動擴展，請清除「自動擴展」核取方塊。停用自動調整比例會使用「應用程式自動調整」將資料表取消註冊為可縮放 若要刪除 Application Auto Scaling 用來存取 Amazon Keyspaces 表格的服務連結角色，請依照中的步驟執行。the section called “刪除 Amazon Keyspaces 的服務鏈接角色”

> **Note**
>
> 若要刪除 Application Auto Scaling 使用的服務連結角色，您必須停用帳戶中所有資料表的自動調度資源。 AWS 區域

在主控台上查看 Amazon Keyspaces 自動擴展活動

您可以使用 Amazon 監控 Amazon Keyspaces 自動擴展如何使用資源 CloudWatch，Amazon 會產生有關您的使用情況和效能的指標。按照使Application Auto Scaling 用者指南中的步驟建立 CloudWatch 儀表板。

使用卡桑德拉查詢語言（CQL）管理 Amazon Keyspaces auto 擴展

要使用 Cassandra 查詢語言（CQL）創建和管理 Amazon Keyspaces 表的 auto 擴展設置，您可以使用. `cqlsh` 本主題提供您可以使用 CQL 以程式設計方式管理的 auto 調整規模工作的概觀。

如需本主題所描述之 CQL 陳述式的詳細資訊，請參閱the section called “DDL 陳述式”。

主題

- 開始之前
- 使用 CQL 建立具有自動調整比例的新資料表
- 使用 CQL 在現有資料表上啟用自動調整
- 使用 CQL 檢視表格的 Amazon Keyspaces auto 擴展組態
- 使用 CQL 關閉表的 Amazon Keyspaces auto 擴展

開始之前

您必須先完成下列工作，才能開始。

設定許可

如果您尚未這麼做，則必須為使用者設定適當的權限，才能建立和管理自動資源調整設定。在 AWS Identity and Access Management (IAM) 中，管理 Amazon Keyspaces 擴展政策`AmazonKeyspacesFullAccess`需要 AWS 受管政策。如需詳細步驟，請參閱the section called “開始之前:為 Amazon Keyspaces 授予使用者許可自動擴展”。

設定 **`cqlsh`**

如果您尚未這樣做，則必須安裝和配置`cqlsh`。若要執行這項操作，請依照中的指示執行the section called “使用 `cqlsh-expansion`”。然後您可以使 AWS CloudShell 用執行下列各節中的指令。

使用 CQL 建立具有自動調整比例的新資料表

當您建立新的 Amazon Keyspaces 資料表時，您可以在CREATE TABLE陳述式中為表格的寫入或讀取容量自動啟用自動擴展。這可讓 Amazon Keyspaces 代表您聯絡 Application Auto Scaling，將表格註冊為可擴展的目標，並調整佈建的寫入或讀取容量。

如需如何為資料表複本建立多區域表格及設定不同 auto 調整設定的詳細資訊，請參閱。the section called “使用默認設置（CQL）創建多區域表”

> **Note**
>
> Amazon Keyspaces auto 動擴展需要存在服務連結角色 (AWSServiceRoleForApplicationAutoScaling_CassandraTable)，才能代表您執行自動擴展動作。系統會自動建立此角色。如需詳細資訊，請參閱 the section called “使用服務連結角色”。

若要以程式設計方式設定資料表的 auto 擴展設定，您可以使用包含 Amazon Keyspaces auto 擴展參數的AUTOSCALING_SETTINGS陳述式。這些參數定義了指示 Amazon Keyspaces 調整表格佈建輸送量的條件，以及要採取的其他選用動作。在此範例中，您會定義 mytable 的 auto 縮放設定。

該政策包含下列元素：

- AUTOSCALING_SETTINGS— 指定是否允許 Amazon Keyspaces 代表您調整輸送量容量。以下是必要的值：
  - provisioned_write_capacity_autoscaling_update:
    - minimum_units
    - maximum_units
  - provisioned_read_capacity_autoscaling_update:
    - minimum_units
    - maximum_units
  - scaling_policy— Amazon Keyspaces 支持目標跟踪政策。若要定義目標追蹤原則，請設定下列參數。
    - target_value— Amazon Keyspaces auto 擴展可確保消耗容量與佈建容量的比例保持在或接近此值。您能以百分比的形式定義 target_value。

- disableScaleIn: (選擇性) Aboolean，指定表格scale-in是否已停用或啟用。依預設，會停用此參數。若要開啟scale-in，請將boolean值設定為FALSE。這表示會代表您自動縮減資料表的容量。
- scale_out_cooldown— 向外延展活動會增加表格的佈建輸送量。若要增加向外延展活動的冷卻時間，請指定一個值 (以秒為單位)。scale_out_cooldown如果未指定值，則預設值為 0。如需目標追蹤和冷卻時間的詳細資訊，請參閱《應用程式自動調整規模使用者指南》中的目標追蹤擴展政策
- scale_in_cooldown— 縮放活動會降低表格的佈建輸送量。若要增加縮放活動的冷卻時間，請指定一個值 (以秒為單位)。scale_in_cooldown如果未指定值，則預設值為 0。如需目標追蹤和冷卻時間的詳細資訊，請參閱《應用程式自動調整規模使用者指南》中的目標追蹤擴展政策

> **Note**
>
> 為了進一步了解 target_value 如何運作，請假設您資料表的佈建輸送量設定為 200 個寫入容量單位。您決定為此資料表建立擴展政策，並將 target_value 設為70 %。
> 現在，假設您開始將寫入流量導向該資料表，那實際的寫入輸送量就會是 150 個容量單位。現在的 consumed-to-provisioned 比率是 (一百五十/二百)，或百分之七十五。此比率超過您的目標，因此 auto 擴展會將佈建的寫入容量增加到 215，以便比例為 (150/ 215) 或 69.77 百分比 — target_value 盡可能接近，但不超過它。

對於 mytable，您可以將讀TargetValue取和寫入容量設定為 50%。Amazon Keyspaces auto 擴展可在 5 到 10 個容量單位範圍內調整表格的佈建輸送量，使 consumed-to-provisioned 比率維持在 50% 或接近 50%。對於讀取容量，您可ScaleInCooldown以將值設定為 ScaleOutCooldown 60 秒。

您可以使用下列陳述式建立已啟用 auto 擴展功能的新 Amazon Keyspaces 資料表。

```
CREATE TABLE mykeyspace.mytable(pk int, ck int, PRIMARY KEY (pk, ck))
WITH CUSTOM_PROPERTIES = {
    'capacity_mode': {
        'throughput_mode': 'PROVISIONED',
        'read_capacity_units': 1,
        'write_capacity_units': 1
    }
} AND AUTOSCALING_SETTINGS = {
    'provisioned_write_capacity_autoscaling_update': {
        'maximum_units': 10,
        'minimum_units': 5,
```

```
        'scaling_policy': {
            'target_tracking_scaling_policy_configuration': {
                'target_value': 50
            }
        }
    },
    'provisioned_read_capacity_autoscaling_update': {
        'maximum_units': 10,
        'minimum_units': 5,
        'scaling_policy': {
            'target_tracking_scaling_policy_configuration': {
                'target_value': 50,
                'scale_in_cooldown': 60,
                'scale_out_cooldown': 60
            }
        }
    }
};
```

使用 CQL 在現有資料表上啟用自動調整

對於現有的 Amazon Keyspaces 資料表，您可以使用`ALTER TABLE`陳述式為資料表的寫入或讀取容量開啟 auto 擴展。如果您正在更新目前處於隨需容量模式的資料表，則不`capacity_mode`是必要的。如果您的表格已經處於佈建容量模式，則可以省略此欄位。

> **Note**
>
> Amazon Keyspaces 自動擴展需要存在代表您執行自動擴展動作的服務連結角色 (`AWSServiceRoleForApplicationAutoScaling_CassandraTable`)。系統會自動建立此角色。如需詳細資訊，請參閱 the section called “使用服務連結角色”。

在下列範例中，陳述式會更新處於隨選容量模式的資料表 mytable。陳述式會將表格的容量模式變更為啟用 auto 擴展的佈建模式。

寫入容量設定在 5 到 10 個容量單位的範圍內，目標值為 50%。讀取容量也設定在 5—10 個容量單位的範圍內，目標值為 50%。對於讀取容量，您可`scale_in_cooldown`以將值設定為 `scale_out_cooldown` 60 秒。

```
ALTER TABLE mykeyspace.mytable
WITH CUSTOM_PROPERTIES = {
```

```
    'capacity_mode': {
        'throughput_mode': 'PROVISIONED',
        'read_capacity_units': 1,
        'write_capacity_units': 1
    }
} AND AUTOSCALING_SETTINGS = {
    'provisioned_write_capacity_autoscaling_update': {
        'maximum_units': 10,
        'minimum_units': 5,
        'scaling_policy': {
            'target_tracking_scaling_policy_configuration': {
                'target_value': 50
            }
        }
    },
    'provisioned_read_capacity_autoscaling_update': {
        'maximum_units': 10,
        'minimum_units': 5,
        'scaling_policy': {
            'target_tracking_scaling_policy_configuration': {
                'target_value': 50,
                'scale_in_cooldown': 60,
                'scale_out_cooldown': 60
            }
        }
    }
};
```

使用 CQL 檢視表格的 Amazon Keyspaces auto 擴展組態

若要檢視資料表 auto 調整比例組態的詳細資訊，請使用下列指令。

```
SELECT * FROM system_schema_mcs.autoscaling WHERE keyspace_name = 'mykeyspace' AND
 table_name = 'mytable';
```

此命令的輸出如下所示。

```
 keyspace_name | table_name | provisioned_read_capacity_autoscaling_update

                                                                                |
 provisioned_write_capacity_autoscaling_update
```

```
--------------+------------
+-------------------------------------------------------------------------------------------------
+-------------------------------------------------------------------------------------------------
 mykeyspace    | mytable    | {'minimum_units': 5, 'maximum_units':
 10, 'scaling_policy': {'target_tracking_scaling_policy_configuration':
 {'scale_out_cooldown': 60, 'disable_scale_in': false, 'target_value':
 50, 'scale_in_cooldown': 60}}} | {'minimum_units': 5, 'maximum_units':
 10, 'scaling_policy': {'target_tracking_scaling_policy_configuration':
 {'scale_out_cooldown': 0, 'disable_scale_in': false, 'target_value': 50,
 'scale_in_cooldown': 0}}}
```

使用 CQL 關閉表的 Amazon Keyspaces auto 擴展

您可以隨時關閉表格的 Amazon Keyspaces 間 auto 擴展。如果您不再需要擴展表格的讀取或寫入容量，則應考慮關閉 auto 擴展，這樣 Amazon Keyspaces space 就不會繼續修改表格的讀取或寫入容量設定。您可以使用陳述式更新資料`ALTER TABLE`表。

下列陳述式會針對資料表 mytable 的寫入容量關閉 auto 調整規模。它也會刪除代表您建立的 CloudWatch 警報。

```
ALTER TABLE mykeyspace.mytable
WITH AUTOSCALING_SETTINGS = {
    'provisioned_write_capacity_autoscaling_update': {
        'autoscaling_disabled': true
    }
};
```

> **Note**
>
> 若要刪除 Application Auto Scaling 使用的服務連結角色，您必須停用帳戶中所有資料表的自動調度資源。 AWS 區域

使用 CLI 管理 Amazon Keyspaces 擴展政策

若要以程式設計方式更新和管理 Amazon Keyspaces auto 擴展設定，您可以使用 AWS Command Line Interface (AWS CLI) 或 AWS API。若要使用卡桑德拉查詢語言 (CQL) 管理 Amazon Keyspaces auto 擴展政策，請參閱。the section called “使用定制列表”本主題提供您可以使用程式設計方式管理的 auto 調整比例工作的概觀 AWS CLI。

如需本主題所述之 Amazon Keyspaces 命 AWS CLI 令的詳細資訊，請參閱命AWS CLI 令參考。

主題

開始之前

您必須先完成下列工作，才能開始。

設定許可

如果您尚未這麼做，則必須為使用者設定適當的權限，才能建立和管理自動資源調整設定。在 AWS Identity and Access Management (IAM) 中，管理 Amazon Keyspaces 擴展政策`AmazonKeyspacesFullAccess`需要 AWS 受管政策。如需詳細步驟，請參閱the section called “開始之前:為 Amazon Keyspaces 授予使用者許可自動擴展”。

安裝 AWS CLI

若您尚未執行此作業，您必須安裝及設定 AWS CLI。若要這麼做，請前往 AWS Command Line Interface 使用者指南，並依照下列指示進行：

- 安裝 AWS CLI
- 設定 AWS CLI

使用建立具有自動縮放比例的新資料表 AWS CLI

當您建立新的 Amazon Keyspaces 資料表時，您可以在`CreateTable`作業中自動為資料表的寫入或讀取容量啟用自動擴展。這可讓 Amazon Keyspaces 代表您聯絡 Application Auto Scaling，將您指定的表格註冊為可擴展目標，並調整佈建的寫入或讀取容量。

如需如何使用 auto 調整規模組態建立多區域表格的詳細資訊，請參閱the section called “使用 auto 擴展 (CLI) 在佈建模式中建立新的多區域表格”。

> **Note**
>
> Amazon Keyspaces auto 動擴展需要存在服務連結角色 (AWSServiceRoleForApplicationAutoScaling_CassandraTable)，才能代表您執行自動擴展動作。系統會自動建立此角色。如需詳細資訊，請參閱 the section called “使用服務連結角色”。

若要以程式設計方式設定表格的 auto autoScalingSpecification 動擴展設定，您可以使用定義 Amazon Keyspaces auto 動擴展參數的動作。這些參數定義了指示 Amazon Keyspaces 調整表格佈建輸送量的條件，以及要採取的其他選用動作。在此範例中，您會定義 mytable 的 auto 縮放設定。

該政策包含下列元素：

- autoScalingSpecification— 指定是否允許 Amazon Keyspaces 代表您調整容量輸送量。您可以分別為讀取和寫入容量啟用 auto 調整功能。然後，您必須為以下參數指定autoScalingSpecification：
  - writeCapacityAutoScaling— 最大和最小寫入容量單位。
  - readCapacityAutoScaling— 最大和最小讀取容量單位。
  - scalingPolicy— Amazon Keyspaces 支持目標跟踪政策。若要定義目標追蹤原則，請設定下列參數。
    - targetValue— Amazon Keyspaces auto 擴展可確保消耗容量與佈建容量的比例保持在或接近此值。您能以百分比的形式定義 targetValue。
    - disableScaleIn: (選擇性) Aboolean，指定表格scale-in是否已停用或啟用。依預設，會停用此參數。若要開啟scale-in，請將boolean值設定為FALSE。這表示會代表您自動縮減資料表的容量。
    - scaleOutCooldown— 向外延展活動會增加表格的佈建輸送量。若要增加向外延展活動的冷卻時間，請指定一個值 (以秒為單位)。ScaleOutCooldown預設值為 0。如需目標追蹤和冷卻時間的詳細資訊，請參閱《應用程式自動調整規模使用者指南》中的目標追蹤擴展政策
    - scaleInCooldown— 縮放活動會降低表格的佈建輸送量。若要增加縮放活動的冷卻時間，請指定一個值 (以秒為單位)。ScaleInCooldown預設值為 0。如需目標追蹤和冷卻時間的詳細資訊，請參閱《應用程式自動調整規模使用者指南》中的目標追蹤擴展政策

> **Note**
>
> 為了進一步了解 TargetValue 如何運作，請假設您資料表的佈建輸送量設定為 200 個寫入容量單位。您決定為此資料表建立擴展政策，並將 TargetValue 設為70 %。
> 現在，假設您開始將寫入流量導向該資料表，那實際的寫入輸送量就會是 150 個容量單位。現在的 consumed-to-provisioned 比率是 (一百五十/二百)，或百分之七十五。此比率超過您的目標，因此 auto 擴展會將佈建的寫入容量增加到 215，以便比例為 (150/ 215) 或 69.77 百分比 — TargetValue 盡可能接近，但不超過它。

對於 mytable，您可以將讀TargetValue取和寫入容量設定為 50%。Amazon Keyspaces auto 擴展可在 5 到 10 個容量單位範圍內調整表格的佈建輸送量，使 consumed-to-provisioned 比率維持在 50% 或接近 50%。對於讀取容量，您可ScaleInCooldown以將值設定為 ScaleOutCooldown 60 秒。

使用複雜的 auto 縮放設定建立資料表時，從 JSON 檔案載入 auto 縮放設定會很有幫助。在下列範例中，您可以從 auto-scaling.zip 下載範例 JSON 檔案並擷取auto-scaling.json，並記下檔案的路徑。在此範例中，JSON 檔案位於目前的目錄中。如需不同的檔案路徑選項，請參閱如何從檔案載入參數。

```
aws keyspaces create-table --keyspace-name mykeyspace --table-name mytable
            \ --schema-definition 'allColumns=[{name=pk,type=int},
{name=ck,type=int}],partitionKeys=[{name=pk},{name=ck}]'
            \ --capacity-specification
 throughputMode=PROVISIONED,readCapacityUnits=1,writeCapacityUnits=1
            \ --auto-scaling-specification file://auto-scaling.json
```

在現有資料表上啟用自動調整比例 AWS CLI

對於現有的 Amazon Keyspaces 表格，您可以使用UpdateTable操作為表格的寫入或讀取容量開啟 auto 擴展。如需如何更新多區域表格之 auto 調整比例設定的詳細資訊，請參閱the section called “更新多區域表格 (CLI) 的佈建容量和 auto 調整規模設定”。

> **Note**
>
> Amazon Keyspaces 自動擴展需要存在代表您執行自動擴展動作的服務連結角色 (AWSServiceRoleForApplicationAutoScaling_CassandraTable)。系統會自動建立此角色。如需詳細資訊，請參閱 the section called “使用服務連結角色”。

您可以使用下列命令為現有資料表開啟 Amazon Keyspaces auto 擴展。資料表的 auto 縮放設定是從 JSON 檔案載入。在下列範例中，您可以從 auto-scaling.zip 下載範例 JSON 檔案並擷取`auto-scaling.json`，並記下檔案的路徑。在此範例中，JSON 檔案位於目前的目錄中。如需不同的檔案路徑選項，請參閱如何從檔案載入參數。

如需下列範例中使用之 auto 縮放比例設定的詳細資訊，請參閱the section called “使用建立具有自動縮放比例的新資料表 AWS CLI”。

```
aws keyspaces update-table --keyspace-name mykeyspace --table-name mytable
           \ --capacity-specification
 throughputMode=PROVISIONED,readCapacityUnits=1,writeCapacityUnits=1
           \ --auto-scaling-specification file://auto-scaling.json
```

檢視表格的 Amazon Keyspaces auto 擴展組態，請使用 AWS CLI

要查看表的 auto 縮放配置，可以使用`get-table-auto-scaling-settings`操作。下面的 CLI 命令就是這樣的一個例子。

```
aws keyspaces get-table-auto-scaling-settings --keyspace-name mykeyspace --table-name
 mytable
```

此命令的輸出如下所示。

```
{
    "keyspaceName": "mykeyspace",
    "tableName": "mytable",
    "resourceArn": "arn:aws:cassandra:us-east-1:5555-5555-5555:/keyspace/mykeyspace/
table/mytable",
    "autoScalingSpecification": {
        "writeCapacityAutoScaling": {
            "autoScalingDisabled": false,
            "minimumUnits": 5,
            "maximumUnits": 10,
            "scalingPolicy": {
                "targetTrackingScalingPolicyConfiguration": {
                    "disableScaleIn": false,
                    "scaleInCooldown": 0,
                    "scaleOutCooldown": 0,
                    "targetValue": 50.0
                }
            }
```

```
        },
        "readCapacityAutoScaling": {
            "autoScalingDisabled": false,
            "minimumUnits": 5,
            "maximumUnits": 10,
            "scalingPolicy": {
                "targetTrackingScalingPolicyConfiguration": {
                    "disableScaleIn": false,
                    "scaleInCooldown": 60,
                    "scaleOutCooldown": 60,
                    "targetValue": 50.0
                }
            }
        }
    }
}
```

關閉使用表的 Amazon Keyspaces auto 擴展 AWS CLI

您可以隨時關閉表格的 Amazon Keyspaces 間 auto 擴展。如果您不再需要擴展表格的讀取或寫入容量，則應考慮關閉 auto 擴展，這樣 Amazon Keyspaces space 就不會繼續修改表格的讀取或寫入容量設定。您可以使用`UpdateTable`操作更新表格。

下列指令會關閉表格讀取容量的 auto 調整比例。它也會刪除代表您建立的 CloudWatch 警報。

```
aws keyspaces update-table --keyspace-name mykeyspace --table-name mytable
            \ --auto-scaling-specification
 readCapacityAutoScaling={autoScalingDisabled=true}
```

> **Note**
>
> 若要刪除 Application Auto Scaling 使用的服務連結角色，您必須停用帳戶中所有資料表的自動調度資源。 AWS 區域

在 Amazon Keyspaces 中有效使用爆發容量

Amazon Keyspaces 透過提供大量容量，為您的每個分區輸送量佈建提供了一些靈活性。每當您未完全使用分割區的輸送量時，Amazon Keyspaces 都會保留未使用容量的一部分，以供日後突增輸送量以處理使用量尖峰。

Amazon Keyspaces 目前最多可保留 5 分鐘 (300 秒) 的未使用讀取和寫入容量。在偶爾爆發讀取或寫入活動期間，這些額外的容量單位可以快速使用，甚至比您為表格定義的每秒佈建輸送量容量還要快。

Amazon Keyspaces 還可以消耗大量容量進行後台維護和其他任務，恕不另行通知。

請注意，高載容量的詳細資訊可能會在將來有所變更。

如何估計 Amazon Keyspaces 中的容量消耗

當您在 Amazon 金 Keyspaces 中讀取或寫入資料時，查詢耗用的讀取/寫入請求單位 (RRUS/WRUS) 或讀取/寫入容量單位 (RCU/WCU) 數量取決於 Amazon 金 Keyspaces 執行查詢所需處理的資料總量。在某些情況下，傳回給用戶端的資料可以是 Amazon Keyspace 處理查詢時必須讀取的資料子集。對於條件式寫入，即使條件式檢查失敗，Amazon Keyspaces 也會消耗寫入容量。

若要估計要求所處理的資料總量，您必須考慮資料列的編碼大小和總列數。本主題涵蓋一些常見案例和存取模式的範例，以示範 Amazon Keyspaces 如何處理查詢，以及如何影響容量消耗。您可以按照範例估算表格的容量需求，並使用 Amazon CloudWatch 觀察這些使用案例的讀取和寫入容量消耗。

有關如何計算 Amazon Keyspaces 中行的編碼大小的信息，請參閱the section called “計算列大小”。

主題

範圍查詢

要查看範圍查詢的讀取容量消耗，我們使用下面的示例表是使用按需容量模式。

```
pk1 | pk2 | pk3 | ck1 | ck2 | ck3 | value
-----+-----+-----+-----+-----+-----+-------
a | b | 1 | a | b | 50 | <any value that results in a row size larger than 4KB>
a | b | 1 | a | b | 60 | value_1
a | b | 1 | a | b | 70 | <any value that results in a row size larger than 4KB>
```

現在，在此表上運行以下查詢。

```
SELECT * FROM amazon_keyspaces.example_table_1 WHERE pk1='a' AND pk2='b' AND pk3=1 AND
 ck1='a' AND ck2='b' AND ck3 > 50 AND ck3 < 70;
```

您會從查詢收到下列結果集，Amazon Keyspaces 執行的讀取作業會在LOCAL_QUORUM一致性模式下消耗 2 個 RRU。

```
pk1 | pk2 | pk3 | ck1 | ck2 | ck3 | value
-----+-----+-----+-----+-----+-----+-------
a | b | 1 | a | b | 60 | value_1
```

Amazon Keyspaces 會消耗 2 個 RRU 來評估含有值的資料列ck3=60並處理ck3=70查詢。不過，Amazon Keyspaces 只會傳回查詢中指定WHERE條件為真的列，也就是具有值ck3=60的資料列。若要評估查詢中指定的範圍，Amazon Keyspaces 會讀取符合範圍上限的資料列，在此情況下ck3 = 70，但不會在結果中傳回該資料列。讀取容量消耗是根據處理查詢時讀取的資料，而不是以傳回的資料為基礎。

限制查詢

當處理使用該LIMIT子句的查詢時，Amazon Keyspaces 會在嘗試符合查詢中指定的條件時，讀取最大頁面大小的列。如果 Amazon Keyspaces space 找不到足夠符合第一頁LIMIT值的相符資料，則可能需要一個或多個分頁呼叫。若要繼續閱讀下一頁，您可以使用分頁權杖。預設頁面大小為 1MB。若要在使用LIMIT子句時消耗較少的讀取容量，您可以減少頁面大小。如需分頁的詳細資訊，請參閱the section called “分頁結果”。

舉個例子，讓我們來看看下面的查詢。

```
SELECT * FROM my_table WHERE partition_key=1234 LIMIT 1;”
```

如果您沒有設置頁面大小，Amazon Keyspaces 會讀取 1MB 的數據，即使它只返回 1 行給您。要只讓 Amazon Keyspaces 讀取一行，您可以將此查詢的頁面大小設置為 1。在這種情況下，如果您沒有基於 T ime-to-live 設置或客戶端時間戳的過期行，則 Amazon Keyspaces 間只會讀取一行。若要消耗較少的讀取容量，我們建議您將頁面大小設定為等於LIMIT值，以減少 Amazon Keyspaces 讀取的資料量。

表格掃描

導致完整資料表掃描的查詢 (例如使用ALLOW FILTERING選項的查詢) 是另一個查詢範例，這些查詢處理的讀取數量超過傳回的結果。讀取容量消耗是基於讀取的數據，而不是返回的數據。

對於資料表掃描範例，我們在隨選容量模式下使用下列範例表格。

```
pk | ck | value
---+----+---------
pk | 10 | <any value that results in a row size larger than 4KB>
pk | 20 | value_1
pk | 30 | <any value that results in a row size larger than 4KB>
```

Amazon Keyspaces 預設會以隨需容量模式建立一個包含四個分割區的表格。在此範例資料表中，所有資料都儲存在一個磁碟分割中，其餘的三個分割區都是空的。

現在，在表上運行以下查詢。

```
SELECT * from amazon_keyspaces.example_table_2;
```

此查詢會產生資料表掃描作業，其中 Amazon Keyspaces 會掃描資料表的所有四個分區，並在LOCAL_QUORUM一致性模式下消耗 6 個 RRU。首先，Amazon Keyspaces 消耗 3 RU 來讀取三行。pk='pk'然後，Amazon Keyspaces 會耗用額外的 3 個 RRU 來掃描表格的三個空白分割區。由於此查詢會產生資料表掃描，因此 Amazon Keyspaces 會掃描資料表中的所有分區，包括沒有資料的分割區。

輕量型交易

輕量型交易 (LWT) 可讓您針對資料表資料執行條件式寫入作業。根據評估目前狀態的條件插入、更新和刪除記錄時，條件式更新作業非常有用。

在 Amazon Keyspaces 中，所有寫入作業都需要 LOCAL_QUARE 一致性，而且使用 LWT 無須額外付費。LWT 的差異在於，當 LWT 條件檢查結果為 FALSE 時，它會消耗寫入容量單位。使用的寫入容量單位數取決於資料列的大小。如果資料列大小為 2 KB，則失敗的條件式寫入會耗用兩個寫入容量單位。如果資料列目前不存在於資料表中，則作業會耗用一個寫入容量單位。透過監視中的ConditionalCheckFailed指標， CloudWatch 您可以判斷 LWT 條件檢查失敗所耗用的容量。

使用 Amazon 估算讀取和寫入容量消耗 CloudWatch

若要估算和監視讀取和寫入容量耗用量，您可以使用 CloudWatch 儀表板。如需 Amazon Keyspaces 可用指標的詳細資訊，請參閱the section called “指標與維度”。

若要監視特定陳述式使用的讀取和寫入容量單位 CloudWatch，您可以遵循下列步驟。

1. 使用範例資料建立新資料表

2. 設定表格的 Amazon Keyspaces CloudWatch 儀表板。若要開始使用，您可以使用 Github 上提供的儀表板範本。
3. 例如，使用`ALLOW FILTERING`選項執行 CQL 陳述式，並在儀表板中檢查用於完整表格掃描的讀取容量單位。

在 Amazon Keyspaces 中使用密鑰空間，表和行（對於 Apache 卡桑德拉）

本章介紹了有關使用 Keyspaces，表，行，以及更多在 Amazon 密鑰空間（Apache 卡桑德拉）工作的詳細信息。要了解如何使用 Amazon 監控密鑰空間和表格 CloudWatch，請參閱the section called “使用監控 CloudWatch”。

主題

- 使用 Amazon Keyspaces 中的密鑰空間
- 使用 Amazon Keyspaces 中的表
- 使用 Amazon Keyspaces 中的行
- 使用 Amazon Keyspaces 中的查詢
- 使用亞馬遜 Keyspaces 中的分區程序
- 使用 Amazon Keyspaces 資源的標籤和標籤

使用 Amazon Keyspaces 中的密鑰空間

本節提供了有關使用 Amazon Keyspaces（對於 Apache 卡桑德拉）密鑰空間的詳細信息。

主題

- 使用 Amazon Keyspaces 中的系統密鑰空間
- 在 Amazon Keyspaces 中創建密鑰空間

使用 Amazon Keyspaces 中的系統密鑰空間

Amazon Keyspaces 使用四個系統密鑰空間：

- `system`
- `system_schema`
- `system_schema_mcs`
- `system_multiregion_info`

以下各節提供 Amazon 金鑰空間支援的系統金鑰空間和系統表格的詳細資訊。

system

這是一個卡桑德拉密鑰空間。Amazon Keyspaces 使用以下表格。

表格名稱	欄位名稱	說明
local	key, bootstrapped, broadcast_address, cluster_name, cql_version, data_center, gossip_generation, host_id, listen_address, native_protocol_version, partitioner, rack, release_version, rpc_address, schema_version, thrift_version, tokens, truncated_at	有關本地密鑰空間的信息。
peers	peer, data_center, host_id, preferred_ip, rack, release_version, rpc_address, schema_version, tokens	查詢此表格以查看可用的端點。例如，如果您透過公用端點進行連線，您會看到九個可用 IP 位址的清單。如果您透過 FIPS 端點連線，您會看到三個 IP 位址的清單。如果您透過 AWS PrivateLink VPC 端點進行連線，您會看到已設定的 IP 位址清單。如需詳細資訊，請參閱 the section called “使用介面 VPC 端點資訊填入system.peers 表格項目”。

表格名稱	欄位名稱	說明
size_estimates	keyspace_name, table_name, range_start, range_end, mean_partition_size, partitions_count	此表定義了每個表中每個 Token 範圍的分區總大小和數量。這是需要的 Apache 卡桑德拉星火連接器，它使用估計的分區大小來分發工作。
prepared_statements	prepared_id, logged_keyspace, query_string	此表格包含有關已存查詢的資訊。

system_schema

這是一個卡桑德拉密鑰空間。Amazon Keyspaces 使用以下表格。

表格名稱	欄位名稱	說明
keyspaces	keyspace_name, durable_writes, replication	有關特定密鑰空間的信息。
tables	keyspace_name, table_name, bloom_filter_fp_chance, caching, comment, compaction, compression, crc_check_chance, dclocal_read_repair_chance, default_time_to_live, extensions, flags, gc_grace_seconds, id, max_index_interval, memtable_flush_period_in_ms, min_index	有關特定表格的資訊。

表格名稱	欄位名稱	說明
	_interval, read_repair_chance, speculative_retry	
columns	keyspace_name, table_name, column_name, clustering_order, column_name_bytes, kind, position, type	有關特定列的信息。

system_schema_mcs

這是一個 Amazon Keyspaces 間，用於存儲有關 AWS 或 Amazon Keyspaces 間特定設置的信息。

表格名稱	欄位名稱	說明
keyspaces	keyspace_name, durable_writes, replication	查詢此表以編程方式查找是否已創建密鑰空間。如需詳細資訊，請參閱 the section called “創建密鑰空間”。
tables	keyspace_name, creation_time, speculative_retry, cdc, gc_grace_seconds, crc_check_chance, min_index_interval, bloom_filter_fp_chance, flags, custom_properties, dclocal_read_repair_chance, table_name, caching, default_time_to_li	查詢此表格以找出特定表格的狀態。如需詳細資訊，請參閱 the section called “建立資料表”。 您也可以查詢此表格以列出 Amazon Keyspaces 特有且儲存為custom_properties 的設定。例如： • capacity_mode • client_side_timestamps

表格名稱	欄位名稱	說明
	ve, read_repair_chance, max_index_interval, extensions, compaction, comment, id, compression, memtable_flush_period_in_ms, status	• encryption_specification • point_in_time_recovery • ttl
tables_history	keyspace_name, table_name, event_time, creation_time, custom_properties, event	查詢此表格以瞭解特定資料表的結構定義變更。
columns	keyspace_name, table_name, column_name, clustering_order, column_name_bytes, kind, position, type	這個表是相同的密鑰空間中的卡桑德拉表system_schema 。
tags	resource_id, keyspace_name, resource_name, resource_type, tags	查詢此表以查找密鑰空間是否具有標籤。如需詳細資訊，請參閱 the section called “使用 CQL 將標籤新增標籤至新的或現有金鑰空間和資料表”。

表格名稱	欄位名稱	說明
autoscaling	keyspace_name, table_name, provisioned_read_capacity_autoscaling_update, provisioned_write_capacity_autoscaling_update	查詢此表格以取得已佈建表格的 auto 調整設定。請注意，在表格啟用之前，這些設定才能使用。要查詢此表，您必須在WHERE子句table_name 中指定keyspace_name 和。如需詳細資訊，請參閱 the section called “使用定制列表”。

system_multiregion_info

這是儲存多區域複寫相關資訊的 Amazon 金鑰空間。

表格名稱	欄位名稱	說明
tables	keyspace_name, table_name, region, status	此表格包含多區域表格的相關資訊，例如複製表格的資訊以及表格的狀態。 AWS 區域 您也可以查詢此表格以列出儲存為custom_properties 的 Amazon Keyspaces 特有的設定。例如： • capacity_mode 要查詢此表，您必須在WHERE子句table_name 中指定keyspace_name 和。如需詳細資訊，請參閱 the section called “建立多區域金鑰空間 (CQL)”。
autoscaling	keyspace_name, table_name, provision	查詢此表格以取得多區域佈建表格的 auto 調整設

表格名稱	欄位名稱	說明
	ed_read_capacity_autoscaling_update, provisioned_write_capacity_autoscaling_update, region	定。請注意，在表格啟用之前，這些設定才能使用。要查詢此表，您必須在WHERE子句table_name 中指定keyspace_name 和。如需詳細資訊，請參閱 the section called “使用定制列表”。

在 Amazon Keyspaces 中創建密鑰空間

Amazon Keyspaces 會以非同步方式執行資料定義語言 (DDL) 作業，例如建立和刪除金鑰空間。

您可以在中監視新密鑰空間的建立狀態 AWS Management Console，這表示金鑰空間何時處於擱置狀態或作用中。您還可以通過使用密鑰空間以編程方式監視新密鑰空間的創建狀態。system_schema_mcs當它準備好使用時，密鑰空間會在system_schema_mcskeyspaces表中可見。

建議檢查新密鑰空間何時可以使用的設計模式是輪詢 Amazon Keyspaces 間system_schema_mcskeyspaces表（system_schema_mcs.* ）。如需索引鍵空間的 DDL 陳述式清單，請參閱 CQL 語言參考中的the section called “Keyspaces”章節。

下面的查詢顯示密鑰空間是否已成功創建。

```
SELECT * FROM system_schema_mcs.keyspaces WHERE keyspace_name = 'mykeyspace';
```

對於已成功創建的密鑰空間，查詢的輸出如下所示。

```
keyspace_name | durable_writes  | replication
--------------+-----------------+--------------
   mykeyspace | true            |{...} 1 item
```

使用 Amazon Keyspaces 中的表

本節提供了有關在 Amazon Keyspaces 間表（Apache 卡桑德拉）工作的詳細信息。

主題

在 Amazon Keyspaces 中創建表

Amazon Keyspaces 會以非同步方式執行資料定義語言 (DDL) 作業，例如建立和刪除資料表。您可以在中監視新表格的建立狀態 AWS Management Console，這表示資料表何時處於閒置狀態或作用中狀態。您也可以使用系統結構描述表格，以程式設計方式監視新資料表的建立狀態。

準備就緒時，表格會在系統結構描述中顯示為作用中狀態。建議檢查新資料表何時可供使用的設計模式是輪詢 Amazon Keyspaces 系統架構表 (`system_schema_mcs.*`)。如需資料表的 DDL 陳述式清單，請參閱 CQL 語言參考中的the section called “資料表”章節。

下面的查詢顯示了一個表的狀態。

```
SELECT keyspace_name, table_name, status FROM system_schema_mcs.tables WHERE
 keyspace_name = 'mykeyspace' AND table_name = 'mytable';
```

對於仍在建立且閒置中的資料表，查詢的輸出如下所示。

```
keyspace_name | table_name | status
--------------+------------+--------
   mykeyspace |    mytable | CREATING
```

對於已成功建立且處於作用中狀態的資料表，查詢的輸出如下所示。

```
keyspace_name | table_name | status
--------------+------------+--------
   mykeyspace |    mytable | ACTIVE
```

使用 Amazon Keyspaces 中的多區域表

多區域表必須以下列其中一種方式設定寫入輸送量容量：

- 隨需容量模式，以寫入請求單位 (WRU) 測量
- 具有 auto 調整的佈建容量模式，以寫入容量單位 (WCU) 測量

您可以使用具有 auto 擴展或隨需容量模式的佈建容量模式，以協助確保多區域表格具有足夠的容量，可以對所有 AWS 區域人執行複寫寫入。

> **Note**
>
> 在其中一個區域中變更表格的容量模式會變更所有複本的容量模式。

根據預設，Amazon Keyspaces 會針對多區域資料表使用隨選模式。使用隨選模式時，您不需要指定預期應用程式執行的讀取和寫入輸送量。Amazon Keyspaces 可在您的工作負載上升或下降到任何先前達到的流量層級時立即容納工作負載。如果工作負載的流量層級達到新的高峰，Amazon Keyspaces 快速調整以適應工作負載。

如果您為表格選擇佈建的容量模式，則必須設定應用程式每秒所需的讀取容量單位 (RCU) 和寫入容量單位 (WCU) 數量。

若要規劃多區域表格的輸送容量需求，您應該先預估每個區域每秒所需的 WCU 數量。然後，您可以從表格複寫的所有區域新增寫入，並使用總和為每個區域佈建容量。這是必要的，因為在一個區域中執行的每個寫入也必須在每個複本區域中重複。

如果資料表沒有足夠的容量來處理來自所有區域的寫入，就會發生容量例外狀況。此外，區域間複寫的等待時間也會增加。

例如，如果您的多區域表格預計在美國東部 (維吉尼亞北部) 每秒寫入 5 次，美國東部 (俄亥俄州) 每秒寫入 10 次，在歐洲 (愛爾蘭) 每秒寫入 5 次，您應該預期該表在每個區域消耗 20 個 WCU：美國東部 (維吉尼亞北部)、美國東部 (俄亥俄) 和歐洲 (愛爾蘭)。這表示在此範例中，您必須為每個資料表的複本佈建 20 個 WCU。您可以使用 Amazon 監控表格的容量消耗 CloudWatch。如需詳細資訊，請參閱 the section called “使用監控 CloudWatch”。

由於每個多區域寫入的費用都是以 1.25 倍的 WCU 計費，因此在此範例中，您總共會看到 75 個計費的 WCU。有關定價的更多信息，請參閱 Amazon Keyspaces（阿帕奇卡桑德拉）定價。

如需使用 Amazon Keyspaces auto 擴展佈建容量的詳細資訊，請參閱the section called “透過 auto 擴充管理輸送量容量”。

> **Note**
>
> 如果表格在具有 auto 擴展的佈建容量模式下執行，則允許佈建的寫入容量在每個區域的這些 auto 擴展設定內浮動。

Amazon Keyspaces 中的靜態列

在具有叢集資料行的 Amazon Keyspaces 表格中，您可以使用STATIC關鍵字建立靜態資料行。存儲在靜態列中的值在邏輯分區中的所有行之間共享。當您更新此資料行的值時，Amazon Keyspaces 會自動將變更套用至分割區中的所有資料列。

本節說明如何在寫入靜態資料行時計算資料的編碼大小。這個程序會與將資料寫入資料列的非靜態資料行的程序分開處理。除了靜態資料的大小配額之外，靜態資料行的讀取和寫入作業也會獨立影響資料表的計量和輸送量容量。

計算 Amazon Keyspaces 中每個邏輯分區的靜態列大小

本節提供有關如何估計 Amazon Keyspaces 中靜態列的編碼大小的詳細信息。當您計算帳單和配額使用時，會使用編碼大小。當您計算表格的佈建輸送量容量需求時，您也應該使用編碼大小。若要計算 Amazon Keyspaces 中靜態資料行的編碼大小，您可以使用下列準則。

- 分割區索引鍵最多可包含 2048 個位元組的資料。分割區索引鍵中的每個索引鍵資料行最多需要 3 個位元組的中繼資料。這些中繼資料位元組會計入每個磁碟分割 1 MB 的靜態資料大小配額。計算靜態資料的大小時，您應該假設每個分區索引鍵資料行都使用完整的 3 個位元組的中繼資料。
- 根據資料類型使用靜態資料行資料值的原始大小。如需資料類型的詳細資訊，請參閱 the section called “資料類型”。
- 將 104 個位元組新增至中繼資料的靜態資料大小。
- 叢集資料行和一般的非主索引鍵資料行不會計入靜態資料的大小。若要瞭解如何估計資料列中非靜態資料的大小，請參閱the section called “計算列大小”。

靜態資料行的總編碼大小以下列公式為基礎：

```
partition key columns + static columns + metadata = total encoded size of static data
```

考慮一個表的下面的例子，其中所有列都是整數類型。此資料表有兩個分割索引鍵資料行、兩個叢集資料行、一個一般資料行，以及一個靜態資料欄。

```
CREATE TABLE mykeyspace.mytable(pk_col1 int, pk_col2 int, ck_col1 int, ck_col2
 int, reg_col1 int, static_col1 int static, primary key((pk_col1, pk_col2),ck_col1,
 ck_col2));
```

在這個例子中，我們計算以下語句的靜態數據的大小：

```
INSERT INTO mykeyspace.mytable (pk_col1, pk_col2, static_col1) values(1,2,6);
```

若要估計此寫入作業所需的位元組總數，您可以使用下列步驟。

1. 新增儲存在資料行中之資料類型的位元組和中繼資料位元組，計算分割索引鍵資料行的大小。對所有分割區索引鍵欄重複此步驟。

   a. 計算分區鍵（pk_col1）的第一列的大小：

   ```
   4 bytes for the integer data type + 3 bytes for partition key metadata = 7
    bytes
   ```

   b. 計算分區鍵（pk_col2）的第二列的大小：

   ```
   4 bytes for the integer data type + 3 bytes for partition key metadata = 7
    bytes
   ```

   c. 新增兩個資料欄以取得分割區索引鍵資料欄的估計總大小：

   ```
   7 bytes + 7 bytes = 14 bytes for the partition key columns
   ```

2. 添加靜態列的大小。在這個例子中，我們只有一個靜態列存儲一個整數（這需要 4 個字節）。
3. 最後，若要取得靜態資料行資料的總編碼大小，請將主索引鍵資料行和靜態資料行的位元組加起來，並為中繼資料新增其他 104 個位元組：

   ```
   14 bytes for the partition key columns + 4 bytes for the static column + 104 bytes
    for metadata = 122 bytes.
   ```

您也可以使用相同的陳述式更新靜態和非靜態資料。若要估計寫入作業的總大小，您必須先計算非靜態資料更新的大小。然後計算行更新的大小，如示例中所示the section called “計算列大小”，並添加結果。

在這種情況下，您總共可以寫入 2 MB — 1 MB 是最大資料列大小配額，而 1 MB 是每個邏輯磁碟分割的最大靜態資料大小的配額。

要計算相同語句中靜態和非靜態數據更新的總大小，可以使用以下公式：

```
(partition key columns + static columns + metadata = total encoded size of static data)
 + (partition key columns + clustering columns + regular columns + row metadata = total
 encoded size of row)
= total encoded size of data written
```

考慮一個表的下面的例子，其中所有列都是整數類型。此資料表有兩個分割索引鍵資料行、兩個叢集資料行、一個一般資料行，以及一個靜態資料欄。

```
CREATE TABLE mykeyspace.mytable(pk_col1 int, pk_col2 int, ck_col1 int, ck_col2
 int, reg_col1 int, static_col1 int static, primary key((pk_col1, pk_col2),ck_col1,
 ck_col2));
```

在這個例子中，我們計算當我們寫一行到表中的數據的大小，如下面的語句所示：

```
INSERT INTO mykeyspace.mytable (pk_col1, pk_col2, ck_col1, ck_col2, reg_col1,
 static_col1) values(2,3,4,5,6,7);
```

若要估計此寫入作業所需的位元組總數，您可以使用下列步驟。

1. 如前所示計算靜態資料的總編碼大小。在這個例子中，它是 122 個字節。
2. 依照上述的步驟，根據非靜態資料的更新，新增列的編碼總大小的大the section called “計算列大小”小。在此範例中，資料列更新的總大小為 134 個位元組。

   ```
   122 bytes for static data + 134 bytes for nonstatic data = 256 bytes.
   ```

計量Amazon 密Keyspaces 間中的靜態數據的讀取/寫入操作

靜態數據與卡桑德拉邏輯分區，而不是單個行相關聯。Amazon Keyspaces 中的邏輯分割區可跨越多個實體儲存分割區，實際上取消繫結大小。因此，Amazon Keyspaces 計量器會分別寫入靜態和非靜態資料上的作業。此外，包含靜態和非靜態資料的寫入需要額外的基礎作業，才能提供資料一致性。

如果您對靜態和非靜態資料執行混合寫入作業，則會產生兩個不同的寫入作業 — 一個用於非靜態資料，一個用於靜態資料。這適用於隨需和佈建的讀取/寫入容量模式。

下列範例提供詳細資料，說明當您計算 Amazon Keyspaces 中具有靜態資料行之表格的佈建輸送量容量需求時，如何估計所需的讀取容量單位 (RCU) 和寫入容量單位 (WCU)。您可以使用下列公式，估計資料表需要多少容量來處理包含靜態和非靜態資料的寫入：

```
2 x WCUs required for nonstatic data + 2 x WCUs required for static data
```

例如，如果您的應用程式每秒寫入 27 KB 的資料，而且每次寫入包含 25.5 KB 的非靜態資料和 1.5 KB 的靜態資料，則您的資料表需要 56 個 WCU (2 x 26 個 WCU + 2 x 2 WCU)。

Amazon Keyspaces 會測量靜態和非靜態資料的讀取與讀取多個資料列相同。因此，在相同作業中讀取靜態和非靜態資料的價格是以執行讀取所處理之資料的彙總大小為基礎。

要了解如何使用 Amazon 監控無伺服器資源 CloudWatch，請參閱the section called “使用監控 CloudWatch”。

使用 Amazon Keyspaces 中的行

本節提供了有關使用 Amazon Keyspaces 間行（對於 Apache 卡桑德拉）的詳細信息。表是在 Amazon Keyspaces 的主數據結構和表中的數據被組織成列和行。

主題

計算 Amazon Keyspaces 中的行大小

Amazon Keyspaces 提供全受管儲存，可提供 10 毫秒的讀取和寫入效能，並在多個可用區域中持久存放資料。AWSAmazon Keyspaces 中繼資料附加到所有列和主索引鍵欄，以支援有效率的資料存取和高可用性。

本節提供有關如何估計 Amazon Keyspaces 中行的編碼大小的詳細信息。計算帳單和配額使用時，會使用編碼的資料列大小。在計算表格的佈建輸送量容量需求時，您也應該使用編碼的資料列大小。要計算 Amazon Keyspaces 中行的編碼大小，您可以使用以下準則。

- 對於不是主索引鍵、叢集資料行或STATIC資料行的一般資料行，請根據資料類型使用儲存格資料的原始大小，並新增必要的中繼資料。如需 Amazon Keyspaces 支援之資料類型的詳細資訊，請參閱the section called “資料類型”。下面列出了 Amazon 密 Keyspaces 存放資料類型值和中繼資料的一些主要差異。

- 每個欄名稱所需的空間會使用欄識別碼儲存，並新增至儲存在欄中的每個資料值。資料行識別碼的儲存值取決於資料表中的整體資料欄數：
  - 1 至 62 個資料欄：1 個位元組
  - 6 至 124 個資料欄：2 個位元組
  - 1 至 6 個資料欄：3 個位元組

  對於每個額外 62 列添加 1 個字節。請注意，在 Amazon Keyspaces 中，最多可以使用單個`INSERT`或`UPDATE`語句修改 225 個常規列。如需詳細資訊，請參閱the section called “Amazon Keyspaces 服務配額”。
- 分割區索引鍵最多可包含 2048 個位元組的資料。分割區索引鍵中的每個索引鍵資料行最多需要 3 個位元組的中繼資料。計算資料列的大小時，您應該假設每個分割區索引鍵資料行都使用完整的 3 個位元組的中繼資料。
- 叢集資料行最多可儲存 850 個位元組的資料。除了資料值的大小之外，每個叢集資料行最多需要中繼資料資料值大小的 20%。計算行的大小時，您應該為每 5 個字節的聚類列數據值添加 1 個字節的元數據。
- Amazon 密 Keyspaces 會儲存每個分區金鑰和叢集索引鍵資料行兩次的資料值。額外的開銷用於高效查詢和內置索引。
- 卡桑德拉 `ASCIITEXT`，和`VARCHAR`字符串數據類型都存儲在 Amazon Keyspaces 使用 Unicode 與 UTF-8 二進制編碼。Amazon Keyspaces 中的字符串大小等於 UTF-8 編碼字節的數量。
- 卡桑德拉`INT`，`BIGINT`，`SMALLINT`，和`TINYINT`數據類型存儲在 Amazon Keyspaces 作為具有可變長度的數據值，具有多達 38 個有效數字。前後的零會截去。任何這些數據類型的大小約為每兩個有效數字 1 個字節 + 1 字節。
- Amazon Keyspaces `BLOB` 中的 A 與值的原始字節長度存儲。
- `Null`值或值的大小為 1 個`Boolean`位元組。
- 儲存集合資料類型 (例如`LIST`或`MAP`需要 3 個位元組的中繼資料) 的資料行，不論其內容為何。a `LIST` 或的大小`MAP`為（列 ID）+ 總和（嵌套元素的大小）+（3 個字節）。空白`LIST`或大小`MAP`為 (資料行識別碼) + (3 個位元組)。每個單獨`LIST`或`MAP`元素還需要 1 個字節的元數據。
- `STATIC`欄資料不會計入 1 MB 的最大列大小。若要計算靜態欄的資料大小，請參閱the section called “計算每個邏輯分區的靜態列大小”。
- 當功能開啟時，會儲存每一列中每一欄的用戶端時間戳記。這些時間戳記大約會佔用 20-40 個位元組 (視您的資料而定)，而且會造成資料列的儲存和輸送量成本。如需詳細資訊，請參閱the section called “亞馬遜 Keyspaces 間中的客戶端時間戳”。
- 將 100 個字節添加到行元數據的每行的大小。

已編碼資料列的總大小以下列公式為基礎：

```
partition key columns + clustering columns + regular columns + row metadata = total
 encoded size of row
```

> **Important**
>
> 所有資料行中繼資料，例如資料行 ID、分割索引鍵中繼資料、叢集資料行中繼資料，以及用戶端時間戳記和資料列中繼資料都會計入最大資料列大小 1 MB。

考慮一個表的下面的例子，其中所有列都是整數類型。此資料表有兩個分割索引鍵資料行、兩個叢集資料行，以及一個一般資料欄。由於此資料表有五個資料行，所以資料行名稱識別碼所需的空間為 1 個位元組。

```
CREATE TABLE mykeyspace.mytable(pk_col1 int, pk_col2 int, ck_col1 int, ck_col2 int,
 reg_col1 int, primary key((pk_col1, pk_col2),ck_col1, ck_col2));
```

在這個例子中，我們計算數據的大小，當我們寫一行到表中顯示在下面的語句：

```
INSERT INTO mykeyspace.mytable (pk_col1, pk_col2, ck_col1, ck_col2, reg_col1)
 values(1,2,3,4,5);
```

若要估計此寫入作業所需的位元組總數，您可以使用下列步驟。

1. 新增儲存在資料行中之資料類型的位元組和中繼資料位元組，計算分割索引鍵資料行的大小。對所有分割區索引鍵欄重複此步驟。

   a. 計算分區鍵（pk_col1）的第一列的大小：

   ```
   (2 bytes for the integer data type) x 2 + 1 byte for the column id + 3 bytes
    for partition key metadata = 8 bytes
   ```

   b. 計算分區鍵（pk_col2）的第二列的大小：

   ```
   (2 bytes for the integer data type) x 2 + 1 byte for the column id + 3 bytes
    for partition key metadata = 8 bytes
   ```

   c. 新增兩個資料欄以取得分割區索引鍵資料欄的估計總大小：

```
8 bytes + 8 bytes = 16 bytes for the partition key columns
```

2. 透過新增儲存在資料行中之資料類型的位元組和中繼資料位元組，計算叢集資料行的大小。對所有叢集資料行重複此步驟。

   a. 計算聚類列（ck_col1）的第一列的大小：

   ```
   (2 bytes for the integer data type) x 2 + 20% of the data value (2 bytes) for
    clustering column metadata + 1 byte for the column id  = 6 bytes
   ```

   b. 計算聚類列（ck_col2）的第二列的大小：

   ```
   (2 bytes for the integer data type) x 2 + 20% of the data value (2 bytes) for
    clustering column metadata + 1 byte for the column id = 6 bytes
   ```

   c. 新增兩個資料行以取得叢集資料行的總估計大小：

   ```
   6 bytes + 6 bytes = 12 bytes for the clustering columns
   ```

3. 添加常規列的大小。在這個例子中，我們只有一個存儲一個數字整數的列，這需要 2 個字節，列 ID 為 1 個字節。
4. 最後，要獲取總編碼的行大小，請將所有列的字節相加，並為行元數據添加額外的 100 個字節：

   ```
   16 bytes for the partition key columns + 12 bytes for clustering columns + 3 bytes
    for the regular column + 100 bytes for row metadata = 131 bytes.
   ```

要了解如何使用 Amazon 監控無伺服器資源 CloudWatch，請參閱the section called “使用監控 CloudWatch”。

使用 Amazon Keyspaces 中的查詢

本節介紹了在 Amazon Keyspaces 間查詢工作（對於 Apache 卡桑德拉）。可用於查詢、轉換和管理資料的 CQL 陳述式為SELECTINSERTUPDATE、和DELETE。下列主題概述了處理查詢時可用的一些較複雜的選項。如需具有範例的完整語言語法，請參閱the section called “DML 陳述式”。

主題

- Amazon Keyspaces 中的分頁結果

使用IN運營商與 Amazon Keyspaces 的SELECT聲明

在中選擇

您可以使用陳述式查詢資料表中的資料，該SELECT陳述式會針對資料表中的一或多個資料列讀取一或多個資料行，並傳回包含與要求相符之資料列的結果集。SELECT陳述式包含決select_clause定要讀取哪些資料行，並在結果集中傳回。該子句可以包含在返回數據之前轉換數據的指令。可選WHERE子句指定哪些行必須查詢，並且由屬於主鍵一部分的列上的關係組成。Amazon Keyspaces 支持WHERE子句中的IN關鍵字。本節使用範例顯示 Amazon Keyspaces 如何使用IN關鍵字處理SELECT陳述式。

這個例子演示了 Amazon Keyspaces 如何將帶有IN關鍵字的SELECT語句分解為子查詢。在這個例子中，我們使用的名稱表my_keyspace.customers。此資料表有一個主索引鍵資料行department_id、兩個叢集資料行sales_region_id和sales_representative_id，以及一個資料行，其中包含資customer_name料行中的客戶名稱。

```
SELECT * FROM my_keyspace.customers;

 department_id | sales_region_id | sales_representative_id | customer_name
---------------+-----------------+-------------------------+--------------
  0            |        0        |            0            |    a
  0            |        0        |            1            |    b
  0            |        1        |            0            |    c
  0            |        1        |            1            |    d
  1            |        0        |            0            |    e
  1            |        0        |            1            |    f
  1            |        1        |            0            |    g
  1            |        1        |            1            |    h
```

使用此表格，您可以執行下列SELECT陳述式，以尋找您對WHERE條款中IN關鍵字感興趣之部門與銷售區域中的客戶。下面的語句是這樣的一個例子。

```
SELECT * FROM my_keyspace.customers WHERE department_id IN (0, 1) AND sales_region_id
 IN (0, 1);
```

Amazon Keyspaces 將此語句分成四個子查詢，如下面的輸出。

```
SELECT * FROM my_keyspace.customers WHERE department_id = 0 AND sales_region_id = 0;
```

```
 department_id | sales_region_id | sales_representative_id | customer_name
---------------+-----------------+-------------------------+--------------
  0            |        0        |           0             |    a
  0            |        0        |           1             |    b

SELECT * FROM my_keyspace.customers WHERE department_id = 0 AND sales_region_id = 1;

 department_id | sales_region_id | sales_representative_id | customer_name
---------------+-----------------+-------------------------+--------------
  0            |        1        |           0             |    c
  0            |        1        |           1             |    d

SELECT * FROM my_keyspace.customers WHERE department_id = 1 AND sales_region_id = 0;

 department_id | sales_region_id | sales_representative_id | customer_name
---------------+-----------------+-------------------------+--------------
  1            |        0        |           0             |    e
  1            |        0        |           1             |    f

SELECT * FROM my_keyspace.customers WHERE department_id = 1 AND sales_region_id = 1;

 department_id | sales_region_id | sales_representative_id | customer_name
---------------+-----------------+-------------------------+--------------
  1            |        1        |           0             |    g
  1            |        1        |           1             |    h
```

使用IN關鍵字時，Amazon Keyspaces 會在下列任何一種情況下自動分頁結果：

- 處理每 10 個子查詢之後。
- 在處理 1MB 的邏輯 IO 之後。
- 如果您設定了PAGE SIZE，Amazon Keyspaces 會在根據集合讀取要處理的查詢數目之後進行分頁。PAGE SIZE
- 當您使用LIMIT關鍵字減少傳回的列數時，Amazon Keyspaces space 會在根據集合讀取要處理的查詢數目之後進行分頁。LIMIT

下表是用一個例子來說明這一點。

如需分頁的詳細資訊，請參閱the section called “分頁結果”。

```
SELECT * FROM my_keyspace.customers;
```

```
 department_id | sales_region_id | sales_representative_id | customer_name
---------------+-----------------+-------------------------+--------------
  2            |        0        |          0              |    g
  2            |        1        |          1              |    h
  2            |        2        |          2              |    i
  0            |        0        |          0              |    a
  0            |        1        |          1              |    b
  0            |        2        |          2              |    c
  1            |        0        |          0              |    d
  1            |        1        |          1              |    e
  1            |        2        |          2              |    f
  3            |        0        |          0              |    j
  3            |        1        |          1              |    k
  3            |        2        |          2              |    l
```

您可以在此表中運行下面的語句，看看分頁是如何工作的。

```
SELECT * FROM my_keyspace.customers WHERE department_id IN (0, 1, 2, 3) AND
 sales_region_id IN (0, 1, 2) AND sales_representative_id IN (0, 1);
```

Amazon Keyspaces 將此陳述式當做 24 個子查詢處理，因為此查詢中包含的所有IN術語的笛卡爾乘積的基數為 24。

```
 department_id | sales_region_id | sales_representative_id | customer_name
---------------+-----------------+-------------------------+--------------
  0            |        0        |          0              |    a
  0            |        1        |          1              |    b
  1            |        0        |          0              |    d
  1            |        1        |          1              |    e

---MORE---
 department_id | sales_region_id | sales_representative_id | customer_name
---------------+-----------------+-------------------------+--------------
  2            |        0        |          0              |    g
  2            |        1        |          1              |    h
  3            |        0        |          0              |    j

---MORE---
 department_id | sales_region_id | sales_representative_id | customer_name
---------------+-----------------+-------------------------+--------------
  3            |        1        |          1              |    k
```

這個範例說明如何在含有IN關鍵字的SELECT陳述式中使用ORDER BY子句。

```
SELECT * FROM my_keyspace.customers WHERE department_id IN (3, 2, 1) ORDER BY
 sales_region_id DESC;

         department_id | sales_region_id | sales_representative_id | customer_name
        ---------------+-----------------+-------------------------+--------------
          3            |        2        |          2              |    l
          3            |        1        |          1              |    k
          3            |        0        |          0              |    j
          2            |        2        |          2              |    i
          2            |        1        |          1              |    h
          2            |        0        |          0              |    g
          1            |        2        |          2              |    f
          1            |        1        |          1              |    e
          1            |        0        |          0              |    d
```

子查詢會依據分割索引鍵和叢集索引鍵資料行在查詢中顯示的順序來處理。在以下範例中，會先處理分割索引鍵值「2」的子查詢，然後是分割區索引鍵值「3」和「1」的子查詢。給定子查詢的結果是根據查詢的排序子句（如果存在）或表創建期間定義的表的聚類順序進行排序。

```
SELECT * FROM my_keyspace.customers WHERE department_id IN (2, 3, 1) ORDER BY
 sales_region_id DESC;

         department_id | sales_region_id | sales_representative_id | customer_name
        ---------------+-----------------+-------------------------+--------------
          2            |        2        |          2              |    i
          2            |        1        |          1              |    h
          2            |        0        |          0              |    g
          3            |        2        |          2              |    l
          3            |        1        |          1              |    k
          3            |        0        |          0              |    j
          1            |        2        |          2              |    f
          1            |        1        |          1              |    e
          1            |        0        |          0              |    d
```

在 Amazon Keyspaces 中訂購結果

子ORDER BY句會指定SELECT陳述式中傳回之結果的排序順序。該語句將列名列表作為參數，並且您可以為每個列指定數據的排序順序。您只能在排序子句中指定叢集資料行，不允許使用非叢集資料行。

傳回結果的兩個可用排序順序選項分別ASC為遞增和DESC遞減排序順序。

```
SELECT * FROM my_keyspace.my_table ORDER BY (col1 ASC, col2 DESC, col3 ASC);

 col1 | col2 | col3
------+------+------
  0   |  6   |  a
  1   |  5   |  b
  2   |  4   |  c
  3   |  3   |  d
  4   |  2   |  e
  5   |  1   |  f
  6   |  0   |  g
```

```
SELECT * FROM my_keyspace.my_table ORDER BY (col1 DESC, col2 ASC, col3 DESC);

 col1 | col2 | col3
------+------+------
  6   |  0   |  g
  5   |  1   |  f
  4   |  2   |  e
  3   |  3   |  d
  2   |  4   |  c
  1   |  5   |  b
  0   |  6   |  a
```

如果您沒有在查詢陳述式中指定排序順序，則會使用叢集資料行的預設順序。

您可以在排序子句中使用的可能排序順序取決於在建立表格時指派給每個叢集資料行的排序順序。查詢結果只能按照在建立資料表時為所有叢集資料行定義的順序排序，或是定義的排序順序相反。不允許使用其他可能的組合。

例如，如果表的CLUSTERING ORDER是（COL1 ASC，COL2 描述，COL3 ASC），則的有效參數是（COL1 ASC，COL2 描述，COL3 ASC）或（COL1 描述，COL2 ASC，COL3 描述）。ORDER BY如需詳細資訊CLUSTERING ORDER，請參閱〈〉中的table_options〈〉the section called “CREATE TABLE”。

Amazon Keyspaces 中的分頁結果

當讀取以處理陳述式的資料超過 1 MB 時，Amazon Keyspaces 會自動分頁SELECT陳述式的SELECT結果。使用分頁時，SELECT陳述式結果會分成大小為 1 MB (或更小) 的資料「頁面」。應用程式可以處理結果的第一頁、第二頁，以此類推。在處理返回多行的SELECT查詢時，客戶端應始終檢查分頁令牌。

如果用戶端提供PAGE SIZE的需要讀取超過 1 MB 的資料，Amazon Keyspaces 會根據 1 MB 的資料讀取增量自動將結果分解為多個頁面。

例如，如果某個資料列的平均大小為 100 KB，而您指定PAGE SIZE的 A 值為 20，則 Amazon Keyspaces 會在讀取 10 個資料列 (讀取資料的 1000 KB) 之後自動分頁資料。

由於 Amazon Keyspaces 會根據其讀取來處理請求的資料列數，而不是結果集中傳回的資料列數目來分頁結果，因此如果您執行篩選的查詢，某些頁面可能不會包含任何資料列。

例如，如果您設定PAGE SIZE為 10，而 Keyspaces 會評估 30 列來處理您的SELECT查詢，Amazon Keyspaces 將會傳回三個頁面。如果只有一個資料列子集符合您的查詢，部分頁面的資料列可能少於 10 列。如需LIMIT查詢如何影響讀取容量PAGE SIZE的範例，請參閱the section called “限制查詢”。

使用亞馬遜 Keyspaces 中的分區程序

在 Apache 卡桑德拉，分區控制哪些節點數據存儲在集群中。磁碟分割程式會使用磁碟分割索引鍵的雜湊值來建立數值 Token。卡桑德拉使用此令牌跨節點分發數據。客戶端還可以在SELECT操作和WHERE子句中使用這些令牌來優化讀取和寫入操作。例如，用戶端可以指定要在每個 parallel 工作中查詢的不同 Token 範圍，有效率地在大型資料表上執行 parallel 查詢。

亞馬遜 Keyspaces 提供了三種不同的分區程序。

雜音 3 分區程序（默認）

阿帕奇卡桑德拉兼Murmur3Partitioner容。該Murmur3Partitioner是默認卡桑德拉分區在亞馬遜 Keyspaces 和卡桑德拉 1.2 及更高版本。

RandomPartitioner

阿帕奇卡桑德拉兼RandomPartitioner容。對於早RandomPartitioner於卡桑德拉 1.2 版本的默認卡桑德拉分區程序。

Keyspaces 默認分區程序

會DefaultPartitioner傳回與相同的token函數結果RandomPartitioner。

分割程式設定會在帳戶層級針對每個區域套用。例如，如果您變更美國東部 (維吉尼亞北部) 的磁碟分割程式，則變更會套用至此區域中相同帳戶中的所有資料表。您可以隨時安全地變更磁碟分割程式。請注意，組態變更大約需要 10 分鐘才能完成。當您變更磁碟分割程式設定時，不需要重新載入 Amazon Keyspaces 資料。用戶端會在下次連線時自動使用新的磁碟分割程式設定。

您可以通過使用AWS Management Console或卡桑德拉查詢語言（CQL）更改分區。

AWS Management Console

使用 Amazon Keyspaces 控制台更改分區程序

1. 登錄到AWS Management Console，並打開亞馬遜 Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。
2. 在導覽窗格中，選擇組態。
3. 在 [組態] 頁面上，移至 [編輯磁碟分割程式]。
4. 選擇與您的卡桑德拉版本兼容的分區程序。套用磁碟分割程式變更大約需要 10 分鐘。

> **Note**
>
> 組態變更完成後，您必須中斷連線並重新連線至 Amazon Keyspaces，才能使用新磁碟分割程式的請求。

Cassandra Query Language (CQL)

1. 若要查看帳戶設定了哪個磁碟分割程式，您可以使用下列查詢。

```
SELECT partitioner from system.local;
```

如果分區程序尚未更改，則查詢具有以下輸出。

```
partitioner
-----------------------------------------
com.amazonaws.cassandra.DefaultPartitioner
```

2. 若要將磁碟分割程式更新為磁Murmur3碟分割程式，您可以使用下列陳述式。

```
UPDATE system.local set
 partitioner='org.apache.cassandra.dht.Murmur3Partitioner' where key='local';
```

3. 請注意，此組態變更大約需要 10 分鐘才能完成。若要確認已設定磁碟分割程式，您可以再次執行SELECT查詢。請注意，由於最終讀取的一致性，響應可能還不會反映最近完成的分區程序更改的結果。如果在短時間後再次重複SELECT操作，則響應應返回最新的數據。

```
SELECT partitioner from system.local;
```

> **Note**
>
> 您必須中斷連接並重新連接到 Amazon Keyspaces，以便請求使用新的磁碟分割程式。

使用 Amazon Keyspaces 資源的標籤和標籤

您可以使用標籤標記 Amazon Keyspaces（對於 Apache 卡桑德拉）資源。標籤可讓您以不同的方式對資源進行分類，例如，依目的、擁有者、環境或其他條件。標籤可協助您執行以下作業：

- 根據您指派給資源的標籤來快速識別資源。
- 請參閱依標籤劃分的 AWS 帳單。
- 根據標籤控制對 Amazon Keyspaces 資源的存取。如需使用標籤的 IAM 政策範例，請參閱the section called “基於 Amazon Keyspaces 標籤的授權”。

Amazon 彈性運算雲（Amazon EC2），Amazon 簡單存儲服務（亞馬遜 S3），亞馬遜 Keyspaces 等服務支持標記。 AWS 有效標記可讓您跨帶有特定標籤的服務來建立報告，以提供成本深入資訊。

若要開始使用標記，請執行以下操作：

1. 了解 亞馬遜 Keyspaces 的標記限制。
2. 使用 亞馬遜 Keyspaces 的標記操作 建立標籤。
3. 用亞馬遜 Keyspaces 的成本分配報告於追蹤每個使用中標籤的 AWS 成本。

最後，最好遵循最佳標記策略。如需相關資訊，請參閱 AWS 標記策略。

亞馬遜 Keyspaces 的標記限制

每個標籤皆包含由您定義的索引鍵和值。將適用以下限制：

- 每個 Amazon Amazon Amazon Val。若您嘗試新增現有的標籤 (相同索引鍵)，現有標籤的值會更新為新的值。
- 套用至金鑰空間的標籤不會自動套用至該金鑰空間內的表格。要將相同的標籤應用於密鑰空間及其所有表，必須單獨標記每個資源。

- 當您建立多區域金鑰空間或表格時，您在建立程序期間定義的任何標籤都會自動套用至所有區域中的所有金鑰空間和表格。當您使用ALTER KEYSPACE或變更現有的標籤時ALTER TABLE，更新只會套用至您要進行變更的區域中的金鑰空間或資料表。
- 值就像標籤類別 (索引鍵) 內的描述項。在亞馬遜 Keyspaces 的值不能為空白或為 null。
- 標籤鍵與值皆區分大小寫。
- 鍵長度上限為 128 個 Unicode 字元。
- 值長度上限為 256 個 Unicode 字元。
- 允許的字元為字母、空格和數字，以及下列特殊字元：`+ - = . _ : /`
- 每一資源標籤數最多為 50。
- AWS指派標籤名稱和值會自動指派字`aws:`首，您無法指派此值。 AWS指定的標籤名稱不會計入 50 的標籤限制。使用者指派的標籤名稱在成本分配報告中具有字首 `user:`。
- 標籤的套用不可回溯。

亞馬遜 Keyspaces 的標記操作

您可以使用亞馬遜 Keyspaces（對於 Apache 卡桑德拉）控制台，AWSCLI 或卡桑德拉查詢語言（CQL）添加，列出，編輯或刪除密鑰空間和表的標籤。然後，您可以啟動這些使用者定義的標籤，讓它們出現在 AWS Billing and Cost Management 主控台中，以便追蹤成本配置。如需詳細資訊，請參閱亞馬遜 Keyspaces 的成本分配報告。

若要進行大量編輯，您也可以使用主控台上的標籤編輯器。如需詳細資訊，請參閱 AWS Resource Groups 使用者指南中的使用標籤編輯器。

主題

- 使用主控台將標籤新增標籤至新的或現有金鑰空間和資料表
- 使用AWS CLI 將標籤新增標籤至新的或現有金鑰空間和資料表
- 使用 CQL 將標籤新增標籤至新的或現有金鑰空間和資料表

使用主控台將標籤新增標籤至新的或現有金鑰空間和資料表

建立新的金鑰空間和資料表時，您可以使用 Amazon KeySpaces 主控台將標籤新增至新的金鑰空間和資料表。您也可以為現有資料表新增、編輯或刪除標籤。

在創建密鑰空間時標記密鑰空間（控制台）

1. 登入AWS Management Console，並在 https://console.aws.amazon.com/keyspaces/home 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇 Keyspaces 間，然後選擇建立金鑰空間。
3. 在 [建立金鑰空間] 頁面上，提供金鑰空間的名稱。為標籤輸入金鑰和值，然後選擇新增標籤。
4. 選擇建立金鑰空間。

若要在建立表格時標記表格 (主控台)

1. 登入AWS Management Console，並在 https://console.aws.amazon.com/keyspaces/home 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇 Tables (資料表)，然後選擇 Create table (建立資料表)。
3. 在「表格詳細資訊」段落的「建立表格」頁面中，選取索引鍵空間並輸入表格的名稱。
4. 在「結構描述」區段中，為您的資料表建立結構定義。
5. 在 [表格設定] 區段中，選擇 [自訂設定]。
6. 繼續前往「表格標籤 — 選用」區段，然後選擇「新增標籤」以建立新標籤。
7. 選擇 建立資料表 。

為現有的資源加上標籤 (主控台)

1. 登入AWS Management Console，並在 https://console.aws.amazon.com/keyspaces/home 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇 Keyspaces 或資料表。
3. 在清單中選擇金鑰空間或資料表。然後選擇管理標籤來新增、編輯或刪除標籤。

如需標籤結構的相關資訊，請參閱 亞馬遜 Keyspaces 的標記限制。

使用AWS CLI 將標籤新增標籤至新的或現有金鑰空間和資料表

本節中的範例將示範如何在建立金鑰空間和資料表時使用AWS CLI 指定標籤、如何從現有資源新增或移除標籤，以及如何列出標籤。

下列範例說明如何建立標籤新資料表。該命令創建一個表 MyTable 在一個已經存在的密鑰空間我的密鑰空間。請注意，該命令已被分解為不同的行，以幫助提高可讀性。

```
aws keyspaces create-table --keyspace-name 'myKeyspace' --table-name 'myTable'
            --schema-definition 'allColumns=[{name=id,type=int},{name=name,type=text},
{name=date,type=timestamp}],partitionKeys=[{name=id}]'
            --tags 'key=key1,value=val1' 'key=key2,value=val2'
```

下列範例說明如何將新標籤新增至現有資料表。

```
aws keyspaces tag-resource --resource-arn 'arn:aws:cassandra:us-east-1:111222333444:/
keyspace/myKeyspace/table/myTable' --tags 'key=key3,value=val3' 'key=key4,value=val4'
```

下列範例說明如何列出指定資源的標籤。

```
aws keyspaces list-tags-for-resource --resource-arn 'arn:aws:cassandra:us-
east-1:111222333444:/keyspace/myKeyspace/table/myTable'
```

最後一個命令的輸出如下所示。

```
{
    "tags": [
        {
            "key": "key1",
            "value": "val1"
        },
        {
            "key": "key2",
            "value": "val2"
        },
        {
            "key": "key3",
            "value": "val3"
        },
        {
            "key": "key4",
            "value": "val4"
        }
    ]
}
```

使用 CQL 將標籤新增標籤至新的或現有金鑰空間和資料表

以下範例說明如何使用 CQL 在建立金鑰空間和資料表時指定標籤，以及在建立金鑰空間和資料表時指定標籤。

下列範例建立標籤的新金鑰空間。

```
CREATE KEYSPACE mykeyspace WITH TAGS = {'key1':'val1', 'key2':'val2'} ;
```

下列範例建立標籤的新資料表。

```
CREATE TABLE mytable(...) WITH TAGS = {'key1':'val1', 'key2':'val2'};
```

使用其他命令標記語句中的資源。

```
CREATE KEYSPACE mykeyspace WITH REPLICATION = {'class': 'Simple Strategy'} AND TAGS
 = {'key1':'val1', 'key2':'val2'};
```

以下範例說明如何在現有金鑰空間和資料表中新增標籤。

```
ALTER KEYSPACE mykeyspace ADD TAGS {'key1':'val1', 'key2':'val2'};
```

```
ALTER TABLE mytable DROP TAGS {'key1':'val1', 'key2':'val2'};
```

若要讀取附加至資源的標籤，請使用下列 CQL 陳述式。

```
SELECT * FROM system_schema_mcs.tags WHERE valid_where_clause;
```

此WHERE條款為必要條款，且必須是下列其中一種格式：

- keyspace_name = **'mykeyspace'** AND resource_type = 'keyspace'
- keyspace_name = **'mykeyspace'** AND resource_name = **'mytable'**
- resource_id = *arn*

範例：

下面的查詢顯示一個密鑰空間是否有標籤。

```
SELECT * FROM system_schema_mcs.tags WHERE keyspace_name = 'mykeyspace' AND
 resource_type = 'keyspace';
```

查詢的輸出結果看起來會如下所示：

```
resource_id                                                        | keyspace_name |
 resource_name | resource_type | tags
--------------------------------------------------------------------+---------------
+---------------+---------------+------
arn:aws:cassandra:us-east-1:123456789:/keyspace/mykeyspace/      | mykeyspace    |
 mykeyspace    | keyspace      | {'key1': 'val1', 'key2': 'val2'}
```

下面的查詢顯示了一個表的標籤。

```
SELECT * FROM system_schema_mcs.tags WHERE keyspace_name = 'mykeyspace' AND
 resource_name = 'mytable';
```

該查詢的輸出結果看起來會如下所示：

```
resource_id                                                                |
 keyspace_name | resource_name | resource_type | tags
-----------------------------------------------------------------------------
+---------------+---------------+---------------+------
arn:aws:cassandra:us-east-1:123456789:/keyspace/mykeyspace/table/mytable    |
 mykeyspace    | mytable       | table         | {'key1': 'val1', 'key2': 'val2'}
```

亞馬遜 Keyspaces 的成本分配報告

AWS 會使用標籤來組織成本分配報告上的資源成本。AWS 提供兩種成本分配標籤：

- AWS 產生的標籤。AWS 為您定義、建立和套用此標籤。

- 使用者定義的標籤。您可以定義、建立和套用這些標籤。

您必須分別啟用這兩種標籤，它們才會顯示在 Cost Explorer 或成本分配報告中。

啟用 AWS 產生的標籤：

1. 登入 AWS Management Console，並於 https://console.aws.amazon.com/billing/home#/ 開啟「帳單和成本管理」主控台。
2. 在導覽窗格中，選擇 Cost Allocation Tags (成本分配標籤)。
3. 在 AWS-Generated Cost Allocation Tags (AWS 產生的成本分配標籤) 下，選擇 Activate (啟用)。

若要啟用使用者定義的標籤：

1. 登入 AWS Management Console，並於 https://console.aws.amazon.com/billing/home#/ 開啟「帳單和成本管理」主控台。
2. 在導覽窗格中，選擇 Cost Allocation Tags (成本分配標籤)。
3. 在 User-Generated Cost Allocation Tags (使用者產生的成本分配標籤) 下，選擇 Activate (啟用)。

建立並啟用標籤後，AWS 會產生成本分配報告，內含按作用中標籤分組的用量與成本。成本分配報告包含每個計費期間的所有 AWS 成本。該報告同時包含有標籤和沒標籤的資源，以便您可清楚地整理資源的費用。

> **Note**
>
> 目前，從 Amazon Keyspaces 傳出的任何資料都不會在成本分配報告上依標籤細分。

如需詳細資訊，請參閱使用成本分配標籤。

使用 Amazon Keyspaces 進行設計和架構的最佳實務

使用本節可在使用 Amazon Keyspaces 時快速找到最大化效能並將輸送量成本降至最低的建議。

內容

    - 瞭解您的 Application Auto Scaling 放設
    - 如何識別目標使用率低的資料表 (<=50%)
    - 如何處理具有季節性差異的工作負載
    - 如何處理具有未知模式的尖峰工作負載
    - 如何處理具有連結應用程式的工作負載
- 識別您未使用的資源
    - 如何識別未使用資源
    - 識別未使用的資料表資源
    - 清除未使用的資料表資源
    - 清除未使用的 point-in-time 復原 (PITR) 備份
- 評估您的資料表用量模式
    - 執行較少高度一致性讀取操作
    - 啟用存留時間 (TTL)
- 評估您是否具有適當大小的佈建容量
    - 如何從 Amazon Keyspaces 表中檢索消費指標
    - 如何識別佈建不足的 Amazon Keyspaces 表
    - 如何識別過度佈建的 Amazon Keyspaces 表

適用於 Amazon Keyspaces 的 NoSQL 設計

像 Amazon Keyspaces 這樣的 NoSQL 資料庫系統會使用替代模型進行資料管理，例如鍵值配對或文件儲存。當您從關聯式資料庫管理系統切換到 NoSQL 資料庫系統 (例如 Amazon 金 Keyspaces) 時，瞭解主要的差異和特定的設計方法非常重要。

主題

- 關聯式資料設計與 NoSQL 間的差異
- NoSQL 設計的兩個主要概念
- NoSQL 設計方法

關聯式資料設計與 NoSQL 間的差異

關聯式資料庫管理系統 (RDBMS) 和 NoSQL 資料庫各有優劣：

- RDBMS 可以彈性地查詢資料，但查詢相對昂貴，而且在高流量的狀況下擴展不易 (請參閱 the section called “建立資料模型”)。
- 在 NoSQL 數據庫（例如 Amazon Keyspaces）中，可以有效地以有限數量的方式查詢數據，在此之外查詢可能昂貴且緩慢。

這些差異讓兩種系統之間的資料庫設計不同：

- 在 RDBMS 中，您會針對靈活性而進行設計，而不需擔心實行詳細資訊或效能。查詢最佳化通常不會影響結構描述設計，但標準化相當重要。
- 在 Amazon Keyspaces 中，您可以專門設計結構描述，以盡可能快速且經濟實惠地進行最常見和最重要的查詢。系統會打造您的資料結構，以符合企業使用案例的特定要求。

NoSQL 設計的兩個主要概念

NoSQL 設計思維與 RDBMS 設計不同。針對 RDBMS，您可以直接建立標準化的資料模型，而不需考量存取模式。您可以在稍後有新問題與查詢要求時擴展此模型。您可以將每種資料整理至其資料表。

NoSQL 設計的不同之處

- 相比之下，在知道需要回答的問題之前，您不應該開始為 Amazon Keyspaces 設計架構。必須事先了解企業問題和應用程式使用案例。
- 您應該在 Amazon Keyspaces 應用程序中保持盡可能少的表。擁有較少的資料表可保持更高的可擴展性、需要較少的許可管理，並減少 Amazon Keyspaces 應用程式的開銷。它還可以幫助降低整體備份成本。

NoSQL 設計方法

設計 Amazon Keyspaces 應用程式的第一步是識別系統必須滿足的特定查詢模式。

特別是，請務必了解應用程式存取模式的三種基本屬性，再開始進行：

- 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。
- 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：Amazon Keyspaces 透過增加可用於處理查詢的實體分區數量，以及在這些分區之間有效地分配資料來擴展。事先了解尖峰查詢負載，將可能有助於判斷如何分割資料以有效使用輸入/輸出容量。

在您識別特定查詢要求後，您可以根據管理效能的一般原則來組織資料：

- 將相關的資料保持在一起。 20 年前針對路由表最佳化的研究發現，「參照本地性」是提升回應時間的最重要的單一因素，也就是將相關資料放在同一個位置。這在 NoSQL 系統也同樣適用，將相關資料保持在鄰近位置對成本與效能有重大影響。您應盡可能將 NoSQL 系統中的相關項目放置在靠近的位置，而非在多個資料表之間分配相關的資料項目。

  一般而言，您應該在 Amazon Keyspaces 應用程式中維護盡可能少的表格。

  例外狀況包含大量時間序列資料涉及其中的案例，或存取模式極為不同的資料集。含反轉索引的單一資料表通常可以啟用簡單查詢，來建立並擷取您應用程式所需的複雜階層資料結構。
- 使用排序。 若相關項目的索引鍵設計為能一起排序，則可以一起分組，以更有效地進行查詢。此為重要的 NoSQL 設計策略。
- 發佈佇列。 亦請注意不要針對資料庫的一部分集中進行大量查詢 (這會超過輸入/輸出容量)。而您應盡可能將資料索引鍵平均分配到這些分割區，避免「熱點」。

這些一般原則轉化為一些常見的設計模式，您可以使用這些模式在 Amazon Keyspaces 中有效地建立資料模型。

客戶端驅動程序連接到 Amazon Keyspaces（對於 Apache 卡桑德拉）

要與 Amazon Keyspaces 進行通信，您可以使用任何您選擇的現有 Apache 卡桑德拉客戶端驅動程序。由於 Amazon Keyspaces 是無伺服器服務，因此建議您針對應用程式的輸送量需求最佳化用戶端驅動程式的連線組態。本主題介紹最佳作法，包括如何計算應用程式需要多少連線，以及監視和錯誤處理連線。

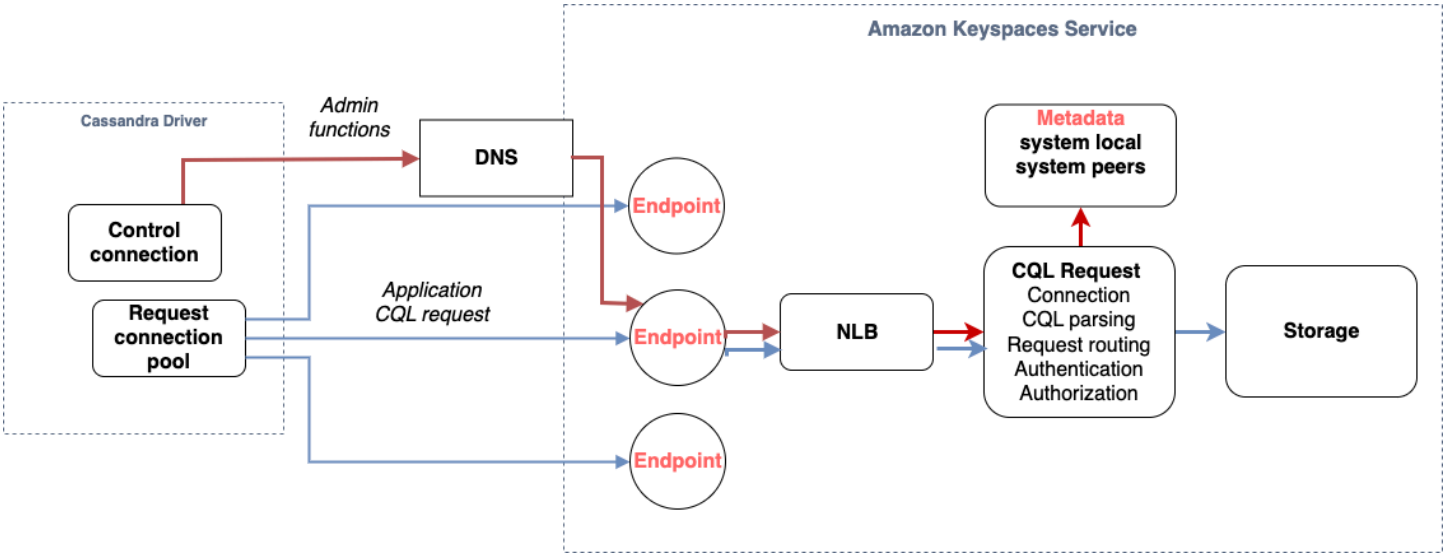
主題

- Amazon Keyspaces 中的連接如何工作
- 如何在 Amazon Keyspaces 中配置連接
- 如何透過 Amazon Keyspaces 中的 VPC 端點設定連線
- 如何監視 Amazon Keyspaces 中的連接
- 如何處理 Amazon Keyspaces 中的連接錯誤

Amazon Keyspaces 中的連接如何工作

本節概述用戶端驅動程式連線在 Amazon Keyspaces 中的運作方式。由於 Cassandra 用戶端驅動程式設定錯誤可能會導致 Amazon Keyspace 中的`PerConnectionRequestExceeded`事件，因此需要在用戶端驅動程式組態中設定適當數量的連線，以避免這些和類似的連線錯誤。

連線至 Amazon Keyspaces 時，驅動程式需要種子端點來建立初始連線。Amazon Keyspaces 使用 DNS 將初始連線路由到眾多可用端點之一。端點會連接至網路負載平衡器，進而與叢集中的其中一個要求處理常式建立連線。建立初始連線之後，用戶端驅動程式會從資料`system.peers`表收集所有可用端點的相關資訊。使用此資訊，用戶端驅動程式可以建立與所列端點的其他連線。用戶端驅動程式可建立的連線數目受用戶端驅動程式設定中指定的本機連線數目限制。默認情況下，大多數客戶端驅動程序為每個端點建立一個連接，並建立連接池到 Cassandra 並通過該連接池進行負載平衡查詢。雖然可以建立多個連接到同一個端點，但在網絡負載平衡器後面，它們可能會連接到許多不同的請求處理程序。透過公用端點連線時，建立一個連線至`system.peers`表格中列出的九個端點，每個端點都會產生九個連線至不同的要求處理程式。



如何在 Amazon Keyspaces 中配置連接

Amazon Keyspaces 每秒每個 TCP 連線最多可支援 3,000 個 CQL 查詢。由於驅動程式可建立的連線數目沒有限制，因此我們建議每個連線每秒只鎖定 500 個 CQL 要求，以允許額外負荷、流量突增，以及更佳的負載平衡。請遵循下列步驟，以確保驅動程式的連線已正確設定，以滿足應用程式的需求。

增加驅動程序在其連接池中維護的每個 IP 地址的連接數。

- 大多數卡桑德拉驅動程序建立了一個連接池卡桑德拉，並通過該連接池進行負載平衡查詢。大多數驅動程序的默認行為是建立到每個端點的單個連接。Amazon Keyspaces 會向驅動程式公開九個對

等 IP 地址，因此根據大多數驅動程式的預設行為，這會產生 9 個連線。Amazon Keyspaces 每秒每個 TCP 連線最多支援 3,000 個 CQL 查詢，因此，使用預設設定的驅動程式最大 CQL 查詢輸送量為每秒 27,000 個 CQL 查詢。如果您使用驅動程式的預設設定，單一連線可能必須處理超過每秒 3,000 CQL 查詢的最大 CQL 查詢輸送量。這可能會導致`PerConnectionRequestExceeded`事件。

- 若要避免發生`PerConnectionRequestExceeded`事件，您必須將驅動程式設定為每個端點建立其他連線以分配輸送量。
- 作為 Amazon Keyspaces 中的最佳實務，假設每個連線每秒可支援 500 個 CQL 查詢。
- 這表示，對於需要支援每秒估計 27,000 個 CQL 查詢分散在九個可用端點的實際執行應用程式，您必須為每個端點設定六個連線。這可確保每個連線每秒處理不超過 500 個要求。

根據應用程式的需求，計算您需要為驅動程式設定的每個 IP 位址的連線數目。

若要判斷您需要為應用程式設定每個端點的連線數目，請考慮下列範例。您有一個應用程式每秒需要支援 20,000 個 CQL 查詢，其中包含 10,000 個`INSERT`、`SELECT`、5,000 個和 5,000 個`DELETE`作業。Java 應用程式在亞馬遜彈性容器服務 (Amazon ECS) 上的三個執行個體上執行，其中每個執行個體都會建立到 Amazon Keyspaces 的單一工作階段。您可以用來估計您需要為驅動程式設定多少連線的計算方式，會使用下列輸入。

1. 您的應用程式需要支援的每秒要求數目。
2. 因維護或失敗而減去一個的可用執行個體數目。
3. 可用端點的數目。如果您透過公用端點進行連線，則會有九個可用端點。如果您使用的是 VPC 端點，則會有兩到五個可用端點之間，視區域而定。
4. 每個連線每秒使用 500 個 CQL 查詢作為 Amazon Keyspaces 的最佳實務。
5. 四捨五入結果。

在此範例中，公式如下所示。

```
20,000 CQL queries / (3 instances - 1 failure) / 9 public endpoints / 500 CQL queries
 per second = ROUND(2.22) = 3
```

根據此計算，您需要在驅動程式組態中為每個端點指定三個本機連線。對於遠端連線，每個端點僅設定一個連線。

如何透過 Amazon Keyspaces 中的 VPC 端點設定連線

透過私有 VPC 端點進行連線時，您最有可能有 3 個端點可用。根據可用區域的數量以及指派的 VPC 中的子網路數目，每個區域的 VPC 端點數目可能會有所不同。美國東部 (維吉尼亞北部) 區域有五個可用區域，您最多可以有五個 Amazon Keyspaces 端點。美國西部 (加利佛尼亞北部) 區域有兩個可用區域，而且您最多可以有兩個 Amazon Keyspaces 端點。端點數目不會影響規模，但會增加您需要在驅動程式組態中建立的連線數目。請考量下列範例。您的應用程式需要支援 20,000 個 CQL 查詢，並且在 Amazon ECS 上的三個執行個體上執行，其中每個執行個體都會建立一個到 Amazon Keyspaces 的工作階段。唯一的區別是不同的端點數量可用 AWS 區域。

美國東部 (維吉尼亞北部) 區域所需的連線：

```
20,000 CQL queries / (3 instances - 1 failure) / 5 private VPC endpoints / 500 CQL
 queries per second = 4 local connections
```

美國西部 (加利佛尼亞北部) 區域所需的連線：

```
20,000 CQL queries / (3 instances - 1 failure) / 2 private VPC endpoints / 500 CQL
 queries per second = 10 local connections
```

> **Important**
>
> 使用私有 VPC 端點時，Amazon Keyspaces 需要其他許可，才能動態探索可用的 VPC 端點並填入表格。`system.peers`如需詳細資訊，請參閱 the section called “使用介面 VPC 端點資訊填入`system.peers`表格項目”。

透過使用不同的私有 VPC 端點存取 Amazon Keyspaces 時 AWS 帳戶，很可能只會看到單一 Amazon Keyspaces 端點。同樣地，這不會影響 Amazon Keyspace 的可能輸送量規模，但可能需要增加驅動程式組態中的連線數量。此範例顯示單一可用端點的相同計算方式。

```
20,000 CQL queries / (3 instances - 1 failure) / 1 private VPC endpoints / 500 CQL
 queries per second = 20 local connections
```

若要進一步了解使用共用 VPC 跨帳戶存取 Amazon Keyspaces 的詳細資訊，請參閱。the section called “共享 VPC 中的跨帳戶存取權”

如何監視 Amazon Keyspaces 中的連接

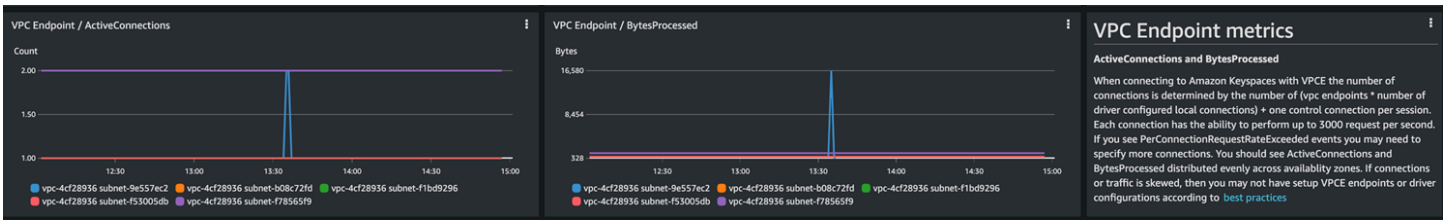
若要協助識別應用程式所連線的端點數目，您可以記錄在`system.peers`表格中探索到的對等端點數目。下面的例子是 Java 代碼的一個例子，它打印的連接已建立後的對等數量。

```
ResultSet result = session.execute(new SimpleStatement("SELECT * FROM system.peers"));

logger.info("number of Amazon Keyspaces endpoints:" + result.all().stream().count());
```

> **Note**
>
> CQL 主控台或主 AWS 控台未部署在 VPC 內，因此會使用公用端點。因此，從位於 VPCE 以外的應用程式執行`system.peers`查詢通常會產生 9 個對等。打印每個對等端的 IP 地址也可能會有所幫助。

您也可以透過設定 VPCE Amazon 指標來觀察使用 VPC 端點時的對等數量。 CloudWatch 在中 CloudWatch，您可以看到與 VPC 端點建立的連線數目。卡桑德拉驅動程序建立每個端點發送 CQL 查詢和控制連接來收集系統表信息的連接。下圖顯示使用驅動程式設定中設定的 1 個連線連線連線至 Amazon Keyspaces 後的 VPC 端點 CloudWatch 指標。此指標顯示六個作用中連線，其中包含一個控制連線和五個連線 (跨可用區域的每個端點 1 個)。



若要開始使用 CloudWatch 圖形監視連線數目，您可以在 Amazon Keyspaces AWS CloudFormation 範本存放庫 GitHub 中部署此範本。

如何處理 Amazon Keyspaces 中的連接錯誤

當超過每個連接配額 3,000 請求時，Amazon Keyspaces 返回一個`PerConnectionRequestExceeded`事件，卡桑德拉驅動程序接收或異常。`WriteTimeout` `ReadTimeout`您應該在 Cassandra 重試策略或應用程序中使用指數輪詢重試此異常。您應該提供指數輪詢，以避免發送額外的請求。

預設重試原則會嘗試`try next host`在查詢計畫中執行。由於 Amazon Keyspaces 在連線到 VPC 端點時可能會有一到三個可用端點，因此您也可能會在應用程式日誌`NoHostAvailableException`中

看到除了WriteTimeout和ReadTimeout例外狀況之外的。您可以使用 Amazon Keyspaces 提供的重試政策，這些政策會在相同的端點上重試，但跨不同的連線重試。

您可以在 Amazon Keyspaces Java 代碼示例存儲庫 GitHub 中找到 Java 的指數重試政策的示例。您可以在 Github 上找到其他語言示例例子 Amazon Keyspaces 代碼示例存儲庫。

Amazon Keyspaces 中的數據建模（阿帕奇卡桑德拉）

本主題介紹了 Amazon Keyspaces 中的數據建模概念（對於 Apache 卡桑德拉）。您可以使用本節尋找設計符合應用程式資料存取模式的資料模型的建議。使用 Amazon Keyspaces 時，實作資料模型最佳實務可改善效能並將輸送量成本降至最低。

若要更輕鬆地視覺化和設計資料模型，您可以使用 NoSQL 工作台。

主題

- 如何在 Amazon 密鑰 Keyspaces 中有效使用分區密鑰

如何在 Amazon 密鑰 Keyspaces 中有效使用分區密鑰

唯一識別 Amazon Keyspaces 表格中每一列的主索引鍵可包含一個或多個分區索引鍵資料欄 (決定資料儲存在哪些分區) 以及一或多個選用的叢集資料行 (定義資料在分割區內叢集和排序方式)。

由於分割區索引鍵會建立資料儲存在的分割區數目，以及資料在這些分割區之間的分散方式，因此您選擇分割區索引鍵的方式會對查詢效能產生重大影響。通常，您應該為磁盤上所有分區的統一活動設計應用程序。

將應用程式的讀取和寫入活動平均分配到所有分割區，有助於將輸送量成本降至最低，這適用於隨需和佈建的讀取/寫入容量模式。例如，如果您使用已佈建的容量模式，您可以決定應用程式所需的存取模式，並估計每個表格所需的讀取容量單位 (RCU) 和寫入容量單位 (WCU) 總計。只要針對指定分區的流量不超過 3,000 個 RCU 和 1,000 個 WCU，Amazon Keyspaces 就會使用您佈建的輸送量支援您的存取模式。

Amazon Keyspaces space 透過提供突增容量，在每個分區輸送量佈建中提供額外的彈性，如需詳細資訊，請參閱。the section called “高載容量”

主題

- 使用寫入分片在 Amazon Keyspaces 中平均分配工作負載

使用寫入分片在 Amazon Keyspaces 中平均分配工作負載

在 Amazon 密鑰空間中更好地將寫入分配到分區之間的一種方法是擴展空間。您可以數種不同的方式來執行此動作：您可以新增一個額外的分割區索引鍵資料欄，以便在分割區之間分配資料列。或者，您可以使用根據您查詢內容計算的數字。

使用複合分區鍵和隨機值進行分片

將負載更均勻地分配到分割區之間的其中一個策略是新增額外的分割區索引鍵資料欄，您可以在其中寫入隨機數字。如此您就能在較大的空間中將寫入隨機化。

例如，請考慮下表，它具有代表日期的單一分區索引鍵。

```
CREATE TABLE IF NOT EXISTS tracker.blogs (
   publish_date date,
   title text,
   description int,
   PRIMARY KEY (publish_date));
```

若要在分割區之間更均勻地分配此表格，您可以加入儲存隨機數shard的額外分割索引鍵資料欄。例如：

```
CREATE TABLE IF NOT EXISTS tracker.blogs (
   publish_date date,
   shard int,
   title text,
   description int,
   PRIMARY KEY ((publish_date, shard)));
```

插入資料時，您可以在1shard和200之間選擇一個隨機數字。這會產生複合分區索引鍵值(2020-07-09, 1)(2020-07-09, 2)，例如、等等(2020-07-09, 200)。因為您隨機化了分割區索引鍵，每天對資料表的寫入會平均分配在多個分割區中。這可帶來更優良的平行處理與更高的整體傳輸量。

但是，要讀取給定日期的所有行，您必須查詢所有碎片的行，然後合併結果。例如，您首先要針對分割區索引鍵值發出SELECT陳述式(2020-07-09, 1)。然後發出另一個SELECT聲明(2020-07-09, 2)，等等，通過(2020-07-09, 200)。最後，您的應用程序必須合併所有這些SELECT語句的結果。

使用複合分區鍵和計算值進行分片

隨機化的策略可大幅改善寫入傳輸量。但是很難讀取特定的行，因為您不知道在寫入行shard時將哪個值寫入列。為了更容易閱讀個別列，您可以使用不同的策略。請不要使用隨機數在分割區之間分配資料列，而是使用可根據您要查詢的項目來計算的數字。

考量先前的範例，其中資料表會使用分割區索引鍵中的今天日期。現在假設每一行都有一個可訪問的title列，並且除了日期之外，您通常需要按標題查找行。在應用程式將資料列寫入資料表之前，它可以根據標題計算雜湊值，並使用它來填入資shard料行。計算應會產生介於 1 到 200 之間的數字，此分佈非常平均，與隨機策略產生的結果相當類似。

一個簡單的計算可能就足夠了, 如 UTF-8 代碼點值的乘積在標題中的字符, 模 200, + 1. 然後，複合分區鍵值將成為日期和計算結果的組合。

有了這項策略，即可在分割區索引鍵值之間，以及實體分割區之間平均分配寫入。您可以輕鬆地針對特定資料列和日期執行SELECT陳述式，因為您可以計算特定值的分區索引鍵title值。

要讀取給定日期的所有行，您仍然必須SELECT每個(2020-07-09, N)鍵（其中N為 1—200），然後您的應用程序必須合併所有結果。好處是您能避免讓單一「經常性」分割區索引鍵值承受所有工作負載。

優化 Amazon Keyspaces 表的成本

本節涵蓋如何優化現有 Amazon Keyspaces 表格成本的最佳實務。您應查看以下策略，了解哪一項成本最佳化策略最適合您的需求，並反覆採用這些策略。每種策略都提供了可能會影響您的成本的原因、如何尋找優化成本的機會，以及如何實作這些最佳實務以協助您節省成本的規範指引。

主題

- 在資料表層級評估您的成本
- 評估表格的容量模式
- 評估表格的 Application Auto Scaling 放設定
- 識別您未使用的資源
- 評估您的資料表用量模式
- 評估您是否具有適當大小的佈建容量

在資料表層級評估您的成本

在中找到的 Cost Explorer 工具 AWS Management Console 可讓您查看依類型劃分的成本，例如讀取、寫入、儲存和備份費用。您還可以查看依期間 (例如月份或日期) 彙總的這些成本。

Cost Explorer 的一個常見挑戰是，您無法輕易只檢視一個特定資料表的成本，因為 Cost Explorer 無法讓您依特定資料表的成本來篩選或分組。您可以在 Amazon Keyspaces 主控台的表格監控標籤中，檢視每個表的指標可計費表格大小 (位元組)。如果每個表需要更多與成本相關的資訊，本節說明如何在 Cost Explorer 中使用標記來執行個別的表格成本分析。

主題

- 如何查看單個 Amazon Keyspaces 表的成本
- Cost Explorer 的預設檢視
- 如何在 Cost Explorer 中使用和套用資料表標籤

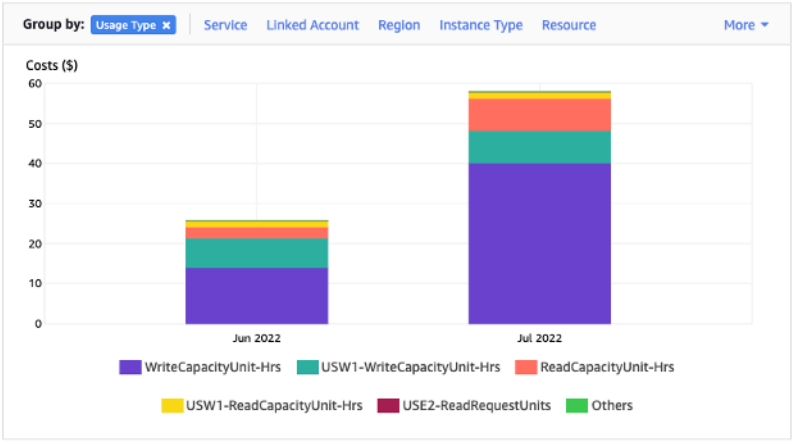
如何查看單個 Amazon Keyspaces 表的成本

您可以在主控台中查看 Amazon Keyspaces 表的基本資訊，包括主索引鍵結構描述、可計費表格大小和容量相關指標。您可以使用表格的大小來計算表格的每月儲存成本。例如，美國東部 (維吉尼亞北部) 的每 GB 0.25 USD。 AWS 區域

如果表格使用佈建的容量模式，則也會傳回目前的讀取容量單位 (RCU) 和寫入容量單位 (WCU) 設定。您可以使用此資訊來計算資料表目前的讀取和寫入成本。請注意，這些成本可能會改變，特別是如果您已使用 Amazon Keyspaces 自動擴展設定表格。

Cost Explorer 的預設檢視

Cost Explorer 中的預設檢視會提供圖表，顯示耗用資源的成本，例如輸送量和儲存體。您可以選擇按期間將這些成本分組，例如按月或按日總計。存儲，讀取，寫入和其他類別的成本也可以細分和比較。



如何在 Cost Explorer 中使用和套用資料表標籤

根據預設，Cost Explorer 不會提供任何一個特定資料表的成本摘要，因為它會將多個資料表的成本合併為一個總計。不過，您可以使用 AWS 資源標記，以中繼資料標籤來識別各個資料表。標籤是可用於各種用途的索引鍵值配對，例如識別屬於某個專案或部門的所有資源。如需詳細資訊，請參閱 the section called “使用標籤”。

在此範例中，我們使用名稱為表格MyTable。

1. 使用表格名稱的鍵和的值來設置一個標籤。MyTable
2. 在 Cost Explorer 中啟用此標籤，然後篩選標籤值，以深入了解各個資料表的成本。

> **Note**
>
> 標籤可能需要一到兩天的時間才會開始出現在 Cost Explorer 中

您可以在主控台中自行設定中繼資料標籤，或使用 CQL、或 AWS SDK 以程式設計方式設定 AWS CLI 中繼資料標籤。請考慮在組織的新標籤建立程序中要求設定 table_name 標籤。如需詳細資訊，請參閱 the section called “亞馬遜 Keyspaces 的成本分配報告”。

評估表格的容量模式

本節提供如何為 Amazon Keyspaces 表選取適當容量模式的概觀。每種模式都經過調整，以滿足不同工作負載在回應輸送量變化以及該用量計費方式方面的需求。在做出決定時，您必須平衡這些因素。

主題

- 有哪些可用的資料表容量模式
- 何時選取隨需容量模式
- 何時選取佈建容量模式
- 選擇資料表容量模式時應考慮的其他因素

有哪些可用的資料表容量模式

建立 Amazon Keyspaces 表格時，您必須選取隨需或佈建容量模式。如需詳細資訊，請參閱 the section called “讀/寫容量模式”。

隨需容量模式

隨需容量模式旨在消除規劃或佈建 Amazon Keyspaces 表的容量的需求。在此模式下，您的資料表會立即容納要求，而不需要擴展或縮減任何資源 (最多是表格先前尖峰輸送量的兩倍)。

隨選資料表的計費方式是根據表格計算實際請求的數量，因此您只需按使用量付費，而不是已佈建的項目付費。

佈建容量模式

佈建的容量模式是較傳統的模型，您可以在其中定義表格可用於請求的容量，直接或在 Application Auto Scaling 的協助下。由於會在任何指定時間為資料表佈建特定容量，因此是根據佈建的容量而非請求數目進行計費。瀏覽已配置的容量也可能導致表格拒絕要求，並減少應用程式使用者的體驗。

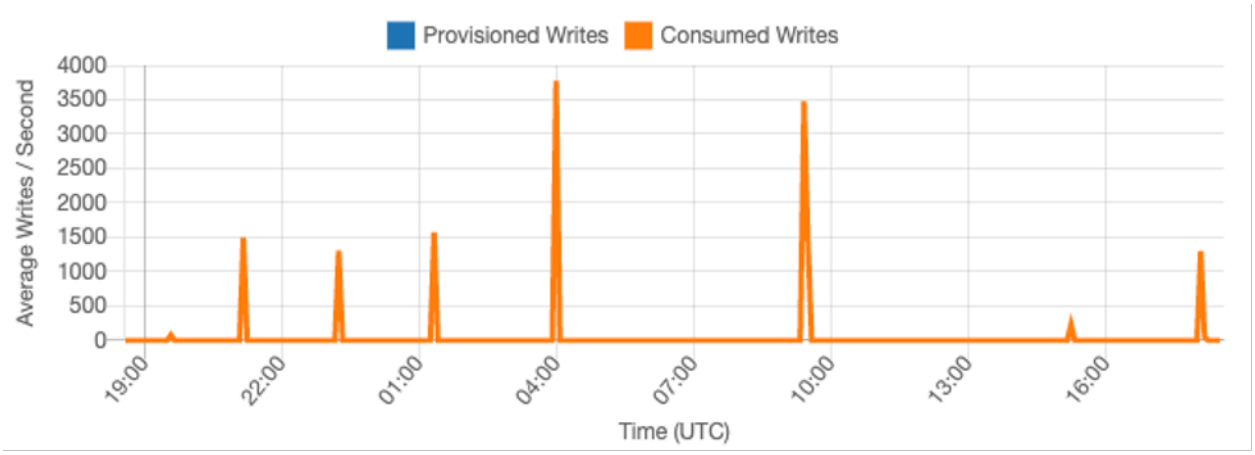
佈建的容量模式需要在未過度佈建或佈建資料表之間取得平衡，以達到兩者、低發生輸送量不足容量錯誤，以及最佳化的成本。

何時選取隨需容量模式

針對成本進行最佳化時，如果您有無法預測的工作負載，類似下圖所示的工作負載，隨選模式是您的最佳選擇。

這些因素會造成此類型的工作負載：

- 無法預測的請求時機 (導致流量尖峰)
- 變動的請求量 (由批次工作負載導致)
- 在特定小時內降至峰值的零或低於 18% (由開發或測試環境產生)



對於具有上述特性的工作負載，使用 Application Auto Scaling 維持足夠的容量，以便表格回應流量尖峰，可能會導致不想要的結果。表格可能會過度佈建且成本超過必要，或者表格可能已佈建，而且要求會導致不必要的低容量輸送量錯誤。在這種情況下，按需表是更好的選擇。

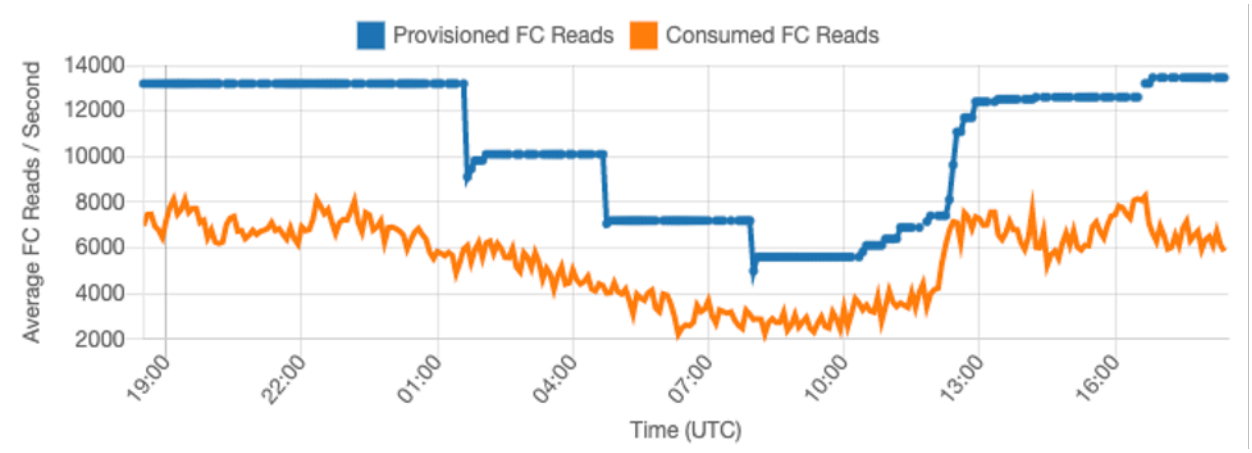
由於隨需資料表是按請求計費，因此您不需要在表格層級執行任何其他操作即可最佳化成本。您應該定期評估隨選資料表，以確認工作負載仍具有上述特性。如果工作負載已穩定，請考慮變更為佈建模式以維持成本最佳化。

何時選取佈建容量模式

佈建容量模式的理想工作負載是具有更可預測的使用模式，如下圖所示。

下列因素會造成可預測的工作負載：

- 特定一小時或一天內的可預測/周期性流量
- 有限度的短期突發流量



由於指定時間或日期內的流量較穩定，因此您可以將佈建的容量設定為相對接近表格實際使用的容量。最佳化佈建容量表格的成本最終是在不增加表格ThrottledRequests事件的情況下，將佈建容量 (藍線) 盡可能接近已耗用容量 (橘線) 的練習。兩條線之間的空間是兩者，浪費了容量，以及由於輸送量容量不足錯誤導致不良的用戶體驗的保險。

Amazon Keyspaces space 為佈建的容量表提供 Application Auto Scaling 能，可代表您自動平衡這個資料表。您可以全天追蹤已使用的容量，並根據少數變數設定表格的佈建容量。

容量單位下限

您可以設定表格的最小容量，以限制發生輸送量容量不足錯誤的情況，但不會降低表格的成本。如果您的表格的使用期間低，然後突然突然爆發高使用率，則設定最小值可防止「Application Auto Scaling」設定表格容量過低。

容量單位上限

您可以設定資料表的容量上限，以避免資料表的規模調整到高於預期的程度。考慮應用最大的開發或測試表，其中不需要大規模的負載測試。您可以為任何資料表設定最大值，但請務必在生產環境中使用資料表基準定期評估此設定，以避免意外的輸送量容量不足錯誤。

目標使用率

對於佈建容量資料表，設定資料表的目標使用率是實現成本最佳化的主要方法。在此處設定較低的百分比值會增加表格超量佈建的程度，從而增加成本，但降低輸送量容量不足錯誤的風險。設定較高的百分比值會隨表格超量佈建而減少，但會增加輸送量容量不足錯誤的風險。

選擇資料表容量模式時應考慮的其他因素

在兩種容量模式之間做出決定時，還有一些額外的因素值得考慮。

在兩種表格模式之間做出決定時，請考慮此額外 discount 對表格成本的影響程度。在許多情況下，即使是相對不可預測的工作負載，在具有預留容量的過度佈建容量表上執行，也能更具成本效益。

改善工作負載的可預測性

在某些情況下，工作負載似乎同時具有可預測且不可預測的模式。雖然這可以透過隨選表格輕鬆支援，但如果可以改善工作負載中無法預測的模式，成本可能會降低。

這些模式最常見的原因之一是批次匯入。這種類型的流量通常可能會超過表格的基準容量，以至於在執行時會發生輸送量容量不足錯誤的程度。若要在佈建容量資料表上執行這類工作負載，請考慮下列選項：

- 如果批次發生在排定的時間，您可以在應用程式執行之前排程增加應用程式 auto 擴展容量。

- 如果批次是隨機發生的，請考慮嘗試延長執行所需的時間，而不是盡可能快地執行。
- 為匯入增加期間，其中匯入的速度開始很小，但會在幾分鐘內緩慢增加，直到 Application Auto Scaling 有機會開始調整資料表容量為止。

評估表格的 Application Auto Scaling 放設定

本節提供如何評估 Amazon Keyspaces 表格上的 Application Auto Scaling 設定的概觀。Amazon Keyspaces Application Auto Scaling 是一項功能，可根據應用程式流量和目標使用率指標來管理表格輸送量。如此可確保您的資料表具有應用程式模式所需的容量。

「Application Auto Scaling」服務會監控您目前的表格使用率，並將其與目標使用率值進行比較：`TargetValue`。它會通知您是否需要增加或減少配置的容量。

主題

- 瞭解您的 Application Auto Scaling 放設
- 如何識別目標使用率低的資料表 (<=50%)
- 如何處理具有季節性差異的工作負載
- 如何處理具有未知模式的尖峰工作負載
- 如何處理具有連結應用程式的工作負載

瞭解您的 Application Auto Scaling 放設

您的營運團隊需定義目標使用率的正確值、初始步驟和最終值。這可讓您根據用來觸發應用程式 Auto Scaling 原則的歷史應用程式使用情況，正確定義這些值。使用率目標是在套用「應用 Application Auto Scaling 規則」規則之前，在一段時間內必須符合的總容量百分比。

當您設定高使用率目標 (約 90% 的目標) 時，表示您的流量必須在一段時間內高於 90%，才能啟 Application Auto Scaling。除非您的應用程式狀態穩定，且不會接收到流量尖峰，否則不應使用高使用率目標。

當您設定非常低的使用率 (目標小於 50%) 時，表示您的應用程式必須達到佈建容量的 50%，才會觸發 Application Auto Scaling 原則。除非您的應用程式流量成長快速，否則這通常會變成未使用容量和資源浪費。

如何識別目標使用率低的資料表 (<=50%)

您可以在 Amazon Keyspaces 資源中使用 AWS CLI 或 AWS Management Console 來監控和識別 Application Auto Scaling 政策：`TargetValues`

AWS CLI

1. 執行下列命令，傳回完整的資源清單：

```
aws application-autoscaling describe-scaling-policies --service-namespace
 cassandra
```

此命令將傳回發給任何 Amazon Keyspaces 資源的 Application Auto Scaling 政策的完整清單。若您只想從特定資料表擷取資源，您可以新增 `–resource-id parameter`。例如：

```
aws application-autoscaling describe-scaling-policies --service-namespace
 cassandra --resource-id "keyspace/keyspace-name/table/table-name”
```

2. 執行下列命令，只傳回特定資料表的 auto 調整規模政策

```
aws application-autoscaling describe-scaling-policies --service-namespace
 cassandra --resource-id "keyspace/keyspace-name/table/table-name”
```

Application Auto Scaling 政策的值會反白顯示如下。您必須確保目標值大於 50%，以避免過度佈建。您應該會獲得類似以下的結果：

```
{
    "ScalingPolicies": [
        {
            "PolicyARN": "arn:aws:autoscaling:<region>:<account-
id>:scalingPolicy:<uuid>:resource/keyspaces/table/table-name-scaling-policy",
            "PolicyName": $<full-gsi-name>”,
            "ServiceNamespace": "cassandra",
            "ResourceId": "keyspace/keyspace-name/table/table-name",
            "ScalableDimension": "cassandra:index:WriteCapacityUnits",
            "PolicyType": "TargetTrackingScaling",
            "TargetTrackingScalingPolicyConfiguration": {
                "TargetValue": 70.0,
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "KeyspacesWriteCapacityUtilization"
                }
            },
            "Alarms": [
                ...
            ],
            "CreationTime": "2022-03-04T16:23:48.641000+10:00"
```

```
        },
        {
            "PolicyARN": "arn:aws:autoscaling:<region>:<account-
id>:scalingPolicy:<uuid>:resource/keyspaces/table/table-name/index/<index-
name>:policyName/$<full-gsi-name>-scaling-policy",
            "PolicyName":$<full-table-name>”,
            "ServiceNamespace": "cassandra",
            "ResourceId": "keyspace/keyspace-name/table/table-name",
            "ScalableDimension": "cassandra:index:ReadCapacityUnits",
            "PolicyType": "TargetTrackingScaling",
            "TargetTrackingScalingPolicyConfiguration": {
                "TargetValue": 70.0,
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "CassandraReadCapacityUtilization"
                }
            },
            "Alarms": [
                ...
            ],
            "CreationTime": "2022-03-04T16:23:47.820000+10:00"
        }
    ]
}
```

AWS Management Console

1. 登入 AWS Management Console 並瀏覽至「開始使用」中的 CloudWatch 服務頁面 AWS Management Console。 AWS 區域 如有必要，請選取適當的。
2. 在左側導覽列中，選取 Tables (資料表)。在 Tables (資料表) 頁面上，選取資料表 Name (名稱)。
3. 在 [容量] 索引標籤的 [表格詳細資料] 頁面上，檢閱表格的 Application Auto Scaling 設定。

若您的目標使用率值小於或等於 50%，您應探索資料表使用率指標，查看指標是否佈建不足或過度佈建。

如何處理具有季節性差異的工作負載

請考慮下列情況：您的應用程式大部分時間都以最小平均值運作，但使用率目標很低，因此應用程式可以快速回應特定時間發生的事件，且您擁有足夠容量避免受到限流。應用程式在正常辦公時間 (上午 9

點至下午 5 點) 非常忙碌，但下班時間僅在基礎層級運作時，這種情況就很常見。由於某些使用者在早上 9 點之前開始連線，因此應用程式會使用此低閾值快速提升，以便在尖峰時段達到所需的容量。

此情況可能如下所示：

- 下午 5 點至上午 9 點之間，ConsumedWriteCapacityUnits 單位停留在 90 和 100 之間
- 使用者在上午 9 點前開始連線到應用程式，且容量單位大幅增加 (您看到的最大值為 1500 WCU)
- 平均而言，在工作期間，應用程式使用量介於 800 到 1200

如果先前的案例適用於您的應用程式，請考慮使用排定的應用程式 auto 調整規模，表格仍可設定「應用程式 Auto Scaling」規則，但目標使用率較低，只會在您需要的特定間隔佈建額外容量。

您可以使用執行下列步驟來建立排定的 auto 調整規則，該規則會根據一天中的時間和星期幾執行。
AWS CLI

1. 將您的 Amazon Keyspaces 表註冊為可擴展的目標 Application Auto Scaling。可擴展的目標是 Application Auto Scaling 可橫向擴展和縮減的資源。

```
aws application-autoscaling register-scalable-target \
    --service-namespace cassandra \
    --scalable-dimension cassandra:table:WriteCapacityUnits \
    --resource-id keyspace/keyspace-name/table/table-name \
    --min-capacity 90 \
    --max-capacity 1500
```

2. 根據您的需求設定排定的動作。

   您需要兩個規則來涵蓋該案例：一個用於擴展，另一個規則縮小規模。下列範例會顯示第一個擴充排程動作的規則。

```
aws application-autoscaling put-scheduled-action \
    --service-namespace cassandra \
    --scalable-dimension cassandra:table:WriteCapacityUnits \
    --resource-id keyspace/keyspace-name/table/table-name \
    --scheduled-action-name my-8-5-scheduled-action \
    --scalable-target-action MinCapacity=800,MaxCapacity=1500 \
    --schedule "cron(45 8 ? * MON-FRI *)" \
    --timezone "Australia/Brisbane"
```

   此範例顯示縮減排程動作的第二個規則。

```
aws application-autoscaling put-scheduled-action \
    --service-namespace cassandra \
    --scalable-dimension cassandra:table:WriteCapacityUnits \
    --resource-id keyspace/keyspace-name/table/table-name \
    --scheduled-action-name my-5-8-scheduled-down-action \
    --scalable-target-action MinCapacity=90,MaxCapacity=1500 \
    --schedule "cron(15 17 ? * MON-FRI *)" \
    --timezone "Australia/Brisbane"
```

3. 執行以下命令，驗證這兩個規則皆已啟用。

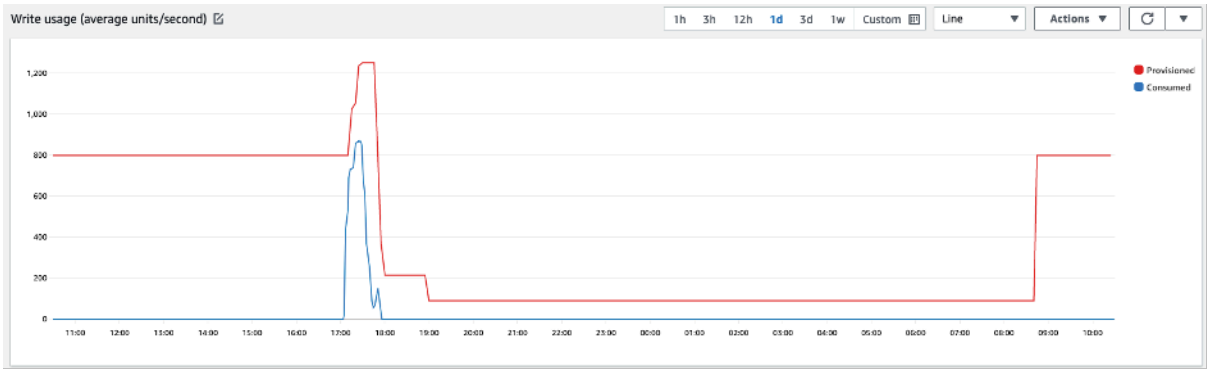
```
aws application-autoscaling describe-scheduled-actions --service-namespace
 cassandra
```

您應該會取得如下結果：

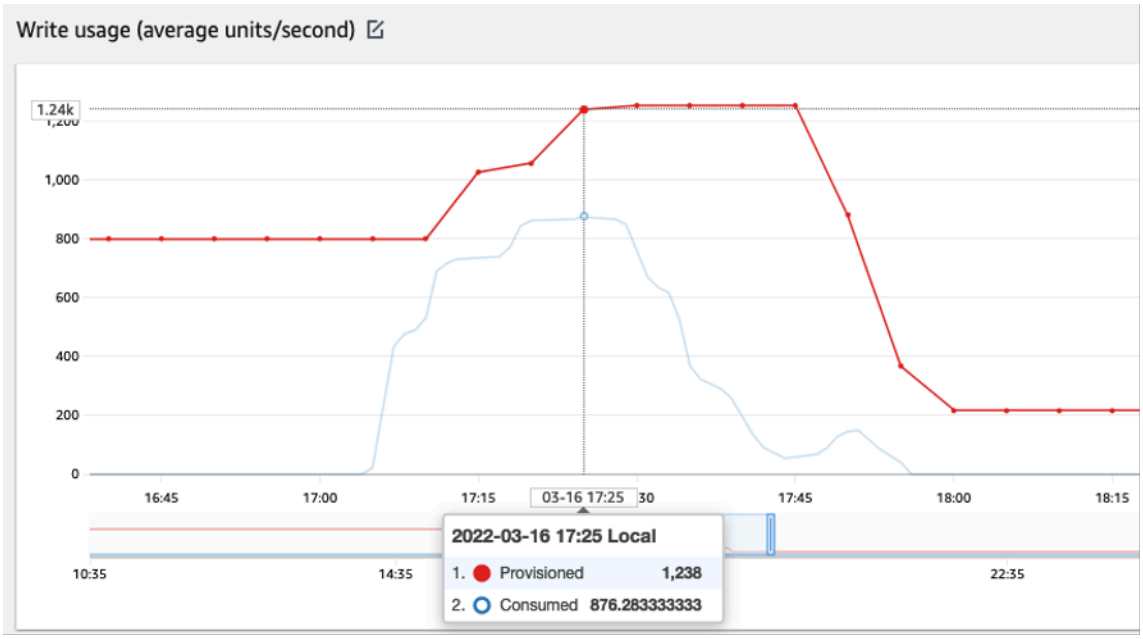
```
{
    "ScheduledActions": [
        {
            "ScheduledActionName": "my-5-8-scheduled-down-action",
            "ScheduledActionARN":
 "arn:aws:autoscaling:<region>:<account>:scheduledAction:<uuid>:resource/keyspaces/
table/table-name:scheduledActionName/my-5-8-scheduled-down-action",
            "ServiceNamespace": "cassandra",
            "Schedule": "cron(15 17 ? * MON-FRI *)",
            "Timezone": "Australia/Brisbane",
            "ResourceId": "keyspace/keyspace-name/table/table-name",
            "ScalableDimension": "cassandra:table:WriteCapacityUnits",
            "ScalableTargetAction": {
                "MinCapacity": 90,
                "MaxCapacity": 1500
            },
            "CreationTime": "2022-03-15T17:30:25.100000+10:00"
        },
        {
            "ScheduledActionName": "my-8-5-scheduled-action",
            "ScheduledActionARN":
 "arn:aws:autoscaling:<region>:<account>:scheduledAction:<uuid>:resource/keyspaces/
table/table-name:scheduledActionName/my-8-5-scheduled-action",
            "ServiceNamespace": "cassandra",
            "Schedule": "cron(45 8 ? * MON-FRI *)",
            "Timezone": "Australia/Brisbane",
```

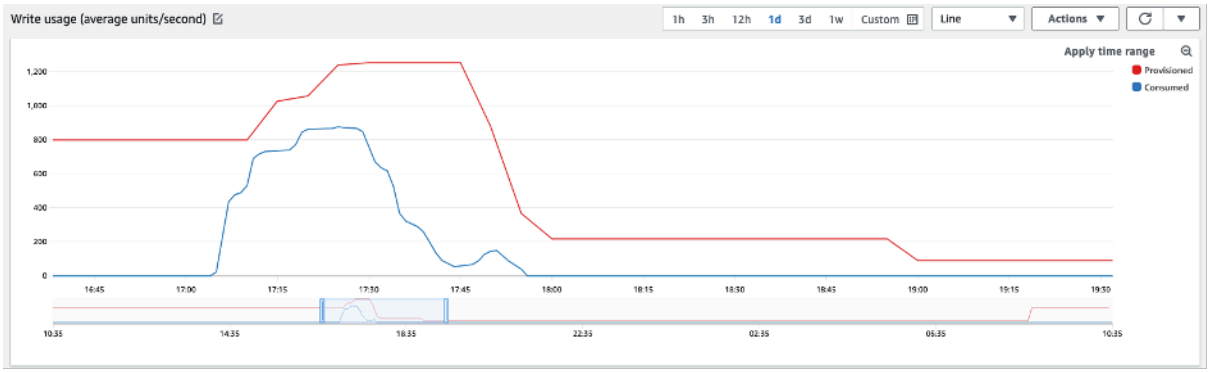
```
            "ResourceId": "keyspace/keyspace-name/table/table-name",
            "ScalableDimension": "cassandra:table:WriteCapacityUnits",
            "ScalableTargetAction": {
                "MinCapacity": 800,
                "MaxCapacity": 1500
            },
            "CreationTime": "2022-03-15T17:28:57.816000+10:00"
        }
    ]
}
```

下圖顯示一律保持 70% 目標使用率的範例工作負載。請注意，auto Scaling 規則仍在套用，而且輸送量不會減少。



放大時可以看到應用程式中，有一個尖峰觸發了 70% 的自動擴展閾值，強制啟動自動擴展，並提供資料表所需的額外容量。排定的 auto 動調整比例作業會影響最大值與最小值，您必須負責設定這些值。





如何處理具有未知模式的尖峰工作負載

在此案例中，應用程式使用使用率非常低的目標，因為您還不知道應用程式模式，而且想要確保工作負載不會遇到低容量輸送量錯誤。

請考慮改用隨需容量模式。隨需資料表非常適合流量模式不明的尖峰工作負載。使用隨需容量模式，您可以按請求支付應用程式在資料表上執行的資料讀取和寫入費用。您不需要指定預期應用程式執行多少讀取和寫入輸送量，因為 Amazon Keyspaces 可在工作負載上升或下降時立即容納工作負載。

如何處理具有連結應用程式的工作負載

在此情況中，應用程式會依賴其他系統，像是在批次處理的情況，您可能會根據應用程式邏輯中的事件，遇到流量出現大峰值。

請考慮開發自訂應用程式 auto-scaling 邏輯，以回應這些事件，您可以`TargetValues`根據您的特定需求增加資料表容量。您可以受益於 Amazon EventBridge 並使用諸如和 Step Functions 之類的 AWS 服務組合來響應您的特定應用程序需求。

識別您未使用的資源

本節概述如何定期評估未使用的資源。隨著應用程式需求的發展，您應確保沒有任何資源未使用，並且會產生不必要的 Amazon Keyspaces 成本。下述程序使用 Amazon CloudWatch 指標識別未使用的資源，並採取行動降低成本。

您可以使用來監控 Amazon Keyspaces CloudWatch，該密鑰空間將來自 Amazon Keyspaces 的原始資料收集並處理為可讀且接近即時的指標。這些統計資料會保留一段時間，以便您存取歷史資訊，並更清楚了解自己的使用率。根據預設，Amazon Keyspaces 指標資料會 CloudWatch 自動傳送到。如需詳細資訊，請參閱什麼是 Amazon CloudWatch？ 和 Amazon CloudWatch 用戶指南中的指標保留。

主題

如何識別未使用資源

若要識別未使用的表格，您可以查看 30 天內的下列 CloudWatch 指標，以瞭解特定資料表是否有任何使用中的讀取或寫入：

`ConsumedReadCapacityUnits`

在指定時段使用的讀取容量單位數目，可讓您追蹤已使用多少使用容量。您可以擷取表格的總使用讀取容量。

`ConsumedWriteCapacityUnits`

在指定時段使用的寫入容量單位數目，可讓您追蹤已使用多少使用容量。您可以擷取資料表的總耗用寫入容量。

識別未使用的資料表資源

Amazon CloudWatch 是一項監控和可觀察性服務，提供 Amazon Keyspaces 表格指標，您可以用來識別未使用的資源。 CloudWatch 您可以透過以 AWS Management Console 及透過檢視度量 AWS Command Line Interface。

AWS Command Line Interface

若要透過檢視表格量度 AWS Command Line Interface，您可以使用下列命令。

1. 首先評估資料表的讀取：

> **Note**
>
> 如果表名在您的帳戶中不是唯一的，您還必須指定密鑰空間的名稱。

```
aws cloudwatch get-metric-statistics --metric-name
```

```
ConsumedReadCapacityUnits --start-time <start-time> --end-time <end-
time> --period <period> --namespace AWS/Cassandra --statistics Sum --
dimensions Name=TableName,Value=<table-name>
```

為了避免將資料表誤認為未使用，請評估較長期間內的指標。選擇適當的開始時間和結束時間範圍，例如 30 天，以及適當的期間，例如 86400。

在傳回的資料中，任何大於 0 的總和都表示您所評估的資料表在該期間內曾經接收讀取流量。

下列結果顯示在評估期間接收讀取流量的資料表：

```
        {
            "Timestamp": "2022-08-25T19:40:00Z",
            "Sum": 36023355.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2022-08-12T19:40:00Z",
            "Sum": 38025777.5,
            "Unit": "Count"
        },
```

下列結果顯示在評估期間未接收讀取流量的資料表：

```
        {
            "Timestamp": "2022-08-01T19:50:00Z",
            "Sum": 0.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2022-08-20T19:50:00Z",
            "Sum": 0.0,
            "Unit": "Count"
        },
```

2. 接下來，評估資料表的寫入數：

```
aws cloudwatch get-metric-statistics --metric-name
ConsumedWriteCapacityUnits --start-time <start-time> --end-time <end-
time> --period <period> --namespace AWS/Cassandra --statistics Sum --
dimensions Name=TableName,Value=<table-name>
```

為了避免將資料表誤認為未使用，建議您評估較長期間內的指標。選擇適當的開始時間和結束時間範圍 (例如 30 天) 及適當的期間 (例如 86400)。

在傳回的資料中，任何大於 0 的總和都表示您所評估的資料表在該期間內曾經接收讀取流量。

下列結果顯示在評估期間接收寫入流量的資料表：

```
        {
            "Timestamp": "2022-08-19T20:15:00Z",
            "Sum": 41014457.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2022-08-18T20:15:00Z",
            "Sum": 40048531.0,
            "Unit": "Count"
        },
```

下列結果顯示在評估期間曾接收寫入流量的資料表：

```
        {
            "Timestamp": "2022-07-31T20:15:00Z",
            "Sum": 0.0,
            "Unit": "Count"
        },
        {
            "Timestamp": "2022-08-19T20:15:00Z",
            "Sum": 0.0,
            "Unit": "Count"
        },
```

AWS Management Console

下列步驟可讓您透過評估資源使用率 AWS Management Console。

1. 登入 AWS Management Console 並瀏覽至 CloudWatch 服務頁面，網址為 https://console.aws.amazon.com/cloudwatch/。如有必要，請 AWS 區域 在主控台右上方選取適當的選項。
2. 在左側導覽列上，找出「度量」區段，然後選擇「所有量度」。

3. 上述動作會開啟包含兩個面板的管控面板。在頂部面板中，您可以看到當前圖形的指標。在底部，您可以選擇可用於繪製圖形的指標。在底部面板中選擇 Amazon Keyspaces。
4. 在 Amazon Keyspace 指標選取面板中，選擇「表格指標」類別，以顯示目前區域中表格的指標。
5. 向下捲動選單來識別您的表格名稱，然後選擇指標`ConsumedReadCapacityUnits`和`ConsumedWriteCapacityUnits`表格。
6. 選擇「圖表化量度 (2)」標籤，然後將「統計值」欄調整為「總和」。
7. 若要避免錯誤地將表格識別為未使用，請在較長時間內評估表格測量結果。在圖形面板頂端，選擇適當的時間範圍 (例如 1 個月) 以評估表格。選擇「自訂」，在下拉式選單中選擇「1 個月」，然後選擇「套用」。
8. 評估資料表的圖表化指標，判斷是否已使用該資料表。如果指標超過 0，就表示在評估期間內已使用該資料表。讀取和寫入均為 0 的平面圖表示資料表未使用。

清除未使用的資料表資源

如果找出未使用的資料表資源，您可以透過下列方式降低其持續產生的成本。

> **Note**
>
> 如果找出未使用的資料表，但仍希望保留以備日後需要時可供存取，請考慮將其轉換為隨需模式。否則，您可以考慮刪除表格。

容量模式

Amazon Keyspaces 會針對讀取、寫入和存放資料在 Amazon Keyspaces 表格中收取費用。

Amazon Keyspaces space 具有兩種容量模式，其中提供用於處理表格上讀取和寫入的特定計費選項：隨需和佈建。讀取/寫入容量模式可控制您變更讀取與寫入傳輸量以及管理容量的方式。

若為隨需模式資料表，不需要指定您預期應用程式將進行的讀取和寫入輸送量。Amazon Keyspaces 會針對應用程式在資料表上執行的讀取和寫入 (以讀取請求單位和寫入請求單位計算) 向您收取費用。如果您的表格上沒有任何活動，您不需要支付輸送量費用，但仍會產生儲存費用。

刪除資料表

如果您發現未使用的資料表並想要刪除它，請考慮先進行備份或匯出資料。

通過備份 AWS Backup 可以利用冷存儲分層，進一步降低成本。有關如何使用生命週期將備份移至冷存儲的詳細資訊，請參閱管理備份計劃文件。

備份資料表之後，即可選擇透過 AWS Management Console 或 AWS Command Line Interface加以刪除。

清除未使用的 point-in-time 復原 (PITR) 備份

Amazon Keyspaces 提供 P oint-in-time 復原功能，可提供 35 天的連續備份，以協助您防止意外寫入或刪除。PITR 備份有相關的成本。

請參閱的文件，瞭Point-in-time 回收解您的資料表是否已啟用可能不再需要的備份。

評估您的資料表用量模式

本節提供如何評估您是否有效地使用 Amazon Keyspaces 表的概觀。某些使用模式對於 Amazon Keyspace 來說並非最佳化，而且從效能和成本的角度來看，它們允許優化的空間。

主題

- 執行較少高度一致性讀取操作
- 啟用存留時間 (TTL)

執行較少高度一致性讀取操作

Amazon Keyspaces 可讓您根據每個請求設定讀取一致性。根據預設，讀取請求最終會保持一致。最終一致性讀取費用為 0.5 RCU，最多可達 4 KB 的資料。

分散式工作負載的多數部分都具有彈性，可以容忍最終一致性。但是，有些存取模式要求高度一致性讀取。高度一致的讀取費用是以 1 RCU 計費，最多可達 4 KB 的資料，實際上是讀取成本的兩倍。Amazon Keyspaces 為您提供在同一個表格上使用這兩種一致性模型的靈活性。

您可以評估工作負載和應用程式的程式碼，確定是否僅在需要時使用高度一致性讀取。

啟用存留時間 (TTL)

上線時間 (TTL) 可協助您簡化應用程式邏輯，並自動將資料表中的資料過期，將儲存價格最佳化。系統會根據您設定的「存留時間」值，自動從表格中刪除不再需要的資料。

評估您是否具有適當大小的佈建容量

本節提供如何評估 Amazon Keyspaces 表格上是否有適當大小的佈建概觀。隨著工作負載的發展，您應該適當地修改操作程序，尤其是在佈建模式下設定 Amazon Keyspaces 表格，並且有過度佈建或佈建表格不足的風險時。

本節中描述的程序需要統計資訊，這些資訊應從支援生產應用程式的 Amazon Keyspace 表格擷取。若要瞭解您的應用程式行為，您應該定義足以擷取應用程式資料季節性的時間段。例如，若應用程式顯示每週模式，三週時間便應足夠來分析應用程式輸送量需求。

若您不知道從何下手，請使用至少一個月的資料用量進行以下計算。

評估容量時，對於 Amazon Keyspaces 表格，您可以獨立設定讀取容量單位 (RCU) 和寫入容量單位 (WCU)。

主題

- 如何從 Amazon Keyspaces 表中檢索消費指標
- 如何識別佈建不足的 Amazon Keyspaces 表
- 如何識別過度佈建的 Amazon Keyspaces 表

如何從 Amazon Keyspaces 表中檢索消費指標

若要評估表格容量，請監督下列 CloudWatch 測量結果，然後選取適當的維度來擷取表格資訊：

讀取容量單位	寫入容量單位
`ConsumedReadCapacityUnits`	`ConsumedWriteCapacityUnits`
`ProvisionedReadCapacityUnits`	`ProvisionedWriteCapacityUnits`
`ReadThrottleEvents`	`WriteThrottleEvents`

您可以透過 AWS CLI 或執行此操作 AWS Management Console。

AWS CLI

在擷取表格使用量度之前，您需要先使用 CloudWatch API 擷取一些歷史資料點。

首先要建立兩個檔案：write-calc.json 和 read-calc.json。這些檔案代表表格的計算。您需要更新某些欄位，如下表所示，以符合您的環境。

> **Note**
>
> 如果表名在您的帳戶中不是唯一的，您還必須指定密鑰空間的名稱。

欄位名稱	定義	範例
<table-name>	您正在分析的資料表名稱	SampleTable
<period>	您用來評估使用率目標的時間期間 (以秒為單位)	針對 1 小時的時間，您應指定：3600
<start-time>	評估間隔的開始時間，以 ISO8601 格式顯示	2022-02-21T23:00:00
<end-time>	評估間隔的結束時間，以 ISO8601 格式顯示	2022-02-22T06:00:00

寫入計算檔案會擷取在指定日期範圍內佈建和使用的 WCU 數目。它也會產生可用於分析的使用率百分比。write-calc.json檔案的完整內容應如下列範例所示。

```
{
  "MetricDataQueries": [
    {
      "Id": "provisionedWCU",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/Cassandra",
          "MetricName": "ProvisionedWriteCapacityUnits",
          "Dimensions": [
            {
              "Name": "TableName",
              "Value": "<table-name>"
            }
          ]
        },
        "Period": <period>,
```

```
        "Stat": "Average"
      },
      "Label": "Provisioned",
      "ReturnData": false
    },
    {
      "Id": "consumedWCU",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/Cassandra",
          "MetricName": "ConsumedWriteCapacityUnits",
          "Dimensions": [
            {
              "Name": "TableName",
              "Value": "<table-name>""
            }
          ]
        },
        "Period": <period>,
        "Stat": "Sum"
      },
      "Label": "",
      "ReturnData": false
    },
    {
      "Id": "m1",
      "Expression": "consumedWCU/PERIOD(consumedWCU)",
      "Label": "Consumed WCUs",
      "ReturnData": false
    },
    {
      "Id": "utilizationPercentage",
      "Expression": "100*(m1/provisionedWCU)",
      "Label": "Utilization Percentage",
      "ReturnData": true
    }
  ],
  "StartTime": "<start-time>",
  "EndTime": "<end-time>",
  "ScanBy": "TimestampDescending",
  "MaxDatapoints": 24
}
```

讀取的計算檔案使用類似的量度。此檔案會擷取在指定日期範圍內佈建和使用的 RCU 數目。read-calc.json檔案的內容應如此範例所示。

```
{
  "MetricDataQueries": [
    {
      "Id": "provisionedRCU",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/Cassandra",
          "MetricName": "ProvisionedReadCapacityUnits",
          "Dimensions": [
            {
              "Name": "TableName",
              "Value": "<table-name>"
            }
          ]
        },
        "Period": <period>,
        "Stat": "Average"
      },
      "Label": "Provisioned",
      "ReturnData": false
    },
    {
      "Id": "consumedRCU",
      "MetricStat": {
        "Metric": {
          "Namespace": "AWS/Cassandra",
          "MetricName": "ConsumedReadCapacityUnits",
          "Dimensions": [
            {
              "Name": "TableName",
              "Value": "<table-name>"
            }
          ]
        },
        "Period": <period>,
        "Stat": "Sum"
      },
      "Label": "",
      "ReturnData": false
    },
```

```
    {
      "Id": "m1",
      "Expression": "consumedRCU/PERIOD(consumedRCU)",
      "Label": "Consumed RCUs",
      "ReturnData": false
    },
    {
      "Id": "utilizationPercentage",
      "Expression": "100*(m1/provisionedRCU)",
      "Label": "Utilization Percentage",
      "ReturnData": true
    }
  ],
  "StartTime": "<start-time>",
  "EndTime": "<end-time>",
  "ScanBy": "TimestampDescending",
  "MaxDatapoints": 24
}
```

建立檔案後，您就可以開始擷取使用率資料。

1. 若要擷取寫入使用率資料，請執行下列命令：

```
aws cloudwatch get-metric-data --cli-input-json file://write-calc.json
```

2. 若要擷取讀取使用率資料，請執行下列命令：

```
aws cloudwatch get-metric-data --cli-input-json file://read-calc.json
```

這兩個查詢的結果都是一系列 JSON 格式的資料點，可用於分析。結果取決於您指定的資料點數量、期間以及您自己的特定工作負載資料。它可能看起來像下面的例子。

```
{
    "MetricDataResults": [
        {
            "Id": "utilizationPercentage",
            "Label": "Utilization Percentage",
            "Timestamps": [
                "2022-02-22T05:00:00+00:00",
                "2022-02-22T04:00:00+00:00",
                "2022-02-22T03:00:00+00:00",
```

```
                "2022-02-22T02:00:00+00:00",
                "2022-02-22T01:00:00+00:00",
                "2022-02-22T00:00:00+00:00",
                "2022-02-21T23:00:00+00:00"
            ],
            "Values": [
                91.55364583333333,
                55.066631944444445,
                2.6114930555555556,
                24.9496875,
                40.94725694444445,
                25.61819444444444,
                0.0
            ],
            "StatusCode": "Complete"
        }
    ],
    "Messages": []
}
```

> **Note**
>
> 如果您指定了短週期和較長的時間範圍，則可能需要修改該`MaxDatapoints`值，該值在指令碼中預設設定為 24。這代表每小時產生一個資料點，每天 24 個資料點。

AWS Management Console

1. 登入 AWS Management Console 並瀏覽至「開始使用」中的 CloudWatch 服務頁面 AWS Management Console。 AWS 區域 如有必要，請選取適當的。
2. 找出左側導覽列上的「量度」區段，然後選擇「所有量度」。
3. 這會開啟具有兩個面板的管控面板。頂端面板會顯示圖形，底部面板包含您要繪製圖形的量度。選擇 Amazon Keyspaces 面板。
4. 從子面板選擇「表格測量結果」類別。這會顯示您目前的表格 AWS 區域。
5. 向下捲動選單找出您的資料表名稱，然後選取寫入操作指標：`ConsumedWriteCapacityUnits` 和 `ProvisionedWriteCapacityUnits`。

> **Note**
>
> 此範例討論寫入操作指標，但您也可以用這些步驟來繪製讀取操作指標的圖形。

6. 選取 Graphed metrics (2) (圖表化指標(2)) 標籤，修改公式。依預設， CloudWatch 選擇圖表的統計函數「平均值」。
7. 同時選取兩個圖形化指標 (左側的核取方塊) 後，選取選單 Add math (新增數學)，再選取 Common (一般) 及 Percentage (百分比) 函數。重複該過程兩次。

   第一次選擇百分比功能。

   第二次選擇百分比功能。
8. 此時，選單底部應有四個指標。接著進行 `ConsumedWriteCapacityUnits` 計算。為了保持一致，您需要將名稱與您在 AWS CLI 部分中使用的名稱進行匹配。按一下 m1 ID，並將此值變更為 consumedWCU。
9. 將統計資料從 Average (平均值) 變更為 Sum (總和)。此動作會自動建立另一個稱為「異常 _ 偵測 _ 頻帶」的量度。針對此程序的範圍，您可以移除新產生之 ad1 測量結果上的核取方塊來忽略此問題。
10. 重複步驟 8，將 m2 ID 重新命名為 provisionedWCU。將統計資料設定保留為 Average (平均值)。
11. 選擇「運算式 1」標籤，並將值更新為 m1，並將標籤更新為「已耗用的 WCU」。

> **Note**
>
> 請確認您只選取 m1 (左側的核取方塊) 及 provisionedWCU，適當地視覺化資料。按一下 Details (詳細資料)，並將公式變更為 consumedWCU/PERIOD(consumedWCU)，以更新公式。此步驟也可能會產生另一個「異常 _ 偵測 _BAND」量度，但對於此程序的範圍，您可以忽略它。

12. 您現在應該有兩個圖形：一個表示表格上已佈建的 WCU，另一個表示已耗用的 WCU。
13. 選取 Expression2 圖形 (e2) 以更新百分比公式。將標籤和 ID 重新命名為 utilizationPercentage。將公式重新命名，以符合 100*(m1/provisionedWCU)。
14. 從除「使用率百分比」以外的所有指標中移除核取方塊，以視覺化您的使用率模式。預設間隔設定為 1 分鐘，但可以根據需要進行修改。

您取得的結果取決於工作負載中的實際資料。使用率超過 100% 的間隔容易發生低輸送量容量錯誤事件。Amazon Keyspaces 供大量容量，但是一旦突增容量用盡，任何超過 100% 的項目都會遇到低輸送量容量錯誤事件。

如何識別佈建不足的 Amazon Keyspaces 表

對於大多數工作負載，當表格持續耗用其佈建容量的 80% 以上時，就會被視為佈建不足。

爆發容量是 Amazon Keyspaces 功能，可讓客戶暫時消耗比原先佈建的 RCU/WCU 多 (超過針對表格定義的每秒佈建輸送量)。高載容量是為了吸收由於特殊事件或使用量尖峰而突增的流量。此成組分解容量有限，如需詳細資訊，請參閱the section called “高載容量”。一旦未使用的 RCU 和 WCU 耗盡，如果您嘗試消耗的容量超過佈建的容量，就可能會遇到低容量輸送量錯誤事件。當您的應用程式流量接近 80% 的使用率時，發生低容量輸送量錯誤事件的風險會大幅提高。

80% 使用率規則會因資料的季節性和流量成長而有所不同。請考量下列情況：

- 如果您的流量在過去 12 個月內穩定保持約 90% 的使用率，那麼您的資料表容量剛好
- 如果您的應用程式流量在 3 個月內以每月 8% 的速度增加，您將達到 100%
- 如果您的應用程式流量在略多於 4 個月的期間以 5% 的速度增加，您仍會達到 100%

上述查詢結果能讓您得知使用率的情況。您可以使用這些結果做為指引，進一步評估其他指標，以協助您根據需要增加資料表容量 (例如：每月或每週成長率)。與您的營運團隊合作，為工作負載和資料表定義合理的百分比。

在某些特殊情況下，當您每天或每週進行分析時，資料會出現偏斜。例如，如果季節性應用程式在工作時間內使用量激增 (但在工作時間以外降到幾乎為零)，您可以從排程應用程式 auto-scaling 的時間中獲益，您可以在其中指定一天中的小時 (以及星期幾) 來增加佈建容量，以及何時減少。如果您的季節性不太明顯，您還可以從 Amazon Keyspace 表 auto-scaling 配置中受益，而不是瞄準更高的容量來滿足繁忙時間。

如何識別過度佈建的 Amazon Keyspaces 表

從以上指令碼獲得的查詢結果提供了執行部分初始分析所需的資料點。若您的資料集在數個間隔內顯示的使用率低於 20%，表示資料表可能過度佈建。若要進一步判斷是否需減少 WCU 和 RCU 數量，您應重新檢視間隔中的其他讀數。

當您的表格包含數個低使用率間隔時，您可以從使用應用程式 Auto Scaling 原則中受益，方法是排定應用程式 Auto Scaling，或僅針對以使用率為基礎的表格設定預設的 Application Auto Scaling 政策。

如果您的工作負載使用率與高節流率 (間隔中的 Max (ThrottleEvents) /Min ()) 較低，則當您的工作負載非常尖銳，而流量在特定日期 (或一天中的時間) 顯著增加，但其他情況一直很低時，就可能會發生這種情況。ThrottleEvents在這些情況下，使用排程的應用 Application Auto Scaling 可能會有所幫助。

使用 NoSQL Workbench 與 Amazon Keyspaces (適用於 Apache Cassandra

NoSQL Workbench 是用戶端應用程式，可協助您更輕鬆地設計和視覺化 Amazon Keyspaces 的非關聯式資料模型。NoSQL Workbench 客戶端可用於 Windows 和 Linux。

設計資料模型並自動建立資源

NoSQL 工作台為您提供了一個 point-and-click 界面來設計和創建亞馬遜 Keyspaces 數據模型。您可以通過定義密鑰空間，表格和列輕鬆地從頭開始創建新的數據模型。您也可以匯入現有的資料模型並進行修改 (例如新增、編輯或移除欄)，以適應新應用程式的資料模型。NoSQL 工作台然後，您可以提交數據模型亞馬遜 Keyspaces 或 Apache 卡桑德拉，並自動創建密鑰空間和表。若要瞭解如何建置資料模型，請參閱the section called “資料模型建立工具”。

視覺化資料模型

使用 NoSQL Workbench，您可以視覺化您的資料模型，以協助確保資料模型可以支援應用程式的查詢和存取模式。您也可以儲存和匯出各種格式的資料模型，以供協同作業、文件和簡報使用。如需詳細資訊，請參閱the section called “資料視覺化工具”。

主題

下載 NoSQL Workbench

請依照這些說明下載並安裝 NoSQL Workbench。

下載並安裝 NoSQL Workbench

1. 使用下列連結之一，免費下載 NoSQL Workbench。

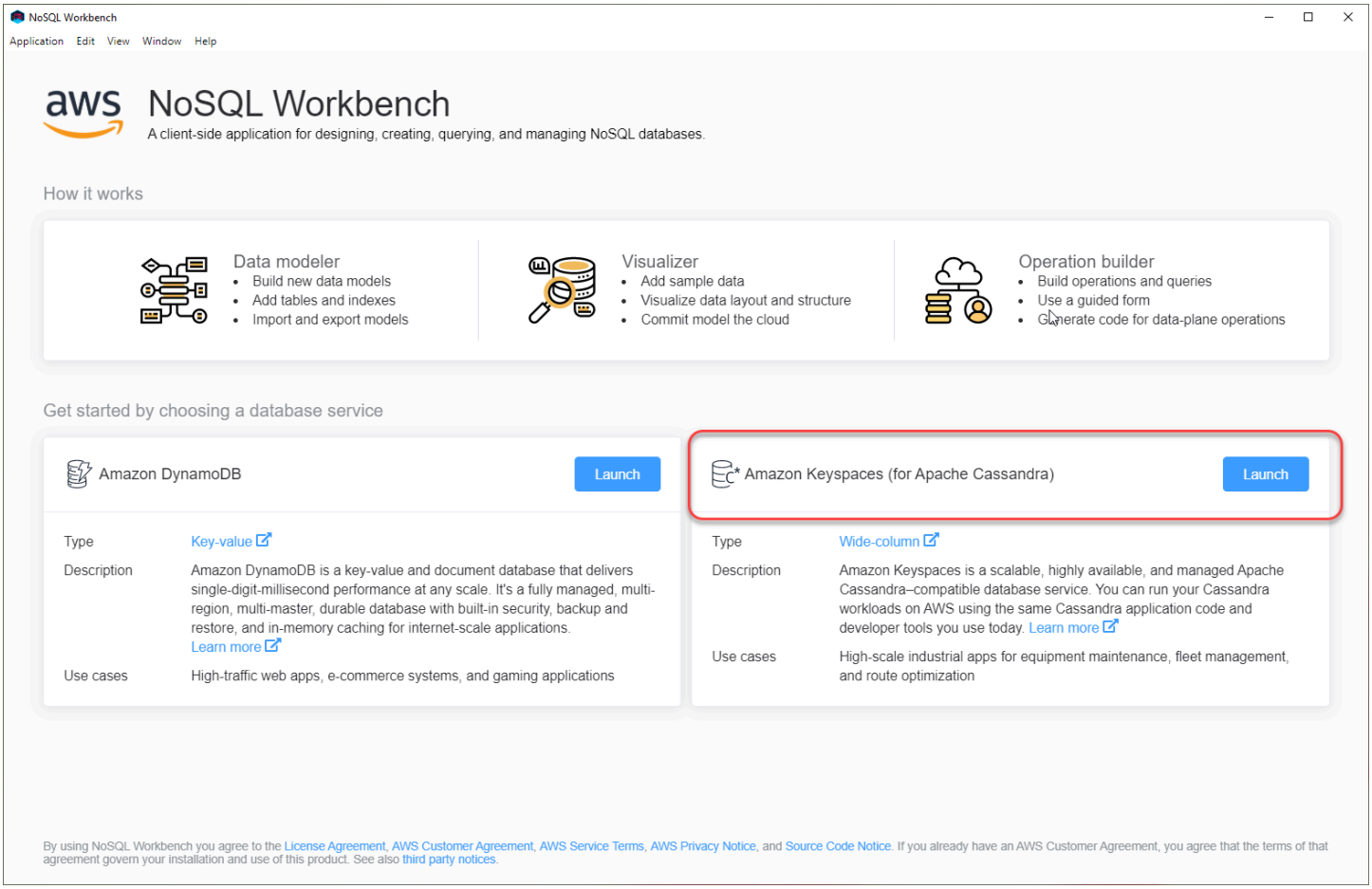
作業系統	下載連結
macOS	適用於 macOS 的下載
Linux*	下載 Linux 版本
Windows	適用於 Windows 的下載

* NoSQL Workbench 支援 Ubuntu 12.04、Fedora 21、Debian 8 或 Linux 發行版本的任何更新版本。

2. 下載完成後，啟動應用程序並按照屏幕上的說明完成安裝。

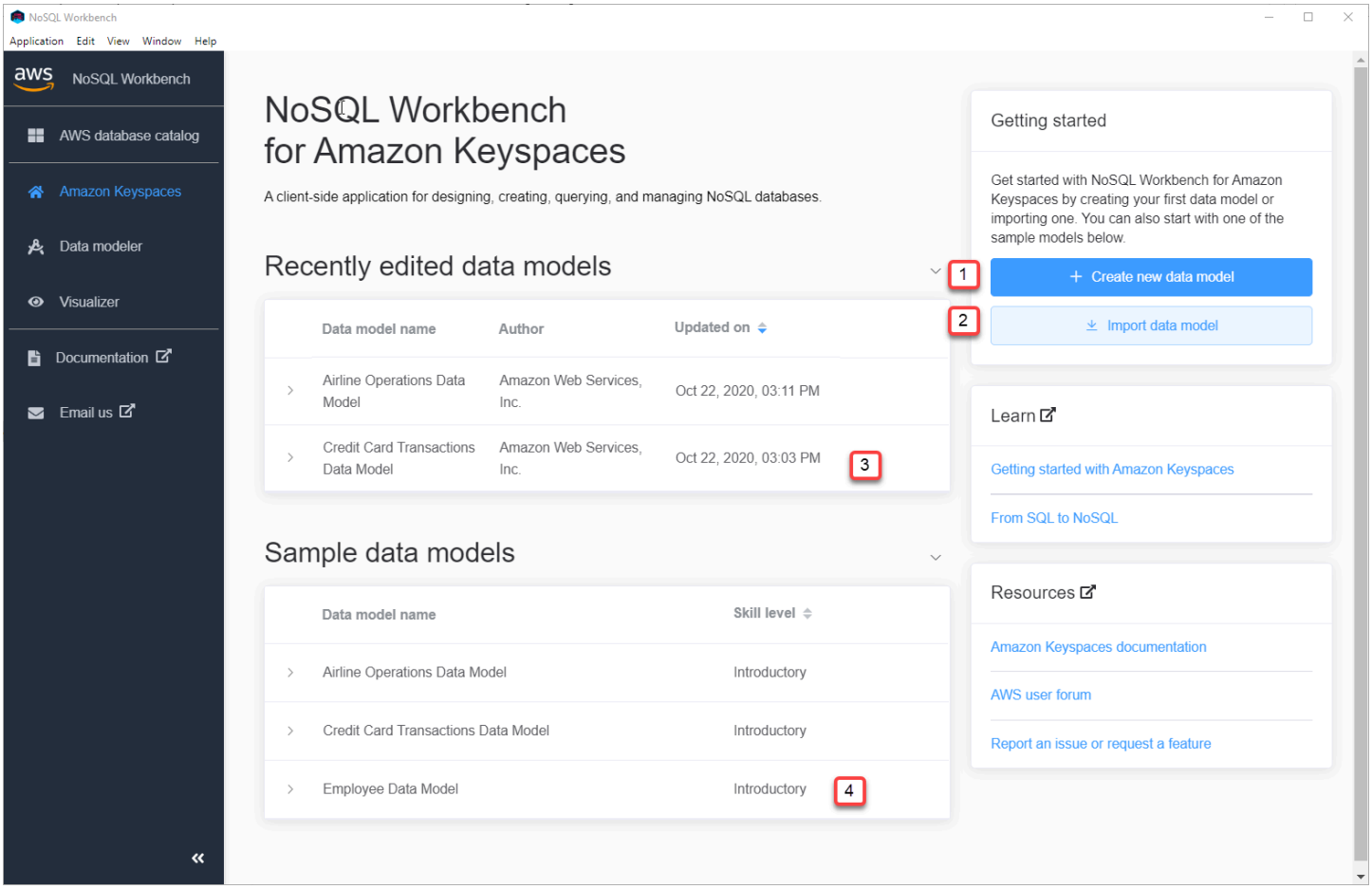
開始使用 NoSQL Workbench

若要開始使用 NoSQL 工作台，請在 NoSQL 工作台的 [資料庫目錄] 頁面上選擇 Amazon Keyspaces，然後選擇 [啟動]。



這將打開 NoSQL 工作台主頁亞馬遜 Keyspaces，您可以在其中開始使用以下選項：

1. 建立新的資料模型。
2. 以 JSON 格式匯入現有的資料模型。
3. 開啟最近編輯的資料模型。
4. 開啟其中一個可用的範例模型。



每個選項都會開啟 NoSQL 工作台資料塑模器。若要繼續建立新資料模型，請參閱the section called "建立資料模型"。若要編輯現有資料模型，請參閱the section called "編輯資料模型"。

如何建立資料模型

您可以使用 NoSQL Workbench 資料建模器，根據應用程式的資料存取模式來設計新的資料模型。您可以使用資料建模器來設計新的資料模型，或匯入和修改使用 NoSQL Workbench 建立的現有資料模型。資料建模工具還包括一些範例資料模型，可協助您開始使用資料模型。

主題

- 使用 NoSQL Workbench 建立新的資料模型
- 使用 NoSQL Workbench 編輯現有的資料模型

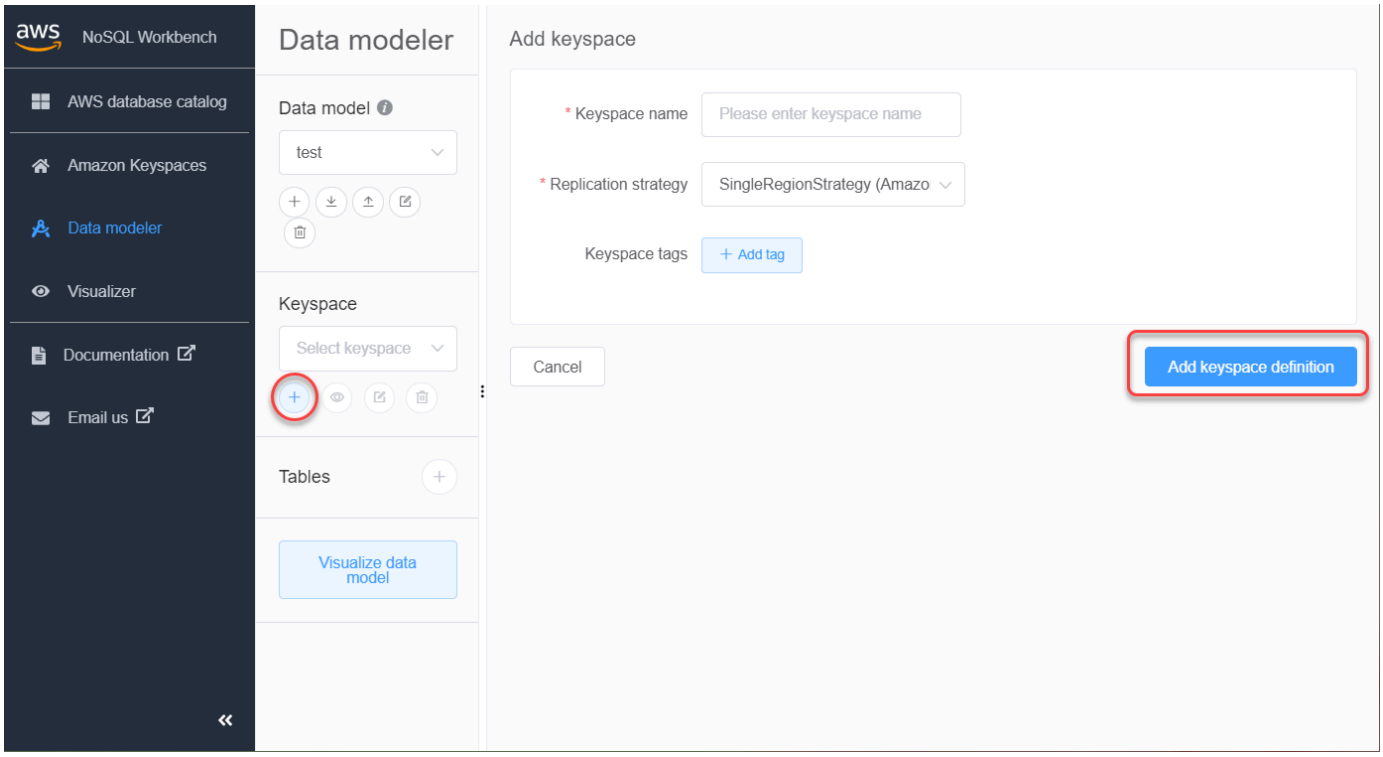
使用 NoSQL Workbench 建立新的資料模型

若要為 Amazon Keyspaces 建立新的資料模型，您可以使用 NoSQL 工作台資料建立模型工具來建立金鑰空間、表格和欄。依照下列步驟建立新的資料模型。

1. 要創建一個新的密鑰空間，選擇密鑰空間下的加號。

   在此步驟中，選擇下列屬性和設定。

   - 密鑰空間名稱 — 輸入新密鑰空間的名稱。
   - 複製策略 — 選擇金鑰空間的複製策略。Amazon Keyspaces 使用在多SingleRegionStrategy個AWS可用區域中自動複寫資料三次。如果你打算提交數據模型到 Apache 卡桑德拉集群，你可以選擇SimpleStrategy或NetworkTopologyStrategy。
   - Keyspaces 標籤 — 您可以用來透過不同方式分類您的資源，例如依據目的、擁有者、環境或其他條件。若要進一步了解 Amazon Keyspaces 資源的標籤，請參閱the section called “使用標籤”。
2. 選擇新增金鑰空間定義以建立金鑰空間。

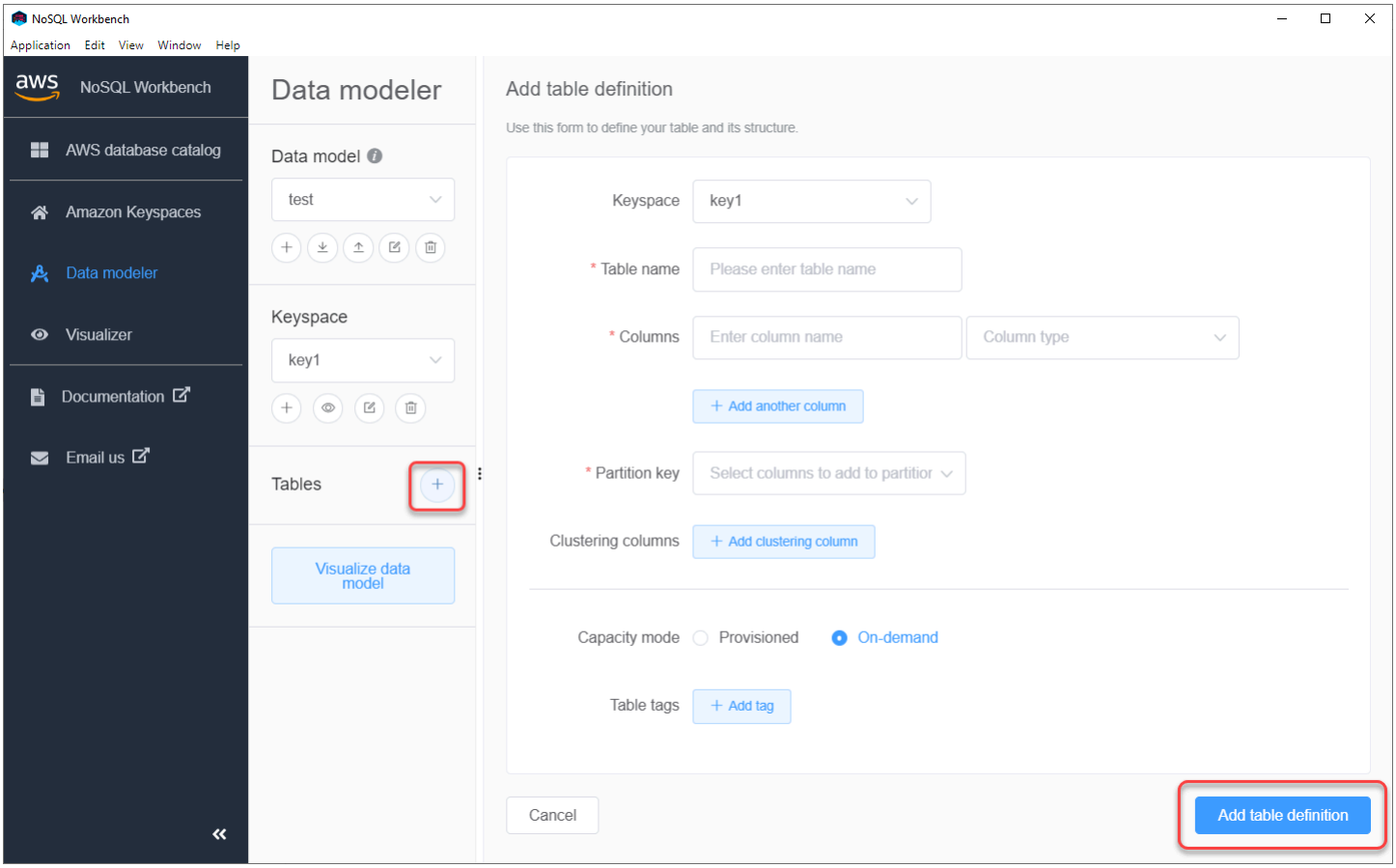


3. 若要建立新資料表，請選擇 [表格] 旁邊的加號。在此步驟中，您可以定義下列屬性和設定。

   - 表格名稱 — 新表格的名稱。
   - 欄 — 新增欄名稱並選擇資料類型。針對結構定義中的每一欄重複這些步驟。

- 分割索引鍵 — 選擇分割索引鍵的資料欄。
- 叢集資料行 — 選擇叢集資料行 (選擇性)。
- 容量模型 — 選擇資料表的讀取/寫入容量模型。您可以選擇佈建或隨需容量。若要進一步了解容量模型，請參the section called “讀/寫容量模式”。
- 資源標籤 — 您可以用來透過不同方式分類您的資源，例如依據目的、擁有者、環境或其他條件。若要進一步了解 Amazon Keyspaces 資源的標籤，請參閱the section called “使用標籤”。

4. 選擇新增資料表定義以建立新表格。
5. 重複這些步驟以建立其他表格。
6. 繼續the section called “視覺化資料模型”以視覺化方式呈現您建立的資料模型。



使用 NoSQL Workbench 編輯現有的資料模型

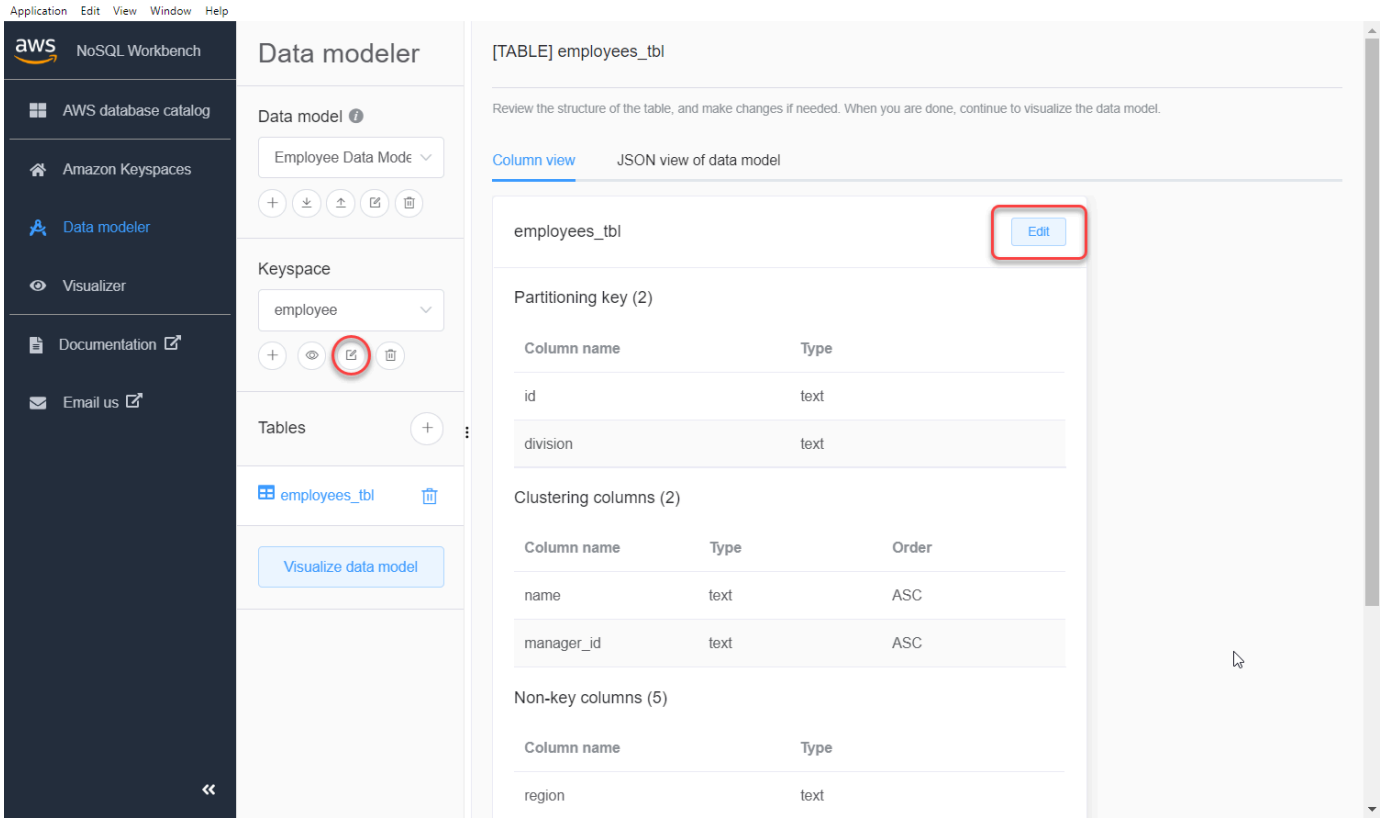
使用 NoSQL 工作台資料建模器，您可以編輯 Amazon Keyspaces 中的現有資料模型。這些資料模型可以是從檔案匯入的資料模型、提供的範例資料模型或您先前建立的資料模型。

1. 要編輯密鑰空間，請在密鑰空間下選擇編輯符號。

在此步驟中，您可以編輯下列屬性和設定。

- 密鑰空間名稱 — 輸入新密鑰空間的名稱。
- 複製策略 — 選擇金鑰空間的複製策略。Amazon Keyspaces 使用在多SingleRegionStrategy個AWS可用區域中自動複寫資料三次。如果你打算提交數據模型到 Apache 卡桑德拉集群，你可以選擇SimpleStrategy或NetworkTopologyStrategy。
- Keyspaces 標籤 — 您可以用來透過不同方式分類您的資源，例如依據目的、擁有者、環境或其他條件。若要進一步了解 Amazon Keyspaces 資源的標籤，請參閱the section called “使用標籤”。

2. 選擇 [儲存編輯] 以更新金鑰空間。



3. 若要編輯資料表，請選擇資料表名稱。在此步驟中，您可以更新下列屬性和設定。

- 表格名稱 — 新表格的名稱。
- 欄 — 新增欄名稱並選擇資料類型。針對結構定義中的每一欄重複這些步驟。
- 分割索引鍵 — 選擇分割索引鍵的資料欄。
- 叢集資料行 — 選擇叢集資料行 (選擇性)。

- 容量模型 — 選擇資料表的讀取/寫入容量模型。您可以選擇佈建或隨需容量。若要進一步了解容量模型，請參the section called “讀/寫容量模式”。
- 資源標籤 — 您可以用來透過不同方式分類您的資源，例如依據目的、擁有者、環境或其他條件。若要進一步了解 Amazon Keyspaces 資源的標籤，請參閱the section called “使用標籤”。

4. 選擇 [儲存編輯] 以更新表格。
5. 繼續the section called “視覺化資料模型”以視覺化方式顯示您更新的資料模型。

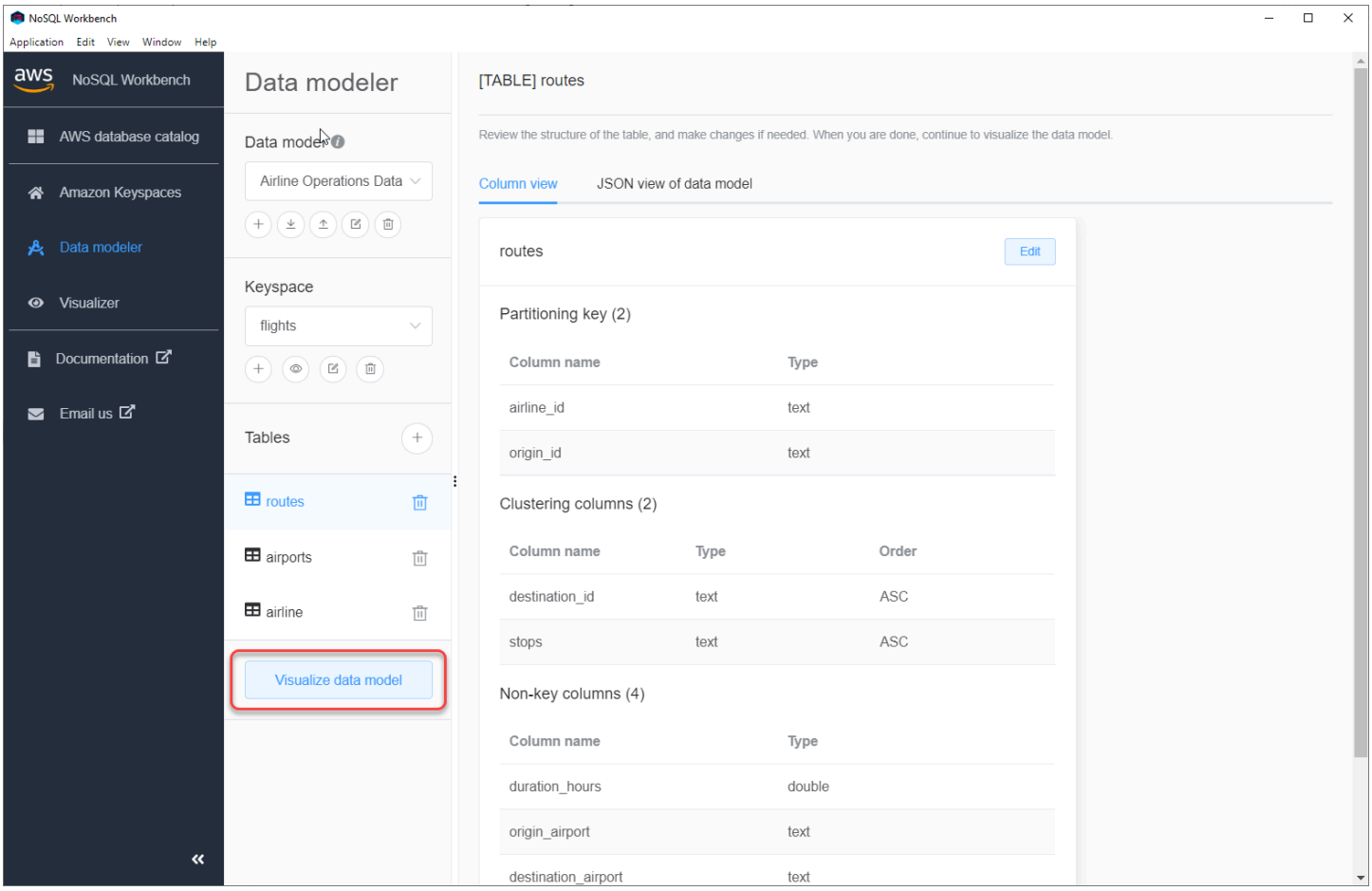
如何視覺化資料模型

使用 NoSQL Workbench，您可以視覺化您的資料模型，以協助確保資料模型可以支援應用程式的查詢和存取模式。您也可以儲存和匯出各種格式的資料模型，以供協同作業、文件和簡報使用。

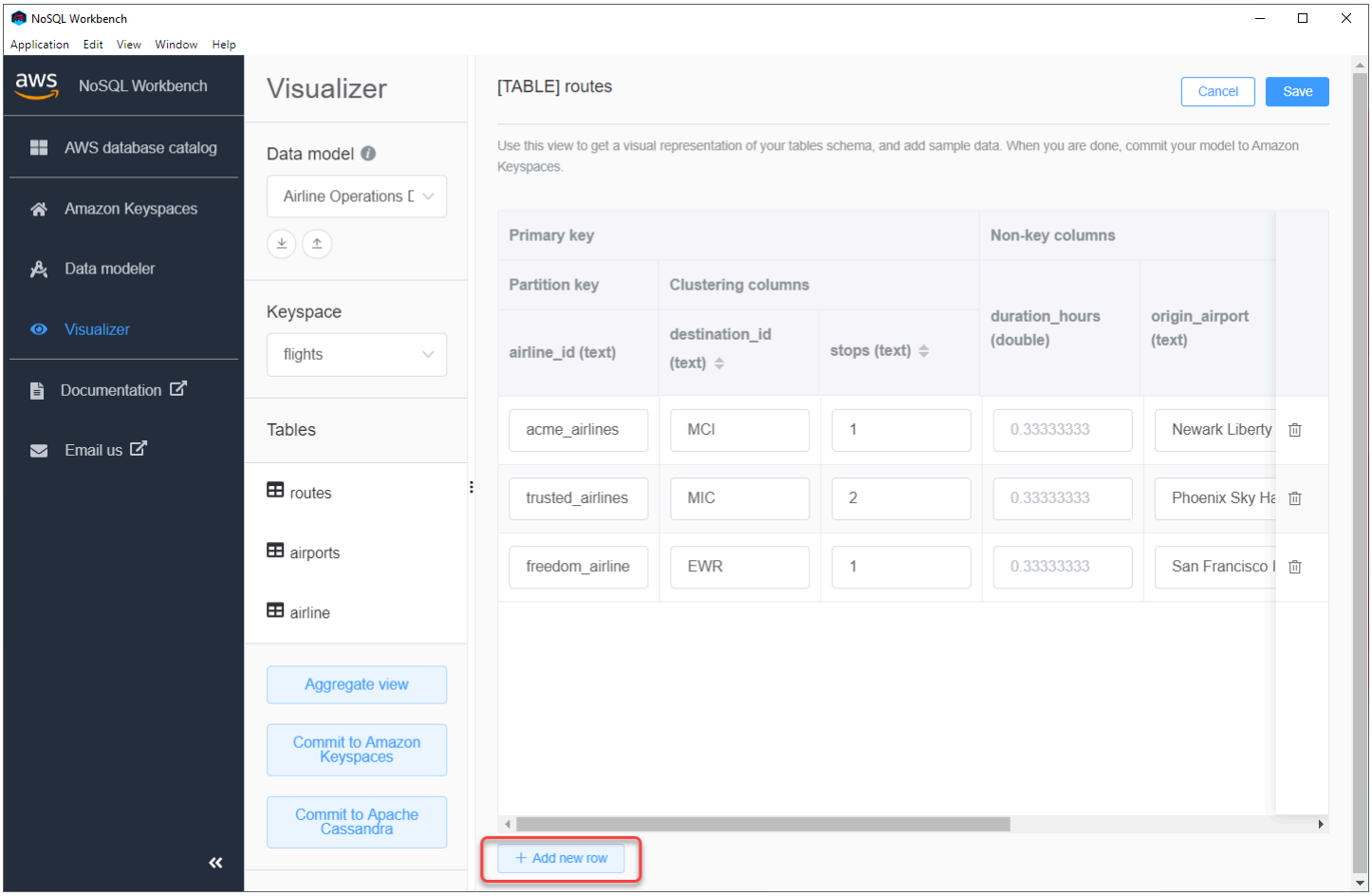
建立新的資料模型或編輯現有的資料模型之後，您可以視覺化模型。

使用 NoSQL 工作台視覺化資料模型

在資料建模工具中完成資料模型後，請選擇「視覺化資料模型」。

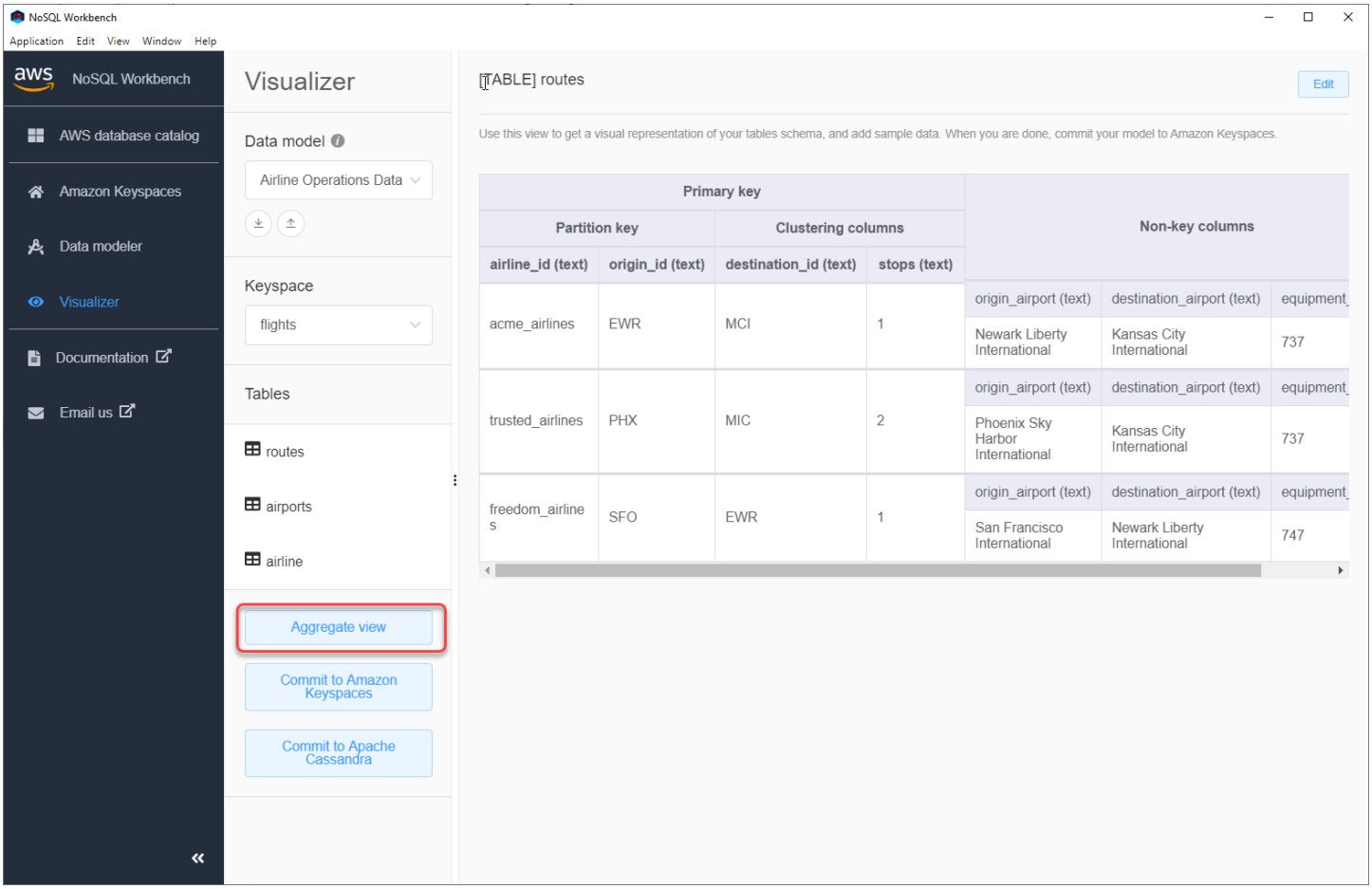


這將帶您到 NoSQL 工作台中的數據可視化視窗。資料視覺化檢視提供資料表結構定義的視覺化表示，並可讓您新增範例資料。若要將範例資料新增至表格，請從模型中選擇表格，然後選擇 [編輯]。若要新增資料列，請選擇畫面底部的 [新增列]。完成後，請選擇 Save (儲存)。



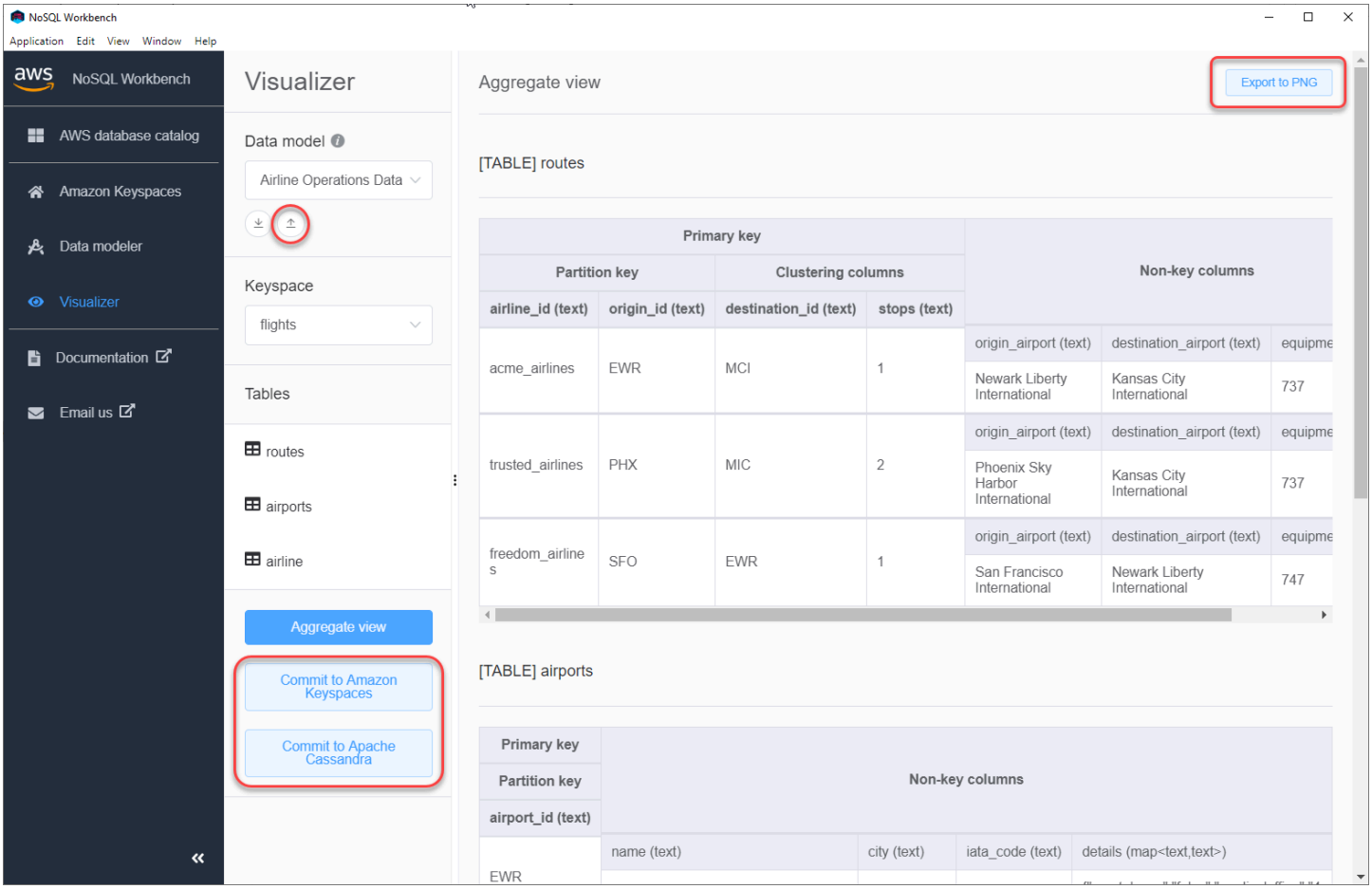
彙總檢視

確認資料表的結構定義之後，您可以彙總資料模型視覺效果。



彙總資料模型的檢視之後，您可以將檢視匯出為 PNG 檔案。要將數據模型導出到 JSON 文件，請選擇數據模型名稱下的上傳符號。

> **Note**
>
> 您可以在設計過程中隨時以 JSON 格式匯出資料模型。



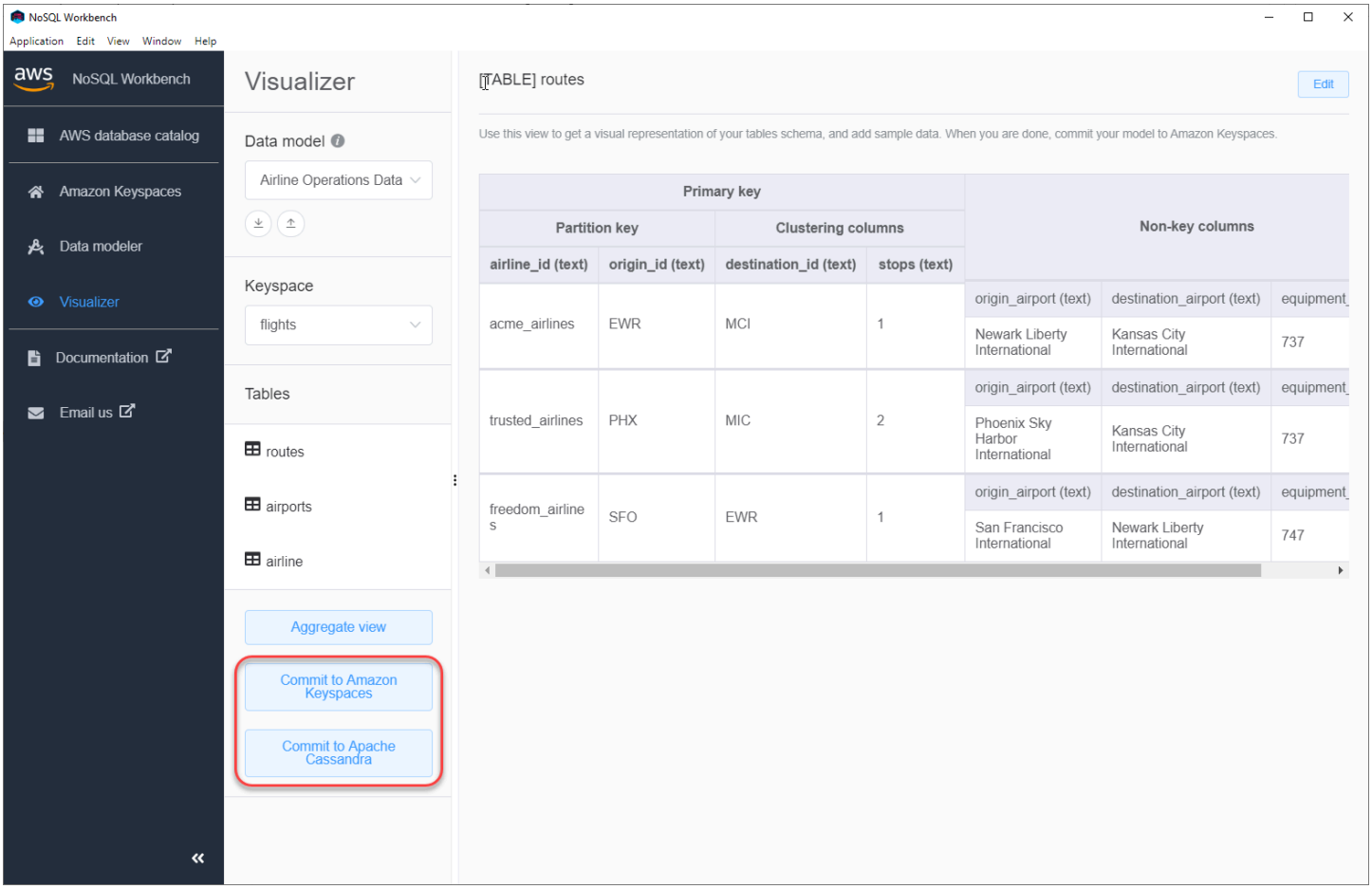
您可以使用下列選項來提交變更：

- 提交到 Amazon Keyspaces
- 提交到一個阿帕奇卡桑德拉集群

若要進一步瞭解如何提交變更，請參閱the section called “遞交資料模型”。

如何提交數據模型亞馬遜密鑰空間和阿帕奇卡桑德拉

本節向您展示如何將完整的數據模型提交到亞馬遜密鑰空間和 Apache 卡桑德拉集群。這個過程會根據您在數據模型中定義的設置自動創建密鑰空間和表的服務器端資源。



主題

- 開始之前
- 使用服務特定登入資料連線至 Amazon 金鑰空間
- 使用 AWS Identity and Access Management (IAM) 登入資料連接到亞馬遜金鑰空間
- 使用已儲存的連線
- 致力於阿帕奇卡桑德拉

開始之前

Amazon 金鑰空間需要使用傳輸層安全性 (TLS) 來協助保護與用戶端的連線安全。若要使用 TLS 連線到 Amazon 金鑰空間，您必須先完成下列任務，才能開始。

- 使用下列指令下載 Starfield 數位憑證，並儲存在`sf-class2-root.crt`本機或主目錄中。

```
curl https://certs.secureserver.net/repository/sf-class2-root.crt -O
```

> **Note**
>
> 您也可以使用 Amazon 數位憑證連線到 Amazon 金鑰空間，如果您的用戶端成功連線至 Amazon 金鑰空間，則可以繼續這麼做。Starfield 憑證為使用舊版憑證授權單位的用戶端提供額外的向後相容性。

```
curl https://certs.secureserver.net/repository/sf-class2-root.crt -O
```

儲存憑證檔案後，您可以連線到 Amazon 金鑰空間。一種選擇是使用服務特定憑據進行連接。服務特定登入資料是與特定 IAM 使用者相關聯的使用者名稱和密碼，只能與指定的服務搭配使用。第二個選項是連接使用AWS簽名版本 4 程序 (Sigv4) 的 IAM 登入資料。若要深入瞭解這兩個選項，請參閱the section called “建立認證”。

若要連線服務特定認證，請參閱the section called “使用服務特定認證連線”。

若要連線 IAM 登入資料，請參閱the section called “使用 IAM 登入資料連線”。

使用服務特定登入資料連線至 Amazon 金鑰空間

本節說明如何使用服務特定認證來確認您使用 NoSQL Workbench 建立或編輯的資料模型。

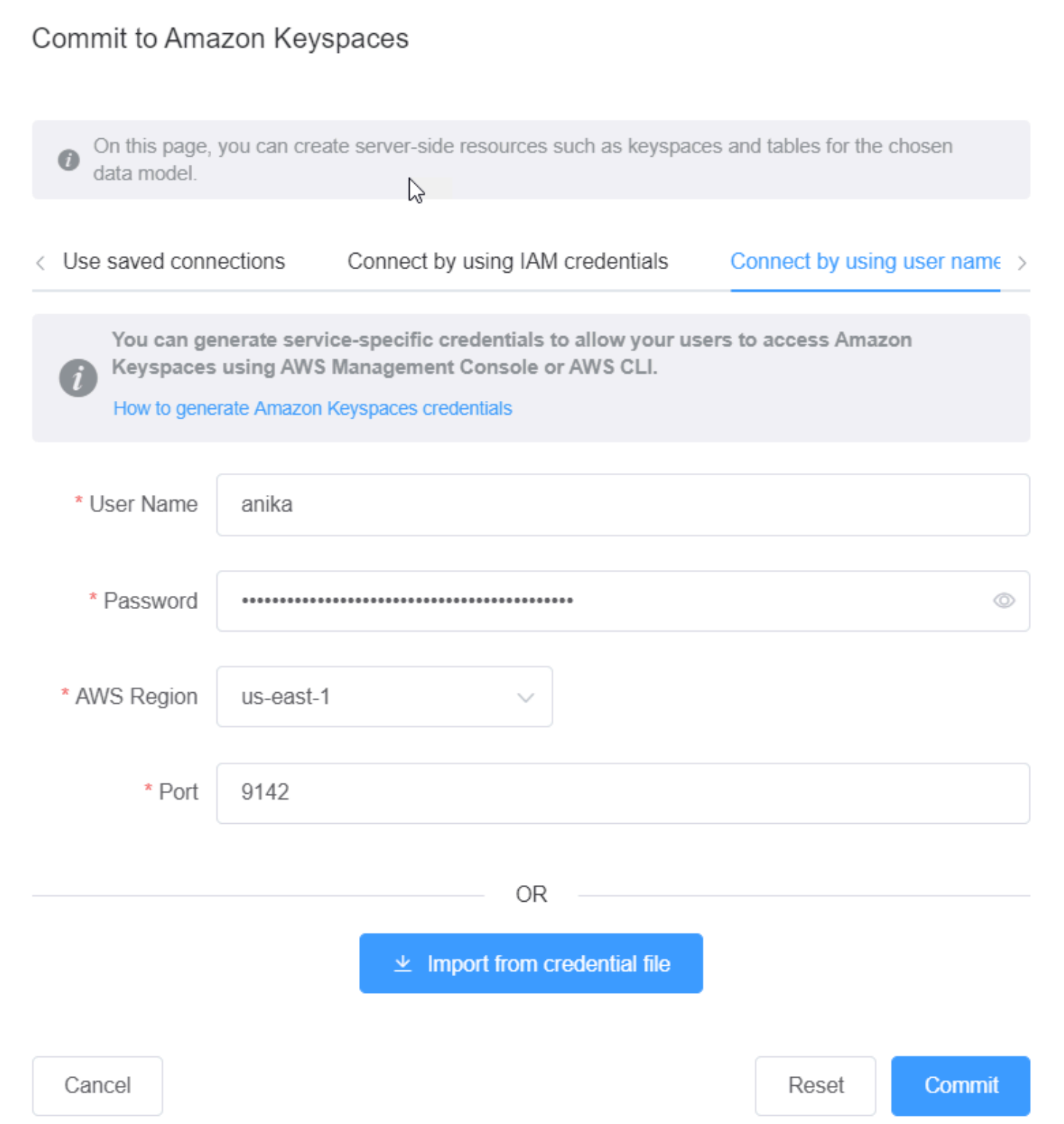
1. 若要使用服務特定認證建立新連線，請選擇使用者名稱和密碼連線索引標籤。

   - 開始之前，您必須使用所述的程序建立服務特定認證。the section called “服務特定認證”

   取得服務特定認證後，您可以繼續設定連線。繼續執行下列其中一項作業：

   - 使用者名稱 — 輸入使用者名稱。
   - 密碼 — 輸入密碼。
   - AWS 區域— 如需可用的區域，請參閱the section called “服務端點”。
   - 端口 — 亞馬遜密鑰空間使用端口 9142。

   或者，您也可以從檔案匯入已儲存的認證。
2. 選擇「提交」以使用資料模型更新 Amazon 金鑰空間。


Commit to Amazon Keyspaces
On this page, you can create server-side resources such as keyspaces and tables for the chosen data model.
Use saved connections
Connect by using IAM credentials
Connect by using user name
You can generate service-specific credentials to allow your users to access Amazon Keyspaces using AWS Management Console or AWS CLI.
How to generate Amazon Keyspaces credentials
* User Name
anika
* Password
* AWS Region
us-east-1
* Port
9142
OR
Import from credential file
Cancel
Reset
Commit

使用 AWS Identity and Access Management (IAM) 登入資料連接到亞馬遜金鑰空間

本節說明如何使用 IAM 登入資料來提交使用 NoSQL 工作台建立或編輯的資料模型。

1. 若要使用 IAM 登入資料建立新連線，請選擇使用 IAM 登入資料連線索引標籤。

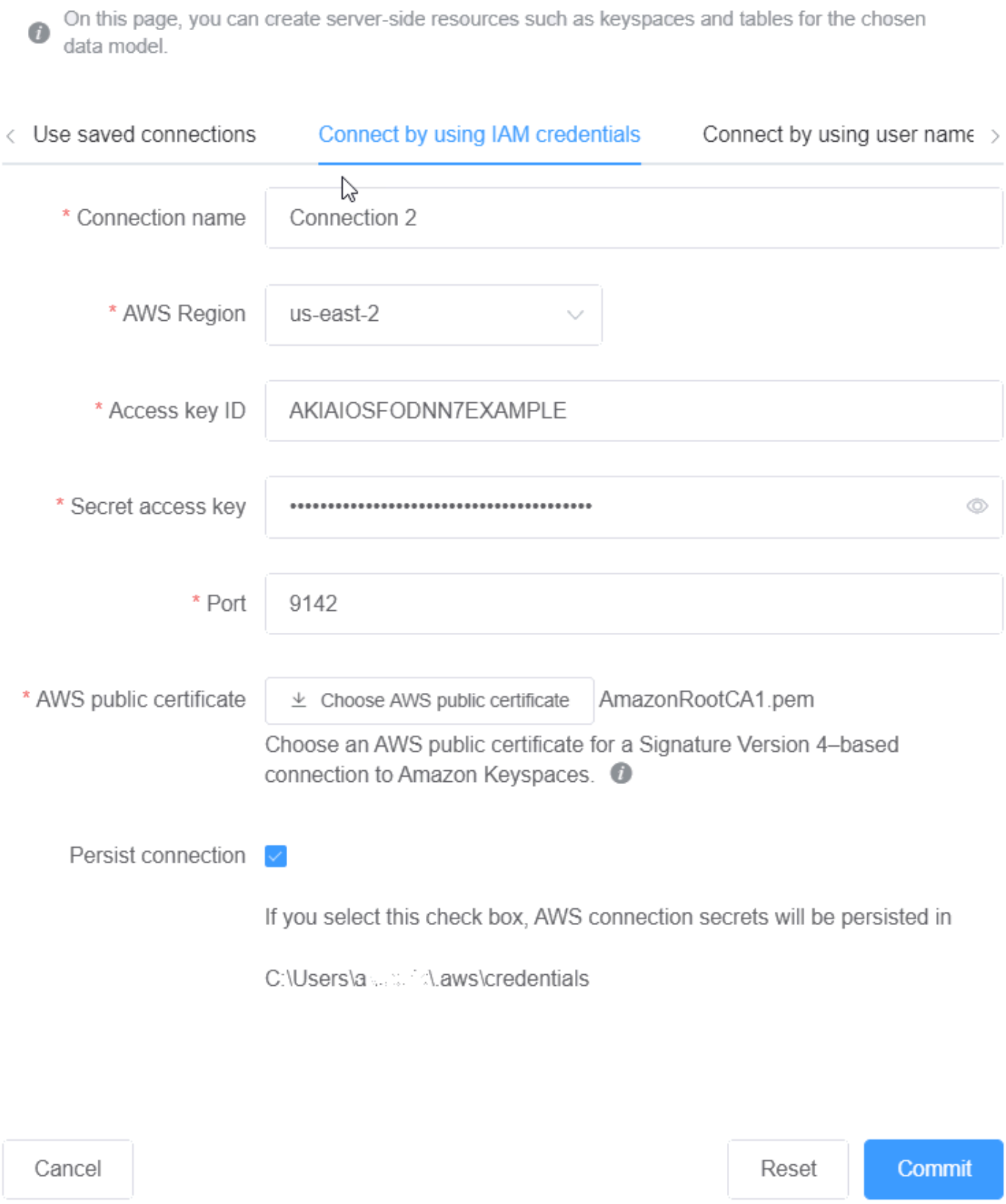
   - 開始之前，您必須使用下列其中一種方法建立 IAM 登入資料。

     - 對於主控台存取權，請使用您的 IAM 使用者名稱和密碼AWS Management Console從 IAM 登入頁面登入。如需AWS安全登入資料的相關資訊，包括程式設計存取和替代長期登入資料，請參閱 IAM 使用者指南中的AWS安全登入資料。如需有關登入的詳細資訊AWS 帳戶，請參閱AWS 登入使用者指南AWS中的如何登入。
     - 對於 CLI 存取，您需要存取金鑰 ID 和私密存取金鑰。盡可能使用臨時憑證，而不是長期存取金鑰。臨時憑證包含存取金鑰 ID、私密存取金鑰，以及指出憑證何時到期的安全符記。如需詳細資訊，請參閱 IAM 使用者指南中的將臨時登入資料與AWS資源搭配使用。
     - 如果是 API 存取，您需要存取金鑰 ID 和私密存取金鑰。使用 IAM 使用者存取金鑰，而非 AWS 帳戶根使用者 存取金鑰。如需建立存取金鑰的詳細資訊，請參閱IAM 使用者指南中的管理 IAM 使用者的存取金鑰。

     如需詳細資訊，請參閱管理 IAM 使用者的存取金鑰。

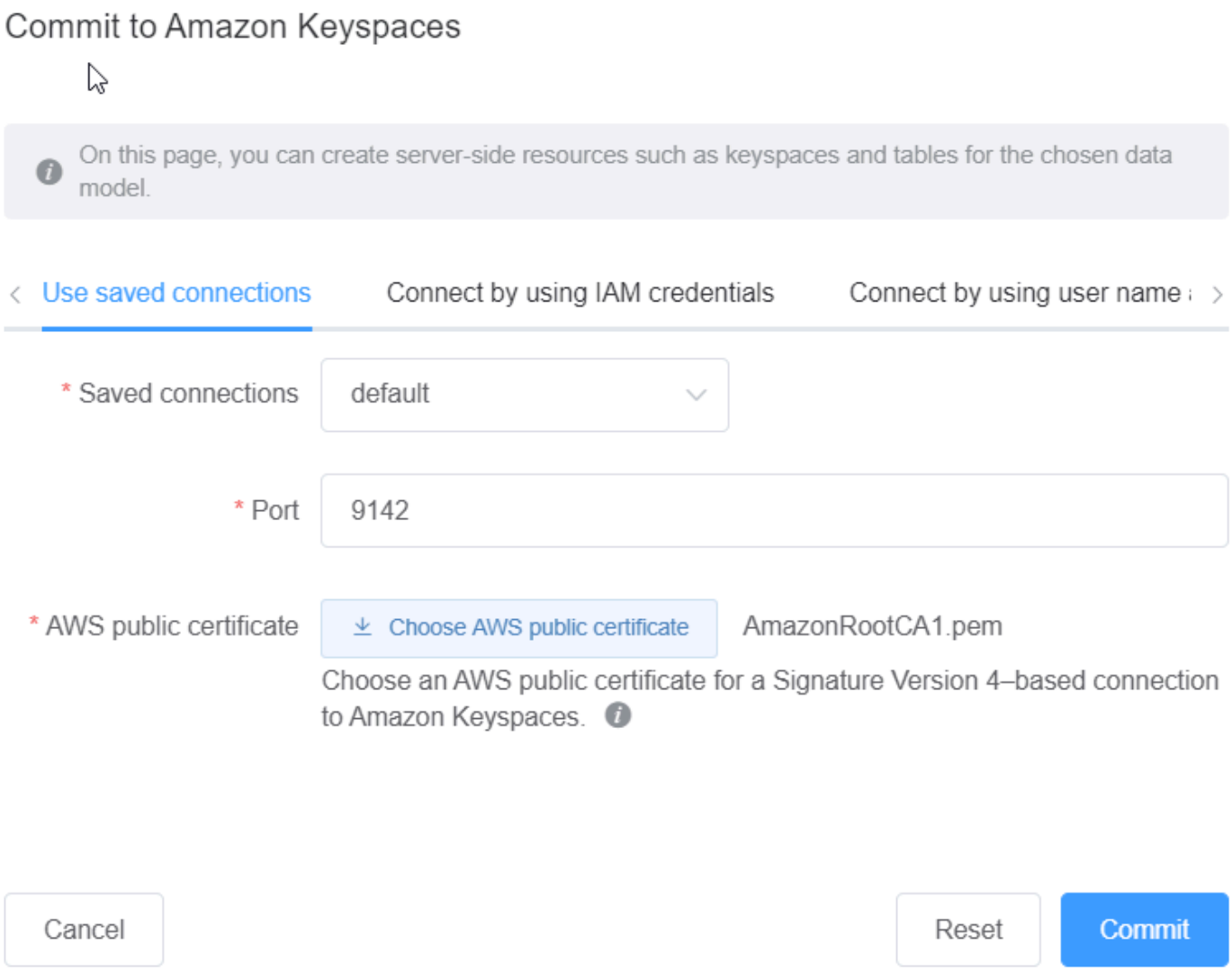
   取得 IAM 登入資料後，您可以繼續設定連線。

   - 連線名稱 — 連線的名稱。
   - AWS 區域— 如需可用的區域，請參閱the section called “服務端點”。
   - 存取金鑰 ID — 輸入存取金鑰 ID。
   - 秘密存取金鑰 — 輸入秘密存取金鑰。
   - 端口 — 亞馬遜密鑰空間使用端口 9142。
   - AWS公用憑證 — 指向第一個步驟中下載的AWS憑證。
   - 持續連線 — 如果您要在本機儲存AWS連線密碼，請選取此核取方塊。
2. 選擇「提交」以使用資料模型更新 Amazon 金鑰空間。


On this page, you can create server-side resources such as keyspaces and tables for the chosen data model.
Use saved connections
Connect by using IAM credentials
Connect by using user name
* Connection name
Connection 2
* AWS Region
us-east-2
* Access key ID
AKIAIOSFODNN7EXAMPLE
* Secret access key
* Port
9142
* AWS public certificate
Choose AWS public certificate
AmazonRootCA1.pem
Choose an AWS public certificate for a Signature Version 4–based connection to Amazon Keyspaces.
Persist connection
If you select this check box, AWS connection secrets will be persisted in
C:\Users\a .aws\credentials
Cancel
Reset
Commit

使用已儲存的連線

如果您先前已設定 Amazon Keyspaces 的連線，則可以使用該連線做為提交資料模型變更的預設連線。選擇 [使用已儲存的連線] 索引標籤，然後繼續認可更新。

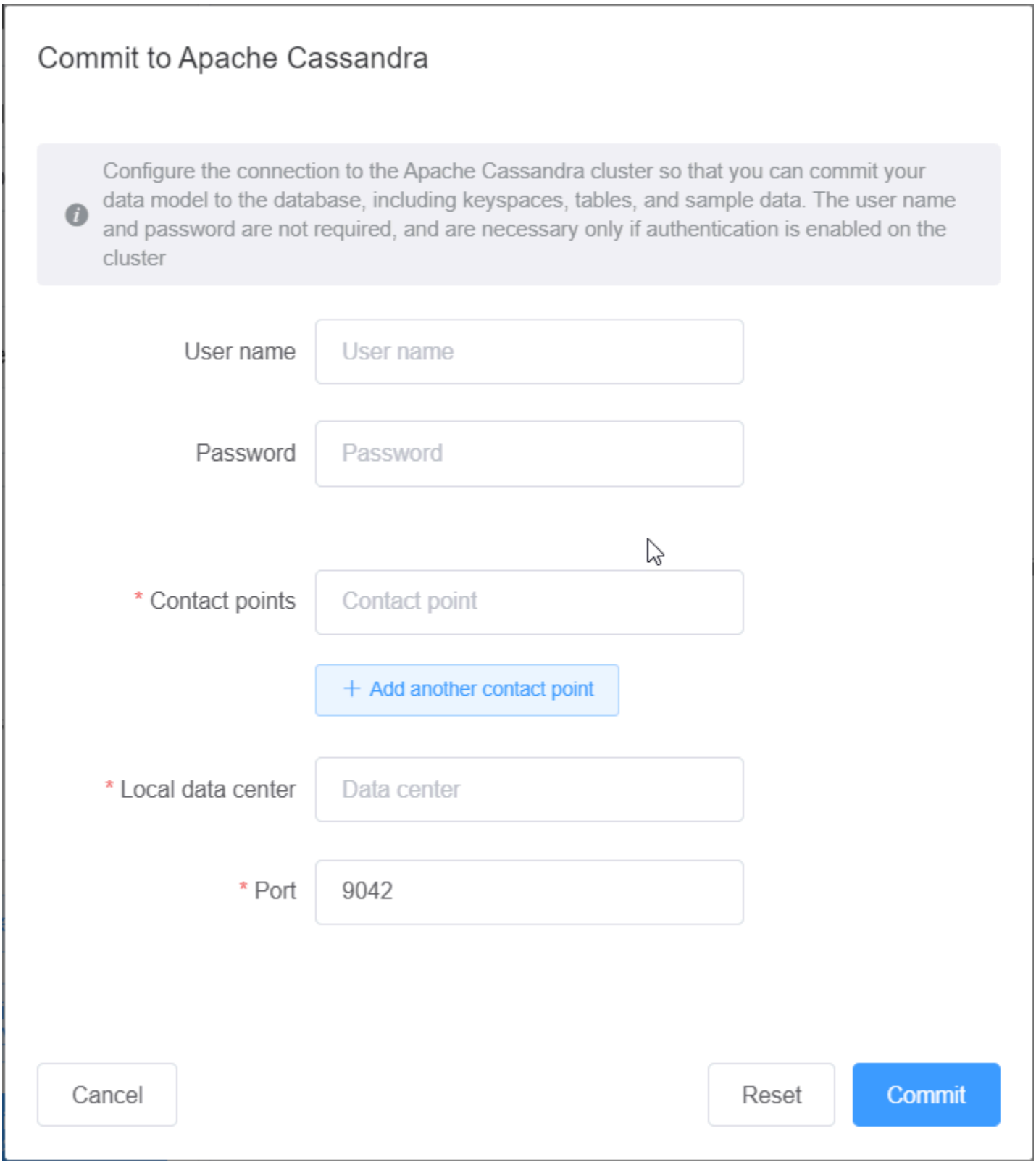


致力於阿帕奇卡桑德拉

本節將引導您完成連接到 Apache 卡桑德拉集群提交創建或使用 NoSQL 工作台編輯的數據模型。

> **Note**
>
> 已經創建`SimpleStrategy`或`NetworkTopologyStrategy`可以提交給 Apache 卡桑德拉集群只有數據模型。若要變更複製策略，請在資料建模器中編輯金鑰空間。

1. • 使用者名稱 — 如果已在叢集上啟用驗證，請輸入使用者名稱。
   • 密碼 — 如果已在叢集上啟用驗證，請輸入密碼。
   • 聯絡點 — 輸入聯絡點。
   • 本機資料中心 — 輸入本機資料中心的名稱。
   • 連接埠 — 連線使用連接埠 9042。
2. 選擇提交更新 Apache 卡桑德拉集群與數據模型。


Commit to Apache Cassandra
Configure the connection to the Apache Cassandra cluster so that you can commit your data model to the database, including keyspaces, tables, and sample data. The user name and password are not required, and are necessary only if authentication is enabled on the cluster
User name
User name
Password
Password
* Contact points
Contact point
+ Add another contact point
* Local data center
Data center
* Port
9042
Cancel
Reset
Commit

NoSQL Workbench 的範例資料模型

建模工具和視覺化檢視的首頁會顯示一些隨附 NoSQL 工作台的範例模型。本節說明這些模型及其潛在用途。

主題

員工資料模型

此資料模型代表員工資料庫應用程式的 Amazon Keyspaces 架構。

訪問給定公司的員工信息的應用程序可以使用此數據模型。

此資料模型支援的存取模式為：

- 具有給定 ID 的員工記錄的檢索。
- 具有給定 ID 和除法的員工記錄的檢索。
- 具有給定 ID 和姓名的員工記錄的檢索。

信用卡交易資料模型

此資料模型代表在零售商店進行信用卡交易的 Amazon Keyspaces 架構。

信用卡交易的存儲不僅可以幫助商店進行簿記，還可以幫助商店經理分析購買趨勢，從而幫助他們進行預測和計劃。

此資料模型支援的存取模式為：

- 按信用卡號，月份和年份以及日期檢索交易。
- 按信用卡號，類別和日期檢索交易。
- 按類別，位置和信用卡號檢索交易。
- 按信用卡號碼和爭議狀態檢索交易。

航空營運資料模型

此數據模型顯示有關飛機航班的數據，包括機場，航空公司和航線。

展示的 Amazon 密 Keyspaces 建模的關鍵元件包括鍵值對、寬欄資料存放區、複合金鑰和複雜資料類型 (例如用於展示常見 NoSQL 資料存取模式的映射)。

此資料模型支援的存取模式為：

- 在指定機場檢索來自特定航空公司的航線。
- 檢索具有給定目的地機場的路線。
- 檢索具有直飛航班的機場。
- 檢索機場詳細信息和航空公司詳細信息

NoSQL Workbench 的版本歷史記錄

下表說明 NoSQL Workbench 的重要變更。

變更	描述	日期
Amazon NoSQL Keyspaces ：。	Amazon NoSQL Workbench 現 Keyspaces 式推出。	2020 年 10 月 28 日
已發行 NoSQL Workbench 預覽。	NoSQL Workbench 是用戶端應用程式，可協助您更輕鬆地設計和視覺化 Amazon Keyspaces 的非關聯式資料模型。NoSQL Workbench 現已可用 macOS SQL Workbench 。如需詳細資訊，請參閱 Amazon NoSQL Keyspaces。	2020 年 10 月 5 日

Amazon Keyspaces 的多區域複寫（適用於阿帕奇卡桑德拉）

您可以使用 Amazon Keyspaces space 多區域複寫，透過自動化、全受管的主動-主動複寫功能複寫您的 AWS 區域 資料。使用主動-主動複寫，每個區域都能夠隔離執行讀取和寫入。您可以通過區域性降級提高可用性和恢復能力，同時還可以受益於全球應用程序的低延遲本地讀取和寫入。

透過多區域複寫，Amazon Keyspace 會在區域之間非同步複寫資料，而資料通常會在一秒鐘內跨區域傳播。此外，透過多區域複寫，您不再需要解決衝突和修正資料分歧問題的困難工作，因此您可以專注於應用程式。

根據預設，Amazon Keyspaces 會在同一個區域內的三個可用區域複寫資料，以提高耐 AWS 區域 用性和高可用性。使用多區域複寫，您可以建立多區域金鑰空間，在您選擇 AWS 區域 的六個不同地理位置複製表格。

主題

- 使用多區域複寫的好處
- 容量模式和定價
- 多區域複寫如何在 Amazon Keyspaces 中運作
- Amazon Keyspaces 多區域複寫使用注意事項
- 如何使用多區域複寫

使用多區域複寫的好處

多區域複寫提供下列優點。

- 具有 10 毫秒延遲的全域讀取和寫入 — 在 Amazon Keyspaces 中，複寫是主動式的。您可以從距離客戶最近的區域提供讀取和寫入服務，任何規模的延遲時間都只有 10 毫秒。您可以在全球任何地方都需要快速回應時間的全球應用程式使用 Amazon Keyspaces 多區域表。
- 改善業務連續性與防護，避免單一區域降級 — 透 AWS 區域 過多區域複寫，您可以將應用程式重新導向至多區域金鑰空間中的不同區域，從單一降級中復原。由於 Amazon Keyspaces 提供主動-主動複寫，因此不會影響您的讀取和寫入。

  Amazon Keyspaces 間會追蹤已在多區域金鑰空間上執行但尚未傳播到所有複本區域的任何寫入。區域重新上線後，Amazon Keyspaces 會自動同步任何遺失的變更，以便您可以在不影響應用程式的情況下進行復原。

- 跨區域的高速複寫 — 多區域複寫使用跨區域快速、以儲存為基礎的實體複製資料，複寫延遲通常少於 1 秒。

  Amazon Keyspaces 中的複寫對資料庫查詢幾乎沒有影響，因為它不會與您的應用程式共用運算資源。這表示您可以處理高寫入輸送量的使用案例，或是突然尖峰或高增輸送量的使用案例，而不會對應用程式造成任何影響。
- 一致性和解決衝突 — 對任何區域中的資料所做的任何變更都會複寫到多區域金鑰空間中的其他區域。如果應用程式同時更新不同區域中的相同資料，可能會發生衝突。

  為了協助提供最終一致性，Amazon Keyspace 使用儲存格層級的時間戳記，而最後一個寫入器會贏得並行更新之間的對帳。衝突解決是完全管理的，並在背景中進行，而不會影響任何應用程式。

如需有關支援的組態和功能的詳細資訊，請參閱the section called “使用須知”。

容量模式和定價

對於多區域金鑰空間，您可以使用隨需容量模式或佈建的容量模式。如需詳細資訊，請參閱 the section called “讀/寫容量模式”。

對於隨需模式，您需支付 1.25 個寫入請求單位 (WRU) 的費用，每列最多可寫入 1 KB 的資料。您需支付多區域金鑰空間中每個區域的寫入費用。例如，在具有兩個區域的多區域金鑰空間中寫入一列 3 KB 的資料需要 7.5 WRU：3 * 1.25 * 2 = 7.5 WRU。此外，包含靜態和非靜態資料的寫入作業也需要額外的寫入作業。

對於佈建模式，您需支付 1.25 個寫入容量單位 (WCU) 的費用，每列最多可寫入 1 KB 的資料。您需支付多區域金鑰空間中每個區域的寫入費用。例如，在具有兩個區域的多區域金鑰空間中，每秒寫入 3 KB 資料的資料列需要 7.5 個 WCU：3 * 1.25 * 2 = 7.5 WCU。此外，包含靜態和非靜態資料的寫入作業也需要額外的寫入作業。

有關定價的更多信息，請參閱 Amazon Keyspaces（阿帕奇卡桑德拉）定價。

多區域複寫如何在 Amazon Keyspaces 中運作

本節提供 Amazon Keyspaces 多區域複寫運作方式的概觀。有關定價的更多信息，請參閱 Amazon Keyspaces（阿帕奇卡桑德拉）定價。

主題

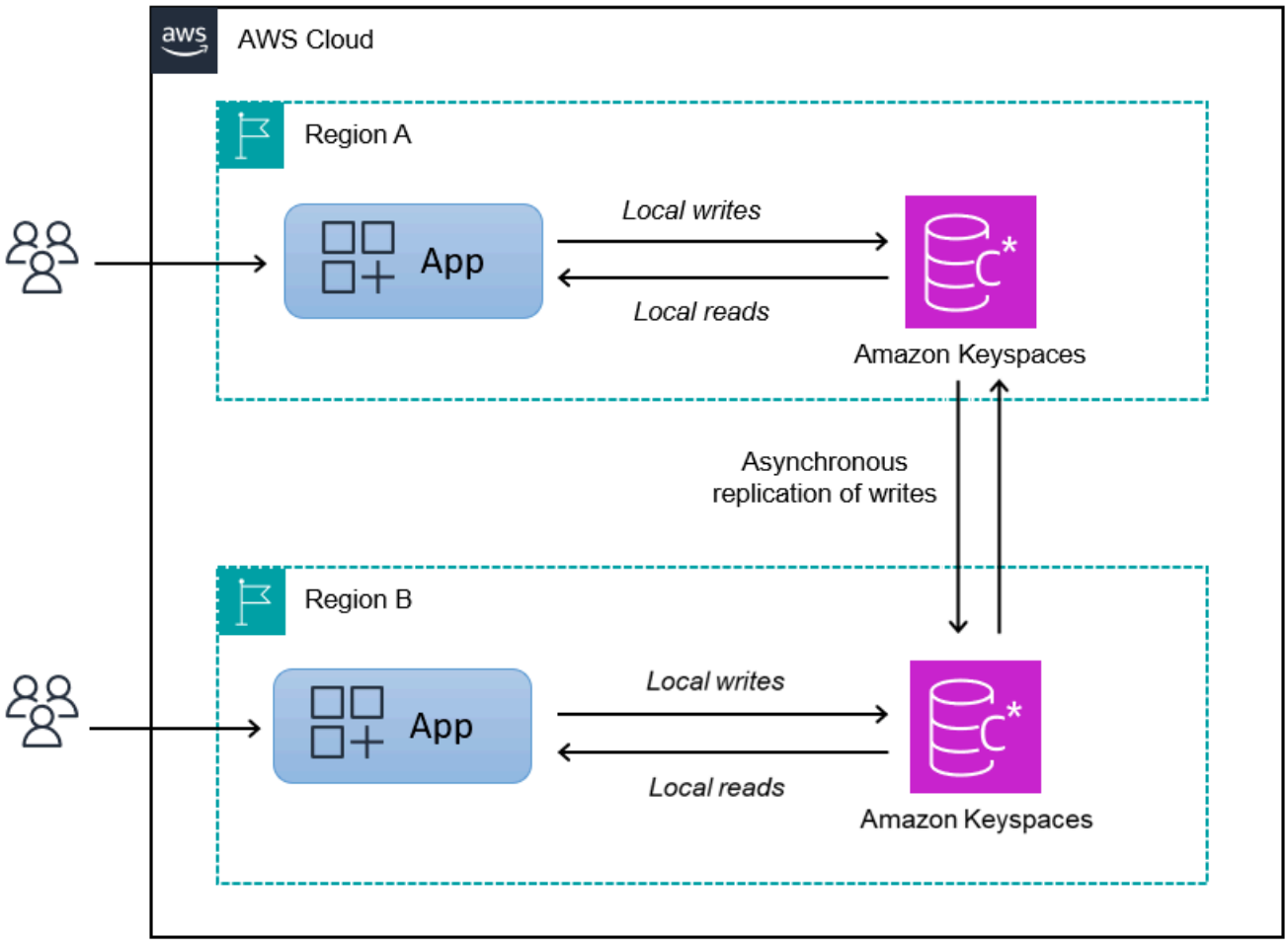
多區域複寫如何在 Amazon Keyspaces 中運作

Amazon Keyspaces 多區域複寫實作資料彈性架構，可將您的資料分散到獨立且分散在地理位置上。AWS 區域它使用主動-主動複寫，提供本機低延遲，每個區域能夠隔離執行讀取和寫入。

建立 Amazon Keyspaces 間多區域金鑰空間時，您最多可以選取五個將資料複寫到的其他區域。您在多區域金鑰空間中建立的每個表都包含多個複本表格 (每個區域一個)，Amazon Keyspaces 間視為單一單元。

每個複本都有相同的資料表名稱和相同的主索引鍵結構描述。當應用程式將資料寫入一個區域中的本機資料表時，資料會使用一LOCAL_QUORUM致性層級持續寫入。Amazon Keyspaces 會自動將資料以非同步方式複寫到其他複寫區域。跨區域的複寫延遲通常少於一秒鐘，而且不會影響應用程式的效能或輸送量。

寫入資料之後，您可以從具有一LOCAL_ONE/LOCAL_QUORUM致性層級的另一個複寫區域中的「多區域」表格中讀取資料。如需支援的組態和功能的詳細資訊，請參閱the section called “使用須知”。



解決多區域複寫衝突

Amazon Keyspaces space 多區域複寫是全受管的，這表示您不必執行複寫任務，例如定期執行修復操作以清除資料同步問題。Amazon Keyspaces 會偵測和修復衝突 AWS 區域 來監控不同資料表之間的資料一致性，並自動同步複本。

Amazon Keyspaces 使用最後一個寫入器 wins 數據對賬的方法。使用這種衝突解決機制，多區域密鑰空間中的所有區域都同意最新更新，並匯合到它們都具有相同數據的狀態。調解程序不會影響應用程式效能。若要支援衝突解決，多區域資料表的用戶端時間戳記會自動開啟，且無法關閉。如需詳細資訊，請參閱 用戶端時間時間。

多區域複寫災難回復

使用 Amazon Keyspaces 多區域複寫，可跨每個區域非同步複寫寫入。在極少數發生單一區域降級或故障的情況下，多區域複寫可協助您從災難中復原，而不會對應用程式造成影響。從災難復原通常使用復原時間目標 (RTO) 和復原點目標 (RPO) 的值來衡量。

復原時間目標 — 系統在災難發生後回復運作狀態所需的時間。RTO 會測量工作負載可容忍的停機時間量，並以時間測量。對於使用「多區域複寫」容錯移轉至未受影響區域的災難復原計畫，RTO 可以為零。RTO 受到應用程式偵測失敗狀況並將流量重新導向至其他區域的速度所限制。

復原點目標 — 可能遺失的資料量 (以時間為單位)。對於使用多區域複寫容錯移轉至未受影響區域的災難復原計劃，RPO 通常為單位數秒。RPO 受到容錯移轉目標複本的複寫延遲所限制。

如果發生區域故障或降級，您不需要提升次要區域或執行資料庫容錯移轉程序，因為 Amazon Keyspaces 中的複寫處於作用中狀態。相反地，您可以使用 Amazon Route 53 將應用程式路由到最近的健康區域。要了解有關 53 號路線的更多信息，請參閱什麼是 Amazon 路線 53？ 。

如果單一 AWS 區域 變成隔離或降級，您的應用程式可以使用 Route 53 將流量重新導向至不同的區域，以針對不同的複本表格執行讀取和寫入。您也可以套用自訂商務邏輯，以決定何時將要求重新導向至其他區域。其中一個範例是讓您的應用程式知道可用的多個端點。

當區域恢復線上狀態時，Amazon Keyspaces 會繼續將該區域的任何擱置寫入傳播到其他區域中的複本表格。其也會繼續將寫入從其他複本列表傳播到目前已重回到線上狀態的區域。

建立多區域金鑰空間和資料表所需的 IAM 許可

若要成功建立多區域金鑰空間和表格，IAM 主體必須能夠建立服務連結角色。此服務連結角色是 Amazon Keyspaces 預先定義的唯一 IAM 角色類型。它包括 Amazon Keyspaces 代表您執行動作所需的所有許可。如需服務連結角色的詳細資訊，請參閱the section called “多區域複製”。

若要建立多區域複寫所需的服務連結角色，IAM 主體的政策需要下列元素：

- `iam:CreateServiceLinkedRole`— 主參與者可執行的動作。
- `arn:aws:iam::*:role/aws-service-role/replication.cassandra.amazonaws.com/AWSServiceRoleForKeyspacesReplication`— 可對其執行動作的資源。
- `iam:AWSServiceName": "replication.cassandra.amazonaws.com`— 此角色可附加到的唯一 AWS 服務是 Amazon Keyspaces。

以下是原則範例，此原則會授與主體建立多區域金鑰空間和資料表所需的最低權限。

```
{
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/
replication.cassandra.amazonaws.com/AWSServiceRoleForKeyspacesReplication",
            "Condition": {"StringLike": {"iam:AWSServiceName":
 "replication.cassandra.amazonaws.com"}}
}
```

如需多區域金 Keyspaces 和表格的其他 IAM 許可，請參閱服務授權參考中的 Amazon 金鑰空間 (適用於 Apache Cassandra) 的動作、資源和條件金鑰。

多區域複寫及與 point-in-time 復原整合 (PITR)

多區域表格支援 P oint-in-time 復原。若要使用 PITR 成功還原多區域表格，必須符合下列條件。

- 來源和目標資料表必須設定為多區域資料表。
- 來源資料表之索引鍵空間與目標資料表之索引鍵空間的複製區域必須相同。

您可以從來源資料表所在的任何區域執行 restore 陳述式。Amazon Keyspaces 會自動還原每個區域中的目標資料表。如需 PITR 的相關詳細資訊，請參閱 the section called “運作方式”。

多區域複寫與 AWS 服務整合

您可以使用 Amazon CloudWatch 指標監控不同 AWS 區域 資料表之間的複寫效能。以下指標提供了對多區域密鑰空間的持續監視。

- `ReplicationLatency`— 此指標測量多區域索引鍵空間中複製`updates`或`deletes`從一個複本表格複製到另一個複本表格所花的時間。`inserts`

如需如何監視 CloudWatch 指標的詳細資訊，請參閱the section called “使用監控 CloudWatch”。

Amazon Keyspaces 多區域複寫使用注意事項

搭配 Amazon Keyspaces 使用多區域複寫時，請考慮下列事項。

- 您最多可以選擇六個可用的公眾 AWS 區域。 AWS GovCloud (US) Regions、中國地區， AWS 區域 以及預設為停用的區域不受支援。
- 請仔細選取金鑰空間的複寫區域，因為您稍後無法新增或移除它們。

- 建立多區域資料表之前，請先完成資料表結構定義，因為您之後無法新增資料欄。
- 若要進行靜態加密，請使用 AWS 擁有的金鑰。多區域資料表不支援客戶管理金鑰。如需詳細資訊，請參閱

  the section called “運作方式”.
- 使用 Amazon Keyspaces auto 擴展的佈建容量管理時，請務必使用 Amazon Keyspaces API 操作來建立和設定多區域表。Amazon Keyspaces 代表您呼叫的基礎 Application Auto Scaling API 操作沒有多區域功能。

  如需詳細資訊，請參閱 the section called “如何使用多區域複寫”。如需如何預估佈建的多區域表格寫入容量輸送量的詳細資訊，請參閱the section called “多區域表”。
- 決定表格是否需要存留時間 (TTL)。您之後將無法將其開啟。如需詳細資訊，請參閱 使用存留時間讓存留時間過期資料。
- 雖然資料會自動跨多區域表格的所選區域複寫，但是當用戶端連線到一個區域中的端點並查詢資料`system.peers`表時，查詢只會傳回本機資訊。查詢結果會顯示為用戶端的單一資料中心叢集。
- Amazon Keyspaces 多區域複寫是非同步的，並且支援寫入的`LOCAL_QUORUM`一致性。`LOCAL_QUORUM`一致性要求在將成功傳回用戶端之前，在本機區域中的兩個複本上持續保留對資料列的更新。然後會以非同步方式執行複寫區域 (或區域) 的寫入傳播。

  Amazon Keyspaces 多區域複寫不支援同步複寫或`QUORUM`一致性。
- 當您建立多區域金鑰空間或表格時，您在建立程序期間定義的任何標籤都會自動套用至所有區域中的所有金鑰空間和表格。當您使用`ALTER KEYSPACE`或變更現有的標籤時`ALTER TABLE`，更新只會套用至您要進行變更的區域中的金鑰空間或資料表。
- Amazon 會為每個複寫區域 CloudWatch 提供`ReplicationLatency`指標。它會追蹤到達資料列、比較抵達時間與初始寫入時間，以及計算平均值，以計算此量度。計時會儲存 CloudWatch 在來源區域內。如需詳細資訊，請參閱 the section called “使用監控 CloudWatch”。

  檢視平均和最大計時，以判斷平均和最差情況的複寫延遲會很有用。此延遲沒有 SLA。
- 在隨選模式下使用多區域表格時，如果表格複本遇到新的流量峰值，您可能會發現寫入的非同步複寫延遲會增加。與 Amazon Keyspaces 自動將單一區域隨需表格的容量調整為其接收到的應用程式流量類似，Amazon Keyspaces 會自動根據所接收的流量調整多區域隨需表格複本的容量。複寫延遲的增加是暫時的，因為 Amazon Keyspaces 會隨著流量增加而自動配置更多容量。一旦所有複本都適應了您的流量，複寫延遲應該會恢復正常。如需詳細資訊，請參閱 the section called “峰值流量與擴展屬性”。
- 在佈建模式下使用多區域表格時，如果您的應用程式超過佈建的輸送量容量，您可能會發現容量不足錯誤並增加複寫延遲。為確保所有多區域表格中的所有表格複本始終有足夠的讀取和寫入容量，我們

建議您設定 Amazon Keyspaces auto 擴展。 AWS 區域 Amazon Keyspaces auto 動擴展可透過自動調整輸送量容量以回應實際應用程式流量，協助您有效地為可變工作負載佈建輸送量容量。如需更多詳細資訊，請參閱 the section called “多區域表格 auto 縮放的運作方式”。

如何使用多區域複寫

您可以創建和使用 Amazon Keyspaces（Apache 卡桑德拉）控制台，卡桑德拉查詢語言（CQL），SDK 和（ ）管理多區域密鑰空間和表。 AWS AWS Command Line Interface AWS CLI

本節提供如何使用隨需和佈建容量模式，透過主控台、使用 CQL 建立多區域金鑰空間和表格的範例。AWS CLI在多區域密鑰空間中創建的所有表都會自動從密鑰空間繼承多區域設置。

本節還包括如何使用主控台、CQL 以及管理已佈建多區域表的 Amazon Keyspaces auto 擴展設定的範例。 AWS CLI 如需一般 auto 調整設定選項及其運作方式的詳細資訊，請參閱the section called “透過 auto 擴充管理輸送量容量”。

請注意，如果您使用多區域表的佈建容量模式，則必須始終使用 Amazon Keyspaces API 呼叫來設定 auto 擴展。這是因為基礎的 Application Auto Scaling API 作業不感知區域。

如需如何預估佈建的多區域表格寫入容量輸送量的詳細資訊，請參閱the section called “多區域表”。

如需有關 Amazon Keyspaces API 的詳細資訊，請參閱 Amazon Keyspaces API 參考。

如需有關支援的組態和多區域複寫功能的詳細資訊，請參閱the section called “使用須知”。

主題

- 使用主控台建立和管理多區域資料表
- 使用 CQL 創建和管理多區域表
- 使用建 AWS CLI 立和管理多區域資料表

使用主控台建立和管理多區域資料表

本節提供如何使用 Amazon Keyspaces (適用於 Apache Cassandra) 主控台以隨需和佈建容量模式建立多區域金鑰空間和表格的範例。您在多區域密鑰空間中創建的所有表都會自動從密鑰空間繼承多區域設置。

如需 CQL 範例，請參閱the section called “使用定制列表”。如需 AWS CLI 範例，請參閱the section called “使用 AWS CLI”。

主題

創建多區域密鑰空間（控制台）

請依照下列步驟使用 Amazon 金鑰空間主控台建立新的多區域 Keyspaces 間。

若要建立多區域金鑰空間 (主控台)

1. 登錄到 AWS Management Console，然後打開 Amazon Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。
2. 在功能窗格中，選擇 [Keyspaces 間]，然後選擇 [建立金鑰空間]。
3. 針對金鑰空間名稱，輸入金鑰空間的名稱。
4. 在「多區域複製」段落中，您最多可以新增五個清單中可用的其他區域。
5. 若要完成，請選擇 [建立金鑰空間]。

> **Note**
>
> 建立多區域金鑰空間時，Amazon Keyspaces space 會在您的帳戶中建立名稱`AWSServiceRoleForAmazonKeyspacesReplication`的服務連結角色。此角色可讓 Amazon Keyspaces 代表您將寫入複寫到多區域表格的所有複本。如需進一步了解，請參閱 the section called “多區域複製”。

使用默認設置創建多區域表（控制台）

請依照下列步驟使用 Amazon Keyspaces 主控台建立多區域表格。

若要建立多區域表格 (主控台)

1. 登錄到 AWS Management Console，然後打開 Amazon Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。
2. 選擇多區域金鑰空間。
3. 在 [表格] 索引標籤上選擇 [建立表格]。
4. 在「表格名稱」中，輸入表格的名稱。正在複製此表格的資訊方塊中會顯示。 AWS 區域
5. 繼續執行資料表結構定義。
6. 在「表格設定」 下，繼續使用「預設設定」 選項。請注意下列多區域表格的預設設定。

   - 容量模式 — 預設容量模式為隨選。如需有關配置已佈建模式的詳細資訊，請參閱the section called “在啟用 auto 擴展的佈建模式中建立多區域表格 (主控台)”。
   - 加密金鑰管理 — 僅支援AWS 擁有的金鑰選項。
   - 用戶端時間戳記 — 多區域資料表需要此功能。

   - 如果您需要為表格及其所有複本開啟存留時間 (TTL)，請選擇「自訂設定」。

     > **Note**
     >
     > 您無法在現有的多區域表格上變更 TTL 設定。

7. 若要完成，請選擇 [建立表格]。

在啟用 auto 擴展的佈建模式中建立多區域表格 (主控台)

> **Note**
>
> Amazon Keyspaces 自動擴展需要存在代表您執行自動擴展動作的服務連結角色 (`AWSServiceRoleForApplicationAutoScaling_CassandraTable`)。系統會自動建立此角色。如需詳細資訊，請參閱 the section called “使用服務連結角色”。

建立已啟用自動調整比例的新多區域表

1. 登錄到 AWS Management Console，然後打開 Amazon Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。

2. 選擇多區域金鑰空間。
3. 在 [表格] 索引標籤上選擇 [建立表格]。
4. 在「表格詳細資訊」段落的「建立表格」頁面中，選取索引鍵空間並輸入新表格的名稱。
5. 在「欄」區段中，建立資料表的結構定義。
6. 在 [主索引鍵] 區段中，定義資料表的主索引鍵，並選取選用的叢集資料行。
7. 在 [表格設定] 區段中，選擇 [自訂設定]。
8. 繼續參閱讀/寫入容量設定。
9. 對於容量模式，選擇已佈建。
10. 在 [讀取容量] 區段中，確認已選取 [自動擴充]。

    您可以選擇為複製表格的所有 AWS 區域 讀取容量單位設定相同的讀取容量單位。或者，您可以清除核取方塊，並以不同方式設定每個區域的讀取容量。

    如果您選擇以不同的方式設定每個區域，您可以為每個表格複本選取最小和最大讀取容量單位，以及目標使用率。

    - 最小容量單位 — 輸入表格應隨時可支援的最小輸送量層次值。此值必須介於 1 和帳戶每秒輸送量上限配額之間 (預設為 40,000)。
    - 最大容量單位 — 輸入您要為表格佈建的最大輸送量量。此值必須介於 1 和帳戶每秒輸送量上限配額之間 (預設為 40,000)。
    - 目標使用率 — 輸入介於 20% 到 90% 之間的目標使用率。當流量超過定義的目標使用率時，容量會自動擴充。當流量低於定義的目標時，它會再次自動縮小。

    - 如果您想要手動佈建表格的讀取容量，請清除 [自動調整] 核取方塊。此設定會套用至表格的所有複本。

      > **Note**
      >
      > 為了確保所有複本都有足夠的讀取容量，我們建議您為佈建的多區域表自動擴展 Amazon Keyspaces。

    > **Note**
    >
    > 若要深入瞭解帳戶的預設配額，以及如何增加配額，請參閱配額。

11. 在 [寫入容量] 區段中，確認已選取 [自動擴充]。然後設定表格的容量單位。寫入容量單位在所有單位之間保持同步， AWS 區域 以確保有足夠的容量來複寫跨區域的寫入事件。
    - 如果您想要手動佈建表格的寫入容量，請清除「自動調整」。此設定會套用至表格的所有複本。

      > **Note**
      >
      > 為了確保所有複本都有足夠的寫入容量，我們建議您為佈建的多區域表自動擴展 Amazon Keyspaces。

12. 選擇 建立資料表 。您的表格是使用指定的自動縮放參數建立的。

為現有的多區域表（控制台）啟用 auto 擴展

請遵循下列步驟，使用 Amazon Keyspaces 主控台在佈建模式下為多區域表格啟用 auto 擴展。

> **Note**
>
> Amazon Keyspaces 自動擴展需要存在代表您執行自動擴展動作的服務連結角色 (`AWSServiceRoleForApplicationAutoScaling_CassandraTable`)。系統會自動建立此角色。如需詳細資訊，請參閱 the section called “使用服務連結角色”。

為現有的多區域表啟用 Amazon Keyspaces 自動擴展

1. 登錄到 AWS Management Console，然後打開 Amazon Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。
2. 選擇您要使用的表格，然後前往容量索引標籤。
3. 在 [容量設定] 區段中，選擇 [編輯]。
4. 在容量模式下，確定表格使用已佈建容量模式。
5. 選取 [自動調整]，並參閱步驟 9 中的在啟用 auto 擴展的佈建模式中建立多區域表格 (主控台)以編輯讀取和寫入容量。
6. 定義自動縮放設定後，請選擇「儲存」。

關閉多區域表（控制台）的 auto 調整

請依照下列步驟使用 Amazon Keyspaces 主控台關閉佈建模式下多區域表格的 auto 擴展功能。

關閉現有多區域表的 Amazon Keyspaces 自動擴展

1. 登錄到 AWS Management Console，然後打開 Amazon Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。
2. 選擇您要使用的表格，然後選擇 [容量] 索引標籤。
3. 在 [容量設定] 區段中，選擇 [編輯]。
4. 若要停用 Amazon Keyspaces 自動擴展，請清除「自動擴展」核取方塊。停用自動調整比例會使用「應用程式自動調整」將資料表取消註冊為可縮放 若要刪除 Application Auto Scaling 用來存取 Amazon Keyspaces 表格的服務連結角色，請依照中的步驟執行。the section called “刪除 Amazon Keyspaces 的服務鏈接角色”

   > **Note**
   >
   > 若要刪除 Application Auto Scaling 使用的服務連結角色，您必須停用帳戶中所有資料表的自動調度資源。 AWS 區域

5. 定義自動縮放設定後，請選擇「儲存」。

在主控台上查看 Amazon Keyspaces auto 動擴展活動

您可以使用 Amazon 監控 Amazon Keyspaces 自動擴展如何使用資源 CloudWatch，Amazon 會產生有關您的使用情況和效能的指標。按照使Application Auto Scaling 用者指南中的步驟建立 CloudWatch 儀表板。

使用 CQL 創建和管理多區域表

您可以使用卡桑德拉查詢語言（CQL）創建和管理 Amazon Keyspaces 中的多區域密鑰空間和表。

本節提供如何使用 CQL 建立和管理多區域資料表的範例。您在多區域密鑰空間中創建的所有表都會自動從密鑰空間繼承多區域設置。如需有關 CQL 的詳細資訊，請參閱 Amazon Keyspaces CQL 語言參考。

如需有關支援的組態和功能的詳細資訊，請參閱the section called “使用須知”。

主題

- 建立多區域金鑰空間 (CQL)
- 使用默認設置（CQL）創建多區域表
- 建立具有佈建容量模式和 auto 擴展 (CQL) 的多區域表

- 更新多區域表格 (CQL) 的佈建容量和 auto 擴展設定
- 檢視多區域表格 (CQL) 的佈建容量和 auto 調整設定
- 關閉多區域表格 (CQL) 的 auto 調整
- 手動設定多區域表格的佈建容量 (CQL)

建立多區域金鑰空間 (CQL)

若要建立多區域金鑰空間，請使用NetworkTopologyStrategy來指定要在 AWS 區域 其中複製金鑰空間的金鑰空間。您必須包括您目前的區域和至少一個額外的區域。下面的 CQL 語句就是這樣的一個例子。

```
CREATE KEYSPACE mykeyspace
WITH REPLICATION = {'class':'NetworkTopologyStrategy', 'us-east-1':'3', 'ap-
southeast-1':'3','eu-west-1':'3' };
```

密鑰空間中的所有表都使用與密鑰空間相同的複製策略。您無法在資料表層級變更複寫策略。

NetworkTopologyStrategy— 每個區域的複寫係數為三 AWS 區域，因為 Amazon Keyspaces 預設會在同一個區域內的三個可用區域複寫資料。

> **Note**
>
> 建立多區域金鑰空間時，Amazon Keyspaces space 會在您的帳戶中建立名稱AWSServiceRoleForAmazonKeyspacesReplication的服務連結角色。此角色可讓 Amazon Keyspaces 代表您將寫入複寫到多區域表格的所有複本。如需進一步了解，請參閱 the section called “多區域複製”。

您可以使用 CQL 陳述式來查詢索引system_multiregion_info鍵空間中的tables資料表，以程式設計方式列出區域和您指定的多區域資料表的狀態。下面的代碼是這樣的一個例子。

```
SELECT * from system_multiregion_info.tables WHERE keyspace_name = 'mykeyspace' AND
 table_name = 'mytable';
```

陳述式的輸出如下所示：

```
 keyspace_name  | table_name     | region         | status
----------------+----------------+----------------+--------
```

```
 mykeyspace     | mytable        | us-east-1      | ACTIVE
 mykeyspace     | mytable        | ap-southeast-1 | ACTIVE
 mykeyspace     | mytable        | eu-west-1      | ACTIVE
```

使用默認設置（CQL）創建多區域表

若要使用預設設定建立多區域表格，您可以使用下列範例。

```
CREATE TABLE mykeyspace.mytable(pk int, ck int, PRIMARY KEY (pk, ck))
    WITH CUSTOM_PROPERTIES = {
 'capacity_mode':{
  'throughput_mode':'PAY_PER_REQUEST'
 },
 'point_in_time_recovery':{
  'status':'enabled'
 },
 'encryption_specification':{
  'encryption_type':'AWS_OWNED_KMS_KEY'
 },
 'client_side_timestamps':{
  'status':'enabled'
 }
};
```

建立具有佈建容量模式和 auto 擴展 (CQL) 的多區域表

若要在佈建模式中建立具有 auto 擴展功能的多區域表格，您必須先`CUSTOM_PROPERTIES`為表格定義來指定容量模式。指定佈建的容量模式後，您可以使用來設定表格的 auto 調整設定`AUTOSCALING_SETTINGS`。

如需有關 auto 擴展設定、目標追蹤原則、目標值和選用設定的詳細資訊，請參閱the section called “使用 CQL 建立具有自動調整比例的新資料表”。

建立多區域表格時，您也可以為表格的每個複本指定不同的讀取容量和讀取 auto 調整設定。您指定的設定會覆寫指定之表格的一般設定 AWS 區域。但是，寫入容量會在所有複本之間保持同步，以確保有足夠的容量可以跨所有區域複寫寫入。

若要定義特定區域中表格複本的讀取容量，您可以將下列參數設定為表格的一部分`replica_updates`：

- 區域

- 佈建的讀取容量單位 (選用)
- 讀取容量的自動縮放設定 (選用)

下列範例顯示佈建模式下多區域表格的CREATE TABLE陳述式。一般的寫入和讀取容量 auto 調整設定相同。但是，讀取 auto 縮放設定會指定 60 秒的額外冷卻時間，然後再擴展或縮減表格的讀取容量。此外，區域美國東部 (維吉尼亞北部) 的讀取容量 auto 調整設定高於其他複本的讀取容量自動調整設定。此外，目標值設定為 70%，而不是 50%。

```
CREATE TABLE mykeyspace.mytable(pk int, ck int, PRIMARY KEY (pk, ck))
WITH CUSTOM_PROPERTIES = {
    'capacity_mode': {
        'throughput_mode': 'PROVISIONED',
        'read_capacity_units': 5,
        'write_capacity_units': 5
    }
} AND AUTOSCALING_SETTINGS = {
    'provisioned_write_capacity_autoscaling_update': {
        'maximum_units': 10,
        'minimum_units': 5,
        'scaling_policy': {
            'target_tracking_scaling_policy_configuration': {
                'target_value': 50
            }
        }
    },
    'provisioned_read_capacity_autoscaling_update': {
        'maximum_units': 10,
        'minimum_units': 5,
        'scaling_policy': {
            'target_tracking_scaling_policy_configuration': {
                'target_value': 50,
                'scale_in_cooldown': 60,
                'scale_out_cooldown': 60
            }
        }
    },
    'replica_updates': {
        'us-east-1': {
            'provisioned_read_capacity_autoscaling_update': {
                'maximum_units': 20,
                'minimum_units': 5,
                'scaling_policy': {
```

```
                    'target_tracking_scaling_policy_configuration': {
                        'target_value': 70
                    }
                }
            }
        }
    }
};
```

更新多區域表格 (CQL) 的佈建容量和 auto 擴展設定

您可以使ALTER TABLE用更新現有表格的容量模式和 auto 調整設定。如果您要更新目前處於隨選容量模式的資料表，則需要capacity_mode這項功能。如果您的表格已經處於佈建容量模式，則可以省略此欄位。

如需有關 auto 擴展設定、目標追蹤原則、目標值和選用設定的詳細資訊，請參閱the section called “使用 CQL 建立具有自動調整比例的新資料表”。

在相同的陳述式中，您也可以更新資料表的replica_updates屬性，更新特定區域中資料表複本的讀取容量和 auto 調整設定。下面的語句是這樣的一個例子。

```
ALTER TABLE mykeyspace.mytable
WITH CUSTOM_PROPERTIES = {
    'capacity_mode': {
        'throughput_mode': 'PROVISIONED',
        'read_capacity_units': 1,
        'write_capacity_units': 1
    }
} AND AUTOSCALING_SETTINGS = {
    'provisioned_write_capacity_autoscaling_update': {
        'maximum_units': 10,
        'minimum_units': 5,
        'scaling_policy': {
            'target_tracking_scaling_policy_configuration': {
                'target_value': 50
            }
        }
    },
    'provisioned_read_capacity_autoscaling_update': {
        'maximum_units': 10,
        'minimum_units': 5,
        'scaling_policy': {
```

```
            'target_tracking_scaling_policy_configuration': {
                'target_value': 50,
                'scale_in_cooldown': 60,
                'scale_out_cooldown': 60
            }
        }
    },
    'replica_updates': {
        'us-east-1': {
            'provisioned_read_capacity_autoscaling_update': {
                'maximum_units': 20,
                'minimum_units': 5,
                'scaling_policy': {
                    'target_tracking_scaling_policy_configuration': {
                        'target_value': 70
                    }
                }
            }
        }
    }
};
```

檢視多區域表格 (CQL) 的佈建容量和 auto 調整設定

若要檢視多區域表格的 auto 調整比例組態，請使用下列指令。

```
SELECT * FROM system_multiregion_info.autoscaling WHERE keyspace_name = 'mykeyspace'
 AND table_name = 'mytable';
```

此命令的輸出如下所示：

```
 keyspace_name  | table_name | region         |
 provisioned_read_capacity_autoscaling_update

                                      | provisioned_write_capacity_autoscaling_update
---------------+------------+---------------
+-----------------------------------------------------------------------------------------------
+-----------------------------------------------------------------------------------------------
  mykeyspace    |  mytable   | ap-southeast-1 | {'minimum_units': 5, 'maximum_units':
 10, 'scaling_policy': {'target_tracking_scaling_policy_configuration':
 {'scale_out_cooldown': 60, 'disable_scale_in': false, 'target_value':
 50, 'scale_in_cooldown': 60}}} | {'minimum_units': 5, 'maximum_units':
 10, 'scaling_policy': {'target_tracking_scaling_policy_configuration':
```

```
 {'scale_out_cooldown': 0, 'disable_scale_in': false, 'target_value': 50,
 'scale_in_cooldown': 0}}}
  mykeyspace    |  mytable   | us-east-1      | {'minimum_units': 5, 'maximum_units':
 20, 'scaling_policy': {'target_tracking_scaling_policy_configuration':
 {'scale_out_cooldown': 60, 'disable_scale_in': false, 'target_value':
 70, 'scale_in_cooldown': 60}}} | {'minimum_units': 5, 'maximum_units':
 10, 'scaling_policy': {'target_tracking_scaling_policy_configuration':
 {'scale_out_cooldown': 0, 'disable_scale_in': false, 'target_value': 50,
 'scale_in_cooldown': 0}}}
  mykeyspace    |  mytable   | eu-west-1      | {'minimum_units': 5, 'maximum_units':
 10, 'scaling_policy': {'target_tracking_scaling_policy_configuration':
 {'scale_out_cooldown': 60, 'disable_scale_in': false, 'target_value':
 50, 'scale_in_cooldown': 60}}} | {'minimum_units': 5, 'maximum_units':
 10, 'scaling_policy': {'target_tracking_scaling_policy_configuration':
 {'scale_out_cooldown': 0, 'disable_scale_in': false, 'target_value': 50,
 'scale_in_cooldown': 0}}}
```

關閉多區域表格 (CQL) 的 auto 調整

您可以使用ALTER TABLE來關閉現有表格的 auto 縮放比例。請注意，您無法關閉個別表格複本的 auto 調度資源調整功能。

在下列範例中，表格讀取容量的 auto 調整功能已關閉。

```
ALTER TABLE mykeyspace.mytable
WITH AUTOSCALING_SETTINGS = {
    'provisioned_read_capacity_autoscaling_update': {
        'autoscaling_disabled': true
    }
};
```

> **Note**
>
> 若要刪除應用 Application Auto Scaling 所使用的服務連結角色，您必須停用帳戶中所有資料表的自動調整功能。 AWS 區域

手動設定多區域表格的佈建容量 (CQL)

如果您必須關閉多區域表格的 auto 動調整規模，您可以使用ALTER TABLE手動為複本表格佈建表格的讀取容量。

```
ALTER TABLE mykeyspace.mytable
WITH CUSTOM_PROPERTIES = {
    'capacity_mode': {
        'throughput_mode': 'PROVISIONED',
        'read_capacity_units': 1,
        'write_capacity_units': 1
    },
    'replica_updates': {
        'us-east-1': {
            'read_capacity_units': 2
         }
    }
};
```

> **Note**
>
> 我們建議對使用佈建容量的多區域表格使用 auto 擴展。如需更多詳細資訊，請參閱 the section called “多區域表”。

使用建 AWS CLI 立和管理多區域資料表

您可以使用 AWS Command Line Interface (AWS CLI) 創建和管理 Amazon Keyspaces 中的多區域密鑰空間和表。

本節提供如何使用建立和管理多區域表格的 AWS CLI範例。您在多區域密鑰空間中創建的所有表都會自動從密鑰空間繼承多區域設置。

如需本主題所述之 Amazon Keyspaces 命 AWS CLI 令的詳細資訊，請參閱 Amazon Keyspaces 的 AWS CLI 命令參考。

主題

- 建立新的多區域金鑰空間 (CLI)
- 使用預設設定 (CLI) 建立新的多區域資料表
- 使用 auto 擴展 (CLI) 在佈建模式中建立新的多區域表格
- 更新多區域表格 (CLI) 的佈建容量和 auto 調整規模設定
- 檢視多區域表格 (CLI) 的佈建容量和 auto 調整規模設定
- 關閉多區域表格 (CLI) 的 auto 調整

- 手動設定多區域表格的佈建容量 (CLI)

建立新的多區域金鑰空間 (CLI)

若要建立多區域金鑰空間，您可以使用下列 CLI 陳述式。在中指定您目前的地區，以及至少一個其他區域`regionList`。

```
aws keyspaces create-keyspace --keyspace-name mykeyspace
             \ --replication-specification
 replicationStrategy=MULTI_REGION,regionList=us-east-1,eu-west-1
```

> **Note**
>
> 建立多區域金鑰空間時，Amazon Keyspaces space 會在您的帳戶中建立名稱`AWSServiceRoleForAmazonKeyspacesReplication`的服務連結角色。此角色可讓 Amazon Keyspaces 代表您將寫入複寫到多區域表格的所有複本。如需進一步了解，請參閱 the section called “多區域複製”。

使用預設設定 (CLI) 建立新的多區域資料表

若要使用預設設定建立多區域表格，您只需要指定結構定義即可。您可以使用下面的例子。

```
aws keyspaces create-table --keyspace-name mykeyspace --table-name mytable
            \ --schema-definition 'allColumns=[{name=pk,type=int}],partitionKeys={name=
 pk}'
```

該命令的輸出是：

```
{
    "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/
table/mytable"
}
```

若要確認資料表的設定，您可以使用下列陳述式。

```
aws keyspaces get-table --keyspace-name mykeyspace --table-name mytable
```

輸出會顯示多區域表格的所有預設設定。

```
{
    "keyspaceName": "mykeyspace",
    "tableName": "mytable",
    "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/
table/mytable",
    "creationTimestamp": "2023-12-19T16:50:37.639000+00:00",
    "status": "ACTIVE",
    "schemaDefinition": {
        "allColumns": [
            {
                "name": "pk",
                "type": "int"
            }
        ],
        "partitionKeys": [
            {
                "name": "pk"
            }
        ],
        "clusteringKeys": [],
        "staticColumns": []
    },
    "capacitySpecification": {
        "throughputMode": "PAY_PER_REQUEST",
        "lastUpdateToPayPerRequestTimestamp": "2023-12-19T16:50:37.639000+00:00"
    },
    "encryptionSpecification": {
        "type": "AWS_OWNED_KMS_KEY"
    },
    "pointInTimeRecovery": {
        "status": "DISABLED"
    },
    "defaultTimeToLive": 0,
    "comment": {
        "message": ""
    },
    "clientSideTimestamps": {
        "status": "ENABLED"
    },
    "replicaSpecifications": [
        {
            "region": "us-east-1",
```

```
            "status": "ACTIVE",
            "capacitySpecification": {
                "throughputMode": "PAY_PER_REQUEST",
                "lastUpdateToPayPerRequestTimestamp": 1702895811.469
            }
        },
        {
            "region": "eu-north-1",
            "status": "ACTIVE",
            "capacitySpecification": {
                "throughputMode": "PAY_PER_REQUEST",
                "lastUpdateToPayPerRequestTimestamp": 1702895811.121
            }
        }
    ]
}
```

使用 auto 擴展 (CLI) 在佈建模式中建立新的多區域表格

若要在具有 auto 擴展配置的佈建模式中建立多區域表格，您可以使用 AWS CLI. 請注意，您必須使用 Amazon Keyspaces CLI `create-table` 命令來設定多區域 auto 擴展設定。這是因為 Amazon Keyspaces 用來代表您執行自動擴展的服務「Application Auto Scaling 用程式自動擴展」不支援多個區域。

如需有關 auto 資源調整設定、目標追蹤原則、目標值和選用設定的詳細資訊，請參閱the section called “使用建立具有自動縮放比例的新資料表 AWS CLI”。

當您在具有 auto 擴展設定的佈建模式中建立新的多區域表格時，您可以指定表格的一般設定，這些設定對複製表格 AWS 區域 的所有資料表都有效。然後，您可以覆寫每個複本的讀取容量設定和讀取 auto 調整設定。但是，寫入容量會在所有複本之間保持同步，以確保有足夠的容量可以跨所有區域複寫寫入。

若要定義特定區域中表格複本的讀取容量，您可以將下列參數設定為表格的一部分`replicaSpecifications`：

- 區域
- 佈建的讀取容量單位 (選用)
- 讀取容量的自動縮放設定 (選用)

當您使用複雜的 auto 調整設定和不同的資料表複本組態建立佈建的多區域表格時，從 JSON 檔案載入資料表的 auto 調整設定和複本組態會很有幫助。

若要使用下列程式碼範例，您可以從 auto-scaling.zip 下載範例 JSON 檔案，然後擷取`auto-scaling.json`和`replication.json`。記下檔案的路徑。

在此範例中，JSON 檔案位於目前的目錄中。如需不同的檔案路徑選項，請參閱如何從檔案載入參數。

```
aws keyspaces create-table --keyspace-name mykeyspace --table-name mytable
           \ --schema-definition 'allColumns=[{name=pk,type=int},
{name=ck,type=int}],partitionKeys=[{name=pk},{name=ck}]'
           \ --capacity-specification
 throughputMode=PROVISIONED,readCapacityUnits=1,writeCapacityUnits=1
           \ --auto-scaling-specification file://auto-scaling.json
           \ --replica-specifications file://replication.json
```

更新多區域表格 (CLI) 的佈建容量和 auto 調整規模設定

若要更新現有資料表的佈建模式和 auto 調整設定，您可以使用 AWS CLI `update-table`命令。

請注意，您必須使用 Amazon Keyspaces CLI 命令來建立或修改多區域 auto 擴展設定。這是因為 Amazon Keyspaces 用來代表您執行表格容量 auto 擴展的服務「Application Auto Scaling 用程式自動擴展」不支援多個 AWS 區域。

當您更新多區域表格的佈建模式或 auto 調整設定時，您可以更新表格每個複本的讀取容量設定和讀取 auto 調整規模組態。

但是，寫入容量會在所有複本之間保持同步，以確保有足夠的容量可以跨所有區域複寫寫入。若要更新特定區域中表格複本的讀取容量，您可以變更表格的下列其中一個選用參數`replicaSpecifications`：

- 佈建的讀取容量單位 (選用)
- 讀取容量的自動縮放設定 (選用)

當您使用複雜的 auto 調整設定和不同的資料表複本設定來更新多區域表格時，從 JSON 檔案載入資料表的 auto 調整設定和複本組態會很有幫助。

若要使用下列程式碼範例，您可以從 auto-scaling.zip 下載範例 JSON 檔案，然後擷取`auto-scaling.json`和`replication.json`。記下檔案的路徑。

在此範例中，JSON 檔案位於目前的目錄中。如需不同的檔案路徑選項，請參閱如何從檔案載入參數。

```
aws keyspaces update-table --keyspace-name mykeyspace --table-name mytable
          \ --capacity-specification
 throughputMode=PROVISIONED,readCapacityUnits=1,writeCapacityUnits=1
          \ --auto-scaling-specification file://auto-scaling.json
          \ --replica-specifications file://replication.json
```

檢視多區域表格 (CLI) 的佈建容量和 auto 調整規模設定

若要檢視多區域表格的 auto 縮放配置，您可以使用此`get-table-auto-scaling-settings`操作。下面的 CLI 命令就是這樣的一個例子。

```
aws keyspaces get-table-auto-scaling-settings --keyspace-name mykeyspace --table-name
 mytable
```

您應該會看到下列輸出。

```
{
    "keyspaceName": "mykeyspace",
    "tableName": "mytable",
    "resourceArn": "arn:aws:cassandra:us-east-1:777788889999:/keyspace/mykeyspace/
table/mytable",
    "autoScalingSpecification": {
        "writeCapacityAutoScaling": {
            "autoScalingDisabled": false,
            "minimumUnits": 5,
            "maximumUnits": 10,
            "scalingPolicy": {
                "targetTrackingScalingPolicyConfiguration": {
                    "disableScaleIn": false,
                    "scaleInCooldown": 0,
                    "scaleOutCooldown": 0,
                    "targetValue": 50.0
                }
            }
        },
        "readCapacityAutoScaling": {
            "autoScalingDisabled": false,
            "minimumUnits": 5,
            "maximumUnits": 20,
            "scalingPolicy": {
                "targetTrackingScalingPolicyConfiguration": {
                    "disableScaleIn": false,
```

```
                    "scaleInCooldown": 60,
                    "scaleOutCooldown": 60,
                    "targetValue": 70.0
                }
            }
        }
    },
    "replicaSpecifications": [
        {
            "region": "us-east-1",
            "autoScalingSpecification": {
                "writeCapacityAutoScaling": {
                    "autoScalingDisabled": false,
                    "minimumUnits": 5,
                    "maximumUnits": 10,
                    "scalingPolicy": {
                        "targetTrackingScalingPolicyConfiguration": {
                            "disableScaleIn": false,
                            "scaleInCooldown": 0,
                            "scaleOutCooldown": 0,
                            "targetValue": 50.0
                        }
                    }
                },
                "readCapacityAutoScaling": {
                    "autoScalingDisabled": false,
                    "minimumUnits": 5,
                    "maximumUnits": 20,
                    "scalingPolicy": {
                        "targetTrackingScalingPolicyConfiguration": {
                            "disableScaleIn": false,
                            "scaleInCooldown": 60,
                            "scaleOutCooldown": 60,
                            "targetValue": 70.0
                        }
                    }
                }
            }
        },
        {
            "region": "eu-north-1",
            "autoScalingSpecification": {
                "writeCapacityAutoScaling": {
                    "autoScalingDisabled": false,
```

```
                    "minimumUnits": 5,
                    "maximumUnits": 10,
                    "scalingPolicy": {
                        "targetTrackingScalingPolicyConfiguration": {
                            "disableScaleIn": false,
                            "scaleInCooldown": 0,
                            "scaleOutCooldown": 0,
                            "targetValue": 50.0
                        }
                    }
                },
                "readCapacityAutoScaling": {
                    "autoScalingDisabled": false,
                    "minimumUnits": 5,
                    "maximumUnits": 10,
                    "scalingPolicy": {
                        "targetTrackingScalingPolicyConfiguration": {
                            "disableScaleIn": false,
                            "scaleInCooldown": 60,
                            "scaleOutCooldown": 60,
                            "targetValue": 50.0
                        }
                    }
                }
            }
        }
    ]
}
```

關閉多區域表格 (CLI) 的 auto 調整

您可以使用指 AWS CLI `update-table`令來關閉現有表格的 auto 縮放比例。請注意，您無法關閉個別表格複本的 auto 調度資源調整功能。

在下列範例中，表格讀取容量的 auto 調整功能已關閉。

```
aws keyspaces update-table --keyspace-name mykeyspace --table-name mytable
           \ --auto-scaling-specification
 readCapacityAutoScaling={autoScalingDisabled=true}
```

> **Note**
>
> 若要刪除應用 Application Auto Scaling 所使用的服務連結角色，您必須停用帳戶中所有資料表的自動調整功能。 AWS 區域

手動設定多區域表格的佈建容量 (CLI)

如果您必須關閉多區域表格的 auto 動調整規模，您可以使用`update-table`手動為複本表格佈建表格的讀取容量。

```
aws keyspaces update-table --keyspace-name mykeyspace --table-name mytable
          \ --capacity-specification
 throughputMode=PROVISIONED,readCapacityUnits=1,writeCapacityUnits=1
          \ --replica-specifications region="us-east-1",readCapacityUnits=5
```

> **Note**
>
> 我們建議對使用佈建容量的多區域表格使用 auto 擴展。如需更多詳細資訊，請參閱 the section called “多區域表”。

Amazon Keyspaces 的 Point-in-time 復原 (適用於 Apache Cassandra) 的 Peyspaces 的復原

Point-in-time 復原 (PITR) 提供表格資料的持續備份，協助保護 Amazon Keyspaces 資料表免於意外寫入或刪除操作。

例如，假設測試指令碼不小心寫入生產 Amazon Keyspaces 資料表。透過 point-in-time 復原，您可以將該資料表的資料還原到過去 35 天內啟用 PITR 後的任何時間。若您在啟用 point-in-time 復原功能的狀態下刪除資料表，您可以在 35 天內查詢已刪除資料表的資料 (無需額外費用)，並將其還原到刪除點之前的狀態。

您可以使用主控台、AWS SDK 和 () 或卡桑德拉查詢語言AWS Command Line Interface (CQLAWS CLI)，將 Amazon Keyspaces 間資料表還原到某個時間點。如需詳細資訊，請參閱還原 Amazon Amazon Keyspaces 間資料表至某個時間點。

Point-in-time 作業對基底資料表沒有效能或可用性的影響，而且還原資料表不會耗用額外的輸送量。

如需 PITR 配額的詳細資訊，請參閱配額。

如需定價的詳細資訊，請參閱 Amazon Keyspaces (適用於 Apache Cassandra) 的定價。

主題

- point-in-time 恢復如何在 Amazon Keyspaces 中工作
- 還原 Amazon Amazon Keyspaces 間資料表至某個時間點

point-in-time 恢復如何在 Amazon Keyspaces 中工作

本節提供 Amazon Keyspaces point-in-time 復原 (PITR) 運作方式的概觀。有關定價的更多信息，請參閱 Amazon Keyspaces（阿帕奇卡桑德拉）定價。

主題

- 啟用 point-in-time 復原 (PITR)
- 還原資料表所需的權限
- PITR 持續備份的時間範圍
- PITR 還原設定
- PITR 還原加密表格

- 多區域表格的 PITR 還原
- 使用 PITR 的資料表還原時間
- Amazon Keyspaces PITR 並與服務集成 AWS

啟用 point-in-time 復原 (PITR)

您可以使用主控台來啟用 PITR，也可以透過程式設計方式啟用它。

透過主控台啟用 PITR

您可以在「自訂設定」選項下管理新資料表的 PITR 設定。依預設，會在透過主控台建立的新表格上啟用 PITR。

若要啟用現有表格的 PITR，請完成以下步驟。

1. 登錄到AWS Management Console，然後打開 Amazon Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。
2. 在導覽窗格中，選擇 [表格]，然後選取您要編輯的表格。
3. 在 [備份] 索引標籤上選擇 [編輯]。
4. 在 [編輯 point-in-time 復原設定值] 區段中，選取 [啟用 P oint-in-time 復原]。

您可以按照以下步驟隨時在表格上停用 PITR。

1. 登錄到AWS Management Console，然後打開 Amazon Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。
2. 在導覽窗格中，選擇 [表格]，然後選取您要編輯的表格。
3. 在 [備份] 索引標籤上選擇 [編輯]。
4. 在 [編輯 point-in-time 復原設定值] 區段中，清除 [啟用 P oint-in-time 復原] 核取方塊。

> **Important**
>
> 停用 PITR 會立即刪除備份歷史記錄，即使您在 35 天內重新啟用表格上的 PITR 也一樣。

若要瞭解如何使用主控台還原資料表，請參閱the section called “還原資料表至某個時間點 (主控台)”。

使用啟用 PITR AWS CLI

您可以使用 UpdateTable API 管理表格的 PITR 設定。

使用建立新表格時AWS CLI，您必須在建立新表格時明確啟用 PITR。

若要在建立新資料表時啟用 PITR，您可以使用下列AWS CLI指令做為範例。該命令已分成單獨的行以提高可讀性。

```
aws keyspaces create-table --keyspace-name 'myKeyspace' --table-name 'myTable'
            --schema-definition 'allColumns=[{name=id,type=int},{name=name,type=text},
{name=date,type=timestamp}],partitionKeys=[{name=id}]'
            --point-in-time-recovery 'status=ENABLED'
```

> **Note**
>
> 如果未指定 point-in-time 復原值，依預設會停用 point-in-time復原。

要確認表的 point-in-time 恢復設置，可以使用以下AWS CLI命令。

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

若要使用啟用現有資料表的 PITRAWS CLI，請執行下列命令。

```
aws keyspaces update-table --keyspace-name 'myKeyspace' --table-name 'myTable' --point-
in-time-recovery 'status=ENABLED'
```

若要停用現有資料表上的 PITR，請執行下列AWS CLI命令。

```
aws keyspaces update-table --keyspace-name 'myKeyspace' --table-name 'myTable' --point-
in-time-recovery 'status=DISABLED'
```

> **Important**
>
> 停用 PITR 會立即刪除備份歷史記錄，即使您在 35 天內重新啟用表格上的 PITR 也一樣。

使用 CQL 啟用 PITR

您可以使用point_in_time_recovery自訂屬性來管理表格的 PITR 設定。

使用 CQL 建立新資料表時，您必須在建立新資料表時明確啟用 PITR。

若要在建立新資料表時啟用 PITR，您可以使用下列 CQL 命令做為範例。

```
CREATE TABLE "my_keyspace1"."my_table1"(
 "id" int,
 "name" ascii,
 "date" timestamp,
 PRIMARY KEY("id"))
WITH CUSTOM_PROPERTIES = {
 'capacity_mode':{'throughput_mode':'PAY_PER_REQUEST'},
 'point_in_time_recovery':{'status':'enabled'}
}
```

> **Note**
>
> 如果未指定任何 point-in-time 復原自訂內容，依預設會停用 point-in-time復原。

若要為使用 CQL 的現有資料表啟用 PITR，請執行下列 CQL 命令。

```
ALTER TABLE mykeyspace.mytable
WITH custom_properties = {'point_in_time_recovery': {'status': 'enabled'}}
```

若要停用現有資料表上的 PITR，請執行下列 CQL 命令。

```
ALTER TABLE mykeyspace.mytable
WITH custom_properties = {'point_in_time_recovery': {'status': 'disabled'}}
```

> **Important**
>
> 停用 PITR 會立即刪除備份歷史記錄，即使您在 35 天內重新啟用表格上的 PITR 也一樣。

如需 CQL 語言參考中的詳細資訊，請參閱the section called “CREATE TABLE”和the section called “ALTER TABLE”。若要瞭解如何使用 CQL 還原資料表，請參閱the section called “還原資料表至某個時間點”。

還原資料表所需的權限

若要成功還原資料表，IAM 使用者或角色需要下列最低權限：

- `cassandra:Restore`— 還原目標表格需要執行還原動作。
- `cassandra:Select`— 需要選取動作才能從來源表格中讀取。
- `cassandra:TagResource`— 標籤動作是選用的，只有在還原作業新增標籤時才需要。

以下是原則範例，該原則會授與使用者在金鑰空間`mykeyspace`中還原資料表的最低必要權限。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "cassandra:Restore",
            "cassandra:Select"
         ],
         "Resource":[
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/*",
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
         ]
      }
   ]
}
```

根據其他選取的功能，可能需要還原表格的其他權限。例如，如果來源資料表使用客戶受管金鑰在靜態時加密，Amazon Keyspaces 必須具有存取來源資料表客戶受管金鑰的權限，才能成功還原表格。如需詳細資訊，請參閱 the section called “PITR 和加密資料表”。

如果您使用 IAM 政策搭配條件金鑰來限制特定來源的傳入流量，則必須確保 Amazon Keyspaces 具有代表您主體執行還原作業的權限。如果您的政策將傳入流量限制為以下任何一種，則必須在 IAM 政策中新增`aws:ViaAWSService`條件金鑰：

- VPC 端端點 `aws:SourceVpce`

- IP 範圍包含 `aws:SourceIp`
- 具有的 VPC `aws:SourceVpc`

當任何AWS服務使用主體的認證發出要求時，`aws:ViaAWSService`條件索引鍵允許存取。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件金鑰。

以下是將來源流量限制為特定 IP 地址的政策範例，並允許 Amazon Keyspaces 代表主體還原表格。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"CassandraAccessForCustomIp",
         "Effect":"Allow",
         "Action":"cassandra:*",
         "Resource":"*",
         "Condition":{
            "Bool":{
               "aws:ViaAWSService":"false"
            },
            "ForAnyValue:IpAddress":{
               "aws:SourceIp":[
                  "123.45.167.89"
               ]
            }
         }
      },
      {
         "Sid":"CassandraAccessForAwsService",
         "Effect":"Allow",
         "Action":"cassandra:*",
         "Resource":"*",
         "Condition":{
            "Bool":{
               "aws:ViaAWSService":"true"
            }
         }
      }
   ]
}
```

如需使用aws:ViaAWSService全域條件索引鍵的範例原則，請參閱the section called “VPC 端點政策和 Amazon Keyspaces point-in-time 恢復 (PITR)”。

PITR 持續備份的時間範圍

Amazon Keyspaces 間 PITR 使用兩個時間戳記來維持可還原備份可供表格使用的時間範圍。

- 最早可還原時間 — 標記最早可還原備份的時間。最早的可還原備份可追溯至 35 天或啟用 PITR 時 (以較新的日期為準)。無法修改 35 天的最大備份時段。
- 目前時間 — 最新可還原備份的時間戳記為目前時間。如果還原期間未提供時間戳記，則會使用目前時間。

啟用 PITR 時，您可以還原到EarliestRestorableDateTime和CurrentTime之間的任何時間點。您只能將表格資料還原到啟用 PITR 的時間。

如果停用 PITR 並稍後再次重新啟用，則會將第一個可用備份的開始時間重設為重新啟用 PITR 的時間。這表示停用 PITR 會清除您的備份記錄。

> **Note**
>
> 資料表上的資料定義語言 (DDL) 作業 (例如結構描述變更) 會以非同步方式執行。您只能在還原的表格資料中看到已完成的作業，但如果來源表格在還原時正在進行中，您可能會看到其他動作。如需 DDL 陳述式的清單，請參閱the section called “DDL 陳述式”。

表不必處於活動狀態才能恢復。如果已刪除的資料表已啟用 PITR，且刪除發生在備份時段內 (或過去 35 天內)，您也可以還原已刪除的表格。

> **Note**
>
> 如果使用與先前刪除的資料表相同的限定名稱 (例如 mykeyspace .mytable) 建立的新資料表，則刪除的資料表將無法復原。如果您嘗試從控制台執行此操作，則會顯示警告。

PITR 還原設定

使用 PITR 還原表格時，Amazon Keyspaces 會根據選取的時間戳記 (day:hour:minute:second) 將來源資料表的結構描述和資料還原到狀態至新資料表。PITR 不會覆寫現有的資料表。

除了資料表的結構定義和資料之外，PITR 還原custom_properties來源資料表中的。與資料表的資料不同，資料表的資料會根據在最早還原時間與目前時間之間選取的時間戳記進行還原，自訂屬性一律會根據目前時間的資料表設定來還原。

還原表格的設定會與來源資料表的設定值與啟動還原時的時間戳記相符。如果您要在還原期間覆寫這些設定，您可以使用WITH custom_properties。自訂內容包括下列設定。

- 讀取/寫入容量模式
- 佈建輸送量容量設定
- PITR 設定

如果表格處於已佈建容量模式且啟用了 auto 調整規模，還原作業也會還原資料表的 auto 調整設定。您可以使用 CQL 中的autoscaling_settings參數或 autoScalingSpecification CLI 覆寫它們。如需 auto 縮放設定的詳細資訊，請參閱the section called “透過 auto 擴充管理輸送量容量”。

當您執行完整資料表還原時，還原資料表的所有資料表設定都是來自還原時來源資料表的目前設定。

例如，假設資料表的佈建輸送量最近降低至 50 個讀取容量單位及 50 個寫入容量單位。然後，您將表格的狀態還原到三週前。此時，其佈建輸送量設定為 100 個讀取容量單位和 100 個寫入容量單位。在這種情況下，Amazon Keyspaces 會將表格資料還原到該時間點，但會使用目前佈建的輸送量設定 (50 個讀取容量單位和 50 個寫入容量單位)。

下列設定不會還原，您必須為新表格手動設定這些設定。

- AWS Identity and Access Management (IAM) 政策
- Amazon CloudWatch 指標和警報
- 標籤（ 可以使WITH TAGS用添加到 CQL RESTORE 語句 ）

PITR 還原加密表格

當您使用 PITR 還原表格時，Amazon Keyspaces 會還原來源表格的加密設定。如果使用 AWS 擁有的金鑰 (預設值) 加密資料表，則會以相同的設定自動還原資料表。如果您要還原的表使用客戶受管金鑰加密，則 Amazon Keyspaces 必須可存取相同的客戶受管金鑰，才能還原表格資料。

您可以在還原時變更表格的加密設定。若要從客戶管理金鑰變更AWS 擁有的金鑰為客戶管理金鑰，您需要在還原時提供有效且可存取的客戶管理金鑰。

如果您想要從客戶受管金鑰變更為AWS 擁有的金鑰，請確認 Amazon Keyspaces 可存取來源資料表的客戶受管金鑰，以使用AWS 擁有的金鑰. 若要取得有關表格的靜態加密設定的更多資訊，請參閱the section called “運作方式”。

> **Note**
>
> 如果表格因為 Amazon Keyspaces space 無法存取客戶受管金鑰而遭到刪除，您需要確保 Amazon Keyspaces space 可以存取客戶受管金鑰，然後再嘗試還原表格。如果 Amazon Keyspace 無法存取該金鑰，則無法還原使用客戶受管金鑰加密的表格。如需詳細資訊，請參閱AWS Key Management Service開發人員指南中的金鑰存取疑難排解。

多區域表格的 PITR 還原

您可以使用 PITR 還原多區域表格。若要成功執行還原作業，來源表格和目的地資料表都必須複製到相同的資料表AWS 區域。

Amazon Keyspaces 間會在屬於金鑰空間一部分的每個複寫區域中還原來源表格的設定。您也可以在還原作業期間覆寫設定。如需還原期間可變更之設定的詳細資訊，請參閱the section called “還原設定”。

如需有關多區域複寫的詳細資訊，請參閱the section called “運作方式”。

使用 PITR 的資料表還原時間

還原資料表所花費的時間取決於多個因素，而且並不總是直接與資料表的大小相關。

以下是還原時間的一些注意事項。

- 將備份還原至新資料表。執行建立新表格和啟動恢復程序的所有動作，最多可能需要 20 分鐘的時間(即使表格是空的)。
- 具有分佈良好資料模型的大型資料表的還原時間可能需要數小時或更長時間。
- 如果來源資料表包含明顯偏斜的資料，還原的時間可能會增加。例如，如果您的資料表的主索引鍵使用年份中的月份做為分區索引鍵，而您的所有資料都是從 12 月份開始，就表示資料偏差了。

規劃災難復原的最佳實務是定期記錄平均還原完成時間，並確認這些時間如何影響整體復原時間目標。

Amazon Keyspaces PITR 並與服務集成 AWS

下列 PITR 作業會使用來記錄，AWS CloudTrail以啟用持續監視和稽核。

- 建立啟用或停用 PITR 的新表格。
- 在現有表格上啟用或停用 PITR。
- 還原使用中或已刪除的表格。

如需詳細資訊，請參閱 記錄 Amazon Keyspaces API 調用 AWS CloudTrail。

您可以使用執行下列 PITR 動作AWS CloudFormation。

- 建立啟用或停用 PITR 的新表格。
- 在現有表格上啟用或停用 PITR。

如需詳細資訊，請參閱使AWS CloudFormation用者指南中的 Cassandra 資源類型參考。

還原 Amazon Amazon Keyspaces 間資料表至某個時間點

亞馬遜 Keyspaces 間（對於阿帕奇卡桑德拉） point-in-time 恢復（PITR）允許您將亞馬遜 Keyspaces 間表數據恢復到過去 35 天內的任何時間點。本教程的第一部分說明如何使用 Amazon Keyspaces 間控制台（ ）和卡桑德拉查詢語言AWS Command Line Interface（CQLAWS CLI）將表還原到某個時間點。第二部分說明如何使用AWS CLI和 CQL 還原已刪除的資料表。

主題

開始之前

如果您尚未這麼做，則必須為使用者設定適當的許可，以將 Amazon Keyspace 還原 Amazon Keyspaces 料表。在AWS Identity and Access Management (IAM) 中，受AWS管政

策AmazonKeyspacesFullAccess包括還原 Amazon Keyspaces 表格的許可。如需實作具有最低必要權限之原則的詳細步驟，請參閱the section called “還原權限”。

還原資料表至某個時間點 (主控台)

以下範例示範如何使用 Amazon Keyspace 主控台將名為某個時間點還原的現有資料表還原mytable至某個時間點。

> **Note**
>
> 此程序假設您已經啟用還 point-in-time 原復原。若要啟用mytable表格的 PITR，請遵循中的步驟the section called “使用主控台”。

1. 登入AWS Management Console，並在 https://console.aws.amazon.com/keyspaces/home 開啟 Amazon Keyspaces 主控台https://console.aws.amazon.com/keyspaces/home。
2. 在主控台左側的導覽窗格中，選擇 Tables (資料表)。
3. 在資料表清單中，選擇 mytable 資料表。
4. 在mytable表格的 [備份] 索引標籤的 [Point-in-time 復原] 區段中，選擇 [還原]。
5. 輸入 **mytable_restored** 做為新資料表的名稱。
6. 若要定義還原作業的時間點，您可以在兩個選項之間進行選擇：
   - 選取預先設定的「最早」時間。
   - 選取「指定日期和時間」，然後輸入要將新表格還原到哪個日期和時間。

   > **Note**
   >
   > 您可還原至最早時間內的任何時間點。Amazon Keyspaces space 會將資料表還原至根據所選日期和時間 (天:小時/分鐘/秒) 將資料表還原至狀態。

7. 選擇恢復開始還原過程。

   正在還原的資料表會顯示為 Restoring (正在還原) 狀態。還原程序完成後，mytable_restored 資料表的狀態會變更為 Active (作用中)。

> **Important**
>
> 復原過程中，請勿修改或刪除 AWS Identity and Access Management (IAM) 授予 IAM 實體的政策 (例如使用者、群組或角色) 許可以執行復原。否則，可能會造成意外行為。例如，假設您在還原該資料表時移除了資料表的寫入權限。在此案例中，基礎 `RestoreTableToPointInTime` 操作無法將任何還原的資料寫入資料表。只有在復原操作完成後，才能修改或刪除許可。

還原資料表至某個時間點AWS CLI

以下程序顯示如何使用 AWS CLI 將名為 `myTable` 的現有資料表還原至某個時間點。

1. 在第一個步驟中，您會建立名為已啟用 PITR`myTable` 的簡單資料表。為了方便閱讀，該命令已分為單獨的行。

```
aws keyspaces create-table --keyspace-name 'myKeyspace' --table-name 'myTable'
            --schema-definition 'allColumns=[{name=id,type=int},
{name=name,type=text},{name=date,type=timestamp}],partitionKeys=[{name=id}]'
            --point-in-time-recovery 'status=ENABLED'
```

2. 確認新表格的內容，並檢`earliestRestorableTimestamp`閱 PITR 的。

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

此令命令命令的輸出結果如下所示：

```
{
    "keyspaceName": "myKeyspace",
    "tableName": "myTable",
    "resourceArn": "arn:aws:cassandra:us-east-1:111222333444:/keyspace/myKeyspace/
table/myTable",
    "creationTimestamp": "2022-06-20T14:34:57.049000-07:00",
    "status": "ACTIVE",
    "schemaDefinition": {
        "allColumns": [
            {
                "name": "id",
                "type": "int"
```

```
            },
            {
                "name": "date",
                "type": "timestamp"
            },
            {
                "name": "name",
                "type": "text"
            }
        ],
        "partitionKeys": [
            {
                "name": "id"
            }
        ],
        "clusteringKeys": [],
        "staticColumns": []
    },
    "capacitySpecification": {
        "throughputMode": "PAY_PER_REQUEST",
        "lastUpdateToPayPerRequestTimestamp": "2022-06-20T14:34:57.049000-07:00"
    },
    "encryptionSpecification": {
        "type": "AWS_OWNED_KMS_KEY"
    },
    "pointInTimeRecovery": {
        "status": "ENABLED",
        "earliestRestorableTimestamp": "2022-06-20T14:35:13.693000-07:00"
    },
    "defaultTimeToLive": 0,
    "comment": {
        "message": ""
    }
}
```

您可以將使用中的表格還原為目前時 point-in-time 間`earliestRestorableTimestamp`與目前時間之間的任何一個間隔。目前時間為預設值。

3. 若要還原資料表至某個時間點，以 ISO 8601 格式將資料表還原至某`restore_timestamp`個時間點。您可以每秒間間隔選擇過去 35 天內的任何時間點。例如，下列命令可復原資料表至`EarliestRestorableDateTime`。

```
aws keyspaces restore-table --source-keyspace-name 'myKeyspace' --source-
table-name 'myTable' --target-keyspace-name 'myKeyspace' --target-table-name
 'myTable_restored' --restore-timestamp "2022-06-20 21:35:14.693"
```

此令命令命令命令的輸出結果含有該還原資料表的 ARN。

```
{
    "restoredTableARN": "arn:aws:cassandra:us-east-1:111222333444:/keyspace/
myKeyspace/table/myTable_restored"
}
```

若要將表格還原到目前的時間，您可以省略`restore-timestamp`。

```
aws keyspaces restore-table --source-keyspace-name 'myKeyspace' --source-
table-name 'myTable' --target-keyspace-name 'myKeyspace' --target-table-name
 'myTable_restored1'"
```

Important

復原過程中，請勿修改或刪除 AWS Identity and Access Management (IAM) 授予 IAM 實體的政策 (例如使用者、群組或角色) 許可以執行復原。否則，可能會造成意外行為。例如，假設您在還原該資料表時移除了資料表的寫入權限。在此案例中，基礎 `RestoreTableToPointInTime` 操作無法將任何還原的資料寫入資料表。只有在復原操作完成後，才能修改或刪除許可。

還原資料表至某個時間點

以下程序示範如何使用 CQL 將名為某個時間點還原的現有資料表還原`mytable`至某個時間點。

Note

此程序假設您已經啟用還 point-in-time 原復原。若要在表格上啟用 PITR，請遵循中的步驟the section called “定制列表”。

1. 您可以將使用中表格還原到目前時 point-in-time 間之間earliest_restorable_timestamp。目前時間為預設值。

   若要確認已啟用mytable資料表的 point-in-time 復原功能，請依照下列方system_schema_mcs.tables式查詢。

   ```
   SELECT custom_properties
   FROM system_schema_mcs.tables
   WHERE keyspace_name = 'mykeyspace' AND table_name = 'mytable';
   ```

   如下面oint-in-time 的示範如下面的輸出結果如下列範例示範如下所示

   ```
   custom_properties
   -----------------
   {
     ...,
       "point_in_time_recovery": {
       "earliest_restorable_timestamp":"2020-06-30T19:19:21.175Z"
       "status":"enabled"
     }
   }
   ```

2. 將資料表還原至 ISO 8601 格式restore_timestamp中指定的時間點。在此情況下，以將mytable資料表還原至目前的時間。您可以省略子WITH restore_timestamp = ...句。如果沒有子句，則會使用目前的時間戳記。

   ```
   RESTORE TABLE mykeyspace.mytable_restored
   FROM TABLE mykeyspace.mytable;
   ```

   您也可還原至特定時間點。您可以指定過去 35 天內的任何時間點。例如，下列命令可復原資料表至 EarliestRestorableDateTime。

   ```
   RESTORE TABLE mykeyspace.mytable_restored
   FROM TABLE mykeyspace.mytable
   WITH restore_timestamp = '2020-06-30T19:19:21.175Z';
   ```

   如需完整的語法描述，請參閱the section called “還原表格”語言參考中的。

若要確認資料表還原是否成功，請查詢system_schema_mcs.tables以確認表格的狀態。

```
SELECT status
FROM system_schema_mcs.tables
WHERE keyspace_name = 'mykeyspace' AND table_name = 'mytable_restored'
```

該查詢顯示以下輸出。

```
status
------
RESTORING
```

正在還原的資料表會顯示為 Restoring (正在還原) 狀態。還原程序完成後，mytable_restored 資料表的狀態會變更為 Active (作用中)。

> **Important**
>
> 復原過程中，請勿修改或刪除 AWS Identity and Access Management (IAM) 授予 IAM 實體的政策 (例如使用者、群組或角色) 許可以執行復原。否則，可能會造成意外行為。例如，假設您在還原該資料表時移除了資料表的寫入權限。在此案例中，基礎 RestoreTableToPointInTime 操作無法將任何還原的資料寫入資料表。只有在復原操作完成後，才能修改或刪除許可。

使用恢復已刪除的表AWS CLI

下列程序顯示如何使用AWS CLI將名為myTable刪除時間的已刪除資料表還原為止。

> **Note**
>
> 此程序假設已在刪除的表格上啟用 PITR。

1. 刪除您在上一個自學課程中建立的資料表。

   ```
   aws keyspaces delete-table --keyspace-name 'myKeyspace' --table-name 'myTable'
   ```

2. 使用以下命令將已刪除的表恢復到刪除的時間。

```
aws keyspaces restore-table --source-keyspace-name 'myKeyspace' --source-
table-name 'myTable' --target-keyspace-name 'myKeyspace' --target-table-name
 'myTable_restored2'
```

此令命令命令命令命令的輸出結果含有該還原資料表的 ARN。

```
{
    "restoredTableARN": "arn:aws:cassandra:us-east-1:111222333444:/keyspace/
myKeyspace/table/myTable_restored2"
}
```

使用 CQL 恢復已刪除的表

下列程序顯示如何使用 CQL 還原名為刪除時間mytable的已刪除資料表。

> **Note**
>
> 此程序假設已在刪除的表格上啟用 PITR。

1. 若要確認已刪除的資料表已啟用 point-in-time 復原功能，請查詢系統資料表。只會顯示啟用 point-in-time 復原的表格。

```
SELECT custom_properties
FROM system_schema_mcs.tables_history
WHERE keyspace_name = 'mykeyspace' AND table_name = 'my_table';
```

該查詢顯示以下輸出。

```
custom_properties
------------------
{
    ...,
   "point_in_time_recovery":{
      "restorable_until_time":"2020-08-04T00:48:58.381Z",
      "status":"enabled"
   }
```

```
}
```

2. 使用下列範例陳述式將資料表還原至刪除時間。

```
RESTORE TABLE mykeyspace.mytable_restored
FROM TABLE mykeyspace.mytable;
```

透過使用 Amazon Keyspaces 存留時間 (TTL) 讓資料過期

Amazon Keyspaces (適用於 Apache Cassandra) 存留時間 (TTL) 透過資料表過期資料表中使用存留時間 (TTL) 讓存留時間 (TTL) 讓存留時間 (TTL) 讓存留時間 (TTL) 讓存留時間 系統會根據您設定的「存留時間」值，自動從表格中刪除不再需要的資料。這樣可以更輕鬆地遵守基於業務、產業或法規要求的資料保留政策，這些規定定義資料需要保留的時間長度，或指定何時必須刪除資料。

舉例來說，您可以在 AdTech 應用程式中使用 TTL 來排定特定廣告的資料到期且客戶無法再看到的時間。您也可以使用 TTL 自動淘汰舊資料，並節省儲存成本。您可以為整個資料表設定預設 TTL 值，並覆寫個別列和欄的該值。TTL 作業不會影響應用程式的效能。此外，標記為以 TTL 過期的列數和欄數不會影響資料表的可用性。

Amazon Keyspaces 會自動篩選出過期的資料，以便不會在查詢結果中傳回過期的資料，也不會用於資料操作語言 (DML) 陳述式。Amazon Keyspaces 通常會在到期日期後 10 天內從儲存中刪除過期的資料。在極少數情況下，如果基礎儲存分割區上有持續的活動以保護可用性，Amazon Keyspaces space 可能無法在 10 天內刪除資料。在這些情況下，一旦分區上的流量減少，Amazon Keyspaces 會繼續嘗試刪除過期的資料。從儲存空間中永久刪除資料後，您就不會產生儲存費用。如需詳細資訊，請參閱the section called “運作方式”。

您可以設置，修改，或通過使用控制台或卡桑德拉查詢語言（CQL）禁用新的和現有的表默認 TTL 設置。在已設定預設 TTL 的資料表上，您可以使用卡桑德拉查詢語言 (CQL) 覆寫預設 TTL 設定，並將自訂 TTL 值套用至資料列和資料行。如需詳細資訊，請參閱the section called “如何使用存留時間”。

TTL 定價是以使用存留時間來刪除或更新的資料列大小為基礎。TTL 作業的計量單位為TTL `deletes`。每個刪除或更新的資料列，每 KB 的資料會耗用一次 TTL 刪除。例如，若要更新儲存 2.5 KB 資料的資料列，並同時刪除資料列中的一或多個資料行，則需要刪除三次 TTL。或者，若要刪除包含 3.5 KB 資料的整個資料列，則需要四次 TTL 刪除。每個資料列的每 KB 刪除資料會耗用一次 TTL 刪除。如需定價的詳細資訊，請參閱 Amazon Keyspaces (適用於 Apache Cassandra) 定價格。

主題

運作方式：亞馬遜 Keyspaces 間存留時間 (TTL)

亞馬遜 Keyspaces 存留時間 (TTL) 是全受管的。您不需要管理壓縮策略等低階系統設定。資料會在您指定的時間到期，而 Amazon Keyspaces 會自動移除過期的資料 (通常在 10 天內)，而不會影響應用程式的效能或可用性。

過期的資料會標示為刪除，不適用於資料處理語言 (DML) 陳述式。當您繼續對包含過期資料的資料列執行讀取和寫入時，過期的資料會繼續計入讀取容量單位 (RCU) 和寫入容量單位 (WCU)，直到從儲存區中刪除為止。

主題

設定資料表的預設 TTL 值

在 Amazon Keyspaces 中，您可以在建立表格時為表格中的所有列設定預設 TTL 值。您也可以編輯現有表格，為插入表格中的新列設定或變更預設 TTL 值。變更資料表的預設 TTL 值並不會修改資料表中任何現有資料的 TTL 值。資料表的預設 TTL 值為零，這表示資料不會自動過期。如果資料表的預設 TTL 值大於零，則會在每一列新增到期時間戳記。

Amazon Keyspaces 間會在每次資料更新時計算新的 TTL 時間戳記。TTL 值以秒為單位設定，最長可設定的值為 630,720,000 秒，相當於 20 年。如需如何使用AWS Management Console或 CQL 設定、修改及停用資料表之預設 TTL 值的詳細資訊，請參閱the section called “如何使用存留時間”。

設定列與欄的自訂 TTL 值

> **Note**
>
> 在為列和欄設定自訂 TTL 值之前，必須先在表格上啟用 TTL。如需詳細資訊，請參閱the section called “如何啟用使用自訂屬性 (TTL) 啟用使用自訂屬性 (TTL)。”。

若要覆寫資料表的預設 TTL 值或設定個別資料列的到期日期，您可以使用下列 CQL 資料操作語言 (DML) 陳述式：

- INSERT— 用於插入具有 TTL 值集的新資料列。
- UPDATE— 用於修改具有新 TTL 值的現有資料列。

設定列的 TTL 值優先於表格的預設 TTL 設定。

如需 CQL 語法和範例，請參閱the section called “使用 CQL INSERT 來編輯自訂的存留時間 (TTL) 設定 (TTL) 設定”。

若要覆寫或設定個別資料欄的 TTL 值，您可以使用下列 CQL DML 陳述式，更新現有資料列內資料行子集的 TTL 設定：

- UPDATE-用於更新一列數據。

為欄設定 TTL 值的優先順序高於表格的預設 TTL 設定和資料列的任何自訂 TTL 設定。如需 CQL 語法和範例，請參閱the section called “使用 CQL UPDATE 來編輯自訂的存留時間 (TTL) 設定 (TTL) 設定”。

在表格上啟用 TTL

當您在CREATE TABLE或ALTER TABLE陳述式中指定大於 0 的default_time_to_live值時，表格會自動啟用 TTL。如果您沒有default_time_to_live為表格指定，但想要使INSERT用或UPDATE操作來指定列或欄的自訂 TTL 值，則必須先為表格啟用 TTL。您可以使用ttl自訂內容為表格啟用 TTL。

當您在表格上啟用 TTL 時，Amazon Keyspaces 會開始為每一列儲存額外的 TTL 相關中繼資料。此外，TTL 會使用到期時間戳記來追蹤資料列或欄到期的時間。時間戳記會儲存為資料列中繼資料，並會增加資料列的儲存成本。

啟用 TTL 功能之後，您就無法停用資料表的 TTL 功能。將資料表設定default_time_to_live為 0 會停用新資料的預設到期時間，但不會停用 TTL 功能或將表格還原回原始 Amazon Keyspace 儲存中繼資料或寫入行為。

亞馬遜 Keyspaces 間的生存時間和與AWS服務集成

亞馬遜提供以下 TTL 指標 CloudWatch 來啟用持續監控。

- TTLDeletes— 使用存留時間 (TTL) 刪除或更新連續資料所耗用的單位。

如需如何監控 CloudWatch 指標的詳細資訊，請參閱the section called “使用監控 CloudWatch”。

使用時AWS CloudFormation，您可以在創建亞馬遜 Keyspaces 表時打開 TTL。如需詳細資訊，請參閱《AWS CloudFormation 使用者指南》。

如何使用存留時間 (TTL)

您可以使用 Amazon Keyspaces (適用於 Apache Cassandra) 主控台或 CQL 啟用、更新和禁用存留時間 (適用於 Apache Cassandra) 主控台或 CQL 啟用、更新和禁用

主題

使用預設的存留時間 (TTL) 設定 (TTL) 設定 (TTL) 設定 (TTL) 設定 (TTL) 設定

依照下列步驟來建立具有 Amazon Keyspaces 主控台啟用啟用存留時間 (Reyspaces) 設定 (Keyspaces) 設定。

1. 登入AWS Management Console，並在 https://console.aws.amazon.com/keyspaces/home 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇 Tables (資料表)，然後選擇 Create table (建立資料表)。
3. 在「表格詳細資訊」段落的「建立表格」頁面中，選取索引鍵空間並輸入新表格的名稱。
4. 在 [結構描述] 區段中，建立資料表的結構定義。
5. 在 [表格設定] 區段中，選擇 [自訂設定]。
6. 繼續前往存留時間 (TTL)。

在此步驟中，您可以選取表格的預設 TTL 設定。

對於「預設 TTL」期間，請輸入到期時間並選擇您輸入的時間單位，例如秒、天或年。Amazon Keyspaces 將以秒為單位儲存留時間 (Amazon Keyspaces)

7. 選擇 建立資料表 。您的資料表會以指定的預設 TTL 值建立。

> **Note**
>
> 您可以使用 CQL 編輯器中的資料操作語言 (DML)，覆寫特定列或欄的資料表預設 TTL 設定。

更新現有資料表 (TTL) 設定 (TTL) 設定 (TTL) 設定 (TTL) 設定

請依照下列步驟使用 Amazon Keyspaces 間主控台更新現有資料表的即時時間設定。

1. 登入AWS Management Console，並在 https://console.aws.amazon.com/keyspaces/home 開啟 Amazon Keyspaces 主控台。
2. 選擇您要更新的資料表，然後選擇其他設定資料表，然後選擇其他設定資料表。
3. 繼續至存留時間 (TTL)，然後選擇 「編輯」。
4. 對於「預設 TTL」期間，請輸入到期時間並選擇您輸入的時間單位，例如秒、天或年。Amazon Keyspaces 將以秒為單位儲存留時間 (Amazon Keyspaces) 這不會變更現有資料列的 TTL 值。
5. 定義 TTL 設定後，請選擇 [儲存變更]。

停用現有資料表 (TTL) 設定 (TTL) 設定 (TTL) 設定 (TTL) 設定

請按照以下步驟使用 Amazon Keyspaces 間停用現有表格的存留時間AWS Management Console設定。

1. 登入AWS Management Console，並在 https://console.aws.amazon.com/keyspaces/home 開啟 Amazon Keyspaces 主控台。
2. 選擇您要更新的資料表，然後選擇其他設定資料表，然後選擇其他設定資料表。
3. 繼續至存留時間 (TTL)，然後選擇 「編輯」。
4. 選取預設 TTL 週期，並將值設定為零。依預設，這會針對 future 資料停用資料表的 TTL。它不會變更現有資料列的 TTL 值。
5. 定義 TTL 設定後，請選擇 [儲存變更]。

使用 CQL 啟用預設的存留時間 (TTL) 設定 (TTL) 設定 (TTL) 設定。

當您建立新資料表時，將預設 TTL 值設為 3,024,000 秒 (代表 35 天) 時，請啟用 TTL。

```
CREATE TABLE my_table (
                userid uuid,
                time timeuuid,
                subject text,
                body text,
                user inet,
                PRIMARY KEY (userid, time)
                ) WITH default_time_to_live = 3024000;
```

若要確認新資料表的 TTL 設定，請使用下列範例所示的`cqlsh`的`describe`陳述式。輸出會將表格的預設 TTL 設定顯示為`default_time_to_live`。

```
describe my_table;
```

使用 CQL **ALTER TABLE** 來編輯預設的存留時間 (TTL) 設定 (TTL) 設定

將現有資料表的 TTL 設定更新為 2,592,000 秒，代表 30 天。

```
ALTER TABLE my_table WITH default_time_to_live = 2592000;
```

若要確認已更新之表格的 TTL 設定，請使用下列範例所示的`cqlsh`的`describe`陳述式。輸出會將表格的預設 TTL 設定顯示為`default_time_to_live`。

```
describe my_table;
```

如何啟用使用自訂屬性 (TTL) 啟用使用自訂屬性 (TTL)

若要啟用可套用至列和欄的自訂設定，而不為整個表格啟用 TTL 預設設定，您可以使用下列 CQL 陳述式。

```
CREATE TABLE my_keyspace.my_table (id int primary key) WITH CUSTOM_PROPERTIES={'ttl':
{'status': 'enabled'}};
```

啟`ttl`用後，您就無法為資料表加以停用。

如何啟用使用自訂屬性 (TTL) 啟用使用自訂屬性 (TTL)。

若要啟用可套用至列和欄的自訂設定，而不為整個表格啟用 TTL 預設設定，您可以使用下列 CQL 陳述式。

```
ALTER TABLE my_table WITH CUSTOM_PROPERTIES={'ttl':{'status': 'enabled'}};
```

啟ttl用後，您就無法為資料表加以停用。

使用 CQL **INSERT** 來編輯自訂的存留時間 (TTL) 設定 (TTL) 設定

下列 CQL 陳述式會將一列資料插入資料表，並將預設 TTL 設定變更為 259,200 秒 (相當於 3 天)。

```
INSERT INTO my_table (userid, time, subject, body, user)
        VALUES (B79CB3BA-745E-5D9A-8903-4A02327A7E09, 96a29100-5e25-11ec-90d7-
b5d91eceda0a, 'Message', 'Hello','205.212.123.123')
        USING TTL 259200;
```

若要確認插入資料列的 TTL 設定，請使用下列陳述式。

```
SELECT TTL (subject) from my_table;
```

使用 CQL **UPDATE** 來編輯自訂的存留時間 (TTL) 設定 (TTL) 設定

若要將先前插入之「主旨」欄的 TTL 設定從 259,200 秒 (3 天) 變更為 86,400 秒 (一天)，請使用下列陳述式。

```
UPDATE my_table USING TTL 86400 set subject = 'Updated Message' WHERE userid =
 B79CB3BA-745E-5D9A-8903-4A02327A7E09 and time = 96a29100-5e25-11ec-90d7-b5d91eceda0a;
```

您可以運行一個簡單的選擇查詢，以查看到期時間之前更新的記錄。

```
SELECT * from my_table;
```

該查詢顯示以下輸出。

```
userid                                | time                                 | body   |
 subject         | user
```

```
--------------------------------------+--------------------------------------+-------
+-----------------+-----------------
b79cb3ba-745e-5d9a-8903-4a02327a7e09  | 96a29100-5e25-11ec-90d7-b5d91eceda0a | Hello |
 Updated Message | 205.212.123.123
50554d6e-29bb-11e5-b345-feff819cdc9f  | cf03fb21-59b5-11ec-b371-dff626ab9620 | Hello |
        Message | 205.212.123.123
```

若要確認到期成功，請在設定的到期時間之後再次執行相同的查詢。

```
SELECT * from my_table;
```

該查詢顯示「主題」列已過期後，以下輸出。

```
userid                               | time                                 | body  |
 subject | user
--------------------------------------+--------------------------------------+-------
+---------+-----------------
b79cb3ba-745e-5d9a-8903-4a02327a7e09  | 96a29100-5e25-11ec-90d7-b5d91eceda0a | Hello |
   null | 205.212.123.123
50554d6e-29bb-11e5-b345-feff819cdc9f  | cf03fb21-59b5-11ec-b371-dff626ab9620 | Hello |
 Message | 205.212.123.123
```

使用亞馬遜 Keyspaces 間中的客戶端時間戳

在 Amazon Keyspaces 間中，用戶端時間戳記是與 Cassandraa 相容的時間戳記，會為表格中的每個儲存格保留。您可以讓用戶端應用程式決定寫入順序，使用用戶端時間戳記來解決衝突。例如，當全域分散式應用程式的用戶端對相同資料進行更新時，用戶端時間戳記會保留在用戶端上進行更新的順序。Amazon Keyspaces 間使用這些時間戳記來處理寫入。如需詳細資訊，請參閱the section called “運作方式”。

開啟資料表的用戶端時間戳記之後，您可以在資料操作語言 (DML) CQL 查詢中使用USING TIMESTAMP子句指定時間戳記。如果您未在 CQL 查詢中指定時間戳記，Amazon Keyspaces space 會使用用戶端驅動程式傳遞的時間戳記。如果用戶端驅動程式未提供時間戳記，Amazon Keyspace 會自動指派儲存格層級的時間戳記。若要查詢時間戳記，您可以在 DML 陳述式中使用WRITETIME函數。如需詳細資訊，請參閱the section called “如何使用用用用用用用用用”。

Amazon Keyspaces 間不會收取額外費用來開啟用戶端時間戳記。但是，使用客戶端時間戳記，您可以為行中的每個值存儲和寫入其他數據。這可能會導致額外的儲存使用量，並在某些情況下增加輸送量使用量。若要深入瞭解如何估算對資料列大小的影響，請參閱the section called “運作方式”。有關 Amazon Keyspaces (適用於 Apache) 中建立的 Keyspaces (適用於 Apache) 中建立的定價。

主題

- 亞馬遜 Keyspaces 間中的客戶端時間戳如何工作
- 使用用用用用用用用用用用用用用

亞馬遜 Keyspaces 間中的客戶端時間戳如何工作

Amazon Keyspaces 間用戶端時間戳記是全受管的。您不必管理低級別的系統設置，例如清理和壓縮策略。

當您刪除資料時，資料列會以標記標示為要刪除。Amazon Keyspaces 會自動移除標記資料 (通常在 10 天內)，而不會影響應用程式的效能或可用性。標記資料不會對資料處理語言 (DML) 陳述式的平行。當您繼續對包含標記資料的資料列執行讀取和寫入時，標記資料會繼續計入儲存區、讀取容量單位 (RCU) 和寫入容量單位 (WCU)，直到從儲存區中刪除為止。

主題

- 亞馬遜 Keyspaces 間中的客戶端時間戳如何工作
- Amazon Keyspaces 間用戶端時間戳記，並與AWS服務整合

亞馬遜 Keyspaces 間中的客戶端時間戳如何工作

在 Amazon Keyspaces 間中開啟用戶端時間戳記時，每一列的每一欄都會儲存一個時間戳記。這些時間戳記大約會佔用 20-40 個位元組 (視您的資料而定)，而且會造成資料列的儲存和輸送量成本。這些中繼資料位元組也會計入您的 1 MB 資料列大小配額。若要判斷儲存空間的整體增加 (以確保資料列大小維持在 1 MB 以下)，請考量表格中的欄數，以及每個資料列中的收集要素數目。例如，如果資料表有 20 個資料行，而每個資料行都儲存 40 個位元組的資料，則資料列的大小會從 800 個位元組增加到 1200 個位元組。如需如何估計資料列大小的詳細資訊，請參閱the section called “計算列大小”。除了用於儲存的額外 400 位元組之外，在此範例中，每次寫入使用的寫入容量單位 (WCU) 數目從 1 個 WCU 增加到 2 個 WCU。如需如何計算讀取和寫入容量的詳細資訊，請參閱the section called “讀/寫容量模式”。

開啟表格的用戶端時間戳記之後，您就無法將其關閉。此外，不能使用時間戳記NULL，因此如果 CQL 陳述式或用戶端驅動程式未提供任何用戶端時間戳記，則會自動新增 Amazon Keyspace 產生的時間戳記。

Amazon Keyspaces 間用戶端時間戳記，並與AWS服務整合

Amazon 提供下列用戶端時間戳記指標， CloudWatch 以啟用持續監控。

- `SystemReconciliationDeletes`— 移除標記資料所需的刪除作業數目。

如需如何監控 CloudWatch 指標的詳細資訊，請參閱the section called “使用監控 CloudWatch”。

使用用用用用用用用用用用用用用用

您可以使用亞馬遜 Keyspaces 間（對於 Apache 卡桑德拉）控制台，卡桑德拉查詢語言（CQL），AWSSDK 和AWS Command Line Interface（AWS CLI）打開客戶端時間戳。本節提供如何開啟新資料表和現有資料表的用戶端時間戳記，以及如何在查詢中使用用戶端時間戳記的範例。如需 API 的詳細資訊，請參閱 Amazon Keyspaces API 參考。

> **Important**
>
> 無法關閉用戶端時間戳記。開啟用戶端時間戳記是一次性的變更。Amazon Keyspaces 不提供在不刪除表格的情況下將其關閉的選項。

主題

建立已開啟用戶端時間戳記的新資料表（主控台）

依照下列步驟使用用用

使用用戶端時間戳記 (主控台) 建立新資料表

1. 登入AWS Management Console，開啟位於 https://console.aws.amazon.com/keyspaces/home 的 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇 Tables (資料表)，然後選擇 Create table (建立資料表)。
3. 在「表格詳細資訊」段落的「建立表格」頁面中，選取索引鍵空間並輸入新表格的名稱。
4. 在「結構描述」區段中，為您的資料表建立結構定義。
5. 在 [表格設定] 區段中，選擇 [自訂設定]。
6. 繼續前往用戶端時間戳記。

   選擇 [開啟用戶端時間戳記] 以開啟表格的用戶端時間戳記。
7. 選擇 建立資料表 。您的資料表是在開啟用戶端時間戳記的情況下建立。

在現有資料表 (主控台) 上開啟用戶端時間戳記

請依照下列步驟，使用 Amazon Keyspaces 間為現有資料表開啟用戶端時間戳記AWS Management Console。

開啟現有資料表 (主控台) 的用戶端時間戳記

1. 登入AWS Management Console，開啟位於 https://console.aws.amazon.com/keyspaces/home 的 Amazon Keyspaces 主控台。
2. 選擇您要更新的資料表，然後選擇 [其他設定] 標籤。

3. 在 [其他設定] 索引標籤上，移至 [修改用戶端時間戳記] 並選取 [開啟用戶端時間戳記
4. 選擇 [儲存變更] 以變更表格的設定。

建立開啟用戶端時間戳記的新資料表 (CQL)

若要在建立新資料表時開啟用戶端時間戳記，您可以使用下列 CQL 陳述式。

```
CREATE TABLE my_table (
   userid uuid,
   time timeuuid,
   subject text,
   body text,
   user inet,
   PRIMARY KEY (userid, time)
) WITH CUSTOM_PROPERTIES = {'client_side_timestamps': {'status': 'enabled'}};
```

若要確認新資料表的用戶端時間戳記設定，請使用SELECT陳述式來檢閱，如下列範例所示。custom_properties

```
SELECT custom_properties from system_schema_mcs.tables where keyspace_name =
 'my_keyspace' and table_name = 'my_table';
```

這個陳述式的輸出會顯示用戶端時間戳記的狀態。

```
'client_side_timestamps': {'status': 'enabled'}
```

使用**ALTER TABLE** (CQL) 開啟現有資料表的用戶端時間戳記

若要開啟現有資料表的用戶端時間戳記，您可以使用下列 CQL 陳述式。

```
ALTER TABLE my_table WITH custom_properties = {'client_side_timestamps': {'status':
 'enabled'}};;
```

若要確認新資料表的用戶端時間戳記設定，請使用SELECT陳述式來檢閱，如下列範例所示。custom_properties

```
SELECT custom_properties from system_schema_mcs.tables where keyspace_name =
 'my_keyspace' and table_name = 'my_table';
```

這個陳述式的輸出會顯示用戶端時間戳記的狀態。

```
'client_side_timestamps': {'status': 'enabled'}
```

建立已開啟用戶端時間戳記的新資料表 (CLI)

若要在建立新的資料表時開啟用用用用用用用用用用用用用用用用用用用用用用用用用用用用

```
./aws keyspaces create-table \
--keyspace-name my_keyspace \
--table-name my_table \
--client-side-timestamps 'status=ENABLED' \
--schema-definition 'allColumns=[{name=id,type=int},{name=date,type=timestamp},
{name=name,type=text}],partitionKeys=[{name=id}]'
```

若要確認已開啟新資料表的用戶端時間戳記，請執行下列程式碼。

```
./aws keyspaces get-table \
--keyspace-name my_keyspace \
--table-name my_table
```

輸出看起來會與此類似。

```
{
    "keyspaceName": "my_keyspace",
    "tableName": "my_table",
    "resourceArn": "arn:aws:cassandra:us-east-2:555555555555:/keyspace/my_keyspace/
table/my_table",
    "creationTimestamp": 1662681206.032,
    "status": "ACTIVE",
    "schemaDefinition": {
        "allColumns": [
            {
                "name": "id",
                "type": "int"
            },
            {
                "name": "date",
                "type": "timestamp"
            },
            {
                "name": "name",
```

```
                "type": "text"
            }
        ],
        "partitionKeys": [
            {
                "name": "id"
            }
        ],
        "clusteringKeys": [],
        "staticColumns": []
    },
    "capacitySpecification": {
        "throughputMode": "PAY_PER_REQUEST",
        "lastUpdateToPayPerRequestTimestamp": 1662681206.032
    },
    "encryptionSpecification": {
        "type": "AWS_OWNED_KMS_KEY"
    },
    "pointInTimeRecovery": {
        "status": "DISABLED"
    },
    "clientSideTimestamps": {
        "status": "ENABLED"
    },
    "ttl": {
        "status": "ENABLED"
    },
    "defaultTimeToLive": 0,
    "comment": {
        "message": ""
    }
}
```

在現有資料表 (CLI) 上開啟用戶端時間戳記

若要使用 CLI 開啟現有資料表的用戶端時間戳記，您可以使用下列程式碼。

```
./aws keyspaces update-table \
--keyspace-name my_keyspace \
--table-name my_table \
--client-side-timestamps 'status=ENABLED'
```

若要確認已開啟資料表的用戶端時間戳記，請執行下列程式碼。

```
./aws keyspaces get-table \
--keyspace-name my_keyspace \
--table-name my_table
```

輸出看起來會與此類似。

```
{
    "keyspaceName": "my_keyspace",
    "tableName": "my_table",
    "resourceArn": "arn:aws:cassandra:us-east-2:555555555555:/keyspace/my_keyspace/
table/my_table",
    "creationTimestamp": 1662681312.906,
    "status": "ACTIVE",
    "schemaDefinition": {
        "allColumns": [
            {
                "name": "id",
                "type": "int"
            },
            {
                "name": "date",
                "type": "timestamp"
            },
            {
                "name": "name",
                "type": "text"
            }
        ],
        "partitionKeys": [
            {
                "name": "id"
            }
        ],
        "clusteringKeys": [],
        "staticColumns": []
    },
    "capacitySpecification": {
        "throughputMode": "PAY_PER_REQUEST",
        "lastUpdateToPayPerRequestTimestamp": 1662681312.906
    },
    "encryptionSpecification": {
        "type": "AWS_OWNED_KMS_KEY"
    },
```

```
    "pointInTimeRecovery": {
        "status": "DISABLED"
    },
    "clientSideTimestamps": {
        "status": "ENABLED"
    },
    "ttl": {
        "status": "ENABLED"
    },
    "defaultTimeToLive": 0,
    "comment": {
        "message": ""
    }
}
```

在資料操作語言 (DML) 陳述式中使用用戶端時間戳記

開啟用戶端時間戳記之後，您可以在INSERT、UPDATE和陳述式中傳遞時間戳記與USING TIMESTAMP子DELETE句。時間戳記值bigint代表自標準基準時間稱為epoch：1970 年 1 月 1 日在格林威治標準時間 00:00:00 以來的微秒數。客戶提供的時間戳記必須介於過去 2 天和 future 從當前掛鐘時間開始的 5 分鐘之間。Amazon Keyspaces 間會在資料的整個過程中保留時間戳記中繼資料。您可以使用該WRITETIME函數查找過去發生的年份的時間戳記。如需 CQL 語法的詳細資訊，請參閱the section called “DML 陳述式”。

下列 CQL 陳述式是如何使用時間戳記做為update_parameter.

```
INSERT INTO catalog.book_awards (year, award, rank, category, book_title, author,
 publisher)
   VALUES (2022, 'Wolf', 4, 'Non-Fiction', 'Science Update', 'Ana Carolina Silva',
 'SomePublisher')
   USING TIMESTAMP 1669069624;
```

如果您未在 CQL 查詢中指定時間戳記，Amazon Keyspaces space 會使用用戶端驅動程式傳遞的時間戳記。如果用戶端驅動程式未提供時間戳記，Amazon Keyspaces 會為您的寫入作業指派伺服器端時間戳記。

若要查看針對特定資料行儲存的時間戳記值，您可以在SELECT陳述式中使用WRITETIME函式，如下列範例所示。

```
SELECT year, award, rank, category, book_title, author, publisher, WRITETIME(year),
 WRITETIME(award), WRITETIME(rank),
```

```
 WRITETIME(category), WRITETIME(book_title), WRITETIME(author), WRITETIME(publisher)
from catalog.book_awards;
```

創建亞馬遜 Keyspaces 資源AWS CloudFormation

Amazon Keyspaces (適用於 Apache Cassandra) 已整合AWS CloudFormation，這項服務可協助您建立AWS資源的模型和設定，以減少建立和管理資源和基礎設施的時間。您建立一個描述所需之AWS資源的範本 (如 Keyspaces 和資料表)，而AWS CloudFormation負責為您佈建與設定這些資源。

當您使用時AWS CloudFormation，您可以重複使用您的範本，重複、一致的設定您的 Amazon Keyspaces 資源。只需描述一次您的資源，即可在多AWS 帳戶個和區域內重複佈建相同資源。

亞馬遜 Keyspaces 和AWS CloudFormation模板

若要佈建和設定 Amazon Keyspaces 的資源，則必須了解AWS CloudFormation範本。範本是以 JSON 或 YAML 格式化的文本檔案。而您亦可以透過這些範本的說明，了解欲在 AWS CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML，您可以使用 AWS CloudFormation Designer 協助您開始使用 AWS CloudFormation 範本。如需詳細資訊，請參閱什麼AWS CloudFormation是？。在《AWS CloudFormation使用者指南》中。

亞馬遜 Keyspaces 支持在中創建密鑰空間和表AWS CloudFormation。對於使用AWS CloudFormation樣板建立的表格，您可以指定綱要、讀取/寫入模式和佈建的輸送量設定值。如需詳細資訊，包括索引鍵空間和資料表的 JSON 和 YAML 範本範例，請參閱AWS CloudFormation使用者指南中的Cassandra 資源類型參考。

進一步了解 AWS CloudFormation

若要進一步了解 AWS CloudFormation，請參閱下列資源：

- AWS CloudFormation
- 《AWS CloudFormation 使用者指南》https://docs.aws.amazon.com/AWSCloudFormation/latest/UserGuide/Welcome.html
- AWS CloudFormation指令行介面使用者指南

監控亞馬遜 Keyspaces（阿帕奇卡桑德拉）

監控是維護 Amazon Keyspaces 和其他AWS解決方案的可靠性、可用性和效能的重要組成部分。AWS提供下列監控工具來觀看 Amazon Keyspaces、在發生錯誤時報告，並在適當時採取自動動作：

- Amazon Keyspaces 提供預先設定的儀表板，其中AWS Management Console顯示帳戶中所有表格中彙總的延遲和錯誤。
- Amazon 會即時 CloudWatch監控您的AWS資源和執行AWS的應用程式。您可以使用自訂儀表板收集和追蹤指標。例如，您可以透過測量不同時間和不同負載條件下的效能，為環境中的一般 Amazon Keyspace 效能建立基準。監控 Amazon Keyspaces 時，存放歷史監控資料，以便您可以將其與目前的效能資料進行比較、識別正常的效能模式和效能異常，並設計解決問題的方法。若要建立基準線，您至少應監視系統錯誤。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。
- Amazon CloudWatch 警示會監控您指定期間內的單一指標，並根據指定臨界值在多個時段內相對於指定閾值的指標值執行一或多個動作。例如，如果您在具有應用程式 auto 動擴展的佈建模式下使用 Amazon Keyspaces，則動作為 Amazon 簡單通知服務 (Amazon SNS) 傳送的通知，以評估應用程式自動擴展政策。

  CloudWatch 警報不會僅僅因為它們處於特定狀態而叫用動作。狀態必須已變更，且在指定的期間數內維持此狀態。如需詳細資訊，請參閱用 Amazon 監控 Amazon Keyspaces CloudWatch。
- Amazon CloudWatch 日誌可讓您從 Amazon Keyspaces 表和其他來源監控 CloudTrail、存放和存取日誌檔。 CloudWatch 記錄檔可以監控記錄檔中的資訊，並在符合特定臨界值時通知您。您也可以將日誌資料存檔在高耐用性的儲存空間。如需詳細資訊，請參閱 Amazon CloudWatch 日誌使用者指南。
- AWS CloudTrail 擷取您 AWS 帳戶 發出或代表發出的 API 呼叫和相關事件，並傳送記錄檔案至您指定的 Amazon S3 儲存貯體。您可以找出哪些使用者和帳戶呼叫 AWS、發出呼叫的來源 IP 地址，以及呼叫的發生時間。如需詳細資訊，請參閱 AWS CloudTrail 使用者指南。

Amazon EventBridge 是一種無伺服器事件匯流排服務，可讓您輕鬆地將應用程式與各種來源的資料連接起來。 EventBridge 從您自己的應用程式、S oftware-as-a 服務 (SaaS) 應用程式以及服務提供即時資料串流，並AWS將該資料路由到目標 (例如 Lambda)。這可讓您監控在服務中發生的事件，並建置事件導向的架構。如需詳細資訊，請參閱 Amazon EventBridge 使用者指南。

主題

- 用 Amazon 監控 Amazon Keyspaces CloudWatch
- 記錄 Amazon Keyspaces API 調用 AWS CloudTrail

用 Amazon 監控 Amazon Keyspaces CloudWatch

您可以使用 Amazon 監控 Amazon Keyspaces CloudWatch，Amazon 會收集原始資料並將其處理為可讀且接近即時的指標。這些統計資料會保留 15 個月，以便您存取歷史資訊，並更清楚 Web 應用程式或服務的執行效能。

您也可以設定留意特定閾值的警示，當滿足這些閾值時傳送通知或採取動作。如需詳細資訊，請參閱 Amazon CloudWatch 使用者指南。

> **Note**
>
> 若要使用顯示 Amazon Keyspaces 常用指標的預先設定 CloudWatch 儀表板快速開始使用，您可以使用提供的 AWS CloudFormation 範本。https://github.com/aws-samples/amazon-keyspaces-cloudwatch-cloudformation-templates

主題

如何使用 Amazon Keyspaces 指標？

Amazon Keyspaces 報告的指標提供您可以使用不同方式進行分析的資訊。下列清單顯示一些常見的指標用途。這些是協助您開始的建議，而不是完整清單。如需有關量度和保留的詳細資訊，請參閱量度。

如何？	相關指標
如何確定是否發生任何系統錯誤？	您可以監視`SystemErrors` 以判斷是否有任何要求導致伺服器錯誤碼。這項指標通常應該等於零。如果不是，您可能要進行調查。
如何比較平均佈建讀取與使用的讀取容量？	監視平均佈建的讀取容量和使用的讀取容量

如何？	相關指標
	1. 將「期間」ConsumedReadCapacityUnits 和設ProvisionedReadCapacityUnits 定為您要監視的間隔。 2. 將 [統計資料] ConsumedReadCapacityUnits 從Average變更為。Sum 3. 建立新的空白數學運算式。 4. 在新數學運算式的「詳細資訊」區段中，輸入度量的 ID，ConsumedReadCapacityUnits 並將度量除以度量的「 CloudWatch 週期」函數 (metric_id/ (句點 (metric_id))。 5. 取消選取。ConsumedReadCapacityUnits 您現在可以將平均使用的讀取容量與佈建的容量進行比較。有關基本算術函數以及如何建立時間序列的詳細資訊，請參閱使用指標數學運算。
如何將平均佈建寫入與取用的寫入容量進行比較？	監視平均佈建的寫入容量和耗用的寫入容量 1. 將「期間」ConsumedWriteCapacityUnits 和設ProvisionedWriteCapacityUnits 定為您要監視的間隔。 2. 將 [統計資料] ConsumedWriteCapacityUnits 從Average變更為。Sum 3. 建立新的空白數學運算式。 4. 在新數學運算式的「詳細資訊」區段中，輸入度量的 ID，ConsumedWriteCapacityUnits 並將度量除以度量的「 CloudWatch 週期」函數 (metric_id/ (句點 (metric_id))。 5. 取消選取。ConsumedWriteCapacityUnits 您現在可以將平均使用的寫入容量與佈建的容量進行比較。有關基本算術函數以及如何建立時間序列的詳細資訊，請參閱使用指標數學運算。

Amazon Keyspaces 指標和維度

當您與 Amazon Keyspaces 進行交互時，它會將以下指標和維度發送到 Amazon CloudWatch。所有指標都會彙總並每分鐘報告一次。您可以使用下列程序來檢視 Amazon Keyspaces 的指標。

使用 CloudWatch 主控台檢視指標

指標會先依服務命名空間分組，再依各命名空間內不同的維度組合分類。

1. 請在以下位置開啟 CloudWatch 主控台。 https://console.aws.amazon.com/cloudwatch/
2. 如有必要請變更區域。在導覽列上，選擇資 AWS 源所在的地區。如需詳細資訊，請參閱 AWS 服務端點。
3. 在導覽窗格中，選擇 指標。
4. 在「所有量度」標籤下，選擇 AWS/Cassandra.

若要使用 AWS CLI 檢視測量結果

- 在命令提示中，使用下列命令。

```
aws cloudwatch list-metrics --namespace "AWS/Cassandra"
```

Amazon Keyspaces 指標和維度

Amazon Keyspaces 發送到 Amazon 的指標和維度在這 CloudWatch 裡列出。

Amazon Keyspaces 指標

Amazon 每隔一分 CloudWatch 鐘彙總 Amazon Keyspaces 間指標。

並非所有統計數字，例如 Average 或 Sum，皆適用於所有指標。但是，所有這些值都可以透過 Amazon Keyspace 主控台取得，或透過使用 CloudWatch 主控台或所有指標的 AWS SDK 來取得。AWS CLI在下表中，每個指標皆有適用於該指標的有效統計數字列表。

指標	描述
AccountMaxTableLevelReads	帳戶表格可使用的最大讀取容量單位數。對於隨選資料表，此限制會限制表格可使用的最大讀取請求單位。

指標	描述
	單位：Count 有效的統計數字： • Maximum— 帳戶表格可使用的最大讀取容量單位數。
AccountMaxTableLevelWrites	帳戶表格可使用的寫入容量單位數目上限。對於隨選資料表，此限制會限制表格可使用的寫入請求單位上限。 單位：Count 有效的統計數字： • Maximum— 帳戶表格可使用的最大寫入容量單位數。
AccountProvisionedReadCapacityUtilization	帳戶可以使用的佈建讀取容量單元百分比。 單位：Percent 有效的統計數字： • Maximum：帳戶使用的佈建讀取容量單元百分比上限。 • Minimum：帳戶使用的佈建讀取容量單元百分比下限。 • Average：帳戶使用的佈建讀取容量單元平均百分比。指標會每隔五分鐘發佈一次。因此，如果您快速調整佈建的讀取容量單位，此統計數字可能無法反映真實的平均值。

指標	描述
AccountProvisionedWriteCapacityUtilization	帳戶使用的佈建寫入容量單元百分比。 單位：Percent 有效的統計數字： • Maximum：帳戶使用的佈建寫入容量單元百分比上限。 • Minimum：帳戶使用的佈建寫入容量單元百分比下限。 • Average：帳戶使用的佈建寫入容量單元平均百分比。指標會每隔五分鐘發佈一次。因此，如果您快速調整佈建的寫入容量單位，此統計數字可能無法反映真實的平均值。
BillableTableSizeInBytes	資料表的可計費大小 (以位元組為單位)。它是表中所有行的編碼大小的總和。此指標可協助您追蹤一段時間內的表格儲存成本。 單位：Bytes 維度：Keyspace, TableName 有效的統計數字： • Maximum— 資料表的最大儲存大小。 • Minimum— 表格的最小儲存大小。 • Average— 表格的平均儲存大小。此指標的計算間隔為 4-6 小時。

指標	描述
ConditionalCheckFailedRequests	失敗的輕量型交易 (LWT) 寫入要求的數目。INSERT、UPDATE 以及 DELETE 操作可讓您提供邏輯條件，該條件必須評估為 true，才能繼續操作。如果此條件評估為 false，ConditionalCheckFailedRequests 則會以 1 遞增。根據資料列大小，評估為 false 的條件檢查會使用寫入容量單位。如需詳細資訊，請參閱 the section called “輕量型交易”。 單位：Count 維度：Keyspace, TableName 有效的統計數字： • Minimum • Maximum • Average • SampleCount • Sum

指標	描述
ConsumedReadCapacityUnits	指定期間內使用的讀取容量單位數目。如需詳細資訊，請參閱讀/寫入容量模式。 **Note** 若要瞭解每秒的平均輸送量使用率，請使用Sum統計資料來計算一分鐘期間的耗用輸送量。然後將總和除以一分鐘 (60) 的秒數，以計算ConsumedReadCapacityUnits 每秒的平均值 (認識到該平均值不會突出顯示該分鐘內發生的讀取活動中任何大而短暫的峰值)。如需有關比較平均消耗讀取容量與佈建讀取容量的詳細資訊，請參閱 the section called “使用指標” 單位：Count 維度：Keyspace, TableName 有效的統計數字： • Minimum— 任何個別要求對表格所使用的最小讀取容量單位數目。 • Maximum— 任何個別要求對表格所使用的讀取容量單位數目上限。 • Average ：每個請求消耗的平均讀取容量。量。 **Note** 此 Average 數值受樣本數值會為零的閒置時段所影響。 • Sum：所耗用的讀取容量單位總數。這是 ConsumedReadCapacityUnits 指標最實用的統計數字。 • SampleCount — 即使沒有消耗讀取容量，對 Amazon Keyspaces 的請求數量也是如此。

指標	描述
	Note 此 SampleCount 數值受樣本數值會為零的閒置時段所影響。

指標	描述
ConsumedWriteCapacityUnits	指定期間內使用的寫入容量單位數目。您可以擷取資料表的總耗用寫入容量。如需詳細資訊，請參閱讀/寫入容量模式。 **Note** 若要瞭解每秒的平均輸送量使用率，請使用Sum統計資料來計算一分鐘期間的耗用輸送量。然後將總和除以一分鐘 (60) 中的秒數，以計算ConsumedWriteCapacityUnits 每秒的平均值 (認識到該平均值不會突出顯示該分鐘期間發生的寫入活動中的任何大型但短暫的峰值)。如需有關比較平均耗用寫入容量與佈建寫入容量的詳細資訊，請參閱 the section called “使用指標” 單位：Count 維度：Keyspace, TableName 有效的統計數字： • Minimum— 任何個別要求對表格所使用的寫入容量單位數目下限。 • Maximum— 任何個別要求對表格所使用的寫入容量單位數目上限。 • Average：每個請求消耗的平均寫入容量。 **Note** 此 Average 數值受樣本數值會為零的閒置時段所影響。 • Sum：所耗用的寫入容量單位總數。這是 ConsumedWriteCapacityUnits 指標最實用的統計數字。

指標	描述
	• SampleCount — 即使沒有消耗寫入容量，對 Amazon Keyspaces 的請求數量也是如此。 **Note** 此 SampleCount 數值受樣本數值會為零的閒置時段所影響。
MaxProvisionedTableReadCapacityUtilization	帳戶最高佈建的讀取表格所使用的佈建讀取容量單位百分比。 單位：Percent 有效的統計數字： • Maximum：帳戶最高佈建讀取資料表所使用的佈建讀取容量單位百分比上限。 • Minimum：帳戶最高佈建讀取資料表所使用的佈建讀取容量單位百分比下限。 • Average：帳戶最高佈建讀取資料表所使用的平均佈建讀取容量單位百分比。指標會每隔五分鐘發佈一次。因此，如果您快速調整佈建的讀取容量單位，此統計數字可能無法反映真實的平均值。

指標	描述
MaxProvisionedTableWriteCapacityUtilization	帳戶最高佈建的寫入表所使用的佈建寫入容量百分比。 單位：Percent 有效的統計數字： • Maximum— 帳戶最高佈建的寫入表格所使用的佈建寫入容量單位的最大百分比。 • Minimum— 帳戶的最高佈建寫入表所使用的佈建寫入容量單位的最小百分比。 • Average— 帳戶最高佈建的寫入表格所使用的佈建寫入容量單位的平均百分比。指標會每隔五分鐘發佈一次。因此，如果您快速調整佈建的寫入容量單位，此統計數字可能無法反映真實的平均值。
PerConnectionRequestRateExceeded	超出每個連線要求率配額的 Amazon Keyspaces 的請求。每個用戶端連線至 Amazon Keyspaces，每秒最多可支援 3000 個 CQL 請求。用戶端可以建立多個連線來增加輸送量。 當您使用多區域複寫時，每個複寫的寫入也會有助於此配額。最佳作法是，建議您增加資料表的連線數目，以避免發PerConnectionRequestRateExceeded 生錯誤。您可以在 Amazon Keyspaces 中擁有的連線數量沒有限制。 單位：Count 維度：Keyspace, TableName, Operation 有效的統計數字： • SampleCount • Sum

指標	描述
ProvisionedReadCapacityUnits	表格佈建的讀取容量單位數目。 TableName 維度會傳回ProvisionedReadCapacityUnits 表格的。 單位：Count 維度：Keyspace, TableName 有效的統計數字： • Minimum：佈建讀取容量的最低設定。如果使用 ALTER TABLE 增加讀取容量，此指標會顯示在此時段內佈建的 ReadCapacityUnits 最低數值。 • Maximum：佈建讀取容量的最高設定。如果使用 ALTER TABLE 減少讀取容量，此指標會顯示在此時段內佈建的 ReadCapacityUnits 最高數值。 • Average：平均佈建讀取容量。ProvisionedReadCapacityUnits 指標會每隔五分鐘發佈一次。因此，如果您快速調整佈建的讀取容量單位，此統計數字可能無法反映真實的平均值。

指標	描述
ProvisionedWriteCapacityUnits	表格佈建的寫入容量單位數目。 TableName 維度會傳回ProvisionedWriteCapacityUnits 表格的。 單位：Count 維度：Keyspace, TableName 有效的統計數字： • Minimum：佈建寫入容量的最低設定。如果使用 ALTER TABLE 增加寫入容量，此指標會顯示在此時段內佈建的 WriteCapacityUnits 最低數值。 • Maximum：佈建寫入容量的最高設定。如果使用 ALTER TABLE 減少寫入容量，此指標會顯示在此時段內佈建的 WriteCapacityUnits 最高數值。 • Average：佈建平均的寫入容量。ProvisionedWriteCapacityUnits 指標會每隔五分鐘發佈一次。因此，如果您快速調整佈建的寫入容量單位，此統計數字可能無法反映真實的平均值。
ReadThrottleEvents	超出表格佈建讀取容量的 Amazon Keyspaces 的請求，或帳戶層級配額、每個連線配額的請求或分區層級配額。 單位：Count 維度：Keyspace, TableName, Operation 有效的統計數字： • SampleCount • Sum

指標	描述
ReplicationLatency	此測量結果僅適用於多區域金鑰空間updates，inserts並測量deletes從一個複本表格複製到多區域索引鍵空間中的另一個複本表格所花的時間。 單位：Millisecond 維度：TableName, ReceivingRegion 有效的統計數字： • Average • Maximum • Minimum
ReturnedItemCountBySelect	指定期間內，多資料列SELECT查詢所傳回的資料列數目。多列SELECT查詢是不包含完全限定主鍵的查詢，例如全表掃描和範圍查詢。 傳回的資料列數目不一定與已評估的資料列數目相同。例如，假設您在SELECT *具有 100 個資料列的資料表ALLOW FILTERING 上要求 a for，但指定了縮小結果的WHERE子句，因此只傳回 15 個資料列。在這種情況下，來自ScanCount 的回應SELECT將包含 100 和 a Count 的 15 個傳回資料列。 單位：Count 維度：Keyspace, TableName, Operation 有效的統計數字： • Minimum • Maximum • Average • SampleCount • Sum

指標	描述
StoragePartitionThroughputCapacityExceeded	對超出分割區輸送量容量的 Amazon Keyspaces 儲存分割區的請求。Amazon Keyspaces 儲存分割區每秒最多可支援 1000 個 WCU/WRU 和每秒 3000 個 RCU/RRU。我們建議您檢閱資料模型，以將讀取/寫入流量分配到更多分割區，以減輕這些例外 **Note** 邏輯 Amazon Keyspaces 分割區可以跨越多個儲存分區，而且大小幾乎沒有限制。 單位：Count 維度：Keyspace, TableName, Operation 有效的統計數字： • SampleCount • Sum
SuccessfulRequestCount	指定期間內處理的成功要求數目。 單位：Count 維度：Keyspace, TableName, Operation 有效的統計數字： • SampleCount

指標	描述
SuccessfulRequestLatency	在指定時間段內成功向 Amazon Keyspaces 間要求。SuccessfulRequestLatency 可以提供兩種不同類型的信息： • 成功請求的經過時間 (Minimum、Maximum、Sum 或 Average)。 • 成功請求的數量 (SampleCount)。 SuccessfulRequestLatency 僅反映 Amazon Keyspaces 內的活動，不會考慮網路延遲或用戶端活動。 單位：Milliseconds 維度：Keyspace, TableName, Operation 有效的統計數字： • Minimum • Maximum • Average • SampleCount
SystemErrors	在指定時間段內產生的 Amazon Keyspaces ServerError 間的請求。A ServerError 通常表示內部服務錯誤。 單位：Count 維度：Keyspace, TableName, Operation 有效的統計數字： • Sum • SampleCount

指標	描述
SystemReconciliationDeletes	啟用用戶端時間戳記時，用來刪除標記資料的單位。每個都SystemReconciliationDelete 提供足夠的容量，可以刪除或更新每列最多 1KB 的資料。例如，若要更新儲存 2.5 KB 資料的資料列，並同時刪除資料列中的一或多個資料行，則需要 3 SystemReconciliationDeletes 。或者，若要刪除包含 3.5 KB 資料的整個資料列，則需要 4 SystemReconciliationDeletes 。 單位：Count 維度：Keyspace, TableName 有效的統計數字： • Sum— 在一段時間內SystemReconciliationDeletes 消耗的總數。
TTLDeletes	使用存留時間 (TTL) 刪除或更新連續資料所耗用的單位。每個都TTLDelete 提供足夠的容量，可以刪除或更新每列最多 1KB 的資料。例如，若要更新儲存 2.5 KB 資料的資料列，並同時刪除資料列中的一或多個資料行，則需要刪除 3 次 TTL。或者，若要刪除包含 3.5 KB 資料的整個資料列，則需要刪除 4 次 TTL。 單位：Count 維度：Keyspace, TableName 有效的統計數字： • Sum— 在一段時間內TTLDeletes 消耗的總數。

指標	描述
UserErrors	對 Amazon Keyspaces 間的請求，這些金鑰空間會在指定的時間段內產生InvalidRequest 錯誤。InvalidRequest 通常表示用戶端錯誤，例如參數組合無效、嘗試更新不存在的資料表或不正確的要求簽章。 UserErrors 代表目前 AWS 區域 和目前無效要求的彙總 AWS 帳戶。 單位：Count 維度：Keyspace, TableName, Operation 有效的統計數字： • Sum • SampleCount
WriteThrottleEvents	超出表格佈建寫入容量或帳戶層級配額、每個連線配額的請求或分區層級配額的 Amazon Keyspaces 的請求。 單位：Count 維度：Keyspace, TableName, Operation 有效的統計數字： • SampleCount • Sum

Amazon Keyspaces 指標的維度

Amazon Keyspaces 的指標由帳戶、表格名稱或作業的值限定。您可以使用 CloudWatch 主控台沿下表中的任何維度擷取 Amazon Keyspaces 資料。

維度	描述
Keyspace	此維度會將資料限制在特定的金鑰空間。該值可以是當前區域和當前中的任何密鑰空間。 AWS 帳戶
Operation	此維度將資料限制在 Amazon Keyspaces CQL 操作之一，例如INSERT或SELECT操作。
TableName	此維度將資料限制為特定資料表。該值可以是當前區域和當前的任何表名 AWS 帳戶。如果資料表名稱在帳戶中不是唯一的，您也必須指定Keyspace。

創建 CloudWatch 警報來監控 Amazon Keyspaces

您可以為 Amazon Keyspaces 創建 Amazon CloudWatch 警報，該警報在警報狀態更改時發送亞馬遜簡單通知服務（Amazon SNS）消息。警示會在您指定的期間監看單一指標。警示會根據在數個期間與指定閾值相關的指標值，來執行一個或多個動作。動作是傳送至 Amazon SNS 主題或 Application Auto Scaling 政策的通知。

當您以「應用程式自動擴展」的佈建模式使用 Amazon Keyspaces 時，服務會代表您建立兩對 CloudWatch 警示。每個配對代表您已佈建和已耗用輸送量設定的上限和下限。當表格的實際使用率在持續一段時間內偏離目標使用率時，就會觸發這些 CloudWatch 警示。若要深入瞭解 Application Auto Scaling 所建立的 CloudWatch 警示，請參閱the section called “Amazon Keyspaces 自動擴展如何工作”。

警示只會呼叫持續狀態變更的動作。 CloudWatch 警報不會僅僅因為它們處於特定狀態而叫用動作。狀態必須已變更，且在指定的期間數內維持此狀態。

如需有關建立 CloudWatch 警示的詳細資訊，請參閱 Amazon 使用 CloudWatch 者指南中的使用 Amazon CloudWatch 警示。

記錄 Amazon Keyspaces API 調用 AWS CloudTrail

Amazon Keyspaces 與服務整合在一起 AWS CloudTrail，該服務可提供 Amazon 密 Keyspaces 中使用者、角色或 AWS 服務所採取的動作記錄。 CloudTrail 將 Amazon Keyspaces 的資料定義語言 (DDL) API 呼叫和資料操縱語言 (DML) API 呼叫擷取為事件。擷取的呼叫包括來自 Amazon Keyspaces 主控台的呼叫，以及對 Amazon Keyspaces API 作業的程式設計呼叫。

如果您建立追蹤，您可以啟用持續交付 CloudTrail 事件到 Amazon Simple Storage Service (Amazon S3) 儲存貯體，包括 Amazon Keyspaces 的事件。

如果您未設定追蹤，您仍然可以在 [事件歷程記錄] 中檢視 CloudTrail主控台上最近支援的事件。使用收集的資訊 CloudTrail，您可以判斷向 Amazon Keyspaces 發出的請求、提出請求的 IP 地址、提出請求的人員、提出請求的時間以及其他詳細資訊。

若要進一步了解 CloudTrail，請參閱使AWS CloudTrail 用者指南。

主題

- 在中配置 Amazon Keyspaces 日誌文件條目 CloudTrail
- Amazon Keyspaces 數據定義語言（DDL）信息 CloudTrail
- Amazon Keyspaces 數據操縱語言（DML）信息 CloudTrail
- 了解 Amazon Keyspaces 日誌文件條目

在中配置 Amazon Keyspaces 日誌文件條目 CloudTrail

登入的每個 Amazon Keyspaces API 動作都 CloudTrail 包含以 CQL 查詢語言表示的請求參數。如需更多資訊，請參閱語言參考。

您可以檢視、搜尋和下載 AWS 帳戶的最新事件。如需詳細資訊，請參閱使用 CloudTrail 事件歷程記錄檢視事件。

對於您的事件的持續記錄 AWS 帳戶，包括 Amazon Keyspaces 的事件，請創建一個跟踪。追蹤可 CloudTrail 將日誌檔交付到 Amazon S3 儲存貯體。根據預設，當您在主控台中建立追蹤時，追蹤會套用至所有 AWS 區域。追蹤記錄來自 AWS 分區中所有區域的事件，並將日誌檔傳送到您指定的 Amazon S3 儲存貯體。此外，您還可以設定其他 AWS 服務，以進一步分析 CloudTrail 記錄中收集的事件資料並採取行動。

如需詳細資訊，請參閱 AWS CloudTrail 使用者指南中的以下主題：

- 建立追蹤的概觀
- CloudTrail 支援的服務與整合
- 設定 Amazon SNS 通知 CloudTrail
- 從多個區域接收 CloudTrail 記錄檔
- 從多個帳戶接收 CloudTrail 日誌文件

每一筆事件或日誌專案都會包含產生請求者的資訊。身分資訊可協助您判斷下列事項：

- 要求是使用根使用者登入資料還是 AWS Identity and Access Management (IAM) 使用者登入資料提出。
- 提出該請求時，是否使用了特定角色或聯合身分使用者的暫時安全憑證。
- 請求是否由其他 AWS 服務提出。

如需詳細資訊，請參閱CloudTrail 使 userIdentity 元素。

Amazon Keyspaces 數據定義語言（DDL）信息 CloudTrail

CloudTrail 在您創建帳戶 AWS 帳戶 時啟用。當在 Amazon Keyspaces 中發生 DDL 活動時，該活動會與事件歷史記錄中的其他 AWS 服務 CloudTrail 事件一起自動記錄為事件。下表顯示了針對 Amazon Keyspaces 記錄的 DDL 語句。

CloudTrail **eventName**	陳述式	定制列表動作	AWS SDK 動作
CreateKeyspace	DDL	`CREATE KEYSPACE`	`CreateKeyspace`
DropKeyspace	DDL	`DROP KEYSPACE`	`DeleteKeyspace`
CreateTable	DDL	`CREATE TABLE`	`CreateTable`
DropTable	DDL	`DROP TABLE`	`DeleteTable`
AlterTable	DDL	`ALTER TABLE`	`UpdateTable`，`TagResource`，`UntagResource`

Amazon Keyspaces 數據操縱語言（DML）信息 CloudTrail

若要啟用記錄 Amazon Keyspaces DML 陳述式 CloudTrail，您必須先在中啟用資料平面 API 活動的記錄。 CloudTrail您可以選擇使用 CloudTrail 主控台記錄資料事件類型 Cassandra 表的活動，或將`resources.type`值設定為使用 AWS CLI 或 API 操作，以在新的或現有的追蹤中開始記錄 Amazon Keyspaces DML 事件。`AWS::Cassandra::Table` CloudTrail 如需詳細資訊，請參閱記錄資料事件。

下表顯示的是由記錄的資料事 CloudTrail 件Cassandra table。

CloudTrail **eventName**	陳述式	定制列表動作	AWS SDK 動作
Select	DML	SELECT	GetKeyspace ，GetTable, ListKeyspaces ，ListTables ListTagsForResource
Insert	DML	INSERT	沒有可用的 AWS SDK 動作
更新	DML	UPDATE	沒有可用的 AWS SDK 動作
Delete	DML	DELETE	沒有可用的 AWS SDK 動作

了解 Amazon Keyspaces 日誌文件條目

CloudTrail 記錄檔包含一或多個記錄項目。事件代表來自任何來源的單一請求，包括有關請求的操作，動作的日期和時間，請求參數等信息。 CloudTrail 日誌文件不是公共 API 調用的有序堆棧跟踪，因此它們不會以任何特定順序顯示。

下列範例顯示示範CreateKeyspace、DropKeyspace、CreateTable和DropTable動作的 CloudTrail 記錄項目 ：

```
{
  "Records": [
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:alice",
        "arn": "arn:aws:sts::111122223333:assumed-role/users/alice",
        "accountId": "111122223333",
```

```
        "sessionContext": {
          "sessionIssuer": {
            "type": "Role",
            "principalId": "AKIAIOSFODNN7EXAMPLE",
            "arn": "arn:aws:iam::111122223333:role/Admin",
            "accountId": "111122223333",
            "userName": "Admin"
          },
          "webIdFederationData": {},
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2020-01-15T18:47:56Z"
          }
        }
      },
      "eventTime": "2020-01-15T18:53:04Z",
      "eventSource": "cassandra.amazonaws.com",
      "eventName": "CreateKeyspace",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "10.24.34.01",
      "userAgent": "Cassandra Client/ProtocolV4",
      "requestParameters": {
        "rawQuery": "\n\tCREATE KEYSPACE \"mykeyspace\"\n\tWITH\n\t\tREPLICATION =
{'class': 'SingleRegionStrategy'}\n\t\t",
        "keyspaceName": "mykeyspace"
      },
      "responseElements": null,
      "requestID": "bfa3e75d-bf4d-4fc0-be5e-89d15850eb41",
      "eventID": "d25beae8-f611-4229-877a-921557a07bb9",
      "readOnly": false,
      "resources": [
        {
          "accountId": "111122223333",
          "type": "AWS::Cassandra::Keyspace",
          "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/"
        }
      ],
      "eventType": "AwsApiCall",
      "apiVersion": "3.4.4",
      "recipientAccountId": "111122223333",
      "managementEvent": true,
      "eventCategory": "Management",
      "tlsDetails": {
          "tlsVersion": "TLSv1.2",
```

```
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
    },
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:alice",
        "arn": "arn:aws:sts::111122223333:assumed-role/users/alice",
        "accountId": "111122223333",
        "sessionContext": {
          "sessionIssuer": {
            "type": "Role",
            "principalId": "AKIAIOSFODNN7EXAMPLE",
            "arn": "arn:aws:iam::111122223333:role/Admin",
            "accountId": "111122223333",
            "userName": "Admin"
          },
          "webIdFederationData": {},
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2020-01-15T18:47:56Z"
          }
        }
      },
      "eventTime": "2020-01-15T19:28:39Z",
      "eventSource": "cassandra.amazonaws.com",
      "eventName": "DropKeyspace",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "10.24.34.01",
      "userAgent": "Cassandra Client/ProtocolV4",
      "requestParameters": {
        "rawQuery": "DROP KEYSPACE \"mykeyspace\"",
        "keyspaceName": "mykeyspace"
      },
      "responseElements": null,
      "requestID": "66f3d86a-56ae-4c29-b46f-abcd489ed86b",
      "eventID": "e5aebeac-e1dd-41e3-a515-84fe6aaabd7b",
      "readOnly": false,
      "resources": [
        {
          "accountId": "111122223333",
          "type": "AWS::Cassandra::Keyspace",
          "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/"
```

```
        }
      ],
      "eventType": "AwsApiCall",
      "apiVersion": "3.4.4",
      "recipientAccountId": "111122223333",
      "managementEvent": true,
      "eventCategory": "Management",
      "tlsDetails": {
          "tlsVersion": "TLSv1.2",
          "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
          "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
    },
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:alice",
        "arn": "arn:aws:sts::111122223333:assumed-role/users/alice",
        "accountId": "111122223333",
        "sessionContext": {
          "sessionIssuer": {
            "type": "Role",
            "principalId": "AKIAIOSFODNN7EXAMPLE",
            "arn": "arn:aws:iam::111122223333:role/Admin",
            "accountId": "111122223333",
            "userName": "Admin"
          },
          "webIdFederationData": {},
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2020-01-15T18:47:56Z"
          }
        }
      },
      "eventTime": "2020-01-15T18:55:24Z",
      "eventSource": "cassandra.amazonaws.com",
      "eventName": "CreateTable",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "10.24.34.01",
      "userAgent": "Cassandra Client/ProtocolV4",
      "requestParameters": {
        "rawQuery": "\n\tCREATE TABLE \"mykeyspace\".\"mytable\"(\n\t\t\"ID\" int,
\n\t\t\"username\" text,\n\t\t\"email\" text,\n\t\t\"post_type\" text,\n\t\tPRIMARY
 KEY((\"ID\", \"username\", \"email\")))",
```

```
        "keyspaceName": "mykeyspace",
        "tableName": "mytable"
      },
      "responseElements": null,
      "requestID": "5f845963-70ea-4988-8a7a-2e66d061aacb",
      "eventID": "fe0dbd2b-7b34-4675-a30c-740f9d8d73f9",
      "readOnly": false,
      "resources": [
        {
          "accountId": "111122223333",
          "type": "AWS::Cassandra::Table",
          "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/
mytable"
        }
      ],
      "eventType": "AwsApiCall",
      "apiVersion": "3.4.4",
      "recipientAccountId": "111122223333",
      "managementEvent": true,
      "eventCategory": "Management",
      "tlsDetails": {
          "tlsVersion": "TLSv1.2",
          "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
          "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
    },
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:alice",
        "arn": "arn:aws:sts::111122223333:assumed-role/users/alice",
        "accountId": "111122223333",
        "sessionContext": {
          "sessionIssuer": {
            "type": "Role",
            "principalId": "AKIAIOSFODNN7EXAMPLE",
            "arn": "arn:aws:iam::111122223333:role/Admin",
            "accountId": "111122223333",
            "userName": "Admin"
          },
          "webIdFederationData": {},
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2020-01-15T18:47:56Z"
```

```
          }
        }
      },
      "eventTime": "2020-01-15T19:27:59Z",
      "eventSource": "cassandra.amazonaws.com",
      "eventName": "DropTable",
      "awsRegion": "us-east-1",
      "sourceIPAddress": "10.24.34.01",
      "userAgent": "Cassandra Client/ProtocolV4",
      "requestParameters": {
        "rawQuery": "DROP TABLE \"mykeyspace\".\"mytable\"",
        "keyspaceName": "mykeyspace",
        "tableName": "mytable"
      },
      "responseElements": null,
      "requestID": "025501b0-3582-437e-9d18-8939e9ef262f",
      "eventID": "1a5cbedc-4e38-4889-8475-3eab98de0ffd",
      "readOnly": false,
      "resources": [
        {
          "accountId": "111122223333",
          "type": "AWS::Cassandra::Table",
          "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/
mytable"
        }
      ],
      "eventType": "AwsApiCall",
      "apiVersion": "3.4.4",
      "recipientAccountId": "111122223333",
      "managementEvent": true,
      "eventCategory": "Management",
      "tlsDetails": {
          "tlsVersion": "TLSv1.2",
          "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
          "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
    }
  ]
}
```

下列記錄檔顯示SELECT陳述式的範例。

```
{
    "eventVersion": "1.09",
```

```
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/alice",
        "accountId": "111122223333",
        "userName": "alice"
    },
    "eventTime": "2023-11-17T10:38:04Z",
    "eventSource": "cassandra.amazonaws.com",
    "eventName": "Select",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "10.24.34.01",
    "userAgent": "Cassandra Client/ProtocolV4",
    "requestParameters": {
          "keyspaceName": "my_keyspace",
          "tableName": "my_table",
          "conditions": [
              "pk = **(Redacted)",
              "ck < 3**(Redacted)0",
              "region = 't**(Redacted)t'"
          ],
          "select": [
              "pk",
              "ck",
              "region"
          ],
          "allowFiltering": true
    },
    "responseElements": null,
    "requestID": "6d83bbf0-a3d0-4d49-b1d9-e31779a28628",
    "eventID": "e00552d3-34e9-4092-931a-912c4e08ba17",
    "readOnly": true,
    "resources": [
        {
            "accountId": "111122223333",
            "type": "AWS::Cassandra::Table",
            "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/
table/my_table"
        }
    ],
    "eventType": "AwsApiCall",
    "apiVersion": "3.4.4",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
```

```
    "eventCategory": "Data",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
    }
}
```

下列記錄檔顯示INSERT陳述式的範例。

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/alice",
        "accountId": "111122223333",
        "userName": "alice"
    },
    "eventTime": "2023-12-01T22:11:43Z",
    "eventSource": "cassandra.amazonaws.com",
    "eventName": "Insert",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "10.24.34.01",
    "userAgent": "Cassandra Client/ProtocolV4",
    "requestParameters": {
          "keyspaceName": "my_keyspace",
          "tableName": "my_table",
          "primaryKeys": {
              "pk": "**(Redacted)",
              "ck": "1**(Redacted)8"
          },
          "columnNames": [
              "pk",
              "ck",
              "region"
          ],
          "updateParameters": {
              "TTL": "2**(Redacted)0"
          }
      }
    },
    "responseElements": null,
```

```
    "requestID": "edf8af47-2f87-4432-864d-a960ac35e471",
    "eventID": "81b56a1c-9bdd-4c92-bb8e-92776b5a3bf1",
    "readOnly": false,
    "resources": [
        {
          "accountId": "111122223333",
          "type": "AWS::Cassandra::Table",
          "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/table/
my_table"
        }
    ],
    "eventType": "AwsApiCall",
    "apiVersion": "3.4.4",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
    }
}
```

下列記錄檔顯示UPDATE陳述式的範例。

```
{
    "eventVersion": "1.09",
    "userIdentity": {
          "type": "IAMUser",
          "principalId": "AKIAIOSFODNN7EXAMPLE",
          "arn": "arn:aws:iam::111122223333:user/alice",
          "accountId": "111122223333",
          "userName": "alice"
    },
    "eventTime": "2023-12-01T22:11:43Z",
    "eventSource": "cassandra.amazonaws.com",
    "eventName": "Update",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "10.24.34.01",
    "userAgent": "Cassandra Client/ProtocolV4",
    "requestParameters": {
          "keyspaceName": "my_keyspace",
          "tableName": "my_table",
```

```
            "primaryKeys": {
                "pk": "'t**(Redacted)t'",
                "ck": "'s**(Redacted)g'"
            },
            "assignmentColumnNames": [
               "nonkey"
            ],
            "conditions": [
                "nonkey < 1**(Redacted)7"
            ]
        },
        "responseElements": null,
        "requestID": "edf8af47-2f87-4432-864d-a960ac35e471",
        "eventID": "81b56a1c-9bdd-4c92-bb8e-92776b5a3bf1",
        "readOnly": false,
        "resources": [
            {
              "accountId": "111122223333",
              "type": "AWS::Cassandra::Table",
              "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/table/
my_table"
            }
        ],
        "eventType": "AwsApiCall",
        "apiVersion": "3.4.4",
        "managementEvent": false,
        "recipientAccountId": "111122223333",
        "eventCategory": "Data",
        "tlsDetails": {
              "tlsVersion": "TLSv1.3",
              "cipherSuite": "TLS_AES_128_GCM_SHA256",
              "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
        }
}
```

下列記錄檔顯示DELETE陳述式的範例。

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AKIAIOSFODNN7EXAMPLE",
        "arn": "arn:aws:iam::111122223333:user/alice",
```

```
        "accountId": "111122223333",
        "userName": "alice",
    },
    "eventTime": "2023-10-23T13:59:05Z",
    "eventSource": "cassandra.amazonaws.com",
    "eventName": "Delete",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "10.24.34.01",
    "userAgent": "Cassandra Client/ProtocolV4",
    "requestParameters": {
        "keyspaceName": "my_keyspace",
        "tableName": "my_table",
        "primaryKeys": {
            "pk": "**(Redacted)",
            "ck": "**(Redacted)"
        },
        "conditions": [],
        "deleteColumnNames": [
            "m",
            "s"
        ],
        "updateParameters": {}
    },
    "responseElements": null,
    "requestID": "3d45e63b-c0c8-48e2-bc64-31afc5b4f49d",
    "eventID": "499da055-c642-4762-8775-d91757f06512",
    "readOnly": false,
    "resources": [
      {
        "accountId": "111122223333",
        "type": "AWS::Cassandra::Table",
        "ARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/table/
my_table"
      }
    ],
    "eventType": "AwsApiCall",
    "apiVersion": "3.4.4",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "cassandra.us-east-1.amazonaws.com"
```

```
    }
}
```

亞馬遜 Keyspaces 中的安全性（阿帕奇卡桑德拉）

雲端安全是 AWS 最重視的一環。身為 AWS 客戶的您，將能從資料中心和網路架構的建置中獲益，以滿足組織最為敏感的安全要求。

安全是 AWS 與您共同肩負的責任。共同的責任模式將其稱為雲端的安全性和雲端中的安全性：

- 雲端本身的安全 – AWS 負責保護執行 AWS 雲端 內 AWS 服務的基礎設施。AWS 提供的服務，也可讓您安全使用。第三方稽核人員定期檢測及驗證安全的效率也是我們 AWS 合規計劃的一部分。若要了解適用於 Amazon Keyspaces 的合規計劃，請參閱合規計劃的範圍內的AWS服務。
- 雲端內部的安全：您的責任取決於所使用的 AWS 服務。您也必須對資料敏感度、組織要求，以及適用法律和法規等其他因素負責。

本文件將協助您瞭解如何在使用 Amazon Keyspaces 時套用共同責任模型。下列主題說明如何設定 Amazon Keyspaces 以符合安全和合規目標。您還將學習如何使用其他可以幫助您監控和保護 Amazon Keyspaces 資源的AWS服務。

主題

- Amazon Keyspaces 中的數據保護
- AWS Identity and Access Management 對於 Amazon Keyspaces
- Amazon Keyspaces 的合規驗證（阿帕奇卡桑德拉）
- Amazon Keyspaces 中的復原和災難復原功能
- Amazon 密鑰空間中的基礎設施安全
- Amazon Keyspaces space 的組態與漏洞分析
- Amazon Keyspaces 的安全最佳實務

Amazon Keyspaces 中的數據保護

AWS共同責任模型適用於 Amazon Keyspaces 中的數據保護（適用於 Apache Cassandra）。如此模型所述，AWS 負責保護執行所有 AWS 雲端 的全球基礎設施。您負責維護在此基礎設施上託管內容的控制權。您也必須負責您所使用 AWS 服務 的安全組態和管理任務。如需有關資料隱私權的更多相關資訊，請參閱資料隱私權常見問答集。如需有關歐洲資料保護的相關資訊，請參閱 AWS 安全性部落格上的 AWS 共同的責任模型和 GDPR 部落格文章。

基於資料保護目的，建議您使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 保護 AWS 帳戶 憑證，並設定個人使用者。如此一來，每個使用者都只會獲得授與完成其任務所必須的許可。我們也建議您採用下列方式保護資料：

- 每個帳戶均要使用多重要素驗證 (MFA)。
- 使用 SSL/TLS 與 AWS 資源通訊。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 使用 AWS CloudTrail 設定 API 和使用者活動日誌記錄。
- 使用 AWS 加密解決方案，以及 AWS 服務 內的所有預設安全控制項。
- 使用進階的受管安全服務（例如 Amazon Macie），協助探索和保護儲存在 Amazon S3 的敏感資料。
- 如果您在透過命令列介面或 API 存取 AWS 時，需要 FIPS 140-2 驗證的加密模組，請使用 FIPS 端點。如需有關 FIPS 和 FIPS 端點的更多相關資訊，請參閱聯邦資訊處理標準 (FIPS) 140-2 概觀。

我們強烈建議您絕對不要將客戶的電子郵件地址等機密或敏感資訊，放在標籤或自由格式的文字欄位中，例如 Name (名稱) 欄位。這包括當您使用主控台、API 或AWS開發套件AWS 服務使用 Amazon Keyspaces 或其他金鑰空間時。AWS CLI您在標籤或自由格式文字欄位中輸入的任何資料都可能用於計費或診斷日誌。如果您提供外部伺服器的 URL，我們強烈建議請勿在驗證您對該伺服器請求的 URL 中包含憑證資訊。

主題

Amazon 密 Keyspaces 的靜態加密

Amazon 金 Keyspaces (適用於 Apache Cassandra) 靜態加密使用存放在 AWS Key Management Service(AWS KMS) 的加密金鑰提供加強的安全性。此功能協助降低了保護敏感資料所涉及的操作負擔和複雜性。透過靜態加密，您可以建置對安全性敏感的應用程式，以符合嚴格的資料保護法規要求。

Amazon Keyspaces 靜態加密功能會使用 256 位元的進階加密標準 (AES-256) 來加密您的資料。如此有助於防止對底層儲存未經授權的存取，保護您的資料。

Amazon Keyspaces 會以透明方式加密和解密資料。Amazon 金 Keyspaces 使用信封加密和金鑰階層來保護資料加密金鑰。它與用AWS KMS於存儲和管理根加密密鑰集成。如需關於加密金鑰階層的更

多資訊，請參閱the section called “運作方式”。如需有關信封加密等AWS KMS概念的詳細資訊，請參閱AWS Key Management Service開發人員指南中的AWS KMS管理服務概念。

建立新的資料表時，您可以選擇下列其中一個金AWS KMS鑰 (KMS 金鑰)：

- AWS 擁有的金鑰— 這是預設的加密類型。該金鑰是由 Amazon 金 Keyspaces 所擁有 (不需額外費用)。
- 客戶受管金鑰：此金鑰會存放在您的帳戶中，並且由您建立、擁有且管理。您有客戶受管金鑰的完整控制權 (須AWS KMS支付費用)。

您可以隨時在AWS 擁有的金鑰和客戶受管金鑰之間切換。當您在現有資料表或變更現有資料表的 KMS 金鑰時，您可以使用主控台或以程式設計方式使用 CQL 陳述式指定客戶受管金鑰。如要瞭解如何作業，請參閱 靜態加密：如何使用客戶受管金鑰加密 Amazon 金 Keyspaces 中的表格。

使用預設選項進行靜態加密無須額外付費。AWS 擁有的金鑰不過，客戶受管金鑰須AWS KMS支付費用。如需定價的詳細資訊，請參閱 AWS KMS 定價。

Amazon Keyspace 靜態加密已在所有區域中使用AWS 區域，包括中AWS國 (北京) 和中國 (北京) 和AWS中國 (北京) 區域。如需詳細資訊，請參閱 靜態加密：如何在 Amazon Keyspaces ockets 中運作。

主題

- 靜態加密：如何在 Amazon Keyspaces ockets 中運作
- 靜態加密：如何使用客戶受管金鑰加密 Amazon 金 Keyspaces 中的表格

靜態加密：如何在 Amazon Keyspaces ockets 中運作

Amazon Sockets 靜態加 Keyspaces 靜態加密靜態加密靜態加密靜態加密功能會使用 256 位元的進階加密標準 (AES-256) 來加密您的資料。這有助於防止對底層儲存未經授權的存取，進而保護您的資料 Amazon Keyspaces 資料表中的所有客戶資料預設為靜態加密，伺服器端加密是透明的，這表示不需要變更應用程式。

靜態加密會與 AWS Key Management Service (AWS KMS) 整合，以管理用於加密資料表的加密金鑰。建立新的資料表或更新現有資料表時，您可以選擇下列其中一個AWS KMS金鑰選項：

- AWS 擁有的金鑰— 這是預設的加密類型。該金鑰是由 Amazon Key Sockets 所擁 Keyspaces (不需額外費用) 。

- 客戶受管金鑰：此金鑰會存放在您的帳戶中，並且由您建立、擁有且管理。您可以完整控制客戶受管金鑰 (須AWS KMS支付費用)。

AWS KMS金鑰 (KMS 金鑰)

靜態加密使用AWS KMS金鑰保護您的所有 Amazon 金 Keyspaces 資料。根據預設，Amazon Keyspaces kets AWS 擁有的金鑰會使用在 Amazon Key Sockets 服務帳戶中建立和管理的多租用戶加密金鑰。

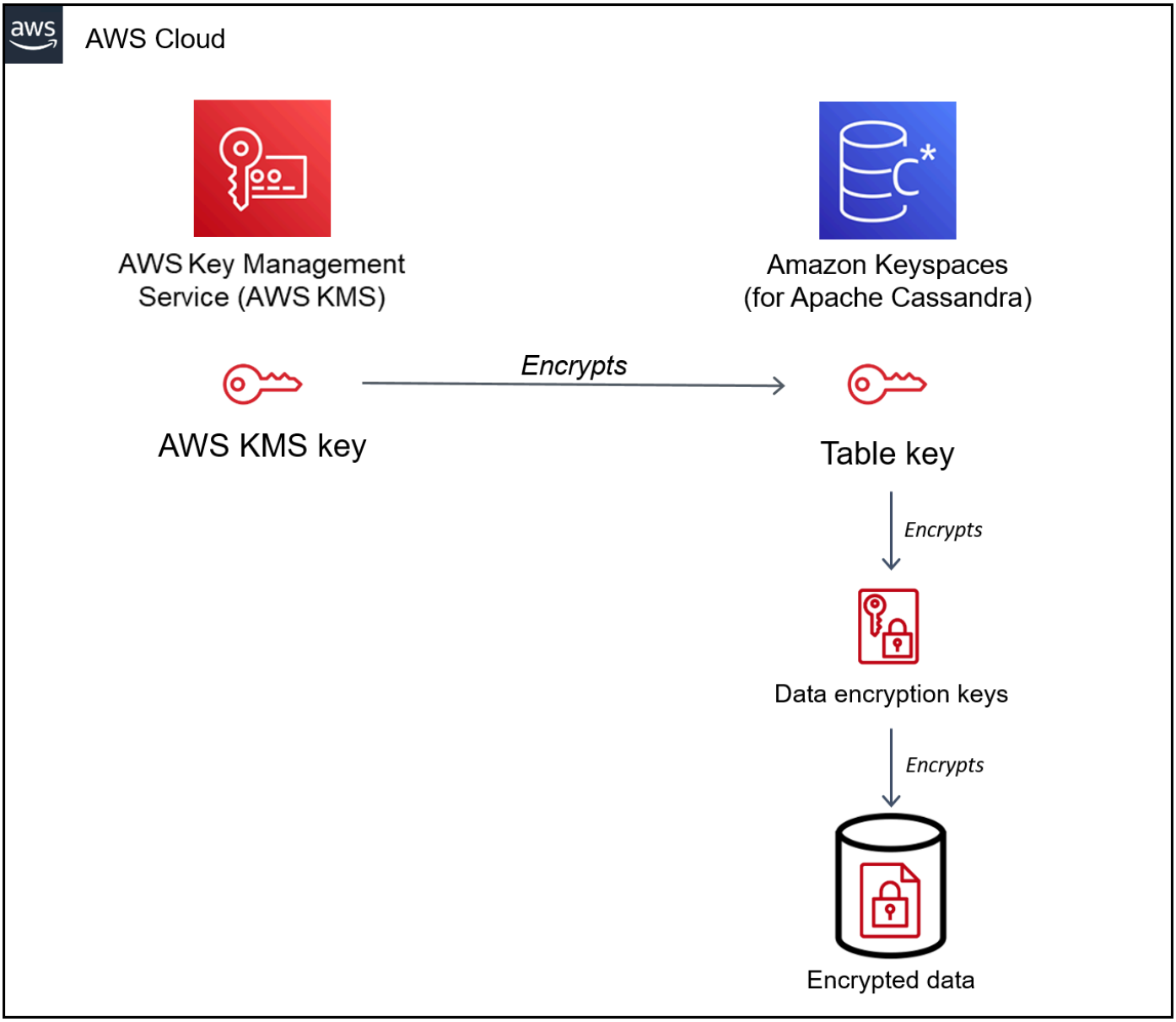
不過，您可以使用您中的客戶管理金鑰來加密 Amazon 金鑰 Keyspaces 表格AWS 帳戶。您可以為金鑰空間中的每個資料表選取不同的 KMS 金鑰。您為資料表選取的 KMS 金鑰也會用來加密其所有中繼資料和可還原的備份。

建立或更新資料表時，您可以選取資料表的 KMS 金鑰。您可以隨時在 Amazon 金鑰空間主控台或使用 ALTER TABLE 陳述式來變更資料表的 KMS 金鑰。切換 KMS 金鑰的程序非常順暢，不需要停機時間或導致服務降級。

金鑰階層

Amazon 金 Keyspaces 使用金鑰階層來加密資料。在此金鑰階層中，KMS 金鑰是根金鑰。它用來加密和解密 Amazon 金 Keyspaces 資料表加密金鑰。表格加密金鑰用於加密 Amazon Keyspaces space 內部使用的加密金鑰，以便在執行讀取和寫入操作時加密和解密資料。

使用加密金鑰階層，您可以變更 KMS 金鑰，而不必重新加密資料或影響應用程式和進行中的資料作業。



表格索引鍵

Amazon 金 Keyspaces 資料表金鑰用作金鑰加密金鑰。Amazon Keyspaces 使用資料表金鑰來保護用於將存放在資料表中的資料、日誌檔和可還原備份中的資料加密金鑰。Amazon Keyspaces 為資料表中的每個基礎結構產生唯一的資料加密金鑰。不過，多個資料表資料列可能會受到相同的資料加密金鑰保護。

當您第一次將 KMS 金鑰設定為客戶受管金鑰時，AWS KMS會產生資料金鑰。數AWS KMS據鍵是指亞馬遜密鑰 Keyspaces 中的表鍵。

當您存取靜態表時，Amazon KMS 金 Keyspaces 會傳送請求AWS KMS給來使用 KMS 金鑰解密資料表金鑰。然後，它會使用純文字表格金鑰來解密 Amazon Keyspaces space 資料加密金鑰，並使用純文字資料加密金鑰來解密表格資料。

Amazon Keyspaces 外部使用和存放資料表金鑰和資料加密金鑰AWS KMS。它使用進階加密標準 (AES) 加密和 256 位元加密金鑰來保護所有金鑰。然後，它將加密密鑰與加密的數據存儲，以便他們可以根據需要解密表數據。

資料表金鑰快取

AWS KMS為了避免每次 Amazon Amazon Sockets 操作都要呼叫，Amazon Keyspaces Sockets 會快取記憶體中每個連線的純文字資料表金鑰。如果 Amazon Keyspaces space 靜態五分鐘後收到快取資料表金鑰的請求，則會傳送新請求給AWS KMS來解密資料表金鑰。此呼叫會擷取自從上次請求解密資料表金鑰以來，AWS KMS或AWS Identity and Access Management (IAM) 中對 KMS 金鑰存取政策所做的任何變更。

封套加密

如果您變更了資料表的客戶受管金鑰，Amazon Keyspace 便會產生新的資料表金鑰。然後，它會使用新的資料表金鑰來重新加密資料加密金鑰。它還使用新的表格密鑰來加密用於保護可還原備份的先前表格密鑰。這個程序稱為封套加密。這樣可確保即使輪換客戶管理的金鑰，您也可以存取可還原的備份。如需封套加密的詳細資訊，請參閱AWS Key Management Service開發人員指南中的封套加密。

主題

- AWS擁有的金鑰
- 客戶受管金鑰
- 靜態加密使用須知

AWS擁有的金鑰

AWS 擁有的金鑰不會儲存在您的AWS 帳戶. 它們是AWS擁有和管理的 KMS 金鑰的一部分，以在多個中使用AWS 帳戶。 AWS服務可用AWS 擁有的金鑰來保護您的資料。

您無法檢視、管理或使用AWS 擁有的金鑰，或稽核其使用方式。不過，您不需要執行任何工作或變更任何程式即可保護會將資料加密的金鑰。

您不需就使用支付每月費用或使用費用AWS 擁有的金鑰，這些費用也不會計入您帳戶的AWS KMS配額中。

客戶受管金鑰

客戶受管金鑰是您AWS 帳戶在中建立、擁有和管理的金鑰。您可以完全控制這些 KMS 金鑰。

使用客戶受管金鑰來取得下列功能：

- 您可以建立和管理客戶受管金鑰，包括設定和維護金鑰政策、IAM 政策和授予，以控制對客戶受管金鑰的存取。您可以啟用和停用客戶受管金鑰、啟用和停用自動輪換金鑰，以及排程在不再使用時刪除該客戶受管金鑰。您可以為您管理的客戶管理金鑰建立標籤和別名。
- 您可以搭配匯入的金鑰材料使用客戶受管金鑰，或是使用位於您所擁有及管理自訂金鑰存放區中的客戶受管金鑰。
- 您可以使用AWS CloudTrail和 Amazon CloudWatch Logs 來追蹤 Amazon Keyspaces Sockets 代表您傳送給AWS KMS的請求。如需詳細資訊，請參閱the section called “步驟 6：使用設定監視AWS CloudTrail”。

客戶受管金鑰會針對每次 API 呼叫產生費用，且這些 KMS 金鑰適用AWS KMS配額。如需詳細資訊，請參閱AWS KMS資源或請求配額。

當您指定一個客戶受管金鑰作為資料表的根加密金鑰時，可還原備份的備份會以建立備份時指定給資料表的加密金鑰相同的加密金鑰。如果資料表的 KMS 金鑰已輪替，金鑰包覆可確保最新的 KMS 金鑰可存取所有可還原的備份。

Amazon 密 Keyspaces 必須能夠存取您的客戶受管金鑰，才能讓您存取表格資料。如果加密金鑰的狀態設定為停用或已排程刪除，Amazon 金 Keyspaces 將無法加密或解密資料。因此，您無法對資料表執行讀取和寫入作業。只要服務偵測到您的加密金鑰無法存取，Amazon Keyspaces 會傳送電子通知提醒您。

您必須在七天內恢復對加密金鑰的存取權，否則 Amazon 密 Keyspaces 會自動刪除您的表格。為了預防措施，Amazon Keyspaces space 會在刪除資料表之前建立表格資料的可還原備份。亞馬遜 Keyspaces 將可恢復的備份保留 35 天。35 天後，您將無法再還原表格資料。您不需支付可還原備份的費用，但需支付標準還原費用。

您可以使用此可還原的備份將資料還原至新表格。若要開始還原，最後一次用於資料表的客戶受管金鑰必須啟用，且 Amazon Keyspace 必須可以存取該資料表。

> **Note**
>
> 當您建立的資料表使用客戶管理的金鑰加密，而該金鑰在建立程序完成之前無法存取或排定要刪除時，就會發生錯誤。建立資料表作業失敗，而且系統會傳送電子郵件通知給您。

靜態加密使用須知

在您使用靜態加密靜態加密靜態加密時請考量以下項目。

- 已在所有 Amazon Amazon Key 資料表表中啟用靜態加密靜態加密，且無法停用。整個表在靜態時都是加密的，您無法選擇特定的列或行進行加密。
- 根據預設，Amazon 密 Keyspaces 會使用單一服務預設金鑰 (AWS 擁有的金鑰) 來加密所有資料表。如果這個金鑰不存在，就會為您建立。服務預設金鑰無法停用。
- 靜態加密只會在永久儲存媒體上靜態 (靜態) 時加密資料。如果資料安全性是傳輸中資料或使用中資料的考量，您必須採取其他措施：
  - 傳輸中資料：Amazon Keyspaces 中的所有資料在傳輸中皆會加密。根據預設，與 Amazon Keyspaces 之間的通訊會受到使用 Secure Sockets Layer (SSL) /Transport Layer (TLS) 加密保護。
  - 使用中資料：在將資料傳送至 Amazon Keyspaces Sockets 前使用用戶端加密來保護資料。
  - 客戶管理金鑰：資料表中的靜態資料一律使用客戶管理的金鑰加密。但是，執行多個資料列不可完全更新的作業會AWS 擁有的金鑰在處理期間使用暫時加密資料 這包括範圍刪除操作和同時訪問靜態和非靜態數據的操作。
- 單一客戶管理金鑰最多可有 50,000 筆授權。與客戶受管金鑰相關聯的每個 Amazon 金鑰 Keyspaces 表會耗用 2 次授權。刪除資料表時，會釋放一個授權。第二個授權用於建立表格的自動快照，以防止 Amazon Keyspaces 無意中無法存取客戶受管金鑰時，防止資料遺失。該授權在刪除表格後 42 天釋放。

靜態加密：如何使用客戶受管金鑰加密 Amazon 金 Keyspaces 中的表格

您可以使用主控台或 CQL 陳述式，AWS KMS key為新資料表指定，並更新 Amazon Key Keyspaces 中現有資料表的加密金鑰。下列主題概述如何為新資料表和現有資料表實作客戶管理金鑰。

主題

- 先決條件：使用建立客戶受管金鑰，AWS KMS並授與權限給 Amazon 金 Keyspaces
- 步驟 3：為新資料表指定客戶受管金鑰
- 步驟 4：更新現有資料表的加密金鑰
- 步驟 5：在日誌中使用 Amazon Keyspaces 加密內容
- 步驟 6：使用設定監視AWS CloudTrail

先決條件：使用建立客戶受管金鑰，AWS KMS並授與權限給 Amazon 金 Keyspaces

您必須先在AWS Key Management Service (AWS KMS) 中建立金鑰，然後授權 Amazon 密鑰空間使用該金鑰，然後授權 Amazon 金鑰空間使用該金鑰，才能使用客戶受管金鑰。

步驟 1：使用建立客戶受管金鑰AWS KMS

若要建立用於保護 Amazon 金鑰空間表的客戶受管金鑰，您可以按照使用主控台或AWS API 建立對稱加密 KMS 金鑰中的步驟進行操作。

步驟 2：授權使用客戶管理的金鑰

在選擇客戶受管金鑰來保護 Amazon Key角資料表之前，該客戶受管金鑰上的政策必須授予 Amazon Keyspaces 許可，以代您使用該金鑰。您有客戶受管金鑰的政策和授予，您有客戶受管金鑰的政策和授予。您可以在金鑰政策、IAM 政策或授予中提供這些許可。

Amazon Keyspaces 不需要額外的授權，就能使用預設AWS 擁有的金鑰保護AWS帳戶中的 Amazon Keyspaces space 資料表。

下列主題說明如何使用允許 Amazon 密鑰 Keyspaces 表使用客戶受管金鑰的 IAM 政策和授權來設定所需的許可。

主題

- 客戶受管理鑰的金鑰政策
- 範例金鑰政策
- 使用授權 Amazon Keyspaces

客戶受管理鑰的金鑰政策

當您選取客戶受管金鑰來保護 Amazon Key角資料表時，Amazon Keyspaces 會取得許可，以代表進行選取的委託人使用客戶受管金鑰。該主體 (使用者或角色) 必須具有 Amazon 金鑰 Keyspaces 所需之客戶受管金鑰的許可。

至少，Amazon 金 Keyspaces 在客戶受管金鑰上需要具備下列許可：

- kms:Encrypt
- kms:Decrypt
- 公里：ReEncrypt* （對於公里：ReEncryptFrom 和公里：ReEncryptTo ）
- 公里：GenerateDataKey* （對於公里：GenerateDataKey和公里：GenerateDataKeyWithoutPlaintext ）

- 公里：DescribeKey
- 公里：CreateGrant

範例金鑰政策

例如，以下範例金鑰政策只會提供必要許可。政策具有下列效果：

- 允許 Amazon Keyspaces space 在加密操作中使用客戶受管金鑰並建立授予，但只有在其代表具備使用 Amazon Keyspace 許可的帳戶中的委託人時才能進行。如果政策陳述式中指定的委託人沒有使用 Amazon Key space 的許可，Keyspaces 叫便會失敗，即使呼叫是來自 Amazon Keyspace 服務也一樣。
- kms:ViaService 條件金鑰只會在請求來自 Amazon KKeyspaces space 時，才會代表政策陳述式中列出的委託人時才會允許可。這些委託人無法直接呼叫這些操作。請注意，`kms:ViaService` 值 (`cassandra.*.amazonaws.com`) 在區域位置中有星號 (*)。Amazon Keyspaces 需要許可獨立於任何特定的權限AWS 區域。
- 給予客戶受管金鑰管理員 (可取得`db-team`角色的使用者) 對客戶受管金鑰的唯讀存取權以及撤銷授予的許可，包括 Amazon Key角需要用來保護資料表的授予。
- 提供 Amazon 金 Keyspaces 對客戶受管金鑰的唯讀存取權。在這種情況下，Amazon Keyspaces 可以直接調用這些操作。不必代表帳戶委託人。

在使用範例金鑰政策前，請將範例委託人替換成您 AWS 帳戶 中的實際委託人。

```
{
  "Id": "key-policy-cassandra",
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid" : "Allow access through Amazon Keyspaces for all principals in the account
 that are authorized to use Amazon Keyspaces",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::111122223333:user/db-lead"},
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey",
        "kms:CreateGrant"
      ],
```

```
      "Resource": "*",
      "Condition": {
         "StringLike": {
           "kms:ViaService" : "cassandra.*.amazonaws.com"
         }
      }
    },
    {
      "Sid":  "Allow administrators to view the customer managed key and revoke
 grants",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/db-team"
       },
      "Action": [
        "kms:Describe*",
        "kms:Get*",
        "kms:List*",
        "kms:RevokeGrant"
      ],
      "Resource": "*"
    }
  ]
}
```

使用授權 Amazon Keyspaces

除了金鑰政策外，Amazon Keyspaces 使用授予在客戶受管金鑰上設定許可。若要檢視您帳戶中客戶管理金鑰的授權，請使用ListGrants作業。Amazon Keyspaces 不需要授予或任何其他許可，就能使用AWS 擁有的金鑰來保護您的資料表。

Amazon Keyspaces space 在執行背景系統維護和持續資料保護任務時使用授予許可。它還使用授與來產生資料表金鑰。

每個授與都專屬於一個資料表。如果帳戶包含使用相同客戶受管金鑰加密的多個資料表，則每個資料表有每一種類型的授予。授權受 Amazon 密 Keyspaces 加密內容的限制，其中包括表名稱和AWS 帳戶 ID。如果不再需要授權，則授予包括退休授權的權限。

若要建立授予，Amazon Keyspaces 必須具有代表建立加密資料表使用者呼叫`CreateGrant`的許可。

金鑰政策也會允許帳戶撤銷客戶受管金鑰的授予。不過，如果您撤銷作用中加密資料表上的授予，Keyspaces space 將無法保護和維護資料表。

步驟 3：為新資料表指定客戶受管金鑰

請依照下列步驟，使用 Amazon Keyspaces space 主控台或 CQL 在新資料表中指定客戶受管金鑰。

使用客戶受管金鑰建立加密資料表 (主控台)

1. 登入AWS Management Console，並在 https://console.aws.amazon.com/keyspaces/home 開啟 Amazon 索 Keyspaces 主控台。
2. 在導覽窗格中，選擇 Tables (資料表)，然後選擇 Create table (建立資料表)。
3. 在「表格詳細資訊」段落的「建立表格」頁面中，選取索引鍵空間並輸入新表格的名稱。
4. 在 [結構描述] 區段中，建立資料表的結構定義。
5. 在 [表格設定] 區段中，選擇 [自訂設定]。
6. 繼續前往 「加密設定」。

   在此步驟中，您會為資料表選取加密設定。

   在 [選擇一個] 下的 [靜態加密] 區段中AWS KMS key，選擇 [選擇其他 KMS 金鑰 (進階)] 選項，然後在搜尋欄位中選擇AWS KMS key或輸入 Amazon 資源名稱 (ARN)。

   > **Note**
   >
   > 如果您選取的金鑰無法存取或遺失必要權限，請參閱AWS Key Management Service開發人員指南中的「疑難排解金鑰存取」。

7. 選擇 Create (建立) 以建立加密資料表。

使用客戶受管金鑰建立新資料表以進行靜態加密 (CQL)

若要建立使用客戶受管金鑰進行靜態加密的新資料表，您可以利用下列範例所示的CREATE TABLE陳述式。請務必使用 ARN 取代金鑰 ARN，以取得授予 Amazon 金鑰 Keyspaces 許可的有效金鑰。

```
CREATE TABLE my_keyspace.my_table(id bigint, name text, place text STATIC, PRIMARY
 KEY(id, name)) WITH CUSTOM_PROPERTIES = {
        'encryption_specification':{
                'encryption_type': 'CUSTOMER_MANAGED_KMS_KEY',
                'kms_key_identifier':'arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111'
            }
    };
```

如果您收到`Invalid Request Exception`，則需要確認客戶受管金鑰是否有效，而且 Amazon 金 Keyspaces 具有必要的許可。若要確認金鑰已正確設定，請參閱AWS Key Management Service開發人員指南中的金鑰存取疑難排解。

步驟 4：更新現有資料表的加密金鑰

您也隨時可以使用 Amazon KMS 主控台或 CQL，變更AWS 擁有的金鑰和客戶受管金鑰之間的現有資料表加密金鑰。

使用新的客戶管理金鑰 (主控台) 更新現有資料表

1. 登入AWS Management Console，並在 https://console.aws.amazon.com/keyspaces/home 開啟 Amazon 索 Keyspaces 主控台。
2. 在導覽窗格中，選擇 Tables (資料表)。
3. 選擇您要更新的資料表，然後選擇 [其他設定] 索引標籤。
4. 在「靜態加密」區段中，選擇「管理加密」以編輯表格的加密設定。

   在 [選擇一個] 下方AWS KMS key，選擇 [選擇不同的 KMS 金鑰 (進階)] 選項，然後在搜尋欄位中選擇AWS KMS key或輸入 Amazon 資源名稱 (ARN)。

   > **Note**
   >
   > 如果您選取的金鑰無效，請參閱AWS Key Management Service開發人員指南中的金鑰存取疑難排解。

   或者，您也可以選擇使AWS 擁有的金鑰用客戶受管金鑰加密的資料表。
5. 選擇儲存變更以儲存對表格的變更。

更新用於現有資料表的加密金鑰

若要變更現有資料表的加密金鑰，您可以使用`ALTER TABLE`陳述式來指定用於靜態加密的客戶管理金鑰。請務必使用 ARN 取代金鑰 ARN，以取得授予 Amazon 金鑰 Keyspaces 許可的有效金鑰。

```
ALTER TABLE my_keyspace.my_table WITH CUSTOM_PROPERTIES = {
              'encryption_specification':{
                     'encryption_type': 'CUSTOMER_MANAGED_KMS_KEY',
                     'kms_key_identifier':'arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111'
```

```
                    }
        };
```

如果您收到`Invalid Request Exception`，則需要確認客戶受管金鑰是否有效，而且 Amazon 金 Keyspaces 具有必要的許可。若要確認金鑰已正確設定，請參閱AWS Key Management Service開發人員指南中的金鑰存取疑難排解。

若要使用將加密金鑰變更回預設的靜態加密選項AWS 擁有的金鑰，您可以使用下列範例所示的`ALTER TABLE`陳述式。

```
ALTER TABLE my_keyspace.my_table WITH CUSTOM_PROPERTIES = {
                'encryption_specification':{
                      'encryption_type' : 'AWS_OWNED_KMS_KEY'
                   }
        };
```

步驟 5：在日誌中使用 Amazon Keyspaces 加密內容

加密內容是一組金鑰/值對，其中包含任意非私密資料。在加密資料的請求中包含加密內容時，AWS KMS 會以密碼編譯方式將加密內容繫結至加密的資料。若要解密資料，您必須傳遞相同的加密內容。

Amazon Keyspaces 在所有密AWS KMS編譯操作中使用相同的加密內容。如果您使用客戶受管金鑰來保護您的 Amazon Keyspaces 資料表，您可以使用加密內容來識別客戶受管金鑰在稽核記錄和日誌中的使用情況。它也會以純文字顯示在日誌中，例如在AWS CloudTrail和 Amazon 日誌的 CloudWatch 日誌中。

在其請求中AWS KMS，Amazon Key space 會使用具有三個鍵值對的加密內容。

```
"encryptionContextSubset": {
    "aws:cassandra:keyspaceName": "my_keyspace",
    "aws:cassandra:tableName": "mytable"
    "aws:cassandra:subscriberId": "111122223333"
}
```

- 鍵空間 — 第一個鍵值對會識別包含 Amazon Keyspace 正在加密資料表的索引鍵空間。金鑰為`aws:cassandra:keyspaceName`。值是索引鍵空間的名稱。

  ```
  "aws:cassandra:keyspaceName": "<keyspace-name>"
  ```

  例如：

```
"aws:cassandra:keyspaceName": "my_keyspace"
```

- 資料表 — 第二個鍵值對會識別 Amazon Keyspaces 正在加密的資料表。金鑰為 `aws:cassandra:tableName`。值是資料表的名稱。

  ```
  "aws:cassandra:tableName": "<table-name>"
  ```

  例如：

  ```
  "aws:cassandra:tableName": "my_table"
  ```

- 帳戶 — 第三個鍵值對會識別AWS 帳戶。金鑰為 `aws:cassandra:subscriberId`。值是帳戶 ID。

  ```
  "aws:cassandra:subscriberId": "<account-id>"
  ```

  例如：

  ```
  "aws:cassandra:subscriberId": "111122223333"
  ```

步驟 6：使用設定監視AWS CloudTrail

如果您使用客戶受管金鑰來保護您的 Amazon Keyspaces space 資料表，您可以使用AWS CloudTrail 日誌來追蹤 Amazon Keyspaces space 代表您傳送到AWS KMS的請求。

`GenerateDataKey`、`DescribeKey`、`Decrypt`、和`CreateGrant`請求將在本節中討論。此外，當您刪除資料表時，Amazon Keyspaces 會使用RetireGrant作業移除授權。

GenerateDataKey

Amazon 金 Keyspaces 會建立唯一的資料表金鑰，用來加密靜態資料表金鑰。它會傳送GenerateDataKey要求AWS KMS，以指定資料表的 KMS 金鑰。

記錄 `GenerateDataKey` 操作的事件類似於以下範例事件。使用者是 Amazon 索 Keyspaces 服務帳戶。參數包括客戶受管金鑰的 Amazon Resource Name (ARN)、需要 256 位元金鑰的金鑰規範，以及識別金鑰空間、資料表和的加密內容AWS 帳戶。

```
{
```

```
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2021-04-16T04:56:05Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keySpec": "AES_256",
        "encryptionContext": {
            "aws:cassandra:keyspaceName": "my_keyspace",
            "aws:cassandra:tableName": "my_table",
            "aws:cassandra:subscriberId": "123SAMPLE012"
        },
        "keyId": "arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111"
    },
    "responseElements": null,
    "requestID": "5e8e9cb5-9194-4334-aacc-9dd7d50fe246",
    "eventID": "49fccab9-2448-4b97-a89d-7d5c39318d6f",
    "readOnly": true,
    "resources": [
        {
            "accountId": "123SAMPLE012",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123SAMPLE012",
    "sharedEventID": "84fbaaf0-9641-4e32-9147-57d2cb08792e"
}
```

DescribeKey

Amazon 金 Keyspaces 會使用DescribeKey操作來判斷您選取的 KMS 金鑰是否存在於帳戶和區域內。

記錄 DescribeKey 操作的事件類似於以下範例事件。使用者是 Amazon 索 Keyspaces 服務帳戶。參數包括客戶受管金鑰的 ARN 以及需要 256 位元金鑰的金鑰規範。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAZ3FNIIVIZZ6H7CFQG",
        "arn": "arn:aws:iam::123SAMPLE012:user/admin",
        "accountId": "123SAMPLE012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "admin",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-04-16T04:55:42Z"
            }
        },
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2021-04-16T04:55:58Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111"
    },
    "responseElements": null,
    "requestID": "c25a8105-050b-4f52-8358-6e872fb03a6c",
    "eventID": "0d96420e-707e-41b9-9118-56585a669658",
    "readOnly": true,
    "resources": [
        {
            "accountId": "123SAMPLE012",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111"
        }
```

```
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123SAMPLE012"
}
```

解密

當您存取 Amazon Keyspaces 料表時，Amazon Keyspaces space 需要解密資料表金鑰，以便解密階層下方的金鑰。然後解密資料表中的資料。為了解密資料表金鑰，Amazon 金 Keyspaces 會傳送解密請求至AWS KMS，指定資料表的 KMS 金鑰。

記錄 Decrypt 操作的事件類似於以下範例事件。使用者是您 AWS 帳戶 中存取資料表的委託人。參數包括加密的資料表金鑰 (作為加密文字 Blob) 以及識別資料表和的加密內容AWS 帳戶。 AWS KMS從加密文字衍生客戶管理金鑰的 ID。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2021-04-16T05:29:44Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "encryptionContext": {
            "aws:cassandra:keyspaceName": "my_keyspace",
            "aws:cassandra:tableName": "my_table",
            "aws:cassandra:subscriberId": "123SAMPLE012"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "50e80373-83c9-4034-8226-5439e1c9b259",
    "eventID": "8db9788f-04a5-4ae2-90c9-15c79c411b6b",
    "readOnly": true,
    "resources": [
        {
```

```
            "accountId": "123SAMPLE012",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123SAMPLE012",
    "sharedEventID": "7ed99e2d-910a-4708-a4e3-0180d8dbb68e"
}
```

CreateGrant

當您使用客戶受管金鑰來保護您的 Amazon Key角資料表時，Amazon Keyspaces 會使用授予來允許服務執行持續的資料保護以及維護和耐用性任務。上不需要這些贈款AWS 擁有的金鑰。

Amazon Keyspaces 建立的授予專屬於特定資料表。CreateGrant請求中的主參與者是建立表格的使用者。

記錄 `CreateGrant` 操作的事件類似於以下範例事件。參數包括資料表客戶受管金鑰的 ARN、受受受受受管金鑰和淘汰委託人 (Amazon Keyspaces 服務)，以及授予涵蓋的操作。它也包含要求所有加密作業都使用指定加密內容的條件約束。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAZ3FNIIVIZZ6H7CFQG",
        "arn": "arn:aws:iam::arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111:user/admin",
        "accountId": "arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "admin",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-04-16T04:55:42Z"
            }
```

```
        },
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2021-04-16T05:11:10Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "a7d328af-215e-4661-9a69-88c858909f20",
        "operations": [
            "DescribeKey",
            "GenerateDataKey",
            "Decrypt",
            "Encrypt",
            "ReEncryptFrom",
            "ReEncryptTo",
            "RetireGrant"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:cassandra:keyspaceName": "my_keyspace",
                "aws:cassandra:tableName": "my_table",
                "aws:cassandra:subscriberId": "123SAMPLE012"
            }
        },
        "retiringPrincipal": "cassandratest.us-east-1.amazonaws.com",
        "granteePrincipal": "cassandratest.us-east-1.amazonaws.com"
    },
    "responseElements": {
        "grantId":
 "18e4235f1b07f289762a31a1886cb5efd225f069280d4f76cd83b9b9b5501013"
    },
    "requestID": "b379a767-1f9b-48c3-b731-fb23e865e7f7",
    "eventID": "29ee1fd4-28f2-416f-a419-551910d20291",
    "readOnly": false,
    "resources": [
        {
            "accountId": "123SAMPLE012",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111"
        }
```

```
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123SAMPLE012"
}
```

Amazon Keyspaces 傳輸中加密傳輸中加密

Amazon Keyspaces 只接受使用 Tran輸層安全性 (TLS) 的安全連線。傳輸中加密為資料提供另一層保護，為資料提供另一層保護，為資料提供另一層 Keyspaces 護。組織原則、產業或政府法規以及合規性要求通常需要在傳輸過程中使用加密，以提高應用程式透過網路傳輸資料時的資料安全性。

若要了解如何使用cqlsh TLS 加密到 Amazon Keyspaces 的連線，請參閱the section called “如何手動設定 TLS 的cqlsh連線”。若要了解如何使用 TLS 加密與用戶端驅動程式，請參閱the section called “使用卡桑德拉客戶端驅動程序”。

亞馬遜密鑰空間中的網絡間流量隱私

本主題說明亞馬遜密鑰空間（適用於 Apache Cassandra）如何保護從現場部署應用程序到亞馬遜密鑰空間以及亞馬遜密鑰空間和其他資源之間的連接。AWS AWS 區域

服務和內部部署用戶端與應用程式之間的流量。

在您的私有網路和 AWS 之間，您有兩個連線選項：

- AWS Site-to-Site VPN 連線。如需詳細資訊，請參閱《AWS Site-to-Site VPN 使用者指南》中的什麼是 AWS Site-to-Site VPN？。
- AWS Direct Connect 連線。如需詳細資訊，請參閱《AWS Direct Connect 使用者指南》中的什麼是 AWS Direct Connect？。

作為一種託管服務，亞馬遜密鑰空間（對於 Apache 卡桑德拉）受到AWS全球網絡安全的保護。如需有關 AWS 安全服務以及 AWS 如何保護基礎設施的詳細資訊，請參閱 AWS 雲端安全。若要使用基礎設施安全性的最佳實務來設計您的 AWS 環境，請參閱安全性支柱 AWS 架構良好的框架中的基礎設施保護。

您可以使用AWS已發佈的 API 呼叫透過網路存取 Amazon 金鑰空間。用戶端必須支援下列項目：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。

- 具備完美轉送私密 (PFS) 的密碼套件，例如 DHE (Ephemeral Diffie-Hellman) 或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。

此外，請求必須使用存取索引鍵 ID 和與 IAM 主體相關聯的私密存取索引鍵來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

Amazon 金鑰空間支援兩種驗證用戶端請求的方法。第一種方法使用服務特定的登入資料，這些登入資料是針對特定 IAM 使用者產生的密碼型認證。您可以使用 IAM 主控台、或 AWS API 建立和管理密碼。AWS CLI如需詳細資訊，請參閱將 IAM 與 Amazon 金鑰空間搭配使用。

第二種方法使用用於卡桑德拉的開源 DataStax Java 驅動程序的身份驗證插件。此外掛程式可讓 IAM 使用者、角色和聯合身分，使用AWS簽名版本 4 程序 (Sigv4) 將身份驗證資訊新增至 Amazon 金鑰空間 (適用於 Apache Cassandra) API 請求。如需詳細資訊，請參閱the section called “身分 AWS 驗證的 IAM 登入資”。

相同區域中 AWS 資源間的流量

介面 VPC 端點可讓您在 Amazon VPC 中執行的虛擬私有雲 (VPC) 與 Amazon 金鑰空間之間進行私有通訊。介面 VPC 端點由支援AWS PrivateLink，這是一項AWS服務，可在 VPC 和AWS服務之間進行私有通訊。 AWS PrivateLink透過在 VPC 中使用具有私有 IP 的彈性網路界面來啟用此功能，以便網路流量不會離開 Amazon 網路。介面 VPC 端點不需要網際網路閘道、NAT 裝置、VPN 連接或是 AWS Direct Connect 連線。如需詳細資訊，請參閱 Amazon 虛擬私有雲端和介面虛擬私有雲端節點 (AWS PrivateLink)。如需取得範例政策，請參閱 the section called “使用 Amazon Keyspaces 的界面 VPC 端點”。

AWS Identity and Access Management 對於 Amazon Keyspaces

AWS Identity and Access Management (IAM) 可協助系統管理員安全地控制 AWS 資源存取權。 AWS 服務 IAM 管理員控制哪些人可以通過身份驗證 (登入) 和授權 (具有許可) 來使用 Amazon Keyspaces 資源。IAM 是您可以使用的 AWS 服務 ，無需額外付費。

主題

- 物件
- 使用身分驗證
- 使用政策管理存取權
- Amazon Keyspaces 如何與 IAM 一起工作

- Amazon Keyspaces 基於身份的政策示例
- AWS亞馬遜 Keyspaces 的受管政策
- 疑難排解 Amazon Keyspaces 身分和存取
- 針對 Amazon Keyspaces 使用服務連結角色

物件

根據您在 Amazon Keyspaces 中執行的工作，使用方式 AWS Identity and Access Management (IAM) 會有所不同。

服務使用者 — 如果您使用 Amazon Keyspaces 服務執行工作，則管理員會為您提供所需的登入資料和許可。當您使用更多 Amazon Keyspaces 功能來完成工作時，您可能需要額外的許可。了解存取許可的管理方式可協助您向管理員請求正確的許可。如果您無法存取 Amazon Keyspaces 中的功能，請參閱疑難排解 Amazon Keyspaces 身分和存取。

服務管理員 — 如果您負責公司的 Amazon Keyspaces 資源，您可能可以完全訪問 Amazon Keyspaces。您的任務是確定服務使用者應存取哪些 Amazon Keyspaces 功能和資源。接著，您必須將請求提交給您的 IAM 管理員，來變更您服務使用者的許可。檢閱此頁面上的資訊，了解 IAM 的基本概念。若要進一步了解貴公司如何將 IAM 與 Amazon Keyspaces 搭配使用，請參閱Amazon Keyspaces 如何與 IAM 一起工作。

IAM 管理員 — 如果您是 IAM 管理員，您可能想要了解如何撰寫政策來管理 Amazon Keyspaces 存取權的詳細資訊。若要檢視可在 IAM 中使用的 Amazon Keyspaces 身分型政策範例，請參閱。Amazon Keyspaces 基於身份的政策示例

使用身分驗證

驗證是您 AWS 使用身分認證登入的方式。您必須以 IAM 使用者身分或假設 IAM 角色進行驗證 (登入 AWS)。 AWS 帳戶根使用者

您可以使用透過 AWS 身分識別來源提供的認證，以聯合身分識別身分登入。 AWS IAM Identity Center (IAM 身分中心) 使用者、貴公司的單一登入身分驗證，以及您的 Google 或 Facebook 登入資料都是聯合身分識別的範例。您以聯合身分登入時，您的管理員先前已設定使用 IAM 角色的聯合身分。當您使 AWS 用同盟存取時，您會間接擔任角色。

根據您的使用者類型，您可以登入 AWS Management Console 或 AWS 存取入口網站。如需有關登入的詳細資訊 AWS，請參閱《AWS 登入 使用指南》 AWS 帳戶中的如何登入您的。

如果您 AWS 以程式設計方式存取，請 AWS 提供軟體開發套件 (SDK) 和命令列介面 (CLI)，以使用您的認證以加密方式簽署您的要求。如果您不使用 AWS 工具，則必須自行簽署要求。如需使用建議的方法自行簽署請求的詳細資訊，請參閱 IAM 使用者指南中的簽署 AWS API 請求。

無論您使用何種身分驗證方法，您可能都需要提供額外的安全性資訊。例如， AWS 建議您使用多重要素驗證 (MFA) 來增加帳戶的安全性。如需更多資訊，請參閱 AWS IAM Identity Center 使用者指南中的多重要素驗證和 IAM 使用者指南中的在 AWS中使用多重要素驗證 (MFA)。

AWS 帳戶 根使用者

當您建立時 AWS 帳戶，您會從一個登入身分開始，該身分可完整存取該帳戶中的所有資源 AWS 服務和資源。此身分稱為 AWS 帳戶 root 使用者，可透過使用您用來建立帳戶的電子郵件地址和密碼登入來存取。強烈建議您不要以根使用者處理日常任務。保護您的根使用者憑證，並將其用來執行只能由根使用者執行的任務。如需這些任務的完整清單，了解需以根使用者登入的任務，請參閱《IAM 使用者指南》中的需要根使用者憑證的任務。

IAM 使用者和群組

IAM 使用者是您內部的身分，具 AWS 帳戶 有單一人員或應用程式的特定許可。建議您盡可能依賴暫時憑證，而不是擁有建立長期憑證 (例如密碼和存取金鑰) 的 IAM 使用者。但是如果特定使用案例需要擁有長期憑證的 IAM 使用者，建議您輪換存取金鑰。如需更多資訊，請參閱 IAM 使用者指南中的為需要長期憑證的使用案例定期輪換存取金鑰。

IAM 群組是一種指定 IAM 使用者集合的身分。您無法以群組身分簽署。您可以使用群組來一次為多名使用者指定許可。群組可讓管理大量使用者許可的程序變得更為容易。例如，您可以擁有一個名為 IAMAdmins 的群組，並給予該群組管理 IAM 資源的許可。

使用者與角色不同。使用者只會與單一人員或應用程式建立關聯，但角色的目的是在由任何需要它的人員取得。使用者擁有永久的長期憑證，但角色僅提供暫時憑證。如需進一步了解，請參閱IAM 使用者指南中的建立 IAM 使用者 (而非角色) 的時機。

IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的身分。它類似 IAM 使用者，但不與特定的人員相關聯。您可以切換角色，在中暫時擔任 IAM 角色。 AWS Management Console 您可以透過呼叫 AWS CLI 或 AWS API 作業或使用自訂 URL 來擔任角色。如需使用角色的方法詳細資訊，請參閱 IAM 使用者指南中的使用 IAM 角色。

使用暫時憑證的 IAM 角色在下列情況中非常有用：

- 聯合身分使用者存取 — 如需向聯合身分指派許可，請建立角色，並為角色定義許可。當聯合身分進行身分驗證時，該身分會與角色建立關聯，並獲授予由角色定義的許可。如需有關聯合角色的相關資訊，請參閱 IAM 使用者指南中的為第三方身分提供者建立角色。如果您使用 IAM Identity Center，則需要設定許可集。為控制身分驗證後可以存取的內容，IAM Identity Center 將許可集與 IAM 中的角色相關聯。如需有關許可集的資訊，請參閱 AWS IAM Identity Center 使用者指南中的許可集。
- 暫時 IAM 使用者許可 – IAM 使用者或角色可以擔任 IAM 角色來暫時針對特定任務採用不同的許可。
- 跨帳戶存取權：您可以使用 IAM 角色，允許不同帳戶中的某人 (信任的主體) 存取您帳戶的資源。角色是授予跨帳戶存取權的主要方式。但是，對於某些策略 AWS 服務，您可以將策略直接附加到資源（而不是使用角色作為代理）。若要了解跨帳戶存取角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。
- 跨服務訪問 — 有些 AWS 服務 使用其他 AWS 服務功能。例如，當您在服務中進行呼叫時，該服務通常會在 Amazon EC2 中執行應用程式或將物件儲存在 Amazon Simple Storage Service (Amazon S3) 中。服務可能會使用呼叫主體的許可、使用服務角色或使用服務連結角色來執行此作業。
  - 轉寄存取工作階段 (FAS) — 當您使用 IAM 使用者或角色在中執行動作時 AWS，您會被視為主體。使用某些服務時，您可能會執行某個動作，進而在不同服務中啟動另一個動作。FAS 會使用主體呼叫的權限 AWS 服務，並結合要求 AWS 服務 向下游服務發出要求。只有當服務收到需要與其 AWS 服務 他資源互動才能完成的請求時，才會發出 FAS 請求。在此情況下，您必須具有執行這兩個動作的許可。如需提出 FAS 請求時的政策詳細資訊，請參閱《轉發存取工作階段》。
  - 服務角色 – 服務角色是服務擔任的 IAM 角色，可代表您執行動作。IAM 管理員可以從 IAM 內建立、修改和刪除服務角色。如需詳細資訊，請參閱 IAM 使用者指南中的建立角色以委派許可給 AWS 服務服務。
  - 服務連結角色 — 服務連結角色是連結至. AWS 服務服務可以擔任代表您執行動作的角色。服務連結角色會顯示在您的中， AWS 帳戶 且屬於服務所有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。
- 在 Amazon EC2 上執行的應用程式 — 您可以使用 IAM 角色來管理在 EC2 執行個體上執行的應用程式以及發出 AWS CLI 或 AWS API 請求的臨時登入資料。這是在 EC2 執行個體內儲存存取金鑰的較好方式。若要將 AWS 角色指派給 EC2 執行個體並提供給其所有應用程式，請建立連接至執行個體的執行個體設定檔。執行個體設定檔包含該角色，並且可讓 EC2 執行個體上執行的程式取得暫時憑證。如需詳細資訊，請參閱 IAM 使用者指南中的利用 IAM 角色來授予許可給 Amazon EC2 執行個體上執行的應用程式。

如需了解是否要使用 IAM 角色或 IAM 使用者，請參閱 IAM 使用者指南中的建立 IAM 角色 (而非使用者) 的時機。

使用政策管理存取權

您可以透 AWS 過建立原則並將其附加至 AWS 身分識別或資源來控制中的存取。原則是一個物件 AWS ，當與身分識別或資源相關聯時，會定義其權限。 AWS 當主參與者 (使用者、root 使用者或角色工作階段) 提出要求時，評估這些原則。政策中的許可決定是否允許或拒絕請求。大多數原則會 AWS 以 JSON 文件的形式儲存在中。如需 JSON 政策文件結構和內容的詳細資訊，請參閱 IAM 使用者指南中的 JSON 政策概觀。

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

預設情況下，使用者和角色沒有許可。若要授予使用者對其所需資源執行動作的許可，IAM 管理員可以建立 IAM 政策。然後，管理員可以將 IAM 政策新增至角色，使用者便能擔任這些角色。

IAM 政策定義該動作的許可，無論您使用何種方法來執行操作。例如，假設您有一個允許 `iam:GetRole` 動作的政策。具有該原則的使用者可以從 AWS Management Console AWS CLI、或 AWS API 取得角色資訊。

身分型政策

身分型政策是可以附加到身分 (例如 IAM 使用者、使用者群組或角色) 的 JSON 許可政策文件。這些政策可控制身分在何種條件下能對哪些資源執行哪些動作。若要了解如何建立身分類型政策，請參閱 IAM 使用者指南中的建立 IAM 政策。

身分型政策可進一步分類成內嵌政策或受管政策。內嵌政策會直接內嵌到單一使用者、群組或角色。受管理的策略是獨立策略，您可以將其附加到您的 AWS 帳戶. 受管政策包括 AWS 受管政策和客戶管理的策略。如需了解如何在受管政策及內嵌政策間選擇，請參閱 IAM 使用者指南中的在受管政策和內嵌政策間選擇。

資源型政策

資源型政策是連接到資源的 JSON 政策文件。資源型政策的最常見範例是 IAM 角色信任政策和 Amazon S3 儲存貯體政策。在支援資源型政策的服務中，服務管理員可以使用它們來控制對特定資源的存取權限。對於附加政策的資源，政策會定義指定的主體可以對該資源執行的動作以及在何種條件下執行的動作。您必須在資源型政策中指定主體。主參與者可以包括帳戶、使用者、角色、同盟使用者或。 AWS 服務

資源型政策是位於該服務中的內嵌政策。您無法在以資源為基礎的政策中使用 IAM 的 AWS 受管政策。

存取控制清單 (ACL)

存取控制清單 (ACL) 可控制哪些主體 (帳戶成員、使用者或角色) 擁有存取某資源的許可。ACL 類似於資源型政策，但它們不使用 JSON 政策文件格式。

Amazon S3 和 Amazon VPC 是支援 ACL 的服務範例。 AWS WAF如需進一步了解 ACL，請參閱 Amazon Simple Storage Service 開發人員指南中的存取控制清單 (ACL) 概觀。

其他政策類型

AWS 支援其他較不常見的原則類型。這些政策類型可設定較常見政策類型授予您的最大許可。

- 許可界限 – 許可範圍是一種進階功能，可供您設定身分型政策能授予 IAM 實體 (IAM 使用者或角色) 的最大許可。您可以為實體設定許可界限。所產生的許可會是實體的身分型政策和其許可界限的交集。會在 `Principal` 欄位中指定使用者或角色的資源型政策則不會受到許可界限限制。所有這類政策中的明確拒絕都會覆寫該允許。如需許可界限的詳細資訊，請參閱 IAM 使用者指南中的 IAM 實體許可界限。
- 服務控制策略（SCP）— SCP 是 JSON 策略，用於指定中組織或組織單位（OU）的最大權限。AWS Organizations AWS Organizations 是一種用於分組和集中管理您企業擁 AWS 帳戶 有的多個服務。若您啟用組織中的所有功能，您可以將服務控制政策 (SCP) 套用到任何或所有帳戶。SCP 限制成員帳戶中實體的權限，包括每個 AWS 帳戶根使用者帳戶。如需 Organizations 和 SCP 的詳細資訊，請參閱 AWS Organizations 使用者指南中的 SCP 運作方式。
- 工作階段政策 – 工作階段政策是一種進階政策，您可以在透過編寫程式的方式建立角色或聯合使用者的暫時工作階段時，作為參數傳遞。所產生工作階段的許可會是使用者或角色的身分型政策和工作階段政策的交集。許可也可以來自資源型政策。所有這類政策中的明確拒絕都會覆寫該允許。如需詳細資訊，請參閱 IAM 使用者指南中的工作階段政策。

多種政策類型

將多種政策類型套用到請求時，其結果形成的許可會更為複雜、更加難以理解。要了解如何在涉及多個政策類型時 AWS 確定是否允許請求，請參閱《IAM 使用者指南》中的政策評估邏輯。

Amazon Keyspaces 如何與 IAM 一起工作

在您使用 IAM 管理 Amazon Keyspaces 的存取權限之前，您應該瞭解哪些 IAM 功能可用於 Amazon Keyspaces。若要深入瞭解 Amazon Keyspaces 和其他 AWS 服務如何與 IAM 搭配使用，請參閱 IAM 使用者指南中的可與 IAM 搭配使用的AWS 服務。

主題

- Amazon Keyspaces 基於身份的政策
- Amazon Keyspaces 資源型政策
- 基於 Amazon Keyspaces 標籤的授權
- Amazon Keyspaces IAM 角色

Amazon Keyspaces 基於身份的政策

使用 IAM 身分型政策，您可以指定允許或拒絕的動作和資源，以及在何種條件下允許或拒絕動作。Amazon 金 Keyspaces 支援特定動作和資源，以及條件金鑰。若要了解您在 JSON 政策中使用的所有元素，請參閱《IAM 使用者指南》中的 IAM JSON 政策元素參考。

若要查看 Amazon 密 Keyspaces 服務特定資源和動作，以及可用於 IAM 許可政策的條件內容金鑰，請參閱服務授權參考中的 Amazon 金 Keyspaces (適用於 Apache Cassandra) 的動作、資源和條件金鑰。

動作

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼条件下可以對什麼資源執行哪些動作。

JSON 政策的 `Action` 元素描述您可以用來允許或拒絕政策中存取的動作。原則動作通常與關聯的 AWS API 作業具有相同的名稱。有一些例外狀況，例如沒有相符的 API 操作的僅限許可動作。也有一些作業需要政策中的多個動作。這些額外的動作稱為相依動作。

政策會使用動作來授予執行相關聯動作的許可。

Amazon Keyspaces 中的政策動作會在動作前使用下列前置詞：`cassandra:`例如，若要授與某人使用 Amazon Keyspaces 間 CREATE CQL 陳述式建立 Amazon 金鑰空間 Keyspaces 間的權限，您可以在他們的政策中加入`cassandra:Create`動作。政策陳述式必須包含 `Action` 或 `NotAction` 元素。Amazon Keyspaces 會定義自己的一組動作，這些動作描述您可以使用此服務執行的任務。

若要在單一陳述式中指定多個動作，請用逗號分隔，如下所示：

```
"Action": [
      "cassandra:CREATE",
      "cassandra:MODIFY"
          ]
```

要查看 Amazon Keyspaces 操作列表，請參閱服務授權參考中由 Amazon Keyspaces 定義的操作（適用於 Apache Cassandra）。

資源

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Resource JSON 政策元素可指定要套用動作的物件。陳述式必須包含 Resource 或 NotResource 元素。最佳實務是使用其 Amazon Resource Name (ARN) 來指定資源。您可以針對支援特定資源類型的動作 (稱為資源層級許可) 來這麼做。

對於不支援資源層級許可的動作 (例如列出操作)，請使用萬用字元 (*) 來表示陳述式適用於所有資源。

```
"Resource": "*"
```

在 Amazon Keyspaces 間密鑰空間和表可以在 IAM 許可的Resource元素中使用。

Amazon 密鑰空間資源具有以下 ARN：

```
arn:${Partition}:cassandra:${Region}:${Account}:/keyspace/${KeyspaceName}/
```

Amazon Keyspaces 表資源具有以下 ARN：

```
arn:${Partition}:cassandra:${Region}:${Account}:/keyspace/${KeyspaceName}/table/
${tableName}
```

如需 ARN 格式的詳細資訊，請參閱 Amazon 資源名稱 (ARN) 和 AWS 服務命名空間。

例如，若要在陳述式中指定mykeyspace金鑰空間，請使用下列 ARN：

```
"Resource": "arn:aws:cassandra:us-east-1:123456789012:/keyspace/mykeyspace/"
```

若要指定屬於特定帳戶的所有金鑰空間，請使用萬用字元 (*)：

```
"Resource": "arn:aws:cassandra:us-east-1:123456789012:/keyspace/*"
```

某些 Amazon Keyspaces 動作 (例如用於建立資源的動作) 無法在特定資源上執行。在這些情況下，您必須使用萬用字元 (*)。

```
"Resource": "*"
```

若要使用標準驅動程式以程式設計方式連接到 Amazon Keyspaces，主體必須具有系統表格的 SELECT 存取權，因為大多數驅動程式會在連線時讀取系統金鑰空間/表格。例如，若要將SELECT權

限授與 IAM 使用mytable者mykeyspace，主體必須具有讀取權限，以mytable及system keyspace. 若要在單一陳述式中指定多項資源，請使用逗號分隔 ARN。

```
"Resource": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/
mytable",
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
```

要查看 Amazon Keyspaces 資源類型及其 ARN 的列表，請參閱服務授權參考中由 Amazon Keyspaces 定義的資源（適用於 Apache Cassandra）。要了解您可以使用哪些操作指定每個資源的 ARN，請參閱 Amazon Keyspaces 定義的操作（適用於 Apache C assandra）。

條件索引鍵

管理員可以使用 AWS JSON 政策來指定誰可以存取哪些內容。也就是說，哪個主體在什麼條件下可以對什麼資源執行哪些動作。

Condition 元素 (或 Condition 區塊) 可讓您指定使陳述式生效的條件。Condition 元素是選用項目。您可以建立使用條件運算子的條件運算式 (例如等於或小於)，來比對政策中的條件和請求中的值。

若您在陳述式中指定多個 Condition 元素，或是在單一 Condition 元素中指定多個索引鍵， AWS 會使用邏輯 AND 操作評估他們。如果您為單一條件索引鍵指定多個值，請使用邏輯OR運算來 AWS 評估條件。必須符合所有條件，才會授與陳述式的許可。

您也可以在指定條件時使用預留位置變數。例如，您可以只在使用者使用其 IAM 使用者名稱標記時，將存取資源的許可授予該 IAM 使用者。如需更多資訊，請參閱 IAM 使用者指南中的 IAM 政策元素：變數和標籤。

AWS 支援全域條件金鑰和服務特定條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

Amazon 金 Keyspaces 會定義自己的一組條件金鑰，並支援使用某些全域條件金鑰。若要查看所有 AWS 全域條件金鑰，請參閱《IAM 使用者指南》中的AWS 全域條件內容金鑰。

所有 Amazon 金 Keyspaces 動作都支援aws:ResourceTag/${TagKey}、和aws:TagKeys條件金鑰。aws:RequestTag/${TagKey}如需詳細資訊，請參閱 the section called “ 基於標籤的 Amazon Keyspaces 資源訪問”。

要查看 Amazon 密鑰 Keyspaces 條件密鑰的列表，請參閱服務授權參考中的 Amazon 密 Keyspaces（適用於 Apache Cassandra）的條件密鑰。若要了解您可以使用條件金鑰的動作和資源，請參閱 Amazon 金 Keyspaces 定義的動作 (適用於 Apache Cass andra)。

範例

若要檢視 Amazon Keyspaces 身分識別政策的範例，請參閱。Amazon Keyspaces 基於身份的政策示例

Amazon Keyspaces 資源型政策

Amazon Keyspaces 不支援以資源為基礎的政策。若要檢視詳細資源類型政策頁面的範例，請參閱 https://docs.aws.amazon.com/lambda/latest/dg/access-control-resource-based.html。

基於 Amazon Keyspaces 標籤的授權

您可以使用標籤來管理對 Amazon Keyspaces 資源的存取。若要根據標籤管理資源存取，您可以使用、或條件索引鍵在策略的條aws:TagKeys件元素中cassandra:ResourceTag/*key-name*提供標籤資訊。aws:RequestTag/*key-name*如需標記 Amazon Keyspaces 資源的詳細資訊，請參閱the section called “使用標籤”。

若要檢視身分型政策範例，用於根據該資源的標籤來限制資源的存取權，請參閱「 基於標籤的 Amazon Keyspaces 資源訪問」。

Amazon Keyspaces IAM 角色

IAM 角色是您 AWS 帳戶 內部具有特定許可的實體。

使用臨時登入資料搭配 Amazon Keyspaces

您可以使用暫時憑證來以聯合身分登入、擔任 IAM 角色，或是擔任跨帳戶角色。您可以透過呼叫AssumeRole或GetFederation權杖等 AWS STS API 作業來取得臨時安全登入資料。

Amazon Keyspaces 支援使用臨時登入資料搭配 Github 儲存庫提供的 AWS 簽名版本 4 (SIGv4) 身份驗證外掛程式，適用於下列語言：

- 爪哇:https://github.com/aws/aws-sigv4-auth-cassandra-java-driver-plugin.
- Node.js:https://github.com/aws/aws-sigv4-auth-cassandra-nodejs-driver-plugin.
- Python: https://github.com/aws/aws-sigv4-auth-cassandra-python-driver-plugin.
- 去：https://github.com/aws/aws-sigv4-auth-cassandra-gocql-driver-plugin。

如需實作驗證外掛程式以程式設計方式存取 Amazon Keyspaces 的範例和教學課程，請參閱the section called “使用卡桑德拉客戶端驅動程序”。

服務連結角色

服務連結角色可讓 AWS 服務存取其他服務中的資源，以代表您完成動作。服務連結角色會顯示在您的 IAM 帳戶中，並由該服務所擁有。IAM 管理員可以檢視，但不能編輯服務連結角色的許可。

如需有關建立或管理 Amazon Keyspaces 服務連結角色的詳細資訊，請參閱。the section called “使用服務連結角色”

服務角色

Amazon Keyspaces 不支援服務角色。

Amazon Keyspaces 基於身份的政策示例

依預設，IAM 使用者和角色沒有建立或修改 Amazon Keyspaces 資源的權限。他們也無法使用主控台、CQLSH 或 API 執行工作。 AWS CLI AWS IAM 管理員必須建立 IAM 政策，授予使用者和角色在指定資源上執行特定 API 作業的所需許可。管理員接著必須將這些政策連接至需要這些許可的 IAM 使用者或群組。

若要了解如何使用這些範例 JSON 政策文件建立 IAM 身分型政策，請參閱《IAM 使用者指南》中的在 JSON 標籤上建立政策。

主題

- 政策最佳實務
- 使用 Amazon Keyspaces 控制台
- 允許使用者檢視他們自己的許可
- 訪問 Amazon Keyspaces 表
- 基於標籤的 Amazon Keyspaces 資源訪問

政策最佳實務

以身分識別為基礎的政策決定某人是否可以在您的帳戶中建立、存取或刪除 Amazon Keyspaces 資源。這些動作可能會讓您的 AWS 帳戶產生費用。當您建立或編輯身分型政策時，請遵循下列準則及建議事項：

- 開始使用 AWS 受管原則並邁向最低權限權限 — 若要開始將權限授與使用者和工作負載，請使用可授與許多常見使用案例權限的AWS 受管理原則。它們可用在您的 AWS 帳戶. 建議您透過定義特定於您使用案例的 AWS 客戶管理政策，進一步降低使用權限。如需更多資訊，請參閱 IAM 使用者指南中的 AWS 受管政策或任務職能的AWS 受管政策。

- 套用最低權限許可 – 設定 IAM 政策的許可時，請僅授予執行任務所需的許可。為實現此目的，您可以定義在特定條件下可以對特定資源採取的動作，這也稱為最低權限許可。如需使用 IAM 套用許可的更多相關資訊，請參閱 IAM 使用者指南中的 IAM 中的政策和許可。
- 使用 IAM 政策中的條件進一步限制存取權 – 您可以將條件新增至政策，以限制動作和資源的存取。例如，您可以撰寫政策條件，指定必須使用 SSL 傳送所有請求。您也可以使用條件來授與對服務動作的存取權 (如透過特定) 使用這些動作 AWS 服務，例如 AWS CloudFormation。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。
- 使用 IAM Access Analyzer 驗證 IAM 政策，確保許可安全且可正常運作 – IAM Access Analyzer 驗證新政策和現有政策，確保這些政策遵從 IAM 政策語言 (JSON) 和 IAM 最佳實務。IAM Access Analyzer 提供 100 多項政策檢查及切實可行的建議，可協助您編寫安全且實用的政策。如需更多資訊，請參閱 IAM 使用者指南中的 IAM Access Analyzer 政策驗證。
- 需要多因素身份驗證 (MFA) — 如果您的案例需要 IAM 使用者或根使用者 AWS 帳戶，請開啟 MFA 以獲得額外的安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需更多資訊，請參閱 IAM 使用者指南中的設定 MFA 保護的 API 存取。

如需 IAM 中最佳實務的相關資訊，請參閱 IAM 使用者指南中的 IAM 安全最佳實務。

使用 Amazon Keyspaces 控制台

Amazon Keyspaces 不需要特定的許可即可存取 Amazon Keyspaces 主控台。您至少需要唯讀許可，才能列 Keyspaces 和檢視. AWS 帳戶如果您建立比最基本必要許可更嚴格的身分型政策，則對於具有該政策的實體 (IAM 使用者或角色) 而言，主控台就無法如預期運作。

兩個 AWS 受管政策可供實體使用 Amazon Keyspaces 主控台存取。

- AmazonKeyspacesReadOnly存取 _v2 — 此政策授予對 Amazon 金 Keyspaces 的唯讀存取權。
- AmazonKeyspacesFullAccess— 此政策授予使用完全存取所有功能之 Amazon 金 Keyspaces 的權限。

如需 Amazon Keyspaces 受管政策的詳細資訊，請參閱the section called “AWS 受管政策”。

允許使用者檢視他們自己的許可

此範例會示範如何建立政策，允許 IAM 使用者檢視附加到他們使用者身分的內嵌及受管政策。此原則包含在主控台上或以程式設計方式使用 AWS CLI 或 AWS API 完成此動作的權限。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

訪問 Amazon Keyspaces 表

以下是授與 Amazon 金 Keyspaces 系統表格唯讀 (SELECT) 存取權的範例政策。對於所有範例，請將 Amazon 資源名稱 (ARN) 中的區域和帳戶 ID 取代為您自己的 ID。

> **Note**
>
> 要與標準驅動程序連接，用戶必須至少具有對系統表的SELECT訪問權限，因為大多數驅動程序在連接時讀取系統密鑰空間/表。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "cassandra:Select"
         ],
         "Resource":[
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
         ]
      }
   ]
}
```

下列範例原則會將唯讀存取權新增至金鑰空間mytablemykeyspace中的使用者資料表。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "cassandra:Select"
         ],
         "Resource":[
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/
mytable",
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
         ]
      }
   ]
}
```

下列範例原則會將讀取/寫入存取權指派給使用者資料表，以及系統表格的讀取存取權。

> **Note**
>
> 系統表格永遠是唯讀的。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "cassandra:Select",
            "cassandra:Modify"
         ],
         "Resource":[
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/
mytable",
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
         ]
      }
   ]
}
```

下列範例原則可讓使用者在金鑰空間mykeyspace中建立資料表。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "cassandra:Create",
            "cassandra:Select"
         ],
         "Resource":[
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/*",
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
         ]
      }
   ]
}
```

基於標籤的 Amazon Keyspaces 資源訪問

您可以使用身分型政策中的條件，根據標籤控制對 Amazon Keyspaces 資源的存取。這些原則會控制帳戶中金鑰空間和表格的可見性。請注意，與通過 Cassandra 驅動程序和開發人員工具的 Cassandra 查詢語言（CQL）API 調用相比，使用 AWS SDK 發出請求時，系統表的基於標籤的權限行為不同。

- 要在使用基於標籤的訪問時使用 AWS SDK 發出List和Get資源請求，調用者需要具有對系統表的讀取權限。例如，需要Select動GetTable作權限才能透過作業從系統表格讀取資料。如果呼叫者只有以標籤為基礎的特定表格存取權，則需要額外存取系統表格的作業將會失敗。
- 為了與既定的 Cassandra 驅動程序行為兼容性，基於標籤的授權策略在使用 Cassandra 查詢語言（CQL）API 調用通過卡桑德拉驅動程序和開發人員工具在系統表上執行操作時不強制執行。

下列範例顯示如何建立政策，以便在資料表中Owner包含該使用者的使用者名稱值時，授與使用者檢視資料表的權限。在此範例中，您也會授與系統資料表的讀取存取權限。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ReadOnlyAccessTaggedTables",
         "Effect":"Allow",
         "Action":"cassandra:Select",
         "Resource":[
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/*",
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
         ],
         "Condition":{
            "StringEquals":{
               "aws:ResourceTag/Owner":"${aws:username}"
            }
         }
      }
   ]
}
```

您可以將此政策連接到您帳戶中的 IAM 使用者。如果名為的使用者richard-roe嘗試檢視 Amazon Keyspaces 資料表，則表格必須加上標籤Owner=richard-roe或owner=richard-roe。否則，他便會被拒絕存取。條件標籤鍵 Owner 符合 Owner 和 owner，因為條件索引鍵名稱不區分大小寫。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件。

如果資料表包含該使用者的使用者名稱值，則下列原則會授與使用者建立Owner含有標籤之資料表的權限。

```
{
    "Version": "2012-10-17",
    "Statement": [
```

```
       {
          "Sid": "CreateTagTableUser",
          "Effect": "Allow",
          "Action": [
              "cassandra:Create",
              "cassandra:TagResource"
          ],
          "Resource": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/
table/*",
          "Condition":{
             "StringEquals":{
                "aws:RequestTag/Owner":"${aws:username}"
            }
          }
       }
   ]
}
```

AWS亞馬遜 Keyspaces 的受管政策

AWS 受管政策是由 AWS 建立和管理的獨立政策。AWS 受管政策的設計在於為許多常見使用案例提供許可，如此您就可以開始將許可指派給使用者、群組和角色。

請記住，AWS 受管政策可能不會授與您特定使用案例的最低權限許可，因為它們可供所有 AWS 客戶使用。我們建議您定義使用案例專屬的客戶管理政策，以便進一步減少許可。

您無法更改 AWS 受管政策中定義的許可。如果 AWS 更新 AWS 受管政策中定義的許可，更新會影響政策連接的所有主體身分 (使用者、群組和角色)。在推出新的 AWS 服務 或有新的 API 操作可供現有服務使用時，AWS 很可能會更新 AWS 受管政策。

如需詳細資訊，請參閱《IAM 使用者指南》https://docs.aws.amazon.com/IAM/latest/UserGuide/access_policies_managed-vs-inline.html#aws-managed-policies中的 AWS 受管政策。

AWS受管理的策略： AmazonKeyspacesReadOnlyAccess_v2

您可將 AmazonKeyspacesReadOnlyAccess_v2 政策連接到 IAM 身分。

此政策授予對 Amazon 金 Keyspaces 的唯讀存取權，並在透過私有 VPC 端點連線時包含必要的許可。

許可詳細資訊

此政策包含以下許可。

- `Amazon Keyspaces`— 提供對 Amazon Keyspaces 的唯讀存取權。
- `Application Auto Scaling`— 允許主參與者從 Application Auto Scaling 檢視組態。這是必要的，使用者才能檢視附加至資料表的自動調度資源調整原則。
- `CloudWatch`— 允許主參與者檢視中 CloudWatch配置的測量結果資料和警示。這是必要的，因此使用者可以檢視已針對表格設定的可計費表格大小和 CloudWatch 警示。
- `AWS KMS`— 允許主參與者檢視中配置的AWS KMS關鍵字。這是必要的，因此使用者可以檢視他們在帳戶中建立和管理的AWS KMS金鑰，以確認指派給 Amazon Keyspaces 的金鑰是已啟用的對稱加密金鑰。
- `Amazon EC2`— 允許主體透過 VPC 端點連線至 Amazon Keyspaces，在 Amazon EC2 執行個體上查詢 VPC 以取得端點和網路界面資訊。需要對 Amazon EC2 執行個體進行唯讀存取，因此 Amazon Keyspaces 可以在用於連線負載平衡的`system.peers`表格中查詢並存放可用的介面 VPC 端點。

若要檢閱JSON格式的原則，請參閱 AmazonKeyspacesReadOnlyAccess_v2。

AWS受管理的策略： AmazonKeyspacesReadOnlyAccess

您可將 `AmazonKeyspacesReadOnlyAccess` 政策連接到 IAM 身分。

此政策授予 Amazon 金 Keyspaces 的唯讀存取權。

許可詳細資訊

此政策包含以下許可。

- `Amazon Keyspaces`— 提供對 Amazon Keyspaces 的唯讀存取權。

- `Application Auto Scaling`— 允許主參與者從 Application Auto Scaling 檢視組態。這是必要的，使用者才能檢視附加至資料表的自動調度資源調整原則。
- `CloudWatch`— 允許主參與者檢視中 CloudWatch配置的測量結果資料和警示。這是必要的，因此使用者可以檢視已針對表格設定的可計費表格大小和 CloudWatch 警示。
- `AWS KMS`— 允許主參與者檢視中配置的AWS KMS關鍵字。這是必要的，因此使用者可以檢視他們在帳戶中建立和管理的AWS KMS金鑰，以確認指派給 Amazon Keyspaces 的金鑰是已啟用的對稱加密金鑰。

若要檢閱JSON格式的策略，請參閱AmazonKeyspacesReadOnlyAccess。

AWS 受管政策： AmazonKeyspacesFullAccess

您可將 `AmazonKeyspacesFullAccess` 政策連接到 IAM 身分。

此政策授予管理許可，允許您的管理員不受限制地存取 Amazon 金 Keyspaces。

許可詳細資訊

此政策包含以下許可。

- `Amazon Keyspaces`— 允許主體存取任何 Amazon Keyspaces 資源並執行所有動作。
- `Application Auto Scaling`— 允許主體建立、檢視和刪除 Amazon Keyspaces 表格的自動擴展政策。這是必要的，以便管理員可以管理 Amazon Keyspaces 表的自動擴展政策。
- `CloudWatch`— 允許主體查看可計費的表格大小，以及建立、檢視和刪除 Amazon Keyspaces 自動擴展政策的 CloudWatch 警示。這是必要的，以便管理員可以檢視可計費表格大小並建立 CloudWatch 儀表板。
- `IAM`— 開啟下列功能時，允許 Amazon Keyspaces 使用 IAM 自動建立服務連結角色：
  - `Application Auto Scaling`— 當管理員為表格啟 Application Auto Scaling 時，Amazon Keyspaces 會建立服務連結角色，以代表您執行自動擴展動作。
  - `Amazon Keyspaces Multi-Region Replication`— 當系統管理員建立多區域金鑰空間時，系統會自動建立服務連結角色，以代表您執行資料複製到選取AWS 區域的角色。

  如需服務連結角色的詳細資訊，請參閱the section called “使用服務連結角色”。

- AWS KMS— 允許主參與者檢視中配置的AWS KMS關鍵字。這是必要的，以便使用者可以檢視他們在帳戶中建立和管理的AWS KMS金鑰，以確認指派給 Amazon 金 Keyspaces 的金鑰是已啟用的對稱加密金鑰。
- Amazon EC2— 允許主體透過 VPC 端點連線至 Amazon Keyspaces，在 Amazon EC2 執行個體上查詢 VPC 以取得端點和網路界面資訊。需要對 Amazon EC2 執行個體進行唯讀存取，因此 Amazon Keyspaces 可以在用於連線負載平衡的system.peers表格中查詢並存放可用的介面 VPC 端點。

若要檢閱JSON格式的策略，請參閱AmazonKeyspacesFullAccess。

亞馬遜 Keyspaces 更新到AWS受管政策

檢視有關 Amazon Keyspace AWS 受管政策更新的詳細資訊，因為此服務開始追蹤這些變更。如需有關此頁面變更的自動提醒，請訂閱 文件歷史紀錄 頁面的 RSS 摘要。

變更	描述	日期
AmazonKeyspacesFullAccess – 更新現有政策	Amazon Keyspaces 為透過界面 VPC 端點連線到 Amazon Keyspaces 的用戶端新增了新的唯讀許可，以存取 Amazon EC2 執行個體以查詢網路資訊。 Amazon Keyspaces 會將可用的介面 VPC 端點存放在system.peers 表格中，以進行連線負載平衡。如需詳細資訊，請參閱the section called “使用 介面 VPC 端點”。	2023 年 10 月 3 日
AmazonKeyspacesReadOnlyAccess_v2 — 新政策	Amazon Keyspaces 建立了新政策，為透過界面 VPC 端點連線到 Amazon Keyspaces 的用戶端新增唯讀許可，以存取	2023 年 9 月 12 日

變更	描述	日期
	Amazon EC2 執行個體以查詢網路資訊。 Amazon Keyspaces 會將可用的介面 VPC 端點存放在`system.peers` 表格中，以進行連線負載平衡。如需詳細資訊，請參閱the section called “使用 介面 VPC 端點”。	
AmazonKeyspacesFullAccess – 更新現有政策	Amazon Keyspaces 間增加了新的許可，以允許 Amazon Keyspaces 間在管理員建立多區域金鑰空間時建立服務連結角色。 Amazon Keyspaces 使用服務連結角色代表您執行資料複寫任務。如需詳細資訊，請參閱the section called “多區域複製”。	2023 年 6 月 5 日
AmazonKeyspacesReadOnlyAccess – 更新現有政策	亞馬遜 Keyspaces 添加了新的許可，允許用戶使用查看表的計費大小。 CloudWatch 亞馬遜 Keyspaces 與亞馬遜 CloudWatch 集成，允許您監控計費表大小。如需詳細資訊，請參閱the section called “Amazon Keyspaces 指標和維度”。	2022 年 7 月 7 日

變更	描述	日期
AmazonKeyspacesFullAccess – 更新現有政策	亞馬遜 Keyspaces 添加了新的許可，允許用戶使用查看表的計費大小。 CloudWatch 亞馬遜 Keyspaces 與亞馬遜 CloudWatch 集成，允許您監控計費表大小。如需詳細資訊，請參閱the section called “Amazon Keyspaces 指標和維度”。	2022 年 7 月 7 日
AmazonKeyspacesReadOnlyAccess – 更新現有政策	亞馬遜密 Keyspaces 添加了新的許可，允許用戶查看已為靜態亞馬遜密 Keyspaces 加密配置的密AWS KMS鑰。 Amazon 靜態密 Keyspaces 加密與AWS KMS整合，可保護和管理用於加密靜態資料的加密金鑰。若要檢視針對 Amazon 金 Keyspaces 設定的 AWS KMS金鑰，已新增唯讀許可。	2021 年 6 月 1 日
AmazonKeyspacesFullAccess – 更新現有政策	亞馬遜密 Keyspaces 添加了新的許可，允許用戶查看已為靜態亞馬遜密 Keyspaces 加密配置的密AWS KMS鑰。 Amazon 靜態密 Keyspaces 加密與AWS KMS整合，可保護和管理用於加密靜態資料的加密金鑰。若要檢視針對 Amazon 金 Keyspaces 設定的 AWS KMS金鑰，已新增唯讀許可。	2021 年 6 月 1 日

變更	描述	日期
亞馬遜 Keyspaces 開始跟踪更改	Amazon Keyspaces 開始追蹤其AWS受管政策的變更。	2021 年 6 月 1 日

疑難排解 Amazon Keyspaces 身分和存取

使用下列資訊可協助您診斷和修正使用 Amazon Keyspaces 和 IAM 時可能遇到的常見問題。

主題

- 我沒有授權在 Amazon Keyspaces 中執行操作
- 我修改了 IAM 使用者或角色，但變更未立即生效
- 我無法使用 Amazon Keyspaces point-in-time 恢復（PITR）恢復表
- 我沒有授權執行 iam：PassRole
- 我是管理員，希望允許其他人訪問 Amazon Keyspaces
- 我想允許我以外的人訪問我 AWS 帳戶 的 Amazon Keyspaces 資源

我沒有授權在 Amazon Keyspaces 中執行操作

如果 AWS Management Console 告訴您您沒有執行動作的授權，則您必須聯絡您的管理員以尋求協助。您的管理員是提供您使用者名稱和密碼的人員。

當 `mateojackson` IAM 使用者嘗試使用主控台來檢視資料表的詳細資料，但沒有資料#的`cassandra:Select`權限時，就會發生下列範例錯誤。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 cassandra:Select on resource: mytable
```

在此情況下，Mateo 會請求管理員更新他的政策，允許他使用 *mytable* 動作存取 `cassandra:Select` 資源。

我修改了 IAM 使用者或角色，但變更未立即生效

IAM 政策變更可能需要長達 10 分鐘的時間才會生效，現有已建立連線至 Amazon Keyspaces 的應用程式。當應用程式建立新連線時，IAM 政策變更會立即生效。如果您對現有 IAM 使用者或角色進

行了修改，但該使用者或角色尚未立即生效，請等待 10 分鐘，或中斷連線並重新連線至 Amazon Keyspaces。

我無法使用 Amazon Keyspaces point-in-time 恢復（PITR）恢復表

如果您嘗試使用 point-in-time 復原 (PITR) 還原 Amazon Keyspaces 表格，並且看到還原程序開始但未成功完成，則可能尚未設定還原程序所需的所有必要許可。您必須聯絡管理員尋求協助，並要求該人員更新您的政策，以允許您還原 Amazon Keyspace 中的表格。

除了使用者許可之外，Amazon Keyspaces space 可能還需要許可，才能代表您的主體在還原程序期間執行動作。如果資料表使用客戶管理的金鑰加密，或者您使用的是限制傳入流量的 IAM 政策，就會發生這種情況。例如，如果您在 IAM 政策中使用條件金鑰將來源流量限制到特定端點或 IP 範圍，則還原作業會失敗。若要允許 Amazon 密 Keyspaces 代表您的主體執行表格還原作業，您必須在 IAM 政策中新增`aws:ViaAWSService`全域條件金鑰。

如需還原資料表之權限的詳細資訊，請參閱the section called “還原權限”。

我沒有授權執行 iam：PassRole

如果您收到未獲授權執行`iam:PassRole`動作的錯誤訊息，則必須更新您的政策以允許您將角色傳遞給 Amazon Keyspaces。

有些 AWS 服務 允許您將現有角色傳遞給該服務，而不是建立新的服務角色或服務連結角色。如需執行此作業，您必須擁有將角色傳遞至該服務的許可。

當名為的 IAM 使用者`marymajor`嘗試使用主控台在 Amazon Keyspaces 中執行動作時，會發生下列範例錯誤。但是，動作請求服務具備服務角色授予的許可。Mary 沒有將角色傳遞至該服務的許可。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在這種情況下，Mary 的政策必須更新，允許她執行 `iam:PassRole` 動作。

如果您需要協助，請聯絡您的 AWS 系統管理員。您的管理員提供您的簽署憑證。

我是管理員，希望允許其他人訪問 Amazon Keyspaces

若要允許其他人存取 Amazon Keyspaces，您必須為需要存取的人員或應用程式建立 IAM 實體 (使用者或角色)。他們將使用該實體的憑證來存取 AWS。然後，您必須將政策附加到實體，以便在 Amazon Keyspaces 中授予他們正確許可。

若要立即開始使用，請參閱《IAM 使用者指南》中的建立您的第一個 IAM 委派使用者及群組。

我想允許我以外的人訪問我 AWS 帳戶 的 Amazon Keyspaces 資源

您可以建立一個角色，讓其他帳戶中的使用者或您組織外部的人員存取您的資源。您可以指定要允許哪些信任物件取得該角色。針對支援基於資源的政策或存取控制清單 (ACL) 的服務，您可以使用那些政策來授予人員存取您的資源的許可。

如需進一步了解，請參閱以下內容：

- 要了解 Amazon Keyspaces 是否支援這些功能，請參閱Amazon Keyspaces 如何與 IAM 一起工作。
- 若要了解如何提供對您所擁有資源 AWS 帳戶 的存取權，請參閱《IAM 使用者指南》中您擁有的另一 AWS 帳戶 個 IAM 使用者提供存取權限。
- 若要了解如何向第三方提供對資源的存取權 AWS 帳戶，請參閱 IAM 使用者指南中的提供第三方 AWS 帳戶 擁有的存取權。
- 如需了解如何透過聯合身分提供存取權，請參閱 IAM 使用者指南中的將存取權提供給在外部進行身分驗證的使用者 (聯合身分)。
- 若要了解跨帳戶存取使用角色和以資源為基礎的政策之間的差異，請參閱 IAM 使用者指南中的 IAM 中的跨帳戶資源存取。

針對 Amazon Keyspaces 使用服務連結角色

Amazon Keyspaces（對於阿帕奇卡桑德拉）使用AWS Identity and Access Management（IAM）服務鏈接的角色。服務連結角色是直接連結至 Amazon Keyspaces 的唯一 IAM 角色類型。服務連結角色由 Amazon Keyspaces space 預先定義，並包含服務代表您呼叫其他AWS服務所需的所有許可。

主題

- 使用 Amazon Keyspaces 應用程式的角色 auto 擴展
- 使用角色進行 Amazon Keyspaces 多區域複寫

使用 Amazon Keyspaces 應用程式的角色 auto 擴展

Amazon Keyspaces（對於阿帕奇卡桑德拉）使用AWS Identity and Access Management（IAM）服務鏈接的角色。服務連結角色是直接連結至 Amazon Keyspaces 的唯一 IAM 角色類型。服務連結角色由 Amazon Keyspaces space 預先定義，並包含服務代表您呼叫其他AWS服務所需的所有許可。

服務連結角色可讓您更輕鬆地設定 Amazon Keyspaces，因為您不必手動新增必要的許可。Amazon Keyspaces 會定義其服務連結角色的許可，除非另有定義，否則只有 Amazon Keyspaces 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除角色的相關資源，才能刪除服務連結角色。這樣可以保護您的 Amazon Keyspaces 資源，因為您無法意外移除存取資源的權限。

如需關於支援服務連結角色的其他服務資訊，請參閱《可搭配 IAM 運作的 AWS 服務》，尋找服務連結角色欄中顯示為是的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

Amazon Keyspaces 的服務連結角色許可

Amazon Keyspaces 使用名為AWSServiceRoleForApplicationAutoScaling_CassandraTable的服務連結角色，允許 Application Auto Scaling 代表您呼叫 Amazon Keyspaces 和 Amazon CloudWatch 。

服 AWSServiceRoleForApplicationAutoScaling_CassandraTable 務連結角色會信任下列服務擔任該角色：

- `cassandra.application-autoscaling.amazonaws.com`

角色許可政策允許應用程式自動擴展在指定的 Amazon Keyspaces 資源上完成下列動作：

- 動作：`arn:*:cassandra:*:*:/keyspace/system/table/*` 上的 `cassandra:Select`
- 動作：對資源 `arn:*:cassandra:*:*:/keyspace/system_schema/table/*` 執行 `cassandra:Select`
- 動作：對資源 `arn:*:cassandra:*:*:/keyspace/system_schema_mcs/table/*` 執行 `cassandra:Select`
- 動作：對資源 `arn:*:cassandra:*:*:"*"` 執行 `cassandra:Alter`

為 Amazon Keyspaces 創建服務鏈接角色

您不需要為 Amazon Keyspaces 自動擴展手動建立服務連結角色。當您使用AWS Management Console、CQL、或 AWS API 在表格上啟用 Amazon Keyspaces auto 動擴展時，應用程式自動擴展會為您建立服務連結角色。AWS CLI

若您刪除此服務連結角色然後需要再次建立，便可在帳戶中使用相同程序重新建立角色。當您為表格啟用 Amazon Keyspaces auto 擴展時，Application Auto Scaling 會再次為您建立服務連結角色。

> **Important**
>
> 此服務連結角色可以顯示在您的帳戶，如果您於其他服務中完成一項動作時，可以使用支援此角色的功能。若要進一步了解，請參閱在我的 AWS 帳戶 中顯示新角色。

若您刪除此服務連結角色然後需要再次建立，便可在帳戶中使用相同程序重新建立角色。當您為表格啟用 Amazon Keyspaces 自動應用程式擴展時，Application Auto Scaling 會再次為您建立服務連結角色。

編輯 Amazon Keyspaces 的服務連結角色

Amazon Keyspaces 不允許您編輯 AWSServiceRoleForApplicationAutoScaling_CassandraTable 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除 Amazon Keyspaces 的服務鏈接角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。不過，您必須先停用帳戶中所有資料表的自動調整功能，AWS 區域才能手動刪除服務連結角色。若要停用 Amazon Keyspaces 表格上的自動擴展功能，請參閱修改或停用 Amazon Keyspaces 自動擴展設定。

> **Note**
>
> 如果您嘗試修改資源時，Amazon Keyspaces 自動擴展正在使用該角色，則取消註冊可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

使用 IAM 手動刪除服務連結角色

使用 IAM 主控台或 AWS API 刪除 AWSServiceRoleForApplicationAutoScaling_CassandraTable 服務連結角色。AWS CLI如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

> **Note**
>
> 若要刪除 Amazon Keyspaces 自動擴展所使用的服務連結角色，您必須先停用帳戶中所有表格的自動擴展功能。

支援 Amazon Keyspaces 服務連結角色的區域

Amazon Keyspaces 支援在所有提供服務的區域中使用服務連結角色。如需詳細資訊，請參閱 Amazon Keyspaces 的服務端點。

使用角色進行 Amazon Keyspaces 多區域複寫

Amazon Keyspaces（對於阿帕奇卡桑德拉）使用AWS Identity and Access Management（IAM）服務鏈接的角色。服務連結角色是直接連結至 Amazon Keyspaces 的唯一 IAM 角色類型。服務連結角色由 Amazon Keyspaces space 預先定義，並包含服務代表您呼叫其他AWS服務所需的所有許可。

服務連結角色可讓您更輕鬆地設定 Amazon Keyspaces，因為您不必手動新增必要的許可。Amazon Keyspaces 會定義其服務連結角色的許可，除非另有定義，否則只有 Amazon Keyspaces 可以擔任其角色。定義的許可包括信任政策和許可政策，且該許可政策無法附加至其他 IAM 實體。

您必須先刪除角色的相關資源，才能刪除服務連結角色。這樣可以保護您的 Amazon Keyspaces 資源，因為您無法意外移除存取資源的權限。

如需關於支援服務連結角色的其他服務資訊，請參閱《可搭配 IAM 運作的 AWS 服務》，尋找服務連結角色欄中顯示為是的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

Amazon Keyspaces 的服務連結角色許可

Amazon Keyspaces 使用名為的服務連結角色，AWSServiceRoleForAmazonKeyspacesReplication允許 Amazon Keyspaces 代表您複寫對多區域表格的所有複本的寫入。

服 AWSServiceRoleForAmazonKeyspacesReplication 務連結角色會信任下列服務擔任該角色：

- `replication.cassandra.amazonaws.com`

名為的角色許可政策 KeyspacesReplicationServiceRolePolicy 允許 Amazon Keyspaces 完成以下動作：

- 動作：`cassandra:Select`
- 動作：`cassandra:SelectMultiRegionResource`
- 動作：`cassandra:Modify`
- 動作：`cassandra:ModifyMultiRegionResource`

儘管 Amazon Keyspaces 服務鏈接角色在政策中 AWSServiceRoleForAmazonKeyspacesReplication 提供了許可：「動作：」為指定的 Amazon 資源名稱（ARN）「arn：*」，但 Amazon Keyspaces 會提供您帳戶的 ARN。

您必須設定許可，以允許您的使用者、群組或角色建立、編輯或刪除服務連結角色。如需詳細資訊，請參閱 IAM 使用者指南中的服務連結角色許可。

為 Amazon Keyspaces 創建服務鏈接角色

您無法手動建立服務連結角色。當您在AWS Management Console、或 AWS API 中建立多區域金鑰空間時AWS CLI，Amazon Keyspaces 間會為您建立服務連結角色。

若您刪除此服務連結角色然後需要再次建立，便可在帳戶中使用相同程序重新建立角色。當您建立多區域金鑰空間時，Amazon Keyspaces 間會再次為您建立服務連結角色。

編輯 Amazon Keyspaces 的服務連結角色

Amazon Keyspaces 不允許您編輯 AWSServiceRoleForAmazonKeyspacesReplication 服務連結角色。因為有各種實體可能會參考服務連結角色，所以您無法在建立角色之後變更角色名稱。然而，您可使用 IAM 來編輯角色描述。如需詳細資訊，請參閱 IAM 使用者指南中的編輯服務連結角色。

刪除 Amazon Keyspaces 的服務鏈接角色

若您不再使用需要服務連結角色的功能或服務，我們建議您刪除該角色。如此一來，您就沒有未主動監控或維護的未使用實體。不過，您必須先刪除帳戶中的所有多區域金鑰空間，AWS 區域才能手動刪除服務連結角色。

清除服務連結角色

您必須先刪除角色使用的任何多區域金鑰空間和表格，才能使用 IAM 刪除服務連結角色。

> **Note**
>
> 如果您嘗試刪除資源時，Amazon Keyspaces 服務正在使用該角色，則刪除可能會失敗。若此情況發生，請等待數分鐘後並再次嘗試操作。

要刪除 Amazon Keyspaces 所使用的資源 AWSServiceRoleForAmazonKeyspacesReplication（控制台）

1. 登錄到AWS Management Console，然後打開 Amazon Keyspaces 控制台 https://console.aws.amazon.com/keyspaces/home。
2. 從左側面板中選擇 Keyspaces。
3. 從清單中選取所有多區域金鑰空間。
4. 選擇刪除確認刪除，然後選擇刪除密鑰空間。

您也可以使用下列任何一種方法，以程式設計方式刪除多區域金鑰空間。

- 該卡桑德拉查詢語言（CQL）語句。???
- CLI 的刪除金鑰空間作業。AWS
- Amazon Keyspaces API 的DeleteKeyspace操作。

手動刪除服務連結角色

使用 IAM 主控台、AWS CLI 或 AWS API 來刪除 AWSServiceRoleForAmazonKeyspacesReplication 服務連結角色。如需詳細資訊，請參閱《IAM 使用者指南》中的刪除服務連結角色。

支援 Amazon Keyspaces 服務連結角色的區域

Amazon Keyspaces 不支援在提供服務的每個區域中使用服務連結角色。您可以在下列區域中使用此 AWSServiceRoleForAmazonKeyspacesReplication 角色。

區域名稱	區域身分	Support Amazon Keyspaces
美國東部 (維吉尼亞北部)	us-east-1	是
美國東部 (俄亥俄)	us-east-2	是
美國西部 (加利佛尼亞北部)	us-west-1	是
美國西部 (奧勒岡)	us-west-2	是
亞太區域 (孟買)	ap-south-1	是
亞太區域 (大阪)	ap-northeast-3	是
亞太區域 (首爾)	ap-northeast-2	是
亞太區域 (新加坡)	ap-southeast-1	是
亞太區域 (雪梨)	ap-southeast-2	是
亞太區域 (東京)	ap-northeast-1	是
加拿大 (中部)	ca-central-1	是

區域名稱	區域身分	Support Amazon Keyspaces
歐洲 (法蘭克福)	eu-central-1	是
歐洲 (愛爾蘭)	eu-west-1	是
歐洲 (倫敦)	eu-west-2	是
歐洲 (巴黎)	eu-west-3	是
南美洲 (聖保羅)	sa-east-1	是
AWS GovCloud (美國東部)	us-gov-east-1	否
AWS GovCloud (美國西部)	us-gov-west-1	否

Amazon Keyspaces 的合規驗證（阿帕奇卡桑德拉）

第三方稽核員會評估 Amazon Keyspaces (適用於 Apache Cassandra) 的安全性和合規性，作為多個合規計劃的一部分。 AWS 其中包含：

- ISO/IEC 27001:2013, 27017:2015, 27018:2019, 和 IEC 9001:2015. 如需詳細資訊，請參閱 AWS ISO 和 CSA STAR 認證與服務。
- 系統和組織控制 (SOC)
- 支付卡產業 (PCI)
- 聯邦風險和授權管理計劃（FedRAMP）高
- 美國健康保險流通與責任法案 (HIPAA)

若要瞭解 AWS 服務 是否屬於特定規範遵循方案的範圍內，請參閱AWS 服務 遵循規範計劃方案中的，並選擇您感興趣的合規方案。如需一般資訊，請參閱AWS 規範計劃AWS。

您可以使用下載第三方稽核報告 AWS Artifact。如需詳細資訊，請參閱下載中的報告中的 AWS Artifact。

您在使用時的合規責任取決 AWS 服務 於資料的敏感性、公司的合規目標以及適用的法律和法規。AWS 提供下列資源以協助遵循法規：

- 安全性與合規性快速入門指南 — 這些部署指南討論架構考量，並提供部署以安全性和合規性 AWS 為重點的基準環境的步驟。
- 在 Amazon Web Services 上架構 HIPAA 安全性與合規性 — 本白皮書說明公司如何使用建立符合 HIPAA 資格的應 AWS 用程式。

> **Note**
>
> 並非所有人 AWS 服務 都符合 HIPAA 資格。如需詳細資訊，請參閱 HIPAA 資格服務參照。

- AWS 合規資源AWS — 此工作簿和指南集合可能適用於您的產業和所在地。
- AWS 客戶合規指南 — 透過合規的角度瞭解共同的責任模式。這份指南總結了在多個架構 (包括美國國家標準技術研究所 (NIST)、支付卡產業安全標準委員會 (PCI) 和國際標準化組織 (ISO)) 中，保 AWS 服務 護指引並對應至安全控制的最佳實務。
- 使用AWS Config 開發人員指南中的規則評估資源 — 此 AWS Config 服務會評估您的資源組態符合內部實務、產業準則和法規的程度。
- AWS Security Hub— 這 AWS 服務 提供了內部安全狀態的全面視圖 AWS。Security Hub 使用安全控制，可評估您的 AWS 資源並檢查您的法規遵循是否符合安全業界標準和最佳實務。如需支援的服務和控制清單，請參閱 Security Hub controls reference。
- Amazon GuardDuty — 透過監控環境中的 AWS 帳戶可疑和惡意活動，藉此 AWS 服務 偵測您的工作負載、容器和資料的潛在威脅。 GuardDuty 可協助您因應各種合規性需求，例如 PCI DSS，滿足特定合規性架構所規定的入侵偵測需求。
- AWS Audit Manager— 這 AWS 服務 有助於您持續稽核您的 AWS 使用情況，以簡化您管理風險的方式，以及遵守法規和業界標準的方式。

Amazon Keyspaces 中的復原和災難復原功能

AWS 全球基礎架構是以 AWS 區域 與可用區域為中心建置的。AWS 區域 提供多個分開且隔離的實際可用區域，並以具備低延遲、高輸送量和高度備援特性的聯網相互連結。透過可用區域，您所設計與操作的應用程式和資料庫，就能夠在可用區域之間自動容錯移轉，而不會發生中斷。可用區域的可用性、容錯能力和擴充能力，均較單一或多個資料中心的傳統基礎設施還高。

Amazon Keyspaces space 會在相同的多個AWS可用區域中自動複寫資料三次，以提高耐AWS 區域用性和高可用性。

如需 AWS 區域 與可用區域的詳細資訊，請參閱 AWS全球基礎設施。

除了AWS全域基礎設施外，Amazon Keyspace 還提供數種支援空間和備份需求的功能。

多區域複寫複寫

如果您需要在更大的地理或應用程式複寫時，Amazon Keyspaces 複寫提供多區域複寫複寫功能。您可以在多達六個不同AWS 區域的選擇之間複製 Amazon 密鑰空間表。如需詳細資訊，請參閱多區域複製。

P oint-in-time 復原 (PITR)

PITR 提供持續備份或刪除操作，PITR 資料表免遭意外寫入或刪除操作。如需詳細資訊，請參閱 Amazon Keyspaces 的 P oint-in-time 復原。

Amazon 密鑰空間中的基礎設施安全

作為一種託管服務，亞馬遜密鑰空間（對於 Apache 卡桑德拉）受到AWS全球網絡安全的保護。如需有關 AWS 安全服務以及 AWS 如何保護基礎設施的詳細資訊，請參閱 AWS 雲端安全。若要使用基礎設施安全性的最佳實務來設計您的 AWS 環境，請參閱安全性支柱 AWS 架構良好的框架中的基礎設施保護。

您可以使用AWS已發佈的 API 呼叫透過網路存取 Amazon 金鑰空間。用戶端必須支援下列項目：

- Transport Layer Security (TLS)。我們需要 TLS 1.2 並建議使用 TLS 1.3。
- 具備完美轉送私密 (PFS) 的密碼套件，例如 DHE (Ephemeral Diffie-Hellman) 或 ECDHE (Elliptic Curve Ephemeral Diffie-Hellman)。現代系統 (如 Java 7 和更新版本) 大多會支援這些模式。

此外，請求必須使用存取索引鍵 ID 和與 IAM 主體相關聯的私密存取索引鍵來簽署。或者，您可以使用 AWS Security Token Service (AWS STS) 來產生暫時安全憑證來簽署請求。

Amazon 金鑰空間支援兩種驗證用戶端請求的方法。第一種方法使用服務特定的登入資料，這些登入資料是針對特定 IAM 使用者產生的密碼型認證。您可以使用 IAM 主控台、或 AWS API 建立和管理密碼。AWS CLI如需詳細資訊，請參閱將 IAM 與 Amazon 金鑰空間搭配使用。

第二種方法使用用於卡桑德拉的開源 DataStax Java 驅動程序的身份驗證插件。此外掛程式可讓 IAM 使用者、角色和聯合身分，使用AWS簽名版本 4 程序 (Sigv4) 將身份驗證資訊新增至 Amazon 金鑰空間 (適用於 Apache Cassandra) API 請求。如需詳細資訊，請參閱the section called “身分 AWS 驗證的 IAM 登入資”。

您可以使用界面 VPC 端點來防止 Amazon VPC 和亞馬遜密鑰空間之間的流量離開亞馬遜網絡。介面虛擬私人雲端端點採用這項AWS技術AWS PrivateLink，可透過彈性網路界面與 Amazon VPC 中的私

有 IP，在AWS服務之間進行私有通訊。如需詳細資訊，請參閱 the section called “使用 介面 VPC 端點”。

將 Amazon Keyspaces 與介面 VPC 端點搭配使用

介面 VPC 端點可讓您在 Amazon VPC 中執行的虛擬私有雲 (VPC) 與 Amazon Keyspaces 間之間進行私有通訊。介面 VPC 端點由支援 AWS PrivateLink，這是一項 AWS 服務，可在 VPC 和 AWS 服務之間進行私有通訊。

AWS PrivateLink 透過在 VPC 中使用具有私有 IP 地址的 elastic network interface 來啟用此功能，以便網路流量不會離開 Amazon 網路。介面 VPC 端點不需要網際網路閘道、NAT 裝置、VPN 連接或是 AWS Direct Connect 連線。如需詳細資訊，請參閱 Amazon Virtual Private Cloud 端和介面 VPC 私有雲端節點 (AWS PrivateLink)。

主題

- 使用 Amazon Keyspaces 的界面 VPC 端點
- 使用介面 VPC 端點資訊填入system.peers表格項目
- 控制 Amazon Keyspaces 對介面 VPC 端點的存取
- 可用性
- VPC 端點政策和 Amazon Keyspaces point-in-time 恢復 (PITR)
- 常見錯誤和警告

使用 Amazon Keyspaces 的界面 VPC 端點

您可以建立介面 VPC 端點，以便 Amazon Keyspaces 間和 Amazon VPC 資源之間的流量開始流經界面 VPC 端點。若要開始使用，請依照下列步驟建立介面端點。接下來，編輯與您在上一個步驟中建立的端點相關聯的安全性群組，並設定連接埠 9142 的輸入規則。如需詳細資訊，請參閱新增、移除和更新規則。

如需透過 VPC 端點設定連線至 Amazon Keyspaces 的 step-by-step 教學課程，請參閱。the section called “連線至 VPC 端點”要了解如何為 Amazon Keyspaces 資源設定跨帳戶存取，與 VPC 中不同 AWS 帳戶 的應用程式分開，請參閱。the section called “跨帳戶存取”

使用介面 VPC 端點資訊填入`system.peers`表格項目

阿帕奇卡桑德拉驅動程序使用該`system.peers`表來查詢有關集群的節點信息。卡桑德拉驅動程序使用節點信息來負載平衡連接並重試操作。Amazon Keyspaces 會為透過公有端點連線的用戶端自動在`system.peers`表格中填入九個項目。

為了讓透過虛擬私人雲端端點連線的用戶端具有類似功能，Amazon Keyspaces space 會在您的帳戶中填入system.peers資料表，並針對可用 VPC 端點的每個可用區域填入一個項目。若要在system.peers表格中查詢並存放可用的介面 VPC 端點，Amazon Keyspaces 要求您授與用於連線到 Amazon Keyspaces 存取權限的 IAM 實體，以查詢 VPC 以取得端點和網路界面資訊。

> **Important**
>
> 將可用的介面 VPC 端點填入system.peers表格，可改善負載平衡並增加讀取/寫入輸送量。建議所有使用界面 VPC 端點訪問 Amazon Keyspaces 的客戶，這是 Apache Spark 所必需的。

若要授與用於連線到 Amazon Keyspaces 許可的 IAM 實體以查詢必要的界面 VPC 端點資訊，您可以更新現有的 IAM 角色或使用者政策，或建立新的 IAM 政策，如以下範例所示。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ListVPCEndpoints",
         "Effect":"Allow",
         "Action":[
            "ec2:DescribeNetworkInterfaces",
            "ec2:DescribeVpcEndpoints"
         ],
         "Resource":"*"
      }
   ]
}
```

> **Note**
>
> 受管政策AmazonKeyspacesReadOnlyAccess_v2和AmazonKeyspacesFullAccess包含允許 Amazon Keyspaces 存取 Amazon EC2 執行個體以讀取有關可用界面 VPC 端點的資訊的必要許可。

若要確認原則已正確設定，請查詢system.peers表格以查看網路資訊。如果資system.peers料表是空的，可能表示政策尚未成功設定，或者您已超過 DescribeVPCEndpoints API 動作

的DescribeNetworkInterfaces要求率配額。 DescribeVPCEndpoints屬於該Describe*類別，被認為是非變異動作。 DescribeNetworkInterfaces屬於未過濾和未分頁的非變異操作的子集，並適用不同的配額。如需詳細資訊，請參閱 Amazon EC2 API 參考中的請求令牌儲存貯體大小和重新填充率。

如果您看到空白資料表，請在幾分鐘後再試一次，以排除要求率配額問題。若要確認您已正確設定 VPC 端點，請參閱the section called “VPC 端點連線錯誤”。如果您的查詢傳回資料表中的結果，表示您的原則已正確設定。

控制 Amazon Keyspaces 對介面 VPC 端點的存取

使用 VPC 端點原則，您可以透過兩種方式控制資源的存取：

- IAM 政策 — 您可以控制允許透過特定 VPC 端點存取 Amazon Keyspaces 的請求、使用者或群組。您可以在附加至 IAM 使用者、群組或角色的政策中使用條件金鑰來執行此操作。
- VPC 政策 — 您可以透過將政策附加到這些 VPC 端點來控制哪些 VPC 端點可以存取 Amazon Keyspaces 資源。若要限制對特定金鑰空間或資料表的存取，以僅允許流量通過特定 VPC 端點，請編輯限制資源存取的現有 IAM 政策，然後新增該 VPC 端點。

以下是存取 Amazon Keyspaces 資源的端點政策範例。

- IAM 政策範例：除非流量來自指定的 VPC 端點，否則限制對特定 Amazon Keyspaces 表的所有存取 — 此範例政策可附加至 IAM 使用者、角色或群組。除非傳入流量來自指定的 VPC 端點，否則它會限制對指定 Amazon Keyspaces 表的存取。

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "UserOrRolePolicyToDenyAccess",
         "Action": "cassandra:*",
         "Effect": "Deny",
         "Resource": [
                         "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
mykeyspace/table/mytable",
                         "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
           ],
         "Condition": { "StringNotEquals" : { "aws:sourceVpce": "vpce-abc123" } }
```

```
      }
   ]
}
```

> **Note**
>
> 若要限制對特定表格的存取，您還必須包括對系統表格的存取權。系統表格是唯讀的。

- VPC 原則範例：唯讀存取 — 此範例原則可附加至 VPC 端點。如需詳細資訊，請參閱控制 Amazon VPC 資源的存取權)。它將動作限制為透過連接到其所連接的 VPC 端點對 Amazon Keyspaces 資源的唯讀存取。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadOnly",
      "Principal": "*",
      "Action": [
        "cassandra:Select"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

- VPC 政策範例：限制對特定 Amazon Keyspaces 表格的存取 — 此範例政策可附加至 VPC 端點。它會透過所連接的 VPC 端點限制對特定資料表的存取。

```
{
   "Version": "2012-10-17",
   "Statement": [
        {
            "Sid": "RestrictAccessToTable",
            "Principal": "*",
            "Action": "cassandra:*",
            "Effect": "Allow",
            "Resource": [
                        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
mykeyspace/table/mytable",
                        "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
```

```
            ]
         }
   ]
}
```

> **Note**
>
> 若要限制對特定表格的存取，您還必須包括對系統表格的存取權。系統表格是唯讀的。

可用性

Amazon Keyspaces 支援在所有可用服務的 AWS 區域 地方使用介面 VPC 端點。如需詳細資訊，請參閱 ???。

VPC 端點政策和 Amazon Keyspaces point-in-time 恢復 (PITR)

如果您使用 IAM 政策搭配條件金鑰來限制傳入流量，表格還原作業可能會失敗。例如，如果您使用`aws:SourceVpce`條件金鑰將來源流量限制到特定的 VPC 端點，則資料表還原作業會失敗。若要允許 Amazon 金 Keyspaces 代表您的主體執行還原作業，您必須將`aws:ViaAWSService`條件金鑰新增至 IAM 政策。當任何 AWS 服務使用主體的認證發出要求時，`aws:ViaAWSService`條件索引鍵允許存取。如需詳細資訊，請參閱 IAM 使用者指南中的 IAM JSON 政策元素：條件金鑰。以下策略是這方面的一個例子。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"CassandraAccessForVPCE",
         "Effect":"Allow",
         "Action":"cassandra:*",
         "Resource":"*",
         "Condition":{
            "Bool":{
               "aws:ViaAWSService":"false"
            },
            "StringEquals":{
               "aws:SourceVpce":[
                  "vpce-12345678901234567"
               ]
            }
```

```
         }
      },
      {
         "Sid":"CassandraAccessForAwsService",
         "Effect":"Allow",
         "Action":"cassandra:*",
         "Resource":"*",
         "Condition":{
            "Bool":{
               "aws:ViaAWSService":"true"
            }
         }
      }
   ]
}
```

常見錯誤和警告

如果您使用的是 Amazon Virtual Private Cloud，並連線到 Amazon Keyspaces，您可能會看到下列警告。

```
Control node cassandra.us-east-1.amazonaws.com/1.111.111.111:9142 has an entry
 for itself in system.peers: this entry will be ignored. This is likely due to a
 misconfiguration;
please verify your rpc_address configuration in cassandra.yaml on all nodes in your
 cluster.
```

發生此警告的原因是`system.peers`表格包含 Amazon Keyspaces 具有檢視權限的所有 Amazon VPC 端點的項目，包括您所連接的 Amazon VPC 端點。您可以放心地忽略此警告。

如需其他錯誤，請參閱the section called “VPC 端點連線錯誤”。

Amazon Keyspaces space 的組態與漏洞分析

AWS 會處理訪客作業系統 (OS) 和資料庫修補、防火牆組態和災難復原等基本安全任務。這些程序已由適當的第三方進行檢閱並認證。如需詳細資訊，請參閱以下 資源：

- 共同責任模式
- Amazon Web Services：安全程序概觀 (白皮書)

Amazon Keyspaces 的安全最佳實務

Amazon Keyspaces (適用於 Apache Cassandra) 提供的多種安全功能，以在您開發和實作自己的安全政策時考慮使用。以下最佳實務為一般準則，並不代表完整的安全解決方案。這些最佳實務可能不適用或無法滿足您的環境需求，因此請將其視為實用建議就好，而不要當作是指示。

主題

Amazon Keyspaces 的預防性安全最佳實務

以下安全最佳實務可視為預防性，因為它們可協助您預測並預防 Amazon Keyspaces 中的安全性事件。

靜態加密

Amazon 金 Keyspaces 使用儲存在 AWS Key Management Service(AWS KMS) 中的加密金鑰，來對存放在資料表中的所有使用者資料進行靜態加密。如此可透過保護您的資料免於發生未經授權的基礎儲存體存取，為資料提供另一層保護。

根據預設，Amazon Keyspaces 會使AWS 擁有的金鑰用一個來加密所有資料表。如果這個金鑰不存在，就會為您建立。服務預設金鑰無法停用預設金鑰。

此外，您可以使用客戶受管金鑰進行靜態加密。如需詳細資訊，請參閱靜態加密。

使用 IAM 角色驗證 Amazon Keyspaces 的存取權

對於要存取 Amazon Keyspaces 的使用者、應用程式和其他AWS服務，其必須在AWS API 請求中包含有效的AWS登入資料。您不應該將 AWS 登入資料直接存放在應用程式或 EC2 執行個體中。這些是不會自動輪換的長期登入資料，因此如果遭到盜用，可能會對業務造成嚴重的影響。IAM 角色可讓您取得臨時存取金鑰，用於存取 AWS 服務和資源。

如需詳細資訊，請參閱 IAM 角色。

使用 IAM 政策進行 Amazon Keyspaces 基礎授權

授予許可時，會決定誰可以取得許可、可以取得哪些 Amazon Keyspaces space API 許可，以及可以對這些資源進行的特定動作。對降低錯誤或惡意意圖所引起的安全風險和影響而言，實作最低權限是其中關鍵。

將許可政策連接至 IAM 身分 (即使用者、群組和角色)，藉此授予可在 Amazon Keyspaces 資源上執行操作的許可。

您可以使用下列內容執行這項作業：

- AWS受管 (預先定義) 政策
- 客戶受管政策

使用 IAM 政策條件進行精細定義存取控制

您在 Amazon Keyspaces 授予許可時，可以指定條件，以決定許可政策的生效方式。對降低錯誤或惡意意圖所引起的安全風險和影響而言，實作最低權限是其中關鍵。

您可以指定使用 IAM 政策授予許可時的條件。例如，您可以執行下列動作：

- 授予許可，允許使用者唯讀存取特定金鑰空間或資料表。
- 根據使用者的身分，授與權限，以允許使用者寫入特定資料表的存取權。

如需詳細資訊，請參閱以身分為基礎的政策範例。

考慮用戶端加密

如果將敏感或機密資料存放在 Amazon Keyspace 中，您可能會想在盡可能接近資料來源的位置加密資料，以便在整個生命週期中保護資料。將您傳輸中和靜態的敏感資料加密，有助於確保您的明文資料不會被任何第三方取得。

Amazon Keyspaces 的 Detective 性安全最佳實務

以下安全最佳實務會被視為偵測性，因為這些實務能幫助您偵測潛在安全弱點與事件。

用AWS CloudTrail來監控AWS Key Management Service (AWS KMS)AWS KMS 金鑰使用情況

如果使用客戶受管AWS KMS金鑰進行靜態加密，那麼使用此金鑰會登入AWS CloudTrail。CloudTrail 透過記錄對帳戶所採取的動作，來提供使用者活動的可見性。 CloudTrail 記錄每個動作的重要資訊，包括提出請求的人、使用過的服務、該動作的參數以及透過AWS服務傳回的回應元素。此資訊可協助您追蹤對AWS資源所做的改變，並診斷解作業問題。 CloudTrail 可讓您更輕鬆地確保內部政策和法規標準的合規性。

您可以使用 CloudTrail 來稽核金鑰使用情形。 CloudTrail 會建立日誌檔，其中包含AWS API 呼叫及您帳戶相關活動歷史記錄的歷史記錄。這些日誌檔包含使用主控台、AWS軟體開發套件和命令列工具，以及透過整合AWS服務提出的所有AWS KMS API 請求。您可以使用這些日誌檔來取得使用AWS KMS金鑰的時間、所請求的操作、請求的身分、請求的來源 IP 地址等資訊。如需詳細資

訊，請參閱《AWS CloudTrail 使用者指南》中的記錄 AWS Key Management Service API 呼叫 和 AWS CloudTrail。

用 CloudTrail 來監控 Amazon Keyspaces 資料定義語言 (DDL) 操作

CloudTrail 透過記錄對帳戶所採取的動作，來提供使用者活動的可見性。 CloudTrail 記錄每個動作的重要資訊，包括提出請求的人、使用過的服務、該動作的參數以及透過AWS服務傳回的回應元素。此資訊可協助您追蹤對AWS資源所做的改變，並診斷解作業問題。 CloudTrail 可讓您更輕鬆地確保內部政策和法規標準的合規性。

所有亞馬遜 Keyspaces DDL 操作都 CloudTrail 會自動登錄。DDL 操作可讓您建立及管理 Amazon Keyspaces 間和資料表。

Amazon Keyspaces space 中發生活動時，該活動會與事件歷史記錄中的其他服務 CloudTrail 事件記錄至事件歷史記錄中的其他AWS服務事件。如需詳細資訊，請參閱使用記錄 Amazon Keyspaces 作業AWS CloudTrail。您可以檢視、搜尋和下載 AWS 帳戶 的最新事件。如需詳細資訊，請參閱AWS CloudTrail使用指南中的檢視具有 CloudTrail 事件歷程記錄的事件。

若要持續記錄您的事件AWS 帳戶，包括 Amazon Keyspaces 的事件，請建立線索。追蹤可讓日誌檔案交付 CloudTrail 至 Amazon Simple Storage Service (Amazon S3) 儲存貯體。根據預設，當您在主控台建立追蹤記錄時，追蹤記錄會套用到所有追蹤記錄AWS 區域。該追蹤會記錄來自 AWS 分割區中所有區域的事件，並將日誌檔案交付到您指定的 S3 儲存貯體。此外，您可以設定其他AWS服務，以進一步分析和處理 CloudTrail 日誌中所收集的事件資料。

為您的 Amazon Keyspaces 資源加上標籤，以便進行識別

您可以用標籤的形式將中繼資料指派給 AWS 資源。每個標記都是由客戶定義金鑰和選用值組成的簡單標籤，能夠更輕鬆地管理、搜尋和篩選資源。

標記允許實現分組控制。雖然標籤不具固有類型，但能讓您依用途、擁有者、環境或其他條件分類資源。下列是一些範例：

- 存取：用於根據標籤控制 Amazon Keyspaces 資源的存取權。如需詳細資訊，請參閱the section called “基於 Amazon Keyspaces 標籤的授權”。
- 安全性：用於確定資料保護設定等需求。
- 機密性：資源支援的特定資料機密等級識別符。
- 環境：用來區分開發、測試和生產基礎設施。

如需詳細資訊，請參閱AWS標記策略和將標籤和標籤新增至資源。

Amazon Keyspaces 的 CQL 語言參考（阿帕奇卡桑德拉）

連接到 Amazon Keyspaces（Apache 卡桑德拉）端點後，您可以使用卡桑德拉查詢語言（CQL）與您的數據庫一起工作。CQL 在許多方面與結構化查詢語言（SQL）相似。

主題

Amazon Keyspaces 中的卡桑德拉查詢語言（CQL）元素

了解 Amazon Keyspaces 支援的 Cassandra 查詢語言 (CQL) 元素，包括識別碼、常數、術語和資料類型。

主題

識別碼

標識符（或名稱）用於標識表，列和其他對象。標識符可以引用或不引用。以下內容適用。

```
identifier          ::=  unquoted_identifier | quoted_identifier
unquoted_identifier ::=  re('[a-zA-Z][a-zA-Z0-9_]*')
quoted_identifier   ::=  '"' (any character where " can appear if doubled)+ '"'
```

常數

定義了下列常數。

```
constant ::=  string | integer | float | boolean | uuid | blob | NULL
string   ::=  '\'' (any character where ' can appear if doubled)+ '\''
              '$$' (any character other than '$$') '$$'
integer  ::=  re('-?[0-9]+')
float    ::=  re('-?[0-9]+(\.[0-9]*)?([eE][+-]?[0-9+])?') | NAN | INFINITY
boolean  ::=  TRUE | FALSE
uuid     ::=  hex{8}-hex{4}-hex{4}-hex{4}-hex{12}
hex      ::=  re("[0-9a-fA-F]")
blob     ::=  '0' ('x' | 'X') hex+
```

條款

術語表示支援的值種類。術語由以下內容定義。

```
term                ::=  constant | literal | function_call | arithmetic_operation |
 type_hint | bind_marker
literal             ::=  collection_literal | tuple_literal
function_call       ::=  identifier '(' [ term (',' term)* ] ')'
arithmetic_operation ::=  '-' term | term ('+' | '-' | '*' | '/' | '%') term
```

資料類型

Amazon Keyspaces 支援下列資料類型：

字串類型

資料類型	描述
ascii	代表一個 ASCII 字元字串。
text	表示一個 UTF-8 編碼的字符串。
varchar	表示 UTF-8 編碼字串 (varchar是的別名text)。

數值類型

資料類型	描述
bigint	表示 64 位元簽署的長度。
counter	表示 64 位元帶正負號整數計數器。如需詳細資訊，請參閱 the section called “欄檯”。
decimal	表示可變精度小數。
double	代表一個 64 位元的 IEEE 754 浮點數。
float	表示一個 32 位元的 IEEE 754 浮點數。
int	表示 32 位元帶正負號整數。
varint	表示任意精度的整數。

欄檯

counter資料行包含 64 位元帶正負號的整數。計數器值會使用the section called “UPDATE”陳述式遞增或遞減，而且無法直接設定。這使得counter列對跟踪計數非常有用。例如，您可以使用計數器來追蹤記錄檔中的項目數目，或在社交網路上檢視貼文的次數。下列限制適用於counter欄：

- 類型的欄counter不能是表格primary key的一部分。
- 在包含一或多個類型欄的資料表中counter，該資料表中的所有資料行都必須是類型counter。

如果計數器更新失敗 (例如，由於逾時或與 Amazon Keyspaces 的連線中斷)，用戶端不知道計數器值是否已更新。如果重試更新，計數器值的更新可能會再次套用。

斑點類型

資料類型	描述
blob	表示任意位元組。

布林值 (Boolean) 類型

資料類型	描述
boolean	代表true或false。

時間相關類型

資料類型	描述
timestamp	64 位元帶正負號的整數，表示自紀元（1970 年 1 月 1 日，格林威治標準時間 00:00:00）以毫秒為單位的日期和時間
timeuuid	表示一個版本 1 的 UUID。

集合類型

資料類型	描述
list	表示常值項目的有序集合。
map	表示鍵值對的無序集合。
set	代表一或多個常值項目的無序集合。

您可以使用集合類型後面加上角括號中的另一種資料類型 (例如TEXT或INT) 來宣告集合資料行。您可以使用SET的建立欄TEXT，也可以建立MAP的TEXT和INT索引鍵值配對，如下列範例所示。

```
SET <TEXT>
MAP <TEXT, INT>
```

非凍結的收集可讓您更新每個個別的收集要素。會儲存個別元素的用戶端時間戳記和存留時間 (TTL) 設定。

當您在集合類型上使用FROZEN關鍵字時，集合的值會序列化為單一不可變值，而 Amazon Keyspaces space 會將它們視為. BLOB 這是一個凍結的集合。INSERT或UPDATE陳述式會覆寫整個凍結集合。您無法更新凍結集合中的個別元素。

用戶端時間戳記和存留時間 (TTL) 設定會套用至整個凍結集合，而非個別元素。 Frozen集合欄可以是表格PRIMARY KEY的一部分。

您可以巢狀凍結的集合。例如，SET如果使用FROZEN關鍵字，您可以在 a MAP 中定義一個，如下列範例所示。MAP

```
SET <FROZEN> <MAP <TEXT, INT>>>
```

Amazon Keyspaces 預設支援嵌套多達五個層級的凍結集合。如需詳細資訊，請參閱 the section called “Amazon Keyspaces 服務配額”。有關功能差異與 Apache 卡桑德拉的更多信息，請參閱。the section called “FROZEN集合”如需 CQL 語法的詳細資訊，請參閱the section called “CREATE TABLE”和the section called “ALTER TABLE”。

元組類型

資tuple料類型代表文字元素的有界群組。您可以使用元組作為user defined type. 您不需要為元組使用FROZEN關鍵字。這是因為一個元組總是被凍結，你不能單獨更新元素。

其他類型

資料類型	描述
inet	代表 IP 位址的字串，採用 IPv4 或 IPv6 格式。

靜態

在具有叢集資料行的 Amazon Keyspace 表格中，您可以使用STATIC關鍵字建立任何類型的靜態資料欄。

下面的語句是這樣的一個例子。

```
my_column INT STATIC
```

如需使用靜態欄的詳細資訊，請參閱the section called “靜態列”。

Amazon Keyspaces 數據類型的 JSON 編碼

Amazon Keyspaces 提供相同的 JSON 數據類型映射作為阿帕奇卡桑德拉。下表說明 Amazon Keyspaces 在INSERT JSON陳述式中接受的資料類型，以及使用陳述式傳回資料時 Amazon Keyspaces 使用的資料類型。SELECT JSON

對於單欄位資料類型 (例如floatint、UUID、和)date，您也可以將資料插入為string. 對於複合資料類型和集合 (例如tuplemap、和)list，您也可以將資料插入為 JSON 或編碼JSON string。

JSON 資料類型	**INSERT JSON**語句中接受的數據類型	**SELECT JSON**陳述式中傳回的資料類型	備註
ascii	string	string	使用 JSON 字符轉義\u。
bigint	integer, string	integer	字串必須是有效的 64 位元整數。
blob	string	string	字符串應該以0x偶數十六進制數字開始。
boolean	boolean, string	boolean	字串必須是true或false。
date	string	string	日期格式YYYY-MM-DD ，時區 UTC。
decimal	integer, float, string	float	在用戶端解碼器中，可以超過 32 位元或 64 位元 IEEE-754 浮點精度。
double	integer, float, string	float	字符串必須是有效的整數或浮點數。
float	integer, float, string	float	字符串必須是有效的整數或浮點數。
inet	string	string	IPv4 或 IPv6 位址。

JSON 資料類型	**INSERT JSON**語句中接受的數據類型	**SELECT JSON**陳述式中傳回的資料類型	備註
int	integer, string	integer	字串必須是有效的 32 位元整數。
list	list, string	list	使用原生 JSON 清單表示法。
map	map, string	map	使用原生 JSON 對應表示法。
smallint	integer, string	integer	字串必須是有效的 16 位元整數。
set	list, string	list	使用原生 JSON 清單表示法。
text	string	string	使用 JSON 字符轉義\u。
time	string	string	以格式顯示的一天中的時間HH-MM-SS[.fffffffff] 。
timestamp	integer, string	string	時間戳記。字符串常量允許您將時間戳存儲為日期。返回帶有格式YYYY-MM-DD HH:MM:SS.SSS 的日期戳。
timeuuid	string	string	類型 1 無線識別碼。如需 UUID 格式的constants資訊，請參閱。
tinyint	integer, string	integer	字串必須是有效的 8 位元整數。

JSON 資料類型	**INSERT JSON**語句中接受的數據類型	**SELECT JSON**陳述式中傳回的資料類型	備註
`tuple`	`list, string`	`list`	使用原生 JSON 清單表示法。
`uuid`	`string`	`string`	如需 UUID 格式的constants資訊，請參閱。
`varchar`	`string`	`string`	使用 JSON 字符轉義\u。
`varint`	`integer, string`	`integer`	可變長度；用戶端解碼器中可能會溢位 32 位元或 64 位元整數。

Amazon Keyspaces 中的 DDL 語句（數據定義語言）

數據定義語言（DDL）是一組卡桑德拉查詢語言（CQL）語句，用於管理 Amazon Keyspaces 中的數據結構（對於 Apache 卡桑德拉），如密鑰空間和表。您可以使用 DDL 來建立這些資料結構、在建立後修改它們，並在不再使用時將其移除。Amazon Keyspaces 以異步方式執行 DDL 操作。如需如何確認非同步作業是否已完成的詳細資訊，請參閱the section called “異步創建和刪除密鑰空間和表”。

支援下列 DDL 陳述式：

- 創建密鑰空間
- 更改密鑰空間
- 刪除密鑰空間
- 建立表格
- ALTER TABLE
- 還原表格
- 拖放表

主題

- Keyspaces

- 資料表

Keyspaces

索引鍵空間會將與一或多個應用程式相關的相關資料表分組。就關係數據庫管理系統（RDBMS）而言，密鑰空間與數據庫，表空間或類似的構造大致相似。

> **Note**
>
> 在 Apache 卡桑德拉，密鑰空間確定數據如何在多個存儲節點之間複製。不過，Amazon Keyspaces 是全受管服務：其儲存層的詳細資料會代表您管理。因此，Amazon Keyspaces 中的密鑰空間僅為邏輯結構，與底層實體儲存無關。

如需 Amazon Keyspaces 的配額限制和限制的相關資訊，請參閱。配額

密鑰空間的語句

- 創建密鑰空間
- 改變密鑰空間
- 刪除密鑰空間

創建密鑰空間

使用該CREATE KEYSPACE語句來創建一個新的密鑰空間。

語法

```
create_keyspace_statement ::=
    CREATE KEYSPACE [ IF NOT EXISTS ] keyspace_name
    WITH options
```

其中：

- *keyspace_name*是要創建的密鑰空間的名稱。
- 選項為下列一或多個選項：
  - REPLICATION-表示密鑰空間複製策略的映射：
    - SingleRegionStrategy— 針對單一區域金鑰空間。(必要)

- NetworkTopologyStrategy— 指定至少兩個和最多六個 AWS 區域。每個區域的複寫因子是三個。(選用)
- DURABLE_WRITES— 寫入 Amazon Keyspaces 始終是耐用的，因此不需要此選項。但是，如果指定，該值必須是true。
- TAGS— 建立資源時要附加至資源的索引鍵值配對標籤清單。(選用)

範例

創建一個密鑰空間，如下所示。

```
CREATE KEYSPACE my_keyspace
    WITH REPLICATION = {'class': 'SingleRegionStrategy'} and TAGS ={'key1':'val1',
 'key2':'val2'} ;
```

若要建立多區域金鑰空間，請指定NetworkTopologyStrategy並包含至少兩個和最多六個。 AWS 區域每個區域的複寫因子是三個。

```
CREATE KEYSPACE my_keyspace
    WITH REPLICATION = {'class':'NetworkTopologyStrategy', 'us-east-1':'3', 'ap-
southeast-1':'3','eu-west-1':'3'};
```

改變密鑰空間

使用ALTER KEYSPACE從金鑰空間新增或移除標籤。

語法

```
alter_keyspace_statement ::=
    ALTER KEYSPACE keyspace_name
    [[ADD | DROP] TAGS
```

其中：

- *keyspace_name*是要修改的密鑰空間的名稱。
- TAGS-要從密鑰空間中添加或刪除的鍵值對標籤列表。

範例

改變密鑰空間如下。

```
ALTER KEYSPACE "myGSGKeyspace" ADD TAGS {'key1':'val1', 'key2':'val2'};
```

刪除密鑰空間

使用DROP KEYSPACE陳述式移除金鑰空間，包括其所有內容，例如表格。

語法

```
drop_keyspace_statement ::=
    DROP KEYSPACE [ IF EXISTS ] keyspace_name
```

其中：

- 密鑰空間名稱是要刪除的密鑰空間的名稱。

範例

```
DROP KEYSPACE "myGSGKeyspace";
```

資料表

表是 Amazon Keyspaces 中的主要數據結構。表中的數據被組織成行和列。這些資料行的子集是透過指定分割索引鍵來判斷資料分割 (最終資料放置)。

另一組資料行可以定義為叢集資料行，這表示它們可以在查詢執行中以述詞的形式參與。

依預設，會以隨需輸送量容量建立新表格。您可以變更新表格和現有資料表的容量模式。如需讀取/寫入容量輸送量模式的詳細資訊，請參閱 the section called “讀/寫容量模式”

對於佈建模式下的表格，您可以設定選用AUTOSCALING_SETTINGS。如需 Amazon Keyspaces auto 擴展和可用選項的詳細資訊，請參閱the section called “使用定制列表”。

如需 Amazon Keyspaces 資料表的配額限制和限制的相關資訊，請參閱配額。

表格的陳述式

- CREATE TABLE
- ALTER TABLE
- 還原表格

- DROP TABLE

CREATE TABLE

使用該CREATE TABLE語句來創建一個新的表。

語法

```
create_table_statement ::=  CREATE TABLE [ IF NOT EXISTS ] table_name
    '('
        column_definition
        ( ',' column_definition )*
        [ ',' PRIMARY KEY '(' primary_key ')' ]
    ')' [ WITH table_options ]

column_definition      ::=  column_name cql_type [ FROZEN ][ STATIC ][ PRIMARY KEY]

primary_key            ::=  partition_key [ ',' clustering_columns ]

partition_key          ::=  column_name
                              | '(' column_name ( ',' column_name )* ')'

clustering_columns     ::=  column_name ( ',' column_name )*

table_options          ::=  [table_options]
                              | CLUSTERING ORDER BY '(' clustering_order
 ')' [ AND table_options ]
                              | options
                              | CUSTOM_PROPERTIES
                              | AUTOSCALING_SETTINGS
                              | default_time_to_live
                              | TAGS

clustering_order       ::=  column_name (ASC | DESC) ( ',' column_name (ASC | DESC) )*
```

其中：

- *table_name*是要建立的資料表名稱。
- *column_definition*由以下內容組成：
  - *column_name*— 欄的名稱。
  - *cql_type*— Amazon Keyspaces 數據類型 (請參閱資料類型).

- *FROZEN*— 將此類型欄 collection (例如LISTSET、或MAP) 指定為凍結。凍結的集合被序列化為單個不可變值，並像. BLOB 如需詳細資訊，請參閱 the section called “集合類型”。
- *STATIC*— 將此欄指定為靜態。靜態資料行會儲存相同資料分割中所有資料列共用的值。
- *PRIMARY KEY*— 將此欄指定為資料表的主索引鍵。

- *primary_key*由以下內容組成：
  - partition_key
  - clustering_columns
- *partition_key*:
  - 分區鍵可以是單個列，也可以是由兩個或多個列組成的複合值。需要主索引鍵的分區索引鍵部分，並決定 Amazon 金 Keyspaces 如何存放資料。
- *clustering_columns*:
  - 主鍵的可選集群列部分決定了數據在每個分區中聚集和排序的方式。
- *table_options*由下列項目組成：
  - *CLUSTERING ORDER BY*— 資料表上的預設叢集順序是由 ASC (遞增) 排序方向上的叢集索引鍵所組成。指定它以覆寫預設排序行為。
  - *CUSTOM_PROPERTIES*— 特定於 Amazon Keyspaces 的設置地圖。
    - capacity_mode：指定表格的讀取/寫入輸送量容量模式。選項包括 throughput_mode:PAY_PER_REQUEST 和 throughput_mode:PROVISIONED. 佈建的容量模式需要read_capacity_units並write_capacity_units作為輸入。預設值為 throughput_mode:PAY_PER_REQUEST。
    - client_side_timestamps：指定表格是否啟用或停用用戶端時間戳記。選項包括 {'status': 'enabled'} 和 {'status': 'disabled'}. 如果未指定，則預設值為status:disabled。啟用資料表的用戶端時間戳記之後，就無法停用此設定。
    - encryption_specification：指定靜態加密的加密選項。如果未指定，則預設值為encryption_type:AWS_OWNED_KMS_KEY。客戶受管金鑰的加密選項需要 Amazon 資源名稱 (ARN) 格式的 AWS KMS 金鑰作為輸入kms_key_identifier:ARN:kms_key_identifier:ARN。
    - point_in_time_recovery：指定是否啟用或停用表格的 point-in-time 還原。選項包括 status:enabled 和 status:disabled. 如果未指定，則預設值為status:disabled。
    - replica_updates：指定特定於的多區域表格的設定。 AWS 區域對於多區域表格，您可以根據 AWS 區域不同的方式設定表格的讀取容量。您可以透過設定下列參數來執行此操作。如需詳細資訊和範例，請參閱 the section called “建立具有佈建容量模式和 auto 擴展 (CQL) 的多區域表”。

- region— 具有下列設定 AWS 區域 的表格複本：
  - read_capacity_units
- TTL：啟用表格的「存留時間」自訂設定。若要啟用，請使用status:enabled。預設值為 status:disabled。啟TTL用之後，您無法針對表格停用它。

- *AUTOSCALING_SETTINGS*包括佈建模式下表格的下列選用設定。如需詳細資訊和範例，請參閱 the section called “使用 CQL 建立具有自動調整比例的新資料表”。
  - provisioned_write_capacity_autoscaling_update:
    - autoscaling_disabled— 若要啟用寫入容量的 auto 調整，請將值設定為false。預設值為 true。(選用)
    - minimum_units— 資料表應隨時可支援的最低寫入輸送量層級。此值必須介於 1 和帳戶每秒最大輸送量配額之間 (預設為 40,000)。
    - maximum_units— 資料表應隨時可支援的最大寫入輸送量層級。此值必須介於 1 和帳戶每秒最大輸送量配額之間 (預設為 40,000)。
    - scaling_policy— Amazon Keyspaces 支持目標跟踪政策。auto 擴展目標是表格的佈建寫入容量。
      - target_tracking_scaling_policy_configuration— 若要定義目標追蹤原則，您必須定義目標值。如需目標追蹤和冷卻時間的詳細資訊，請參閱《應用程式自動調整規模使用者指南》中的目標追蹤擴展政策
        - target_value— 表格的目標使用率。Amazon Keyspaces auto 擴展可確保消耗容量與佈建容量的比例保持在或接近此值。您能以百分比的形式定義 target_value。在 20 和 90 之間的雙倍。(必要)
        - scale_in_cooldown-縮放活動之間的冷卻時間（以秒為單位），使桌子在活動開始的另一個規模之前穩定下來。如果未提供任何值，則預設值為 0。(選用)
        - scale_out_cooldown-縮放活動之間的冷卻時間（以秒為單位），使桌子在另一個擴展活動開始之前穩定下來。如果未提供任何值，則預設值為 0。(選用)
        - disable_scale_in：指boolean定表格scale-in是否停用或啟用。依預設，會停用此參數。若要開啟scale-in，請將boolean值設定為FALSE。這表示會代表您自動縮減資料表的容量。(選用)
  - provisioned_read_capacity_autoscaling_update:
    - autoscaling_disabled— 若要啟用讀取容量的 auto 調整，請將值設定為false。預設值為 true。(選用)
    - minimum_units— 資料表應隨時準備好支援的最低輸送量層級。此值必須介於 1 和帳戶每秒最大輸送量配額之間 (預設為 40,000)。

- `maximum_units`— 資料表應隨時準備好支援的最大輸送量層級。此值必須介於 1 和帳戶每秒最大輸送量配額之間 (預設為 40,000)。
- `scaling_policy`— Amazon Keyspaces 支持目標跟踪政策。auto 擴展目標是表格的佈建讀取容量。
  - `target_tracking_scaling_policy_configuration`— 若要定義目標追蹤原則，您必須定義目標值。如需目標追蹤和冷卻時間的詳細資訊，請參閱《應用程式自動調整規模使用者指南》中的目標追蹤擴展政策
    - `target_value`— 表格的目標使用率。Amazon Keyspaces auto 擴展可確保消耗容量與佈建容量的比例保持在或接近此值。您能以百分比的形式定義 `target_value`。在 20 和 90 之間的雙倍。(必要)
    - `scale_in_cooldown`-縮放活動之間的冷卻時間（以秒為單位），使桌子在活動開始的另一個規模之前穩定下來。如果未提供任何值，則預設值為 0。(選用)
    - `scale_out_cooldown`-縮放活動之間的冷卻時間（以秒為單位），使桌子在另一個擴展活動開始之前穩定下來。如果未提供任何值，則預設值為 0。(選用)
    - `disable_scale_in`：指boolean定表格scale-in是否停用或啟用。依預設，會停用此參數。若要開啟scale-in，請將boolean值設定為FALSE。這表示會代表您自動縮減資料表的容量。(選用)
- `replica_updates`：指定多區域表格的 AWS 區域 特定 auto 調整比例設定。對於多區域表格，您可以根據 AWS 區域不同的方式設定表格的讀取容量。您可以透過設定下列參數來執行此操作。如需詳細資訊和範例，請參閱 the section called “建立具有佈建容量模式和 auto 擴展 (CQL) 的多區域表”。
  - `region`— 具有下列設定 AWS 區域 的表格複本：
    - `provisioned_read_capacity_autoscaling_update`
      - `autoscaling_disabled`— 若要啟用表格讀取容量的 auto 調整，請將值設定為`false`。預設值為 `true`。(選用)

        > **Note**
        >
        > 必須為表格的所有複本啟用或停用多區域表格的自動調整功能。

      - `minimum_units`— 資料表應隨時可支援的最低讀取輸送量層級。此值必須介於 1 和帳戶每秒最大輸送量配額之間 (預設為 40,000)。
      - `maximum_units`— 表格應隨時可支援的最大讀取輸送量層級。此值必須介於 1 和帳戶每秒最大輸送量配額之間 (預設為 40,000)。

- scaling_policy— Amazon Keyspaces 支持目標跟踪政策。auto 擴展目標是表格的佈建讀取容量。
  - target_tracking_scaling_policy_configuration— 若要定義目標追蹤原則，您必須定義目標值。如需目標追蹤和冷卻時間的詳細資訊，請參閱《應用程式自動調整規模使用者指南》中的目標追蹤擴展政策
    - target_value— 表格的目標使用率。Amazon Keyspaces auto 擴展可確保消耗的讀取容量與佈建讀取容量的比例保持在或接近此值。您能以百分比的形式定義 target_value。在 20 和 90 之間的雙倍。(必要)
    - scale_in_cooldown-縮放活動之間的冷卻時間（以秒為單位），使桌子在活動開始的另一個規模之前穩定下來。如果未提供任何值，則預設值為 0。(選用)
    - scale_out_cooldown-縮放活動之間的冷卻時間（以秒為單位），使桌子在另一個擴展活動開始之前穩定下來。如果未提供任何值，則預設值為 0。(選用)
    - disable_scale_in：指boolean定表格scale-in是否停用或啟用。依預設，會停用此參數。若要開啟scale-in，請將boolean值設定為FALSE。這表示會代表您自動縮減資料表的讀取容量。(選用)
- default_time_to_live— 表格的預設存留時間設定 (以秒為單位)。
- TAGS— 建立資源時要附加至資源的索引鍵值配對標籤清單。

- *clustering_order*由以下內容組成：
  - *column_name*— 欄的名稱。
  - *ASC | DESC*— 設定上升 (ASC) 或後代 (DESC) 順序修飾詞。如果未指定，則預設順序為 ASC。

範例

```
CREATE TABLE IF NOT EXISTS "my_keyspace".my_table (
                                        id text,
                                        name text,
                                        region text,
                                        division text,
                                        project text,
                                        role text,
                                        pay_scale int,
                                        vacation_hrs float,
                                        manager_id text,
                                        PRIMARY KEY (id,division))
                                        WITH CUSTOM_PROPERTIES={
                                            'capacity_mode':{
```

```
                                                    'throughput_mode':
 'PROVISIONED', 'read_capacity_units': 10, 'write_capacity_units': 20
                                                        },
                                                    'point_in_time_recovery':{'status':
 'enabled'},
                                                    'encryption_specification':{
                                                            'encryption_type':
 'CUSTOMER_MANAGED_KMS_KEY',

 'kms_key_identifier':'arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111'
                                                        }
                                                }
                                                AND CLUSTERING ORDER BY (division ASC)
                                                AND TAGS={'key1':'val1', 'key2':'val2'}
                                                AND default_time_to_live = 3024000;
```

在使用叢集資料行的資料表中，非叢集資料行可在資料表定義中宣告為靜態。如需靜態欄的詳細資訊，請參閱the section called “靜態列”。

範例

```
CREATE TABLE "my_keyspace".my_table (
                                                id int,
                                                name text,
                                                region text,
                                                division text,
                                                project text STATIC,
                                                PRIMARY KEY (id,division));
```

ALTER TABLE

使用ALTER TABLE陳述式來新增欄、新增標記或變更資料表的自訂屬性。

語法

```
alter_table_statement ::=  ALTER TABLE table_name

        [ ADD ( column_definition | column_definition_list)  ]
        [[ADD | DROP] TAGS {'key1':'val1', 'key2':'val2'}]
        [ WITH table_options [ , ... ] ] ;

column_definition      ::=  column_name cql_type
```

其中：

- *table_name*是要變更的資料表名稱。
- *column_definition*是要加入的資料行和資料類型的名稱。
- *column_definition_list*是放置在括號內的欄的逗號分隔清單。
- *table_options*由下列項目組成：
  - *CUSTOM_PROPERTIES*— Amazon Keyspaces 特定設置的地圖。
    - capacity_mode：指定表格的讀取/寫入輸送量容量模式。選項包括 throughput_mode:PAY_PER_REQUEST 和 throughput_mode:PROVISIONED. 佈建的容量模式需要read_capacity_units並write_capacity_units作為輸入。預設值為 throughput_mode:PAY_PER_REQUEST。
    - client_side_timestamps：指定表格是否啟用或停用用戶端時間戳記。選項包括 {'status': 'enabled'} 和 {'status': 'disabled'}. 如果未指定，則預設值為status:disabled。啟用資料表的用戶端時間戳記之後，就無法停用此設定。
    - encryption_specification：指定靜態加密的加密選項。選項包括 encryption_type:AWS_OWNED_KMS_KEY 和 encryption_type:CUSTOMER_MANAGED_KMS_KEY. 客戶受管金鑰的加密選項需要 Amazon 資源名稱 (ARN) 格式的 AWS KMS 金鑰作為輸入：kms_key_identifier:ARN。
    - point_in_time_recovery：指定是否啟用或停用表格的 point-in-time 還原。選項包括 status:enabled 和 status:disabled. 預設值為 status:disabled。
    - replica_updates：指 AWS 區域 定多區域表格的特定設定。對於多區域表格，您可以根據 AWS 區域不同的方式設定表格的讀取容量。您可以透過設定下列參數來執行此操作。如需詳細資訊和範例，請參閱 the section called “更新多區域表格 (CQL) 的佈建容量和 auto 擴展設定”。
      - region— 具有下列設定 AWS 區域 的表格複本：
        - read_capacity_units
    - ttl：啟用表格的「存留時間」自訂設定。若要啟用，請使用status:enabled。預設值為 status:disabled。啟ttl用之後，您無法針對表格停用它。
  - *AUTOSCALING_SETTINGS*包含已佈建表格的選用 auto 調整設定。如需語法和詳細描述，請參閱the section called “CREATE TABLE”。如需範例，請參閱 the section called “使用 CQL 在現有資料表上啟用自動調整”。
- default_time_to_live：表格的預設存留時間設定 (以秒為單位)。

- *TAGS*是要附加至資源的索引鍵值配對標籤清單。

> **Note**
>
> 使用 ALTER TABLE，您只能變更單一自訂屬性。您不能在同一個語句中組合多個 ALTER TABLE 命令。

範例

下面的語句演示了如何一列添加到現有的表。

```
ALTER TABLE mykeyspace.mytable ADD (ID int);
```

這個語句顯示了如何將兩個集合列添加到現有的表：

- 包含巢狀凍結集合col_frozen_list的凍結集合的凍結集合欄
- 包含嵌套凍結集合col_map的非凍結集合列

```
ALTER TABLE my_Table ADD(col_frozen_list FROZEN<LIST<FROZEN<SET<TEXT>>>>, col_map
 MAP<INT, FROZEN<SET<INT>>>);
```

若要變更資料表的容量模式並指定讀取和寫入容量單位，您可以使用下列陳述式。

```
ALTER TABLE mykeyspace.mytable WITH CUSTOM_PROPERTIES={'capacity_mode':
{'throughput_mode': 'PROVISIONED', 'read_capacity_units': 10, 'write_capacity_units':
 20}};
```

下列陳述式會指定資料表的客戶管理 KMS 金鑰。

```
ALTER TABLE mykeyspace.mytable WITH CUSTOM_PROPERTIES={
              'encryption_specification':{
                     'encryption_type': 'CUSTOMER_MANAGED_KMS_KEY',
                     'kms_key_identifier':'arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111'
                  }
         };
```

若要啟用資料表的 point-in-time 還原，您可以使用下列陳述式。

```
ALTER TABLE mykeyspace.mytable WITH CUSTOM_PROPERTIES={'point_in_time_recovery':
 {'status': 'enabled'}};
```

若要將資料表的預設「時間」設定為「即時」值 (以秒為單位)，您可以使用下列陳述式。

```
ALTER TABLE my_table WITH default_time_to_live = 2592000;
```

此陳述式會啟用資料表的自訂存留時間設定。

```
ALTER TABLE mytable WITH CUSTOM_PROPERTIES={'ttl':{'status': 'enabled'}};
```

還原表格

使用RESTORE TABLE陳述式將資料表還原到某個時間點。此陳述式需要在資料表上啟用 point-in-time 復原。如需詳細資訊，請參閱 Point-in-time 回收。

語法

```
restore_table_statement ::=
    RESTORE TABLE restored_table_name FROM TABLE source_table_name
                    [ WITH table_options [ , ... ] ];
```

其中：

- *restored_table_name*是還原資料表的名稱。
- *source_table_name*是來源資料表的名稱。
- *table_options*由以下內容組成：
  - *restore_timestamp*是 ISO 8601 格式的還原點時間。如果未指定，則會使用目前的時間戳記。
  - *CUSTOM_PROPERTIES*— Amazon Keyspaces 特定設置的地圖。
    - capacity_mode：指定表格的讀取/寫入輸送量容量模式。選項包括 throughput_mode:PAY_PER_REQUEST 和 throughput_mode:PROVISIONED. 佈建的容量模式需要read_capacity_units並write_capacity_units作為輸入。預設值是來源表格中的目前設定。
    - encryption_specification：指定靜態加密的加密選項。選項包括 encryption_type:AWS_OWNED_KMS_KEY 和

encryption_type:CUSTOMER_MANAGED_KMS_KEY. 客戶受管金鑰的加密選項需要 Amazon 資源名稱 (ARN) 格式的 AWS KMS 金鑰作為輸入：kms_key_identifier:ARN。若要將使用客戶受管金鑰加密的表格還原到使用加密的表格 AWS 擁有的金鑰，Amazon Keyspaces 需要存取來源表格的 AWS KMS 金鑰。

- point_in_time_recovery：指定是否啟用或停用表格的 point-in-time 還原。選項包括 status:enabled 和 status:disabled. 與建立新表格時不同，還原表格的預設狀態是status:enabled因為設定是從來源表格繼承而來。若要針對還原的資料表停用 PITR，您必須status:disabled明確設定。
- replica_updates：指 AWS 區域 定多區域表格的特定設定。對於多區域表格，您可以根據 AWS 區域不同的方式設定表格的讀取容量。您可以透過設定下列參數來執行此操作。
  - region— 具有下列設定 AWS 區域 的表格複本：
    - read_capacity_units

- *AUTOSCALING_SETTINGS*包含已佈建表格的選用 auto 調整設定。如需詳細的語法和描述，請參閱the section called “CREATE TABLE”。
- *TAGS*是要附加至資源的索引鍵值配對標籤清單。

> **Note**
>
> 刪除的表格只能還原到刪除的時間。

範例

```
RESTORE TABLE mykeyspace.mytable_restored from table mykeyspace.my_table
WITH restore_timestamp = '2020-06-30T04:05:00+0000'
AND custom_properties = {'point_in_time_recovery':{'status':'disabled'},
 'capacity_mode':{'throughput_mode': 'PROVISIONED', 'read_capacity_units': 10,
 'write_capacity_units': 20}}
AND TAGS={'key1':'val1', 'key2':'val2'};
```

DROP TABLE

使用該DROP TABLE語句從密鑰空間中刪除表。

語法

```
drop_table_statement ::=
```

```
    DROP TABLE [ IF EXISTS ] table_name
```

其中：

- IF EXISTS如果表不存在，則防止DROP TABLE失敗。(選用)
- *table_name*是要刪除的表的名稱。

範例

```
DROP TABLE "myGSGKeyspace".employees_tbl;
```

Amazon Keyspaces 中的 DML 語句（數據操作語言）

數據操作語言（DML）是一組卡桑德拉查詢語言（CQL）語句，用於管理 Amazon Keyspaces（對於阿帕奇卡桑德拉）表中的數據。您可使用 DML 陳述式新增、修改或刪除資料表中的資料。

您也可以使用 DML 陳述式來查詢資料表中的資料。（請注意，CQL 不支持聯接或子查詢。）

主題

- SELECT
- INSERT
- UPDATE
- DELETE

SELECT

使用 SELECT 陳述式來查詢資料。

語法

```
select_statement ::=  SELECT  [ JSON ] ( select_clause | '*' )
                      FROM table_name
                      [ WHERE 'where_clause' ]
                      [ ORDER BY 'ordering_clause' ]
                      [ LIMIT (integer | bind_marker) ]
                      [ ALLOW FILTERING ]
```

```
select_clause    ::=  selector [ AS identifier ] ( ',' selector [ AS identifier ] )
selector         ::=  column_name
                      | term
                      | CAST '(' selector AS cql_type ')'
                      | function_name '(' [ selector ( ',' selector )* ] ')'
where_clause     ::=  relation ( AND relation )*
relation         ::=  column_name operator term
                      TOKEN
operator         ::=  '=' | '<' | '>' | '<=' | '>=' | IN | CONTAINS | CONTAINS KEY
ordering_clause  ::=  column_name [ ASC | DESC ] ( ',' column_name [ ASC | DESC ] )*
```

範例

```
SELECT name, id, manager_id FROM "myGSGKeyspace".employees_tbl ;

SELECT JSON name, id, manager_id FROM "myGSGKeyspace".employees_tbl ;
```

如需將 JSON 編碼資料類型對應至 Amazon Keyspaces 資料類型的表格，請參閱。the section called “Amazon Keyspaces 數據類型的 JSON 編碼”

使用關**IN**鍵字

IN關鍵字會指定一或多個值的相等性。它可以應用於分區索引鍵和叢集資料行。結果會以索引鍵在SELECT陳述式中呈現的順序傳回。

範例

```
SELECT * from mykeyspace.mytable WHERE primary.key1 IN (1,2) and clustering.key1 = 2;
SELECT * from mykeyspace.mytable WHERE primary.key1 IN (1,2) and clustering.key1 <= 2;
SELECT * from mykeyspace.mytable WHERE primary.key1 = 1 and clustering.key1 IN (1, 2);
SELECT * from mykeyspace.mytable WHERE primary.key1 <= 2 and clustering.key1 IN (1, 2)
 ALLOW FILTERING;
```

如需關IN鍵字以及 Amazon Keyspaces 如何處理陳述式的詳細資訊，請參閱the section called “INSELECT聲明”。

訂購結果

該ORDER BY子句指定返回結果的排序順序。它需要作為參數列的列名列表以及每列的排序順序。您只能在排序子句中指定叢集資料行。不允許使用非叢集資料行。排序順序選項用ASC於升序和DESC降

序排序順序。如果省略排序順序，則會使用叢集資料行的預設順序。如需可能的排序順序，請參閱the section called “訂購結果”。

範例

```
SELECT name, id, division, manager_id FROM "myGSGKeyspace".employees_tbl WHERE id =
 '012-34-5678' ORDER BY division;
```

使ORDER BY用IN關鍵字時，結果會在頁面中排序。不支持禁用分頁完全重新排序。

令牌

您可以將TOKEN函數套用至SELECT和WHERE子句中的PARTITION KEY資料行。使用該TOKEN函數，Amazon Keyspaces 返回基於的對應令牌值，PARTITION_KEY而不是對的值的PARTITION KEY行。

TOKEN關IN鍵字不支援關係。

範例

```
SELECT TOKEN(id) from my_table;

SELECT TOKEN(id) from my_table WHERE TOKEN(id) > 100 and TOKEN(id) < 10000;
```

TTL 函數

您可以將TTL函數與SELECT陳述式搭配使用，擷取為資料行儲存的到期時間 (以秒為單位)。如果未設定任何TTL值，則函數會傳回null。

範例

```
SELECT TTL(my_column) from my_table;
```

此TTL函數無法用於多儲存格欄 (例如集合)。

WRITETIME 函式

只有在資料表使用用戶端時間戳記時，您才能搭配使用此WRITETIME函數，擷取儲存為資料行值中繼資料的時間戳記。SELECT如需詳細資訊，請參閱 用戶端時間時間。

```
SELECT WRITETIME(my_column) from my_table;
```

此WRITETIME函數無法用於多儲存格欄 (例如集合)。

> **Note**
>
> 為了與既定的 Cassandra 驅動程序行為兼容性，當您通過 Cassandra 驅動程序和開發人員工具使用 Cassandra 查詢語言（CQL）API 調用在系統表上執行操作時，基於標籤的授權策略不會強制執行。如需詳細資訊，請參閱 the section called “ 基於標籤的 Amazon Keyspaces 資源訪問”。

INSERT

使用該INSERT語句將行添加到表中。

語法

```
insert_statement ::=  INSERT INTO table_name ( names_values | json_clause )
                      [ IF NOT EXISTS ]
                      [ USING update_parameter ( AND update_parameter )* ]
names_values     ::=  names VALUES tuple_literal
json_clause      ::=  JSON string [ DEFAULT ( NULL | UNSET ) ]
names            ::=  '(' column_name ( ',' column_name )* ')'
```

範例

```
INSERT INTO "myGSGKeyspace".employees_tbl (id, name, project, region, division, role,
 pay_scale, vacation_hrs, manager_id)
VALUES ('012-34-5678','Russ','NightFlight','US','Engineering','IC',3,12.5,
 '234-56-7890') ;
```

更新參數

INSERT支援下列值update_parameter：

- TTL— 以秒為單位的時間值。最大可設定的值為 630,720,000 秒，相當於 20 年。
- TIMESTAMP— 代表自標準基準時間之後的微秒數bigint值，稱為epoch：1970 年 1 月 1 日格林威治標準時間 00:00:00。Amazon Keyspaces 間中的時間戳記必須介於過去的 2 天和 future 5 分鐘之間。

範例

```
INSERT INTO my_table (userid, time, subject, body, user)
        VALUES (B79CB3BA-745E-5D9A-8903-4A02327A7E09, 96a29100-5e25-11ec-90d7-
b5d91eceda0a, 'Message', 'Hello','205.212.123.123')
        USING TTL 259200;
```

支援

如需將 JSON 編碼資料類型對應至 Amazon Keyspaces 資料類型的表格，請參閱。the section called “Amazon Keyspaces 數據類型的 JSON 編碼”

您可以使用JSON關鍵字將JSON已編碼的對應作為單一列插入。對於存在於資料表中但在 JSON insert 陳述式中省略的資料行，請使用DEFAULT UNSET來保留現有的值。用於DEFAULT NULL將 NULL 值寫入每一列省略的資料欄，並覆寫現有的值 (需支付標準寫入費用)。 DEFAULT NULL為預設選項。

範例

```
INSERT INTO "myGSGKeyspace".employees_tbl JSON '{"id":"012-34-5678",
                                                 "name": "Russ",
                                                 "project": "NightFlight",
                                                 "region": "US",
                                                 "division": "Engineering",
                                                 "role": "IC",
                                                 "pay_scale": 3,
                                                 "vacation_hrs": 12.5,
                                                 "manager_id": "234-56-7890"}';
```

如果 JSON 數據包含重複的密鑰，Amazon 密 Keyspaces 存儲密鑰的最後一個值（類似於阿帕奇卡桑德拉）。在下列範例中，重複索引鍵所在id的位置會使234-56-7890用值。

範例

```
INSERT INTO "myGSGKeyspace".employees_tbl JSON '{"id":"012-34-5678",
                                                 "name": "Russ",
                                                 "project": "NightFlight",
                                                 "region": "US",
                                                 "division": "Engineering",
                                                 "role": "IC",
                                                 "pay_scale": 3,
                                                 "vacation_hrs": 12.5,
```

```
                                                    "id": "234-56-7890"}';
```

UPDATE

使用UPDATE陳述式來修改資料表中的資料列。

語法

```
update_statement ::=  UPDATE table_name
                      [ USING update_parameter ( AND update_parameter )* ]
                      SET assignment ( ',' assignment )*
                      WHERE where_clause
                      [ IF ( EXISTS | condition ( AND condition )*) ]
update_parameter ::=  ( integer | bind_marker )
assignment       ::=  simple_selection '=' term
                     | column_name '=' column_name ( '+' | '-' ) term
                     | column_name '=' list_literal '+' column_name
simple_selection ::=  column_name
                     | column_name '[' term ']'
                     | column_name '.' `field_name
condition        ::=  simple_selection operator term
```

範例

```
UPDATE "myGSGKeyspace".employees_tbl SET pay_scale = 5 WHERE id = '567-89-0123' AND
 division = 'Marketing' ;
```

若要遞增 acounter，請使用下列語法。如需詳細資訊，請參閱 the section called “櫃檯”。

```
UPDATE ActiveUsers SET counter = counter + 1  WHERE user = A70FE1C0-5408-4AE3-
BE34-8733E5K09F14 AND action = 'click';
```

更新參數

UPDATE支援下列值update_parameter：

- TTL— 以秒為單位的時間值。最大可設定的值為 630,720,000 秒，相當於 20 年。
- TIMESTAMP— 代表自標準基準時間之後的微秒數bigint值，稱為epoch：1970 年 1 月 1 日格林威治標準時間 00:00:00。Amazon Keyspaces 間中的時間戳記必須介於過去的 2 天和 future 5 分鐘之間。

範例

```
UPDATE my_table (userid, time, subject, body, user)
        VALUES (B79CB3BA-745E-5D9A-8903-4A02327A7E09, 96a29100-5e25-11ec-90d7-
b5d91eceda0a, 'Message', 'Hello again','205.212.123.123')
        USING TIMESTAMP '2022-11-03 13:30:54+0400';
```

DELETE

使用該DELETE語句從表中刪除一行。

語法

```
delete_statement ::=  DELETE [ simple_selection ( ',' simple_selection ) ]
                      FROM table_name
                      [ USING update_parameter ( AND update_parameter )* ]
                      WHERE where_clause
                      [ IF ( EXISTS | condition ( AND condition )*) ]

simple_selection ::=  column_name
                     | column_name '[' term ']'
                     | column_name '.' `field_name

condition        ::=  simple_selection operator term
```

其中：

- *table_name*是包含您要刪除之列的表格。

範例

```
DELETE manager_id FROM "myGSGKeyspace".employees_tbl WHERE id='789-01-2345' AND
 division='Executive' ;
```

DELETE支援下列值update_parameter：

- TIMESTAMP— 代表自標準基準時間之後的微秒數bigint值，稱為epoch：1970 年 1 月 1 日格林威治標準時間 00:00:00。

Amazon Keyspaces 中的內置函數

Amazon Keyspaces（對於 Apache 卡桑德拉）支持各種內置的功能，你可以在卡桑德拉查詢語言（CQL）語句中使用。

主題

純量函數

標量函數對單個值執行計算，並將結果作為單個值返回。Amazon Keyspaces 支援下列純量函數。

函式	描述
blobAsType	傳回指定資料類型的值。
cast	一種原生資料類型轉換成另一個原生資料類型。
currentDate	返回當前日期/時間作為日期。
currentTime	返回當前日期/時間作為時間。
currentTimestamp	返回當前日期/時間戳記。
currentTimeUUID	返回當前日期/時間作為timeuuid。
fromJson	將 JSON 字串轉換為所選資料行的資料類型。
maxTimeuuid	返回最大可能的時timeuuid間戳或日期字符串。
minTimeuuid	返回時間戳或日期字符串的最小值。timeuuid
now	返回一個新的唯一timeuuid。支援INSERT、UPDATE和DELETE陳述式，以及作為陳述式中WHERE子SELECT句的一部分。
toDate	將timeuuid或時間戳轉換為日期類型。

函式	描述
toJson	以 JSON 格式傳回所選欄的資料行值。
token	返回分區鍵的哈希值。
toTimestamp	將timeuuid或日期轉換為時間戳記。
TTL	傳回資料行的到期時間 (以秒為單位)。
typeAsBlob	將指定的資料類型轉換成blob。
toUnixTimestamp	將timeuuid或時間戳轉換為bigInt.
uuid	返回一個隨機版本 4 的 UUID。支援INSERT、UPDATE和DELETE陳述式，以及作為陳述式中WHERE子SELECT句的一部分。
writetime	返回指定列的值的時間戳。
dateOf	(已取代) 擷取 a 的時間戳記timeuuid，並以日期的形式傳回值。
unixTimestampOf	(已取代) 擷取 a 的時間戳記timeuuid，並以原始 64 位元整數時間戳記的形式傳回值。

配額 Amazon Keyspaces（阿帕奇卡桑德拉）

本節介紹 Amazon Keyspaces 的當前配額和默認值（適用於 Apache 卡桑德拉）。

主題

Amazon Keyspaces 服務配額

下表包含 Amazon Keyspaces（阿帕奇卡桑德拉）配額和默認值。您可以在 Service Quotas 主控台中取得有關可調整配額的資訊，您也可以在其中要求增加配額。如需配額的詳細資訊，請聯絡 AWS Support。

配額	描述	Amazon Keyspaces 默認
每個密鑰空間上限 AWS 區域	每個區域中此訂戶的金鑰空間數目上限。您可以在「Service Quotas」主控台中調整此預設值。	256
每個表格數上限 AWS 區域	每個區域中此訂閱者的所有金鑰空間表格數目上限。您可以在「Service Quotas」主控台中調整此預設值。	256
資料表結構定義大小	資料表結構定義的大小上限。	350 KB
最大並行 DDL 作業	每個區域允許此訂戶的並行 DDL 作業數目上限。	50
每個連線的最大查詢	每秒可由單一用戶端 TCP 連線處理的 CQL 查詢數目上限。	3000

配額	描述	Amazon Keyspaces 默認
最大列大小	資料列的大小上限，不包括靜態資料行資料。如需詳細資訊，請參閱 the section called “計算列大小”。	1 MB
INSERT和UPDATE陳述式中的最大欄數	CQL INSERT 或UPDATE語句中允許的最大列數。 關閉存留時間 (TTL) 時，INSERT或UPDATE陳述式最多可支援 225 個一般資料行。如果開啟 TTL，單一作業中最多可修改 166 個一般資料行。	225/166
每個邏輯分區的最大靜態數據	邏輯磁碟分割中靜態資料的最大彙總大小。如需詳細資訊，請參閱 the section called “計算每個邏輯分區的靜態列大小”。	1 MB
每INSELECT個陳述式的最大子查詢	您可以在SELECT陳述式中用於IN關鍵字的子查詢數目上限。您可以在「Service Quotas」主控台中調整此預設值。	100
每個巢狀凍結集合的最大數目 AWS 區域	當您針對具有集合資料類型的欄使用FROZEN關鍵字時，所支援的巢狀集合數目上限。如需凍結集合的詳細資訊，請參閱the section called “集合類型”。若要增加巢狀層級，請聯絡 AWS Support。	5

配額	描述	Amazon Keyspaces 默認
每秒最大讀取吞吐量	每秒最大讀取輸送量 (讀取要求單位 (RRU) 或讀取容量單位 (RCU) — 可配置給每個區域的資料表。您可以在「Service Quotas」主控台中調整此預設值。	40,000
每秒最大寫入輸送量	每個區域可配置給資料表的每秒最大寫入輸送量 — 寫入要求單位 (WRU) 或寫入容量單位 (WCU)。您可以在「Service Quotas」主控台中調整此預設值。	40,000
帳戶層級讀取輸送量 (已佈建)	為每個區域的帳戶配置的彙總讀取容量單位 (RCU) 數目上限。這僅適用於佈建的讀取/寫入容量模式下的表格。您可以在「Service Quotas」主控台中調整此預設值。	80,000
帳戶層級寫入輸送量 (已佈建)	針對每個區域的帳戶配置的彙總寫入容量單位 (WCU) 數目上限。這僅適用於佈建的讀取/寫入容量模式下的表格。您可以在「Service Quotas」主控台中調整此預設值。	80,000

配額	描述	Amazon Keyspaces 默認
每個帳戶每個區域的最大可擴展目標數	每個區域的帳戶可擴充目標數目上限。如果已啟用讀取容量的 auto 擴展，Amazon Keyspaces 表會計為一個可擴展目標；如果為寫入容量啟用了 auto 擴展，則會計為另一個可擴展目標。您可以選擇 Amazon Keyspaces 的可擴展目標，在應用程式自動擴展的 Ser vice Quotas 主控台中調整此預設值。	1,500
最大分割區金鑰大小	複合分割區索引鍵的大小上限。將最多 3 個位元組的額外儲存空間新增至中繼資料的分割索引鍵所包含的每個資料欄的原始大小。	2048 位元組
叢集金鑰大小上限	所有叢集資料行的最大組合大小。中繼資料的每個叢集資料行的原始大小最多可新增 4 個位元組的額外儲存空間。	850 個字節
使用 P oint-in-time 復原 (PITR) 的最大並行資料表還原	每位訂閱者使用 PITR 的並行資料表還原數目上限為 4。您可以在「Service Quotas」主控台中調整此預設值。	4
使用復原 (PITR) 還 point-in-time 原的最大資料量	可在 24 小時內使用 PITR 還原的資料大小上限。您可以在「Service Quotas」主控台中調整此預設值。	5 TB

提高或降低輸送量 (已佈建的資料表)

提高佈建輸送量

您可以通過使用控制台ReadCapacityUnits或WriteCapacityUnitsALTER TABLE語句增加或根據需要經常。新的設定值要在 ALTER TABLE 操作完成後才會生效。

新增佈建的容量時，不能超過每個帳戶的配額。而且，您可以根據需要增加表格的佈建容量。如需每個帳戶配額的詳細資訊，請參閱前一節 the section called “Amazon Keyspaces 服務配額”。

降低佈建輸送量

對於ALTER TABLE語句中的每個表，您可以減少ReadCapacityUnits或WriteCapacityUnits（或兩者）。新的設定值要在 ALTER TABLE 操作完成後才會生效。

每天可以隨時調降，最多四次。一天是根據國際標準時間 (UTC) 來定義。此外，如果過去一小時都未調降，可增加一次調降。這有效地使一天中的最大減少次數達到 27（第一個小時減少 4 個，每天後續的 1 小時窗口減少 1 個）。

Amazon 密 Keyspaces 靜態加密

您可以在 24 小時內變更 AWS 擁有 AWS KMS 金鑰與客戶管理 AWS KMS 金鑰之間的加密選項，最多可以在每個表格建立時間開始變更四次。如果過去 6 小時內沒有變化，則允許額外變更。這有效地使一天中的最大變更次數達到八個 (前六個小時的四個變更，以及每天後續六小時的每個時段變更一次)。

即使先前的配額已用盡，您也可以視需要變更加密選項以使用 AWS 擁有的 AWS KMS 金鑰。

除非您請求較高的數量，否則配額如下。若要要求增加服務配額，請參閱AWS Support。

Amazon Keyspaces 的文檔歷史記錄（阿帕奇卡桑德拉）

下表描述了自 Amazon Keyspaces 的最後一個版本以來的文檔的重要變化（Apache 卡桑德拉）。如需有關此文件更新的通知，您可以訂閱 RSS 訂閱源。

- 最新文件更新：2024 年 2 月 7 日

變更	描述	日期
從 Amazon 彈性 Kubernetes 服務 Connect 到亞馬遜 Keyspaces	現在，您可以按照 step-by-step 教程從 Amazon EKS 連接到 Amazon Keyspaces。	2024年2月7日
Amazon Keyspaces auto 擴展已佈建表格的 API	Amazon Keyspaces 現在提供CQL和 AWS API 支援，可用於使用佈建容量模式設定 auto 擴展。	2024 年 1 月 23 日
已佈建表格的 Amazon Keyspaces 多區域複寫支援	Amazon Keyspaces 現在支援多區域表的佈建容量模式。	2024 年 1 月 23 日
日誌中包含的 Amazon Keyspaces DML 活動 CloudTrail	您現在可以在中稽核 Amazon Keyspaces 資料操縱語言 (DML) API 呼叫。 AWS CloudTrail	2023 年 12 月 20 日
對於關鍵字 Amazon Keyspaces 支持 FROZEN	Amazon Keyspaces 現在支援收集資料類型的FROZEN關鍵字。	2023 年 11 月 15 日
Amazon Keyspaces 受管政策更新	Amazon Keyspaces 為AmazonKeyspacesFullAccess 受管政策新增了新的許可，允許透過界面 VPC 端點連線到 Amazon Keyspaces 的用戶端存取 Amazon EC2 執行個體，以使用 VPC 中的網路	2023 年 10 月 3 日

	資訊更新 Amazon Keyspaces `system.peers` 表格。	
Amazon Keyspaces 受管政策更新	Amazon Keyspaces 建立了新的`AmazonKeyspacesReadOnlyAccess_v2` 受管政策，允許用戶端透過界面 VPC 端點存取 Amazon EC2 執行個體連線到 Amazon 金鑰空間，以 VPC 中的網路資訊更新 Amazon 金鑰空間`system.peers` 表格。	2023 年 9 月 12 日
在 Amazon Keyspaces 中建立連線的最佳實務	了解如何改善和優化 Amazon Keyspaces 中的用戶端驅動程式組態。	2023 年 6 月 30 日
系統 Keyspaces 現在被記錄在 Amazon 密鑰空間	了解系統 Keyspaces 中存儲的內容，以及如何在 Amazon 密鑰空間中查詢它們以獲取有用的信息。	2023 年 6 月 21 日
Amazon Keyspaces 現在支援多區域複寫	Amazon Keyspaces space 多區域複寫可提供更好的容錯能力、穩定性和彈性，協助您維護全球分散式應用程式。	2023 年 6 月 5 日
Amazon Keyspaces 受管政策更新	Amazon Keyspaces 間為`AmazonKeyspacesFullAccess` 受管政策新增了許可，以允許 Amazon Keyspaces 間在管理員建立多區域金鑰空間時建立服務連結角色。	2023 年 6 月 5 日
對於關鍵字 Amazon Keyspaces 支持 IN	Amazon Keyspaces 現在支持`SELECT`語句中的`IN`關鍵字。	2023 年 4 月 25 日

Amazon Keyspaces 和虛擬私人雲端端點的跨帳戶存取	了解如何使用 VPC 端點為 Amazon Keyspaces 實作跨帳戶存取。	2023 年 4 月 20 日
Amazon Keyspaces 間支援用戶端時間戳記	Amazon Keyspaces space 用戶端時間戳記是與 Cassandra 相容的儲存格層級時間戳記，可協助分散式應用程式在不同用戶端對相同資料進行變更時判斷寫入操作的順序。	2023 年 3 月 14 日
開始使用 Amazon Keyspaces 和介面 VPC 端點	在本 step-by-step 教學中，了解如何從 VPC 連接到 Amazon Keyspaces。	2023 年 3 月 1 日
優化 Amazon Keyspaces 表的成本	提供最佳實務和指導，協助您識別針對現有 Amazon Keyspaces 表格成本最佳化的策略。	2023 年 2 月 17 日
現在Murmur3Partitioner 是預設值	現在Murmur3Partitioner 是 Amazon Keyspaces 中的默認分區程序。	2022 年 11 月 17 日
Amazon Keyspaces 現在支持Murmur3Partitioner	現在可Murmur3Partitioner 在 Amazon Keyspaces 中使用。	2022 年 11 月 9 日
空字符串和 blob 值的 Support 更新	Amazon Keyspaces 現在也支援空字串和 blob 值做為叢集資料行值。	2022 年 10 月 19 日

Amazon Keyspaces 現在可用 AWS GovCloud (US)	Amazon Keyspaces 現已在中提供， AWS GovCloud (US) Region 並且在 FedRAMP 高合規範圍內。如需可用端點的相關資訊，請參閱 AWS GovCloud (US) Region FIPS 端點。	2022 年 8 月 4 日
使用 Amazon 監控 Amazon Keyspaces 表存儲成本 CloudWatch	Amazon Keyspaces 間現在可協助您使用`BillableTableSizeInBytes` CloudWatch 指標監控和追蹤一段時間內的表格儲存成本。	2022 年 6 月 14 日
Amazon Keyspaces 現在支持地形	您現在可以使用地形表單在 Amazon Keyspaces 中執行資料定義語言 (DDL) 操作。	2022 年 6 月 9 日
Amazon Keyspaces `token` 功能支持	Amazon Keyspaces 現在可協助您使用`token`函數優化應用程式查詢。	2022 年 4 月 19 日
Amazon Keyspaces 與 Apache 星火集成	Amazon Keyspaces 現在可以幫助您通過使用開源的星火卡桑德拉連接器更輕鬆地讀取和寫入 Apache 星火數據。	2022 年 4 月 19 日
Amazon Keyspaces API 參考	Amazon Keyspaces 支援控制平面操作，以使用 AWS SDK 和. AWS CLI API 參考指南詳細說明支援的控制平面作業。	2022 年 3 月 2 日
如何疑難排解使用 Amazon Keyspaces 時的常見組態問題。	進一步了解如何解決使用 Amazon Keyspaces 時可能遇到的常見組態問題。	2021 年 11 月 22 日

Amazon Keyspaces 間支持存活時間（TTL）。	Amazon Keyspaces 間存留時間 (TTL) 可協助您簡化應用程式邏輯，並透過自動將資料表中的資料過期，優化儲存價格。	2021 年 10 月 18 日
使用 DSBulk 將數據遷移到 Amazon Keyspaces。	S tep-by-step 教程從阿帕奇卡桑德拉數據遷移到 Amazon Keyspaces 使用 DataStax 批量加載器（DSBulk）。	2021 年 8 月 9 日
Amazon Keyspaces 支援表格中的 `system.peers` VPC 端點項目。	Amazon Keyspaces 可讓您在`system.peers` 表格中填入可用的介面 VPC 端點資訊，以改善負載平衡並提高讀取/寫入輸送量。	2021 年 7 月 29 日
更新 IAM 受管政策以支援客戶受管 AWS KMS 金鑰。	Amazon 密 Keyspaces 的 IAM 受管政策現在包含列出和檢視存放在其中 AWS KMS的可用客戶受管 AWS KMS 金鑰的許可。	2021 年 6 月 1 日
Amazon 金 Keyspaces 支援客戶受管 AWS KMS 金鑰。	Amazon 金 Keyspaces 可讓您控制存放在其中的客戶受管 AWS KMS 金鑰，以 AWS KMS 進行靜態加密。	2021 年 6 月 1 日
Amazon Keyspaces 支持 JSON 語法	Amazon Keyspaces 可支援插入和選取作業的 JSON 語法，協助您更輕鬆地讀取和寫入 JSON 文件。	2021 年 1 月 21 日
Amazon Keyspaces 支持靜態列	Amazon Keyspaces 間現在可協助您使用靜態資料欄，有效率地在多個資料列之間更新和存放通用資料。	2020 年 11 月 9 日

GA 版本的 NoSQL 工作台支持 Amazon Keyspaces	NoSQL Workbench 是用戶端應用程式，可協助您更輕鬆地設計和視覺化 Amazon Keyspaces 的非關聯式資料模型。NoSQL 工作台用戶端可用於視窗、macOS 和 Linux。	2020 年 10 月 28 日
預覽版本的 NoSQL 工作台支持 Amazon Keyspaces	NoSQL Workbench 是用戶端應用程式，可協助您更輕鬆地設計和視覺化 Amazon Keyspaces 的非關聯式資料模型。NoSQL 工作台用戶端可用於視窗、macOS 和 Linux。	2020 年 10 月 5 日
程式化存取 Amazon Keyspaces 的新程式碼範例	我們將繼續新增程式碼範例，以程式設計方式存取 Amazon Keyspaces。樣品現在可用於 Java，Python，圍棋，C# 和 Perl 卡桑德拉驅動程序，支持阿帕奇卡桑德拉版本 3.11.2。	2020 年 7 月 17 日
Amazon Keyspaces point-in-time 恢復（PITR）	Amazon Keyspaces 現在提供 point-in-time 資料表資料的持續備份，可協助保護資料表免於意外寫入或刪除操作的情況。	2020 年 7 月 9 日
Amazon Keyspaces 式推出	與 Amazon Keyspaces 間，以前在預覽期間稱為 Amazon 託管 Apache 卡桑德拉服務（MCS），您可以使用卡桑德拉查詢語言（CQL）代碼，Apache 2.0 許可卡桑德拉驅動程序和開發人員工具，你已經使用今天。	2020 年 4 月 23 日

Amazon Keyspaces 自動擴展	Amazon Keyspaces (適用於 Apache Cassandra) 與「應用程式自動擴展」整合，可協助您透過自動調整輸送量容量，有效地為可變工作負載佈建輸送量容量，以回應實際應用程式流量。	2020 年 4 月 23 日
為 Amazon Keyspaces 介面虛擬私有雲端 (VPC) 端點	Amazon Keyspaces 間提供服務與 VPC 之間的私人通訊，因此網路流量不會離開 Amazon 網路。	2020 年 4 月 16 日
基於標籤的訪問策略	您現在可以在 IAM 政策中使用資源標籤來管理對 Amazon Keyspaces 的存取。	2020 年 4 月 8 日
計數器資料類型	Amazon Keyspaces 現在可協助您使用計數器協調欄值的增量和遞減。	2020 年 4 月 7 日
標記資源	Amazon Keyspaces 現在可讓您使用標籤來標記和分類資源。	2020 年 3 月 31 日
AWS CloudFormation 支持	Amazon Keyspaces 現在可協助您透過使 AWS CloudFormation用自動化資源的建立和管理。	2020 年 3 月 25 日

Support IAM 角色和政策以及 SIGv4 身份驗證	已新增有關如何使用 AWS Identity and Access Management (IAM) 管理 Amazon Keyspaces 的存取許可和實作安全政策，以及如何使用適用於 Cassandra DataStax Java 驅動程式的身份驗證外掛程式，透過 IAM 角色和聯合身分以程式設計方式存取 Amazon Keyspaces 的資訊。	2020 年 3 月 17 日
讀/寫容量模式	Amazon Keyspaces 現在支援兩種讀取/寫入輸送量容量模式。讀取/寫入容量模式可控制讀取和寫入輸送量的費用，以及管理表格輸送量容量的方式。	2020 年 2 月 20 日
初始版本	本文檔涵蓋了 Amazon Keyspaces 的初始版本（阿帕奇卡桑德拉）。	2019 年 12 月 3 日

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。