



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

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

Table of Contents

什麼是 Amazon Keyspaces ?	1
運作方式	2
高層級架構	2
Cassandra 資料模型	4
存取 Amazon Keyspaces	5
使用案例	6
什麼是 CQL ?	6
比較 Amazon Keyspaces 與 Cassandra	8
Apache Cassandra 的功能差異	9
Apache Cassandra APIs、操作和資料類型	9
非同步建立和刪除金鑰空間和資料表	9
身分驗證和授權	10
批次	10
變更資料擷取 (CDC)	10
叢集組態	10
連線	11
IN 關鍵字	11
FROZEN 集合	11
輕量型交易	12
Load balancing	12
分頁	12
分割區	12
備妥的陳述式	13
刪除範圍	13
系統表	13
時間戳記	14
使用者定義的類型 (UDTs)	14
支援的 Cassandra APIs、操作、函數和資料類型	14
Cassandra API 支援	15
Cassandra 控制平面 API 支援	16
Cassandra 資料平面 API 支援	17
Cassandra 函數支援	17
Cassandra 資料類型支援	18
支援的 Cassandra 一致性層級	19

寫入一致性層級	19
讀取一致性層級	20
不支援的一致性層級	21
遷移至 Amazon Keyspaces	22
從 Cassandra 遷移	23
相容性	23
預估定價	24
遷移策略	32
線上遷移	32
離線移轉	43
混合移轉	45
遷移工具	48
使用 cqlsh 載入資料	50
使用 DSBulk 載入資料	60
存取 Amazon Keyspaces	77
設定 AWS Identity and Access Management	77
註冊 AWS 帳戶	77
建立具有管理存取權的使用者	77
設定 Amazon Keyspaces	78
使用主控台	79
CloudShell 主控台整合	80
使用 AWS CloudShell	81
AWS CloudShell 整合選項	81
先決條件	82
入門 AWS CloudShell	82
取得的 IAM 許可 AWS CloudShell	82
安裝 cqlsh 用戶端	83
設定 TLS 憑證	83
連線至 Amazon Keyspaces	83
搭配 Amazon Keyspaces 使用 CQLSH	84
在中保留資料 AWS CloudShell	84
最佳實務	84
疑難排解	84
範例：如何使用與 Amazon Keyspaces 互動 AWS CloudShell	85
建立程式設計存取憑證	86
建立服務特定的登入資料	87

建立 IAM 登入資料以進行 AWS 身分驗證	89
服務端點	96
連接埠和通訊協定	96
全域端點	97
AWS GovCloud (US) Region FIPS 端點	101
中國區域端點	102
串流端點	102
連線至雙堆疊端點	104
Amazon Keyspaces 中的 IPv6 支援	105
Amazon Keyspaces 中的 DNS 解析	105
使用 cqlsh	106
使用 cqlsh-expansion	107
如何手動設定 TLS 的 cqlsh 連線	111
使用 AWS CLI	113
下載和設定 AWS CLI	113
AWS CLI 搭配 Amazon Keyspaces 使用	113
使用 API	117
使用 Cassandra 用戶端驅動程式	118
使用 Cassandra Java 用戶端驅動程式	119
使用 Cassandra Python 用戶端驅動程式	135
使用 Cassandra Node.js 用戶端驅動程式	138
使用 Cassandra .NET Core 用戶端驅動程式	142
使用 Cassandra Go 用戶端驅動程式	144
使用 Cassandra Perl 用戶端驅動程式	149
設定跨帳戶存取	151
在共用 VPC 中設定跨帳戶存取	151
設定沒有共用 VPC 的跨帳戶存取	155
開始使用	160
先決條件	161
建立金鑰空間	161
檢查金鑰空間建立狀態	164
建立資料表	165
檢查資料表建立狀態	173
CRUD 操作	174
建立	174
讀取	178

更新	182
刪除	183
刪除資料表	185
刪除金鑰空間	187
教學課程和解決方案	191
與 VPC 端點連線	191
先決條件	192
步驟 1：啟動 Amazon EC2 執行個體	192
步驟 2：設定您的 Amazon EC2 執行個體	194
步驟 3：建立 Amazon Keyspaces 的 VPC 端點	196
步驟 4：設定 VPC 端點連線的許可	202
步驟 5：設定監控	205
步驟 6：(選用) 連線的最佳實務	206
步驟 7：(選用) 清除	208
與 Apache Spark 整合	210
先決條件	210
步驟 1：設定 Amazon Keyspaces	211
步驟 2：設定 Apache Cassandra Spark 連接器	212
步驟 3：建立應用程式組態檔案	214
步驟 4：準備來源資料和目標資料表	216
步驟 5：寫入和讀取 Amazon Keyspaces 資料	218
疑難排解	221
從 Amazon EKS 連線	222
先決條件	222
步驟 1：設定 Amazon EKS 叢集	225
步驟 2：設定應用程式	229
步驟 3：建立應用程式映像	232
步驟 4：將應用程式部署至 Amazon EKS	233
步驟 5：(選用) 清除	238
將資料匯出至 Amazon S3	241
先決條件	241
步驟 1：建立 Amazon S3 儲存貯體、下載工具和設定環境	243
步驟 2：設定 AWS Glue 任務	245
步驟 3：從執行匯出 AWS Glue 任務 AWS CLI	247
步驟 4：(選用) 排程匯出任務	249
步驟 5：(選用) 清除	250

管理無伺服器資源	252
估算資料列大小	253
估算欄的編碼大小	253
根據資料類型估計資料值的編碼大小	254
考慮 Amazon Keyspaces 功能對資料列大小的影響	255
選擇正確的公式來計算資料列的編碼大小	256
資料列大小計算範例	256
估計容量耗用量	258
估計範圍查詢的容量耗用	259
預估限制查詢的讀取容量耗用量	259
估算資料表掃描的讀取容量耗用量	260
LWT 的預估容量使用量	261
估算靜態資料欄的容量耗用量	261
多區域資料表的預估容量	265
使用 CloudWatch 估計容量使用量	266
設定讀取/寫入容量模式	266
設定隨需容量模式	267
設定佈建容量模式	269
檢視資料表的容量模式	270
變更容量模式	272
設定預熱前	274
使用自動擴展管理輸送量容量	287
Amazon Keyspaces 自動擴展的運作方式	287
自動擴展如何適用於多區域資料表	289
使用須知	289
設定和更新自動擴展政策	290
使用高載容量	302
使用 Amazon Keyspaces 功能	303
系統金鑰空間	304
system	304
system_schema	306
system_schema_mcs	307
system_multiregion_info	309
使用者定義的類型 (UDTs)	311
設定 許可	312
建立 UDT	315

檢視 UDTs	319
刪除 UDT	322
使用 CQL 查詢	323
使用 IN SELECT	324
使用批次陳述式	327
訂單結果	333
分頁結果	334
使用 CDC 串流	334
CDC 串流的運作方式	335
使用 CDC 串流	338
使用分割區	370
變更分割區	371
用戶端時間戳記	372
與 AWS 服務整合	373
使用用戶端時間戳記建立資料表	373
設定用戶端時間戳記	376
在查詢中使用用戶端時間戳記	379
多區域複製	380
優勢	380
容量模式和定價	381
運作方式	382
使用須知	386
設定多區域複寫	387
使用point-in-time復原進行備份和還原	412
運作方式	413
使用point-in-time復原	417
使用存留時間過期資料	429
與 AWS 服務的整合	430
使用預設 TTL 值建立資料表	431
更新資料表預設 TTL 值	434
使用自訂 TTL 建立資料表	437
更新資料表自訂 TTL	440
使用 INSERT 設定新資料列的自訂 TTL	442
使用 UPDATE 設定資料列和資料欄的自訂 TTL	442
使用 AWS SDKs	444
使用標籤	445

標記限制	445
標記鍵空間、資料表和串流	446
建立成本分配報告	464
建立 CloudFormation 資源	465
Amazon Keyspaces 和 CloudFormation 範本	465
進一步了解 CloudFormation	465
NoSQL Workbench	466
下載	466
開始使用	467
視覺化資料模型	469
建立資料模型	473
編輯資料模型	475
遞交資料模型	477
範例資料模型	486
版本歷史記錄	487
程式碼範例	488
基本概念	488
Hello Amazon Keyspaces	489
了解基本概念	494
動作	556
程式庫和工具	608
程式庫和範例	608
Amazon Keyspaces (適用於 Apache Cassandra) 開發人員工具組	608
Amazon Keyspaces (適用於 Apache Cassandra) 範例	608
AWS Signature 第 4 版 (SigV4) 身分驗證外掛程式	608
反白的範例和開發人員工具庫	609
Amazon Keyspaces 通訊協定緩衝區	609
CloudFormation 為 Amazon Keyspaces (適用於 Apache Cassandra) 指標建立 Amazon CloudWatch 儀表板的 範本	609
搭配使用 Amazon Keyspaces (適用於 Apache Cassandra) AWS Lambda	609
搭配 Spring 使用 Amazon Keyspaces (適用於 Apache Cassandra)	610
搭配 Scala 使用 Amazon Keyspaces (適用於 Apache Cassandra)	610
搭配使用 Amazon Keyspaces (適用於 Apache Cassandra) AWS Glue	610
要 CloudFormation 轉換的 Amazon Keyspaces (適用於 Apache Cassandra) Cassandra 查 詢語言 (CQL)	610

適用於 Java 的 Apache Cassandra 驅動程式的 Amazon Keyspaces (適用於 Apache Cassandra) 協助程式	610
Amazon Keyspaces (適用於 Apache Cassandra) 快照壓縮示範	611
Amazon Keyspaces (適用於 Apache Cassandra) 和 Amazon S3 轉碼器示範	611
最佳實務	612
NoSQL 設計	613
NoSQL 與 RDBMS 對比	614
兩個主要概念	614
一般方法	614
連線	615
它們的運作方式	616
如何設定連線	616
如何設定重試政策	618
VPC 端點連線	618
如何監控連線	619
如何處理連線錯誤	620
建立資料模型	620
分割區索引鍵設計	621
成本最佳化	623
在資料表層級評估您的成本	624
評估資料表的容量模式	625
評估資料表的 Application Auto Scaling 設定	629
識別未使用的資源	636
評估您的資料表用量模式	640
評估您是否具有適當大小的佈建容量	641
疑難排解	650
一般錯誤	651
一般錯誤	651
連線錯誤	652
連線至 Amazon Keyspaces 端點時發生錯誤	652
容量管理錯誤	663
無伺服器容量錯誤	663
資料定義語言錯誤	667
資料定義語言錯誤	667
監控 Amazon Keyspaces	672
使用 CloudWatch 進行監控	673

使用指標	673
指標與維度	675
檢視 指標	693
建立 警示	693
使用 CloudTrail 進行記錄	694
在 CloudTrail 中設定日誌檔案項目	694
CloudTrail 中的 DDL 資訊	695
CloudTrail 中的 DML 資訊	696
了解 日誌檔案項目	697
安全	708
資料保護	708
靜態加密	709
傳輸中加密	728
網際網路流量隱私權	728
AWS Identity and Access Management	729
目標對象	730
使用身分驗證	730
使用政策管理存取權	731
Amazon Keyspaces 如何與 IAM 搭配使用	732
身分型政策範例	737
AWS 受管政策	744
疑難排解	753
使用服務連結角色	756
法規遵循驗證	764
恢復能力	764
基礎架構安全	765
使用 介面 VPC 端點	766
搭配串流使用介面 VPC 端點	772
Amazon Keyspaces 的組態和漏洞分析	775
安全最佳實務	775
預防性安全最佳實務	775
偵測性安全最佳實務	777
CQL 語言參考	779
語言元素	779
識別碼	780
常數	780

條款	780
資料類型	780
Amazon Keyspaces 資料類型的 JSON 編碼	784
DDL 陳述式	787
Keyspaces	788
表格	791
類型	803
DML 陳述式	805
SELECT	806
INSERT	808
UPDATE	810
DELETE	811
內建函數	812
純量函數	812
配額	814
Amazon Keyspaces 服務配額	814
提高或降低輸送量 (已佈建的資料表)	819
提高佈建輸送量	819
降低佈建輸送量	819
Amazon Keyspaces 靜態加密	819
Amazon Keyspaces 中使用者定義類型 (UDTs) 的配額和預設值	819
Amazon Keyspaces UDT 配額和預設值	819
文件歷史紀錄	821
.....	dccccxxii

什麼是 Amazon Keyspaces (適用於 Apache Cassandra) ?

Amazon Keyspaces (適用於 Apache Cassandra) 是一種可擴展、高可用性且受管的 Apache Cassandra 相容資料庫服務。使用 Amazon Keyspaces，您不需要佈建、修補或管理伺服器，也不需要安裝、維護或操作軟體。

Amazon Keyspaces 是無伺服器，因此您只需支付使用的資源，服務會自動擴展和縮減資料表，以回應應用程式流量。您可以建置應用程式，透過幾乎無限制的輸送量和儲存，每秒為數千個請求提供服務。

Note

Apache Cassandra 是開放原始碼的廣欄資料存放區，旨在處理大量資料。如需詳細資訊，請參閱 [Apache Cassandra](#)。

Amazon Keyspaces 可讓您在 中輕鬆遷移、執行和擴展 Cassandra 工作負載 AWS 雲端。只要按幾下 AWS 管理主控台或幾行程式碼，您就可以在 Amazon Keyspaces 中建立金鑰空間和資料表，而無需部署任何基礎設施或安裝軟體。

使用 Amazon Keyspaces，您可以使用您目前使用的 AWS 相同 Cassandra 應用程式程式碼和開發人員工具，在 上執行現有的 Cassandra 工作負載。

透過 Github 上提供的 [Amazon Keyspaces \(適用於 Apache Cassandra\) 定價計算器](#)，您可以根據現有的 Apache Cassandra 工作負載預估 Amazon Keyspaces 的每月成本。輸入來自 Cassandra Nodetool 狀態輸出的指標和 Amazon Keyspaces 的預期無伺服器組態，以比較兩個解決方案之間的直接成本。請注意，相較於現有的 Cassandra 部署，此計算器僅著重於 Amazon Keyspaces 的操作成本。它不包括基礎設施維護、營運開銷或 Cassandra 支援成本等總體擁有成本 (TCO) 因素。

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：運作方式](#)
- [Amazon Keyspaces 使用案例](#)
- [什麼是 Cassandra 查詢語言 \(CQL\) ?](#)

Amazon Keyspaces：運作方式

Amazon Keyspaces 消除管理 Cassandra 的管理開銷。若要了解原因，從 Cassandra 架構開始，然後將其與 Amazon Keyspaces 進行比較會很有幫助。

主題

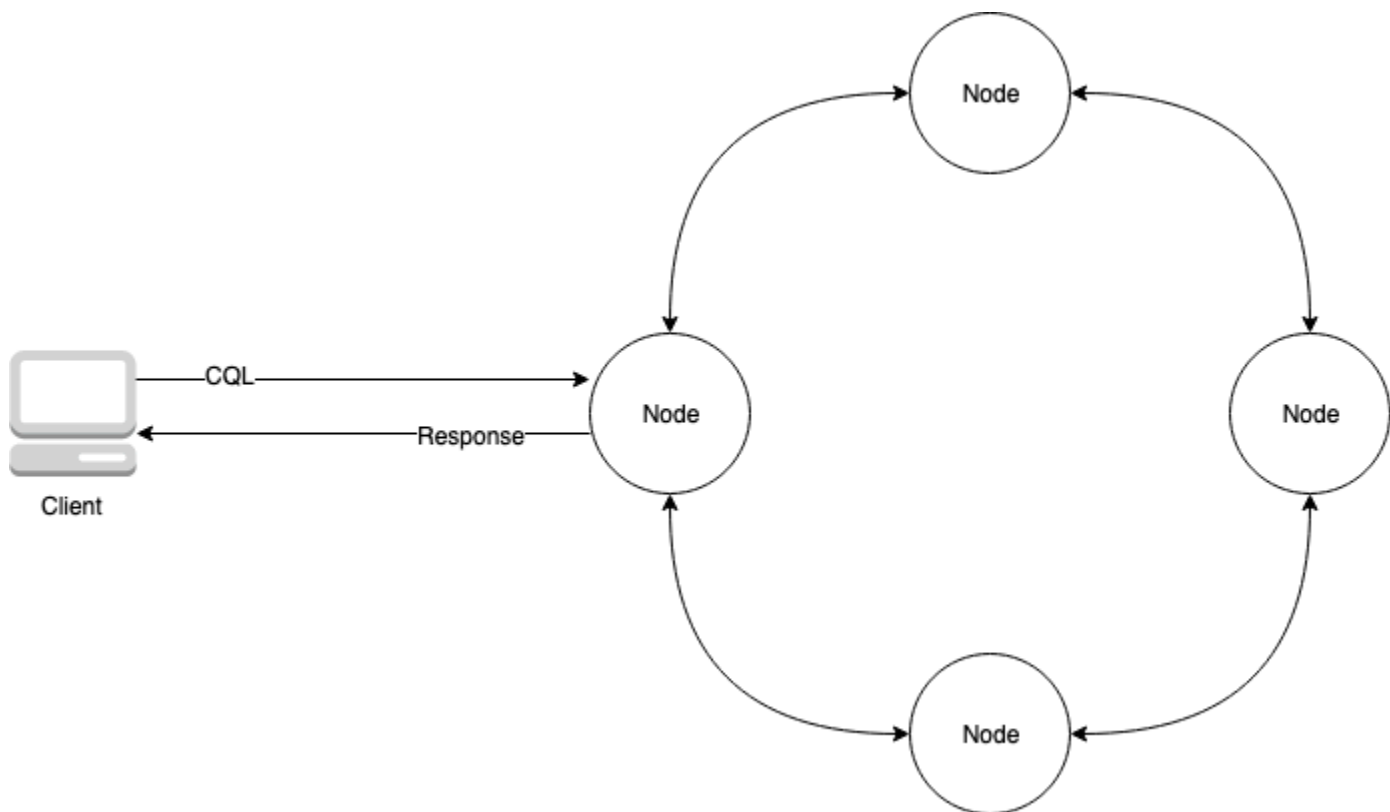
- [高階架構：Apache Cassandra 與 Amazon Keyspaces](#)
- [Cassandra 資料模型](#)
- [從應用程式存取 Amazon Keyspaces](#)

高階架構：Apache Cassandra 與 Amazon Keyspaces

傳統 Apache Cassandra 部署在由一或多個節點組成的叢集中。您負責管理每個節點，並在叢集擴展時新增和移除節點。

用戶端程式透過連線至其中一個節點並發出 Cassandra 查詢語言 (CQL) 陳述式來存取 Cassandra。CQL 類似於 SQL，這是關聯式資料庫中使用的常用語言。即使 Cassandra 不是關聯式資料庫，CQL 也提供熟悉的界面來查詢和操作 Cassandra 中的資料。

下圖顯示簡單的 Apache Cassandra 叢集，由四個節點組成。



生產 Cassandra 部署可能包含數百個節點，在一個或多個實體資料中心的數百部實體電腦上執行。這可能會對需要佈建、修補和管理伺服器的應用程式開發人員造成操作負擔，以及安裝、維護和操作軟體。

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。您可以選擇資料表的輸送量容量模式，根據工作負載的可預測性和可變性來最佳化讀取和寫入的價格。

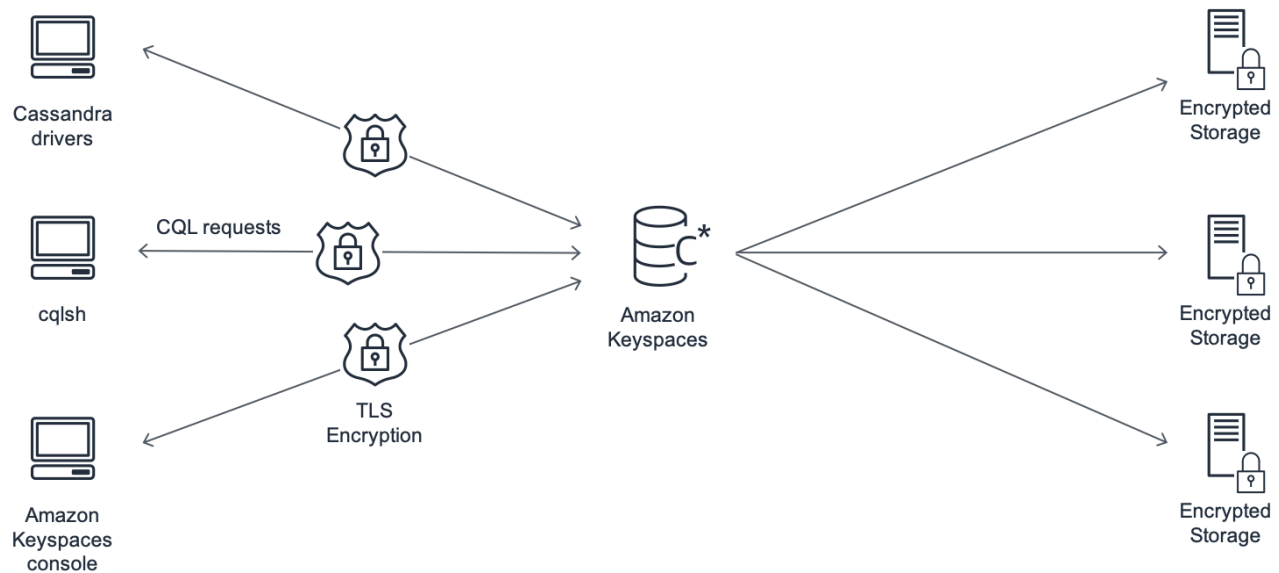
使用隨需模式時，您只需支付應用程式實際執行的讀取和寫入費用。您不需要事先指定資料表的輸送量容量。Amazon Keyspaces 會在應用程式流量上升或下降時，幾乎立即容納您的應用程式流量，使其成為具有不可預測流量之應用程式的理想選擇。

如果您有可預測的應用程式流量，佈建的容量模式可協助您最佳化輸送量價格，並可事先預測資料表的容量需求。使用佈建容量模式時，您可以指定您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](#)，自動增加和減少資料表的佈建容量。

當您進一步了解工作負載的流量模式時，或如果您預期流量暴增，例如您預期會驅動大量資料表流量的主要事件，您可以每天變更一次資料表的容量模式。如需讀取和寫入容量佈建的詳細資訊，請參閱[the section called “設定讀取/寫入容量模式”](#)。

Amazon Keyspaces (適用於 Apache Cassandra) 會將資料的三個副本存放在多個[可用區域中](#)，以提供耐用性和高可用性。此外，您還可以受益於資料中心和網路架構，該架構專為滿足最安全敏感組織的需求而建置。當您建立新的 Amazon Keyspaces 資料表且所有用戶端連線都需要 Transport Layer Security (TLS) 時，會自動啟用靜態加密。其他 AWS 安全功能包括[監控 AWS Identity and Access Management](#)、和[虛擬私有雲端 \(VPC\) 端點](#)。如需所有可用安全功能的概觀，請參閱[安全](#)。

下圖顯示 Amazon Keyspaces 的架構。



用戶端程式透過連線至預定端點（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”](#)。

Cassandra 資料模型

如何為業務案例建立資料模型，對於從 Amazon Keyspaces 實現最佳效能至關重要。不良的資料模型可能會大幅降低效能。

即使 CQL 看起來與 SQL 類似，Cassandra 和關聯式資料庫的後端也非常不同，必須以不同的方式處理。以下是一些較重要的問題需要考慮：

儲存

您可以在資料表中視覺化 Cassandra 資料，其中每一列代表記錄，每一欄都代表該記錄內的欄位。

資料表設計：先查詢

CQL JOIN 中沒有。因此，您應該設計具有資料形狀的資料表，以及如何針對業務使用案例存取資料表。這可能會導致重複資料的去標準化。您應該專門針對特定存取模式設計每個資料表。

分區

您的資料會存放在磁碟的分割區中。資料存放於 中的分割區數量，以及其在分割區間的分佈方式，取決於您的分割區索引鍵。如何定義分割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 Cassandra 中，資料會儲存為索引鍵/值對。每個 Cassandra 資料表都必須有一個主索引鍵，這是資料表中每一列的唯一索引鍵。主索引鍵是必要分割區索引鍵和選用叢集資料欄的複合。包含主索引鍵的資料在資料表中的所有記錄中必須是唯一的。

- 分割區索引鍵 – 需要主索引鍵的分割區索引鍵部分，並決定資料存放於叢集的哪個分割區。分割區索引鍵可以是單一資料欄，也可以是由兩個或多個資料欄組成的複合值。如果單一資料欄分割區索引鍵會導致單一分割區或具有大部分資料的極少分割區，因此包含大部分磁碟 I/O 操作，則您會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欄部分會決定如何在每個分割區中叢集和排序資料。如果您在主索引鍵中包含叢集資料欄，叢集資料欄可以有一或多個資料欄。如果叢集資料欄中有多個資料欄，排序順序取決於從左到右在叢集資料欄中列出的資料欄順序。

如需 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 (適用於 Apache Cassandra) 實作 Apache Cassandra 查詢語言 (CQL) API，因此您可以使用您已使用的 CQL 和 Cassandra 驅動程式。更新您的應用程式就像更新您的 Cassandra 驅動程式或 `cqlsh` 組態以指向 Amazon Keyspaces 服務端點一樣簡單。如需所需憑證的詳細資訊，請參閱 [the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

Note

為了協助您開始使用，您可以在 [GitHub](#) 上的 Amazon Keyspaces 程式碼範例儲存庫中使用各種 Cassandra 用戶端驅動程式，尋找連接至 Amazon Keyspaces 的 end-to-end 程式碼範例。

請考慮下列 Python 程式，其會連線至 Cassandra 叢集並查詢資料表。

```
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)
```

若要針對 Amazon Keyspaces 執行相同的程式，您需要：

- 新增叢集端點和連接埠：例如，主機可以替換為服務端點，例如 `cassandra.us-east-1.amazonaws.com` 和連接埠號碼為：9142。
- 新增 TLS/SSL 組態：如需使用 Cassandra 用戶端 Python 驅動程式新增 TLS/SSL 組態以連線至 Amazon Keyspaces 的詳細資訊，請參閱 [使用 Cassandra Python 用戶端驅動程式以程式設計方式存取 Amazon Keyspaces](#)。

Amazon Keyspaces 使用案例

以下是您可以使用 Amazon Keyspaces 的一些方式：

- 建置需要低延遲的應用程式 – 以高速處理需要 single-digit-millisecond 延遲的應用程式資料，例如工業設備維護、交易監控、機群管理和路由最佳化。
- 使用開放原始碼技術建置應用程式：AWS 使用適用於多種程式設計語言的開放原始碼 Cassandra APIs 和驅動程式，在上建置應用程式，例如 Java、Python、Ruby、Microsoft .NET、Node.js、PHP、C++、Perl 和 Go。如需程式碼範例，請參閱 [程式庫和工具](#)。
- 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 資料表，AWS 雲端而無需管理基礎設施。如需詳細資訊，請參閱 [管理無伺服器資源](#)。

什麼是 Cassandra 查詢語言 (CQL)？

Cassandra 查詢語言 (CQL) 是與 Apache Cassandra 通訊的主要語言。Amazon Keyspaces (適用於 Apache Cassandra) 與 CQL 3.x API (與 2.x 版回溯相容) 相容。

在 CQL 中，資料存放在資料表、資料欄和資料列中。在此意義上，CQL 類似於結構化查詢語言 (SQL)。這些是 CQL 中的重要概念。

- CQL 元素 – CQL 的基本元素是識別符、常數、術語和資料類型。
- 資料定義語言 (DDL) – DDL 陳述式用於管理資料結構，例如 Amazon Keyspaces 中的 AWS 資源的金鑰空間和資料表。DDL 陳述式是其中的控制平面操作 AWS。
- 資料處理語言 (DML) – DML 陳述式用於管理資料表中的資料。DML 陳述式用於選取、插入、更新和刪除資料。這些是 中的資料平面操作 AWS。
- 內建函數 – Amazon Keyspaces 支援各種內建純量函數，可用於 CQL 陳述式。

如需 CQL 的詳細資訊，請參閱 [Amazon Keyspaces 的 CQL 語言參考 \(適用於 Apache Cassandra\)](#)。
如需 Apache Cassandra 的功能差異，請參閱 [the section called “Apache Cassandra 的功能差異”](#)。

若要執行 CQL 查詢，您可以執行下列其中一項操作：

- 使用 中的 CQL 編輯器 AWS 管理主控台。
- 使用 AWS CloudShell 和 [cqlsh-expansion](#)。
- 使用 cqlsh 用戶端。
- 使用 Apache 2.0 授權的 Cassandra 用戶端驅動程式。

除了 CQL 之外，您還可以使用 AWS SDKs 和 在 Amazon Keyspaces 中執行資料定義語言 (DDL) 操作 AWS Command Line Interface。

如需使用這些方法來存取 Amazon Keyspaces 的詳細資訊，請參閱 [存取 Amazon Keyspaces \(適用於 Apache Cassandra\)](#)。

Amazon Keyspaces (適用於 Apache Cassandra) 與 Apache Cassandra 相比如何？

若要建立與 Amazon Keyspaces 的連線，您可以在 Amazon Virtual Private Cloud 中使用公有[AWS 服務端點](#)或使用[介面 VPC 端點 \(AWS PrivateLink\)](#) 的私有端點。[Amazon Virtual Private Cloud](#) 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[公有端點](#)建立的連線。在此情況下，Amazon Keyspaces 會顯示為用戶端的九節點 Apache Cassandra 3.11.2 叢集。

介面 VPC 端點連線

這是使用[介面 VPC 端點](#)建立的私有連線。在此情況下，Amazon Keyspaces 會顯示為用戶端的三節點 Apache Cassandra 3.11.2 叢集。

Amazon Keyspaces 提供幾乎無限的輸送量和儲存，與用戶端可見的連線類型和節點數量無關。為此，Amazon Keyspaces 會將節點映射到負載平衡器，將查詢路由到許多基礎儲存分割區之一。如需連線的相關資訊，請參閱[the section called “它們的運作方式”](#)。

Amazon Keyspaces 會將資料存放在分割區中。分割區是資料表的儲存配置，由固態硬碟 (SSDs) 提供支援。Amazon Keyspaces 會自動跨 內的多個[可用區域](#)複寫您的資料，AWS 區域 以實現耐用性和高可用性。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，並自動佈建所需的額外分割區。

Amazon Keyspaces 支援所有常用的 Cassandra 資料平面操作，例如建立金鑰空間和資料表、讀取資料和寫入資料。Amazon Keyspaces 是[無伺服器](#)，因此您不需要佈建、修補或管理伺服器。您不需要安裝、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 會自動設定複寫因素和一致性層級等設定，為您提供高可用性、耐用性和single-digit-millisecond效能。如需更多彈性和低延遲本機讀取，Amazon Keyspaces 提供[多區域複寫](#)。

主題

- [功能差異：Amazon Keyspaces 與 Apache Cassandra](#)
- [支援的 Cassandra APIs、操作、函數和資料類型](#)
- [支援的 Apache Cassandra 讀取和寫入一致性層級和相關成本](#)

功能差異：Amazon Keyspaces 與 Apache Cassandra

以下是 Amazon Keyspaces 和 Apache Cassandra 之間的功能差異。

主題

- [Apache Cassandra APIs、操作和資料類型](#)
- [非同步建立和刪除金鑰空間和資料表](#)
- [身分驗證和授權](#)
- [批次](#)
- [變更資料擷取 \(CDC\)](#)
- [叢集組態](#)
- [連線](#)
- [IN 關鍵字](#)
- [FROZEN 集合](#)
- [輕量型交易](#)
- [Load balancing](#)
- [分頁](#)
- [分割區](#)
- [備妥的陳述式](#)
- [刪除範圍](#)
- [系統表](#)
- [時間戳記](#)
- [使用者定義的類型 \(UDTs\)](#)

Apache Cassandra APIs、操作和資料類型

Amazon Keyspaces 支援所有常用的 Cassandra 資料平面操作，例如建立金鑰空間和資料表、讀取資料和寫入資料。若要查看目前支援的項目，請參閱[支援的 Cassandra APIs、操作、函數和資料類型](#)。

非同步建立和刪除金鑰空間和資料表

Amazon Keyspaces 會執行資料定義語言 (DDL) 操作，例如非同步建立和刪除金鑰空間、資料表和類型。若要了解如何監控資源的建立狀態，請參閱 [the section called “檢查金鑰空間建立狀態”](#)和 [the](#)

section called “[檢查資料表建立狀態](#)”。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”](#)。

身分驗證和授權

Amazon Keyspaces (適用於 Apache Cassandra) 使用 AWS Identity and Access Management (IAM) 進行使用者身分驗證和授權，並支援與 Apache Cassandra 同等的授權政策。因此，Amazon Keyspaces 不支援 Apache Cassandra 的安全組態命令。

批次

Amazon Keyspaces 支援下列批次命令：

- 已記錄的批次 – 每個批次最多支援 100 個命令。
- 未記錄的批次 – 每個批次最多支援 30 個命令。

批次中只允許 INSERT、UPDATE 或 DELETE 命令。

變更資料擷取 (CDC)

在 Apache Cassandra 中，變更資料擷取 (CDC) 複製遞交日誌區段，以擷取節點上指定目錄的資料表更新。CDC 日誌中的每個變更記錄可以包含跨多個金鑰空間和資料表的更新。更新或刪除資料列時，Apache Cassandra CDC 日誌只會顯示修改過的資料欄。日誌不包含變更之前的先前資料狀態。若要了解資料修改的完整詳細資訊，您需要使用特定的 Cassandra 工具還原序列化這些日誌。

使用 Amazon Keyspaces CDC 串流，您可以選取 CDC 串流為每個資料列收集的資訊類型。根據預設，Amazon Keyspaces CDC 串流會在變更發生前後擷取資料列的版本。Amazon Keyspaces CDC 在資料表層級提供重複和排序的變更記錄。

在 Amazon Keyspaces 中，每個 CDC 串流都是具有 Amazon Resource Name (ARN) AWS 的資源，您的應用程式可以使用其中一個可用的[端點](#)，使用 Amazon Keyspaces CDC API 來使用 Amazon Keyspaces CDC 串流。

如需 Amazon Keyspaces CDC 的詳細資訊，請參閱 [the section called “使用 CDC 串流”](#)。

叢集組態

Amazon Keyspaces 是無伺服器，因此不需要設定叢集、主機或 Java 虛擬機器 (JVMs)。Cassandra 的壓縮、壓縮、快取、垃圾收集和花粉篩選設定不適用於 Amazon Keyspaces，如果指定，則會予以忽略。

連線

您可以使用現有的 Cassandra 驅動程式與 Amazon Keyspaces 通訊，但您需要以不同的方式設定驅動程式。Amazon Keyspaces 每秒支援每個 TCP 連線最多 3,000 個 CQL 查詢，但驅動程式可以建立的連線數目沒有限制。

大多數開放原始碼 Cassandra 驅動程式會建立連至 Cassandra 的連線集區，並透過該連線集區進行負載平衡查詢。Amazon Keyspaces 向驅動程式公開 9 個對等 IP 地址，大多數驅動程式的預設行為是與每個對等 IP 地址建立單一連線。因此，使用預設設定的驅動程式 CQL 查詢輸送量上限為每秒 27,000 個 CQL 查詢。

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

最佳實務是，建議您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分佈。在此案例中，規劃每秒 18,000 個 CQL 查詢需要 36 個連線。將驅動程式設定為跨 9 個端點的 4 個連線，可提供每秒執行 500 個請求的 36 個連線。如需連線最佳實務的詳細資訊，請參閱 [the section called “連線”](#)。

與 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。如需 VPC 連線最佳實務的詳細資訊，請參閱 [the section called “VPC 端點連線”](#)。

IN 關鍵字

Amazon Keyspaces 支援 SELECT 陳述式中的 IN 關鍵字。UPDATE 和 IN 不支援 DELETE。在 SELECT 陳述式中使用 IN 關鍵字時，查詢的結果會依照在 SELECT 陳述式中顯示金鑰的順序傳回。在 Cassandra 中，結果會以詞典方式排序。

使用時 ORDER BY，不支援使用已停用分頁進行完整重新排序，並在頁面中排序結果。IN 關鍵字不支援配量查詢。IN 關鍵字 TOKENS 不支援。Amazon Keyspaces 會透過建立子查詢來使用 IN 關鍵字處理查詢。每個子查詢都算作每秒每個 TCP 連線限制 3,000 個 CQL 查詢的連線。如需詳細資訊，請參閱 [the section called “使用 IN SELECT”](#)。

FROZEN 集合

Cassandra 中的 FROZEN 關鍵字會將集合資料類型的多個元件序列化為視為的單一不可變值 BLOB。INSERT 和 UPDATE 陳述式會覆寫整個集合。

根據預設，Amazon Keyspaces 最多支援 8 個層級的凍結集合巢狀化。如需詳細資訊，請參閱 [the section called “Amazon Keyspaces 服務配額”](#)。

Amazon Keyspaces 不支援在條件式UPDATE或SELECT陳述式中使用整個凍結集合的不等式比較。Amazon Keyspaces 中集合和凍結集合的行為相同。

當您使用凍結集合搭配用戶端時間戳記時，如果寫入操作的時間戳記與未過期或已淘汰的現有資料欄的時間戳記相同，Amazon Keyspaces 不會執行比較。相反地，它可讓伺服器判斷最新的寫入器，而最新的寫入器獲勝。

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 (適用於 Apache Cassandra) 完全支援在 INSERT、UPDATE 和 DELETE 命令上比較和設定功能，這些命令在 Apache Cassandra 中稱為輕量型交易 (LWTs)。作為無伺服器產品，Amazon Keyspaces (適用於 Apache Cassandra) 可在任何規模提供一致的效能，包括輕量型交易。使用 Amazon Keyspaces 時，使用輕量型交易不會受到效能懲罰。

Load balancing

`system.peers` 資料表項目對應至 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負載平衡政策，並調校每個 IP 的連線數，以符合應用程式的需求。

分頁

Amazon Keyspaces 會根據讀取處理請求的資料列數來分頁結果，而不是結果集中傳回的資料列數。因此，某些頁面可能包含的資料列可能少於您在 PAGE SIZE 中為篩選的查詢指定的資料列。此外，Amazon Keyspaces 會在讀取 1 MB 的資料後自動分頁結果，為客戶提供一致的單一位數毫秒讀取效能。如需詳細資訊，請參閱[the section called “分頁結果”](#)。

在具有靜態資料欄的資料表中，Apache Cassandra 和 Amazon Keyspaces 都會在多頁查詢中每個頁面的開頭建立分割區的靜態資料欄值。當資料表具有大型資料列時，由於 Amazon Keyspaces 分頁行為，範圍讀取操作結果傳回 Amazon Keyspaces 頁面的可能性高於 Apache Cassandra。因此，Amazon Keyspaces 中同時更新靜態資料欄的可能性較高，可能會導致範圍讀取結果集的不同頁面中的靜態資料欄值不同。

分割區

Amazon Keyspaces 中的預設分割區是與 Cassandra 相容的 Murmur3Partitioner。此外，您可以選擇使用 Amazon Keyspaces DefaultPartitioner 或 Cassandra 相容的 RandomPartitioner。

使用 Amazon Keyspaces，您可以安全地變更帳戶的分割區，而無需重新載入 Amazon Keyspaces 資料。組態變更完成後，大約需要 10 分鐘，用戶端會在下次連線時自動看到新的分割區設定。如需詳細資訊，請參閱[the section called “使用分割區”](#)。

備妥的陳述式

Amazon Keyspaces 支援將預備陳述式用於資料處理語言 (DML) 操作，例如讀取和寫入資料。Amazon Keyspaces 目前不支援將預備陳述式用於資料定義語言 (DDL) 操作，例如建立資料表和金鑰空間。DDL 操作必須在預備陳述式之外執行。

刪除範圍

Important

從 2026 年 7 月 1 日開始，範圍刪除操作將從同步轉換為非同步處理。在此變更之後，成功的回應會確認您的刪除請求已被接受，但刪除可能仍在進行中。若要確認刪除已完成，請查詢指定範圍內的項目。當查詢未傳回任何結果時，刪除即完成。

Amazon Keyspaces 支援刪除範圍內的資料列。範圍是分割區內一組連續的資料列。您可以使用 WHERE 子句在 DELETE 操作中指定範圍。您可以指定範圍為整個分割區。

此外，您可以使用關聯式運算子（例如，'>'、'<'），或包含分割區索引鍵並省略一或多個叢集資料欄，將範圍指定為分割區內連續資料列的子集。使用 Amazon Keyspaces，您可以在單一操作中刪除範圍內最多 1,000 個資料列。

範圍刪除不會隔離。正在進行範圍刪除時，其他操作可以看到個別的資料列刪除。

系統表

Amazon Keyspaces 會填入 Apache 2.0 開放原始碼 Cassandra 驅動程式所需的系統資料表。用戶端可見的系統資料表包含已驗證使用者唯一的資訊。系統資料表完全由 Amazon Keyspaces 控制，且為唯讀。如需詳細資訊，請參閱[the section called “系統金鑰空間”](#)。

系統資料表需要唯讀存取權，您可以使用 IAM 存取政策來控制它。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，以不同的方式定義系統資料表的標籤型存取控制政策。若要進一步了解系統資料表的標籤型存取控制，請參閱 [the section called “根據標籤的 Amazon Keyspaces 資源存取”](#)。

如果您使用 Amazon [VPC 端點](#) 存取 Amazon Keyspaces，您會在 `system.peers` Amazon Keyspaces 有權查看的每個 Amazon VPC 端點表格中看到項目。因此，您的 Cassandra 驅動程式可能會發出有關 `system.peers` 資料表中控制節點本身的 [警告訊息](#)。您可以安全地忽略此警告。

時間戳記

在 Amazon Keyspaces 中，與 Apache Cassandra 中的預設時間戳記相容的儲存格層級時間戳記是一項選擇加入功能。

只有在資料表的用戶端時間戳記開啟時，才能使用 `USING TIMESTAMP` 子句和 `WRITETIME` 函數。若要進一步了解 Amazon Keyspaces 中的用戶端時間戳記，請參閱 [the section called “用戶端時間戳記”](#)。

使用者定義的類型 (UDTs)

Amazon Keyspaces 中的 UDTs 不支援不等式運算子。

若要了解如何在 UDTs，請參閱 [the section called “使用者定義的類型 \(UDTs\)”](#)。

若要檢閱每個金鑰空間支援的 UDTs 數量、支援的巢狀層級，以及其他與 UDTs 相關的預設值和配額，請參閱 [the section called “Amazon Keyspaces 中使用者定義類型 \(UDTs\) 的配額和預設值”](#)。

支援的 Cassandra APIs、操作、函數和資料類型

Amazon Keyspaces (適用於 Apache Cassandra) 與 Cassandra Query Language (CQL) 3.11 API (與 2.x 版回溯相容) 相容。

Amazon Keyspaces 支援所有常用的 Cassandra 資料平面操作，例如建立金鑰空間和資料表、讀取資料和寫入資料。

下列各節列出支援的功能。

主題

- [Cassandra API 支援](#)
- [Cassandra 控制平面 API 支援](#)
- [Cassandra 資料平面 API 支援](#)
- [Cassandra 函數支援](#)
- [Cassandra 資料類型支援](#)

Cassandra API 支援

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
LOGGED BATCH	是
SELECT	是
INSERT	是
DELETE	是
UPDATE	是
USE	是
CREATE TYPE	是
ALTER TYPE	否
DROP TYPE	是
CREATE TRIGGER	否

API 操作	支援
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

Cassandra 控制平面 API 支援

由於 Amazon Keyspaces 受管，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，下列 Cassandra 功能不適用。

功能	Reason
耐久寫入切換	所有寫入都耐用
讀取修復設定	不適用
GC 寬限期秒數	不適用
Bloom 篩選條件設定	不適用
壓縮設定	不適用
Compression settings (壓縮設定)	不適用
快取設定	不適用

功能	Reason
安全性設定	由 IAM 取代

Cassandra 資料平面 API 支援

功能	支援
SELECT 和 INSERT 陳述式的 JSON 支援	是
靜態資料欄	是
存留時間 (TTL)	是

Cassandra 函數支援

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

函式	支援
Aggregate 函式	否
Blob 轉換	是
Cast	是
Datetime 函式	是
Timeconversion 函數	是
TimeUuid 函式	是
Token	是
User defined functions (UDF)	否
Uuid	是

Cassandra 資料類型支援

下表列出 Amazon Keyspaces 中支援的 Apache Cassandra 資料類型。如需 Amazon Keyspaces 中資料類型的詳細資訊，請參閱 [the section called “資料類型”](#)。

資料類型	支援	
ascii	是	
bigint	是	
blob	是	
boolean	是	
counter	是	
date	是	
decimal	是	
double	是	
float	是	
frozen	是	
inet	是	
int	是	
list	是	
map	是	
set	是	
smallint	是	
text	是	

資料類型	支援	
time	是	
timestamp	是	
timeuuid	是	
tinyint	是	
tuple	是	
user-defined types (UDTs)	是	
uuid	是	
varchar	是	
varint	是	

支援的 Apache Cassandra 讀取和寫入一致性層級和相關成本

本節中的主題說明 Amazon Keyspaces (適用於 Apache Cassandra) 中的讀取和寫入操作支援哪些 Apache Cassandra 一致性層級。

主題

- [寫入一致性層級](#)
- [讀取一致性層級](#)
- [不支援的一致性層級](#)

寫入一致性層級

Amazon Keyspaces 會跨多個可用區域複寫所有寫入操作三次，以提供耐用性和高可用性。在使用 LOCAL_QUORUM 一致性層級進行確認之前，寫入會永久儲存。對於每個 1 KB 寫入，您需要為使用佈建容量模式的資料表支付 1 個寫入容量單位 (WCU)，或為使用隨需模式的資料表支付 1 個寫入請求單位 (WRU)。

您可以使用 `cqlsh` 將目前工作階段中所有查詢的一致性設定為 `LOCAL_QUORUM` 使用下列程式碼。

```
CONSISTENCY LOCAL_QUORUM;
```

若要以程式設計方式設定一致性層級，您可以使用適當的 Cassandra 用戶端驅動程式來設定一致性。例如，4.x 版 Java 驅動程式可讓您在 `app config` 檔案中設定一致性層級，如下所示。

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

如果您使用的是 3.x 版 Java Cassandra 驅動程式，您可以透過新增 `來指定工作階段的一致性層級` `.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-1"))
    .withSSL()
    .withQueryOptions(new
QueryOptions().setConsistencyLevel(ConsistencyLevel.LOCAL_QUORUM)
    .build())
    .connect();
```

若要設定特定寫入操作的一致性層級，您可以在使用 Java 驅動程式使用 `QueryBuilder.insertInto` 時定義一致性。

讀取一致性層級

Amazon Keyspaces 支援三種讀取一致性層級：ONE、LOCAL_ONE 和 LOCAL_QUORUM。

在 LOCAL_QUORUM 讀取期間，Amazon Keyspaces 會傳回回應，反映先前所有成功寫入操作的最新更新。使用一致性層級 ONE 或 LOCAL_ONE 可以改善讀取請求的效能和可用性，但回應可能不會反映最近完成寫入的結果。

對於使用 ONE 或 LOCAL_ONE 一致性的每個 4 KB 讀取，您需要為使用佈建容量模式的資料表支付 0.5 個讀取容量單位 (RCUs)，或為使用隨需模式的資料表支付 0.5 個讀取請求單位 (RRUs)。對於使用 LOCAL_QUORUM 一致性的每個 4 KB 讀取，您需要為使用佈建容量模式的資料表支付 1 個讀取容量單位 (RCU)，或為使用隨需模式的資料表支付 1 個讀取請求單位 (RRU)。

針對每 4 KB 讀取，根據每個資料表的讀取一致性和讀取容量輸送量模式計費

一致性層級	佈建	隨需
ONE	0.5 RCUs	0.5 RRUs
LOCAL_ONE	0.5 RCUs	0.5 RRUs
LOCAL_QUORUM	1 個 RCU	1 個 RRU

若要為讀取操作指定不同的一致性，當您使用 Java 驅動程式時，`QueryBuilder.select` 請使用 `setConsistencyLevel` 引數呼叫。

不支援的一致性層級

Amazon Keyspaces 不支援下列一致性層級，並會導致例外狀況。

不支援的一致性層級

Apache Cassandra	Amazon Keyspaces
EACH_QUORUM	不支援
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

遷移至 Amazon Keyspaces (適用於 Apache Cassandra)

遷移至 Amazon Keyspaces (適用於 Apache Cassandra) 可為企業和組織帶來一系列令人信服的好處。以下是讓 Amazon Keyspaces 成為遷移之吸引人選擇的一些主要優勢。

- **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磁碟區和流量。使用傳統 Cassandra，擴展不會隨需執行，需要規劃未來的尖峰。使用 Amazon Keyspaces，您可以根據需求輕鬆擴展或縮減資料表，確保您的應用程式可以在不影響效能的情況下處理流量突然激增的情況。
- **效能** – Amazon Keyspaces 提供低延遲的資料存取，讓應用程式能夠以卓越的速度擷取和處理資料。其分散式架構可確保讀取和寫入操作分散在多個節點，即使在高請求率下也能提供一致的單一位數毫秒回應時間。
- **全受管** – Amazon Keyspaces 是由提供的全受管服務 AWS。這表示會 AWS 處理資料庫管理的操作層面，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，而非資料庫管理任務。
- **無伺服器架構** – Amazon Keyspaces 是無伺服器。您只需為消耗的容量付費，而不需要預先佈建容量。您沒有要管理的伺服器或要選擇的執行個體。此 pay-per-request 模式提供成本效益和最低的營運開銷，因為您只需支付使用的資源，而不需要佈建和監控容量。
- **具有結構描述的 NoSQL 彈性** – Amazon Keyspaces 遵循 NoSQL 資料模型，在結構描述設計中提供彈性。使用 Amazon Keyspaces，您可以存放結構化、半結構化和非結構化資料，使其非常適合處理多樣化和不斷變化的資料類型。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。這種靈活性可加快開發週期，並更輕鬆地適應不斷變化的業務需求。
- **高可用性和耐久性** – Amazon Keyspaces 會跨中的多個[可用區域](#)複寫資料 AWS 區域，以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至最低。Amazon Keyspaces 提供高達 99.999% 的可用性 SLA。如需更多彈性和低延遲本機讀取，Amazon Keyspaces 提供[多區域複寫](#)。
- **安全性與合規** – Amazon Keyspaces 與整合，AWS Identity and Access Management 以進行精細存取控制。它提供靜態和傳輸中的加密，有助於提高資料的安全性。第三方稽核人員已評估 Amazon Keyspaces 對特定計劃的安全性和合規性，包括 HIPAA、PCI DSS 和 SOC，讓您能夠符合法規要求。如需詳細資訊，請參閱[the section called “法規遵循驗證”](#)。
- **與 AWS 生態系統整合** – 作為生態系統的一部分 AWS，Amazon Keyspaces 與其他無縫整合 AWS 服務，例如 AWS CloudFormation Amazon CloudWatch 和 AWS CloudTrail。此整合可讓您建置無伺服器架構、利用基礎設施做為程式碼，以及建立即時資料驅動型應用程式。如需詳細資訊，請參閱[監控 Amazon Keyspaces](#)。

主題

- [建立遷移計畫，以從 Apache Cassandra 遷移至 Amazon Keyspaces](#)
- [如何選取正確的工具，將資料大量上傳或遷移至 Amazon Keyspaces](#)

建立遷移計畫，以從 Apache Cassandra 遷移至 Amazon Keyspaces

若要成功從 Apache Cassandra 遷移至 Amazon Keyspaces，建議您檢閱適用的遷移概念和最佳實務，以及可用選項的比較。

本主題概述遷移程序如何運作，方法是介紹幾個關鍵概念，以及您可以使用的工具和技術。您可以評估不同的遷移策略，以選取最符合您需求的遷移策略。

主題

- [功能相容性](#)
- [預估 Amazon Keyspaces 定價](#)
- [選擇遷移策略](#)
- [線上遷移至 Amazon Keyspaces：策略和最佳實務](#)
- [離線遷移程序：Apache Cassandra 到 Amazon Keyspaces](#)
- [使用混合遷移解決方案：Apache Cassandra 到 Amazon Keyspaces](#)

功能相容性

在遷移之前，請仔細考慮 Apache Cassandra 和 Amazon Keyspaces 之間的功能差異。Amazon Keyspaces 支援所有常用的 Cassandra 資料平面操作，例如建立金鑰空間和資料表、讀取資料和寫入資料。

不過，Amazon Keyspaces 不支援一些 Cassandra APIs。如需支援 APIs 的詳細資訊，請參閱 [the section called “支援的 Cassandra APIs、操作、函數和資料類型”](#)。如需 Amazon Keyspaces 和 Apache Cassandra 之間所有功能差異的概觀，請參閱 [the section called “Apache Cassandra 的功能差異”](#)。

若要比較您使用的 Cassandra APIs 和結構描述與 Amazon Keyspaces 中支援的功能，您可以執行 [GitHub](#) 上的 Amazon Keyspaces 工具組中提供的相容性指令碼。

如何使用相容性指令碼

1. 從 [GitHub](#) 下載相容性 Python 指令碼，並將其移至可存取現有 Apache Cassandra 叢集的位置。
2. 相容性指令碼使用與類似的參數 CQLSH。針對 `--host` 和 `--port` 輸入 IP 地址，以及您用來連接和執行查詢到叢集中其中一個 Cassandra 節點的連接埠。

如果您的 Cassandra 叢集使用身分驗證，您也需要提供 `-username` 和 `-password`。若要執行相容性指令碼，您可以使用下列命令。

```
python toolkit-compat-tool.py --host hostname or IP -u "username" -p "password" --  
port native transport port
```

預估 Amazon Keyspaces 定價

本節提供從 Apache Cassandra 資料表收集所需資訊的概觀，以計算 Amazon Keyspaces 的預估成本。每個資料表都需要不同的資料類型、需要支援不同的 CQL 查詢，並維護獨特的讀取/寫入流量。

根據資料表來考慮您的需求，符合 Amazon Keyspaces 資料表層級的資源隔離和[讀取/寫入輸送量容量模式](#)。使用 Amazon Keyspaces，您可以獨立定義資料表的讀取/寫入容量和[自動擴展政策](#)。

了解資料表需求可協助您根據功能、成本和遷移工作來排定遷移資料表的優先順序。

在遷移之前收集下列 Cassandra 資料表指標。此資訊有助於估算 Amazon Keyspaces 上的工作負載成本。

- 資料表名稱 – 完整金鑰空間和資料表名稱的名稱。
- 描述 – 資料表的描述，例如其使用方式，或其中存放的資料類型。
- 每秒平均讀取次數 – 在大型時間間隔內，相對於資料表的平均座標層級讀取次數。
- 每秒平均寫入次數 – 在大量時間間隔內，資料表的座標層級寫入平均次數。
- 以位元組為單位的平均資料列大小 – 以位元組為單位的平均資料列大小。
- 以 GBs 為單位的儲存體大小 – 資料表的原始儲存體大小。
- 讀取一致性明細 – 使用最終一致性 (LOCAL_ONE 或 ONE) 與強式一致性 () 的讀取百分比 LOCAL_QUORUM。

此資料表顯示您在規劃遷移時需要彙整的資料表相關資訊範例。

資料表名稱	Description	每秒平均讀取數	每秒平均寫入數	平均資料列大小，以位元組為單位	儲存體大小，以 GBs 為單位	讀取一致性明細
mykeyspace.mytable	用來存放購物車歷史記錄	10,000	5,000	2,200	2,000	100% LOCAL_ONE
mykeyspace.mytable2	用來存放最新的設定檔資訊	20,000	1,000	850	1,000	25% LOCAL_QUORUM 75% LOCAL_ONE

如何收集資料表指標

本節提供如何從現有 Cassandra 叢集收集必要資料表指標的逐步說明。這些指標包括資料列大小、資料表大小和每秒讀取/寫入請求數 (RPS)。它們可讓您評估 Amazon Keyspaces 資料表的輸送量容量需求，並預估定價。

如何在 Cassandra 來源資料表上收集資料表指標

1. 判斷資料列大小

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 Cassandra 字元範圍內的典型資料分佈。



您可以使用 [GitHub](#) 上可用的資料列大小採樣器指令碼，收集 Cassandra 叢集中每個資料表的資料列大小指標。

指令碼會使用 `cqlsh` 和 從 Apache Cassandra 匯出資料表資料，`awk` 以計算資料列大小與可設定之資料表資料範例集之間的最小、最大、平均和標準差。資料列大小取樣器會將引數傳遞至 `cqlsh`，因此相同的參數可用於連接和讀取 Cassandra 叢集。

下列陳述式為範例。

```
./row-size-sampler.sh 10.22.33.44 9142 \\  
-u "username" -p "password" --ssl
```

如需如何在 Amazon Keyspaces 中計算資料列大小的詳細資訊，請參閱 [the section called “估算資料列大小”](#)。

2. 決定資料表大小

使用 Amazon Keyspaces，您不需要預先佈建儲存體。Amazon Keyspaces 會持續監控資料表的計費大小，以判斷您的儲存費用。儲存空間按 GB 月計費。Amazon Keyspaces 資料表大小是根據單一複本的原始大小（未壓縮）。

若要監控 Amazon Keyspaces 中的資料表大小，您可以使用指標 `BillableTableSizeInBytes`，該指標會針對 中的每個資料表顯示 AWS 管理主控台。

若要預估 Amazon Keyspaces 資料表的計費大小，您可以使用下列兩種方法之一：

- 使用平均資料列大小，並乘以數字或資料列。

您可以將平均資料列大小乘以 Cassandra 來源資料表中的資料列數，以估計 Amazon Keyspaces 資料表的大小。使用上一節的資料列大小範例指令碼來擷取平均的資料列大小。若要擷取資料列計數，您可以使用 等工具 `dsbulk count` 來判斷來源資料表中的資料列總數。

- 使用 `nodetool` 收集資料表中繼資料。

`Nodetool` 是 Apache Cassandra 分佈中提供的管理工具，可讓您深入了解 Cassandra 程序的狀態並傳回資料表中繼資料。您可以使用 `nodetool` 來取樣資料表大小的中繼資料，並使用 來推斷 Amazon Keyspaces 中的資料表大小。

要使用的命令是 `nodetool tablestats`。Tablestats 會傳回資料表的大小和壓縮比。資料表的大小會儲存為資料表 `tablelivespace` 的，您可以將其除以 `compression ratio`。然後，將此大小值乘以節點數量。最後除以複寫係數（通常是三個）。

這是計算的完整公式，您可以用來評估資料表大小。

```
((tablelivespace / compression ratio) * (total number of nodes)) / (replication factor)
```

假設您的 Cassandra 叢集有 12 個節點。執行 `nodetool tablestats` 命令會傳回 `tablelivespace` 200 GB 的 和 0.5 `compression ratio` 的。金鑰空間的複寫係數為 3。

這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. 擷取讀取和寫入次數

若要判斷 Amazon Keyspaces 資料表的容量和擴展需求，請在遷移之前擷取 Cassandra 資料表的讀取和寫入請求率。

Amazon Keyspaces 是無伺服器，您只需為使用量付費。一般而言，Amazon Keyspaces 中的讀取/寫入輸送量價格取決於請求的數量和大小。

Amazon Keyspaces 有兩種容量模式：

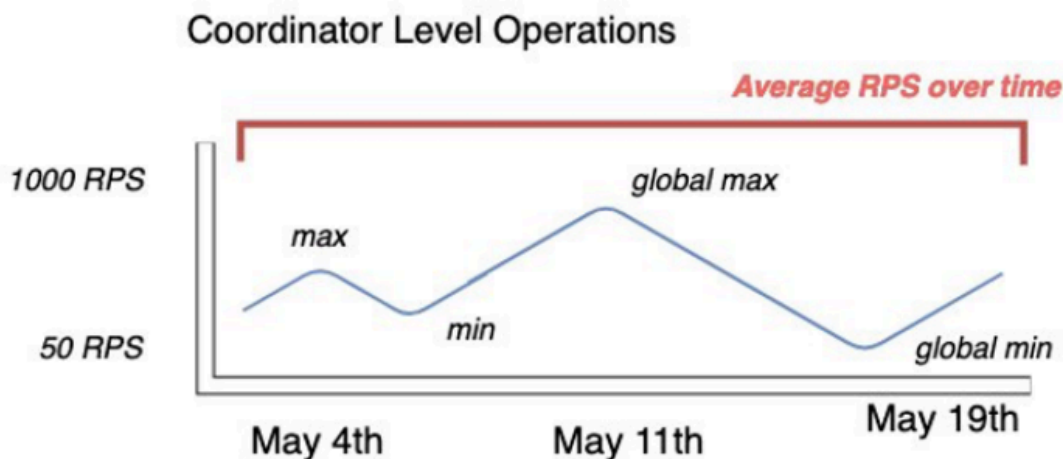
- [隨需](#) – 這是一個靈活的計費選項，能夠每秒處理數千個請求，而無需進行容量規劃。它提供讀取和寫入請求 `pay-per-request` 定價，因此您只需支付使用量的費用。
- [佈建](#) – 如果您選擇佈建輸送量容量模式，您可以指定應用程式所需的每秒讀取和寫入次數。這可協助您管理 Amazon Keyspaces 用量，以維持或低於定義的請求率，以維持可預測性。

佈建模式提供 [自動擴展](#) 功能，可自動調整佈建速率以擴展或縮減規模，以提升營運效率。如需無伺服器資源管理的詳細資訊，請參閱 [管理無伺服器資源](#)。

由於您在 Amazon Keyspaces 中分別佈建讀取和寫入輸送量容量，因此您需要獨立測量現有資料表中讀取和寫入的請求率。

若要從現有的 Cassandra 叢集收集最準確的使用率指標，請針對在單一資料中心所有節點上彙總的資料表，擷取長時間內協調器層級讀取和寫入操作的平均每秒請求數 (RPS)。

擷取至少數週內的平均 RPS 會擷取流量模式中的尖峰和山谷，如下圖所示。



您有兩個選項可判斷 Cassandra 資料表的讀取和寫入請求率。

- 使用現有的 Cassandra 監控

您可以使用下表所示的指標來觀察讀取和寫入請求。請注意，指標名稱可能會根據您使用的監控工具而變更。

維度	Cassandra JMX 指標
寫入	<code>org.apache.cassandra.metrics:type=ClientRequest,scope=Write,name=Latency#Count</code>
讀取	<code>org.apache.cassandra.metrics:type=ClientRequest,scope=Read,name=Latency#Count</code>

- 使用 `nodetool`

使用 `nodetool tablestats` 和 從資料表 `nodetool info` 擷取平均讀取和寫入操作。會 `tablestats` 傳回節點啟動後的總讀取和寫入計數。會以秒為單位 `nodetool info` 提供節點的執行時間。

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正常執行時間，以秒為單位。然後，對於讀取，您將除以一致性層級廣告，對於寫入，您將除以複寫係數。這些計算會以下列公式表示。

每秒平均讀取的公式：

```
((number of reads * number of nodes in cluster) / read consistency quorum
(2)) / uptime
```

每秒平均寫入的公式：

```
((number of writes * number of nodes in cluster) / replication factor of 3) /
uptime
```

假設我們有 12 個節點叢集，已持續運作 4 週。會 `nodetool info` 傳回 2,419,200 秒的正常運作時間，並 `nodetool tablestats` 傳回 10 億筆寫入和 20 億筆讀取。此範例將導致以下計算。

```
((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. 判斷資料表的容量使用率

若要估計平均容量使用率，請從平均請求率和 Cassandra 來源資料表的平均資料列大小開始。

Amazon Keyspaces 使用讀取容量單位 (RCUs) 和寫入容量單位 (WCUs) 來測量資料表讀取和寫入的佈建輸送量容量。在此預估中，我們會使用這些單位來計算遷移後新 Amazon Keyspaces 資料表的讀取和寫入容量需求。

本主題稍後將討論佈建容量模式和隨需容量模式之間的選擇如何影響帳單。但是在此範例中，對於容量使用率的預估，我們假設資料表處於佈建模式。

- 讀取 – 一個 RCU 代表最大 4 KB 的資料列一個LOCAL_QUORUM讀取請求或兩個LOCAL_ONE讀取請求。如果您需要讀取大於 4 KB 的資料列，則讀取操作會使用其他 RCUs。所需的 RCUs 總數取決於資料列大小，以及您是否要使用或LOCAL_QUORUMLOCAL_ONE讀取一致性。

例如，讀取 8 KB 資料列需要 2 個使用LOCAL_QUORUM讀取一致性RCUs，如果您選擇LOCAL_ONE讀取一致性，則需要 1 個 RCU。

- 寫入 – 一個 WCU 代表大小上限為 1 KB 的資料列一個寫入。所有寫入都使用LOCAL_QUORUM一致性，而且使用輕量型交易 (LWTs) 無需額外付費。

所需的 WCUs 總數取決於資料列大小。如果您需要寫入大於 1 KB 的資料列，則寫入操作會使用其他 WCUs。例如，如果您的資料列大小為 2 KB，則需要 2 WCUs 來執行一個寫入請求。

下列公式可用來預估所需的 RCUs 和 WCUs。

- RCUs中的讀取容量可以透過將每秒讀取數乘以每個讀取的讀取資料列數乘以平均資料列大小除以 4KB 並四捨五入至最接近的整數來判斷。
- WCUs中的寫入容量可以透過將請求數量乘以平均資料列大小除以 1KB 並四捨五入至最接近的整數來決定。

這在下列公式中表示。

```
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 Cassandra 資料表上執行的資料列大小為 2.5KB 的 4,960 個讀取請求，則需要 Amazon Keyspaces 中的 4,960 RCUs。如果您目前在 Cassandra 資料表上執行每秒 1,653 個寫入請求，且資料列大小為 2.5KB，則 Amazon Keyspaces 中需要每秒 4,959 個 WCUs。

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 可讓您在每個讀取請求上儲存高達輸送量容量的一半。每個最終一致讀取最多可耗用 8KB。您可以計算最終一致性讀取，方法是將先前的計算乘以 0.5，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 的每月定價預估

若要根據讀取/寫入容量輸送量預估資料表的每月帳單，您可以使用不同的公式計算隨需和佈建模式的定價，並比較資料表的選項。

佈建模式 – 讀取和寫入容量消耗會根據每秒容量單位的每小時費率計費。首先，將該速率除以 0.7，表示預設自動擴展目標使用率為 70%。然後乘以 30 個日曆天、每天 24 小時，以及區域費率定價。

此計算摘要於下列公式中。

```

(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 – 讀取和寫入容量是以每個請求速率計費。首先，將請求率乘以 30 個日曆天和每天 24 小時。然後除以一百萬個請求單位。最後，將乘以區域費率。

此計算摘要於下列公式中。

```

((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

```


選擇遷移策略

從 Apache Cassandra 遷移至 Amazon Keyspaces 時，您可以選擇下列遷移策略：

- 線上 – 這是使用雙寫入開始同時將新資料寫入 Amazon Keyspaces 和 Cassandra 叢集的即時遷移。對於在遷移期間需要零停機時間並在寫入一致性後讀取的應用程式，建議使用此遷移類型。

如需如何規劃和實作線上遷移策略的詳細資訊，請參閱 [the section called “線上遷移”](#)。

- 離線 – 此遷移技術涉及在停機時間期間將資料集從 Cassandra 複製到 Amazon Keyspaces。離線遷移可以簡化遷移程序，因為它不需要變更您的應用程式，或在歷史資料和新寫入之間解決衝突。

如需如何規劃離線遷移的詳細資訊，請參閱 [the section called “離線移轉”](#)。

- 混合 – 此遷移技術允許近乎即時地將變更複寫至 Amazon Keyspaces，但在寫入一致性後無需讀取。

如需如何規劃混合遷移的詳細資訊，請參閱 [the section called “混合移轉”](#)。

檢閱本主題中討論的遷移技術和最佳實務後，您可以在決策樹中放置可用的選項，以根據您的需求和可用的資源設計遷移策略。

線上遷移至 Amazon Keyspaces：策略和最佳實務

如果您需要在從 Apache Cassandra 遷移至 Amazon Keyspaces 期間維持應用程式的可用性，您可以實作本主題討論的關鍵元件，以準備自訂線上遷移策略。透過遵循線上遷移的這些最佳實務，您可以確保在整個遷移過程中維持應用程式可用性和read-after-write一致性，將對使用者的影響降至最低。

設計從 Apache Cassandra 到 Amazon Keyspaces 的線上遷移策略時，您需要考慮下列關鍵步驟。

1. 寫入新資料

- Amazon Keyspaces 遷移的 ZDM Dual Write Proxy – 使用 [Github](#) 上提供的 ZDM Dual Write Proxy，從 Apache Cassandra 到 Amazon Keyspaces 執行零停機時間遷移。ZDM Proxy 執行雙寫入，無需重構現有應用程式，並執行雙讀取以進行查詢驗證。
- 應用程式雙寫入：您可以使用現有的 Cassandra 用戶端程式庫和驅動程式，在應用程式中實作雙寫入。指定一個資料庫做為領導者，另一個資料庫做為跟隨者。對跟隨者資料庫的寫入失敗會記錄在[無效字母佇列 \(DLQ\)](#) 中進行分析。
- 訊息層雙寫入：或者，您可以設定現有的訊息平台，使用其他取用者將寫入傳送至 Cassandra 和 Amazon Keyspaces。這會在兩個資料庫之間建立最終一致的檢視。

2. 遷移歷史資料

- 複製歷史資料：您可以使用 AWS Glue 或自訂擷取、轉換和載入 (ETL) 指令碼，將歷史資料從 Cassandra 遷移到 Amazon Keyspaces。使用輕量型交易或時間戳記等技術來處理雙寫入和大量載入之間的衝突解決。
- 使用 Time-To-Live (TTL)：對於較短的資料保留期間，您可以在 Cassandra 和 Amazon Keyspaces 中使用 TTL，以避免上傳不必要的歷史資料。隨著 Cassandra 中的舊資料過期，並透過雙寫入寫入新資料，Amazon Keyspaces 最終會追上進度。

3. 驗證資料

- 雙重讀取：實作來自 Cassandra（主要）和 Amazon Keyspaces（次要）資料庫的雙重讀取，以非同步方式比較結果。差異會記錄或傳送至 DLQ。
- 範例讀取：使用 Δ 函數定期取樣和比較兩個系統的資料，將任何差異記錄到 DLQ。

4. 遷移應用程式

- 藍綠策略：切換您的應用程式，在單一步驟中將 Amazon Keyspaces 視為主要資料存放區，並將 Cassandra 視為次要資料存放區。監控效能，並在發生問題時轉返。
- Canary 部署：先逐步將遷移推展到一部分的使用者，逐步增加 Amazon Keyspaces 作為主要使用者的流量，直到完全遷移為止。

5. 停用 Cassandra

一旦應用程式完全遷移至 Amazon Keyspaces 並驗證資料一致性，您就可以根據資料保留政策計劃停用 Cassandra 叢集。

透過利用這些元件規劃線上遷移策略，您可以順暢地轉換到全受管 Amazon Keyspaces 服務，並將停機時間或中斷降至最低。以下各節會更詳細地介紹每個元件。

主題

- [在線上遷移期間撰寫新資料](#)
- [在線上遷移期間上傳歷史資料](#)
- [驗證線上遷移期間的資料一致性](#)
- [在線上遷移期間遷移應用程式](#)
- [線上遷移後停用 Cassandra](#)

在線上遷移期間撰寫新資料

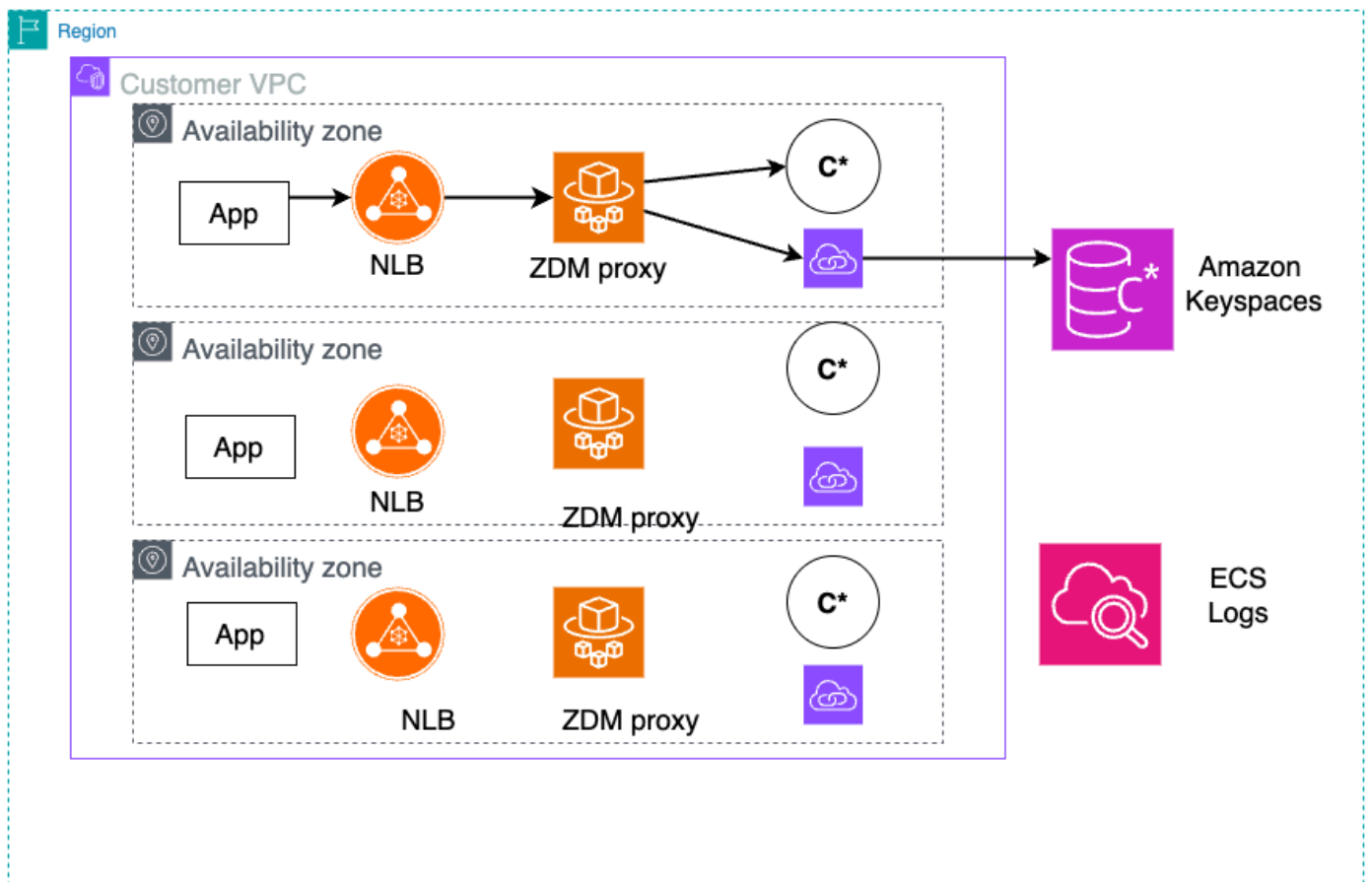
線上遷移計畫的第一步是確保應用程式寫入的任何新資料都存放在兩個資料庫、您現有的 Cassandra 叢集和 Amazon Keyspaces 中。目標是跨兩個資料存放區提供一致的檢視。您可以將所有新的寫入套用到兩個資料庫來執行此操作。若要實作雙寫入，請考慮下列三個選項之一。

- ZDM Dual Write Proxy for Amazon Keyspaces Migration – 使用 [Github](#) 上提供的 ZDM Proxy for Amazon Keyspaces，您可以將 Apache Cassandra 工作負載遷移至 Amazon Keyspaces，而不會造成應用程式停機。此增強型解決方案實作 AWS 最佳實務，並擴展官方 ZDM Proxy 功能。
- 在 Apache Cassandra 和 Amazon Keyspaces 之間執行線上遷移。
- 在不重構應用程式的情況下，同時將資料寫入來源和目標資料表。
- 透過雙讀取操作驗證查詢。

解決方案提供下列增強功能，以使用 AWS 和 Amazon Keyspaces。

- 容器部署 – 使用來自 Amazon Elastic Container Registry (Amazon ECR) 的預先設定 Docker 映像進行 VPC 可存取的部署。
- 基礎設施即程式碼 – 使用 AWS CloudFormation 範本部署以自動設定 AWS Fargate。
- Amazon Keyspaces 相容性 – 使用 Amazon Keyspaces 的自訂調整來存取系統資料表。

解決方案透過 Fargate 在 Amazon ECS 上執行，根據您的工作負載需求提供無伺服器可擴展性。網路負載平衡器會將傳入的應用程式流量分散到多個 Amazon ECS 任務，以獲得高可用性。



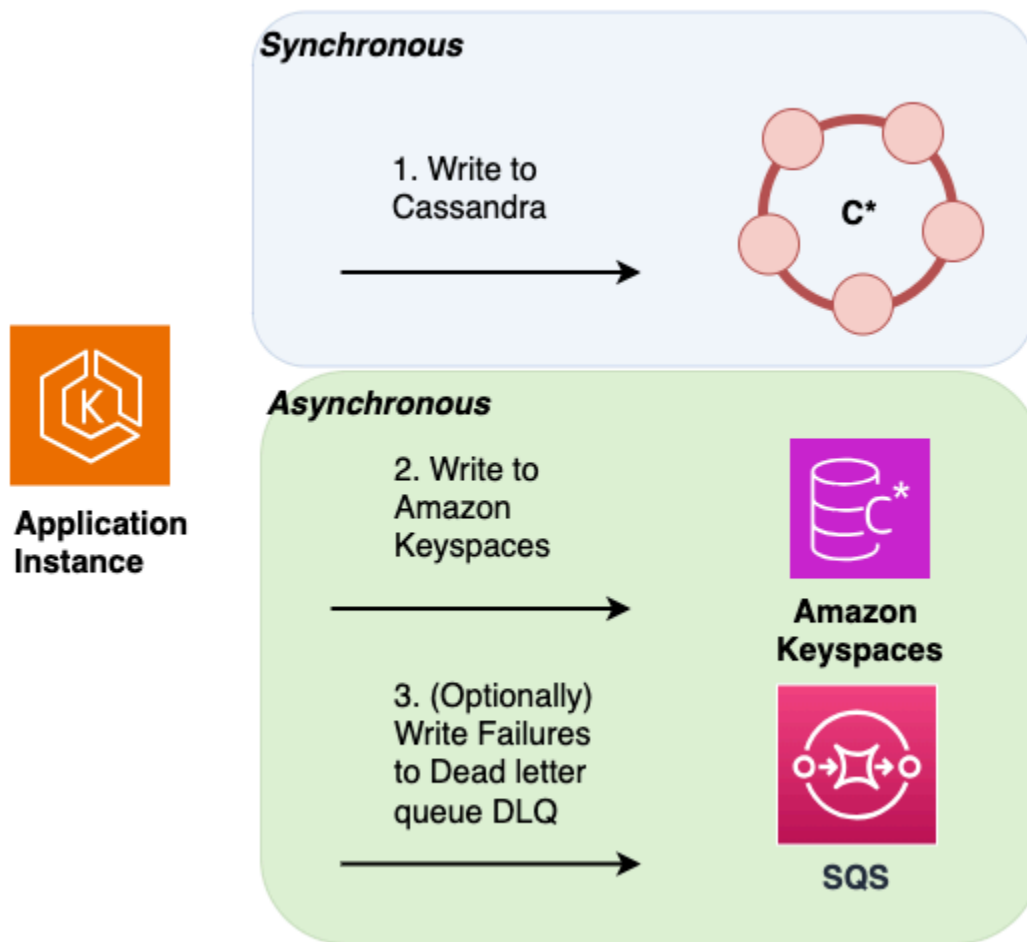
- 應用程式雙重寫入 – 您可以利用現有的 Cassandra 用戶端程式庫和驅動程式，在應用程式程式碼變更最少的情況下實作雙重寫入。您可以在現有應用程式中實作雙寫入，或在架構中建立新的 layer 來處理雙寫入。如需詳細資訊，以及顯示如何在現有應用程式中實作雙重寫入的客戶案例研究，請參閱 [Cassandra 遷移案例研究](#)。

實作雙重寫入時，您可以將一個資料庫指定為領導者，將另一個資料庫指定為跟隨者。這可讓您繼續寫入原始來源或領導資料庫，而不會讓後續的寫入失敗，或目的地資料庫中斷應用程式的關鍵路徑。

您可以使用 Amazon Simple Queue Service 在[無效字母佇列 \(DLQ\)](#) 中記錄失敗的寫入，而不是重試失敗的寫入至追隨者。DLQ 可讓您分析對追隨者的失敗寫入，並判斷為什麼在目的地資料庫中處理未成功。

如需更複雜的雙寫入實作，您可以遵循使用 [saga 模式](#) 設計一系列本機交易的 AWS 最佳實務。saga 模式可確保如果交易失敗，saga 會執行補償性交易，以還原先交易所做的資料庫變更。

使用雙寫入進行線上遷移時，您可以依照 saga 模式設定雙寫入，以便每次寫入都是本機交易，以確保異質資料庫之間的原子操作。如需使用建議的 設計模式來設計分散式應用程式的詳細資訊 AWS 雲端，請參閱[雲端設計模式、架構和實作](#)。



- 訊息層雙重寫入 – 您可以使用現有的訊息層對 Cassandra 和 Amazon Keyspaces 執行雙重寫入，而不是在應用程式層實作雙重寫入。

若要這樣做，您可以將其他消費者設定為您的簡訊平台，將寫入傳送至這兩個資料存放區。此方法使用訊息層提供簡單的低程式碼策略，在兩個資料庫之間建立兩個最終一致的檢視。

在線上遷移期間上傳歷史資料

實作雙重寫入以確保即時將新資料寫入兩個資料存放區後，遷移計劃的下一個步驟是評估您需要多少歷史資料才能從 Cassandra 複製或大量上傳至 Amazon Keyspaces。這可確保在您遷移應用程式之前，新資料和歷史資料都可以在新的 Amazon Keyspaces 資料庫中使用。

根據您的資料保留需求，例如，您需要根據組織政策保留多少歷史資料，您可以考慮以下兩個選項之一。

- 大量上傳歷史資料 – 歷史資料從現有 Cassandra 部署遷移到 Amazon Keyspaces 可以透過各種技術實現，例如使用 AWS Glue 或自訂指令碼來擷取、轉換和載入 (ETL) 資料。如需使用 AWS Glue 上傳歷史資料的詳細資訊，請參閱 [the section called “離線移轉”](#)。

規劃大量上傳歷史資料時，您需要考慮如何解決新寫入嘗試更新上傳過程中相同資料時可能發生的衝突。大量上傳預期最終會一致，這表示資料最終將到達所有節點。

如果因為新的寫入而同時發生相同資料的更新，您希望確保不會被歷史資料上傳覆寫。為了確保即使在大量匯入期間仍能保留資料的最新更新，您必須將衝突解決方法新增至大量上傳指令碼或應用程式邏輯中，以進行雙重寫入。

例如，您可以使用 [the section called “輕量型交易”\(LWT\)](#) 來比較和設定操作。若要這樣做，您可以將代表修改或狀態時間的額外欄位新增至資料模型。

此外，Amazon Keyspaces 支援 Cassandra WRITETIME 時間戳記函數。您可以使用 Amazon Keyspaces 用戶端時間戳記來保留來源資料庫時間戳記，並實作 last-writer-wins 衝突解決方案。如需詳細資訊，請參閱 [the section called “用戶端時間戳記”](#)。

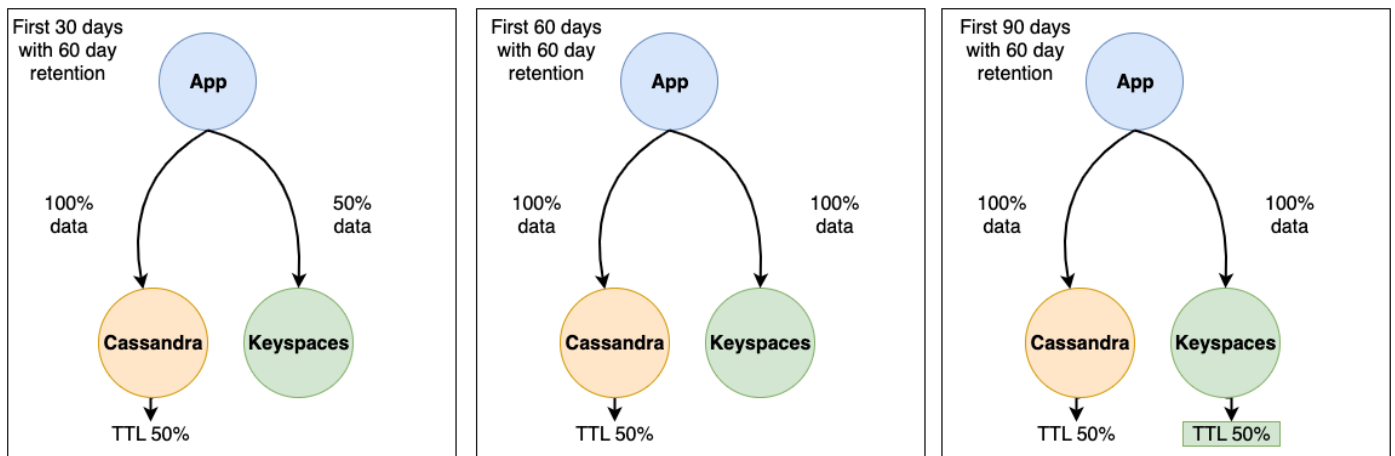
- 使用 Time-to-Live (TTL) – 對於少於 30、60 或 90 天的資料保留期，您可以在遷移期間在 Cassandra 和 Amazon Keyspaces 中使用 TTL，以避免將不必要的歷史資料上傳到 Amazon Keyspaces。TTL 可讓您設定一段時間，之後資料會自動從資料庫移除。

在遷移階段，您可以設定 TTL 設定，讓歷史資料在舊系統 (Cassandra) 中自動過期，同時僅使用雙寫入方法將新寫入套用至 Amazon Keyspaces。隨著時間的推移，使用舊資料持續在 Cassandra 叢集中過期，並使用雙寫入方法寫入的新資料，Amazon Keyspaces 會自動追上，以包含與 Cassandra 相同的資料。

這種方法可以大幅減少要遷移的資料量，進而產生更有效率且簡化的遷移程序。您可以在處理具有不同資料保留需求的大型資料集時考慮此方法。如需 TTL 的詳細資訊，請參閱 [the section called “使用存留時間過期資料”](#)。

請考慮下列使用 TTL 資料過期從 Cassandra 遷移至 Amazon Keyspaces 的範例。在此範例中，我們將兩個資料庫的 TTL 設定為 60 天，並顯示遷移程序在 90 天期間如何進行。在此期間，兩個資料庫都會使用雙寫入方法來接收相同的新寫入資料。我們將查看遷移的三個不同階段，每個階段為期 30 天。

遷移程序在每個階段的運作方式會顯示在下列影像中。



1. 在前 30 天之後，Cassandra 叢集和 Amazon Keyspaces 一直收到新的寫入。Cassandra 叢集也包含尚未達到 60 天保留的歷史資料，由叢集中 50% 的資料組成。

超過 60 天的資料會在 Cassandra 叢集中使用 TTL 自動刪除。此時，Amazon Keyspaces 包含存放在 Cassandra 叢集中的 50% 資料，由新寫入減去歷史資料組成。

2. 60 天後，Cassandra 叢集和 Amazon Keyspaces 都包含過去 60 天內寫入的相同資料。
3. 在 90 天內，Cassandra 和 Amazon Keyspaces 都包含相同的資料，並以相同的速率過期資料。

此範例說明如何使用過期日期設定為 60 天的 TTL 來避免上傳歷史資料的步驟。

驗證線上遷移期間的資料一致性

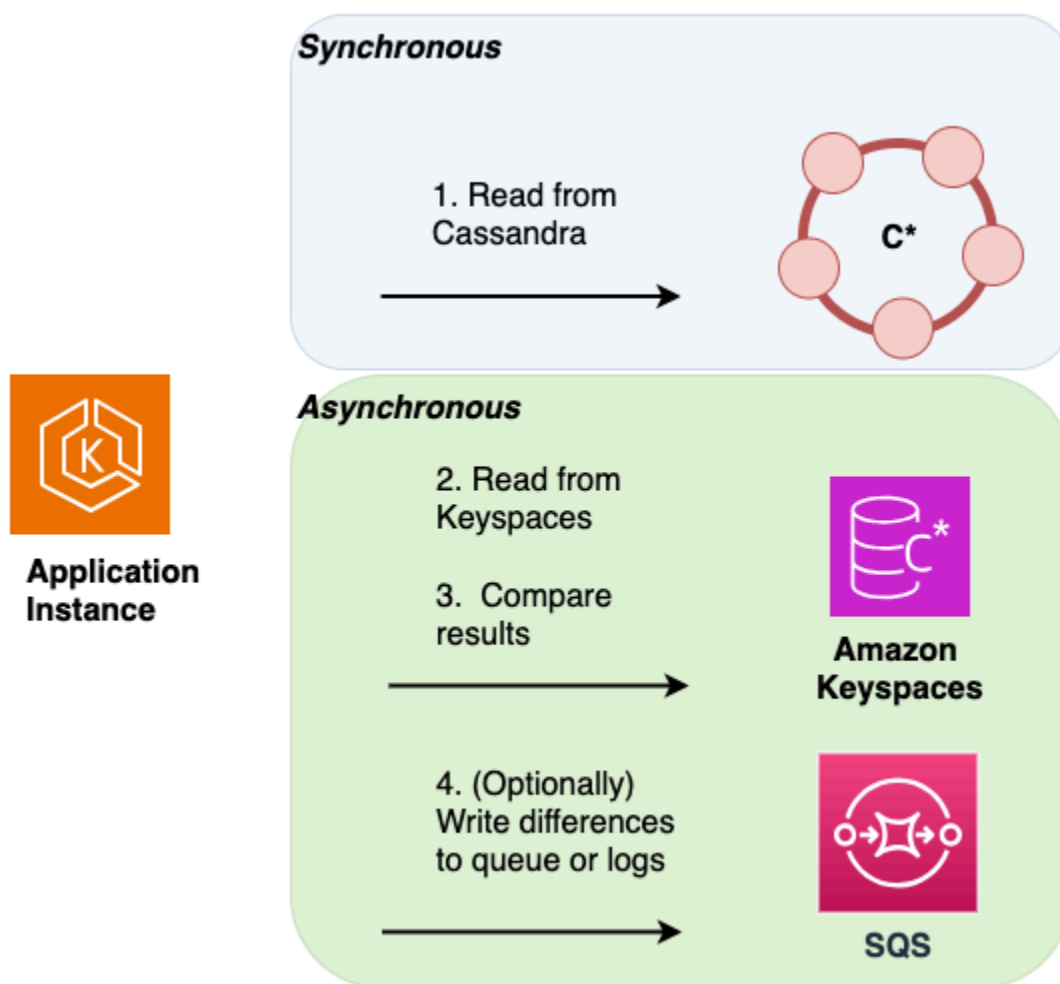
線上遷移程序的下一個步驟是資料驗證。雙重寫入正在將新資料新增至 Amazon Keyspaces 資料庫，而且您已完成使用大量上傳或資料過期搭配 TTL 遷移歷史資料。

現在，您可以使用驗證階段來確認兩個資料存放區實際上都包含相同的資料，並傳回相同的讀取結果。您可以選擇以下兩個選項之一，以驗證兩個資料庫都包含相同的資料。

- **雙重讀取** – 若要驗證來源和目的地資料庫是否包含相同的一組新寫入和歷史資料，您可以實作雙重讀取。若要這樣做，您會從主要 Cassandra 和次要 Amazon Keyspaces 資料庫讀取，類似於雙重寫入方法，並以非同步方式比較結果。

來自主要資料庫的結果會傳回給用戶端，而來自次要資料庫的結果會用來驗證主要結果集。找到的差異可以記錄或傳送至[無效字母佇列 \(DLQ\)](#) 以供日後調校。

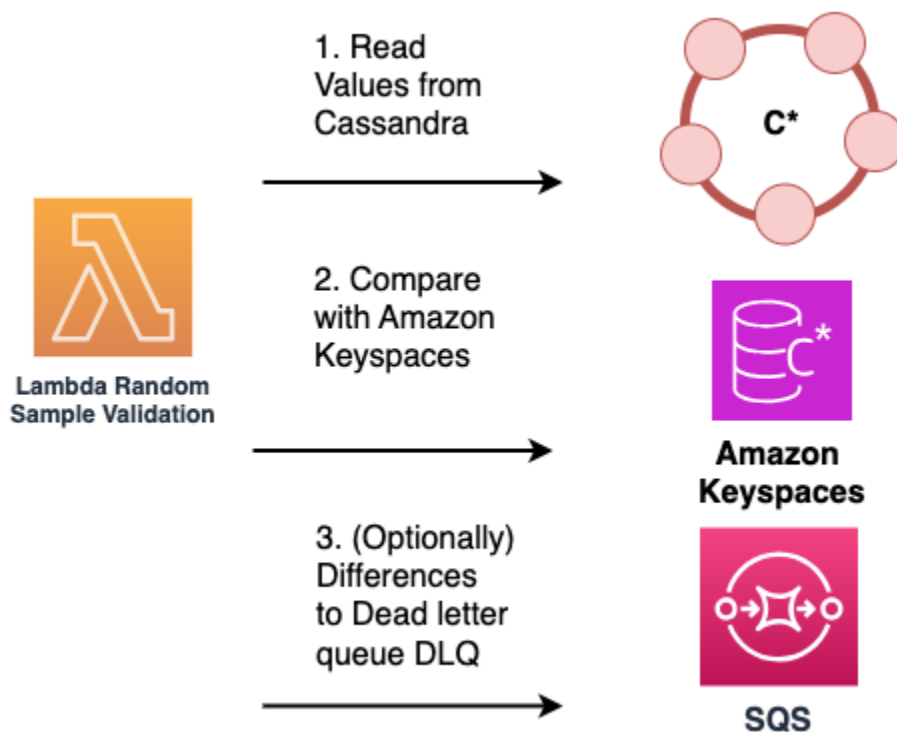
在下圖中，應用程式正在從主要資料存放區 Cassandra 執行同步讀取，並從次要資料存放區 Amazon Keyspaces 執行非同步讀取。



- 範例讀取 – 不需要變更應用程式程式碼的替代解決方案是使用 AWS Lambda 函數，定期並隨機從來源 Cassandra 叢集和目的地 Amazon Keyspaces 資料庫取樣資料。

這些 Lambda 函數可設定為定期執行。Lambda 函數會從來源和目的地系統擷取資料的隨機子集，然後執行取樣資料的比較。兩個資料集之間的任何差異或不相符都可以記錄並傳送到專用[無效字母佇列 \(DLQ\)](#) 以供日後調校。

下表說明此程序。



在線上遷移期間遷移應用程式

在線上遷移的第四階段中，您要遷移應用程式，並轉換為 Amazon Keyspaces 做為主要資料存放區。這表示您將應用程式切換為直接從 Amazon Keyspaces 讀取和寫入。為了確保對使用者的干擾降到最低，這應該是妥善規劃和協調的程序。

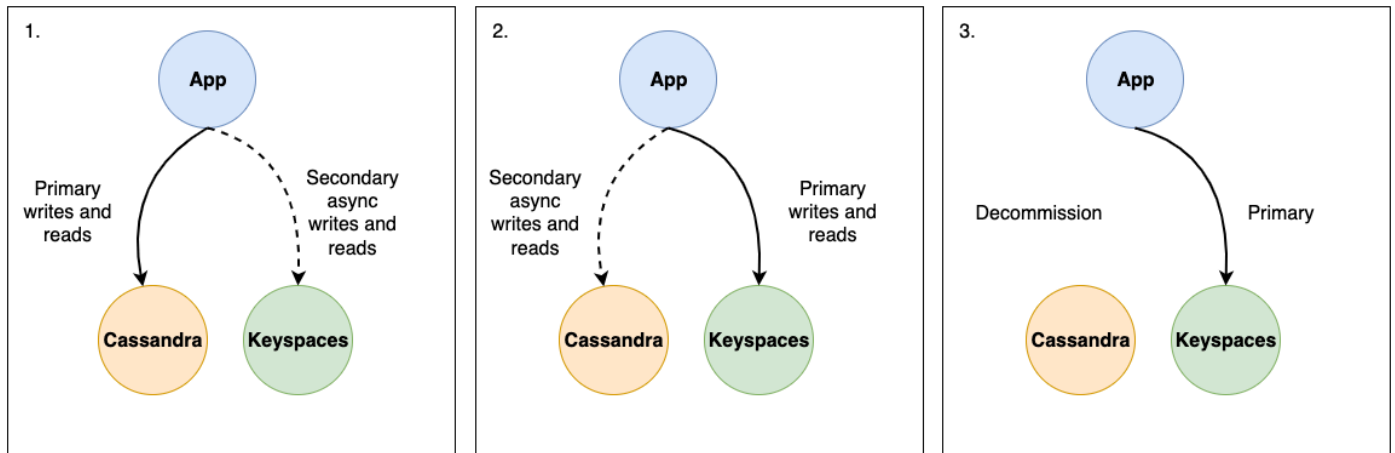
提供兩種不同的應用程式遷移建議解決方案：藍色綠色切入策略和金絲雀切入策略。下列各節會更詳細地概述這些策略。

- **藍色綠色策略** – 使用此方法，您可以切換應用程式，將 Amazon Keyspaces 視為主要資料存放區，並在單一步驟中將 Cassandra 視為次要資料存放區。您可以使用 AWS AppConfig 功能旗標來控制整個應用程式執行個體中主要和次要資料存放區的選擇。如需特徵標記的詳細資訊，請參閱在 [中建立特徵標記組態描述AWS AppConfig](#) 檔。

讓 Amazon Keyspaces 成為主要資料存放區後，您可以監控應用程式的行為和效能，確保 Amazon Keyspaces 符合您的需求，且遷移成功。

例如，如果您為應用程式實作雙讀取，在應用程式遷移階段，您將主要讀取從 Cassandra 轉換為 Amazon Keyspaces，次要讀取從 Amazon Keyspaces 轉換為 Cassandra。轉換後，您可以繼續監控和比較 [資料驗證](#) 一節中所述的結果，以確保在停用 Cassandra 之前兩個資料庫之間的一致性。

如果您偵測到任何問題，您可以透過還原至 Cassandra 做為主要資料存放區，快速復原至先前的狀態。只有在 Amazon Keyspaces 符合您作為主要資料存放區的所有需求時，您才能繼續遷移的停用階段。

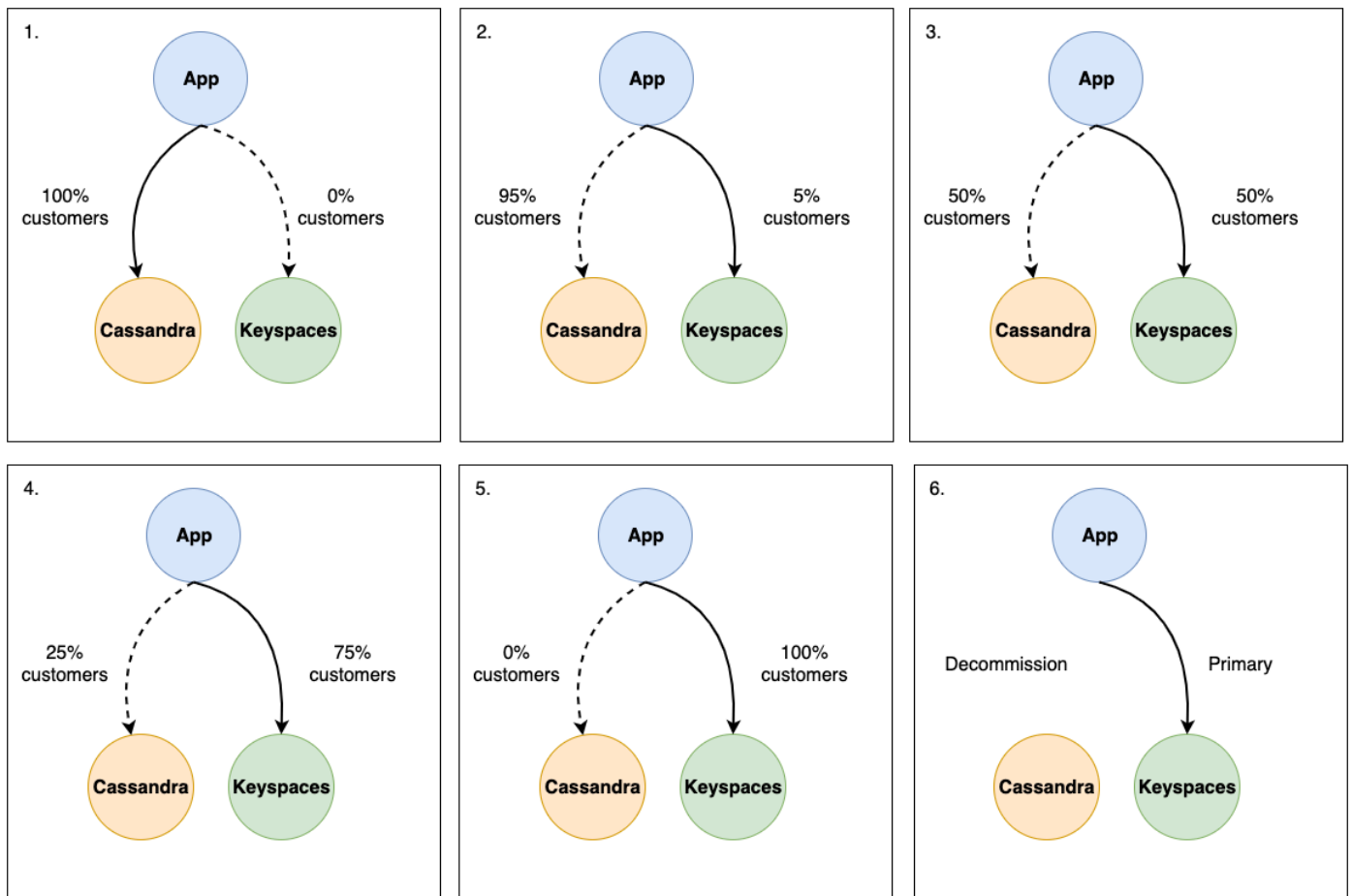


- **Canary 策略** – 在此方法中，您會逐步將遷移推展到使用者或流量的子集。最初，您應用程式流量的一小部分，例如 5% 的所有流量會使用 Amazon Keyspaces 做為主要資料存放區路由至版本，而其餘流量則會繼續使用 Cassandra 做為主要資料存放區。

這可讓您使用實際流量徹底測試遷移的版本，並監控其效能、穩定性和調查潛在問題。如果您未偵測到任何問題，您可以逐步增加路由到 Amazon Keyspaces 的流量百分比，直到它成為所有使用者和流量的主要資料存放區為止。

此階段推展可將廣泛服務中斷的風險降至最低，並允許更受控制的遷移程序。如果在 Canary 部署期間發生任何重大問題，您可以使用 Cassandra 作為受影響流量區段的主要資料存放區，快速復原至先前的版本。只有在您驗證 Amazon Keyspaces 可如預期處理 100% 的使用者和流量之後，您才能繼續遷移的解除委任階段。

下圖說明 Canary 策略的個別步驟。



線上遷移後停用 Cassandra

應用程式遷移完成後，您的應用程式會在 Amazon Keyspaces 上完全執行，而且您已驗證一段時間的資料一致性，您可以計劃停用 Cassandra 叢集。在此階段，您可以評估 Cassandra 叢集中剩餘的資料是否需要封存或可以刪除。這取決於您組織的資料處理和保留政策。

透過遵循此策略，並在規劃從 Cassandra 到 Amazon Keyspaces 的線上遷移時考慮本主題中描述的建議最佳實務，您可以確保無縫轉換至 Amazon Keyspaces，同時保持應用程式的read-after-write一致性和可用性。

從 Apache Cassandra 遷移到 Amazon Keyspaces 可以提供許多好處，包括降低營運開銷、自動擴展、改善安全性，以及協助您實現合規目標的架構。透過規劃具有雙重寫入、歷史資料上傳、資料驗證和漸進推展的線上遷移策略，您可以確保順暢的轉換，並將對應用程式及其使用者的干擾降至最低。

實作本主題中討論的線上遷移策略，可讓您驗證遷移結果、識別和解決任何問題，最終停用現有的 Cassandra 部署，以享有全受管 Amazon Keyspaces 服務。

離線遷移程序：Apache Cassandra 到 Amazon Keyspaces

當您可以負擔執行遷移的停機時間時，離線遷移是合適的。企業之間通常會有修補、大型版本或停機時間的維護時段，以進行硬體升級或主要升級。離線遷移可以使用此視窗來複製資料，並將應用程式流量從 Apache Cassandra 切換到 Amazon Keyspaces。

離線遷移可減少對應用程式的修改，因為它不需要同時與 Cassandra 和 Amazon Keyspaces 通訊。此外，在資料流程暫停的情況下，可以在不維護變動的情況下複製確切狀態。

在此範例中，我們使用 Amazon Simple Storage Service (Amazon S3) 做為離線遷移期間資料的預備區域，將停機時間降至最低。您可以使用 Spark Cassandra 連接器和 `spark-cassandra-connector`，將您以 Parquet 格式存放在 Amazon S3 中的資料自動匯入 Amazon Keyspaces 資料表 AWS Glue。下一節將展示程序的高階概觀。您可以在 [Github](#) 上找到此程序的程式碼範例。

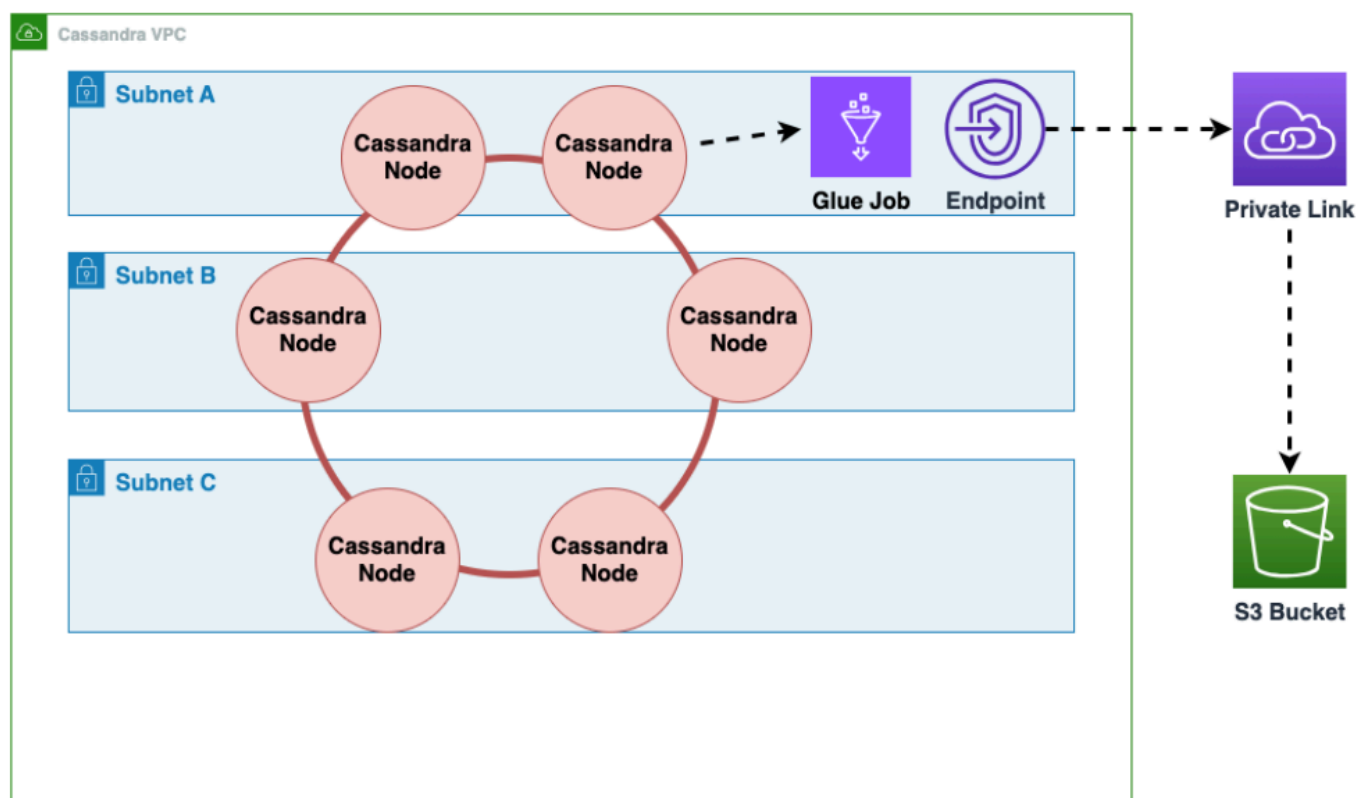
使用 Amazon S3 從 Apache Cassandra 到 Amazon Keyspaces 的離線遷移程序，AWS Glue 需要下列 AWS Glue 任務。

1. 擷取和轉換 CQL 資料並將其存放在 Amazon S3 儲存貯體的 ETL 任務。
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個任務。

如何在 Amazon Virtual Private Cloud 中從在 Amazon EC2 上執行的 Cassandra 執行離線遷移至 Amazon Keyspaces Amazon Virtual Private Cloud

1. 首先，使用從 Cassandra 以 Parquet 格式 AWS Glue 匯出資料表資料，並將其儲存至 Amazon S3 儲存貯體。您需要使用連接至執行 Cassandra 之 Amazon EC2 執行個體所在 VPC 的 AWS Glue 連接器來執行 AWS Glue 任務。然後，使用 Amazon S3 私有端點，您可以將資料儲存到 Amazon S3 儲存貯體。

下圖說明這些步驟。



2. 隨機播放 Amazon S3 儲存貯體中的資料，以改善資料隨機化。平均匯入的資料允許目標資料表中的更多分散式流量。

從具有大型分割區（超過 1000 個資料列的分割區）的 Cassandra 匯出資料時需要此步驟，以避免將資料插入 Amazon Keyspaces 時出現熱鍵模式。熱鍵問題會在 Amazon Keyspaces WriteThrottleEvents 中造成，並導致載入時間增加。



3. 使用另一個 AWS Glue 任務將資料從 Amazon S3 儲存貯體匯入 Amazon Keyspaces。Amazon S3 儲存貯體中的隨機資料會以 Parquet 格式儲存。



如需離線遷移程序的詳細資訊，請參閱研討會 [Amazon Keyspaces with AWS Glue](#)

使用混合遷移解決方案：Apache Cassandra 到 Amazon Keyspaces

下列遷移解決方案可視為線上和離線遷移之間的混合。透過這種混合方法，資料會以近乎即時的方式寫入目的地資料庫，而不會在寫入一致性後提供讀取。這表示新寫入的資料不會立即可用，且預期會發生延遲。如果您需要寫入一致性後的讀取，請參閱 [the section called “線上遷移”](#)。

若要從 Apache Cassandra 近乎即時地遷移至 Amazon Keyspaces，您可以選擇兩種可用的方法。

- CQLReplicator – (建議) CQLReplicator 是 [Github](#) 上提供的開放原始碼公用程式，可協助您近乎即時地將資料從 Apache Cassandra 遷移至 Amazon Keyspaces。

為了判斷要傳播到目的地資料庫的寫入和更新，CQLReplicator 會掃描 Apache Cassandra 字串範圍，並使用 AWS Glue 任務來移除重複的事件，並直接將寫入和更新套用至 Amazon Keyspaces。

- 變更資料擷取 (CDC) – 如果您熟悉 Cassandra CDC，則透過將遞交日誌複製到單獨的 CDC 目錄來允許擷取變更的 Apache Cassandra 內建 CDC 功能是實作混合遷移的另一個選項。

您可以將資料變更複寫至 Amazon Keyspaces，讓 CDC 成為資料遷移案例的替代選項。

如果您在寫入一致性後不需要讀取，則可以使用 CQLReplicator 或 CDC 管道，根據您的偏好和對工具的熟悉程度，將資料從 Apache Cassandra 遷移到 Amazon Keyspaces AWS 服務。使用這些方法來近乎即時地遷移資料，可視為遷移的混合方法，提供線上遷移的替代方案。

此策略被視為混合式方法，因為除了本主題中概述的選項之外，您還必須實作線上遷移進度的一些步驟，例如歷史資料複製和 [線上遷移主題中討論的應用程式遷移策略](#)。

以下章節會詳細說明混合遷移選項。

主題

- [使用 CQLReplicator 遷移資料](#)

- [使用變更資料擷取 \(CDC\) 遷移資料](#)

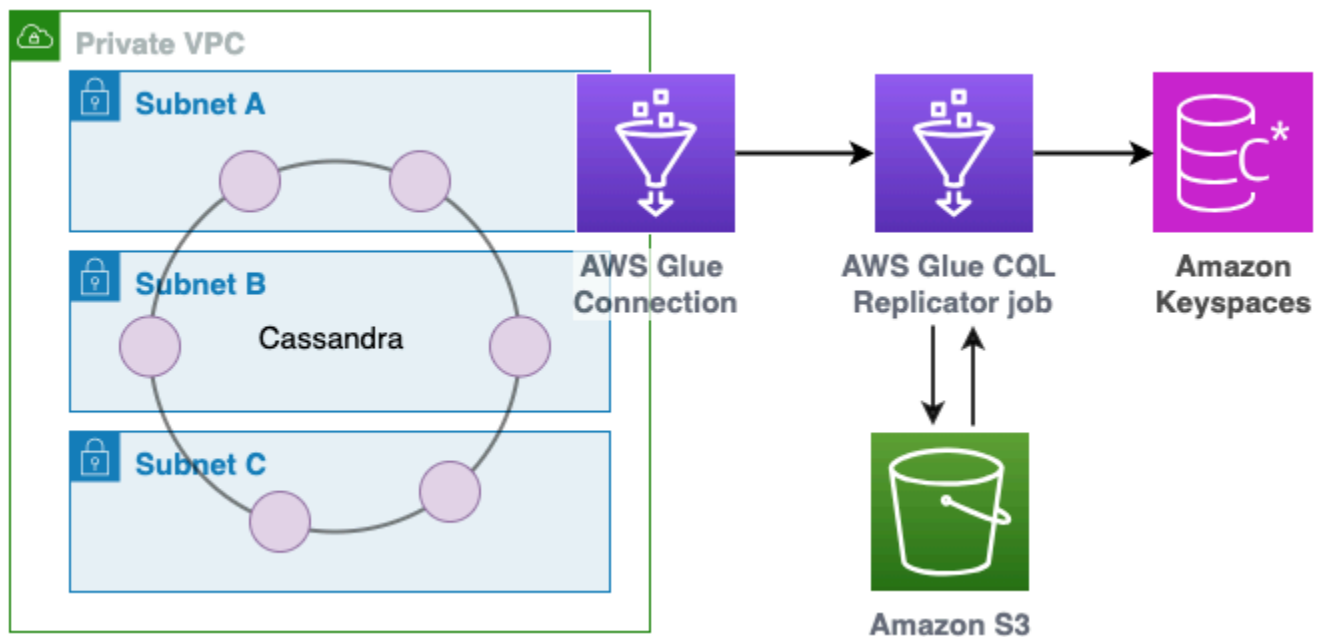
使用 CQLReplicator 遷移資料

使用 [CQLReplicator](#)，您可以使用 CQL 查詢智慧掃描 Cassandra 字環，近乎即時地從 Apache Cassandra 讀取資料。CQLReplicator 不使用 Cassandra CDC，而是實作快取策略，以減少完整掃描的效能懲罰。

為了減少目的地的寫入次數，CQLReplicator 會自動移除重複的複寫事件。使用 CQLReplicator，您可以調整從來源資料庫到目的地資料庫的變更複寫，以便近乎即時地將資料從 Apache Cassandra 遷移到 Amazon Keyspaces。

下圖顯示 CQLReplicator 任務使用的一般架構 AWS Glue。

1. 若要允許存取在私有 VPC 中執行的 Apache Cassandra，請設定與 AWS Glue 連線類型 Network 的連線。
2. 若要使用 CQLReplicator 任務移除重複項目並啟用金鑰快取，請設定 Amazon Simple Storage Service (Amazon S3)。
3. CQLReplicator 任務會將已驗證的來源資料庫直接變更為 Amazon Keyspaces。



如需使用 CQLReplicator 遷移程序的詳細資訊，請參閱下列文章：[AWS 資料庫部落格 使用 CQLReplicator 將 Cassandra 工作負載遷移至 Amazon Keyspaces](#)，AWS 以及 [使用 將 Apache Cassandra 工作負載遷移至 Amazon Keyspaces](#) [AWS Glue](#) 的規範指引。

使用變更資料擷取 (CDC) 遷移資料

如果您已經熟悉使用 [Debezium](#) 設定變更資料擷取 (CDC) 管道，您可以使用此選項將資料遷移到 Amazon Keyspaces，作為使用 CQLReplicator 的替代方案。Debezium 是 CDC 的開放原始碼分散式平台，旨在監控資料庫並可靠地擷取資料列層級的變更。

[Apache Cassandra 的 Debezium 連接器](#) 會將變更上傳至 Amazon Managed Streaming for Apache Kafka (Amazon MSK)，讓下游消費者可以取用和處理這些變更，進而將資料寫入 Amazon Keyspaces。如需詳細資訊，請參閱 [從 Apache Cassandra 到 Amazon Keyspaces 的持續資料遷移指南](#)。

若要解決任何潛在的資料一致性問題，您可以使用 Amazon MSK 實作程序，其中取用者會將 Cassandra 中的金鑰或分割區與 Amazon Keyspaces 中的金鑰或分割區進行比較。

若要成功實作此解決方案，建議您考慮下列事項。

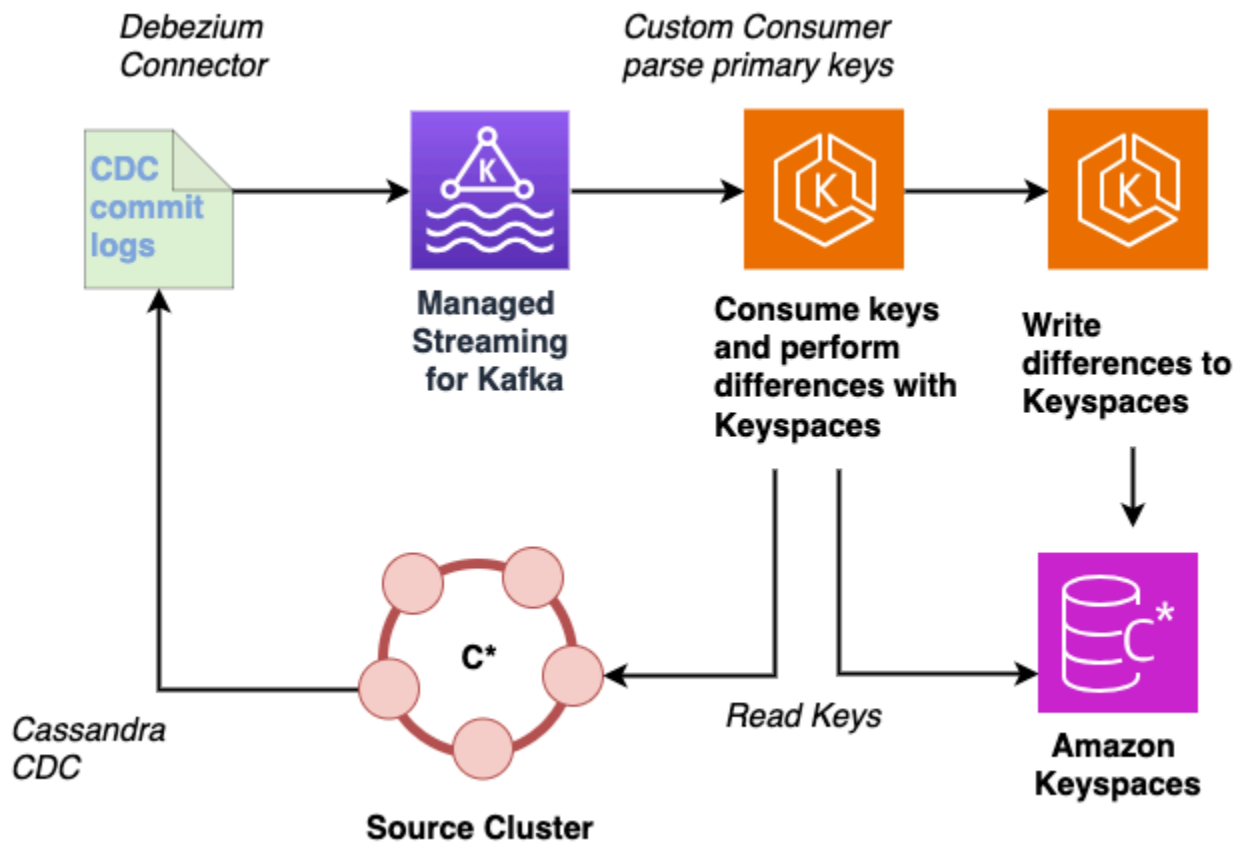
- 如何剖析 CDC 遞交日誌，例如如何移除重複的事件。
- 如何維護 CDC 目錄，例如如何刪除舊日誌。
- 如何處理 Apache Cassandra 中的部分故障，例如，如果寫入在三個複本中只有一個成功。
- 如何處理資源配置，例如增加執行個體的大小，以考量節點上 CDC 程序的額外 CPU、記憶體、DIK 和 IO 需求。

此模式會將來自 Cassandra 的變更視為「提示」，表示金鑰可能已從先前的狀態變更。若要判斷是否有變更要傳播到目的地資料庫，您必須先使用 LOCAL_QUORUM 操作從來源 Cassandra 叢集讀取，以接收最新的記錄，然後將其寫入 Amazon Keyspaces。

在範圍刪除或範圍更新的情況下，您可能需要對整個分割區執行比較，以判斷哪些寫入或更新事件需要寫入目的地資料庫。

如果寫入不等羣，您也需要在寫入 Amazon Keyspaces 之前，將寫入與目的地資料庫中已存在的寫入進行比較。

下圖顯示使用 Debezium 和 Amazon MSK 的 CDC 管道典型架構。



如何選取正確的工具，將資料大量上傳或遷移至 Amazon Keyspaces

在本節中，您可以檢閱可用來大量上傳資料或將資料遷移至 Amazon Keyspaces 的不同工具，並了解如何根據您的需求選取正確的工具。此外，本節提供可用 step-by-step 教學課程的概觀和使用案例，示範如何將資料匯入 Amazon Keyspaces。

若要檢閱將工作負載從 Apache Cassandra 遷移至 Amazon Keyspaces 的可用策略，請參閱 [the section called “從 Cassandra 遷移”](#)。

• 遷移工具

- 透過 Github 上提供的 [Amazon Keyspaces \(適用於 Apache Cassandra\) 定價計算器](#)，您可以根據現有的 Apache Cassandra 工作負載預估 Amazon Keyspaces 的每月成本。輸入來自 Cassandra 節點工具狀態輸出的指標和 Amazon Keyspaces 的預期無伺服器組態，以比較兩個解決方案之間的直接成本。請注意，相較於現有的 Cassandra 部署，此計算器僅著重於 Amazon Keyspaces 的操作成本。它不包括基礎設施維護、營運開銷或 Cassandra 支援成本等總體擁有成本 (TCO) 因素。

- ZDM Dual Write Proxy for Amazon Keyspaces Migration – [Github](#) 上提供的 ZDM Dual Write Proxy 支援從 Apache Cassandra 到 Amazon Keyspaces 的零停機時間遷移。
- CQLReplicator – CQLReplicator 是 [Github](#) 上提供的開放原始碼公用程式，可協助您近乎即時地將資料從 Apache Cassandra 遷移至 Amazon Keyspaces。

如需詳細資訊，請參閱[the section called “CQLReplicator”](#)。

- 若要進一步了解如何使用 Amazon Managed Streaming for Apache Kafka 透過雙寫入實作[線上遷移](#)程序，請參閱[從 Apache Cassandra 到 Amazon Keyspaces 的持續資料遷移指南](#)。
- 對於大型遷移，請考慮使用擷取、轉換和載入 (ETL) 工具。您可以使用 AWS Glue 快速且有效地執行資料轉換遷移。如需詳細資訊，請參閱[the section called “離線移轉”](#)。
- 若要了解如何使用 Apache Cassandra Spark 連接器將資料寫入 Amazon Keyspaces，請參閱 [the section called “與 Apache Spark 整合”](#)。
- 使用 cqlsh COPY FROM命令快速開始將資料載入 Amazon Keyspaces。cqlsh 隨附於 Apache Cassandra，最適合載入小型資料集或測試資料。如需逐步說明，請參閱 [the section called “使用 cqlsh 載入資料”](#)。
- 您也可以使用 DataStax Bulk Loader for Apache Cassandra，使用 dsbulk命令將資料載入 Amazon Keyspaces。DSBulk 提供比 cqlsh 更強大的匯入功能，可從 [GitHub 儲存庫](#)取得。如需逐步說明，請參閱 [the section called “使用 DSBulk 載入資料”](#)。

資料上傳至 Amazon Keyspaces 的一般考量事項

- 將資料上傳細分為較小的元件。

請考慮以下遷移單位及其在原始資料大小方面的潛在足跡。在一或多個階段上傳較少量的資料，可能有助於簡化遷移。

- 依叢集：一次遷移所有 Cassandra 資料。此方法對於較小的叢集可能沒問題。
- 依金鑰空間或資料表 – 將您的遷移分成金鑰空間或資料表群組。此方法可協助您根據每個工作負載的需求分階段遷移資料。
- 依資料 – 考慮遷移特定使用者或產品群組的資料，以進一步縮減資料大小。
- 根據簡單性優先上傳哪些資料。

考慮您是否有可先更輕鬆地遷移的資料，例如，在特定時間不會變更的資料、夜間批次工作的資料、離線時間未使用的資料，或內部應用程式的資料。

- [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

本教學課程將引導您使用 `cqlsh COPY FROM` 命令，將資料從 Apache Cassandra 遷移至 Amazon Keyspaces。`cqlsh COPY FROM` 命令有助於快速且輕鬆地將小型資料集上傳至 Amazon Keyspaces，以供學術或測試之用。如需如何遷移生產工作負載的詳細資訊，請參閱 [the section called “離線移轉”](#)。在本教學課程中，您將完成下列步驟：

先決條件 – 設定具有登入資料的 AWS 帳戶、建立憑證的 JKS 信任存放區檔案，以及設定 `cqlsh` 以連線至 Amazon Keyspaces。

1. 建立來源 CSV 和目標資料表 – 準備 CSV 檔案做為來源資料，並在 Amazon Keyspaces 中建立目標金鑰空間和資料表。
2. 準備資料 – 隨機化 CSV 檔案中的資料並加以分析，以判斷平均和最大資料列大小。
3. 設定輸送量容量 – 根據資料大小和所需的載入時間計算所需的寫入容量單位 (WCUs)，並設定資料表的佈建容量。
4. 設定 `cqlsh` 參數 – 決定 `cqlsh COPY FROM` 參數的最佳值 `NUMPROCESSES`，例如 `INGESTRATE`、`MAXBATCHSIZE`、和 `CHUNKSIZE`，以平均分配工作負載。
5. 執行 **`cqlsh COPY FROM`** 命令 – 執行 `cqlsh COPY FROM` 命令，將資料從 CSV 檔案上傳至 Amazon Keyspaces 資料表，並監控進度。

故障診斷 – 在資料上傳過程中解決常見問題，例如無效請求、剖析器錯誤、容量錯誤和 `cqlsh` 錯誤。

主題

- [先決條件：使用上傳資料之前要完成的步驟 `cqlsh COPY FROM`](#)
- [步驟 1：建立來源 CSV 檔案和目標資料表以進行資料上傳](#)
- [步驟 2：準備來源資料以成功上傳資料](#)
- [步驟 3：設定資料表的輸送量容量](#)
- [步驟 4：設定 `cqlsh COPY FROM` 設定](#)
- [步驟 5：執行 `cqlsh COPY FROM` 命令，將資料從 CSV 檔案上傳至目標資料表](#)
- [疑難排解](#)

先決條件：使用上傳資料之前要完成的步驟 `cqlsh COPY FROM`

您必須先完成下列任務，才能開始本教學課程。

1. 如果您尚未這麼做，AWS 帳戶 請依照 中的步驟註冊 [the section called “設定 AWS Identity and Access Management”](#)。
2. 遵循 中的步驟建立服務特定的登入資料[the section called “建立服務特定的登入資料”](#)。
3. 設定 Cassandra Query Language Shell (cqlsh) 連線，並依照 中的步驟確認您可以連線至 Amazon Keyspaces[the section called “使用 cqlsh”](#)。

步驟 1：建立來源 CSV 檔案和目標資料表以進行資料上傳

在本教學課程中，我們使用逗號分隔值 (CSV) 檔案，其名稱 `keyspaces_sample_table.csv` 為資料遷移的來源檔案。提供的範例檔案包含名為 之資料表的幾列資料 `book_awards`。

1. 建立來源檔案。您可以選擇下列其中一個選項：
 - 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`。
 - 若要使用存放在 Apache Cassandra 資料庫中的自有資料填入 CSV 檔案，您可以使用 `cqlsh COPY TO` 陳述式填入來源 CSV 檔案，如下列範例所示。

```
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 檔案符合下列要求：

- 第一列包含資料欄名稱。
 - 來源 CSV 檔案中的資料欄名稱符合目標資料表中的資料欄名稱。
 - 資料以逗號分隔。
 - 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 Keyspaces 中建立目標金鑰空間和資料表。
 - a. 使用 連線至 Amazon Keyspaces `cqlsh`，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-1.amazonaws.com 9142 -u "111122223333" -p
"wJa1rXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY" --ssl
```

- b. 建立名為 `catalog` 的新金鑰空間，如下列範例所示。

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

- c. 當新的金鑰空間可用時，請使用下列程式碼來建立目標資料表 `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 是原始資料來源，建立具有相符標頭的 Amazon Keyspaces 目標資料表的簡單方法是從來源資料表產生 CREATE TABLE 陳述式，如下列陳述式所示。

```
cqlsh localhost 9042 -u "username" -p "password" --execute "DESCRIBE
TABLE mykeyspace.mytable;"
```

然後，使用與 Cassandra 來源資料表描述相符的資料欄名稱和資料類型，在 Amazon Keyspaces 中建立目標資料表。

步驟 2：準備來源資料以成功上傳資料

準備來源資料以進行有效率的傳輸是兩個步驟。首先，您將資料隨機化。在第二個步驟中，您會分析資料，以判斷適當的 cqlsh 參數值和所需的資料表設定，以確保資料上傳成功。

隨機化資料

cqlsh COPY FROM 命令會以 CSV 檔案中顯示的相同順序讀取和寫入資料。如果您使用 cqlsh COPY TO 命令來建立來源檔案，資料會以 CSV 中的金鑰排序順序寫入。在內部，Amazon Keyspaces 會使用分割區索引鍵分割資料。雖然 Amazon Keyspaces 具有內建邏輯，可協助負載平衡相同分割區

索引鍵的請求，但如果您隨機排列順序，則載入資料會更快且更有效率。這是因為您可以利用 Amazon Keyspaces 寫入不同分割區時發生的內建負載平衡。

若要將寫入平均分散到分割區，您必須隨機化來源檔案中的資料。您可以撰寫應用程式來執行此操作，或使用開放原始碼工具，例如 [Shuf](#)。Shuf 可在 Linux 發行版本、macOS（透過在 [homebrew](#) 中安裝 coreutils）和 Windows（透過使用適用於 Linux 的 Windows 子系統 (WSL)）上免費使用。需要額外的步驟，以防止具有資料欄名稱的標頭列在此步驟中隨機播放。

若要在保留 標頭時隨機化來源檔案，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 會將資料重寫至名為 `keyspace.table.csv` 的新 CSV 檔案。您現在可以刪除 `keyspaces_sample_table.csv` 檔案 - 您不再需要它。

分析資料

透過分析資料來判斷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預估要傳輸的資料總量。
- 您需要平均資料列大小來佈建資料上傳所需的寫入容量。
- 您可以確保每個資料列的大小小於 1 MB，這是 Amazon Keyspaces 中的資料列大小上限。

Note

此配額是指資料列大小，而非分割區大小。與 Apache Cassandra 分割區不同，Amazon Keyspaces 分割區的大小幾乎沒有限制。分割區索引鍵和叢集資料欄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 “估算資料列大小”](#)。

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
```

執行此程式碼會產生下列輸出。

```
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：設定資料表的輸送量容量

本教學課程說明如何調校 `cqlsh`，以在設定的時間範圍內載入資料。由於您知道預先執行的讀取和寫入數量，請使用佈建容量模式。完成資料傳輸後，您應該設定資料表的容量模式，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 [管理無伺服器資源](#)。

使用佈建容量模式時，您可以指定要預先佈建至資料表的讀取和寫入容量。寫入容量會以每小時計費，並以寫入容量單位 (WCUs) 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。當您載入資料時，寫入速率必須低於目標資料表上設定的最大 WCUs (參數：`write_capacity_units`)。

根據預設，您最多可以將 40,000 WCUs 佈建到資料表，並在您帳戶中所有資料表中佈建 80,000 WCUs。如果您需要額外的容量，您可以在 [Service Quotas](#) 主控台中請求增加配額。如需配額的詳細資訊，請參閱 [配額](#)。

計算插入所需的 WCUs 平均數量

每秒插入 1 KB 的資料需要 1 個 WCU。如果您的 CSV 檔案有 360,000 個資料列，而且您想要在 1 小時內載入所有資料，則必須每秒寫入 100 個資料列 (360,000 個資料列 / 60 分鐘 / 60 秒 = 每秒 100 個資料列)。如果每一列最多有 1 KB 的資料，若要每秒插入 100 列，您必須將 100 WCUs 佈建到您的資料表。如果每個資料列有 1.5 KB 的資料，您需要兩個 WCUs 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 WCUs。

若要判斷每秒需要插入一列的 WCUs 數量，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 位元組，您需要三個 WCUs 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資料列數，您可以計算在指定時間內載入資料所需的 WCUs 數量，以及使用不同的 WCU 設定載入 CSV 檔案中所有資料所需的大約時間。

例如，如果檔案中的每一列都是 1 KB，而 CSV 檔案中有 1,000,000 列，若要在 1 小時內載入資料，您需要在該小時內佈建至少 278 個 WCUs 到您的資料表。

```
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或使用 ALTER TABLE CQL 命令時設定資料表的寫入容量設定。以下是使用 ALTER TABLE CQL 陳述式修改資料表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：設定 **cqlsh COPY FROM** 設定

本節概述如何判斷的參數值 **cqlsh COPY FROM**。**cqlsh COPY FROM** 命令會讀取您先前準備的 CSV 檔案，並使用 CQL 將資料插入 Amazon Keyspaces。命令會分割資料列，並在一組工作者之間分配 INSERT 操作。每個工作者都會與 Amazon Keyspaces 建立連線，並沿著此頻道傳送 INSERT 請求。

cqlsh COPY 命令沒有內部邏輯，可在其工作者之間平均分配工作。不過，您可以手動設定它，以確保工作均勻分佈。首先檢閱這些金鑰 **cqlsh** 參數：

- **DELIMITER** – 如果您使用逗號以外的分隔符號，您可以設定此參數，預設為逗號。
- **輸入** – **cqlsh COPY FROM** 每秒嘗試處理的列數目標。如果取消設定，則預設為 100,000。
- **NUMPROCESSES** – **cqlsh COPY FROM** 任務建立的子工作者程序數目。此設定的最大值為 16，預設值為 `num_cores - 1`，其中 `num_cores` 是執行 **cqlsh** 之主機上的處理核心數目。
- **MAXBATCHSIZE** – 批次大小決定單一批次中插入目的地資料表的資料列數目上限。如果取消設定，**cqlsh** 會使用 20 個插入資料列的批次。
- **CHUNKSIZE** – 傳遞給子工作者的工作單位大小。根據預設，它會設定為 5,000。

- MAXATTEMPTS – 重試失敗工作者區塊的次數上限。達到嘗試次數上限後，失敗的記錄會寫入新的 CSV 檔案，您可以在調查失敗之後再次執行。

INGESTRATE 根據您佈建至目標目的地資料表的 WCUs 數目來設定。cqlsh COPY FROM 命令INGESTRATE的不是限制，而是目標平均值。這表示它可以（且通常可以）爆量超過您設定的數字。若要允許爆量，並確保有足夠的容量來處理資料載入請求，請將 INGESTRATE 設為資料表寫入容量的 90%。

```
INGESTRATE = WCUs * .90
```

接著，將 NUMPROCESSES 參數設定為比系統上的核心數目少一個。若要了解系統的核心數量，您可以執行下列程式碼。

```
python -c "import multiprocessing; print(multiprocessing.cpu_count())"
```

在本教學課程中，我們使用下列值。

```
NUMPROCESSES = 4
```

每個程序都會建立工作者，而且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。這表示您必須確保每個工作者每秒處理少於 3,000 個請求。

如同 INGESTRATE，工作者通常會爆量超過您設定的數字，不受時鐘秒的限制。因此，若要考慮爆量，請將 cqlsh 參數設定為以每個工作者為目標，每秒處理 2,500 個請求。若要計算分配給工作者的工作量，請使用下列準則。

- INGESTRATE 除以 NUMPROCESSES。
- 如果 $INGESTRATE / NUMPROCESSES > 2,500$ ，請降低 INGESTRATE 使此公式成為 true。

```
INGESTRATE / NUMPROCESSES <= 2,500
```

在您設定設定以最佳化上傳我們的範例資料之前，讓我們檢閱 cqlsh 預設設定，並了解使用它們如何影響資料上傳程序。由於 cqlsh COPY FROM 使用 CHUNKSIZE 來建立工作區塊 (INSERT 陳述式) 以分發給工作者，因此工作不會自動平均分發。有些工作者可能會閒置，視 INGESTRATE 設定而定。

若要在工作者之間平均分配工作，並將每個工作者保持在最佳每秒 2,500 個請求速率，您必須變更輸入參數 `INGESTRATE` 來設定 `MAXBATCHSIZE`、`CHUNKSIZE` 和 `NUMPROCESSES`。若要在資料載入期間最佳化網路流量使用率，請選擇 `MAXBATCHSIZE` 接近最大值 30 的值。透過將 `CHUNKSIZE` 變更為 100 和 `MAXBATCHSIZE` 25，10,000 個資料列平均分散在四個工作者之間 ($10,000 / 2500 = 4$)。

下列程式碼範例說明了這一點。

```
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
```

若要摘要，請在設定 `cqlsh COPY FROM` 參數時使用下列公式：

- `INGESTRATE` = `write_capacity_units` * .90
- `NUMPROCESSES` = `num_cores` - 1 (預設)
- 擷取 / `NUMPROCESSES` = 2,500 (這必須是 true 陳述式。)
- `MAXBATCHSIZE` = 30 (預設為 20。Amazon Keyspaces 接受最多 30 個批次。)
- 區塊大小 = (擷取 / `NUMPROCESSES`) / `MAXBATCHSIZE`

現在您已計算 `NUMPROCESSES`、`INGESTRATE` 和 `CHUNKSIZE`，您已準備好載入資料。

步驟 5：執行 `cqlsh COPY FROM` 命令，將資料從 CSV 檔案上傳至目標資料表

若要執行 `cqlsh COPY FROM` 命令，請完成下列步驟。

1. 使用 `cqlsh` 連線至 Amazon Keyspaces。
2. 使用下列程式碼選擇您的金鑰空間。

```
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。請參閱下列程式碼。

```
CONSISTENCY LOCAL_QUORUM;
```

4. 使用以下程式碼範例準備 `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. 執行上一個步驟中準備的陳述式。 `cqlsh` 會回傳您已設定的所有設定。

- a. 請確定設定符合您的輸入。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. 檢閱傳輸的資料列數和目前的平均速率，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、執行時間和略過的資料列數），如下列範例所示。

```
15556824 rows imported from 1 files in 8 minutes and 8.321 seconds (0 skipped).
```

在教學課程的最後一個步驟中，您已將資料上傳至 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，以符合應用程式的一般流量模式。在您變更佈建容量之前，會按小時費率產生費用。

疑難排解

資料上傳完成後，請檢查資料列是否已略過。若要這樣做，請導覽至來源 CSV 檔案的來源目錄，並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。如果檔案存在於來源目錄中，且其中有資料，則這些資料列不會上傳至 Amazon Keyspaces。若要重試這些資料列，請先檢查上傳期間遇到的任何錯誤，並相應地調整資料。若要重試這些資料列，您可以重新執行程序。

常見錯誤

未載入資料列的最常見原因是容量錯誤和剖析錯誤。

將資料上傳至 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欄，從 cqlsh COPY 命令產生記錄的批次呼叫。Amazon Keyspaces 不支援已記錄的批次呼叫。

```
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 遷移資料。如需詳細資訊，請參閱[the section called “使用 DSBulk 載入資料”](#)。

將資料上傳至 Amazon Keyspaces 時發生剖析器錯誤

下列範例顯示由於 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符合 Amazon Keyspaces 中的資料表結構描述。檢閱匯入檔案是否有剖析錯誤。您可以使用 INSERT 陳述式嘗試使用單一資料列來隔離錯誤。

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 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 CloudWatch 進行監控”](#)。

將資料上傳至 Amazon Keyspaces 時的 `cqlsh` 錯誤

為了協助對 `cqlsh` 錯誤進行疑難排解，請使用 `--debug` 旗標重新執行失敗命令。

使用不相容的 `cqlsh` 版本時，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，確認已安裝正確的 `cqlsh` 版本。

```
cqlsh --version
```

您應該會看到類似以下內容的輸出。

```
cqlsh 5.0.1
```

如果您使用的是 Windows，請將 的所有執行個體取代 `cqlsh` 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` 用戶端從伺服器收到任何類型的連續三次錯誤後，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符合 Amazon Keyspaces 中的資料表結構描述。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。

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 教學課程將引導您使用 [GitHub](#) 上提供的 DataStax 大量載入器 (DSBulk)，將資料從 Apache Cassandra 遷移至 Amazon Keyspaces。使用 DSBulk 有助於將資料集上傳至 Amazon

Keyspaces 以供學術或測試之用。如需如何遷移生產工作負載的詳細資訊，請參閱 [the section called “離線移轉”](#)。在本教學課程中，您會完成下列步驟。

先決條件 – 使用登入資料設定 AWS 帳戶、為憑證建立 JKS 信任存放區檔案、設定 cqlsh、下載並安裝 DSBulk，以及設定 application.conf 檔案。

1. 建立來源 CSV 和目標資料表 – 準備 CSV 檔案做為來源資料，並在 Amazon Keyspaces 中建立目標金鑰空間和資料表。
2. 準備資料 – 隨機化 CSV 檔案中的資料並加以分析，以判斷平均和最大資料列大小。
3. 設定輸送量容量 – 根據資料大小和所需的載入時間計算所需的寫入容量單位 (WCUs)，並設定資料表的佈建容量。
4. 設定 DSBulk 設定 – 使用身分驗證、SSL/TLS、一致性層級和連線集區大小等設定建立 DSBulk 組態檔案。
5. 執行 DSBulk 載入命令 – 執行 DSBulk 載入命令，將 CSV 檔案的資料上傳至 Amazon Keyspaces 資料表，並監控進度。

主題

- [先決條件：您必須先完成的步驟，才能使用 DSBulk 上傳資料](#)
- [步驟 1：使用 DSBulk 建立來源 CSV 檔案和資料上傳的目標資料表](#)
- [步驟 2：使用 DSBulk 準備要上傳的資料](#)
- [步驟 3：設定目標資料表的輸送量容量](#)
- [步驟 4：設定將資料從 CSV 檔案上傳至目標資料表 DSBulk 的設定](#)
- [步驟 5：執行 DSBulk load 命令，將 CSV 檔案的資料上傳至目標資料表](#)

先決條件：您必須先完成的步驟，才能使用 DSBulk 上傳資料

您必須完成下列任務，才能開始本教學課程。

1. 如果您尚未這麼做，請依照 中的步驟註冊 AWS 帳戶 [the section called “設定 AWS Identity and Access Management”](#)。
2. 遵循 中的步驟建立登入資料 [the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。
3. 建立 JKS 信任存放區檔案。
 - a. 下載下列數位憑證，並將檔案儲存在本機或您的主目錄中。

1. AmazonRootCA1
2. AmazonRootCA2
3. AmazonRootCA3
4. AmazonRootCA4
5. Starfield Class 2 根目錄 (選用 – 用於回溯相容性)

若要下載憑證，您可以使用下列命令。

```
curl -O https://www.amazontrust.com/repository/AmazonRootCA1.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA2.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA3.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA4.pem
curl -O https://certs.secureserver.net/repository/sf-class2-root.crt
```

 Note

Amazon Keyspaces 先前使用錨定至 Starfield 類別 2 CA 的 TLS 憑證。AWS 正在將所有遷移 AWS 區域至根據 Amazon Trust Services (Amazon 根 CAs 發行的憑證。在此轉換期間，請將用戶端設定為信任 Amazon 根 CAs1–4 和 Starfield 根，以確保所有區域的相容性。

- b. 將數位憑證轉換為 trustStore 檔案，並將其新增至金鑰存放區。

```
openssl x509 -outform der -in AmazonRootCA1.pem -out temp_file.der
keytool -import -alias amazon-root-ca-1 -keystore cassandra_truststore.jks -
file temp_file.der

openssl x509 -outform der -in AmazonRootCA2.pem -out temp_file.der
keytool -import -alias amazon-root-ca-2 -keystore cassandra_truststore.jks -
file temp_file.der

openssl x509 -outform der -in AmazonRootCA3.pem -out temp_file.der
keytool -import -alias amazon-root-ca-3 -keystore cassandra_truststore.jks -
file temp_file.der

openssl x509 -outform der -in AmazonRootCA4.pem -out temp_file.der
keytool -import -alias amazon-root-ca-4 -keystore cassandra_truststore.jks -
file temp_file.der
```

```
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
```

在最後一個步驟中，您需要為金鑰存放區建立密碼，並信任每個憑證。互動式命令看起來像這樣。

```
Enter keystore password:
Re-enter new password:
Owner: CN=Amazon Root CA 1, O=Amazon, C=US
Issuer: CN=Amazon Root CA 1, O=Amazon, C=US
Serial number: 66c9fcf99bf8c0a39e2f0788a43e696365bca
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sun Jan 17 00:00:00 UTC 2038
Certificate fingerprints:
    SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
    SHA256:
      8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
  CA:true
  PathLen:2147483647
]

#2: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
  DigitalSignature
  Key_CertSign
  Crl_Sign
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 84 18 CC 85 34 EC BC 0C   94 94 2E 08 59 9C C7 B2   ....4.....Y...
0010: 10 4E 0A 08                               .N..
]
```

```
]
```

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

```
Certificate was added to keystore
```

```
Enter keystore password:
```

```
Owner: CN=Amazon Root CA 2, O=Amazon, C=US
```

```
Issuer: CN=Amazon Root CA 2, O=Amazon, C=US
```

```
Serial number: 66c9fd29635869f0a0fe58678f85b26bb8a37
```

```
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sat May 26 00:00:00 UTC 2040
```

```
Certificate fingerprints:
```

```
SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
```

```
SHA256:
```

```
1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E
```

```
Signature algorithm name: SHA384withRSA
```

```
Subject Public Key Algorithm: 4096-bit RSA key
```

```
Version: 3
```

```
Extensions:
```

```
#1: ObjectId: 2.5.29.19 Criticality=true
```

```
BasicConstraints:[
```

```
CA:true
```

```
PathLen:2147483647
```

```
]
```

```
#2: ObjectId: 2.5.29.15 Criticality=true
```

```
KeyUsage [
```

```
DigitalSignature
```

```
Key_CertSign
```

```
Crl_Sign
```

```
]
```

```
#3: ObjectId: 2.5.29.14 Criticality=false
```

```
SubjectKeyIdentifier [
```

```
KeyIdentifier [
```

```
0000: B0 0C F0 4C 30 F4 05 58 02 48 FD 33 E5 52 AF 4B ...L0..X.H.3.R.K
```

```
0010: 84 E3 66 52 ..fR
```

```
]
```

```
]
```

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

```
Certificate was added to keystore
```

```
Enter keystore password:
```

```
Owner: CN=Amazon Root CA 3, O=Amazon, C=US
```



```

Issuer: CN=Amazon Root CA 3, O=Amazon, C=US
Serial number: 66c9fd5749736663f3b0b9ad9e89e7603f24a
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sat May 26 00:00:00 UTC 2040
Certificate fingerprints:
    SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
    SHA256:
18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4
Signature algorithm name: SHA256withECDSA
Subject Public Key Algorithm: 256-bit EC (secp256r1) key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
    CA:true
    PathLen:2147483647
]

#2: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
    DigitalSignature
    Key_CertSign
    Crl_Sign
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: AB B6 DB D7 06 9E 37 AC   30 86 07 91 70 C7 9C C4   .....7.0...p...
0010: 19 B1 78 C0                               ...x.
]
]

Trust this certificate? [no]: yes
Certificate was added to keystore
Enter keystore password:
Owner: CN=Amazon Root CA 4, O=Amazon, C=US
Issuer: CN=Amazon Root CA 4, O=Amazon, C=US
Serial number: 66c9fd7c1bb104c2943e5717b7b2cc81ac10e
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sat May 26 00:00:00 UTC 2040
Certificate fingerprints:
    SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE

```

```

SHA256:
E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25
Signature algorithm name: SHA384withECDSA
Subject Public Key Algorithm: 384-bit EC (secp384r1) key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
    CA:true
    PathLen:2147483647
]

#2: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
    DigitalSignature
    Key_CertSign
    Crl_Sign
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
    KeyIdentifier [
0000: D3 EC C7 3A 65 6E CC E1   DA 76 9A 56 FB 9C F3 86   ...:en...v.V....
0010: 6D 57 E5 81                               mW..
    ]
]

Trust this certificate? [no]: yes
Certificate was added to keystore
Enter keystore 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:
    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 (weak)
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                                ....
  ]
]

Warning:
The input uses the SHA1withRSA signature algorithm which is considered a
security risk. This algorithm will be disabled in a future update.

Trust this certificate? [no]: yes
Certificate was added to keystore

```

4. 設定 Cassandra Query Language shell (cqlsh) 連線，並依照 [the section called “使用 cqlsh”](#) 中的步驟確認您可以連線至 Amazon Keyspaces。
5. 下載並安裝 DSBulk。

 Note

本教學課程中顯示的版本可能不是最新版本。下載 DSBulk 之前，請檢查 [DataStax Bulk Loader 下載頁面](#) 以取得最新版本，並相應地更新下列命令中的版本編號。

- a. 若要下載 DSBulk，您可以使用下列程式碼。

```
curl -OL https://downloads.datastax.com/dsbulk/dsbulk-1.8.0.tar.gz
```

- b. 然後解壓縮 tar 檔案，並將 DSBulk 新增至您的 PATH，如下列範例所示。

```
tar -zxvf dsbulk-1.8.0.tar.gz
# add the DSBulk directory to the path
export PATH=$PATH:./dsbulk-1.8.0/bin
```

- c. 建立 application.conf 檔案以存放 DSBulk 要使用的設定。您可以將下列範例儲存為 ./dsbulk_keyspaces.conf。如果您不在本機節點上，請將 localhost 取代為本機 Cassandra 叢集的聯絡點，例如 DNS 名稱或 IP 地址。請記下檔案名稱和路徑，因為您稍後需要在 dsbulk load 命令中指定此項目。

```
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 [GitHub](#) 下載著色 jar 檔案，並將其放在 DSBulk lib 資料夾中，如下列範例所示。

```
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：使用 DSBulk 建立來源 CSV 檔案和資料上傳的目標資料表

在本教學課程中，我們使用逗號分隔值 (CSV) 檔案，其名稱 `keyspaces_sample_table.csv` 為資料遷移的來源檔案。提供的範例檔案包含名為 `book_awards` 之資料表的幾列資料。

1. 建立來源檔案。您可以選擇下列其中一個選項：

- 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`。
- 若要使用存放在 Apache Cassandra 資料庫中的自有資料填入 CSV 檔案，您可以使用 填入來源 CSV 檔案 `dsbulk unload`，如下列範例所示。

```
dsbulk unload -k mykeyspace -t mytable -f ./my_application.conf  
> keyspace_sample_table.csv
```

請確定您建立的 CSV 檔案符合下列要求：

- 第一列包含資料欄名稱。
- 來源 CSV 檔案中的資料欄名稱符合目標資料表中的資料欄名稱。
- 資料以逗號分隔。
- 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 Keyspaces 中建立目標金鑰空間和資料表。

- a. 使用 連線至 Amazon Keyspaces `cqlsh`，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-1.amazonaws.com 9142 -u "111122223333" -p  
"wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY" --ssl
```

- b. 建立名為 `catalog` 的新金鑰空間，如下列範例所示。

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

- c. 在新的金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`。若要進一步了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 “檢查金鑰空間建立狀態”](#)。

```
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 是原始資料來源，使用相符的標頭建立 Amazon Keyspaces 目標資料表的簡單方法是從來源資料表產生 CREATE TABLE 陳述式，如下列陳述式所示。

```
cqlsh localhost 9042 -u "username" -p "password" --execute "DESCRIBE  
TABLE mykeyspace.mytable;"
```

然後使用與 Cassandra 來源資料表中的描述相符的資料欄名稱和資料類型，在 Amazon Keyspaces 中建立目標資料表。

步驟 2：使用 DSBulk 準備要上傳的資料

準備來源資料以進行有效率的傳輸是兩個步驟。首先，您將資料隨機化。在第二個步驟中，您會分析資料，以判斷適當的 dsbulk 參數值和所需的資料表設定。

隨機化資料

dsbulk 命令會以 CSV 檔案中顯示的相同順序讀取和寫入資料。如果您使用 dsbulk 命令來建立來源檔案，資料會以 CSV 中的金鑰排序順序寫入。在內部，Amazon Keyspaces 會使用分割區索引鍵分割資料。雖然 Amazon Keyspaces 具有內建邏輯，可協助負載平衡相同分割區索引鍵的請求，但如果您隨機排列順序，則載入資料會更快、更有效率。這是因為您可以利用 Amazon Keyspaces 寫入不同分割區時發生的內建負載平衡。

若要將寫入平均分散到分割區，您必須隨機化來源檔案中的資料。您可以撰寫應用程式來執行此操作，或使用開放原始碼工具，例如 [Shuf](#)。Shuf 可在 Linux 發行版本、macOS（透過在 [homebrew](#) 中安裝 coreutils）和 Windows（透過使用適用於 Linux 的 Windows 子系統 (WSL)）上免費使用。需要額外的步驟，以防止具有資料欄名稱的標頭列在此步驟中隨機播放。

若要在保留 標頭時隨機化來源檔案，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 會將資料重寫至名為 `keyspace.table.csv` 的新 CSV 檔案。您現在可以刪除 `keyspaces_sample_table.csv` 檔案 - 您不再需要它。

分析資料

透過分析資料來判斷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預估要傳輸的資料總量。
- 您需要平均資料列大小來佈建資料上傳所需的寫入容量。
- 您可以確保每個資料列的大小小於 1 MB，這是 Amazon Keyspaces 中的資料列大小上限。

Note

此配額是指資料列大小，而不是分割區大小。與 Apache Cassandra 分割區不同，Amazon Keyspaces 分割區的大小幾乎沒有限制。分割區索引鍵和叢集資料欄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 “估算資料列大小”](#)。

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
```

執行此程式碼會產生下列輸出。

```
using 10,000 samples:  
{lines: 10000, avg: 123 bytes, max: 225 bytes}
```

請確定您的資料列大小上限不超過 1 MB。如果是這樣，您必須分解資料列或壓縮資料，使資料列大小低於 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：設定目標資料表的輸送量容量

本教學課程說明如何調校 DSBulk，以在設定的時間範圍內載入資料。由於您知道預先執行的讀取和寫入數量，請使用佈建的容量模式。完成資料傳輸後，您應該設定資料表的容量模式，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 [管理無伺服器資源](#)。

使用佈建容量模式時，您可以指定要預先佈建至資料表的讀取和寫入容量。寫入容量會以每小時計費，並以寫入容量單位 (WCUs 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。當您載入資料時，寫入速率必須低於目標資料表上設定的最大 WCUs (參數：write_capacity_units)。

根據預設，您最多可以將 40,000 WCUs 佈建到資料表，並在您帳戶中所有資料表中佈建 80,000 WCUs。如果您需要額外的容量，您可以在 [Service Quotas](#) 主控台中請求增加配額。如需配額的詳細資訊，請參閱 [配額](#)。

計算插入所需的 WCUs 平均數量

每秒插入 1 KB 的資料需要 1 個 WCU。如果您的 CSV 檔案有 360,000 個資料列，而且您想要在 1 小時內載入所有資料，則必須每秒寫入 100 個資料列 (360,000 個資料列/60 分鐘/60 秒 = 每秒 100 個資料列)。如果每一列最多有 1 KB 的資料，若要每秒插入 100 列，您必須將 100 WCUs 佈建到您的資料表。如果每個資料列有 1.5 KB 的資料，您需要兩個 WCUs 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 WCUs。

若要判斷每秒需要插入一列的 WCUs 數量，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 位元組，您需要三個 WCUs 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資料列數，您可以計算在指定時間內需要載入資料的 WCUs 數量，以及使用不同的 WCU 設定載入 CSV 檔案中所有資料所需的大約時間。

例如，如果檔案中的每一列都是 1 KB，而且 CSV 檔案中有 1,000,000 列，若要在 1 小時內載入資料，您需要在該小時內將至少 278 WCUs 佈建至資料表。


```
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或使用 ALTER TABLE 命令時設定資料表的寫入容量設定。以下是使用 ALTER TABLE 命令修改資料表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，請參閱 [the section called “CREATE TABLE”](#) 和 [the section called “ALTER TABLE”](#)。

步驟 4：設定將資料從 CSV 檔案上傳至目標資料表 DSBulk 的設定

本節概述設定 DSBulk 以將資料上傳至 Amazon Keyspaces 所需的步驟。您可以使用組態檔案來設定 DSBulk。您直接從命令列指定組態檔案。

1. 建立 DSBulk 組態檔案以遷移至 Amazon Keyspaces，在此範例中，我們使用檔案名稱 `dsbulk_keyspaces.conf`。在 DSBulk 組態檔案中指定下列設定。
 - a. *PlainTextAuthProvider* – 使用 *PlainTextAuthProvider* 類別建立身分驗證提供者。*ServiceUserName* 和 *ServicePassword* 應符合您在產生服務特定登入資料時，依照 中的步驟取得的使用者名稱和密碼 [the section called “建立程式設計存取憑證”](#)。
 - b. *local-datacenter* – 將 `local-datacenter` 設定為您要連線 AWS 區域的。例如，如果應用程式正在連線至 `cassandra.us-east-1.amazonaws.com`，請將本機資料中心設定為 `us-east-1`。如需所有可用的 AWS 區域，請參閱 [the section called “服務端點”](#)。若要避免複本，請將 `slow-replica-avoidance` 設定為 `false`。
 - c. *SSLEngineFactory* – 若要設定 SSL/TLS，*SSLEngineFactory* 請在組態檔案中新增區段以使用指定類別的單行來初始化 `class = DefaultSslEngineFactory`。提供的路徑 `cassandra_truststore.jks` 和您先前建立的密碼。
 - 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 “支援的 Cassandra 一致性層級”](#)。
 - e. 每個集區的連線數可在 Java 驅動程式中設定。在此範例中，將 `advanced.connection.pool.local.size` 設定為 3。

以下是完整的範例組態檔案。

```
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
  }

  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* – 負載命令每秒嘗試同時處理的列數上限。如果取消設定，則會使用 -1 停用此設定。

executor.maxPerSecond 根據您佈建至目標目的地資料表的 WCUs 數目來設定。load 命令 *executor.maxPerSecond* 的不是限制，而是目標平均值。這表示它可以（且通常可以）爆量超過您設定的數字。若要允許爆量，並確保有足夠的容量來處理資料載入請求，請將 *executor.maxPerSecond* 設為資料表寫入容量的 90%。

```
executor.maxPerSecond = WCUs * .90
```

在本教學課程中，我們會將 `executor.maxPerSecond` 設定為 5。

 Note

如果您使用的是 DSBulk 1.6.0 或更高版本，則可以 `dsbulk.engine.maxConcurrentQueries` 改用。

b. 為 DSBulk load 命令設定這些額外參數。

- `batch-mode` – 此參數會告知系統依分割區索引鍵將操作分組。建議您停用批次模式，因為它可能會導致快速鍵案例並導致 `WriteThrottleEvents`。
- `driver.advanced.retry-policy-max-retries` – 這會決定重試失敗查詢的次數。如果取消設定，則預設值為 10。您可以視需要調整此值。
- `driver.basic.request.timeout` – 系統等待查詢傳回的時間，以分鐘為單位。如果取消設定，則預設值為「5 分鐘」。您可以視需要調整此值。

步驟 5：執行 DSBulk **load** 命令，將 CSV 檔案的資料上傳至目標資料表

在本教學課程的最後一個步驟中，您會將資料上傳至 Amazon Keyspaces。

若要執行 DSBulk load 命令，請完成下列步驟。

1. 執行下列程式碼，將資料從 csv 檔案上傳至 Amazon Keyspaces 資料表。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. 輸出包含日誌檔案的位置，其中會詳細說明成功和失敗的操作。檔案會存放在下列目錄中。

```
Operation directory: /home/user_name/logs/UNLOAD_20210308-202317-801911
```

3. 日誌檔案項目將包含指標，如下列範例所示。檢查以確保資料列數目與 csv 檔案中的資料列數目一致。

```
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，以符合應用程式的一般流量模式。在您變更佈建容量之前，會依每小時費率產生費用。如需詳細資訊，請參閱[the section called “設定讀取/寫入容量模式”](#)。

存取 Amazon Keyspaces (適用於 Apache Cassandra)

您可以使用主控台 AWS CloudShell、透過執行 `cqlsh` 用戶端、軟體 AWS 開發套件或使用 Apache 2.0 授權的 Cassandra 驅動程式，以程式設計方式存取 Amazon Keyspaces。Amazon Keyspaces 支援與 Apache Cassandra 3.11.2 相容的驅動程式和用戶端。存取 Amazon Keyspaces 之前，您必須完成設定，AWS Identity and Access Management 然後將 IAM 身分存取許可授予 Amazon Keyspaces。

設定 AWS Identity and Access Management

註冊 AWS 帳戶

如果您沒有 AWS 帳戶，請完成下列步驟來建立一個。

註冊 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或簡訊，並在電話鍵盤輸入驗證碼。

當您註冊時 AWS 帳戶，AWS 帳戶根使用者會建立。根使用者有權存取該帳戶中的所有 AWS 服務和資源。作為安全最佳實務，請將管理存取權指派給使用者，並且僅使用根使用者來執行 [需要根使用者存取權的任務](#)。

AWS 會在註冊程序完成後傳送確認電子郵件給您。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。

建立具有管理存取權的使用者

註冊後 AWS 帳戶，請保護 AWS 帳戶根使用者、啟用 AWS IAM Identity Center 和建立管理使用者，以免將根使用者用於日常任務。

保護您的 AWS 帳戶根使用者

1. 選擇根使用者並輸入 AWS 帳戶您的電子郵件地址，以帳戶擁有者 [AWS 管理主控台](#) 身分登入。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)。

如需說明，請參閱《IAM 使用者指南》中的[為您的 AWS 帳戶 根使用者（主控台）啟用虛擬 MFA 裝置](#)。

建立具有管理存取權的使用者

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 Identity Center 中，將管理存取權授予使用者。

如需使用 IAM Identity Center 目錄 做為身分來源的教學課程，請參閱AWS IAM Identity Center 《使用者指南》中的[使用預設值設定使用者存取 IAM Identity Center 目錄](#)。

以具有管理存取權的使用者身分登入

- 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 Identity Center 使用者登入的說明，請參閱AWS 登入 《使用者指南》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 Identity Center 中，建立一個許可集來遵循套用最低權限的最佳實務。

如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的[建立許可集](#)。

2. 將使用者指派至群組，然後對該群組指派單一登入存取權。

如需指示，請參閱《AWS IAM Identity Center 使用者指南》中的[新增群組](#)。

設定 Amazon Keyspaces

使用 [IAM](#) 管理對 Amazon Keyspaces 資源的存取。使用 IAM，您可以將政策連接至 IAM 使用者、角色和聯合身分，以授予 Amazon Keyspaces 中特定資源的讀取和寫入許可。

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

- [AmazonKeyspacesFullAccess](#) – 此政策授予許可，以存取 Amazon Keyspaces 中的所有資源，並完整存取所有功能。
- [AmazonKeyspacesReadOnlyAccess_v2](#) – 此政策會將唯讀許可授予 Amazon Keyspaces。

如需 受管政策中定義之動作的詳細說明，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。您也可以限制對特定金鑰空間或資料表的存取。如需如何限制動作或限制存取 Amazon Keyspaces 中特定資源的詳細資訊，請參閱 [the section called “Amazon Keyspaces 如何與 IAM 搭配使用”](#)。

若要在建立 AWS 帳戶 並建立授予 IAM 身分存取 Amazon Keyspaces 的政策之後存取 Amazon Keyspaces，請繼續下列其中一個區段：

- [使用主控台](#)
- [使用 AWS CloudShell](#)

使用主控台存取 Amazon Keyspaces

您可以在 存取 Amazon Keyspaces 的主控台 <https://console.aws.amazon.com/keyspaces/home>。如需 AWS 管理主控台 存取的詳細資訊，請參閱 [《IAM 使用者指南》中的控制 IAM 使用者對 的存取 AWS 管理主控台](#)。

您可以使用 主控台在 Amazon Keyspaces 中執行下列動作：

- 建立、刪除和管理金鑰空間和資料表。
- 在資料表的監控標籤上監控重要資料表指標：
 - 計費資料表大小（位元組）
 - 容量指標
- 使用 CQL 編輯器或透過 [整合式 AWS CloudShell 體驗](#) 執行查詢，例如插入、更新和刪除資料。
- 透過整合 AWS CloudShell 的 CQLSH 支援，直接連線至 Amazon Keyspaces。
- 變更帳戶的分割區組態。
- 在儀表板上檢視帳戶的效能和錯誤指標。

AWS CloudShell 從主控台使用 連線至 Amazon Keyspaces

AWS CloudShell 提供從主控台直接連線至 Amazon Keyspaces 的簡化方法。AWS CloudShell 整合會自動 `cqlsh-expansion` 為您初始化，並建立與特定金鑰空間的連線。

AWS CloudShell 整合入門

1. 存取 AWS CloudShell 整合 – 從 Amazon Keyspaces 主控台導覽至您要連線的金鑰空間，然後選擇連線至金鑰空間 AWS CloudShell 按鈕。
2. 預先填入的設定命令 – 當您選擇 AWS CloudShell 按鈕時，系統會提供預先填入的命令，其中包含必要的初始化和連線參數：

```
cqlsh-expansion.init ; cqlsh-expansion cassandra.region.amazonaws.com 9142 --ssl  
-my_keyspace
```

3. 按一下執行
4. 等待 Amazon Keyspaces 建立連線 – 讓設定程序有時間完成，並建立與所選金鑰空間的連線。
5. 執行命令 – 連線時，您可以直接針對選取的金鑰空間寫入和執行 CQL 命令。

可用的 CQLSH 命令

AWS CloudShell 整合可讓您存取標準 CQLSH 功能，包括支援基本 CQL 以外的命令。

- 描述 – 列出和檢視鍵空間和資料表的詳細說明。
- 複製 – 將查詢輸出匯出至檔案以供日後處理。
- 來源 – 從檔案執行 CQL 陳述式。
- 命令歷史記錄導覽 – 使用鍵盤快速鍵捲動先前的命令。
- 查詢執行 – 直接從鍵盤界面執行查詢。
- 輸出複製 – 將結果複製為簡單文字，以用於其他應用程式。

功能考量

使用 AWS CloudShell 整合時，請考慮下列取捨。

不支援下列功能：

- CQL 語法建議和自動完成功能。

- 編輯器工具，例如顯示自動完成、顯示工具提示、尋找、尋找和取代、重做、切換區塊註解、切換程式碼摺疊、切換行註解和復原。

修改的功能：

- CSV 下載 – 若要以 CSV 格式下載結果，請指定 > 運算子到管道命令輸出：

```
SELECT * FROM my_table > file_name.csv
```

- JSON 檢視 - 您可以 JSON 格式檢視項目，而不需要資料轉換。
- 終端機式界面 - 體驗以終端機界面運作，而不是功能完整的程式碼編輯器。

AWS CloudShell 整合可簡化連線程序，同時提供直接從 管理 Amazon Keyspaces 資料所需的 CQL 功能 AWS 管理主控台。

若要了解如何建立 Amazon Keyspaces 金鑰空間和資料表，並使用範例應用程式資料進行設定，請參閱 [Amazon Keyspaces 入門 \(適用於 Apache Cassandra\)](#)。

使用 AWS CloudShell 存取 Amazon Keyspaces

AWS CloudShell 是以瀏覽器為基礎的預先驗證 Shell，您可以直接從 啟動 AWS 管理主控台。您可以使用您偏好的 shell (Bash、PowerShell 或 Z shell) 對 AWS 服務執行 AWS CLI 命令。使用您的主控台登入資料 AWS CloudShell 進行預先驗證，並包含使用 AWS 服務的常用工具，包括 AWS CLI、Python 和 Node.js。

您[AWS CloudShell 從 啟動 AWS 管理主控台](#)，而且您用來登入主控台的 AWS 登入資料會自動在新的 shell 工作階段中使用。此預先驗證 AWS CloudShell 使用者可讓您在使用 cqlsh 或 第 2 AWS CLI 版（預先安裝在 shell 的運算環境）與 Amazon Keyspaces 等 AWS 服務互動時，略過設定登入資料。

AWS CloudShell 整合選項

Amazon Keyspaces 提供兩種使用方式 AWS CloudShell：

1. 整合 AWS CloudShell 體驗 - 直接從 Amazon Keyspaces 主控台提供的簡化連線方法，可自動初始化 cqlsh-expansion 並連接至特定的金鑰空間。如需使用此整合體驗的完整說明，請參閱 [the section called “CloudShell 主控台整合”](#)。
2. 手動 AWS CloudShell 設定 - 在 cqlsh 中手動安裝和設定 AWS CloudShell。這些步驟會在以下各節中說明，並提供更多對設定程序的控制，以及啟用自訂組態。

先決條件

搭配使用 Amazon Keyspaces 之前 AWS CloudShell，請確定您有：

- 可存取 Amazon Keyspaces 和 AWS 的帳戶 AWS CloudShell
- 使用這兩種服務的適當 IAM 許可
- 對 Cassandra 查詢語言 (CQL) 的基本熟悉度

入門 AWS CloudShell

若要存取 AWS CloudShell：

1. 登入 AWS 管理主控台
2. AWS CloudShell 選擇主控台頂端導覽列中的 AWS CloudShell 圖示，或導覽至 <https://console.aws.amazon.com/cloudshell/> 來啟動
3. 等待 AWS CloudShell 環境初始化

取得的 IAM 許可 AWS CloudShell

AWS Identity and Access Management 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，讓他們可以存取 AWS CloudShell 和使用環境的功能。

管理員授予使用者存取權的最快速方法是透過 AWS 受管政策。[AWS 受管政策](#)是由 AWS 建立並管理的獨立政策。下列 CloudShell 受 AWS 管政策可以連接到 IAM 身分：

- `AWSCloudShellFullAccess`：授予許可，以使用 AWS CloudShell 並完整存取所有功能。

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。如需限制 CloudShell 中使用者可用的動作的詳細資訊，請參閱 AWS CloudShell 《使用者指南》中的[使用 IAM 政策管理 AWS CloudShell 存取和用量](#)。

Note

您的 IAM 身分也需要政策，授予對 Amazon Keyspaces 進行呼叫的許可。

您可以使用 AWS 受管政策為您的 Amazon Keyspaces 提供 IAM 身分存取權，或從 受管政策作為範本開始，並移除您不需要的許可。您也可以限制對特定金鑰空間和資料表的存取，以建立自訂政策。下列 Amazon Keyspaces 的受管政策可以連接到 IAM 身分：

- [AmazonKeyspacesFullAccess](#) – 此政策授予許可，以使用具有所有功能完整存取權的 Amazon Keyspaces。

如需受管政策中定義之動作的詳細說明，請參閱 [the section called “AWS 受管政策”](#)。

如需如何限制動作或限制存取 Amazon Keyspaces 中特定資源的詳細資訊，請參閱 [the section called “Amazon Keyspaces 如何與 IAM 搭配使用”](#)。

安裝 cqlsh 用戶端

AWS CloudShell 不會隨附 cqlsh 預先安裝的。您需要安裝它才能與 Amazon Keyspaces 互動。您可以安裝的 Amazon Keyspaces 特定版本 cqlsh。如需 cqlsh-expansion 安裝說明，請參閱 [the section called “使用 cqlsh-expansion”](#)。

設定 TLS 憑證

若要使用 TLS 連線連線至 Amazon Keyspaces，您需要下載所需的 Amazon 數位憑證。如需下載和設定 SSL 憑證的詳細說明，請參閱 [the section called “如何手動設定 TLS 的 cqlsh 連線”](#)。

連線至 Amazon Keyspaces

您可以使用 IAM 登入資料（建議）或服務特定的登入資料來連線至 Amazon Keyspaces。

使用 IAM 登入資料連線（建議）

AWS CloudShell 已使用您的 IAM 登入資料進行身分驗證。若要使用這些登入資料進行連線，您可以使用 SigV4 身分驗證外掛程式。如需搭配使用 IAM 登入資料的詳細指示 cqlsh，請參閱 [the section called “使用 cqlsh-expansion”](#)。

使用服務特定的登入資料進行連線

如果您已為 Amazon Keyspaces 建立服務特定的登入資料，您可以使用它們進行連線。如需建立和使用服務特定登入資料的資訊，請參閱 [the section called “建立服務特定的登入資料”](#)。

搭配 Amazon Keyspaces 使用 CQLSH

連線時，您可以使用標準 CQL 命令與金鑰空間和資料表互動。如需支援的 CQL 命令和操作的詳細資訊，請參閱 [the section called “使用 cqlsh-expansion”](#)。

在 中保留資料 AWS CloudShell

AWS CloudShell 會在您的主目錄中提供 1 GB 的持久性儲存。這表示您建立的指令碼、憑證和組態檔案在工作階段之間仍然可用。您可以為建立組態檔案 `cqlsh`，以簡化連線。

使用組態檔案，您可以執行 `cqlsh` 而無需指定其他參數。如需建立和使用 `cqlsh` 組態檔案的詳細資訊，請參閱 [the section called “使用 cqlsh-expansion”](#)。

最佳實務

搭配使用 Amazon Keyspaces 時 AWS CloudShell，我們建議下列最佳實務：

- 建立重複任務的指令碼，並將其存放在您的持久性主目錄中。
- 將環境變數用於敏感資訊，而不是硬式編碼。
- 請記住，AWS CloudShell 工作階段可能會在閒置後逾時。
- 請注意 AWS CloudShell 服務配額和限制。
- 考慮使用搭配 Amazon Keyspaces AWS CLI 進行自動化操作。

疑難排解

如果您在從下列位置連線至 Amazon Keyspaces 時遇到問題 AWS CloudShell：

- 確認您的 IAM 許可包含必要的 Amazon Keyspaces 動作。
- 請確定您使用的是區域的正確端點。
- 檢查 SSL 憑證是否已正確下載並參考。
- 確認您的服務特定登入資料正確且未過期。
- 如果使用 `cqlshrc` 檔案，請檢查其格式和許可。

範例：如何使用 與 Amazon Keyspaces 互動 AWS CloudShell

AWS CloudShell 從 啟動後 AWS 管理主控台，您可以立即開始使用 `cqlsh` 或命令列界面與 Amazon Keyspaces 互動。如果您尚未安裝 `cqlsh-expansion`，請參閱 [the section called “使用 cqlsh-expansion”](#) 以取得詳細步驟。

Note

在 `cqlsh-expansion` 中使用 時 AWS CloudShell，您不需要在呼叫之前設定登入資料，因為您已在 shell 中驗證。

連線至 Amazon Keyspaces 並建立新的金鑰空間。然後從系統資料表讀取，以確認使用 建立金鑰空間 AWS CloudShell

1. 從 中 AWS 管理主控台，您可以選擇導覽列上可用的下列選項來啟動 CloudShell：
 - 選擇 CloudShell 圖示。
 - 開始cloudshell在搜尋方塊中輸入，然後選擇 CloudShell 選項。
2. 您可以使用下列命令建立與 Amazon Keyspaces 的連線。請務必 `cassandra.us-east-1.amazonaws.com` 將 取代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，您應該會看到類以下列範例的輸出。

```
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. 使用名稱 建立新的金鑰空間 `mykeyspace`。您可以使用下列命令來執行此操作。

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

4. 若要確認金鑰空間已建立，您可以使用下列命令從系統資料表讀取。

```
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)
```

建立 Amazon Keyspaces 的程式設計存取憑證

若要為使用者和應用程式提供以程式設計方式存取 Amazon Keyspaces 資源的登入資料，您可以執行下列其中一項操作：

- 建立與 Cassandra 用於身分驗證和存取管理的傳統使用者名稱和密碼類似的服務特定登入資料。AWS 服務特定登入資料會與特定 AWS Identity and Access Management (IAM) 使用者建立關聯，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》中的搭配使用 IAM 與 Amazon Keyspaces \(適用於 Apache Cassandra\)](#)。

Warning

IAM 使用者具有長期憑證，這會造成安全風險。為了協助降低此風險，建議您只為這些使用者提供執行任務所需的許可，並在不再需要這些使用者時將其移除。

- 為了增強安全性，建議您建立用於所有 AWS 服務的 IAM 身分，並使用暫時登入資料。Cassandra 用戶端驅動程式的 Amazon Keyspaces SigV4 身分驗證外掛程式可讓您使用 IAM 存取金鑰來驗證對 Amazon Keyspaces 的呼叫，而不是使用者名稱和密碼。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 請求中進行身分驗證，請參閱 [AWS 簽章第 4 版程序 \(SigV4\)](#)。

您可以從下列位置下載 SigV4 外掛程式。

- Java：<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>。
- Go：<https://github.com/aws/aws-sigv4-auth-cassandra-gocql-driver-plugin>。

如需示範如何使用 SigV4 身分驗證外掛程式建立連線的程式碼範例，請參閱 [the section called “使用 Cassandra 用戶端驅動程式”](#)。

主題

- [建立服務特定的登入資料，以程式設計方式存取 Amazon Keyspaces](#)
- [建立和設定 Amazon Keyspaces 的 AWS 登入資料](#)

建立服務特定的登入資料，以程式設計方式存取 Amazon Keyspaces

服務特定的登入資料類似於 Cassandra 用於身分驗證和存取管理的傳統使用者名稱和密碼。服務特定的登入資料可讓 IAM 使用者存取特定 AWS 服務。這些長期登入資料無法用於存取其他 AWS 服務。它們與特定 IAM 使用者相關聯，無法供其他 IAM 使用者使用。

Important

服務特定憑證是與特定 IAM 使用者關聯的長期憑證，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憑證存取所有 AWS 資源的許可，您應該使用 [AWS Amazon Keyspaces 的 SigV4 身分驗證外掛程式進行身分驗證](#)。

使用下列其中一個程序來產生服務特定的登入資料。

Console

使用主控台建立服務特定的登入資料

1. 登入 AWS 管理主控台，並在開啟 AWS Identity and Access Management 主控台<https://console.aws.amazon.com/iam/home>。
2. 在導覽窗格中，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檢視密碼的時間。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。

CLI

使用 建立服務特定的登入資料 AWS CLI

在產生服務特定的登入資料之前，您需要下載、安裝和設定 AWS Command Line Interface (AWS CLI)：

1. 在 AWS CLI <https://aws.amazon.com/cli> 下載。

Note

AWS CLI 會在 Windows、macOS 或 Linux 上執行。

2. 遵循AWS Command Line Interface 《使用者指南》中[安裝 AWS CLI](#) 和[設定 AWS CLI](#) 的指示。
3. 使用 AWS CLI，執行下列命令來產生使用者 的服務特定登入資料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，記下 ServiceUserName和 的值ServicePassword。將這些值儲存在安全的位置，因為您稍後會需要這些值。

 Important

這是唯一ServicePassword可供您使用的時間。

建立和設定 Amazon Keyspaces 的 AWS 登入資料

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 Keyspaces，您需要具有存取金鑰的 IAM 使用者。當您以程式設計方式使用 Amazon Keyspaces 時，您會提供 AWS 存取金鑰，讓 AWS 在程式設計呼叫中驗證您的身分。您的存取金鑰包含存取金鑰 ID（例如 AKIAIOSFODNN7EXAMPLE）和私密存取金鑰（例如 wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY）。本主題會逐步引導您完成此程序的必要步驟。

安全最佳實務建議您建立具有有限許可的 IAM 使用者，並將 IAM 角色與執行特定任務所需的許可建立關聯。然後，IAM 使用者可以暫時擔任 IAM 角色來執行所需的任務。例如，您帳戶中使用 Amazon Keyspaces 主控台的 IAM 使用者可以切換至角色，以在主控台中暫時使用角色的許可。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。使用者用來擔任角色的登入資料是暫時的。相反地，IAM 使用者具有長期登入資料，如果 IAM 使用者沒有直接指派給他們的許可，則存在安全風險。為了協助降低此風險，建議您只為這些使用者提供執行任務所需的許可，並在不再需要這些使用者時將其移除。如需角色的詳細資訊，請參閱《IAM 使用者指南》中的[角色的常見案例：使用者、應用程式和服務](#)。

主題

- [Cassandra 用戶端驅動程式的 AWS CLI、AWS 開發套件或 Amazon Keyspaces SigV4 外掛程式所需的登入資料](#)
- [使用 IAM 角色和 SigV4 外掛程式建立臨時登入資料以連線至 Amazon Keyspaces](#)
- [建立 IAM 使用者，以程式設計方式存取您 AWS 帳戶中的 Amazon Keyspaces](#)
- [為 IAM 使用者建立新的存取金鑰](#)
- [存放用於程式設計存取的存取金鑰](#)

Cassandra 用戶端驅動程式的 AWS CLI、AWS 開發套件或 Amazon Keyspaces SigV4 外掛程式所需的登入資料

驗證 IAM 使用者或角色需要下列登入資料：

AWS_ACCESS_KEY_ID

指定與 IAM 使用者或角色相關聯的 AWS 存取金鑰。

`aws_access_key_id` 需要存取金鑰才能以程式設計方式連線至 Amazon 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」。

`aws_secret_access_key` 必須以程式設計方式連線至 Amazon Keyspaces。

AWS_SESSION_TOKEN – 選用

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 “建立臨時登入資料以連線至 Amazon Keyspaces”](#)。

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

使用 IAM 角色和 SigV4 外掛程式建立臨時登入資料以連線至 Amazon Keyspaces

以程式設計方式存取 Amazon Keyspaces 的建議方法是使用[臨時登入](#)資料來驗證 SigV4 外掛程式。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。反之，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[切換到 IAM 角色 \(AWS API\)](#)。

若要開始使用臨時登入資料，您必須先建立 IAM 角色。

建立授予 Amazon Keyspaces 唯讀存取權的 IAM 角色

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
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類型下，選擇AWS 服務。在選擇使用案例下，選擇 Amazon EC2，然後選擇下一步。
4. 在新增許可頁面的許可政策下，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許可區段。您也可以在此頁面上為角色新增選用標籤。完成後，請選取建立角色。請記住此名稱，因為您在啟動 Amazon EC2 執行個體時需要此名稱。

若要在程式碼中使用臨時安全登入資料，請以程式設計方式呼叫類似的 AWS Security Token Service API，AssumeRole 並從您在上一個步驟中建立的 IAM 角色擷取產生的登入資料和工作階段字符。然後，您可以將這些值用作後續呼叫的登入資料 AWS。下列範例顯示如何使用臨時安全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 驅動程式來存取 Amazon Keyspaces 實作臨時登入資料的範例，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 SDK SDKs 和工具組區段的主要文件頁面上找到所有 AWS SDKs 的文件。 [AWS](#)

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

若要取得使用 AWS CLI、AWS SDK 或 SigV4 外掛程式以程式設計方式存取 Amazon Keyspaces 的登入資料，您必須先建立 IAM 使用者或角色。下列步驟顯示建立 IAM 使用者並設定該 IAM 使用者以程式設計方式存取 Amazon Keyspaces 的程序：

1. 在 AWS 管理主控台、AWS CLI、Tools for Windows PowerShell 或使用 AWS API 操作建立使用者。如果您在 中建立使用者 AWS 管理主控台，則會自動建立登入資料。
2. 如果您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 和私密存取金鑰）。
3. 授予使用者存取 Amazon Keyspaces 的許可。

如需建立 IAM 使用者所需的許可資訊，請參閱 [存取 IAM 資源所需的許可](#)。

Console

建立具有程式設計存取的 IAM 使用者（主控台）

1. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。

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。您可以在前往最終頁面時檢視或下載存取金鑰。

選擇下一步：許可。

5. 在設定許可頁面上，選擇直接連接現有政策，將許可指派給新使用者。

此選項會顯示您帳戶中可用的 AWS 受管和客戶受管政策清單。您可以在搜尋欄位中輸入 `keyspaces`，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`。如需每個政策的詳細資訊，請參閱 [the section called “AWS 受管政策”](#)。

為了測試目的和遵循連線教學課程，請選取新 IAM 使用者 `AmazonKeyspacesReadOnlyAccess` 的政策。注意：最佳實務是，建議您遵循最低權限原則，並建立自訂政策來限制對特定資源的存取，並僅允許必要的動作。如需 IAM 政策和檢視 Amazon Keyspaces 範例政策的詳細資訊，請參閱 [the section called “Amazon Keyspaces 身分型政策”](#)。建立自訂許可政策之後，請將政策連接至角色，然後讓使用者暫時擔任適當的角色。

選擇下一步：標籤。

6. 在新增標籤（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. 在檢閱頁面上，您可以看到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

這是您檢視或下載私密存取金鑰的唯一機會，您需要此資訊才能使用 SigV4 外掛程式。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。

CLI

建立具有程式設計存取的 IAM 使用者 (AWS CLI)

1. 使用下列 AWS CLI 程式碼建立使用者。
 - [aws iam create-user](#)
2. 提供使用者程式設計存取權。這需要存取金鑰，可透過下列方式產生。
 - AWS CLI: [aws iam create-access-key](#)
 - Tools for Windows PowerShell : [New-IAMAccessKey](#)
 - IAM API : [CreateAccessKey](#)

 Important

這是您檢視或下載私密存取金鑰的唯一機會，您需要此資訊才能使用 SigV4 外掛程式。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政策連接至定義使用者許可的使用者。注意：最佳實務是，建議您將使用者新增至群組並將政策連接至群組，而不是直接連接至使用者，以管理使用者許可。
 - 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，您可以隨時建立新的存取金鑰。如需金鑰管理的詳細資訊，例如如何更新存取金鑰，請參閱[管理 IAM 使用者的存取金鑰](#)。

為 IAM 使用者建立存取金鑰（主控台）

1. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。
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，選擇安全登入資料索引標籤。
5. 在存取金鑰最佳實務和替代方案下的存取金鑰區段中，選擇使用案例其他。按一下下一步，視需要輸入選用資訊，然後選擇建立存取金鑰。

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
- 私密存取金鑰：wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

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。

請考慮下列您已建立之金鑰對的最佳實務。

- 切勿將您的存取金鑰以純文字、程式碼儲存庫或程式碼形式存放。
 - 不再需要時，停用或刪除存取金鑰。
 - 啟用最低權限許可。
 - 定期輪換存取金鑰。
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。

存放用於程式設計存取的存取金鑰

最佳實務是，建議您不要將存取金鑰直接嵌入程式碼。「AWS SDKs」和「AWS 命令列工具」可讓您將存取金鑰放在已知位置，讓您不必將其保留在程式碼中。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。`credentials` 檔案位於 Linux、macOS 或 Unix `~/.aws/credentials` 上的，或 Windows `C:\Users\USERNAME\.aws\credentials` 上的。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。`config` 檔案位於 Linux、macOS 或 Unix `~/.aws/config` 上的，或 Windows `C:\Users\USERNAME\.aws\config` 上的。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 “Java 4.x 的身分驗證外掛程式”](#) 的先決條件。請注意，這包含預設值 AWS 區域。用戶端會使用預設登入資料提供者鏈結搜尋登入資料，而存放為環境變數的存取金鑰會優先於所有其他位置，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>
$ export AWS_DEFAULT_REGION=us-east-1
```

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>
C:\> setx AWS_DEFAULT_REGION us-east-1
```

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="wJa1rXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY"
PS C:\> $Env:AWS_SESSION_TOKEN="AQoDYXdzEJr...<remainder of security token>"
PS C:\> $Env:AWS_DEFAULT_REGION="us-east-1"
```

如果您在 PowerShell 提示字元中設定環境變數 (如前一個範例所示)，它只會在目前的工作階段儲存該數值。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在，請使用 Control Panel (控制面板) 中的 System (系統) 應用程式。或者，您也可以將變數新增到 PowerShell 設定檔，為所有未來 PowerShell 工作階段設定變數。如需有關存放環境變數或跨工作階段持續存放的詳細資訊，請參閱 [PowerShell 文件](#)。

Amazon Keyspaces 的服務端點

主題

- [連接埠和通訊協定](#)
- [全域端點](#)
- [AWS GovCloud \(US\) Region FIPS 端點](#)
- [中國區域端點](#)
- [串流端點](#)
- [連線至雙堆疊端點](#)

連接埠和通訊協定

您可以使用 Apache 2.0 授權的 Cassandra 驅動程式 cqlsh，或使用 AWS CLI 和 AWS SDK，以程式設計方式存取 Amazon Keyspaces。

下表顯示不同存取機制的連接埠和通訊協定。

程式設計存取	站點	通訊協定
CQLSH	9142	TLS
Cassandra 驅動程式	9142	TLS
AWS CLI	443	HTTPS
AWS 開發套件	443	HTTPS

對於 TLS 連線，Amazon Keyspaces 會使用 Amazon Trust Services (Amazon Root CAs1–4) 發行的憑證來驗證伺服器。如需詳細資訊，請參閱 [the section called “如何手動設定 TLS 的 cqlsh 連線”](#) 或 [the section called “使用 Cassandra 用戶端驅動程式”](#) 章節中的驅動程式 [開始之前](#) 一節。

全域端點

Amazon Keyspaces 同時支援 IPv4 和 IPv6 公有端點。您可以選擇 IPv4 端點和雙堆疊端點。端點使用以下命名慣例，您可以在其中將 *us-east-1* 取代為資料表 AWS 區域中可用的另一個。

- IPv4 端點 – `cassandra.us-east-1.amazonaws.com`
- 雙堆疊端點 – `cassandra.us-east-1.api.aws`

如需雙堆疊端點以及如何設定連線的詳細資訊，請參閱 [the section called “連線至雙堆疊端點”](#)。

Amazon Keyspaces 可在下列區域使用。

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
		cassandra.us-east-2.api.aws	HTTPS 和 TLS
美國東部 (維吉尼亞 北部)	us-east-1	cassandra.us-east-1.amazonaws.com	HTTPS 和 TLS
		cassandra-fips.us-east-1.api.aws	HTTPS 和 TLS
		cassandra-fips.us-east-1.amazonaws.com	HTTPS 和 TLS
		cassandra.us-east-1.api.aws	HTTPS 和 TLS
美國西部 (加州北 部)	us-west-1	cassandra.us-west-1.amazonaws.com cassandra.us-west-1.api.aws	HTTPS 和 TLS

區域名稱	區域	端點	通訊協定	
			HTTPS 和 TLS	
美國西部 (奧勒岡)	us-west-2	cassandra.us-west-2.amazonaws.com cassandra-fips.us-west-2.api.aws cassandra-fips.us-west-2.amazonaws.com cassandra.us-west-2.api.aws	HTTPS 和 TLS HTTPS 和 TLS HTTPS 和 TLS HTTPS 和 TLS	
非洲 (開 普敦)	af-south-1	cassandra.af-south-1.amazonaws.com cassandra.af-south-1.api.aws	HTTPS 和 TLS HTTPS 和 TLS	
亞太區域 (香港)	ap-east-1	cassandra.ap-east-1.amazonaws.com cassandra.ap-east-1.api.aws	HTTPS 和 TLS HTTPS 和 TLS	
亞太區域 (孟買)	ap-south-1	cassandra.ap-south-1.amazonaws.com cassandra.ap-south-1.api.aws	HTTPS 和 TLS HTTPS 和 TLS	
亞太區域 (首爾)	ap-northeast-2	cassandra.ap-northeast-2.amazonaws.com cassandra.ap-northeast-2.api.aws	HTTPS 和 TLS HTTPS 和 TLS	

區域名稱	區域	端點	通訊協定	
亞太區域 (新加坡)	ap-southeast-1	cassandra.ap-southeast-1.amazonaws.com	HTTPS 和 TLS	
		cassandra.ap-southeast-1.api.aws	HTTPS 和 TLS	
亞太區域 (雪梨)	ap-southeast-2	cassandra.ap-southeast-2.amazonaws.com	HTTPS 和 TLS	
		cassandra.ap-southeast-2.api.aws	HTTPS 和 TLS	
亞太區域 (東京)	ap-northeast-1	cassandra.ap-northeast-1.amazonaws.com	HTTPS 和 TLS	
		cassandra.ap-northeast-1.api.aws	HTTPS 和 TLS	
加拿大 (中部)	ca-central-1	cassandra.ca-central-1.amazonaws.com	HTTPS 和 TLS	
		cassandra.ca-central-1.api.aws	HTTPS 和 TLS	
歐洲 (法蘭克福)	eu-central-1	cassandra.eu-central-1.amazonaws.com	HTTPS 和 TLS	
		cassandra.eu-central-1.api.aws	HTTPS 和 TLS	
歐洲 (愛爾蘭)	eu-west-1	cassandra.eu-west-1.amazonaws.com	HTTPS 和 TLS	
		cassandra.eu-west-1.api.aws	HTTPS 和 TLS	

區域名稱	區域	端點	通訊協定	
歐洲 (倫敦)	eu-west-2	cassandra.eu-west-2.amazonaws.com	HTTPS 和 TLS	
		cassandra.eu-west-2.api.aws	HTTPS 和 TLS	
歐洲 (巴黎)	eu-west-3	cassandra.eu-west-3.amazonaws.com	HTTPS 和 TLS	
		cassandra.eu-west-3.api.aws	HTTPS 和 TLS	
歐洲 (斯德哥爾摩)	eu-north-1	cassandra.eu-north-1.amazonaws.com	HTTPS 和 TLS	
		cassandra.eu-north-1.api.aws	HTTPS 和 TLS	
中東 (巴林)	me-south-1	cassandra.me-south-1.amazonaws.com	HTTPS 和 TLS	
		cassandra.me-south-1.api.aws	HTTPS 和 TLS	
中東 (阿拉伯聯合大公國)	me-central-1	cassandra.me-central-1.amazonaws.com	HTTPS 和 TLS	
		cassandra.me-central-1.api.aws	HTTPS 和 TLS	
南美洲 (聖保羅)	sa-east-1	cassandra.sa-east-1.amazonaws.com	HTTPS 和 TLS	
		cassandra.sa-east-1.api.aws	HTTPS 和 TLS	

區域名稱	區域	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	
		cassandra.us-gov-east-1.api.aws	HTTPS 和 TLS	
AWS GovCloud (美國西部)	us-gov-west-1	cassandra.us-gov-west-1.amazonaws.com	HTTPS 和 TLS	
		cassandra.us-gov-west-1.api.aws	HTTPS 和 TLS	

AWS GovCloud (US) Region FIPS 端點

中的可用 FIPS 端點 AWS GovCloud (US) Region。Amazon Keyspaces 同時支援 IPv4 和 IPv6 FIPS 端點。您可以選擇 IPv4 端點和雙堆疊端點。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 IPv4	FIPS 端點雙堆疊	通訊協定
AWS GovCloud (美國東部)	us-gov-east-1	cassandra.us-gov-east-1.amazonaws.com	cassandra.us-gov-east-1.api.aws	HTTPS 和 TLS
AWS GovCloud (美國西部)	us-gov-west-1	cassandra.us-gov-west-1.amazonaws.com	cassandra.us-gov-west-1.api.aws	HTTPS 和 TLS

中國區域端點

Amazon Keyspaces 在中國 AWS 區域支援 IPv4 端點。

若要存取這些端點，您必須註冊一組中國區域獨有的個別帳戶登入資料。如需詳細資訊，請參閱[中國註冊、帳戶和登入資料](#)。

區域名稱	區域	端點	通訊協定
中國 (北京)	cn-north-1	cassandra.cn-north-1.amazonaws.com.cn	HTTPS 和 TLS
中國 (寧夏)	cn-northwest-1	cassandra.cn-northwest-1.amazonaws.com.cn	HTTPS 和 TLS

串流端點

Amazon Keyspaces CDC streams 在以下提供 AWS 區域。此資料表顯示每個區域的可用雙堆疊服務端點。如需的詳細資訊 Amazon Keyspaces CDC streams，請參閱 [the section called “存取 CDC 串流端點”](#)。

區域名稱	區域	端點	通訊協定
美國東部 (俄亥俄)	us-east-2	cassandra-streams.us-east-2.api.aws	HTTPS
美國東部 (維吉尼亞北部)	us-east-1	cassandra-streams.us-east-1.api.aws	HTTPS
美國西部 (加州北部)	us-west-1	cassandra-streams.us-west-1.api.aws	HTTPS
美國西部 (奧勒岡)	us-west-2	cassandra-streams.us-west-2.api.aws	HTTPS

區域名稱	區域	端點	通訊協定	
非洲 (開普敦)	af-south-1	cassandra-streams.af-south-1.api.aws	HTTPS	
亞太區域 (香港)	ap-east-1	cassandra-streams.ap-east-1.api.aws	HTTPS	
亞太區域 (孟買)	ap-south-1	cassandra-streams.ap-south-1.api.aws	HTTPS	
亞太區域 (首爾)	ap-northeast-2	cassandra-streams.ap-northeast-2.api.aws	HTTPS	
亞太區域 (新加坡)	ap-southeast-1	cassandra-streams.ap-southeast-1.api.aws	HTTPS	
亞太區域 (雪梨)	ap-southeast-2	cassandra-streams.ap-southeast-2.api.aws	HTTPS	
亞太區域 (東京)	ap-northeast-1	cassandra-streams.ap-northeast-1.api.aws	HTTPS	
加拿大 (中部)	ca-central-1	cassandra-streams.ca-central-1.api.aws	HTTPS	
歐洲 (法蘭克福)	eu-central-1	cassandra-streams.eu-central-1.api.aws	HTTPS	
歐洲 (愛爾蘭)	eu-west-1	cassandra-streams.eu-west-1.api.aws	HTTPS	
歐洲 (倫敦)	eu-west-2	cassandra-streams.eu-west-2.api.aws	HTTPS	
歐洲 (巴黎)	eu-west-3	cassandra-streams.eu-west-3.api.aws	HTTPS	

區域名稱	區域	端點	通訊協定
歐洲 (斯德哥爾摩)	eu-north-1	cassandra-streams.eu-north-1.api.aws	HTTPS
中東 (巴林)	me-south-1	cassandra-streams.me-south-1.api.aws	HTTPS
中東 (阿拉伯聯合大公國)	me-central-1	cassandra-streams.me-central-1.api.aws	HTTPS
南美洲 (聖保羅)	sa-east-1	cassandra-streams.sa-east-1.api.aws	HTTPS
AWS GovCloud (美國東部)	us-gov-east-1	cassandra-streams.us-gov-east-1.api.aws	HTTPS
AWS GovCloud (美國西部)	us-gov-west-1	cassandra-streams.us-gov-west-1.api.aws	HTTPS

連線至雙堆疊端點

Amazon Keyspaces 全域端點是接受 IPv4 和 IPv6 請求的雙堆疊端點。

使用 IPv6 連線至 Amazon Keyspaces 時，服務會自動調整系統資料表回應，以符合您的連線通訊協定。這可確保您的應用程式接收與其連線類型相符的一致網路地址資訊。這可為用戶端提供準確的網路拓撲資訊，同時保持現有 CQL 應用程式的回溯相容性。

Amazon Keyspaces 會自動偵測用戶端連線所使用的網路通訊協定 (IPv4 或 IPv6)，並相應地調整系統資料表回應。此偵測會在初始連線交握期間透明發生，不需要用戶端應用程式的額外組態。

Amazon Keyspaces 會根據您的連線通訊協定傳回 IP 地址。例如，來自 IPv4 網路的請求會傳回下列回應。


```
SELECT * FROM system.peers;  
-- Returns IPv4 addresses in peer column  
-- Example: 172.31.1.1, 172.31.1.2, etc.
```

從 IPv6 網路到雙堆疊端點的連線，例如 `cassandra.us-east-1.api.aws`，會傳回下列回應。

```
SELECT * FROM system.peers;  
-- Returns IPv6 addresses in peer column  
-- Example: 2001:db8::1, 2001:db8::2, etc.
```

如需 Amazon Keyspaces 中 IPv6 支援的詳細資訊，請參閱 [the section called “Amazon Keyspaces 中的 IPv6 支援”](#)。

Amazon Keyspaces 中的 IPv6 支援

Amazon Keyspaces 中的 IPv6 支援允許應用程式使用網際網路通訊協定第 6 版建立連線，後者是新一代網際網路通訊協定，可提供與 IPv4 相比大幅擴展的地址空間。實作使用同時支援 IPv4 和 IPv6 的雙堆疊端點，確保回溯相容性，同時實現面向未來的連線能力。如需端點清單，請參閱 [the section called “全域端點”](#)。

Amazon Keyspaces 透過雙堆疊架構實作 IPv6 支援，可維持完整的回溯相容性，同時啟用 IPv6 連線。

Amazon Keyspaces 中的 DNS 解析

當應用程式連線到雙堆疊端點時，DNS 解析程序會傳回兩種地址類型：

記錄 (IPv4)

用於回溯相容性的傳統 IPv4 地址

AAAA 記錄 (IPv6)

用於現代連線的新 IPv6 地址

用戶端的作業系統和網路堆疊會根據本機組態、網路可用性和系統偏好設定，自動選取最適合的通訊協定。

Cassandra 查詢語言 (CQL) 通訊協定可無縫支援 IPv6 連線，而無需變更應用程式碼。

自動通訊協定選擇

- 應用程式指定雙堆疊端點
- 網路堆疊會根據可用性選擇 IPv4 或 IPv6
- 現有 CQL 應用程式不需要修改程式碼

驅動程式相容性

- 所有主要 CQL 驅動程式都以透明的方式支援 IPv6
- DataStax 驅動程式原生處理 IPv6 地址
- 開放原始碼驅動程式可在不修改的情況下運作

連線一致性

- 系統資料表反映使用的連線通訊協定
- IPv6 連線會在 `system.peers` 中顯示 IPv6 地址
- IPv4 連線會繼續顯示 IPv4 地址

使用 **cqlsh** 連線至 Amazon Keyspaces

若要使用連線到 Amazon Keyspaces `cqlsh`，您可以使用 `cqlsh-expansion`。這是一個工具組，其中包含常見的 Apache Cassandra 工具，例如 `cqlsh` 和針對 Amazon Keyspaces 預先設定的協助程式，同時保持與 Apache Cassandra 的完整相容性。`cqlsh-expansion` 整合了 SigV4 身分驗證外掛程式，並可讓您使用 IAM 存取金鑰來連線，而不是使用者名稱和密碼。您只需要安裝 `cqlsh` 指令碼來建立連線，而不是完整的 Apache Cassandra 分佈，因為 Amazon Keyspaces 是無伺服器。此輕量型安裝套件包含 `cqlsh-expansion` 和傳統指令碼，您可以在任何支援 Python 的平台上安裝這些 `cqlsh` 指令碼。

Note

`Murmur3Partitioner` 是 Amazon Keyspaces 的建議分割區。`cqlsh-expansion` 不支援 Amazon Keyspaces `DefaultPartitioner`。如需詳細資訊，請參閱 [the section called “使用分割區”](#)。

如需的一般資訊 `cqlsh`，請參閱 [cqlsh : CQL shell](#)。

主題

- [使用 cqlsh-expansion 連線至 Amazon Keyspaces](#)

- [如何手動設定 TLS 的 cqlsh 連線](#)

使用 **cqlsh-expansion** 連線至 Amazon Keyspaces

安裝和設定 **cqlsh-expansion**

1. 若要安裝 cqlsh-expansion Python 套件，您可以執行 pip 命令。這會使用 pip 安裝，以及包含相依性清單的檔案，在您的機器上安裝 cqlsh-expansion 指令碼。--user flag pip 告知 為您的平台使用 Python 使用者安裝目錄。在以 Unix 為基礎的系統上，應該是 ~/.local/目錄。

您需要 Python 3 才能安裝 cqlsh-expansion，若要了解您的 Python 版本，請使用 Python --version。若要安裝，您可以執行下列命令。

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

輸出看起來應該與此類似。

```
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 boto3<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, boto3, 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 `PATH`，您需要依照作業系統的指示新增它。以下是 Ubuntu Linux 的一個範例。

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

若要確認套件已安裝，您可以執行下列命令。

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

輸出看起來應該像這樣。

```
cqlsh 6.1.0
```

2. 若要設定 `cqlsh-expansion`，您可以執行安裝後指令碼，以自動完成下列步驟：
 1. 如果 `.cassandra` 目錄尚未存在，請在使用者主目錄中建立目錄。
 2. 將預先設定的 `cqlshrc` 組態檔案複製到 `.cassandra` 目錄。
 3. 將合併的憑證檔案複製到 `.cassandra` 目錄。Amazon Keyspaces 使用此憑證來設定與 Transport Layer Security (TLS) 的安全連線。傳輸中加密可在資料往返 Amazon Keyspaces 時

加密資料，提供多一層的資料保護。如需憑證的詳細資訊，請參閱 [the section called “如何手動設定 TLS 的 cqlsh 連線”](#)。

若要先檢閱指令碼，您可以在的 Github 儲存庫中存取指令碼 [post_install.py](#)。

若要使用指令碼，您可以執行下列命令。

```
cqlsh-expansion.init
```

 Note

當您 cqlsh-expansion 使用解除安裝時，不會移除安裝後指令碼建立的目錄和檔案 pip uninstall，而且必須手動刪除。

使用連線至 Amazon Keyspaces **cqlsh-expansion**

1. 設定您的 AWS 區域 並將其新增為使用者環境變數。

若要將預設區域新增為 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 us-east-1。

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

如需如何設定環境變數的詳細資訊，包括其他平台的，請參閱 [如何設定環境變數](#)。

2. 尋找您的服務端點。

為您的區域選擇適當的服務端點。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. 設定身分驗證方法。

使用 IAM 存取金鑰 (IAM 使用者、角色和聯合身分) 連線是增強安全性的建議方法。

您需要完成下列步驟，才能使用 IAM 存取金鑰進行連線：

- a. 建立 IAM 使用者，或遵循最佳實務並建立 IAM 使用者可以擔任的 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 “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

- b. 建立 IAM 政策，授予角色（或 IAM 使用者）至少對 Amazon Keyspaces 的唯讀存取權。如需 IAM 使用者或角色連線到 Amazon Keyspaces 所需的許可的詳細資訊，請參閱 [the section called “存取 Amazon Keyspaces 資料表”](#)。
- 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的詳細資訊，包括其他平台，請參閱[如何設定環境變數](#)。

 Note

如果您是從 Amazon EC2 執行個體連線，您也需要在安全群組中設定傳出規則，以允許從執行個體到 Amazon Keyspaces 的流量。如需如何檢視和編輯 EC2 傳出規則的詳細資訊，請參閱《[Amazon EC2 使用者指南](#)》中的[將規則新增至安全群組](#)。

4. 使用 `cqlsh-expansion` 和 SigV4 身分驗證連線至 Amazon Keyspaces。

若要使用 連線到 Amazon Keyspaces `cqlsh-expansion`，您可以使用下列命令。請務必將服務端點取代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，您應該會看到類以下列範例的輸出。

```
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>
```

如果您遇到連線錯誤，請參閱 [the section called “Cqlsh 連線錯誤”](#) 以取得故障診斷資訊。

- 使用服務特定的登入資料連線至 Amazon Keyspaces。

若要與 Cassandra 用於身分驗證的傳統使用者名稱和密碼組合連線，您必須先建立 Amazon Keyspaces 的服務特定憑證，如中所述[the section called “建立服務特定的登入資料”](#)。您也必須授予該使用者存取 Amazon Keyspaces 的許可，如需詳細資訊，請參閱 [the section called “存取 Amazon Keyspaces 資料表”](#)。

為使用者建立服務特定的登入資料和許可之後，您必須更新 `cqlshrc` 檔案，通常可在使用者目錄路徑 中找到 `~/.cassandra/`。在 `cqlshrc` 檔案中，前往 `Cassandra [authentication]` 區段，並使用 `[auth_provider]` 「；」字元在下註解 `SigV4` 模組和類別，如下列範例所示。

```
[auth_provider]

; module = cassandra_sigv4.auth

; classname = SigV4AuthProvider
```

更新 `cqlshrc` 檔案之後，您可以使用下列命令，使用服務特定的登入資料連線至 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` 命令。

```
pip3 uninstall cqlsh-expansion
```

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

如何手動設定 TLS 的 `cqlsh` 連線

Amazon Keyspaces 只接受使用 Transport Layer Security (TLS) 的安全連線。您可以使用 `cqlsh-expansion` 公用程式來自動為您下載憑證，並安裝預先設定的 `cqlshrc` 組態檔案。如需詳細資訊，請參閱此頁面 [the section called “使用 cqlsh-expansion”](#) 上的。

如果您想要下載憑證並手動設定連線，您可以使用下列步驟執行此操作。

- 下載下列數位憑證，並將檔案儲存在本機或主目錄中。
 - AmazonRootCA1

- b. AmazonRootCA2
- c. AmazonRootCA3
- d. AmazonRootCA4
- e. Starfield Class 2 根目錄（選用 – 用於回溯相容性）

若要下載憑證，您可以使用下列命令。

```
curl -O https://www.amazontrust.com/repository/AmazonRootCA1.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA2.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA3.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA4.pem
curl -O https://certs.secureserver.net/repository/sf-class2-root.crt
```

Note

Amazon Keyspaces 先前使用以 Starfield 類別 2 CA 為基礎的 TLS 憑證。AWS 正在將所有遷移 AWS 區域至根據 Amazon Trust Services (Amazon 根 CAs 發行的憑證。在此轉換期間，請將用戶端設定為信任 Amazon 根 CAs1–4 和 Starfield 根，以確保所有區域的相容性。

2. 使用範例中的名稱 *keyspaces-bundle.pem*，將所有下載的憑證合併成單一 pem 檔案。您可以執行下列命令，即可進行新增：請記下檔案的路徑，您稍後需要此操作。

```
cat AmazonRootCA1.pem \
    AmazonRootCA2.pem \
    AmazonRootCA3.pem \
    AmazonRootCA4.pem \
    sf-class2-root.crt \
    > keyspaces-bundle.pem
```

3. 開啟 Cassandra 主目錄中的 cqlshrc 組態檔案，例如 `${HOME}/.cassandra/cqlshrc`，並新增以下行。

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

[ssl]
validate = true
```



```
certfile = path_to_file/keysaces-bundle.pem
```

使用 AWS CLI 連線至 Amazon Keyspaces

您可以使用 AWS Command Line Interface (AWS CLI) 從命令列控制多個 AWS 服務，並透過指令碼將其自動化。透過 Amazon Keyspaces，您可以使用 AWS CLI 進行資料定義語言 (DDL) 操作，例如建立資料表。此外，您可以使用基礎設施做為程式碼 (IaC) 服務和工具，例如 AWS CloudFormation 和 Terraform。

您必須先取得存取金鑰 ID 和私密存取金鑰，才能 AWS CLI 搭配 Amazon Keyspaces 使用。如需詳細資訊，請參閱[the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

如需 中可用於 Amazon Keyspaces 的所有命令的完整清單 AWS CLI，請參閱 [AWS CLI 命令參考](#)。

主題

- [下載和設定 AWS CLI](#)
- [AWS CLI 搭配 Amazon Keyspaces 使用](#)

下載和設定 AWS CLI

AWS CLI 可在 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. 遵循[安裝 AWS CLI](#)和[設定 AWS CLI](#)的指示

AWS CLI 搭配 Amazon Keyspaces 使用

命令列格式包含 Amazon Keyspaces 操作名稱，後面接著該操作的參數。AWS CLI 支援參數值以及 JSON 的速記語法。下列 Amazon Keyspaces 範例使用 AWS CLI 速記語法。如需詳細資訊，請參閱[搭配 CLI AWS 使用速記語法](#)。

下列命令會使用 名稱目錄建立金鑰空間。

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

命令會在輸出中傳回資源 Amazon Resource Name (ARN)。

```
{
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/catalog/"
}
```

若要確認金鑰空間目錄是否存在，您可以使用下列命令。

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

命令的輸出會傳回下列值。

```
{
  "keyspaceName": "catalog",
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/catalog/"
}
```

下列命令會建立名為 book_awards 的資料表。資料表的分割區索引鍵由欄 yearaward和 組成，叢集索引鍵由欄 category和 組成rank，兩個叢集欄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-1:111122223333:/keyspace/catalog/table/
book_awards"
}
```

若要確認資料表的中繼資料和屬性，您可以使用下列命令。

```
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-1:111122223333:/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": ""
}
}

```

建立具有複雜結構描述的資料表時，從 JSON 檔案載入資料表的結構描述定義會很有幫助。以下是此範例。從 [schema_definition.zip](#) 下載結構描述定義範例 JSON 檔案，並擷取 `schema_definition.json`，並記下檔案的路徑。在此範例中，結構描述定義 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 'catalog'
```

```
--table-name 'book_awards' --schema-definition 'file://  
schema_definition.json'
```

下列範例示範如何建立名為 myTable 的簡單資料表，以及其他選項。請注意，命令會細分為不同的資料列，以改善可讀性。此命令說明如何建立資料表和：

- 設定資料表的容量模式
- 啟用資料表的Point-in-time復原
- 將資料表的預設存留時間 (TTL) 值設定為一年
- 為資料表新增兩個標籤

```
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'
```

此範例說明如何建立新的資料表，該資料表使用客戶受管金鑰進行加密，並啟用 TTL 以允許您設定資料欄和資料列的過期日期。若要執行此範例，您必須將客戶受管 AWS KMS 金鑰的資源 ARN 取代為您自己的金鑰，並確保 Amazon Keyspaces 可以存取該金鑰。

```
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-1:111122223333:key/11111111-2222-3333-4444-555555555555'  
    --ttl 'status=ENABLED'
```

使用 API 連線至 Amazon Keyspaces

您可以使用 AWS SDK 和 AWS Command Line Interface (AWS CLI) 以互動方式使用 Amazon Keyspaces。您可以使用 API 進行資料語言定義 (DDL) 操作，例如建立金鑰空間或資料表。此外，您可以使用基礎設施做為程式碼 (IaC) 服務和工具，例如 AWS CloudFormation 和 Terraform。

您必須先取得存取金鑰 ID 和私密存取金鑰，才能 AWS CLI 搭配 Amazon Keyspaces 使用。如需詳細資訊，請參閱[the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

如需 API 中可用於 Amazon Keyspaces 的所有操作的完整清單，請參閱 [Amazon Keyspaces API 參考](#)。

使用 Cassandra 用戶端驅動程式以程式設計方式存取 Amazon Keyspaces

您可以使用許多第三方開放原始碼 Cassandra 驅動程式來連線至 Amazon Keyspaces。Amazon Keyspaces 與支援 Apache Cassandra 3.11.2 版的 Cassandra 驅動程式相容。以下是我們已測試並建議與 Amazon Keyspaces 搭配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

如需 Cassandra 驅動程式的詳細資訊，請參閱 [Apache Cassandra 用戶端驅動程式](#)。

Note

為了協助您開始使用，您可以檢視和下載 end-to-end 程式碼範例，這些範例使用熱門驅動程式建立與 Amazon Keyspaces 的連線。請參閱 GitHub 上的 [Amazon Keyspaces 範例](#)。

本章中的教學課程包含簡單的 CQL 查詢，以確認已成功建立與 Amazon Keyspaces 的連線。若要了解如何在連線至 Amazon Keyspaces 端點後使用金鑰空間和資料表，請參閱 [CQL 語言參考](#)。如需示範如何從 Amazon VPC 端點連線至 Amazon Keyspaces 的 step-by-step 教學課程，請參閱 [the section called “與 VPC 端點連線”](#)。

主題

- [使用 Cassandra Java 用戶端驅動程式以程式設計方式存取 Amazon Keyspaces](#)
- [使用 Cassandra Python 用戶端驅動程式以程式設計方式存取 Amazon Keyspaces](#)

- [使用 Cassandra Node.js 用戶端驅動程式以程式設計方式存取 Amazon Keyspaces](#)
- [使用 Cassandra .NET Core 用戶端驅動程式以程式設計方式存取 Amazon Keyspaces](#)
- [使用 Cassandra Go 用戶端驅動程式以程式設計方式存取 Amazon Keyspaces](#)
- [使用 Cassandra Perl 用戶端驅動程式以程式設計方式存取 Amazon Keyspaces](#)

使用 Cassandra Java 用戶端驅動程式以程式設計方式存取 Amazon Keyspaces

本節說明如何使用 Java 用戶端驅動程式連線至 Amazon Keyspaces。

Note

Java 17 和 DataStax Java 驅動程式 4.17 目前僅支援 Beta 版。如需詳細資訊，請參閱https://docs.datastax.com/en/developer/java-driver/4.17/upgrade_guide/。

若要為使用者和應用程式提供以程式設計方式存取 Amazon Keyspaces 資源的登入資料，您可以執行下列其中一項操作：

- 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，我們建議為所有 AWS 服務中使用的 IAM 身分建立 IAM 存取金鑰。Cassandra 用戶端驅動程式的 Amazon Keyspaces SigV4 身分驗證外掛程式可讓您使用 IAM 存取金鑰來驗證對 Amazon Keyspaces 的呼叫，而不是使用者名稱和密碼。如需詳細資訊，請參閱[the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

Note

如需如何使用 Amazon Keyspaces 搭配 Spring Boot 的範例，請參閱 <https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/java/datastax-v4/spring>。

主題

- [開始之前](#)
- [使用服務特定登入資料，使用適用於 Apache Cassandra 的 DataStax Java 驅動程式連線至 Amazon Keyspaces 的 Step-by-step 教學課程](#)

- [使用適用於 Apache Cassandra 的 4.x DataStax Java 驅動程式和 SigV4 身分驗證外掛程式連線至 Amazon Keyspaces 的 Step-by-step 教學課程](#)
- [使用適用於 Apache Cassandra 的 3.x DataStax Java 驅動程式和 SigV4 身分驗證外掛程式連線至 Amazon Keyspaces](#)

開始之前

若要連線至 Amazon Keyspaces，您需要先完成下列任務，才能開始。

1. Amazon Keyspaces 需要使用 Transport Layer Security (TLS) 來協助保護與用戶端的連線。
 - a. 下載下列數位憑證，並將檔案儲存在本機或主目錄中。
 1. AmazonRootCA1
 2. AmazonRootCA2
 3. AmazonRootCA3
 4. AmazonRootCA4
 5. Starfield Class 2 根目錄（選用 – 用於回溯相容性）

若要下載憑證，您可以使用下列命令。

```
curl -O https://www.amazontrust.com/repository/AmazonRootCA1.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA2.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA3.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA4.pem
curl -O https://certs.secureserver.net/repository/sf-class2-root.crt
```

Note

Amazon Keyspaces 先前使用以 Starfield 類別 2 CA 為基礎的 TLS 憑證。AWS 正在將所有遷移 AWS 區域至根據 Amazon Trust Services (Amazon 根 CAs 發行的憑證。在此轉換期間，請將用戶端設定為信任 Amazon 根 CAs1–4 和 Starfield 根，以確保所有區域的相容性。

- b. 將數位憑證轉換為 trustStore 檔案，並將其新增至金鑰存放區。

```
openssl x509 -outform der -in AmazonRootCA1.pem -out temp_file.der
```



```
keytool -import -alias amazon-root-ca-1 -keystore cassandra_truststore.jks -
file temp_file.der

openssl x509 -outform der -in AmazonRootCA2.pem -out temp_file.der
keytool -import -alias amazon-root-ca-2 -keystore cassandra_truststore.jks -
file temp_file.der

openssl x509 -outform der -in AmazonRootCA3.pem -out temp_file.der
keytool -import -alias amazon-root-ca-3 -keystore cassandra_truststore.jks -
file temp_file.der

openssl x509 -outform der -in AmazonRootCA4.pem -out temp_file.der
keytool -import -alias amazon-root-ca-4 -keystore cassandra_truststore.jks -
file temp_file.der

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
```

在最後一個步驟中，您需要為金鑰存放區建立密碼，並信任每個憑證。互動式命令看起來像這樣。

```
Enter keystore password:
Re-enter new password:
Owner: CN=Amazon Root CA 1, O=Amazon, C=US
Issuer: CN=Amazon Root CA 1, O=Amazon, C=US
Serial number: 66c9fcf99bf8c0a39e2f0788a43e696365bca
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sun Jan 17 00:00:00 UTC 2038
Certificate fingerprints:
    SHA1: 8D:A7:F9:65:EC:5E:FC:37:91:0F:1C:6E:59:FD:C1:CC:6A:6E:DE:16
    SHA256:
    8E:CD:E6:88:4F:3D:87:B1:12:5B:A3:1A:C3:FC:B1:3D:70:16:DE:7F:57:CC:90:4F:E1:CB:97:C6:AE
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
    CA:true
    PathLen:2147483647
]
```

```
#2: ObjectId: 2.5.29.15 Criticality=true
```

```
KeyUsage [
```

```
    DigitalSignature
```

```
    Key_CertSign
```

```
    Crl_Sign
```

```
]
```

```
#3: ObjectId: 2.5.29.14 Criticality=false
```

```
SubjectKeyIdentifier [
```

```
KeyIdentifier [
```

```
0000: 84 18 CC 85 34 EC BC 0C    94 94 2E 08 59 9C C7 B2    ....4.....Y...
```

```
0010: 10 4E 0A 08                      .N..
```

```
]
```

```
]
```

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

```
Certificate was added to keystore
```

```
Enter keystore password:
```

```
Owner: CN=Amazon Root CA 2, O=Amazon, C=US
```

```
Issuer: CN=Amazon Root CA 2, O=Amazon, C=US
```

```
Serial number: 66c9fd29635869f0a0fe58678f85b26bb8a37
```

```
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sat May 26 00:00:00 UTC 2040
```

```
Certificate fingerprints:
```

```
    SHA1: 5A:8C:EF:45:D7:A6:98:59:76:7A:8C:8B:44:96:B5:78:CF:47:4B:1A
```

```
    SHA256:
```

```
    1B:A5:B2:AA:8C:65:40:1A:82:96:01:18:F8:0B:EC:4F:62:30:4D:83:CE:C4:71:3A:19:C3:9C:01:1E
```

```
Signature algorithm name: SHA384withRSA
```

```
Subject Public Key Algorithm: 4096-bit RSA key
```

```
Version: 3
```

```
Extensions:
```

```
#1: ObjectId: 2.5.29.19 Criticality=true
```

```
BasicConstraints:[
```

```
    CA:true
```

```
    PathLen:2147483647
```

```
]
```

```
#2: ObjectId: 2.5.29.15 Criticality=true
```

```
KeyUsage [
```

```
    DigitalSignature
```

```
    Key_CertSign
```

```
    Crl_Sign
```

```
]
```

```
#3: ObjectId: 2.5.29.14 Criticality=false
```

```
SubjectKeyIdentifier [
```

```
KeyIdentifier [
```

```
0000: B0 0C F0 4C 30 F4 05 58    02 48 FD 33 E5 52 AF 4B    ...L0..X.H.3.R.K
```

```
0010: 84 E3 66 52                                ..fR
```

```
]
```

```
]
```

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

```
Certificate was added to keystore
```

```
Enter keystore password:
```

```
Owner: CN=Amazon Root CA 3, O=Amazon, C=US
```

```
Issuer: CN=Amazon Root CA 3, O=Amazon, C=US
```

```
Serial number: 66c9fd5749736663f3b0b9ad9e89e7603f24a
```

```
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sat May 26 00:00:00 UTC 2040
```

```
Certificate fingerprints:
```

```
SHA1: 0D:44:DD:8C:3C:8C:1A:1A:58:75:64:81:E9:0F:2E:2A:FF:B3:D2:6E
```

```
SHA256:
```

```
18:CE:6C:FE:7B:F1:4E:60:B2:E3:47:B8:DF:E8:68:CB:31:D0:2E:BB:3A:DA:27:15:69:F5:03:43:B4
```

```
Signature algorithm name: SHA256withECDSA
```

```
Subject Public Key Algorithm: 256-bit EC (secp256r1) key
```

```
Version: 3
```

```
Extensions:
```

```
#1: ObjectId: 2.5.29.19 Criticality=true
```

```
BasicConstraints:[
```

```
CA:true
```

```
PathLen:2147483647
```

```
]
```

```
#2: ObjectId: 2.5.29.15 Criticality=true
```

```
KeyUsage [
```

```
DigitalSignature
```

```
Key_CertSign
```

```
Crl_Sign
```

```
]
```

```
#3: ObjectId: 2.5.29.14 Criticality=false
```

```
SubjectKeyIdentifier [
```

```
KeyIdentifier [
```

```
0000: AB B6 DB D7 06 9E 37 AC    30 86 07 91 70 C7 9C C4    .....7.0...p...
```

```

0010: 19 B1 78 C0                                     ...x.
]
]

Trust this certificate? [no]: yes
Certificate was added to keystore
Enter keystore password:
Owner: CN=Amazon Root CA 4, O=Amazon, C=US
Issuer: CN=Amazon Root CA 4, O=Amazon, C=US
Serial number: 66c9fd7c1bb104c2943e5717b7b2cc81ac10e
Valid from: Tue May 26 00:00:00 UTC 2015 until: Sat May 26 00:00:00 UTC 2040
Certificate fingerprints:
    SHA1: F6:10:84:07:D6:F8:BB:67:98:0C:C2:E2:44:C2:EB:AE:1C:EF:63:BE
    SHA256:
    E3:5D:28:41:9E:D0:20:25:CF:A6:90:38:CD:62:39:62:45:8D:A5:C6:95:FB:DE:A3:C2:2B:0B:FB:25
Signature algorithm name: SHA384withECDSA
Subject Public Key Algorithm: 384-bit EC (secp384r1) key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.19 Criticality=true
BasicConstraints:[
    CA:true
    PathLen:2147483647
]

#2: ObjectId: 2.5.29.15 Criticality=true
KeyUsage [
    DigitalSignature
    Key_CertSign
    Crl_Sign
]

#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: D3 EC C7 3A 65 6E CC E1    DA 76 9A 56 FB 9C F3 86    ...:en...v.V....
0010: 6D 57 E5 81                mW..
]
]

Trust this certificate? [no]: yes
Certificate was added to keystore

```

```

Enter keystore 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:
    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 (weak)
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                                ....
  ]
]

Warning:

```

```
The input uses the SHA1withRSA signature algorithm which is considered a security risk. This algorithm will be disabled in a future update.
```

```
Trust this certificate? [no]: yes
Certificate was added to keystore
```

2. 在 JVM 引數中連接 trustStore 檔案：

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

使用服務特定登入資料，使用適用於 Apache Cassandra 的 DataStax Java 驅動程式連線至 Amazon Keyspaces 的 Step-by-step 教學課程

下列 step-by-step 教學課程會逐步引導您使用適用於 Cassandra 的 Java 驅動程式，使用服務特定的登入資料連線至 Amazon Keyspaces。具體而言，您將使用適用於 Apache Cassandra 的 DataStax Java 驅動程式 4.0 版本。

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

若要遵循本教學課程，您需要產生服務特定的登入資料，並將適用於 Apache Cassandra 的 DataStax Java 驅動程式新增至您的 Java 專案。

- 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 “建立服務特定的登入資料”](#)。如果您偏好使用 IAM 存取金鑰進行身分驗證，請參閱 [the section called “Java 4.x 的身分驗證外掛程式”](#)。
- 將 Apache Cassandra 的 DataStax Java 驅動程式新增至您的 Java 專案。請確定您使用的驅動程式版本支援 Apache Cassandra 3.11.2。如需詳細資訊，請參閱 [Apache Cassandra 的 DataStax Java 驅動程式文件](#)。

步驟 2：設定驅動程式

您可以為應用程式建立組態檔案，以指定 DataStax Java Cassandra 驅動程式的設定。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 連線至 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. 身分驗證提供者 – 使用 `PlainTextAuthProvider` 類別建立身分驗證提供者。`ServiceUserName` 和 `ServicePassword` 應符合您在產生服務特定登入資料時，依照中的步驟取得的使用者名稱和密碼[建立服務特定的登入資料，以程式設計方式存取 Amazon Keyspaces](#)。

Note

您可以使用適用於 Apache Cassandra 的 DataStax Java 驅動程式的身分驗證外掛程式來使用短期憑證，而不是驅動程式組態檔案中的硬式編碼憑證。若要進一步了解，請遵循的指示[the section called “Java 4.x 的身分驗證外掛程式”](#)。

2. 本機資料中心 – 將 `local-datacenter` 的值設定為您要連線 `local-datacenter` 的區域。例如，如果應用程式正在連線至 `cassandra.us-east-2.amazonaws.com`，請將本機資料中心設定為 `us-east-2`。如需所有可用的 AWS 區域，請參閱 [???](#)。`slow-replica-avoidance = false` 設定為與較少節點進行負載平衡。
3. SSL/TLS – 使用指定具有類別的單行，在組態檔案中新增區段，以初始化 `SSLEngineFactory`。`class = DefaultSslEngineFactory`。提供 `trustStore` 檔案的路徑以及您先前建立的密碼。Amazon Keyspaces 不支援 `hostname-validation` 對等，因此請將此選項設定為 `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 `trustStore` 路徑，或是將 `trustStore` 路徑新增至 JVM 引數。

步驟 3：執行範例應用程式

此程式碼範例顯示簡單的命令列應用程式，該應用程式使用我們先前建立的組態檔案，建立與 Amazon Keyspaces 的連線集區。它透過執行簡單的查詢來確認已建立連線。

```

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` 區塊來建立連線，以確保一律關閉。如果您不使用 `try` 區塊，請記得關閉您的連線，以避免資源洩漏。

使用適用於 Apache Cassandra 的 4.x DataStax Java 驅動程式和 SigV4 身分驗證外掛程式連線至 Amazon Keyspaces 的 Step-by-step 教學課程

下一節說明如何使用開放原始碼 4.x DataStax Java 驅動程式的 SigV4 身分驗證外掛程式，讓 Apache Cassandra 存取 Amazon Keyspaces（適用於 Apache Cassandra）。外掛程式可從 [GitHub 儲存庫](#) 取得。

SigV4 身分驗證外掛程式可讓您在連線至 Amazon Keyspaces 時使用使用者或角色的 IAM 登入資料。此外掛程式不使用使用者名稱和密碼，而是使用存取金鑰簽署 API 請求。如需詳細資訊，請參閱 [the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

步驟 1：事前準備

若要遵循本教學課程，您需要完成下列任務。

- 如果您尚未這麼做，請依照 中的步驟為 IAM 使用者或角色建立登入資料 [the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。本教學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 “管理存取金鑰”](#)。
- 將 Apache Cassandra 的 DataStax Java 驅動程式新增至您的 Java 專案。請確定您使用的驅動程式版本支援 Apache Cassandra 3.11.2。如需詳細資訊，請參閱 [DataStax Java Driver for Apache Cassandra 文件](#)。
- 將身分驗證外掛程式新增至您的應用程式。身分驗證外掛程式支援適用於 Apache Cassandra 的 DataStax Java 驅動程式 4.x 版。如果您使用的是 Apache Maven，或是可以使用 Maven 相依性的建置系統，請將下列相依性新增至您的 `pom.xml` 檔案。

⚠ Important

將外掛程式的版本取代為最新版本，如 [GitHub 儲存庫](#) 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

您可以為應用程式建立組態檔案，以指定 DataStax Java Cassandra 驅動程式的設定。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 連線至 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. 身分驗證提供者 – 將 `advanced.auth-provider.class` 設定為 的新執行個體 `software.aws.mcs.auth.SigV4AuthProvider`。 `SigV4AuthProvider` 是由外掛程式提供的身分驗證處理常式，用於執行 SigV4 身分驗證。
2. 本機資料中心 – 將 的值設定為您要連線 `local-datacenter` 的區域。例如，如果應用程式正在連線至 `cassandra.us-east-2.amazonaws.com`，請將本機資料中心設定為 `us-east-2`。如需所有可用的 AWS 區域，請參閱 [???](#)。 `slow-replica-avoidance = false` 設定為針對所有可用節點進行負載平衡。
3. 冪等性 – 將應用程式的預設值設定為 `true`，以設定驅動程式一律重試失敗 `idempotence` 的 `read/write/prepare` 執行請求。這是分散式應用程式的最佳實務，可透過重試失敗的請求，協助處理暫時性故障。
4. SSL/TLS – 初始化 `SSLConnectionFactory`，方法是在組態檔案中新增一個區段，並使用 指定類別 `class = DefaultSslConnectionFactory`。提供 `trustStore` 檔案的路徑以及您先前建立的密碼。Amazon Keyspaces 不支援 `hostname-validation` 對等，因此請將此選項設定為 `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`。這是最佳實務，可協助應用程式處理額外負荷和流量暴增。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 – 實作 Amazon Keyspaces 重試政

策DefaultRetryPolicy, AmazonKeyspacesExponentialRetryPolicy而非 Cassandra 驅動程式隨附的。這可讓您為AmazonKeyspacesExponentialRetryPolicy符合您需求的 設定重試嘗試次數。根據預設, 的重試嘗試次數AmazonKeyspacesExponentialRetryPolicy設定為 3。如需詳細資訊, 請參閱[the section called “如何設定重試政策”](#)。

7. 預備陳述式 – prepare-on-all-nodes設定為 false 以最佳化網路用量。

```
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 `trustStore trustStore` 路徑，或是將 `trustStore` 路徑新增至 JVM 引數。

步驟 3：執行應用程式

此程式碼範例顯示簡單的命令列應用程式，該應用程式使用我們先前建立的組態檔案，建立與 Amazon Keyspaces 的連線集區。它透過執行簡單的查詢來確認已建立連線。

```

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` 區塊來建立連線，以確保一律關閉。如果您不使用 `try` 區塊，請記得關閉您的連線，以避免資源洩漏。

使用適用於 Apache Cassandra 的 3.x DataStax Java 驅動程式和 SigV4 身分驗證外掛程式連線至 Amazon Keyspaces

下一節說明如何使用 3.x 開放原始碼 DataStax Java 驅動程式的 SigV4 身分驗證外掛程式，讓 Apache Cassandra 存取 Amazon Keyspaces。外掛程式可從 [GitHub 儲存庫](#) 取得。

SigV4 身分驗證外掛程式可讓您在連線至 Amazon Keyspaces 時使用使用者和角色的 IAM 登入資料。此外掛程式不使用使用者名稱和密碼，而是使用存取金鑰簽署 API 請求。如需詳細資訊，請參閱 [the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

步驟 1：事前準備

若要執行此程式碼範例，您必須先完成下列任務。

- 遵循 中的步驟，為您的 IAM 使用者或角色建立登入資料 [the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。本教學假設存取金鑰會儲存為環境變數。如需詳細資訊，請參閱 [the section called “管理存取金鑰”](#)。
- 請依照 中的步驟 [the section called “開始之前”](#) 下載數位憑證、將其轉換為 `trustStore` 檔案，並將 JVM 引數中的金鑰存放區連接至您的應用程式。
- 將 Apache Cassandra 的 DataStax Java 驅動程式新增至您的 Java 專案。請確定您使用的驅動程式版本支援 Apache Cassandra 3.11.2。如需詳細資訊，請參閱 [DataStax Java Driver for Apache Cassandra 文件](#)。
- 將身分驗證外掛程式新增至您的應用程式。身分驗證外掛程式支援適用於 Apache Cassandra 的 DataStax Java 驅動程式 3.x 版。如果您使用的是 Apache Maven，或是可以使用 Maven 相依性的建置系統，請將下列相依性新增至您的 `pom.xml` 檔案。將外掛程式的版本取代為最新版本，如 [GitHub 儲存庫](#) 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：執行應用程式

此程式碼範例顯示簡單的命令列應用程式，可建立與 Amazon Keyspaces 的連線集區。它透過執行簡單的查詢來確認已建立連線。

```
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"));
    }
}
```

用量備註：

如需可用端點的清單，請參閱 [the section called “服務端點”](#)。

如需使用 Java 驅動程式搭配 Amazon Keyspaces 時的實用 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 Cassandra Python 用戶端驅動程式以程式設計方式存取 Amazon Keyspaces

在本節中，我們會示範如何使用 Python 用戶端驅動程式連線至 Amazon Keyspaces。若要為使用者和應用程式提供以程式設計方式存取 Amazon Keyspaces 資源的登入資料，您可以執行下列其中一項操作：

- 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，我們建議為所有 AWS 服務中使用的 IAM 使用者或角色建立 IAM 存取金鑰。Cassandra 用戶端驅動程式的 Amazon Keyspaces SigV4 身分驗證外掛程式可讓您使用 IAM 存取金鑰來驗證對 Amazon Keyspaces 的呼叫，而不是使用者名稱和密碼。如需詳細資訊，請參閱[the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

主題

- [開始之前](#)
- [使用適用於 Apache Cassandra 的 Python 驅動程式和服務特定憑證連線至 Amazon Keyspaces](#)
- [使用適用於 Apache Cassandra 的 DataStax Python 驅動程式和 SigV4 身分驗證外掛程式連線至 Amazon Keyspaces](#)

開始之前

您需要先完成下列任務，才能開始。

Amazon Keyspaces 需要使用 Transport Layer Security (TLS) 來協助保護與用戶端的連線。若要使用 TLS 連線至 Amazon Keyspaces，您需要下載 Amazon 數位憑證，並將 Python 驅動程式設定為使用 TLS。

下載下列數位憑證，並將檔案儲存在本機或主目錄中。

1. AmazonRootCA1
2. AmazonRootCA2

3. AmazonRootCA3
4. AmazonRootCA4
5. Starfield Class 2 根目錄 (選用 – 用於回溯相容性)

若要下載憑證，您可以使用下列命令。

```
curl -O https://www.amazontrust.com/repository/AmazonRootCA1.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA2.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA3.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA4.pem
curl -O https://certs.secureserver.net/repository/sf-class2-root.crt
```

Note

Amazon Keyspaces 先前使用錨定至 Starfield 類別 2 CA 的 TLS 憑證。AWS 正在將所有遷移 AWS 區域至根據 Amazon Trust Services (Amazon 根 CAs 發行的憑證。在此轉換期間，請將用戶端設定為信任 Amazon 根 CAs1–4 和 Starfield 根，以確保所有區域的相容性。

使用範例中的名稱 *keyspaces-bundle.pem*，將所有下載的憑證合併成單一 pem 檔案。您可以執行下列命令，即可進行新增：請記下檔案的路徑，您稍後會需要此項目。

```
cat AmazonRootCA1.pem \
    AmazonRootCA2.pem \
    AmazonRootCA3.pem \
    AmazonRootCA4.pem \
    sf-class2-root.crt \
    > keyspaces-bundle.pem
```

使用適用於 Apache Cassandra 的 Python 驅動程式和服務特定憑證連線至 Amazon Keyspaces

下列程式碼範例示範如何使用 Python 用戶端驅動程式和服務特定的登入資料連線到 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/keyspaces-bundle.pem')

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)

```

用量備註：

1. "[path_to_file/keyspaces-bundle.pem](#)" 以第一個步驟中儲存的合併憑證檔案路徑取代。
2. 依照 的步驟，確保 [ServiceUserName](#) 和 [ServicePassword](#) 與您產生服務特定登入資料時取得的使用者名稱和密碼相符[建立服務特定的登入資料](#)，以程式設計方式存取 [Amazon Keyspaces](#)。
3. 如需可用端點的清單，請參閱 [the section called “服務端點”](#)。

使用適用於 Apache Cassandra 的 DataStax Python 驅動程式和 SigV4 身分驗證外掛程式連線至 Amazon Keyspaces

下一節說明如何使用開放原始碼 DataStax Python 驅動程式的 SigV4 身分驗證外掛程式，讓 Apache Cassandra 存取 Amazon Keyspaces（適用於 Apache Cassandra）。

如果您尚未這麼做，請先依照 中的步驟，為您的 IAM 角色建立登入資料[the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。本教學課程使用臨時登入資料，需要 IAM 角色。如需臨時登入資料的詳細資訊，請參閱 [the section called “建立臨時登入資料以連線至 Amazon Keyspaces”](#)。

然後，從 [GitHub 儲存庫](#) 將 Python SigV4 身分驗證外掛程式新增至您的環境。

```
pip install cassandra-sigv4
```

下列程式碼範例示範如何使用適用於 Cassandra 的開放原始碼 DataStax Python 驅動程式和 SigV4 身分驗證外掛程式連線至 Amazon Keyspaces。外掛程式取決於適用於 Python 的 AWS SDK (Boto3)。它使用 `boto3.session` 來取得臨時登入資料。

```

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/keyspaces-bundle.pem')
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)

```

用量備註：

1. "[path_to_file/keyspaces-bundle.pem](#)" 將取代為第一個步驟中儲存的憑證路徑。
2. 確定 [aws_access_key_id](#)、[aws_secret_access_key](#) 和 [aws_session_token](#) 符合 Access Key、Secret Access Key Session Token 以及您使用取得的 `boto3.session`。如需詳細資訊，請參閱 [登入資料](#) 適用於 Python (Boto3) 的 AWS SDK。
3. 如需可用端點的清單，請參閱 [the section called “服務端點”](#)。

使用 Cassandra Node.js 用戶端驅動程式以程式設計方式存取 Amazon Keyspaces

本節說明如何使用 Node.js 用戶端驅動程式連線至 Amazon Keyspaces。若要為使用者和應用程式提供以程式設計方式存取 Amazon Keyspaces 資源的登入資料，您可以執行下列其中一項操作：

- 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，我們建議為所有 AWS 服務中使用的 IAM 使用者或角色建立 IAM 存取金鑰。Cassandra 用戶端驅動程式的 Amazon Keyspaces SigV4 身分驗證外掛程式可讓您使用 IAM 存取金鑰來驗證對 Amazon Keyspaces 的呼叫，而不是使用者名稱和密碼。如需詳細資訊，請參閱[the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

主題

- [開始之前](#)
- [使用適用於 Apache Cassandra 的 Node.js DataStax 驅動程式和服務特定登入資料連線至 Amazon Keyspaces](#)
- [使用適用於 Apache Cassandra 的 DataStax Node.js 驅動程式和 SigV4 身分驗證外掛程式連線至 Amazon Keyspaces](#)

開始之前

您需要先完成下列任務，才能開始。

Amazon Keyspaces 需要使用 Transport Layer Security (TLS) 來協助保護與用戶端的連線。若要使用 TLS 連線至 Amazon Keyspaces，您需要下載 Amazon 數位憑證，並將 Python 驅動程式設定為使用 TLS。

下載下列數位憑證，並將檔案儲存在本機或主目錄中。

1. AmazonRootCA1
2. AmazonRootCA2
3. AmazonRootCA3
4. AmazonRootCA4
5. Starfield Class 2 根目錄（選用 – 用於回溯相容性）

若要下載憑證，您可以使用下列命令。

```
curl -O https://www.amazontrust.com/repository/AmazonRootCA1.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA2.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA3.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA4.pem
curl -O https://certs.secureserver.net/repository/sf-class2-root.crt
```

 Note

Amazon Keyspaces 先前使用錨定至 Starfield 類別 2 CA 的 TLS 憑證。AWS 正在將所有 遷移 AWS 區域 至根據 Amazon Trust Services (Amazon 根 CAs 發行的憑證。在此轉換期間，請將用戶端設定為信任 Amazon 根 CAs1–4 和 Starfield 根，以確保所有區域的相容性。

使用範例中的名稱 *keyspaces-bundle.pem*，將所有下載的憑證合併成單一 pem 檔案。您可以執行下列 命令，即可進行新增：請記下 檔案的路徑，您稍後會需要此項目。

```
cat AmazonRootCA1.pem \  
    AmazonRootCA2.pem \  
    AmazonRootCA3.pem \  
    AmazonRootCA4.pem \  
    sf-class2-root.crt \  
    > keyspaces-bundle.pem
```

使用適用於 Apache Cassandra 的 Node.js DataStax 驅動程式和服務特定登入資料連線至 Amazon Keyspaces

將驅動程式設定為使用 TLS *keyspaces-bundle.pem* 的合併憑證檔案，並使用服務特定的憑證進行驗證。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/keyspaces-bundle.pem', '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}`));
```

用量備註：

1. "[path_to_file/keyspaces-bundle.pem](#)" 以第一個步驟中儲存的合併憑證檔案路徑取代。
2. 依照 的步驟，確保 [ServiceUserName](#) 和 [ServicePassword](#) 與您產生服務特定登入資料時取得的使用者名稱和密碼相符[建立服務特定的登入資料，以程式設計方式存取 Amazon Keyspaces](#)。
3. 如需可用端點的清單，請參閱 [the section called “服務端點”](#)。

使用適用於 Apache Cassandra 的 DataStax Node.js 驅動程式和 SigV4 身分驗證外掛程式連線至 Amazon Keyspaces

下一節說明如何針對 Apache Cassandra 的開放原始碼 DataStax Node.js 驅動程式使用 SigV4 身分驗證外掛程式來存取 Amazon Keyspaces（適用於 Apache Cassandra）。

如果您尚未這麼做，請依照 中的步驟為 IAM 使用者或角色建立登入資料[the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

從 [GitHub 儲存庫](#) 將 Node.js SigV4 身分驗證外掛程式新增至您的應用程式。外掛程式支援適用於 Cassandra 的 DataStax Node.js 驅動程式 4.x 版，並取決於適用於 Node.js 的 AWS SDK。它使用 `AWSCredentialsProvider` 來取得登入資料。

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

此程式碼範例示範如何將 的區域特定執行個體設定為 `SigV4AuthProvider` 身分驗證提供者。

```
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_file/keyspaces-bundle.pem', '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}`));
```

用量備註：

1. "[path_to_file/keyspaces-bundle.pem](#)" 將取代為第一個步驟中儲存的憑證路徑。
2. 確定 [accessKeyId](#) 和 [secretAccessKey](#) 與您使用 取得的存取金鑰和私密存取金鑰相符 [AWSCredentialsProvider](#)。如需詳細資訊，請參閱 [Node.js 中適用於 JavaScript 的 SDK 中的在 Node.js 中設定登入](#) 資料。AWS JavaScript
3. 若要將存取金鑰存放在程式碼之外，請參閱 的最佳實務 [the section called “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 Cassandra .NET Core 用戶端驅動程式以程式設計方式存取 Amazon Keyspaces

本節說明如何使用 .NET Core 用戶端驅動程式連線至 Amazon Keyspaces。設定步驟會根據您的環境和作業系統而有所不同，您可能需要相應地進行修改。Amazon Keyspaces 需要使用 Transport Layer Security (TLS) 來協助保護與用戶端的連線。若要使用 TLS 連線至 Amazon Keyspaces，請將驅動程式設定為使用系統信任存放區，其中包含 Amazon 根 CAs1-4。

1. 使用 nuget 主控台透過 nuget 安裝 CassandraCSharpDriver。

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

2. 下列範例使用 .NET Core C# 主控台專案連線至 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)
        {
            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);
    }
}
}
```

用量備註：

1. 請確定您使用預設的系統信任存放區，其中包含 Amazon 根 CAs1-4。
2. 依照 的步驟，確定 *ServiceUserName* 和 *ServicePassword* 與您產生服務特定登入資料時取得的使用者名稱和密碼相符[建立服務特定的登入資料，以程式設計方式存取 Amazon Keyspaces](#)。
3. 如需可用端點的清單，請參閱 [the section called “服務端點”](#)。

使用 Cassandra Go 用戶端驅動程式以程式設計方式存取 Amazon Keyspaces

本節說明如何使用 Go Cassandra 用戶端驅動程式連線至 Amazon Keyspaces。若要為使用者和應用程式提供以程式設計方式存取 Amazon Keyspaces 資源的登入資料，您可以執行下列其中一項操作：

- 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，我們建議為所有 AWS 服務中使用的 IAM 主體建立 IAM 存取金鑰。Cassandra 用戶端驅動程式的 Amazon Keyspaces SigV4 身分驗證外掛程式可讓您使用 IAM 存取金鑰來驗證對 Amazon Keyspaces 的呼叫，而不是使用者名稱和密碼。如需詳細資訊，請參閱[the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

主題

- [開始之前](#)
- [使用適用於 Apache Cassandra 的 Gocql 驅動程式和服務特定憑證連線至 Amazon Keyspaces](#)
- [使用適用於 Apache Cassandra 的 Go 驅動程式和 SigV4 身分驗證外掛程式連線至 Amazon Keyspaces](#)

開始之前

您需要先完成下列任務，才能開始。

Amazon Keyspaces 需要使用 Transport Layer Security (TLS) 來協助保護與用戶端的連線。若要使用 TLS 連線至 Amazon Keyspaces，您需要下載 Amazon 數位憑證，並將 Go 驅動程式設定為使用 TLS。

下載下列數位憑證，並將檔案儲存在本機或主目錄中。

1. AmazonRootCA1
2. AmazonRootCA2
3. AmazonRootCA3
4. AmazonRootCA4
5. Starfield Class 2 根目錄（選用 – 用於回溯相容性）

若要下載憑證，您可以使用下列命令。

```
curl -O https://www.amazontrust.com/repository/AmazonRootCA1.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA2.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA3.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA4.pem
curl -O https://certs.secureserver.net/repository/sf-class2-root.crt
```

Note

Amazon Keyspaces 先前使用錨定至 Starfield 類別 2 CA 的 TLS 憑證。AWS 正在將所有遷移 AWS 區域至根據 Amazon Trust Services (Amazon 根 CAs 發行的憑證。在此轉換期間，請將用戶端設定為信任 Amazon 根 CAs1–4 和 Starfield 根，以確保所有區域的相容性。

使用範例中的名稱 *keyspaces-bundle.pem*，將所有下載的憑證合併成單一 pem 檔案。您可以執行下列命令，即可進行新增：請記下檔案的路徑，您稍後會需要此項目。

```
cat AmazonRootCA1.pem \
    AmazonRootCA2.pem \
    AmazonRootCA3.pem \
    AmazonRootCA4.pem \
    sf-class2-root.crt \
    > keyspaces-bundle.pem
```

使用適用於 Apache Cassandra 的 Gocql 驅動程式和服務特定憑證連線至 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 keyspaces-bundle.pem
    cluster.SslOpts = &gocql.SslOptions{
        CaPath: "path_to_file/keyspaces-bundle.pem",
```

```
        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()
}
```

用量備註：

- a. "*path_to_file/keyspaces-bundle.pem*" 以第一個步驟中儲存的合併憑證檔案路徑取代。
 - b. 依照 的步驟，確保 *ServiceUserName* 和 *ServicePassword* 與您產生服務特定登入資料時取得的使用者名稱和密碼相符 [建立服務特定的登入資料，以程式設計方式存取 Amazon Keyspaces](#)。
 - 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
```

使用適用於 Apache Cassandra 的 Go 驅動程式和 SigV4 身分驗證外掛程式連線至 Amazon Keyspaces

下列程式碼範例示範如何使用開放原始碼 Go 驅動程式的 SigV4 身分驗證外掛程式來存取 Amazon Keyspaces (適用於 Apache Cassandra)。

如果您尚未這麼做，請依照 中的步驟建立 IAM 主體登入資料[the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。如果應用程式在 Lambda 或 Amazon EC2 執行個體上執行，您的應用程式會自動使用執行個體的登入資料。若要在本機執行本教學課程，您可以將登入資料儲存為本機環境變數。

從 [GitHub 儲存庫](#) 將 Go SigV4 身分驗證外掛程式新增至您的應用程式。外掛程式支援適用於 Cassandra 的開放原始碼 Go 驅動程式 1.2.x 版，並取決於適用於 Go 的 AWS SDK。

```
$ 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。它使用 AwsAuthenticator 做為叢集的驗證器屬性，以取得登入資料。

```
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

    // the authenticator uses the default credential chain to find AWS credentials
    cluster.Authenticator = sigv4.NewAwsAuthenticator()

    cluster.SslOpts = &gocql.SslOptions{
        CaPath: "path_to_file/keyspaces-bundle.pem",
        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)
}
}
```

用量備註：

1. "[*path_to_file/keyspaces-bundle.pem*](#)" 將取代為第一個步驟中儲存的憑證檔案路徑。
2. 若要在本機執行此範例，您需要將下列變數定義為環境變數：
 - AWS_ACCESS_KEY_ID
 - AWS_SECRET_ACCESS_KEY
 - AWS_DEFAULT_REGION
3. 若要將存取金鑰存放在程式碼之外，請參閱 [the section called “管理存取金鑰”](#) 的最佳實務。
4. 如需可用端點的清單，請參閱 [the section called “服務端點”](#)。

使用 Cassandra Perl 用戶端驅動程式以程式設計方式存取 Amazon Keyspaces

本節說明如何使用 Perl 用戶端驅動程式連線至 Amazon Keyspaces。對於此程式碼範例，我們使用 Perl 5。Amazon Keyspaces 需要使用 Transport Layer Security (TLS) 來協助保護與用戶端的連線。

⚠ Important

Amazon Keyspaces certificates 正在轉換至 Amazon Trust Services (ATS) 階層。確保您的環境信任 Amazon 根 CAs1–4，以避免在此輪換期間發生連線錯誤。Perl 驅動程式不會驗證伺服器的 Amazon SSL 憑證，這表示您無法確認您連線到 Amazon Keyspaces。第二個步驟是設定驅動程式在連線至 Amazon Keyspaces 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 Cassandra DBI 驅動程式，<https://metacpan.org/pod/DBD::Cassandra> 並將驅動程式安裝到您的 Perl 環境。確切的步驟取決於環境。以下是常見的範例。

```
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

依照 的步驟，確定 *ServiceUserName* 和 *ServicePassword* 與您產生服務特定登入資料時取得的使用者名稱和密碼相符 [建立服務特定的登入資料](#)，以程式設計方式存取 [Amazon Keyspaces](#)。

 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
```

使用 VPC 端點設定 Amazon Keyspaces 的跨帳戶存取權

您可以建立並使用個別 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。本主題使用 中的介面 VPC 端點，逐步引導您存取 Amazon Keyspaces 的跨帳戶 Amazon Virtual Private Cloud。如需 IAM 跨帳戶存取組態的詳細資訊，請參閱《IAM 使用者指南》中的 [使用個別開發和生產帳戶的範例案例](#)。

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

- [使用共用 VPC 中的 VPC 端點設定 Amazon Keyspaces 的跨帳戶存取權](#)
- [設定沒有共用 VPC 的 Amazon Keyspaces 跨帳戶存取權](#)

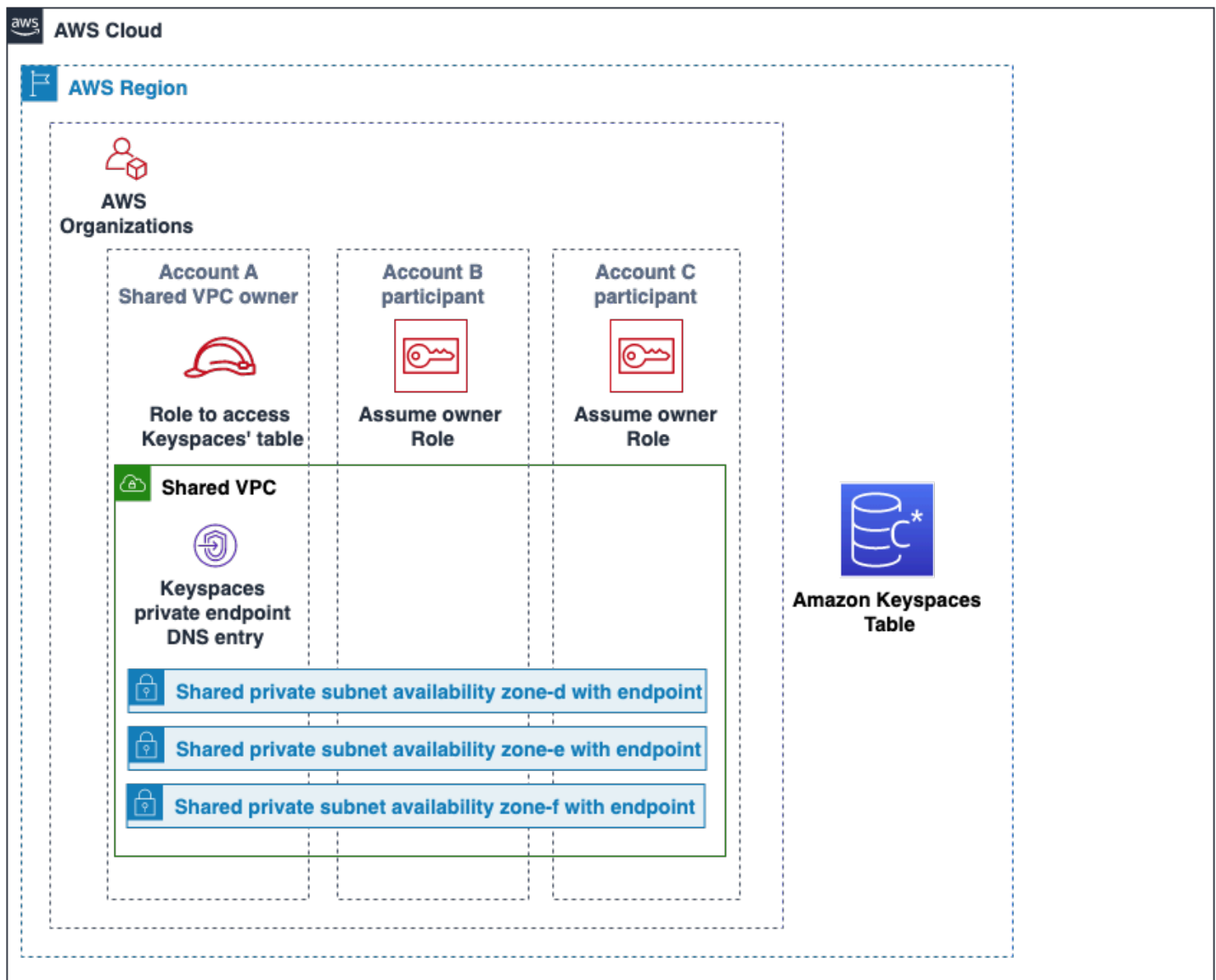
使用共用 VPC 中的 VPC 端點設定 Amazon Keyspaces 的跨帳戶存取權

您可以建立不同的 AWS 帳戶，將資源與應用程式分開。例如，您可以為 Amazon Keyspaces 資料表建立一個帳戶、為開發環境中的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。本主題會逐步解說在共用 VPC 中使用介面 VPC 端點設定 Amazon Keyspaces 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：建立 Amazon Keyspaces 的 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:111111111111 – 此帳戶包含基礎設施，包括 VPC 端點、VPC 子網路 and Amazon Keyspaces 資料表。
- Account B:222222222222 – 此帳戶包含在開發環境中的應用程式，需要連線到 中的 Amazon Keyspaces 資料表Account A:111111111111。
- Account C:333333333333 – 此帳戶包含生產環境中的應用程式，需要連線到 中的 Amazon Keyspaces 資料表Account A:111111111111。



Account A:111111111111 是包含 Account B:222222222222 和 Account C:333333333333 需要存取之資源 (Amazon Keyspaces 資料表) 的帳戶，Account A:111111111111 是信任帳戶。Account B:222222222222 和 Account C:333333333333 是具有中需要存取資源 (Amazon Keyspaces 資料表) 之主體的帳戶 Account A:111111111111，因此 Account B:222222222222 和 Account C:333333333333 是信任帳戶。信任帳戶透過共用 IAM 角色，將許可授予信任的帳戶。下列程序概述中所需的組態步驟 Account A:111111111111。

的組態 **Account A:111111111111**

1. 使用 為子網路 AWS Resource Access Manager 建立資源共用，並與 Account B:222222222222 和 共用私有子網路 Account C:333333333333。

Account B:222222222222 和 Account C:333333333333 現在可以在已與其共用的子網路中查看和建立資源。

2. 建立採用技術的 Amazon Keyspaces 私有 VPC 端點 AWS PrivateLink。這會在共用子網路和 Amazon Keyspaces 服務端點的 DNS 項目之間建立多個端點。
3. 建立 Amazon Keyspaces 金鑰空間和資料表。
4. 在中建立 IAM 角色 Account A:111111111111，該角色具有 Amazon Keyspaces 資料表的完整存取權、Amazon Keyspaces 系統資料表的讀取存取權，並且能夠描述 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. 在中設定 IAM 角色的信任政策，Account A:111111111111 讓 Account B:222222222222 和 Account C:333333333333 可以擔任該角色做為信任的帳戶。如以下範例所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::222222222222:role/Cross-Account-Role-B",
          "arn:aws:iam::333333333333:role/Cross-Account-Role-C"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

如需跨帳戶 IAM 政策的詳細資訊，請參閱《IAM 使用者指南》中的[跨帳戶政策](#)。

Account B:222222222222 和 中的組態 Account C:333333333333

1. 在 Account B:222222222222 和 中 Account C:333333333333，建立新的角色並連接下列政策，以允許委託人擔任在 中建立的共用角色 Account A:111111111111。

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

允許委託人擔任共用角色是使用 AWS Security Token Service (AWS STS) 的 AssumeRole API 實作。如需詳細資訊，請參閱《[IAM 使用者指南](#)》中的在您擁有 AWS 帳戶 的另一個 IAM 使用者 [中提供存取權](#)。

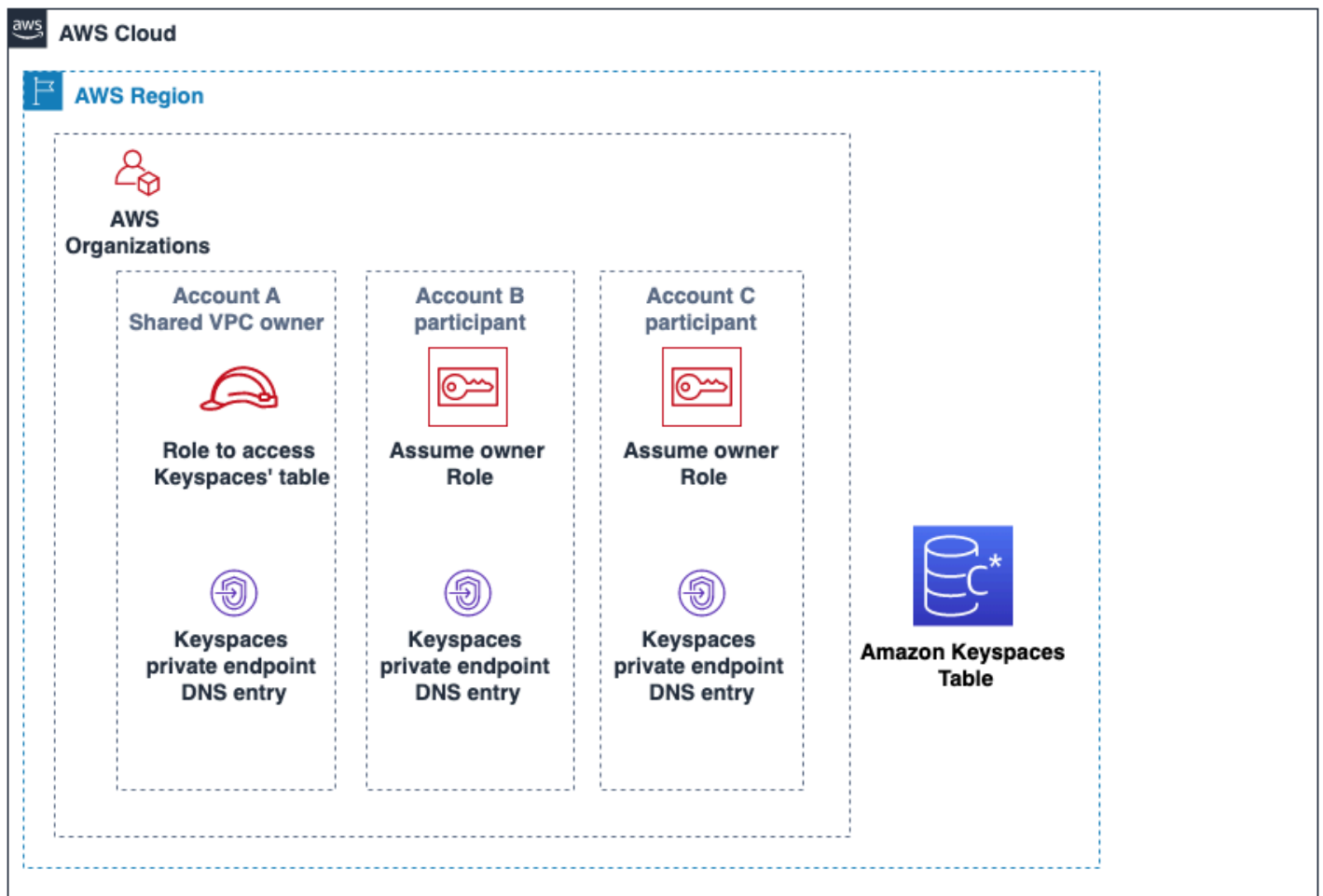
2. 在 Account B:222222222222 和 Account C:333333333333，您可以建立使用 SIGV4 身分驗證外掛程式的應用程式，允許應用程式擔任共用角色，Account A:111111111111 透過共用 VPC 中的 VPC 端點連線到位於其中的 Amazon Keyspaces 資料表。如需 SIGV4 身分驗證外掛程式的詳細資訊，請參閱 [the section called “建立程式設計存取憑證”](#)。如需如何設定應用程式以在另一個 AWS 帳戶中擔任角色的詳細資訊，請參閱 AWS SDKs 和工具參考指南中的 [身分驗證和存取](#)。

設定沒有共用 VPC 的 Amazon Keyspaces 跨帳戶存取權

如果 Amazon Keyspaces 資料表和私有 VPC 端點為不同帳戶所擁有，但未共用 VPC，則應用程式仍然可以使用 VPC 端點來連接跨帳戶。由於帳戶不會共用 VPC 端點，Account B:222222222222 因此 Account A:111111111111、和 Account C:333333333333 需要自己的 VPC 端點。對於 Cassandra 用戶端驅動程式，Amazon Keyspaces 會顯示為單一節點，而不是多節點叢集。連線時，用戶端驅動程式會到達 DNS 伺服器，傳回帳戶 VPC 中其中一個可用的端點。

您也可以使用公有端點或在每個帳戶中部署私有 VPC 端點，存取不同帳戶的 Amazon Keyspaces 資料表，而無需共用 VPC 端點。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 A:111111111111、Account B:222222222222 和 Account C:333333333333 需要自己的 VPC 端點才能存取其中的資料表 Account A:111111111111。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，而非多節點叢集。連線時，用戶端驅動程式會到達 DNS 伺服器，傳回帳戶 VPC 中其中一個可用的端點。但用戶端驅動程式無法存取 `system.peers` 資料表來探索其他端點。由於可用的主機較少，驅動程式的連線較少。若要調整此值，請將驅動程式的連線集區設定增加 3 倍。



Account A:111111111111 是包含 Account B:222222222222 和 Account C:333333333333 需要存取之資源 (Amazon Keyspaces 資料表) 的帳戶，Account A:111111111111 是信任帳戶。Account B:222222222222 和 Account C:333333333333 是具有中需要存取資源 (Amazon Keyspaces 資料表) 之主體的帳戶 Account A:111111111111，因此 Account B:222222222222 和 Account C:333333333333 是信任帳戶。信任帳戶透過共用 IAM 角色，將許可授予信任的帳戶。下列程序概述中所需的組態步驟 Account A:111111111111。

的組態 **Account A:111111111111**

1. 在中建立 Amazon Keyspaces 金鑰空間和資料表 Account A:111111111111。
2. 在中建立 Account A:111111111111 具有 Amazon Keyspaces 資料表完整存取權和 Amazon Keyspaces 系統資料表讀取存取權的 IAM 角色。

```
{
  "Version": "2012-10-17",
  "Statement": [
```

```
{
  "Effect": "Allow",
  "Action": [
    "cassandra:Select",
    "cassandra:Modify"
  ],
  "Resource": [
    "arn:aws:cassandra:us-east-1:111111111111:/keyspace/mykeyspace/table/mytable",
    "arn:aws:cassandra:us-east-1:111111111111:/keyspace/system*"
  ]
}
```

3. 在 中設定 IAM 角色的信任政策，Account A:111111111111 讓 Account B:222222222222 和 中的主體 Account C:333333333333 可以擔任信任帳戶的角色。如以下範例所示。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::222222222222:role/Cross-Account-Role-B",
          "arn:aws:iam::333333333333:role/Cross-Account-Role-C"
        ]
      },
      "Action": "sts:AssumeRole",
      "Condition": {}
    }
  ]
}
```

如需跨帳戶 IAM 政策的詳細資訊，請參閱《IAM 使用者指南》中的[跨帳戶政策](#)。

4. 在 中設定 VPC 端點，Account A:111111111111 並將許可連接到允許來自 Account B:222222222222 和 的角色 Account A 使用 VPC 端點 Account C:333333333333 擔任 中角色的端點。這些許可對其連接的 VPC 端點有效。如需 VPC 端點政策的詳細資訊，請參閱 [the section called “控制對 Amazon Keyspaces 介面 VPC 端點的存取”](#)。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessfromSpecificIAMroles",
      "Effect": "Allow",
      "Action": "cassandra:*",
      "Resource": "*",
      "Principal": "*",
      "Condition": {
        "ArnEquals": {
          "aws:PrincipalArn": [
            "arn:aws:iam::222222222222:role/Cross-Account-Role-B",
            "arn:aws:iam::333333333333:role/Cross-Account-Role-C"
          ]
        }
      }
    }
  ]
}
```

Account B:222222222222 和 中的組態 Account C:333333333333

1. 在 Account B:222222222222和 中Account C:333333333333，建立新的角色並連接下列政策，以允許委託人擔任在 中建立的共用角色Account A:111111111111。

```
{
  "Version": "2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::111111111111:role/keyspaces_access"
  }
}
```

允許委託人擔任共用角色是使用 AWS Security Token Service (AWS STS) 的 AssumeRole API 實作。如需詳細資訊，請參閱 [《IAM 使用者指南》](#) 中的在您擁有 AWS 帳戶 的另一個 中為 IAM 使用者提供存取權。

2. 在 Account B:222222222222 和 Account C:333333333333，您可以建立使用 SIGV4 身分驗證外掛程式的應用程式，允許應用程式擔任共用角色以連線到位於 中的 Amazon Keyspaces 資料表 Account A:111111111111。如需 SIGV4 身分驗證外掛程式的詳細資訊，請參閱 [the section called “建立程式設計存取憑證”](#)。如需如何設定應用程式以在另一個 AWS 帳戶中擔任角色的詳細資訊，請參閱 AWS SDKs 和工具參考指南中的 [身分驗證和存取](#)。

Amazon Keyspaces 入門 (適用於 Apache Cassandra)

如果您是初次使用 Apache Cassandra 和 Amazon Keyspaces，本教學課程會引導您完成安裝必要的程式和工具，以成功使用 Amazon Keyspaces。您將了解如何使用 Cassandra 查詢語言 (CQL) AWS 管理主控台、或 AWS Command Line Interface () 建立金鑰空間和資料表 AWS CLI。然後，您可以使用 Cassandra 查詢語言 (CQL) 對 Amazon Keyspaces 資料表中的資料執行建立、讀取、更新和刪除 (CRUD) 操作。

本教學課程涵蓋下列步驟。

- 先決條件 – 開始教學課程之前，請遵循 AWS 設定說明註冊 AWS 並建立可存取 Amazon Keyspaces 的 IAM 使用者。然後，設定 `cqlsh-expansion` 和 AWS CloudShell。或者，您可以使用 AWS CLI 在 Amazon Keyspaces 中建立資源。
- 步驟 1：建立金鑰空間和資料表 – 在此區段中，您將建立名為 "catalog" 的金鑰空間，以及其中名為 "book_awards" 的資料表。您將使用、CQL 或 指定資料表的資料欄、資料類型 AWS 管理主控台、分割區索引鍵和叢集資料欄 AWS CLI。
- 步驟 2：執行 CRUD 操作 – 在這裡，您將使用 CloudShell `cqlsh-expansion` 中的 來插入、讀取、更新和刪除「book_awards」資料表中的資料。您將了解如何使用各種 CQL 陳述式，例如 SELECT、INSERT、UPDATE 和 DELETE，以及練習篩選和修改資料。
- 步驟 3：清除資源 – 為了避免未使用的資源產生費用，本節將引導您使用主控台、CQL 或 刪除「book_awards」資料表和「catalog」鍵空間 AWS CLI。

如需使用不同 Apache Cassandra 用戶端驅動程式以程式設計方式連線至 Amazon Keyspaces 的教學課程，請參閱 [the section called “使用 Cassandra 用戶端驅動程式”](#)。如需使用不同 AWS SDKs 程式碼範例，請參閱 [使用 AWS SDKs 的 Amazon Keyspaces 程式碼範例](#)。

主題

- [教學的先決條件和考量事項](#)
- [在 Amazon Keyspaces 中建立金鑰空間](#)
- [檢查 Amazon Keyspaces 中的金鑰空間建立狀態](#)
- [在 Amazon Keyspaces 中建立資料表](#)
- [在 Amazon Keyspaces 中檢查資料表建立狀態](#)
- [在 Amazon Keyspaces 中使用 CQL 建立、讀取、更新和刪除資料 \(CRUD\)](#)
- [在 Amazon Keyspaces 中刪除資料表](#)

- [在 Amazon Keyspaces 中刪除金鑰空間](#)

教學的先決條件和考量事項

在開始使用 Amazon Keyspaces 之前，請遵循中的 AWS 設定說明[存取 Amazon Keyspaces \(適用於 Apache Cassandra\)](#)。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。

若要完成教學課程的所有步驟，您需要安裝 cqlsh。您可以遵循的設定指示[使用 cqlsh 連線至 Amazon Keyspaces](#)。

若要使用 cqlsh 或 存取 Amazon Keyspaces AWS CLI，建議使用 AWS CloudShell。CloudShell 是以瀏覽器為基礎、預先驗證身分的 Shell，您可以直接從 AWS 管理主控台啟動。您可以使用您偏好的 shell (Bash、PowerShell 或 Z shell AWS CLI) 對 Amazon Keyspaces 執行 AWS Command Line Interface () 命令。若要使用 cqlsh，您必須安裝 cqlsh-expansion。如需 cqlsh-expansion 安裝說明，請參閱 [the section called “使用 cqlsh-expansion”](#)。如需 CloudShell 的詳細資訊，請參閱 [the section called “使用 AWS CloudShell”](#)。

若要使用 AWS CLI 在 Amazon Keyspaces 中建立、檢視和刪除資源，請遵循中的設定說明[the section called “下載和設定 AWS CLI”](#)。

完成先決條件步驟後，請繼續 [在 Amazon Keyspaces 中建立金鑰空間](#)。

在 Amazon Keyspaces 中建立金鑰空間

在本節中，您會使用主控台、cqlsh 或 建立金鑰空間 AWS CLI。

Note

開始之前，請確定您已設定所有[教學的先決條件](#)。

與一或多個應用程式相關的鍵空間群組相關資料表。金鑰空間包含一或多個資料表，並為其包含的所有資料表定義複寫策略。如需金鑰空間的詳細資訊，請參閱下列主題：

- CQL 語言參考中的資料定義語言 (DDL) 陳述式：[Keyspaces](#)
- [Amazon Keyspaces 配額 \(適用於 Apache Cassandra\)](#)

在本教學課程中，我們會建立單一區域金鑰空間，而金鑰空間的複寫策略為 `SingleRegionStrategy`。Amazon Keyspaces 使用 `SingleRegionStrategy` 在一個可用區域中跨三個可用區域複寫資料 AWS 區域。若要了解如何建立多區域金鑰空間，請參閱 [the section called “建立多區域金鑰空間”](#)。

使用主控台

使用主控台建立金鑰空間

1. 登入 AWS 管理主控台，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 金鑰空間。
3. 選擇建立金鑰空間。
4. 在金鑰空間名稱方塊中，輸入 **catalog** 做為金鑰空間的名稱。

名稱限制條件：

- 名稱不可為空白。
 - 允許的字元：英數字元和底線 `_` ()。
 - 長度上限為 48 個字元。
5. 在下 AWS 區域，確認單一區域複寫是 金鑰空間的複寫策略。
 6. 若要建立金鑰空間，請選擇建立金鑰空間。
 7. 執行下列動作，確認 **catalog** 已建立 金鑰空間：
 - a. 在導覽窗格中，選擇 金鑰空間。
 - b. **catalog** 在金鑰空間清單中找到您的金鑰空間。

使用 CQL

下列程序會使用 CQL 建立金鑰空間。

使用 CQL 建立金鑰空間

1. 使用下列命令開啟 AWS CloudShell 並連線至 Amazon Keyspaces。請務必使用您自己的區域更新 ***us-east-1***。

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

該命令的輸出應如下所示。

```
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.
```

2. 使用下列 CQL 命令建立您的金鑰空間。

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

SingleRegionStrategy 使用 3 的複寫係數，並在其區域中的三個 AWS 可用區域中複寫資料。

Note

Amazon Keyspaces 會將所有輸入預設為小寫，除非您將其括在引號中。

3. 確認您的 金鑰空間已建立。

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

此命令的輸出看起來應該與此類似。

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

keyspace_name          | durable_writes | replication
-----+-----
+-----+-----+-----
      system_schema |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
      system_schema_mcs |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
           system |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
system_multiregion_info |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
           catalog |             True | {'class':
'org.apache.cassandra.locator.SimpleStrategy', 'replication_factor': '3'}
```

```
(5 rows)
```

使用 AWS CLI

下列程序會使用 建立金鑰空間 AWS CLI。

使用 建立金鑰空間 AWS CLI

1. 若要確認您的環境已設定，您可以在 CloudShell 中執行下列命令。

```
aws keyspaces help
```

2. 使用下列 AWS CLI 陳述式建立您的 金鑰空間。

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

3. 確認您的 金鑰空間已使用下列 AWS CLI 陳述式建立

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

此命令的輸出看起來應該與此範例類似。

```
{
  "keyspaceName": "catalog",
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/catalog/",
  "replicationStrategy": "SINGLE_REGION"
}
```

檢查 Amazon Keyspaces 中的金鑰空間建立狀態

Amazon Keyspaces 會以非同步方式執行資料定義語言 (DDL) 操作，例如建立和刪除金鑰空間。

您可以在 中監控新金鑰空間的建立狀態 AWS 管理主控台，這表示金鑰空間何時處於待定或作用中狀態。您也可以使用 金鑰空間，以程式設計方式監控新system_schema_mcs金鑰空間的建立狀態。準備好使用時，鍵空間會出現在system_schema_mcskeyspaces資料表中。

檢查新金鑰空間何時可供使用的建議設計模式是輪詢 Amazon Keyspaces system_schema_mcskeyspaces資料表 (system_schema_mcs.*)。如需金鑰空間的 DDL 陳述式清單，請參閱 CQL 語言參考中的[the section called “Keyspaces”](#)一節。

下列查詢顯示是否已成功建立金鑰空間。

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

對於已成功建立的金鑰空間，查詢的輸出如下所示。

```
keyspace_name | durable_writes | replication
-----+-----+-----
mykeyspace | true | {...} 1 item
```

在 Amazon Keyspaces 中建立資料表

在本節中，您會使用 主控台、cqlsh 或 建立資料表 AWS CLI。

資料表是資料整理和存放的位置。資料表的主索引鍵決定如何在資料表中分割資料。主索引鍵由必要的分割區索引鍵和一或多個選用叢集資料欄組成。構成主索引鍵的組合值在所有資料表的資料中必須是唯一的。如需資料表的詳細資訊，請參閱下列主題：

- 分割區索引鍵設計：[the section called “分割區索引鍵設計”](#)
- 使用資料表：[the section called “檢查資料表建立狀態”](#)
- CQL 語言參考中的 DDL 陳述式：[表格](#)
- 資料表資源管理：[管理無伺服器資源](#)
- 監控資料表資源使用率：[the section called “使用 CloudWatch 進行監控”](#)
- [Amazon Keyspaces 配額 \(適用於 Apache Cassandra\)](#)

建立資料表時，您可以指定下列項目：

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

使用下列程序建立具有指定資料欄、資料類型、分割區索引鍵和叢集資料欄的資料表。

使用主控台

下列程序會建立 `book_awards` 具有這些資料欄和資料類型的資料表。

```
year          int
award         text
rank          int
category      text
book_title    text
author        text
publisher     text
```

使用主控台建立資料表

1. 登入 AWS 管理主控台，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 金鑰空間。
3. 選擇 `catalog` 做為您要建立此資料表的金鑰空間。
4. 選擇 Create Table (建立資料表)。
5. 在資料表名稱方塊中，輸入 **book_awards** 做為資料表的名稱。

名稱限制條件：

- 名稱不可為空白。
 - 允許的字元：英數字元和底線 `_` ()。
 - 長度上限為 48 個字元。
6. 在資料欄區段中，針對您要新增至此表格的每個資料欄重複下列步驟。

新增下列資料欄和資料類型。

```
year          int
award         text
rank          int
category      text
book_title    text
author        text
publisher     text
```

- a. 名稱 – 輸入資料欄的名稱。

名稱限制條件：
 - 名稱不可為空白。
 - 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. 若要新增其他資料欄，請選擇新增資料欄。
7. 選擇 `award` 和 `year` 做為分割區索引鍵下的分割區索引鍵。每個資料表都需要分割區索引鍵。分割區索引鍵可由一或多個資料欄組成。
 8. 新增 `category` 和 `rank` 做為叢集資料欄。叢集資料欄是選用的，並決定每個分割區內的排序順序。
 - a. 若要新增叢集資料欄，請選擇新增叢集資料欄。
 - b. 在資料欄清單中，選擇類別。在順序清單中，選擇 `ASC` 以遞增順序排序此欄中的值。（選擇遞減順序的 `DESC`。）
 - c. 然後選取新增叢集欄，然後選擇排名。
 9. 在資料表設定區段中，選擇預設設定。
 10. 選擇 `Create Table` (建立資料表)。
 11. 確認您的資料表已建立。
 - a. 在導覽窗格中，選擇 `Tables` (資料表)。
 - b. 確認您的資料表位於資料表清單中。
 - c. 選擇資料表的名稱。
 - d. 確認您的所有資料欄和資料類型都正確。

 Note

這些欄可能不會以您將其新增至資料表的相同順序列出。

使用 CQL

此程序會使用 CQL 建立具有下列資料欄和資料類型的資料表。year 和 award 欄是具有 category 和 rank 做為叢集欄的分割區索引鍵，它們共同構成資料表的主索引鍵。

year	int
award	text
rank	int
category	text
book_title	text
author	text
publisher	text

使用 CQL 建立資料表

1. 使用下列命令開啟 AWS CloudShell 並連線至 Amazon Keyspaces。請務必使用您自己的區域更新 *us-east-1*。

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

該命令的輸出應如下所示。

```
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.
```

2. 在金鑰空間提示字元 (cqlsh:keyspace_name>) 中，在命令視窗中輸入下列程式碼來建立資料表。

```
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)
);
```


Note

ASC 是預設叢集順序。您也可以DESC為遞減順序指定。

請注意，year和award是分割區索引鍵資料欄。然後，category和rank是依遞增順序排序的叢集資料欄(ASC)。這些資料欄共同形成資料表的主索引鍵。

3. 確認您的資料表已建立。

```
SELECT * from system_schema.tables WHERE keyspace_name='catalog.book_awards' ;
```

輸出看起來應該與此類似。

```
keyspace_name | table_name | bloom_filter_fp_chance | caching | cdc | 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_interval |
read_repair_chance | speculative_retry
-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+-----+

(0 rows)
```

4. 驗證資料表的結構。

```
SELECT * FROM system_schema.columns WHERE keyspace_name = 'catalog' AND table_name
= 'book_awards';
```

此陳述式的輸出看起來應該與此範例類似。

```
keyspace_name | table_name | column_name | clustering_order | column_name_bytes
| kind      | position | type
-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
catalog | book_awards | year | none |
0x79656172 | partition_key | 0 | int
```

```

    catalog | book_awards |      award |      none |
0x6177617264 | partition_key |      1 | text
    catalog | book_awards |    category |      asc |
0x63617465676f7279 | clustering |      0 | text
    catalog | book_awards |      rank |      asc |
0x72616e6b | clustering |      1 | int
    catalog | book_awards |    author |      none |
0x617574686672 | regular |     -1 | text
    catalog | book_awards | book_title |      none |
0x626f666b5f7469746c65 | regular |     -1 | text
    catalog | book_awards | publisher |      none |
0x7075626c6973686572 | regular |     -1 | text

```

(7 rows)

確認所有資料欄和資料類型都如預期。資料欄的順序可能與 CREATE 陳述式中的順序不同。

使用 AWS CLI

此程序會使用 建立具有下列資料欄和資料類型的資料表 AWS CLI。year 和 award 資料欄使用 category 和 rank 做為叢集資料欄組成分割區索引鍵。

```

year          int
award         text
rank          int
category      text
book_title    text
author        text
publisher     text

```

使用 建立資料表 AWS CLI

下列命令會建立名為 book_awards 的資料表。資料表的分割區索引鍵由欄 yearaward 和 組成，叢集索引鍵由欄 category 和 組成 rank，兩個叢集欄都使用遞增排序順序。（為了更容易閱讀，本節中資料表建立命令 schema-definition 的 會分成不同的行。）

1. 您可以使用下列陳述式建立資料表。

```

aws keyspace 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-1:111122223333:/keyspace/catalog/
table/book_awards"
}
```

- 若要確認資料表的中繼資料和屬性，您可以使用下列命令。

```
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-1:111122223333:/keyspace/catalog/
table/book_awards",
  "creationTimestamp": "2024-07-11T15:12:55.571000+00:00",
  "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": "2024-07-11T15:12:55.571000+00:00"
  },
  "encryptionSpecification": {
    "type": "AWS_OWNED_KMS_KEY"
  },
  "pointInTimeRecovery": {
```

```

        "status": "DISABLED"
    },
    "defaultTimeToLive": 0,
    "comment": {
        "message": ""
    },
    "replicaSpecifications": []
}

```

若要對資料表中的資料執行 CRUD（建立、讀取、更新和刪除）操作，請繼續。[the section called “CRUD 操作”](#)

在 Amazon Keyspaces 中檢查資料表建立狀態

Amazon Keyspaces 會以非同步方式執行資料定義語言 (DDL) 操作，例如建立和刪除資料表。您可以在 [中](#) 監控新資料表的建立狀態 AWS 管理主控台，這表示資料表何時處於待定或作用中狀態。您也可以使用系統結構描述資料表，以程式設計方式監控新資料表的建立狀態。

準備好使用時，資料表會在系統結構描述中顯示為作用中。檢查新資料表何時可供使用的建議設計模式是輪詢 Amazon Keyspaces 系統結構描述資料表 (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 “表格”](#)一節。

下列查詢顯示資料表的狀態。

```

SELECT keyspace_name, table_name, status FROM system_schema_mcs.tables WHERE
keyspace_name = 'mykeyspace' AND table_name = 'mytable';

```

對於仍在建立且待定的資料表，查詢的輸出如下所示。

```

keyspace_name | table_name | status
-----+-----+-----
mykeyspace   | mytable   | CREATING

```

對於已成功建立且作用中的資料表，查詢的輸出如下所示。

```

keyspace_name | table_name | status
-----+-----+-----
mykeyspace   | mytable   | ACTIVE

```

在 Amazon Keyspaces 中使用 CQL 建立、讀取、更新和刪除資料 (CRUD)

在教學課程的此步驟中，您將了解如何使用 CQL 資料處理語言 (DML) 陳述式在 Amazon Keyspaces 資料表中插入、讀取、更新和刪除資料。在 Amazon Keyspaces 中，您只能以 CQL 語言建立 DML 陳述式。在本教學課程中，您將練習在 [AWS CloudShell](#) 中使用 `cqlsh-expansion` 搭配執行 DML 陳述式 AWS 管理主控台。

- 插入資料 – 本節涵蓋使用 INSERT 陳述式將單一和多個記錄插入資料表。您將了解如何從 CSV 檔案上傳資料，並使用 SELECT 查詢驗證成功插入。
- 讀取資料 – 在這裡，您將探索 SELECT 陳述式的不同變化，以從資料表擷取資料。主題包括選取所有資料、選取特定資料欄、使用 WHERE 子句根據條件篩選資料列，以及了解簡單和複合條件。
- 更新資料 – 在本節中，您將了解如何使用 UPDATE 陳述式修改資料表中的現有資料。您將練習更新單一資料欄和多個資料欄，同時了解更新主索引鍵資料欄的限制。
- 刪除資料 – 最後一節涵蓋使用 DELETE 陳述式從資料表刪除資料。您將了解如何刪除特定儲存格、整個資料列，以及刪除資料相對於刪除整個資料表或金鑰空間的影響。

在整個教學課程中，您會找到範例、秘訣和機會，練習為各種案例撰寫自己的 CQL 查詢。

主題

- [將資料插入和載入 Amazon Keyspaces 資料表](#)
- [使用 Amazon Keyspaces 中的 CQL SELECT 陳述式從資料表讀取資料](#)
- [使用 CQL 更新 Amazon Keyspaces 資料表中的資料](#)
- [使用 CQL DELETE 陳述式從資料表刪除資料](#)

將資料插入和載入 Amazon Keyspaces 資料表

若要在 `book_awards` 資料表中建立資料，請使用 INSERT 陳述式來新增單一資料列。

1. 使用下列命令開啟 AWS CloudShell 並連線至 Amazon Keyspaces。請務必使用您自己的區域更新 `us-east-1`。

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

該命令的輸出應如下所示。

```
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.
```

2. 您必須先將目前 cqlsh 工作階段的寫入一致性設定為 `LOCAL_QUORUM`，才能使用 cqlsh 將資料寫入 Amazon Keyspaces 資料表 `LOCAL_QUORUM`。如需支援的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 `CQL 編輯器` 中使用 CQL 編輯器，則不需要此步驟 AWS 管理主控台。

```
CONSISTENCY LOCAL_QUORUM;
```

3. 若要插入單一記錄，請在 CQL 編輯器中執行下列命令。

```
INSERT INTO catalog.book_awards (award, year, category, rank, author, book_title,
publisher)
VALUES ('Wolf', 2023, 'Fiction', 3, 'Shirley Rodriguez', 'Mountain', 'AnyPublisher');
```

4. 執行下列命令，確認資料已正確新增至資料表。

```
SELECT * FROM catalog.book_awards;
```

陳述式的輸出應該如下所示。

```
year | award | category | rank | author          | book_title | publisher
-----+-----+-----+-----+-----+-----+-----
2023 | Wolf  | Fiction  | 3    | Shirley Rodriguez | Mountain   | AnyPublisher

(1 rows)
```

使用 cqlsh 從檔案插入多筆記錄

1. 下載封存檔案 [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`。

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 Chateau	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 Ice	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

- 在 AWS CloudShell 中開啟 AWS 管理主控台，並使用下列命令連線至 Amazon Keyspaces。請務必使用您自己的區域更新 `us-east-1`。

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

- 在 cqlsh 提示 (cqlsh>) 中，指定金鑰空間。

```
USE catalog ;
```

- 將寫入一致性設定為 LOCAL_QUORUM。如需支援的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;
```

- 在畫面右上角 AWS CloudShell 選擇動作，然後選擇上傳檔案以上傳先前下載的 csv 檔案。請記下檔案的路徑。
- 在金鑰空間提示 (cqlsh:catalog>) 中，執行下列陳述式。

```
COPY book_awards (award, year, category, rank, author, book_title, publisher) FROM
'/home/cloudshell-user/keyspaces_sample_table.csv' WITH header=TRUE ;
```

陳述式的輸出看起來應該與此類似。

```
cqlsh:catalog> COPY book_awards (award, year, category, rank, author,
book_title, publisher) FROM '/home/cloudshell-user/
keyspaces_sample_table.csv' WITH delimiter=',' AND header=TRUE ;
cqlsh current consistency level is LOCAL_QUORUM.
Reading options from /home/cloudshell-user/.cassandra/cqlshrc:[copy]:
{'numprocesses': '16', 'maxattempts': '1000'}
```



```

Reading options from /home/cloudshell-user/.cassandra/cqlshrc:[copy-from]:
{'ingestrate': '1500', 'maxparseerrors': '1000', 'maxinserterrors': '-1',
 'maxbatchsize': '10', 'minbatchsize': '1', 'chunksize': '30'}
Reading options from the command line: {'delimiter': ',', 'header': 'TRUE'}
Using 16 child processes

Starting copy of catalog.book_awards with columns [award, year, category, rank,
author, book_title, publisher].
OSError: handle is closed      0 rows/s; Avg. rate:      0 rows/s
Processed: 9 rows; Rate:      0 rows/s; Avg. rate:      0 rows/s
9 rows imported from 1 files in 0 day, 0 hour, 0 minute, and 26.706 seconds (0
skipped).

```

7. 執行下列查詢，確認資料已正確新增至資料表。

```
SELECT * FROM book_awards ;
```

您應該會看到下列輸出。

year	award publisher	category	rank	author	book_title
2020	Wolf Example Books	Non-Fiction	1	Wang Xiulan	History of Ideas
2020	Wolf SomePublisher	Non-Fiction	2	Ana Carolina Silva	Science Today
2020	Wolf AnyPublisher	Non-Fiction	3	Shirley Rodriguez	The Future of Sea Ice
2020	Kwesi Manu Prize SomePublisher	Fiction	1	Akua Mansa	Where did you go?
2020	Kwesi Manu Prize Example Books	Fiction	2	John Stiles	Yesterday
2020	Kwesi Manu Prize AnyPublisher	Fiction	3	Nikki Wolf	Moving to the Chateau
2020	Richard Roe SomePublisher	Fiction	1	Alejandro Rosalez	Long Summer
2020	Richard Roe Example Books	Fiction	2	Arnav Desai	The Key
2020	Richard Roe AnyPublisher	Fiction	3	Mateo Jackson	Inside the Whale

(9 rows)

若要進一步了解如何使用 `cqlsh COPY` 將資料從 `csv` 檔案上傳至 Amazon Keyspaces 資料表，請參閱 [the section called “使用 cqlsh 載入資料”](#)。

使用 Amazon Keyspaces 中的 CQL **SELECT** 陳述式從資料表讀取資料

在 [將資料插入和載入 Amazon Keyspaces 資料表](#) 區段中，您使用 `SELECT` 陳述式來驗證您已成功將資料新增至資料表。在本節中，您可以精簡使用 `SELECT` 以顯示特定資料欄，以及僅顯示符合特定條件的資料列。

`SELECT` 陳述式的一般形式如下。

```
SELECT column_list FROM table_name [WHERE condition [ALLOW FILTERING]] ;
```

主題

- [選取資料表中的所有資料](#)
- [選取欄的子集](#)
- [選取資料列的子集](#)

選取資料表中的所有資料

最簡單的 `SELECT` 陳述式形式會傳回資料表中的所有資料。

Important

在生產環境中，執行此命令通常不是最佳實務，因為它會傳回資料表中的所有資料。

若要選取資料表的所有資料

1. 使用下列命令開啟 AWS CloudShell 並連線至 Amazon Keyspaces。請務必使用您自己的區域更新 `us-east-1`。

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

2. 執行下列查詢。

```
SELECT * FROM catalog.book_awards ;
```

針對 使用萬用字元 * () 會column_list選取所有資料欄。陳述式的輸出如下所示。

year	award publisher	category	rank	author	book_title
2020	Wolf AnyPublisher	Non-Fiction	1	Wang Xiulan	History of Ideas
2020	Wolf SomePublisher	Non-Fiction	2	Ana Carolina Silva	Science Today
2020	Wolf AnyPublisher	Non-Fiction	3	Shirley Rodriguez	The Future of Sea Ice
2020	Kwesi Manu Prize SomePublisher	Fiction	1	Akua Mansa	Where did you go?
2020	Kwesi Manu Prize Example Books	Fiction	2	John Stiles	Yesterday
2020	Kwesi Manu Prize AnyPublisher	Fiction	3	Nikki Wolf	Moving to the Chateau
2020	Richard Roe SomePublisher	Fiction	1	Alejandro Rosalez	Long Summer
2020	Richard Roe Example Books	Fiction	2	Arnav Desai	The Key
2020	Richard Roe AnyPublisher	Fiction	3	Mateo Jackson	Inside the Whale

選取欄的子集

查詢欄的子集

1. 使用下列命令開啟 AWS CloudShell 並連線至 Amazon Keyspaces。請務必使用您自己的區域更新 *us-east-1*。

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

2. 若要僅擷取 award、category 和 year 資料欄，請執行下列查詢。

```
SELECT award, category, year FROM catalog.book_awards ;
```

輸出僅包含SELECT依陳述式中所列順序列出的指定資料欄。

award	category	year
Wolf	Non-Fiction	2020
Wolf	Non-Fiction	2020
Wolf	Non-Fiction	2020
Kwesi Manu Prize	Fiction	2020
Kwesi Manu Prize	Fiction	2020
Kwesi Manu Prize	Fiction	2020
Richard Roe	Fiction	2020
Richard Roe	Fiction	2020
Richard Roe	Fiction	2020

選取資料列的子集

查詢大型資料集時，您可能只想要符合特定條件的記錄。若要這樣做，您可以將 WHERE 子句附加到SELECT陳述式的結尾。

查詢資料列子集

1. 使用下列命令開啟 AWS CloudShell 並連線至 Amazon Keyspaces。請務必使用您自己的區域更新 *us-east-1*。

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

2. 若要僅擷取指定年份獎勵的記錄，請執行下列查詢。

```
SELECT * FROM catalog.book_awards WHERE year=2020 AND award='Wolf' ;
```

上述SELECT陳述式會傳回下列輸出。

year	award	category	rank	author	book_title	publisher
2020	Wolf	Non-Fiction	1	Wang Xiulan	History of Ideas	AnyPublisher
2020	Wolf	Non-Fiction	2	Ana Carolina Silva	Science Today	SomePublisher

2020 | Wolf | Non-Fiction | 3 | Shirley Rodriguez | The Future of Sea Ice | AnyPublisher

了解 **WHERE** 子句

WHERE 子句用於篩選資料，並僅傳回符合指定條件的資料。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

- 簡單條件 – 單一資料欄。

```
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：

- 資料欄是資料表的唯一分割區索引鍵資料欄。
- 您可以在 **WHERE** 子句中的條件 **ALLOW FILTERING** 之後新增。

請注意，使用 **ALLOW FILTERING** 可能會導致效能不一致，特別是對於大型和多分割區資料表。

- 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...
```

如果符合下列任一條件，您可以在 **WHERE** 子句中使用複合條件：

- 您可以在 **WHERE** 子句中使用的資料欄需要包含資料表分割區索引鍵中所有或一部分的資料欄。如果只想在 **WHERE** 子句中使用資料欄的子集，您必須從左到右包含一組連續的分割區索引鍵資料欄，從分割區索引鍵的前導資料欄開始。例如，如果分割區索引鍵資料欄是 `year`、`month` 和 `award`，則您可以在 **WHERE** 子句中使用下列資料欄：
 - `year`
 - `year` 和 `month`
 - `year AND month` 和 `award`
- 您可以在 **WHERE** 子句中的複合條件 **ALLOW FILTERING** 之後新增，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，特別是對於大型和多分割區資料表。

試試看

建立您自己的 CQL 查詢，從book_awards資料表尋找下列項目：

- 尋找 2020 Wolf 獎勵的得主，並顯示依排名排序的書名和作者。
- 顯示 2020 年所有獎勵的第一個得主，並顯示書籍標題和獎勵名稱。

使用 CQL 更新 Amazon Keyspaces 資料表中的資料

若要更新book_awards資料表中的資料，請使用 UPDATE陳述式。

UPDATE 陳述式的一般形式如下。

```
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 col3='1' ;
```

- 如果主索引鍵是由多個資料欄組成，則所有主索引鍵資料欄及其值都必須包含在 WHERE子句中。
- 您無法更新主索引鍵中的任何資料欄，因為這會變更記錄的主索引鍵。

更新單一儲存格

使用您的book_awards資料表，為 2020 年非小說 Wolf 獎勵的得主變更發佈者的名稱。

```
UPDATE book_awards SET publisher='new Books' WHERE year = 2020 AND award='Wolf' AND category='Non-Fiction' AND rank=1;
```

確認發佈者現在是 new Books。

```
SELECT * FROM book_awards WHERE year = 2020 AND award='Wolf' AND category='Non-Fiction' AND rank=1;
```

陳述式應傳回下列輸出。

year	award	category	rank	author	book_title	publisher
2020	Wolf	Non-Fiction	1	Wang Xiulan	History of Ideas	new Books

試試看

進階：2020 年虛構「Kwezi Manu Prize」的獲勝者已變更其名稱。更新此記錄，將名稱變更為 'Akua Mansa-House'。

使用 CQL DELETE 陳述式從資料表刪除資料

若要刪除 book_awards 資料表中的資料，請使用 DELETE 陳述式。

您可以從資料列或分割區刪除資料。刪除資料時請小心，因為刪除是不可復原的。

從資料表刪除一行或所有資料列並不會刪除資料表。因此，您可以將資料重新填入其中。刪除資料表會刪除資料表及其中的所有資料。若要再次使用資料表，您必須重新建立資料表並將其新增資料。刪除金鑰空間會刪除金鑰空間及其中的所有資料表。若要使用金鑰空間和資料表，您必須重新建立它們，然後將資料填入它們。您可以使用 Amazon Keyspaces Point-in-time(PITR) 復原來協助還原已刪除的資料表，進一步了解 [the section called “使用 point-in-time 復原進行備份和還原”](#)。若要了解如何還原已啟用 PITR 的已刪除資料表，請參閱 [the section called “還原已刪除的資料表”](#)。

刪除儲存格

從資料列刪除資料欄會從指定的儲存格移除資料。當您使用 SELECT 陳述式顯示該資料欄時，資料會顯示為 *null*，但 null 值不會存放在該位置。

刪除一或多個特定資料欄的一般語法如下所示。

```
DELETE column_name1[, column_name2...] FROM table_name WHERE condition ;
```

在您的 book_awards 資料表中，您可以看到 2020 年「Richard Roe」價格中第一個價格的書名是「Long Summer」。假設此標題已召回，您需要從此儲存格刪除資料。

刪除特定儲存格

1. 使用下列命令開啟 AWS CloudShell 並連線至 Amazon Keyspaces。請務必使用您自己的區域更新 *us-east-1*。

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

2. 執行下列DELETE查詢。

```
DELETE book_title FROM catalog.book_awards WHERE year=2020 AND award='Richard Roe'
AND category='Fiction' AND rank=1;
```

3. 確認已如預期提出刪除請求。

```
SELECT * FROM catalog.book_awards WHERE year=2020 AND award='Richard Roe' AND
category='Fiction' AND rank=1;
```

此陳述式的輸出如下所示。

year	award	category	rank	author	book_title	publisher
2020	Richard Roe	Fiction	1	Alejandro Rosalez	null	SomePublisher

刪除資料列

有時候您可能需要刪除整個資料列，例如符合資料刪除請求。刪除資料列的一般語法如下。

```
DELETE FROM table_name WHERE condition ;
```

刪除資料列

1. 使用下列命令開啟 AWS CloudShell 並連線至 Amazon Keyspaces。請務必使用您自己的區域更新 *us-east-1*。

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

2. 執行下列DELETE查詢。

```
DELETE FROM catalog.book_awards WHERE year=2020 AND award='Richard Roe' AND
category='Fiction' AND rank=1;
```

3. 確認已如預期進行刪除。


```
SELECT * FROM catalog.book_awards WHERE year=2020 AND award='Richard Roe' AND
category='Fiction' AND rank=1;
```

刪除資料列後，此陳述式的輸出看起來像這樣。

```
year | award | category | rank | author | book_title | publisher
-----+-----+-----+-----+-----+-----+-----
(0 rows)
```

您可以使用 Amazon Keyspaces 存留時間自動刪除資料表中的過期資料，如需詳細資訊，請參閱 [the section called “使用存留時間過期資料”](#)。

在 Amazon Keyspaces 中刪除資料表

為了避免對您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。當您刪除資料表時，資料表及其資料會遭到刪除，而且您會停止產生這些資料表的費用。不過，金鑰空間仍會保留。當您刪除金鑰空間時，金鑰空間及其所有資料表都會遭到刪除，並停止產生費用。

您可以使用 主控台、CQL 或 刪除資料表 AWS CLI。當您刪除資料表時，資料表及其所有資料都會遭到刪除。

使用主控台

下列程序會使用 刪除資料表及其所有資料 AWS 管理主控台。

使用主控台刪除資料表

1. 登入 AWS 管理主控台，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)。
3. 選擇您要刪除之每個資料表名稱左側的方塊。
4. 選擇 刪除。
5. 在刪除資料表畫面上，**Delete**在方塊中輸入。然後，選擇刪除資料表。
6. 若要驗證資料表是否已刪除，請在導覽窗格中選擇資料表，並確認book_awards資料表不再列出。

使用 CQL

下列程序會使用 CQL 刪除資料表及其所有資料。

使用 CQL 刪除資料表

1. 使用下列命令開啟 AWS CloudShell 並連線至 Amazon Keyspaces。請務必使用您自己的區域更新 *us-east-1*。

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

2. 輸入下列陳述式來刪除資料表。

```
DROP TABLE IF EXISTS catalog.book_awards ;
```

3. 確認您的資料表已刪除。

```
SELECT * FROM system_schema.tables WHERE keyspace_name = 'catalog' ;
```

輸出看起來應該像這樣。請注意，這可能需要一些時間，因此如果您沒有看到此結果，請在一分鐘後重新執行陳述式。

```
keyspace_name | table_name | bloom_filter_fp_chance | caching | cdc | 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_interval |
read_repair_chance | speculative_retry
```

```
-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+
```

```
(0 rows)
```

使用 AWS CLI

下列程序會使用 刪除資料表及其所有資料 AWS CLI。

使用 刪除資料表 AWS CLI

1. 開啟 CloudShell
2. 使用下列陳述式刪除資料表。

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

3. 若要確認您的資料表已刪除，您可以列出金鑰空間中的所有資料表。

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

您應該會看到下列輸出。請注意，此非同步操作可能需要一些時間。稍後再次重新執行命令，以確認資料表已刪除。

```
{
  "tables": []
}
```

在 Amazon Keyspaces 中刪除金鑰空間

若要避免收取金鑰空間的費用，請刪除您未使用的所有金鑰空間。當您刪除金鑰空間時，金鑰空間及其所有資料表都會遭到刪除，並停止產生費用。

您可以使用 主控台、CQL 或 刪除金鑰空間 AWS CLI。

使用主控台

下列程序會使用主控台刪除金鑰空間及其所有資料表和資料。

使用主控台刪除金鑰空間

1. 登入 AWS 管理主控台，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 金鑰空間。
3. 選擇您要刪除之每個金鑰空間名稱左側的方塊。
4. 選擇 刪除。
5. 在刪除鍵空間畫面上，**Delete**在方塊中輸入。然後，選擇刪除金鑰空間。

- 若要確認金鑰空間catalog已刪除，請在導覽窗格中選擇金鑰空間，並確認它不再列出。由於您已刪除其金鑰空間，book_awards資料表下的資料表也不應列出。

使用 CQL

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 刪除金鑰空間

- 使用下列命令開啟 AWS CloudShell 並連線至 Amazon Keyspaces。請務必使用您自己的區域更新 *us-east-1*。

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

- 輸入下列陳述式來刪除您的金鑰空間。

```
DROP KEYSPACE IF EXISTS catalog ;
```

- 確認您的 金鑰空間已刪除。

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

不應列出您的 金鑰空間。請注意，由於這是非同步操作，因此在刪除金鑰空間之前，可能會有延遲。刪除金鑰空間後，陳述式的輸出應如下所示。

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

(4 rows)

使用 AWS CLI

下列程序會使用 刪除金鑰空間及其所有資料表和資料 AWS CLI。

使用 刪除金鑰空間 AWS CLI

1. 開啟 AWS CloudShell
2. 輸入下列陳述式來刪除您的金鑰空間。

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

3. 確認您的 金鑰空間已刪除。

```
aws keyspaces list-keyspaces
```

此陳述式的輸出看起來應該與此類似，並且只列出系統金鑰空間。請注意，由於這是非同步操作，因此在刪除您的金鑰空間之前，可能會有延遲。

```
{
  "keyspaces": [
    {
      "keyspaceName": "system_schema",
      "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system_schema/",
      "replicationStrategy": "SINGLE_REGION"
    },
    {
      "keyspaceName": "system_schema_mcs",
      "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system_schema_mcs/",
      "replicationStrategy": "SINGLE_REGION"
    },
    {
      "keyspaceName": "system",
      "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system/",
      "replicationStrategy": "SINGLE_REGION"
    },
    {
      "keyspaceName": "system_multiregion_info",
      "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system_multiregion_info/",

```

```
        "replicationStrategy": "SINGLE_REGION"  
    }  
]  
}
```

Amazon Keyspaces 的教學課程和解決方案

本主題包含各種教學課程，逐步解說以程式設計方式連線至 Amazon Keyspaces，或搭配其他服務使用 Amazon Keyspaces AWS 來建立解決方案所需的詳細步驟。可用的教學課程涵蓋不同的熱門跨服務存取案例，並示範如何將 Amazon Keyspaces 與 Amazon Virtual Private Cloud、Amazon Elastic Kubernetes Service、Amazon Simple Storage Service 和 AWS Glue 整合，或與 Apache Spark 等開放原始碼技術整合。

如需示範如何使用 Apache Cassandra 驅動程式連線至 Amazon Keyspaces 的 step-by-step 教學課程，請參閱 [the section called “使用 Cassandra 用戶端驅動程式”](#)。

主題

- [教學課程：使用介面 VPC 端點連線至 Amazon Keyspaces](#)
- [教學課程：與 Apache Spark 整合以匯入或匯出資料](#)
- [教學課程：從託管在 Amazon Elastic Kubernetes Service 上的容器化應用程式連線](#)
- [教學課程：使用 將 Amazon Keyspaces 資料表匯出至 Amazon S3 AWS Glue](#)

教學課程：使用介面 VPC 端點連線至 Amazon Keyspaces

本教學課程會逐步引導您設定和使用 Amazon Keyspaces 的介面 VPC 端點。

介面 VPC 端點可在 Amazon VPC 和 Amazon Keyspaces 中執行的虛擬私有雲端 (VPC) 之間啟用私有通訊。介面 VPC 端點採用技術 AWS PrivateLink，這項 AWS 服務可啟用 VPCs 和 AWS 服務之間的私有通訊。如需詳細資訊，請參閱 [the section called “使用 介面 VPC 端點”](#)。

主題

- [教學的先決條件和考量事項](#)
- [步驟 1：啟動 Amazon EC2 執行個體](#)
- [步驟 2：設定您的 Amazon EC2 執行個體](#)
- [步驟 3：建立 Amazon Keyspaces 的 VPC 端點](#)
- [步驟 4：設定 VPC 端點連線的許可](#)
- [步驟 5：使用 CloudWatch 設定監控](#)
- [步驟 6：\(選用\) 為應用程式設定連線集區大小的最佳實務](#)
- [步驟 7：\(選用\) 清除](#)

教學的先決條件和考量事項

開始本教學課程之前，請遵循中的 AWS 設定說明[存取 Amazon Keyspaces \(適用於 Apache Cassandra\)](#)。這些步驟包括註冊 AWS 和建立可存取 Amazon 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，因為您稍後在本教學課程中會需要它們。

在本教學稍後使用 VPC 端點建立具有名稱myKeyspace和至少一個資料表的金鑰空間，以測試連線。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 [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，您可以建立和使用 Amazon Keyspaces 的 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，選擇啟動執行個體，然後從出現的選項中選擇啟動執行個體。

在名稱和標籤下，針對名稱輸入執行個體的描述性名稱。

在應用程式和作業系統映像 (Amazon Machine Image) 下：

- 選擇 Quick Start，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 Machine Image (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。

在執行個體類型下：

- 從執行個體類型清單中，選擇預設選取的 t2.micro 執行個體類型。

在金鑰對（登入）下，針對金鑰對名稱，選擇本教學課程的下列其中一個選項：

- 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有金鑰檔案 (.pem 檔案)。稍後當您登入 Amazon EC2 執行個體時，您將需要此檔案，因此請注意檔案路徑。
- 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組態的摘要。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，選擇連線至執行個體圖磚。下一個畫面顯示連接到新執行個體的必要資訊和必要步驟。請注意下列資訊：
- 保護金鑰檔案的範例命令
 - 連線字串
 - 公有 IPv4 DNS 名稱

記下此頁面上的資訊後，您可以繼續本教學課程的下一個步驟 ([步驟 2：設定您的 Amazon EC2 執行個體](#))。

 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，您可以登入並準備第一次使用。

Note

下列步驟假設您從執行 Linux 的電腦連線至 Amazon EC2 執行個體。如需其他連線方式，請參閱《Amazon EC2 使用者指南》中的[連線至 Linux 執行個體](#)。

設定 Amazon 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。若要這樣做，請建立新的 EC2 安全群組，然後將安全群組指派給您的 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：
 - 安全群組名稱 – 輸入安全群組的名稱。例如：my-ssh-access
 - 描述 – 輸入安全群組的簡短描述。
 - VPC – 選擇您的預設 VPC。
 - 在傳入規則區段中，選擇新增規則並執行下列動作：
 - 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，然後選擇變更安全群組。
 - f. 在變更安全群組中，前往關聯的安全群組，然後輸入您在此程序中稍早建立的安全群組（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，然後選擇儲存。

2. ~~使用以下命令保護您的私有金鑰檔案免於存取。如果您略過此步驟，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
```

您需要指定私有金鑰檔案 (.pem 檔案) 和執行個體的公有 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，以及相關 AWS CLI 指示，請參閱《Amazon EC2 使用者指南》中的[授權 Linux 執行個體的傳入流量](#)。

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 登入資料，如下列範例所示。出現提示時，輸入您的 AWS 存取金鑰 ID、私密金鑰和預設區域名稱。

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. 將規則新增至您的 VPCs 安全群組，以允許來自 IPv6 地址的傳入 HTTP、HTTPS 和 SSH 存取。
7. 若要確認您的 VPC 端點已正確設定，您必須使用 Amazon Keyspaces 的 `cqlsh` 連線。如果您使用本機環境或中的 Amazon Keyspaces CQL 編輯器 AWS 管理主控台，連線會自動通過公有端點，而不是您的 VPC 端點。若要在本教學課程中使用 `cqlsh` 測試您的 VPC 端點連線，請完成中的設定說明[使用 cqlsh 連線至 Amazon Keyspaces](#)。

您現在可以為 Amazon Keyspaces 建立雙堆疊 VPC 端點。

步驟 3：建立 Amazon Keyspaces 的 VPC 端點

在此步驟中，您會使用為 Amazon Keyspaces 建立雙堆疊 VPC 端點 AWS CLI。若要使用 VPC 主控台建立 VPC 端點，您可以遵循 AWS PrivateLink 指南中的[建立 VPC 端點](#)說明。篩選服務名稱時，請輸入 **Cassandra**。

使用 建立 VPC 端點 AWS CLI

1. 開始之前，請確認您可以使用其公有端點與 Amazon Keyspaces 通訊。

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

輸出會顯示包含在指定金鑰空間中的 Amazon Keyspaces 資料表清單。如果您沒有任何資料表，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"
    "api.aws.us-east-1.cassandra-streams"
  ]
}

```

如果 Amazon Keyspaces 是命令輸出中可用的服務之一，您可以繼續建立 VPC 端點。

3. 若要使用啟用 IPv6 的雙堆疊端點連線至 Amazon Keyspaces，請確認您的 VPC 支援 IPv6，並使用 IPv6 支援設定子網路。若要將 IPv6 支援新增至目前僅支援 IPv4 的現有 VPC，請參閱《Amazon VPC 使用者指南》中的 [VPC 的 IPv6 支援](#)。
4. 確定您的 VPC 識別碼。

```
aws ec2 describe-vpcs
```

```

{
  "Vpcs": [
    {
      "OwnerId": "111122223333",
      "InstanceTenancy": "default",
      "Ipv6CidrBlockAssociationSet": [

```

```

        {
            "AssociationId": "vpc-cidr-assoc-0000aaa0a00a00aa0",
            "Ipv6CidrBlock": "2600:1f18:e19:7d00::/56",
            "Ipv6CidrBlockState": {
                "State": "associated"
            },
            "NetworkBorderGroup": "us-east-1",
            "Ipv6Pool": "Amazon",
            "Ipv6AddressAttribute": "public",
            "IpSource": "amazon"
        },
        {
            "AssociationId": "vpc-cidr-assoc-00a0000a",
            "CidrBlock": "111.11.0.0/16",
            "CidrBlockState": {
                "State": "associated"
            }
        },
        {
            "IsDefault": true,
            "BlockPublicAccessStates": {
                "InternetGatewayBlockMode": "off"
            },
            "VpcId": "vpc-a1234bcd",
            "State": "available",
            "CidrBlock": "111.11.0.0/16",
            "DhcpOptionsId": "dopt-a00aaaaa"
        }
    ]
}

```

在範例輸出中，VPC ID 為 vpc-a1234bcd。

5. 使用篩選條件來收集 VPC 子網路的詳細資訊。

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

```

{
    "Subnets": [
        {
            "AvailabilityZoneId": "use1-az1",
            "MapCustomerOwnedIpOnLaunch": false,

```

```

      "OwnerId": "111122223333",
      "AssignIpv6AddressOnCreation": false,
      "Ipv6CidrBlockAssociationSet": [
        {
          "AssociationId": "subnet-cidr-assoc-05d75732736740283",
          "Ipv6CidrBlock": "*****",
          "Ipv6CidrBlockState": {
            "State": "associated"
          },
          "Ipv6AddressAttribute": "public",
          "IpSource": "amazon"
        }
      ],
      "SubnetArn": "arn:aws:ec2:us-east-1:111122223333:subnet/
subnet-70b24b16",
      "EnableDns64": false,
      "Ipv6Native": false,
      "PrivateDnsNameOptionsOnLaunch": {
        "HostnameType": "ip-name",
        "EnableResourceNameDnsARecord": false,
        "EnableResourceNameDnsAAAARecord": false
      },
      "BlockPublicAccessStates": {
        "InternetGatewayBlockMode": "off"
      },
      "SubnetId": "subnet-70b24b16",
      "State": "available",
      "VpcId": "vpc-a1234bcd",
      "CidrBlock": "*****/20",
      "AvailableIpAddressCount": 4096,
      "AvailabilityZone": "us-east-1a",
      "DefaultForAz": true,
      "MapPublicIpOnLaunch": true
    },
    {
      "AvailabilityZoneId": "use1-az2",
      "MapCustomerOwnedIpOnLaunch": false,
      "OwnerId": "111122223333",
      "AssignIpv6AddressOnCreation": false,
      "Ipv6CidrBlockAssociationSet": [
        {
          "AssociationId": "subnet-cidr-assoc-0ec6fb253e05b17eb",
          "Ipv6CidrBlock": "*****",
          "Ipv6CidrBlockState": {

```

```

        "State": "associated"
      },
      "Ipv6AddressAttribute": "public",
      "IpSource": "amazon"
    }
  ],
  "SubnetArn": "arn:aws:ec2:us-east-1:111122223333:subnet/subnet-
c63ffbe7",
  "EnableDns64": false,
  "Ipv6Native": false,
  "PrivateDnsNameOptionsOnLaunch": {
    "HostnameType": "ip-name",
    "EnableResourceNameDnsARecord": false,
    "EnableResourceNameDnsAAAARecord": false
  },
  "BlockPublicAccessStates": {
    "InternetGatewayBlockMode": "off"
  },
  "SubnetId": "subnet-c63ffbe7",
  "State": "available",
  "VpcId": "vpc-a1234bcd",
  "CidrBlock": "*/20",
  "AvailableIpAddressCount": 4087,
  "AvailabilityZone": "us-east-1b",
  "DefaultForAz": true,
  "MapPublicIpOnLaunch": true
}
]
}

```

在範例輸出中，有兩個可用的子網路 IDsubnet-70b24b16和 subnet-c63ffbe7。

6. 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-ids` 參數，指定上一個步驟 IDs。使用 `--vpc-endpoint-type` 參數將端點定義為界面。若要建立雙堆疊端點，請使用 `--ip-address-type dualstack`。如需 命令的詳細資訊，請參閱《AWS CLI 命令參考[create-vpc-endpoint](#)》中的。

```

aws ec2 create-vpc-endpoint \
  --vpc-endpoint-type Interface \
  --vpc-id vpc-a1234bcd \
  --ip-address-type dualstack \
  --service-name com.amazonaws.us-east-1.cassandra \
  --subnet-ids subnet-70b24b16 subnet-c63ffbe7

```



```

{
  "VpcEndpoint": {
    "VpcEndpointId": "vpce-0000000abc111d2ef3",
    "VpcEndpointType": "Interface",
    "VpcId": "vpc-a1234bcd",
    "ServiceName": "com.amazonaws.us-east-1.cassandra",
    "State": "pending",
    "RouteTableIds": [],
    "SubnetIds": [
      "subnet-70b24b16",
      "subnet-c63ffbe7"
    ],
    "Groups": [
      {
        "GroupId": "sg-0123456789",
        "GroupName": "default"
      }
    ],
    "IpAddressType": "dualstack",
    "DnsOptions": {
      "DnsRecordIpType": "dualstack"
    },
    "PrivateDnsEnabled": true,
    "RequesterManaged": false,
    "NetworkInterfaceIds": [
      "eni-08cd525f72ea6f1fa",
      "eni-07b1f6c895169d8fb"
    ],
    "DnsEntries": [
      {
        "DnsName": "vpce-0000000000-1234567.cassandra.us-
east-1.vpce.amazonaws.com",
        "HostedZoneId": "Z7HUB22UULQXV"
      },
      {
        "DnsName": "vpce-0000000000-1234567-us-east-1a.cassandra.us-
east-1.vpce.amazonaws.com",
        "HostedZoneId": "Z7HUB22UULQXV"
      },
      {
        "DnsName": "cassandra.us-east-1.amazonaws.com",
        "HostedZoneId": "ZONEIDPENDING"
      }
    ]
  }
}

```

```
{
  "DnsName": "cassandra.us-east-1.api.aws",
  "HostedZoneId": "ZONEIDPENDING"
},
"CreationTimestamp": "2025-09-19T15:19:19.266000+00:00",
"OwnerId": "111122223333",
"ServiceRegion": "us-east-1"
}
```

步驟 4：設定 VPC 端點連線的許可

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

設定新端點的傳入規則，以允許 TCP 傳入流量

1. 在 Amazon VPC 主控台的左側面板中，選擇端點，然後選擇您在先前步驟中建立的端點。
2. 選擇管理安全群組，然後選擇與此端點相關聯的安全群組。
3. 選擇傳入規則，然後選擇編輯傳入規則。
4. 新增類型為 CQLSH / CASSANDRA 的傳入規則。這會將連接埠範圍自動設定為 9142。
5. 若要儲存新的傳入規則，請選擇儲存規則。

設定 IAM 使用者許可

1. 確認用於連線至 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 AmazonKeyspacesReadOnlyAccess 授予 IAM 使用者對 Amazon Keyspaces 的讀取存取權。
 - a. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。
 - 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. 從政策清單中，選擇 AmazonKeyspacesReadOnlyAccess，然後選擇下一步：檢閱。
 - 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'
```

如果需要，您可以為 Amazon Keyspaces 嘗試其他一些 AWS CLI 命令。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資料表的讀取許可，如下列政策所示。如需政策型許可的詳細資訊，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-1:111122223333:/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。

當您搭配 VPC 端點使用 Amazon Keyspaces 時，您需要將存取 Amazon Keyspaces 唯讀許可的 IAM 使用者或角色授予 Amazon EC2 執行個體和 VPC，以收集端點和網路介面資料。Amazon Keyspaces 會將此資訊存放在 `system.peers` 資料表中，並使用它來管理連線。

Note

受管政

策 AmazonKeyspacesReadOnlyAccess_v2 並 AmazonKeyspacesFullAccess 包含必

要許可，讓 Amazon Keyspaces 存取 Amazon EC2 執行個體，以讀取可用介面 VPC 端點的相關資訊。

- a. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/iam/> 開啟 IAM 主控台。
- b. 在 IAM 主控台儀表板上，選擇政策。
- c. 選擇建立政策，然後選擇 JSON 標籤。
- d. 複製下列政策，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：檢閱，輸入keyspacesVPCendpoint政策的名稱，然後選擇建立政策。
 - 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. 從政策清單中，選擇 keyspacesVPCendpoint，然後選擇下一步：檢閱。
 - j. 選擇新增許可。
4. 若要驗證 Amazon Keyspaces system.peers 資料表是否使用 VPC 資訊更新，請使用 從您的 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，請遵循 中的指示[the section called “使用 cqlsh-expansion”](#)。

```
SELECT * FROM system.peers;
```

輸出會傳回具有私有 IPv6 IP 地址的節點，視您區域中的 VPC 和子網路設定而定 AWS。

```

peer                                | data_center | host_id
      | preferred_ip                                | rack      | release_version
| rpc_address                                | schema_version |
tokens
-----+-----
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+-----+-----+
2600:1111:2222:3333:283b:8e6:d04f | us-east-1 | dddddd-7a22-3582-
a73d-49338a686a53 | 2600:1111:2222:3333:283b:8e6:d04f | us-east-1 |
3.11.2 | 2600:1111:2222:3333:283b:8e6:d04f | 05deae2d-6405-494d-a965-
c0e5836bcb3c | {'85070591730234615865843651857942052863'}
2600:1111:2222:4444:7d26:5a09:1b44 | us-east-1 | 66666666-035d-37ef-
a247-19a6a867ab09 | 2600:1111:2222:4444:7d26:5a09:1b44 | us-east-1 |
3.11.2 | 2600:1111:2222:4444:7d26:5a09:1b44 | 05deae2d-6405-494d-a965-
c0e5836bcb3c | {'170141183460469231731687303715884105726'}

```

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 管理主控台，連線會自動通過公有端點，而不是您的 VPC 端點。如果您看到九個 IP 地址，這些是 Amazon Keyspaces 自動寫入 `system.peers` 資料表以進行公有端點連線的項目。

步驟 5：使用 CloudWatch 設定監控

此步驟說明如何使用 Amazon CloudWatch 來監控與 Amazon Keyspaces 的 VPC 端點連線。

AWS PrivateLink 會將介面端點的資料點發佈至 CloudWatch。您可以使用指標來確認系統的運作符合預期。CloudWatch 中的 `AWS/PrivateLinkEndpoints` 命名空間包含介面端點的指標。如需詳細資訊，請參閱《AWS PrivateLink 指南》中的 [的 CloudWatch 指標 AWS PrivateLink](#)。

使用 VPC 端點指標建立 CloudWatch 儀表板

1. 在 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。

2. 在導覽窗格中，選擇 Dashboards (儀表板)。然後選擇建立儀表板。輸入儀表板的名稱，然後選擇建立。
3. 在新增小工具下，選擇數字。
4. 在指標下，選擇 AWS/PrivateLinkEndpoints。
5. 選擇端點類型、服務名稱、VPC 端點 ID、VPC ID。
6. 選取指標 ActiveConnections 和 NewConnections，然後選擇建立小工具。
7. 儲存儀表板。

ActiveConnections 指標定義為端點在過去一分鐘期間內收到的並行作用中連線數目。NewConnections 指標定義為過去一分鐘期間內透過端點建立的新連線數目。

如需建立儀表板的詳細資訊，請參閱《CloudWatch 使用者指南》中的[建立儀表板](#)。

步驟 6：(選用) 為應用程式設定連線集區大小的最佳實務

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 每個 TCP 連線每秒最多允許 3,000 個 CQL 查詢。因此，驅動程式可以使用 Amazon Keyspaces 建立的連線數幾乎沒有限制。不過，我們建議您將連線集區大小符合應用程式的需求，並在搭配 VPC 端點連線使用 Amazon Keyspaces 時考慮可用的端點。

您可以在用戶端驅動程式中設定連線集區大小。例如，根據本機集區大小 2 和跨 3 個可用區域建立的 VPC 介面端點，驅動程式會建立 6 個用於查詢的連線（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。

如果您的應用程式需要每秒支援 40,000 個 CQL 查詢，請從判斷所需連線集區大小所需的查詢數目向後工作。若要支援每秒 40,000 個 CQL 查詢，您需要將本機集區大小設定為至少 5 個，這支援每秒至少 45,000 個 CQL 查詢。

您可以使用 AWS/Cassandra 命名空間中的 PerConnectionRequestRateExceeded CloudWatch 指標，監控您是否超過每個連線每秒最大操作數的配額。

此 PerConnectionRequestRateExceeded 指標會顯示對 Amazon Keyspaces 的請求數量，這些請求超過每個連線請求率的配額。

此步驟中的程式碼範例示範如何在使用介面 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。如需 Java 用戶端驅動程式連線的完整範例，請參閱 [the section called “使用 Cassandra Java 用戶端驅動程式”](#)。

當用戶端驅動程式啟動時，首先會為管理任務建立控制連線，例如結構描述和拓撲變更。然後建立額外的連線。

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 端點跨 VPC 內的 3 個子網路建立，這會導致介面端點在 CloudWatch NewConnections 中為 7，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  
    }  
}
```

如果作用中連線的數量不符合您設定的集區大小（跨子網路彙總）+ 1 個控制連線，則會阻止建立連線。

Node.js

您可以在 Node.js 驅動程式中設定每個集區的連線數。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 Cassandra Node.js 用戶端驅動程式”](#)。

以下程式碼範例使用數位憑證的 *keyspaces-bundle.pem* ###如需詳細資訊，請參閱[the section called “開始之前”](#)。

對於下列程式碼範例，本機集區大小驅動程式組態指定為 1。如果 VPC 端點跨 VPC 內的 4 個子網路建立，這會導致介面端點在 CloudWatch NewConnections 中為 5，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/keyspaces-bundle.pem',
    '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：（選用）清除

如果您想要刪除在本教學課程中建立的資源，請遵循這些程序。

移除 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 ID。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\":\"2000-00-00\",\"Statement\":[{\"Effect\":\"Allow\",\"Principal\":\"*\",\"Action\":\"*\",\"Resource\":\"*\"}]}\",
    "VpcId": "vpc-a1234bcd",
    "State": "available",
    "ServiceName": "com.amazonaws.us-east-1.cassandra",
    "RouteTableIds": [],
    "VpcEndpointId": "vpce-1a23b4c5",
    "CreationTimestamp": "2025-07-26T22:00:14Z"
  }
}
```

在範例輸出中，VPC 端點 ID 為 *vpce-1a23b4c5*。請務必將此值取代為您自己的值。

3. 刪除 VPC 端點。

```
aws ec2 delete-vpc-endpoints --vpc-endpoint-ids vpce-1a23b4c5

{
  "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. 前往 <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，選擇執行個體狀態，然後選擇終止。

5. 在確認視窗中，請選擇 Yes, Terminate (是，終止)。

教學課程：與 Apache Spark 整合以匯入或匯出資料

Apache Spark 是用於大規模資料分析的開放原始碼引擎。Apache Spark 可讓您更有效率地對存放在 Amazon Keyspaces 中的資料進行分析。您也可以使用 Amazon Keyspaces 為應用程式提供一致的 single-digit-millisecond 讀取存取 Spark 的分析資料。開放原始碼 Spark Cassandra 連接器可簡化 Amazon Keyspaces 和 Spark 之間的讀取和寫入資料。

Amazon Keyspaces 對 Spark Cassandra Connector 的支援使用全受管無伺服器資料庫服務，簡化在 Spark 型分析管道中執行 Cassandra 工作負載的過程。使用 Amazon Keyspaces，您不需要擔心 Spark 與資料表競爭相同的基礎基礎設施資源。Amazon Keyspaces 資料表會根據您的應用程式流量自動擴展和縮減。

下列教學課程會逐步解說使用 Spark Cassandra 連接器將資料讀取和寫入 Amazon Keyspaces 所需的步驟和最佳實務。本教學課程示範如何使用 Spark Cassandra 連接器從檔案載入資料並將其寫入 Amazon Keyspaces 資料表，以將資料遷移至 Amazon Keyspaces。然後，本教學課程示範如何使用 Spark Cassandra 連接器從 Amazon Keyspaces 讀取資料。您會這麼做，以在 Spark 型分析管道中執行 Cassandra 工作負載。

主題

- [使用 Spark Cassandra 連接器建立 Amazon Keyspaces 連線的先決條件](#)
- [步驟 1：設定 Amazon Keyspaces 以與 Apache Cassandra Spark Connector 整合](#)
- [步驟 2：設定 Apache Cassandra Spark 連接器](#)
- [步驟 3：建立應用程式組態檔案](#)
- [步驟 4：準備 Amazon Keyspaces 中的來源資料和目標資料表](#)
- [步驟 5：使用 Apache Cassandra Spark 連接器寫入和讀取 Amazon Keyspaces 資料](#)
- [對搭配 Amazon Keyspaces 使用 Spark Cassandra 連接器時的常見錯誤進行故障診斷](#)

使用 Spark Cassandra 連接器建立 Amazon Keyspaces 連線的先決條件

使用 Spark Cassandra 連接器連線到 Amazon Keyspaces 之前，您需要確定已安裝下列項目。Amazon Keyspaces 與 Spark Cassandra Connector 的相容性已經過下列建議版本測試：

- Java 第 8 版
- Scala 2.12

- Spark 3.4
- Cassandra 連接器 2.5 及更高版本
- Cassandra 驅動程式 4.12

1. 若要安裝 Scala，請遵循 中的指示<https://www.scala-lang.org/download/scala2.html>。
2. 若要安裝 Spark 3.4.1，請遵循此範例。

```
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
````
```

## 步驟 1：設定 Amazon Keyspaces 以與 Apache Cassandra Spark Connector 整合

在此步驟中，您確認帳戶的分割區器與 Apache Spark Connector 相容，並設定所需的 IAM 許可。下列最佳實務可協助您為資料表佈建足夠的讀取/寫入容量。

1. 確認Murmur3Partitioner分割器是您帳戶的預設分割器。此分割區器與 Spark Cassandra 連接器相容。如需分割器以及如何變更分割器的詳細資訊，請參閱 [the section called “使用分割區”](#)。
2. 使用介面 VPC 端點搭配 Apache Spark 來設定 Amazon Keyspaces 的 IAM 許可。
  - 指派使用者資料表的讀取/寫入存取權，以及系統資料表的讀取存取權，如下列 IAM 政策範例所示。
  - 對於透過 VPC 端點透過 Spark 存取 Amazon Keyspaces 的用戶端，需要使用可用的介面 [VPC 端點](#)填入 system.peers 資料表。

```
{
 "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，為您的 Amazon Keyspaces 資料表設定足夠的讀取/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資料表輸送量成本，請針對連接器的流量使用速率限制器，並將資料表設定為使用佈建容量搭配自動擴展。如需詳細資訊，請參閱[the section called “使用自動擴展管理輸送量容量”](#)。
  - 您可以使用 Cassandra 驅動程式隨附的固定速率限制器。[AWS 範例](#)儲存庫中有一些[針對 Amazon Keyspaces 量身打造的速率限制器](#)。
  - 如需容量管理的詳細資訊，請參閱 [the section called “設定讀取/寫入容量模式”](#)。

## 步驟 2：設定 Apache Cassandra Spark 連接器

Apache Spark 是以不同方式設定的通用運算平台。若要設定 Spark 和 Spark Cassandra Connector 以與 Amazon Keyspaces 整合，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進行增加。

- 建立小於 8 MBs Spark 分割區大小。

在 Spark 中，分割區代表可平行執行的資料原子區塊。當您使用 Spark Cassandra 連接器將資料寫入 Amazon Keyspaces 時，Spark 分割區越小，任務要寫入的記錄量就越小。如果 Spark 任務遇到多個錯誤，則會在指定的重試次數用盡之後失敗。為了避免重新播放大型任務和重新處理大量資料，請將 Spark 分割區的大小保持較小。

- 對具有大量重試的每個執行器使用少量並行寫入。

Amazon Keyspaces 會將容量不足錯誤傳回 Cassandra 驅動程式，做為操作逾時。您無法透過變更設定的逾時持續時間來解決容量不足所造成的逾時，因為 Spark Cassandra Connector 會嘗試使用透明地重試請求 `MultipleRetryPolicy`。為了確保重試不會超過驅動程式的連線集區，請針對具有大量重試的每個執行器，使用較低的並行寫入次數。下列程式碼片段為範例。

```
spark.cassandra.query.retry.count = 500
spark.cassandra.output.concurrent.writes = 3
```

- 細分總輸送量，並將其分散到多個 Cassandra 工作階段。
- Cassandra Spark Connector 會為每個 Spark 執行器建立一個工作階段。將此工作階段視為規模單位，以判斷所需的輸送量和所需的連線數。
- 定義每個執行器的核心數量和每個任務的核心數量時，從低開始並視需要增加。
- 設定 Spark 任務失敗，以在發生暫時性錯誤時允許處理。熟悉應用程式的流量特性和需求後，建議您 `spark.task.maxFailures` 將設定為限定值。
- 例如，下列組態可以處理每個執行器、每個工作階段的兩個並行任務：

```
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。
- 我們建議您關閉批次處理，以改善隨機存取模式。下列程式碼片段為範例。

```
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)
```

- 將 **SPARK\_LOCAL\_DIRS** 設定為具有足夠空間的快速本機磁碟。

- 根據預設，Spark 會將映射輸出檔案和彈性分散式資料集 (RDDs) 儲存到/tmp 資料夾。視 Spark 主機的組態而定，這可能會導致裝置樣式錯誤沒有剩餘空間。
- 若要將SPARK\_LOCAL\_DIRS環境變數設定為名為 的目錄/example/spark-dir，您可以使用下列命令。

```
export SPARK_LOCAL_DIRS=/example/spark-dir
```

## 步驟 3：建立應用程式組態檔案

若要搭配 Amazon Keyspaces 使用開放原始碼 Spark Cassandra Connector，您需要提供應用程式組態檔案，其中包含與 DataStax Java 驅動程式連線所需的設定。您可以使用服務特定的登入資料或 SigV4 外掛程式來連線。

如果您尚未這麼做，則需要將用來建立 TLS 連線的數位憑證轉換為 trustStore 檔案。您可以從 [the section called “開始之前”](#) Java 驅動程式連線教學課程中遵循的詳細步驟。請記下 trustStore 檔案路徑和密碼，因為您在建立應用程式組態檔案時需要此資訊。

### 使用 SigV4 身分驗證連線

本節顯示範例 application.conf 檔案，您可以在與 AWS 登入資料和 SigV4 外掛程式連線時使用。如果您尚未這麼做，則需要產生您的 IAM 存取金鑰（存取金鑰 ID 和私密存取金鑰），並將其儲存在您的 AWS 組態檔案中或做為環境變數。如需詳細說明，請參閱 [the section called “身分 AWS 驗證所需的登入資料”](#)。

在下列範例中，取代 trustStore 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
}

```

更新此組態檔案並將其儲存為 `/home/user1/application.conf`。下列範例使用此路徑。

## 使用服務特定的登入資料連線

本節顯示範例 `application.conf` 檔案，您可以在使用服務特定登入資料進行連線時使用。如果您尚未這麼做，則需要為 Amazon Keyspaces 產生服務特定的登入資料。如需詳細說明，請參閱 [the section called “建立服務特定的登入資料”](#)。

在下列範例中，`password` 使用您自己的登入資料取代 `username` 和 `password`。此外，請取代 `trustStore` 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
 }
 }
}
}

```

更新此組態檔案並儲存為 `/home/user1/application.conf` 以搭配程式碼範例使用。

## 使用固定速率連線

若要強制每個 Spark 執行器的固定速率，您可以定義請求調節器。此請求限流器會限制每秒請求的速率。Spark Cassandra 連接器會為每個執行器部署 Cassandra 工作階段。使用下列公式可協助您針對資料表實現一致的輸送量。

$$\text{max-request-per-second} * \text{numberOfExecutors} = \text{total throughput against a table}$$

您可以將此範例新增至您先前建立的應用程式組態檔案。

```

datastax-java-driver {
 advanced.throttler {
 class = RateLimitingRequestThrottler

 max-requests-per-second = 3000
 max-queue-size = 30000
 drain-interval = 1 millisecond
 }
}

```

## 步驟 4：準備 Amazon Keyspaces 中的來源資料和目標資料表

在此步驟中，您會使用範例資料和 Amazon Keyspaces 資料表建立來源檔案。

1. 建立來源檔案。您可以選擇下列其中一個選項：

- 在本教學課程中，您會使用逗號分隔值 (CSV) 檔案，其名稱 `keyspaces_sample_table.csv` 為資料遷移的來源檔案。提供的範例檔案包含名為 `資料表` 的幾列資料 `book_awards`。



- 下載以下封存檔案 [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。
- 如果您想要遵循自己的 CSV 檔案，將資料寫入 Amazon Keyspaces，請確定資料已隨機化。直接從資料庫讀取或匯出至一般檔案的資料，通常由分割區和主索引鍵排序。將排序資料匯入 Amazon Keyspaces 可能會導致它寫入 Amazon Keyspaces 分割區的較小區段，這會導致流量分佈不均勻。這可能會導致效能較慢和錯誤率較高。

相反地，隨機化資料有助於透過更均勻地跨分割區分配流量，來利用 Amazon Keyspaces 的內建負載平衡功能。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 [Shuf](#) 的範例，請參閱 資料遷移教學[the section called “步驟 2：準備資料”](#)中的。以下是示範如何以 隨機播放資料的範例DataFrame。

```
import org.apache.spark.sql.functions.randval
shuffledDF = dataframe.orderBy(rand())
```

## 2. 在 Amazon Keyspaces 中建立目標金鑰空間和資料表。

- a. 使用 連線至 Amazon Keyspacescqlsh-expansion。如需cqlsh-expansion安裝說明，請參閱 [the section called “使用 cqlsh-expansion”](#)。

將下列範例中的服務端點取代為您自己的值。

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

- b. 建立名稱為 的新金鑰空間catalog，如下列範例所示。

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

- c. 在新的金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。若要進一步了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 “檢查金鑰空間建立狀態”](#)。

```
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)
);
```

## 步驟 5：使用 Apache Cassandra Spark 連接器寫入和讀取 Amazon Keyspaces 資料

在此步驟中，您會先 DataFrame 使用 Spark Cassandra 連接器，將範例檔案中的資料載入。接著，您將資料從 寫入 DataFrame Amazon Keyspaces 資料表。您也可以獨立使用此部分，例如，將資料遷移至 Amazon Keyspaces 資料表。最後，您會 DataFrame 使用 Spark Cassandra 連接器，將資料表中的資料讀取至。您也可以獨立使用此部分，例如，從 Amazon Keyspaces 資料表讀取資料，以使用 Apache Spark 執行資料分析。

1. 啟動 Spark Shell，如下列範例所示。請注意，此範例使用 SigV4 身分驗證。

```
./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. 使用下列程式碼匯入 Spark Cassandra 連接器。

```
import org.apache.spark.sql.cassandra._
```

3. 若要從 CSV 檔案讀取資料並將其存放在 DataFrame 中，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_table.csv")
```

您可以使用下列命令顯示結果。

```
scala> df.show();
```

輸出看起來應該與此類似。

```
+-----+-----+-----+-----+-----+-----+
+-----+
```

```

| 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 中資料的結構描述DataFrame，如下列範例所示。

```
scala> df.printSchema
```

輸出看起來應該像這樣。

```

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. 使用下列命令，將 中的資料寫入 DataFrame Amazon Keyspaces 資料表。

```
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架，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資料框架中的資料。

```
scala> newDf.show()
```

該命令的輸出應如下所示。

```
+-----+-----+-----+-----+
+----+----+
| 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|
+-----+-----+-----+-----+
+----+----+
```

# 對搭配 Amazon Keyspaces 使用 Spark Cassandra 連接器時的常見錯誤進行故障診斷

如果您使用 Amazon Virtual Private Cloud 並連線至 Amazon Keyspaces，則使用 Spark 連接器時最常見的錯誤是由下列組態問題所造成。

- VPC 中使用的 IAM 使用者或角色缺少存取 Amazon Keyspaces 中 `system.peers` 資料表所需的許可。如需詳細資訊，請參閱 [the section called “使用界面 VPC 端點資訊填入 `system.peers` 資料表項目”](#)。
- IAM 使用者或角色缺少使用者資料表的必要讀取/寫入許可，以及對 Amazon Keyspaces 中系統資料表的讀取存取權。如需詳細資訊，請參閱 [the section called “步驟 1：設定 Amazon Keyspaces”](#)。
- Java 驅動程式組態不會在建立 SSL/TLS 連線時停用主機名稱驗證。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，請參閱 [the section called “VPC 端點連線錯誤”](#)。

此外，您可以使用 Amazon CloudWatch 指標來協助您疑難排解 Amazon Keyspaces 中 Spark Cassandra Connector 組態的問題。若要進一步了解如何搭配 CloudWatch 使用 Amazon Keyspaces，請參閱 [the section called “使用 CloudWatch 進行監控”](#)。

下一節說明當您使用 Spark Cassandra 連接器時，應觀察的最有用指標。

## PerConnectionRequestRateExceeded

Amazon Keyspaces 的配額為每個連線每秒 3,000 個請求。每個 Spark 執行器都會建立與 Amazon Keyspaces 的連線。執行多次重試可能會耗盡每個連線請求率配額。如果您超過此配額，Amazon Keyspaces 會在 CloudWatch 中發出 `PerConnectionRequestRateExceeded` 指標。

如果您看到 `PerConnectionRequestRateExceeded` 事件與其他系統或使用者錯誤一起出現，則 Spark 可能會執行超過每個連線所分配請求數的多次重試。

如果您在沒有其他錯誤的情況下看到 `PerConnectionRequestRateExceeded` 事件，則可能需要增加驅動程式設定中的連線數目，以允許更多輸送量，或者您可能需要增加 Spark 任務中的執行器數目。

## StoragePartitionThroughputCapacityExceeded

Amazon Keyspaces 配額為每秒 1,000 WCUs 或 WRUs/每秒每個分割區 3,000 RCUs 或 RRUs。如果您看到 `StoragePartitionThroughputCapacityExceeded` CloudWatch 事件，

可能表示資料未在載入時隨機化。如需如何隨機播放資料的範例，請參閱 [the section called “步驟 4：準備來源資料和目標資料表”](#)。

## 常見錯誤和警告

如果您使用 Amazon Virtual Private Cloud 並連線至 Amazon Keyspaces，Cassandra 驅動程式可能會在 `system.peers` 資料表中發出有關控制節點本身的警告訊息。如需詳細資訊，請參閱 [the section called “常見錯誤和警告”](#)。您可以安全地忽略此警告。

## 教學課程：從託管在 Amazon Elastic Kubernetes Service 上的容器化應用程式連線

本教學課程會逐步引導您設定 Amazon Elastic Kubernetes Service (Amazon EKS) 叢集，以託管使用 SigV4 身分驗證連線至 Amazon Keyspaces 的容器化應用程式。

Amazon EKS 是一項受管服務，無需安裝、操作和維護您自己的 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。

本教學提供 step-by-step 指引，以設定、建置和部署容器化 Java 應用程式至 Amazon EKS。在最後一個步驟中，您執行應用程式將資料寫入 Amazon Keyspaces 資料表。

### 主題

- [從 Amazon EKS 連線至 Amazon Keyspaces 的先決條件](#)
- [步驟 1：設定 Amazon EKS 叢集和設定 IAM 許可](#)
- [步驟 2：設定應用程式](#)
- [步驟 3：建立應用程式映像，並將 Docker 檔案上傳至您的 Amazon ECR 儲存庫](#)
- [步驟 4：將應用程式部署至 Amazon EKS，並將資料寫入資料表](#)
- [步驟 5：\(選用\) 清除](#)

## 從 Amazon EKS 連線至 Amazon Keyspaces 的先決條件

請先建立下列 AWS 資源，再開始教學課程

1. 開始本教學課程之前，請遵循中的 AWS 設定說明 [存取 Amazon Keyspaces \(適用於 Apache Cassandra\)](#)。這些步驟包括註冊 AWS 和建立可存取 Amazon Keyspaces 的 AWS Identity and Access Management (IAM) 主體。

2. 在本教學課程稍後從在 Amazon EKS 中執行的容器化應用程式建立名為 `aws` 的 Amazon Keyspaces 金鑰空間和名為 `user` 的資料表。您可以使用 AWS CLI 或使用 `cqlsh` 來執行此操作。

## AWS CLI

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

若要確認金鑰空間已建立，您可以使用下列命令。

```
aws keyspaces list-keyspaces
```

若要建立資料表，您可以使用下列命令。

```
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}]'
```

若要確認您的資料表已建立，您可以使用下列命令。

```
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
);
```

若要確認您的資料表已建立，您可以使用下列陳述式。

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

您的資料表應列於此陳述式的輸出中。請注意，在建立資料表之前，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 Pod，而無需管理 Amazon EC2 執行個體。若要遵循本教學課程，而無需更新所有範例命令中的叢集名稱，my-eks-cluster 請依照《[Amazon EKS 使用者指南](#)》中的 [Amazon EKS eksctl](#) 入門中的指示，使用名稱建立叢集。建立叢集時，請確認您的節點和兩個預設 Pod 正在執行且運作狀態良好。您可以使用下列命令來執行此操作。

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

您應該會看到類似此輸出的內容。

| 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 的說明，請參閱《[Amazon Elastic Container Registry 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，可與您偏好的 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 Elastic Container Registry 使用者指南](#)》。您可以使用下列命令來建立名為 my-ecr-repository 的儲存庫。

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

完成先決條件步驟後，請繼續 [the section called “步驟 1：設定 Amazon EKS 叢集”](#)。



## 步驟 1：設定 Amazon EKS 叢集和設定 IAM 許可

設定 Amazon EKS 叢集並建立允許 Amazon EKS 服務帳戶連線到 Amazon Keyspaces 資料表所需的 IAM 資源

1. 為 Amazon EKS 叢集建立 Open ID Connect (OIDC) 提供者。這是針對服務帳戶使用 IAM 角色的必要條件。如需 OIDC 提供者以及如何建立它們的詳細資訊，請參閱《Amazon EKS 使用者指南》中的[為您的叢集建立 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，請記得更新所有未來命令中的名稱。

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

- b. 確認 OIDC 身分提供者已使用下列命令向 IAM 註冊。

```
aws iam list-open-id-connect-providers --region us-east-1
```

輸出看起來應該與此類似。請記下 OIDC 的 Amazon Resource Name (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.us-
east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
 }
]
}
```

2. 為 Amazon EKS 叢集建立服務帳戶。服務帳戶為在 Pod 中執行的程序提供身分。Pod 是最小且最簡單的 Kubernetes 物件，可用來部署容器化應用程式。接著，建立服務帳戶可擔任的 IAM 角色，以取得資源的許可。您可以從已設定為使用服務帳戶的 AWS Pod 存取任何服務，該帳戶可擔任具有該服務存取許可的 IAM 角色。
  - a. 為服務帳戶建立新的命名空間。命名空間有助於隔離為此教學課程建立的叢集資源。您可以使用下列命令建立新的命名空間。

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

- b. 若要使用自訂命名空間，您必須將其與 Fargate 設定檔建立關聯。下列程式碼為範例。

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

- c. 使用下列命令，在 Amazon EKS 叢集 *my-eks-namespace* 的命名空間 *my-eks-serviceaccount* 中建立名為 *my-eks-serviceaccount* 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. 執行下列命令來建立信任政策檔案，指示 IAM 角色信任您的服務帳戶。在委託人可以擔任角色之前，需要此信任關係。您需要對 檔案進行下列編輯：
- 針對 *Principal*，輸入 IAM 傳回命令的 ARN *list-open-id-connect-providers*。ARN 包含您的帳戶號碼和區域。
  - 在 *condition* 陳述式中，取代 AWS 區域 和 OIDC ID。
  - 確認服務帳戶名稱和命名空間正確無誤。

建立 IAM 角色時，您需要在下一個步驟中連接信任政策檔案。

```
cat >trust-relationship.json <<EOF
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
```

```

 "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.us-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
 },
 "Action": "sts:AssumeRoleWithWebIdentity",
 "Condition": {
 "StringEquals": {
 "oidc.eks.us-east-1.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:my-eks-
namespace:my-eks-serviceaccount",
 "oidc.eks.us-east-1.amazonaws.com/
id/EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com"
 }
 }
}
]
}
EOF

```

選用：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。若要讓您的服務帳戶在不同帳戶中 AWS 擔任 IAM 角色，請參閱《Amazon EKS 使用者指南》中的[跨帳戶 IAM 許可](#)。

3. 使用要擔任 `my-iam-role` 的 Amazon EKS 服務帳戶的名稱建立 IAM 角色。將最後一個步驟中建立的信任政策檔案連接到角色。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. 連接存取政策，將 IAM 角色許可指派給 Amazon Keyspaces。
  - a. 連接存取政策，以定義 IAM 角色可在特定 Amazon Keyspaces 資源上執行的動作。在本教學課程中，我們使用 AWS 受管政策 `AmazonKeyspacesFullAccess`，因為我們的應用程式會將資料寫入您的 Amazon Keyspaces 資料表。不過，做為最佳實務，建議建立實作最低權限原則的自訂存取政策。如需詳細資訊，請參閱[the section called “Amazon Keyspaces 如何與 IAM 搭配使用”](#)。

```
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連接至 IAM 角色。

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

輸出看起來應該像這樣。

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

- b. 使用可以擔任的 IAM 角色的 Amazon Resource Name (ARN) 來標註服務帳戶。請務必使用您的帳戶 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 角色和服務帳戶已正確設定。

- a. 確認 IAM 角色的信任政策已使用下列陳述式正確設定。

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

輸出看起來應該與此類似。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "Federated": "arn:aws:iam::111122223333:oidc-provider/
oidc.eks.us-east-1.amazonaws.com/id/EXAMPLED539D4633E53DE1B71EXAMPLE"
 },
 "Action": "sts:AssumeRoleWithWebIdentity",
 "Condition": {
 "StringEquals": {
```

```

 "oidc.eks.us-east-1/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:aud": "sts.amazonaws.com",
 "oidc.eks.us-east-1.amazonaws.com/id/
EXAMPLED539D4633E53DE1B71EXAMPLE:sub": "system:serviceaccount:my-eks-
namespace:my-eks-serviceaccount"
 }
}
]
}

```

- b. 確認 Amazon EKS 服務帳戶已標註 IAM 角色。

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

輸出看起來應該與此類似。

```

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>
[...]

```

在您建立 Amazon EKS 服務帳戶、IAM 角色並設定必要的關係和許可之後，請繼續[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，您會建置使用 SigV4 外掛程式連線至 Amazon Keyspaces 的應用程式。您可以從 [Github](#) 上的 Amazon Keyspaces 範例程式碼儲存庫檢視和下載範例 Java 應用程式。或者，您可以使用自己的應用程式遵循，請務必完成所有組態步驟。

設定您的應用程式並新增必要的相依性。

1. 您可以使用下列命令複製 Github 儲存庫來下載範例 Java 應用程式。

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

2. 下載 Github 儲存庫之後，請解壓縮下載的檔案，然後導覽至 `application.conf` 檔案的 `resources` 目錄。

- a. 應用程式組態

在此步驟中，您會設定 SigV4 身分驗證外掛程式。您可以在應用程式中使用以下範例。如果您尚未這麼做，則需要產生 IAM 存取金鑰（存取金鑰 ID 和私密存取金鑰），並將其儲存在 AWS 您的組態檔案中或做為環境變數。如需詳細說明，請參閱 [the section called “身分 AWS 驗證所需的登入資料”](#)。視需要更新 Amazon Keyspaces AWS 的區域和服務端點。如需更多服務端點，請參閱 [the section called “服務端點”](#)。將信任存放區位置、信任存放區名稱和信任存放區密碼取代為您自己的位置。

```
datastax-java-driver {
 basic.contact-points = ["cassandra.us-east-1.amazonaws.com:9142"]
 basic.load-balancing-policy.local-datacenter = "us-east-1"
 advanced.auth-provider {
 class = software.aws.mcs.auth.SigV4AuthProvider
 aws-region = "us-east-1"
 }
 advanced.ssl-engine-factory {
 class = DefaultSslEngineFactory
 truststore-path = "truststore_locationtruststore_name.jks"
 truststore-password = "truststore_password;"
 }
}
```

- b. 新增 STS 模組相依性。

這會新增使用的功能 `WebIdentityTokenCredentialsProvider`，以傳回應用程式需要提供的 AWS 登入資料，以便服務帳戶可以擔任 IAM 角色。您可以根據下列範例執行此操作。

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

- c. 新增 SigV4 相依性。

此套件實作向 Amazon Keyspaces 進行身分驗證所需的 SigV4 身分驗證外掛程式

```
<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。

如果沒有日誌，就無法對連線問題進行疑難排解。在本教學課程中，我們使用 `slf4j` 做為記錄架構，並使用 `logback.xml` 來存放日誌輸出。我們將記錄層級設定為 `debug` 以建立連線。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>
```

您可以使用下列程式碼片段來設定記錄。

```
<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" />
 </root>
</configuration>
```

**Note**

需要 debug 層級來調查連線失敗。從應用程式成功連線至 Amazon Keyspaces 後，您可以以 warning 視需要將記錄層級變更為 info 或。

## 步驟 3：建立應用程式映像，並將 Docker 檔案上傳至您的 Amazon ECR 儲存庫

在此步驟中，您將編譯範例應用程式、建置 Docker 映像，並將映像推送到您的 Amazon ECR 儲存庫。

建置您的應用程式、建置 Docker 映像，並將其提交至 Amazon Elastic Container Registry

1. 為定義的建置設定環境變數 AWS 區域。將範例中的區域取代為您自己的區域。

```
export CASSANDRA_HOST=cassandra.us-east-1.amazonaws.com:9142
export CASSANDRA_DC=us-east-1
```

2. 使用以下命令，使用 Apache Maven 3.6.3 版或更新版本編譯您的應用程式。

```
mvn clean install
```

這會建立包含 target 目錄中所有相依性 JAR 的檔案。

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

```
aws ecr describe-repositories --region us-east-1
```

輸出看起來應該如下例所示。

```
"repositories": [
 {
 "repositoryArn": "arn:aws:ecr:us-east-1:111122223333:repository/my-ecr-repository",
 "registryId": "111122223333",
 "repositoryName": "my-ecr-repository",
 "repositoryUri": "111122223333.dkr.ecr.us-east-1.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，使用最後一個步驟的儲存庫 URI 建置 Docker 映像。視需要修改 Docker 檔案。在建置命令中，請務必取代您的帳戶 ID，並將設定為 my-ecr-repository Amazon ECR 儲存庫所在的 AWS 區域 區域。

```
docker build -t 111122223333.dkr.ecr.us-east-1.amazonaws.com/my-ecr-
repository:latest .
```

5. 擷取身分驗證字符，將 Docker 映像推送至 Amazon ECR。您可以使用下列命令來執行此操作。

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --
password-stdin 111122223333.dkr.ecr.us-east-1.amazonaws.com
```

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

```
aws ecr describe-images --repository-name my-ecr-repository --region us-east-1
```

然後，將 Docker 映像推送至儲存庫。您可以使用下列命令。

```
docker push 111122223333.dkr.ecr.us-east-1.amazonaws.com/my-ecr-repository:latest
```

## 步驟 4：將應用程式部署至 Amazon EKS，並將資料寫入資料表

在教學課程的此步驟中，您可以為應用程式設定 Amazon EKS 部署，並確認應用程式正在執行，並且可以連線至 Amazon Keyspaces。

若要將應用程式部署到 Amazon EKS，您需要在名為 的檔案中設定所有相關設定 deployment.yaml。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。在本教學課程中，我們使用 my-keyspaces-app。

- **Kubernetes 命名空間** Amazon EKS 叢集的命名空間。在本教學課程中，我們使用 `my-eks-namespace`。
- **Amazon EKS 服務帳戶名稱** Amazon EKS 服務帳戶的名稱。在本教學課程中，我們使用 `my-eks-serviceaccount`。
- **映像名稱** 應用程式映像的名稱。在本教學課程中，我們使用 `my-keyspaces-app`。
- **映像 URI** 來自 Amazon ECR 的 Docker 映像 URI。
- **AWS 帳戶 ID** 您的 AWS 帳戶 ID。
- **IAM 角色 ARN** 為服務帳戶所建立之 IAM 角色的 ARN。在本教學課程中，我們使用 `my-iam-role`。
- **AWS 區域** 您建立 AWS 區域 Amazon EKS 叢集 所在的 Amazon EKS 叢集。

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

1. 設定 `deployment.yaml` 檔案。您需要取代下列值：

- `name`
- `namespace`
- `serviceName`
- `image`
- `AWS_ROLE_ARN` value
- AWS 區域 中的 `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.us-east-1.amazonaws.com/my-ecr-repository:latest
 ports:
 - containerPort: 8080
 env:
 - name: CASSANDRA_HOST
 value: "cassandra.us-east-1.amazonaws.com:9142"
 - name: CASSANDRA_DC
 value: "us-east-1"
 - 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: "us-east-1"

```

## 2. 部署 deployment.yaml。

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

輸出看起來應該像這樣。

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

## 3. 檢查 Amazon EKS 叢集命名空間中 Pod 的狀態。

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

輸出看起來應該與此範例類似。

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

如需詳細資訊，您可以使用下列命令。

```
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://41ff7811d33ae4bc398755800abdc132335d51d74f218ba81da0700a6f8c67b
 Image: 111122223333.dkr.ecr.us-east-1.amazonaws.com/
my_eks_repository:latest
 Image ID: 111122223333.dkr.ecr.us-east-1.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.us-east-1.amazonaws.com:9142
 CASSANDRA_DC: us-east-1
 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: us-east-1
 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.us-east-1.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.us-east-1.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.us-east-1.amazonaws.com/
my_eks_repository:latest" in 222ms (222ms including waiting)
Normal Started 53s (x2 over 75s) kubelet Started container
my-keysaces-app

```

4. 檢查 Pod 的日誌，以確認您的應用程式正在執行，並可以連線至您的 Amazon Keyspaces 資料表。您可以使用下列命令來執行此操作。請務必取代部署的名稱。

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

您應該能夠看到應用程式日誌項目，確認與 Amazon Keyspaces 的連線，如以下範例所示。

```

2:47:20.553 [s0-admin-0] DEBUG c.d.o.d.i.c.metadata.MetadataManager
- [s0] Adding initial contact points [Node(endPoint=cassandra.us-
east-1.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.c.core.context.EventBus - [s0] Registering
com.datastax.oss.driver.internal.core.metadata.LoadBalancingPolicyWrapper$$Lambda
$812/0x00000000801105e88@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：(選用) 清除

請依照下列步驟移除在本教學課程中建立的所有資源。

## 移除在本教學課程中建立的資源

1. 刪除您的部署。您可以使用下列命令來執行此操作。

```
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 Pod。這也會刪除相關資源，例如服務帳戶和 OIDC 身分提供者。您可以使用下列命令來執行此操作。

```
eksctl delete cluster --name my-eks-cluster --region us-east-1
```

3. 刪除具有 Amazon Keyspaces 存取許可的 Amazon EKS 服務帳戶所使用的 IAM 角色。首先，您必須移除連接到角色的 受管政策。

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

然後，您可以使用下列命令刪除角色。

```
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，包括其中存放的所有映像。您可以使用下列命令來執行此操作。

```
aws ecr delete-repository \
 --repository-name my-ecr-repository \
 --force \
 --region us-east-1
```

請注意，刪除包含映像的儲存庫需要 `force` 旗標。若要先刪除映像，您可以使用下列命令執行此操作。

```
aws ecr batch-delete-image \
 --repository-name my-ecr-repository \
 --image-ids imageTag=latest \
 --region us-east-1
```

如需詳細資訊，請參閱 [《Amazon Elastic Container Registry 使用者指南》中的刪除映像](#)。

5. 刪除 Amazon Keyspaces 金鑰空間和資料表。刪除金鑰空間會自動刪除該金鑰空間中的所有資料表。您可以使用下列其中一個選項來執行此操作。

## AWS CLI

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

若要確認金鑰空間已刪除，您可以使用下列命令。

```
aws keyspaces list-keyspaces
```

若要先刪除資料表，您可以使用下列命令。

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

若要確認您的資料表已刪除，您可以使用下列命令。

```
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";
```

若要確認您的金鑰空間已刪除，您可以使用下列陳述式。

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

您的金鑰空間不應列在此陳述式的輸出中。請注意，在刪除金鑰空間之前，可能會有延遲。如需詳細資訊，請參閱[the section called “DROP KEYSPACE”](#)。

若要先刪除資料表，您可以使用下列命令。

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

若要確認您的資料表已刪除，您可以使用下列命令。



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

您的資料表不應列在此陳述式的輸出中。請注意，在刪除資料表之前，可能會有延遲。如需詳細資訊，請參閱[the section called “DROP TABLE”](#)。

## 教學課程：使用 將 Amazon Keyspaces 資料表匯出至 Amazon S3 AWS Glue

本教學課程說明如何使用 將 Amazon Keyspaces 資料表匯出至 Amazon S3 儲存貯體 AWS Glue。在本教學課程中，許多手動步驟會使用 Amazon Keyspaces [Github](#) 儲存庫中提供的 shell 指令碼來自動化。使用此程序，您可以將 Amazon Keyspaces 資料匯出至 Amazon S3，而不必設定 Spark 叢集。

### 主題

- [從 Amazon Keyspaces 匯出資料至 Amazon S3 的先決條件](#)
- [步驟 1：建立 Amazon S3 儲存貯體、下載所需的工具，以及設定環境](#)
- [步驟 2：設定匯出 Amazon Keyspaces 資料表 AWS Glue 的任務](#)
- [步驟 3：執行 AWS Glue 任務，將 Amazon Keyspaces 資料表從 匯出至 Amazon S3 儲存貯體 AWS CLI](#)
- [步驟 4：（選用）建立觸發以排程匯出任務](#)
- [步驟 5：（選用）清除](#)

## 從 Amazon Keyspaces 匯出資料至 Amazon S3 的先決條件

請先確認下列先決條件並建立 Amazon Keyspaces 資源，再開始教學課程

1. 開始本教學課程之前，請遵循中的 AWS 設定說明[存取 Amazon Keyspaces（適用於 Apache Cassandra）](#)。這些步驟包括註冊 AWS 和建立可存取 Amazon Keyspaces 的 AWS Identity and Access Management (IAM) 主體。
2. 本教學課程中的指令碼會使用您的登入資料，以及 AWS 區域 存放在已知位置的預設。如需詳細資訊，請參閱[the section called “管理存取金鑰”](#)。

下列範例顯示如何將必要值儲存為預設使用者的環境變數。

```
$ export AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
```

```
$ export AWS_SECRET_ACCESS_KEY=wJa1rXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
$ export AWS_DEFAULT_REGION=us-east-1
```

3. 若要執行本教學課程中的指令碼，您需要在機器上安裝下列軟體和工具：

- [Java](#)
- [Apache Maven](#)
- [Git](#)
- [AWS CLI](#)

本教學課程已使用 AWS CLI 2、Java 17.0.13 和 Apache Maven 3.8.7 進行測試。

4. 您需要具有範例資料的 Amazon Keyspaces 資料表，才能在本教學課程稍後進行匯出。您可以使用自己的 Amazon Keyspaces 資料表，或依照[開始使用](#)教學課程中的步驟建立範例資料表。
- 若要安裝 cqlsh-expansion，請遵循 中的步驟[the section called “使用 cqlsh-expansion”](#)。
  - 確認Murmur3Partitioner分割器是您帳戶的預設分割器。此分割器與 Apache Spark Cassandra 連接器和 相容 AWS Glue。如需分割器的詳細資訊，請參閱 [the section called “使用分割區”](#)。

若要變更帳戶的分割區，您可以使用下列陳述式。

```
SELECT partitioner FROM system.local;

UPDATE system.local set
partitioner='org.apache.cassandra.dht.Murmur3Partitioner' where key='local';
```

- 若要建立 Amazon Keyspaces 金鑰空間，請遵循 中的步驟[the section called “建立金鑰空間”](#)。
- 若要建立 Amazon Keyspaces 資料表，請遵循 中的步驟[the section called “建立資料表”](#)。
- 若要將範例資料載入資料表以匯出至 Amazon S3，請遵循 中的步驟[the section called “建立”](#)。

完成先決條件步驟後，請繼續 [the section called “步驟 1：建立 Amazon S3 儲存貯體、下載工具和設定環境”](#)。

## 步驟 1：建立 Amazon S3 儲存貯體、下載所需的工具，以及設定環境

在此步驟中，您會下載外部工具，AWS 並使用 AWS Glue 任務建立和設定 Amazon Keyspaces 資料表自動資料匯出解決方案至 Amazon S3 儲存貯體所需的資源。為了以有效率的方式執行所有這些任務，我們會執行具有 [Github](#) 上 `setup-connector.sh` 可用名稱的 shell 指令碼。

指令碼 `setup-connector.sh` 會自動執行下列步驟。

1. 使用 建立 Amazon S3 儲存貯體 CloudFormation。此儲存貯體會存放下載的 jar 和組態檔案，以及匯出的資料表資料。
2. 使用 建立 IAM 角色 CloudFormation。AWS Glue jobs 使用此角色存取 Amazon Keyspaces 和 Amazon S3。
3. 下載 [Apache Spark Cassandra 連接器](#) 並將其上傳至 Amazon S3 儲存貯體。
4. 下載 [SigV4 身分驗證外掛程式](#)，並將其上傳至 Amazon S3 儲存貯體。
5. 下載 [Apache Spark 延伸](#) 模組並將其上傳至 Amazon S3 儲存貯體。
6. 從 Github 下載 [Keyspaces 重試政策](#)、使用 Maven 編譯程式碼，並將輸出上傳至 Amazon S3 儲存貯體。
7. 將 `keyspaces-application.conf` 檔案上傳至 Amazon S3 儲存貯體。

使用 **`setup-connector.sh`** shell 指令碼來自動化設定和組態步驟。

1. 將檔案從 Github 上的 [aws-glue](#) 儲存庫複製到本機電腦。此目錄包含 shell 指令碼和其他必要檔案。
2. 執行 shell 指令碼 `setup-connector.sh`。您可以指定下列三個選用參數。
  1. `SETUP_STACKNAME` – 這是用來建立 AWS 資源的 CloudFormation 堆疊名稱。
  2. `S3_BUCKET_NAME` – 這是 Amazon S3 儲存貯體的名稱。
  3. `GLUE_SERVICE_ROLE_NAME` – 這是 IAM 服務角色的名稱，AWS Glue 用於執行連線至 Amazon Keyspaces 和 Amazon S3 的任務。

您可以使用下列命令來執行 shell 指令碼，提供具有下列名稱的三個參數。

```
./setup-connector.sh cfn-setup s3-keyspaces iam-export-role
```

若要確認您的儲存貯體已建立，您可以使用下列 AWS CLI 命令。

```
aws s3 ls s3://s3-keyspaces
```

命令的輸出看起來應該像這樣。

```
PRE conf/
PRE jars/
```

若要確認已建立 IAM 角色並檢閱詳細資訊，您可以使用下列 AWS CLI 陳述式。

```
aws iam get-role --role-name "iam-export-role"
```

```
{
 "Role": {
 "Path": "/",
 "RoleName": "iam-export-role",
 "RoleId": "AKIAIOSFODNN7EXAMPLE",
 "Arn": "arn:aws:iam::111122223333:role/iam-export-role",
 "CreateDate": "2025-01-28T16:09:03+00:00",
 "AssumeRolePolicyDocument": {
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Principal": {
 "Service": "glue.amazonaws.com"
 },
 "Action": "sts:AssumeRole"
 }
]
 },
 "Description": "AWS Glue service role to import and export data from Amazon
Keyspaces",
 "MaxSessionDuration": 3600,
 "RoleLastUsed": {
 "LastUsedDate": "2025-01-29T12:03:54+00:00",
 "Region": "us-east-1"
 }
 }
}
```

如果 CloudFormation 堆疊程序失敗，您可以在 CloudFormation 主控台中檢閱有關失敗堆疊的詳細錯誤資訊。

建立包含所有指令碼和工具的 Amazon S3 儲存貯體並設定 IAM 角色後，請繼續 [the section called “步驟 2：設定 AWS Glue 任務”](#)。

## 步驟 2：設定匯出 Amazon Keyspaces 資料表 AWS Glue 的任務

在教學課程的第二個步驟中，您可以使用 [Github](#) 上 `setup-export.sh` 提供的指令碼來建立和設定使用 SigV4 外掛程式連線至 Amazon Keyspaces AWS Glue 的任務，然後將指定的資料表匯出至上一個步驟中建立的 Amazon S3 儲存貯體。使用指令碼可讓您從 Amazon Keyspaces 匯出資料，而無需設定 Apache Spark 叢集。

建立 AWS Glue 任務，將 Amazon Keyspaces 資料表匯出至 Amazon S3 儲存貯體。

- 在此步驟中，您會執行位於 `export-to-s3/` 目錄中的 `setup-export.sh` shell 指令碼，以用來 CloudFormation 建立和設定 AWS Glue 匯出任務。指令碼採用下列參數。

```
PARENT_STACK_NAME, EXPORT_STACK_NAME, KEYSpace_NAME, TABLE_NAME, S3_URI, FORMAT
```

- PARENT\_STACK\_NAME – 在上一個步驟中建立的 CloudFormation 堆疊名稱。
- EXPORT\_STACK\_NAME – 建立 AWS Glue 匯出任務的 CloudFormation 堆疊名稱。
- KEYSPACE\_NAME 和 TABLE\_NAME – 要匯出之金鑰空間和資料表的完整名稱。在本教學課程中，我們使用 `catalog.book_awards`，但您可以將此取代為您自己的完整資料表名稱。
- S3URI – Amazon S3 儲存貯體的選用 URI。預設值是父堆疊的 Amazon S3 儲存貯體。
- FORMAT – 選用的資料格式。預設值為 `parquet`。在本教學課程中，為了讓資料載入和轉換更容易，我們使用預設值。

您可以使用下列命令做為範例。

```
setup-export.sh cfn-setup cfn-glue catalog book_awards
```

若要確認任務已建立，您可以使用下列陳述式。

```
aws glue list-jobs
```

陳述式的輸出看起來應該與此類似。

```
{
 "JobNames": [
 "AmazonKeyspacesExportToS3-cfn-setup-cfn-glue"
]
}
```

若要查看任務的詳細資訊，您可以使用下列命令。

```
aws glue get-job --job-name AmazonKeyspacesExportToS3-cfn-setup-cfn-glue
```

命令的輸出會顯示任務的所有詳細資訊。這包括您在執行任務時可覆寫的預設引數。

```
{
 "Job": {
 "Name": "AmazonKeyspacesExportToS3-cfn-setup-cfn-glue",
 "JobMode": "SCRIPT",
 "JobRunQueuingEnabled": false,
 "Description": "export to s3",
 "Role": "iam-export-role",
 "CreatedOn": "2025-01-30T15:53:30.765000+00:00",
 "LastModifiedOn": "2025-01-30T15:53:30.765000+00:00",
 "ExecutionProperty": {
 "MaxConcurrentRuns": 1
 },
 "Command": {
 "Name": "glueetl",
 "ScriptLocation": "s3://s3-keyspaces/scripts/cfn-setup-cfn-glue-export.scala",
 "PythonVersion": "3"
 },
 "DefaultArguments": {
 "--write-shuffle-spills-to-s3": "true",
 "--S3_URI": "s3://s3-keyspaces",
 "--TempDir": "s3://s3-keyspaces/shuffle-space/export-sample/",
 "--extra-jars": "s3://s3-keyspaces/jars/spark-cassandra-connector-assembly_2.12-3.1.0.jar,s3://s3-keyspaces/jars/aws-sigv4-auth-cassandra-java-driver-plugin-4.0.9-shaded.jar,s3://s3-keyspaces/jars/spark-extension_2.12-2.8.0-3.4.jar,s3://s3-keyspaces/jars/amazon-keyspaces-helpers-1.0-SNAPSHOT.jar",
 "--class": "GlueApp",
 "--user-jars-first": "true",

```

```

 "--enable-metrics": "true",
 "--enable-spark-ui": "true",
 "--KEYSPACE_NAME": "catalog",
 "--spark-event-logs-path": "s3://s3-keyspaces/spark-logs/",
 "--enable-continuous-cloudwatch-log": "true",
 "--write-shuffle-files-to-s3": "true",
 "--FORMAT": "parquet",
 "--TABLE_NAME": "book_awards",
 "--job-language": "scala",
 "--extra-files": "s3://s3-keyspaces/conf/keyspaces-application.conf",
 "--DRIVER_CONF": "keyspaces-application.conf"
 },
 "MaxRetries": 0,
 "AllocatedCapacity": 4,
 "Timeout": 2880,
 "MaxCapacity": 4.0,
 "WorkerType": "G.2X",
 "NumberOfWorkers": 2,
 "GlueVersion": "3.0"
}
}

```

如果 CloudFormation 堆疊程序失敗，您可以在 CloudFormation 主控台中檢閱失敗堆疊的錯誤。您可以在左側選單中選擇 ETL 任務，以在 AWS Glue 主控台中檢閱匯出任務的詳細資訊。

在您確認 AWS Glue 匯出任務的詳細資訊後，請繼續[the section called “步驟 3：從 執行匯出 AWS Glue 任務 AWS CLI”](#)執行任務以從 Amazon Keyspaces 資料表匯出資料。

## 步驟 3：執行 AWS Glue 任務，將 Amazon Keyspaces 資料表從 匯出至 Amazon S3 儲存貯體 AWS CLI

在此步驟中，您可以使用 AWS CLI 執行在上一個步驟中建立 AWS Glue 的任務，將 Amazon Keyspaces 資料表匯出至 Amazon S3 中的儲存貯體。

### 從 執行匯出任務 AWS CLI

1. 在下列範例中，AWS CLI 命令會執行在上一個步驟中建立的任務。

```
aws glue start-job-run --job-name AmazonKeyspacesExportToS3-cfn-setup-cfn-glue
```

- 您可以覆寫任何 AWS Glue 任務參數，包括 AWS CLI 命令中的預設引數。若要覆寫任務的任何預設引數，例如鍵空間或資料表名稱，您可以將它們做為引數傳遞。如需引數的完整清單，請參閱《AWS Glue 命令列參考》中的 [start-job-run](#)。

下列命令會執行 AWS Glue 匯出任務，但會覆寫 AWS Glue 工作者數量、工作者類型和資料表名稱。

```
aws glue start-job-run --job-name AmazonKeyspacesExportToS3-cfn-setup-cfn-glue \
 --number-of-workers 8 --worker-type G.2X \
 --arguments '{"--TABLE_NAME":"my_table"}'
```

2. 確認您的資料表已匯出至 Amazon S3 儲存貯體。根據資料表的大小，這可能需要一些時間。匯出任務完成後，您可以使用範例命令在儲存貯體中看到下列資料夾。

```
aws s3 ls s3://s3-keyspaces
```

輸出會在您的儲存貯體中顯示下列結構。

```
PRE conf/
PRE export/
PRE jars/
PRE scripts/
PRE spark-logs/
```

您的檔案將位於 下的下列資料夾結構中 `export`，資料/時間值會顯示您自己的值。

```
\----- export
 \----- keyspace_name
 \----- table_name
 \----- snapshot
 \----- year=2025
 \----- month=01
 \----- day=02
 \----- hour=09
 \----- minute=22
 \--- YOUR DATA HERE
```



若要排程您剛手動執行 AWS Glue 的任務，請繼續 [the section called “步驟 4：\(選用\) 排程匯出任務”](#)。

## 步驟 4：(選用) 建立觸發以排程匯出任務

若要定期執行在上一個步驟中建立的匯出任務，您可以建立排程的觸發。如需詳細資訊，請參閱《AWS Glue 開發人員指南》中的 [AWS Glue 觸發條件](#)。

### 排程 AWS Glue 任務

1. 下列 AWS CLI 命令是簡單觸發的範例，其名為 `KeyspacesExportWeeklyTrigger`，每週在星期一 12:00 UTC 以名稱執行 AWS Glue 任務 `AmazonKeyspacesExportToS3-cfn-setup-cfn-glue` 一次。

```
aws glue create-trigger \
 --name KeyspacesExportWeeklyTrigger \
 --type SCHEDULED \
 --schedule "cron(0 12 ? * MON *)" \
 --start-on-creation \
 --actions '[
 "JobName": "AmazonKeyspacesExportToS3-cfn-setup-cfn-glue"
]'
```

- 若要覆寫排程任務的任何預設設定，您可以將它們做為引數傳遞。在此範例中，我們將金鑰空間名稱、資料表名稱、工作者數量和工作者類型做為引數傳遞來覆寫。下列命令是此範例。

```
aws glue create-trigger \
 --name KeyspacesExportWeeklyTrigger \
 --type SCHEDULED \
 --schedule "cron(0 12 ? * MON *)" \
 --start-on-creation \
 --actions '[
 "JobName": "AmazonKeyspacesExportToS3-cfn-setup-cfn-glue",
 "Arguments": {
 "--number-of-workers": "8",
 "--worker-type": "G.2X",
 "--table_name": "my_table",
 "--keyspace_name": "my_keyspace"
 }
]'
```

2. 若要確認已建立觸發，請使用下列命令。

```
aws glue list-triggers
```

命令的輸出看起來應該與此類似。

```
{
 "TriggerNames": [
 "KeyspacesExportWeeklyTrigger"
]
}
```

若要清除本教學課程中建立 AWS 的資源，請繼續 [the section called “步驟 5：（選用）清除”](#)。

## 步驟 5：（選用）清除

請依照下列步驟移除在本教學課程中建立的所有 AWS 資源。

移除在本教學課程中建立的資源

1. 刪除在本教學課程中建立的第二個 CloudFormation 堆疊。這會移除在本教學課程中建立 AWS Glue 的任務和觸發。您可以使用下列命令執行此操作。

```
aws cloudformation delete-stack --stack-name cfn-glue
```

2. 刪除 Amazon S3 儲存貯體以及其中存放的所有資料。您可以使用下列命令來執行此操作。

```
aws s3 rm s3://s3-keyspaces --recursive
```

3. 刪除在本教學課程中建立的第一個堆疊。這會刪除在本教學課程中建立的 IAM 角色和相關聯的許可。您可以使用下列命令做為範例。

```
aws cloudformation delete-stack --stack-name cfn-setup
```

4. 刪除 Amazon Keyspaces 金鑰空間和資料表。刪除金鑰空間會自動刪除該金鑰空間中的所有資料表。您可以使用下列其中一個選項來執行此操作。

AWS CLI

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

若要確認金鑰空間已刪除，您可以使用下列命令。

```
aws keyspaces list-keyspaces
```

若要先刪除資料表，您可以使用下列命令。

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

若要確認您的資料表已刪除，您可以使用下列命令。

```
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";
```

若要確認您的金鑰空間已刪除，您可以使用下列陳述式。

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

此陳述式的輸出中不應列出您的 金鑰空間。請注意，在刪除金鑰空間之前，可能會有延遲。如需詳細資訊，請參閱[the section called “DROP KEYSPACE”](#)。

若要先刪除資料表，您可以使用下列命令。

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

若要確認您的資料表已刪除，您可以使用下列命令。

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

您的資料表不應列在此陳述式的輸出中。請注意，在刪除資料表之前，可能會有延遲。如需詳細資訊，請參閱[the section called “DROP TABLE”](#)。

# 在 Amazon Keyspaces 中管理無伺服器資源（適用於 Apache Cassandra）

Amazon Keyspaces (適用於 Apache Cassandra) 是無伺服器服務。Amazon Keyspaces 會將儲存體和讀取/寫入輸送量資源直接配置給資料表，而不是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。

Amazon Keyspaces 會根據儲存在資料表中的資料自動佈建儲存體。當您寫入、更新和刪除資料時，它會擴展和縮減儲存體，而且您只需為使用的儲存體付費。資料會跨多個[可用區域](#)複寫，以提供高可用性。Amazon Keyspaces 會持續監控資料表的大小，以判斷您的儲存費用。如需 Amazon Keyspaces 如何計算資料計費大小的詳細資訊，請參閱 [the section called “估算資料列大小”](#)。

本章涵蓋 Amazon Keyspaces 中資源管理的重要層面。

- 估算資料列大小 – 若要估算 Amazon Keyspaces 中資料列的編碼大小，請考慮分割區索引鍵中繼資料、叢集資料欄中繼資料、資料欄識別符、資料類型和資料列中繼資料等因素。此編碼的資料列大小用於計費、配額管理和佈建的輸送量容量規劃。
- 估計容量耗用 – 本節涵蓋如何預估範圍查詢、限制查詢、資料表掃描、輕量型交易、靜態資料欄和多區域資料表等常見案例的讀取和寫入容量耗用的範例。您可以使用 Amazon CloudWatch 來監控實際容量使用率。如需使用 CloudWatch 監控的詳細資訊，請參閱 [the section called “使用 CloudWatch 進行監控”](#)。
- 設定讀取/寫入容量模式 – 您可以選擇兩種容量模式，以處理資料表上的讀取和寫入：
  - 隨需模式（預設） – 讀取和寫入輸送量的每個請求付費。Amazon Keyspaces 可以立即將容量擴展到任何先前達到的流量層級。
  - 佈建模式 – 事先指定所需的讀取和寫入容量單位數量。此模式有助於維持可預測的輸送量效能。
- 使用自動擴展管理輸送量容量 – 對於佈建的資料表，您可以啟用自動擴展，以根據實際應用程式流量自動調整輸送量容量。Amazon Keyspaces 使用目標追蹤來增加或減少佈建容量，將使用率保持在指定的目標。
- 有效使用高載容量 – Amazon Keyspaces 透過保留一部分未使用的輸送量來處理流量尖峰，提供高載容量。這種靈活性允許偶爾超出佈建輸送量的活動暴增。

若要疑難排解容量錯誤，請參閱 [the section called “無伺服器容量錯誤”](#)。

## 主題

- [估計 Amazon Keyspaces 中的資料列大小](#)

- [預估 Amazon Keyspaces 中讀取和寫入輸送量的容量耗用](#)
- [在 Amazon Keyspaces 中設定讀取/寫入容量模式](#)
- [使用 Amazon Keyspaces 自動擴展自動管理輸送量容量](#)
- [在 Amazon Keyspaces 中有效使用高載容量](#)

## 估計 Amazon Keyspaces 中的資料列大小

Amazon Keyspaces 提供全受管儲存，可提供單一位數毫秒的讀取和寫入效能，並持久地跨多個 AWS 可用區域存放資料。Amazon Keyspaces 會將中繼資料連接至所有資料列和主索引鍵資料欄，以支援高效率的資料存取和高可用性。

本主題提供如何估計 Amazon Keyspaces 中資料列編碼大小的詳細資訊。計算帳單和配額用量時，會使用編碼的資料列大小。在估算資料表的佈建輸送量容量需求時，您也可以使用編碼的資料列大小。

若要計算 Amazon Keyspaces 中資料列的編碼大小，您可以使用下列準則。

### 主題

- [估算欄的編碼大小](#)
- [根據資料類型估計資料值的編碼大小](#)
- [考慮 Amazon Keyspaces 功能對資料列大小的影響](#)
- [選擇正確的公式來計算資料列的編碼大小](#)
- [資料列大小計算範例](#)

## 估算欄的編碼大小

本節說明如何預估 Amazon Keyspaces 中資料欄的編碼大小。

- 一般資料欄 – 對於非主索引鍵、叢集資料欄或STATIC資料欄的資料欄的一般資料欄，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。在 Amazon Keyspaces 存放資料類型值和中繼資料的方式方面，資料類型和一些關鍵差異會列在下一節中。
- 分割區索引鍵資料欄 – 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欄最多需要 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欄使用完整 3 個位元組的中繼資料。

- 叢集資料欄 – 叢集資料欄最多可存放 850 個位元組的資料。除了資料值的大小之外，每個叢集資料欄最多需要中繼資料資料值大小的 20%。計算資料列的大小時，您應該為叢集資料欄資料值的每 5 個位元組新增 1 個位元組的中繼資料。

#### Note

為了支援高效查詢和內建索引，Amazon Keyspaces 會儲存每個分割區索引鍵和叢集索引鍵資料欄的資料值兩次。

- 資料欄名稱 – 每個資料欄名稱所需的空間會使用資料欄識別符存放，並新增至存放在資料欄中的每個資料值。資料欄識別符的儲存值取決於資料表中的資料欄總數：
  - 1–62 欄：1 個位元組
  - 63–124 欄：2 個位元組
  - 125–186 欄：3 個位元組

對於每個額外的 62 欄，新增 1 個位元組。請注意，在 Amazon Keyspaces 中，最多可以使用單一 INSERT 或 UPDATE 陳述式修改 225 個一般資料欄。如需詳細資訊，請參閱[the section called “Amazon Keyspaces 服務配額”](#)。

## 根據資料類型估計資料值的編碼大小

本節說明如何預估 Amazon Keyspaces 中不同資料類型的編碼大小。

- 字串類型 – Cassandra TEXT、ASCII 和 VARCHAR 字串資料類型都存放在使用 Unicode 搭配 UTF-8 二進位編碼的 Amazon Keyspaces 中。Amazon Keyspaces 中字串的大小等於 UTF-8 編碼位元組的數量。
- 數值類型 – Cassandra INT、BIGINT、TINYINT、SMALLINT 和 VARINT 資料類型會以可變長度的資料值儲存在 Amazon Keyspaces 中，最多 38 個有效數字。前後的零會截去。任何這些資料類型的大小約為每兩個有效數字 1 個位元組 + 1 個位元組。
- Blob 類型 – Amazon Keyspaces BLOB 中的 是以值的原始位元組長度存放。
- 布林值類型 – Boolean 值的大小 Null 為 1 個位元組。
- 集合類型 – 儲存集合資料類型的資料欄，例如 LIST 或 MAP 需要 3 個位元組的中繼資料，無論其內容為何。LIST 或 的大小 MAP 是 (資料欄 ID) + 總和 (巢狀元素的大小) + (3 個位元組)。空白 LIST 或 的大小 MAP 是 (資料欄 ID) + (3 位元組)。每個個別 LIST 或 MAP 元素也需要 1 個位元組的中繼資料。

- 使用者定義類型 – [使用者定義類型 \(UDT\)](#) 需要 3 個位元組的中繼資料，無論其內容為何。對於每個 UDT 元素，Amazon Keyspaces 需要額外的 1 位元組中繼資料。

若要計算 UDT 的編碼大小，請從 UDT 欄位的 field name 和 field value 開始：

- 欄位名稱 – 最上層 UDT 的每個欄位名稱都是使用 識別符 存放。識別符的儲存值取決於最上層 UDT 中的欄位整體數量，並且可能介於 1 到 3 個位元組之間：
  - 1–62 個欄位：1 位元組
  - 63–124 個欄位：2 個位元組
  - 125 – 欄位上限：3 個位元組
- 欄位值 – 儲存最上層 UDT 欄位值所需的位元組取決於儲存的資料類型：
  - 純量資料類型 – 儲存所需的位元組與存放在一般資料欄中的相同資料類型相同。
  - 凍結 UDT – 對於每個凍結巢狀 UDT，巢狀 UDT 的大小與 CQL 二進位通訊協定中的大小相同。對於巢狀 UDT，每個欄位（包括空白欄位）會儲存 4 個位元組，而儲存欄位的值是欄位值的 CQL 二進位通訊協定序列化格式。
  - 凍結集合：
    - LIST 和 SET – 對於巢狀凍結 LIST 或 SET，會為集合的每個元素儲存 4 個位元組，加上集合值的 CQL 二進位通訊協定序列化格式。
    - MAP – 對於巢狀凍結 MAP，每個鍵值對具有下列儲存需求：
      - 對於每個金鑰配置 4 個位元組，然後新增金鑰的 CQL 二進位通訊協定序列化格式。
      - 對於每個值配置 4 個位元組，然後新增值的 CQL 二進位通訊協定序列化格式。
- FROZEN 關鍵字 – 對於巢狀在凍結集合中的凍結集合，Amazon Keyspaces 不需要中繼資料的任何額外位元組。
- STATIC 關鍵字 – STATIC 資料欄資料不會計入 1 MB 的資料列大小上限。若要計算靜態資料欄的資料大小，請參閱 [the section called “計算每個邏輯分割區的靜態資料欄大小”](#)。

## 考慮 Amazon Keyspaces 功能對資料列大小的影響

本節顯示 Amazon Keyspaces 中的功能如何影響資料列的編碼大小。

- 用戶端時間戳記 – 開啟功能時，會為每個資料列中的每個資料欄存放用戶端時間戳記。這些時間戳記大約需要 20–40 個位元組（取決於您的資料），並對資料列的儲存和輸送量成本做出貢獻。如需用戶端時間戳記的詳細資訊，請參閱 [the section called “用戶端時間戳記”](#)。
- 存留時間 (TTL) – 開啟功能時，TTL 中繼資料一列大約需要 8 個位元組。此外，也會為每個資料列的每一欄存放 TTL 中繼資料。儲存純量資料類型或凍結集合的每個資料欄，TTL 中繼資料會佔用約 8



個位元組。如果資料欄存放的集合資料類型未凍結，則集合 TTL 的每個元素都需要大約 8 個額外位元組的中繼資料。對於在啟用 TTL 時存放集合資料類型的資料欄，您可以使用下列公式。

```
total encoded size of column = (column id) + sum (nested elements + collection
metadata (1 byte) + TTL metadata (8 bytes)) + collection column metadata (3 bytes)
```

TTL 中繼資料對資料列的儲存和輸送量成本有貢獻。如需 TTL 的詳細資訊，請參閱 [the section called “使用存留時間過期資料”](#)。

## 選擇正確的公式來計算資料列的編碼大小

本節顯示不同的公式，您可以用來估計 Amazon Keyspaces 中資料列的儲存體或容量輸送量需求。

一列資料的總編碼大小可根據您的目標，根據下列其中一個公式估計：

- 輸送量容量 – 預估資料列的編碼大小，以評估所需的讀取/寫入請求單位 (RRUs/WRUs) 或讀取/寫入容量單位 RCUs/WCUs)：

```
total encoded size of row = partition key columns + clustering columns + regular
columns
```

- 儲存體大小 – 若要預估要預測的資料列編碼大小 `BillableTableSizeInBytes`，請新增資料列儲存體所需的中繼資料：

```
total encoded size of row = partition key columns + clustering columns + regular
columns + row metadata (100 bytes)
```

### Important

所有資料欄中繼資料，例如資料欄 ID、分割區索引鍵中繼資料、叢集資料欄中繼資料，以及用戶端時間戳記、TTL 和資料列中繼資料，都會計入 1 MB 的資料列大小上限。

## 資料列大小計算範例

請考慮下列資料表範例，其中所有資料欄都是整數類型。資料表有兩個分割區索引鍵資料欄、兩個叢集資料欄和一個一般資料欄。由於此資料表有五個資料欄，資料欄名稱識別符所需的空間為 1 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));
```

在此範例中，我們會在將資料列寫入資料表時計算資料的大小，如下列陳述式所示：

```
INSERT INTO mykeyspace.mytable (pk_col1, pk_col2, ck_col1, ck_col2, reg_col1)
values(1,2,3,4,5);
```

若要估計此寫入操作所需的總位元組數，您可以使用下列步驟。

1. 透過新增資料欄中存放之資料類型的位元組和中繼資料位元組，來計算分割區索引鍵資料欄的大小。對所有分割區索引鍵資料欄重複此操作。

- a. 計算分割區索引鍵 (pk\_col1) 第一欄的大小：

```
(2 bytes for the integer data type) x 2 + 1 byte for the column id + 3 bytes
for partition key metadata = 8 bytes
```

- b. 計算分割區索引鍵 (pk\_col2) 第二欄的大小：

```
(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總預估大小：

```
8 bytes + 8 bytes = 16 bytes for the partition key columns
```

2. 透過新增存放在資料欄中的資料類型位元組和中繼資料位元組，來計算叢集資料欄的大小。對所有叢集資料欄重複此操作。

- a. 計算叢集資料欄 (ck\_col1) 第一欄的大小：

```
(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. 計算叢集資料欄 (ck\_col2) 第二欄的大小：

```
(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. 新增兩個資料欄，以取得叢集資料欄的總預估大小：

6 bytes + 6 bytes = 12 bytes for the clustering columns

3. 新增一般資料欄的大小。在此範例中，我們只有一個資料欄存放單一數字整數，這需要 2 個位元組，資料欄 ID 為 1 個位元組。
4. 最後，若要取得總編碼資料列大小，請為所有資料欄加總位元組。若要估計儲存體的計費大小，請為資料列中繼資料新增額外的 100 個位元組：

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.

若要了解如何使用 Amazon CloudWatch 監控無伺服器資源，請參閱 [the section called “使用 CloudWatch 進行監控”](#)。

## 預估 Amazon Keyspaces 中讀取和寫入輸送量的容量耗用

當您在 Amazon Keyspaces 中讀取或寫入資料時，查詢使用的讀取/寫入請求單位 (RRUs/WRUs) 或讀取/寫入容量單位 (RCUs/WCUs) 數量取決於 Amazon Keyspaces 執行查詢所需的總資料量。在某些情況下，傳回給用戶端的資料可以是 Amazon Keyspaces 必須讀取才能處理查詢的資料子集。對於條件式寫入，即使條件式檢查失敗，Amazon Keyspaces 也會使用寫入容量。

若要估計為請求處理的資料總量，您必須考慮資料列的編碼大小和資料列總數。本主題涵蓋一些常見案例和存取模式的範例，以顯示 Amazon Keyspaces 如何處理查詢，以及其如何影響容量消耗。您可以依照範例來估計資料表的容量需求，並使用 Amazon CloudWatch 來觀察這些使用案例的讀取和寫入容量耗用。

如需如何計算 Amazon Keyspaces 中資料列編碼大小的資訊，請參閱 [the section called “估算資料列大小”](#)。

### 主題

- [估計 Amazon Keyspaces 中範圍查詢的容量消耗](#)
- [預估限制查詢的讀取容量耗用量](#)
- [估算資料表掃描的讀取容量耗用量](#)
- [預估 Amazon Keyspaces 中輕量型交易的容量消耗](#)
- [預估 Amazon Keyspaces 中靜態資料欄的容量耗用](#)
- [Amazon Keyspaces 中多區域資料表的估計和佈建容量](#)

- [使用 Amazon Keyspaces 中的 Amazon CloudWatch 估計讀取和寫入容量消耗](#)

## 估計 Amazon Keyspaces 中範圍查詢的容量消耗

若要查看範圍查詢的讀取容量耗用，我們使用下列使用隨需容量模式的範例資料表。

```
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>
```

現在在此資料表上執行下列查詢。

```
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;
```

您從查詢收到下列結果集，Amazon Keyspaces 執行的讀取操作會在LOCAL\_QUORUM一致性模式中使用 2 RRUs。

```
pk1 | pk2 | pk3 | ck1 | ck2 | ck3 | value
-----+-----+-----+-----+-----+-----+-----
a | b | 1 | a | b | 60 | value_1
```

Amazon Keyspaces 會使用 2 RRUs 來評估具有值的資料列ck3=70，ck3=60以及處理查詢。不過，Amazon Keyspaces 只會傳回查詢中指定WHERE條件為 true 的資料列，也就是值為 的資料列ck3=60。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，在此情況下為 ck3 = 70，但不會傳回結果中的該資料列。讀取容量消耗是根據處理查詢時讀取的資料，而非傳回的資料。

## 預估限制查詢的讀取容量耗用量

處理使用 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資料列。如果 Amazon Keyspaces 找不到符合第一頁LIMIT上值的足夠相符資料，則可能需要一或多個分頁呼叫。若要在下一頁繼續讀取，您可以使用分頁字符。預設頁面大小為 1MB。若要在使用 LIMIT子句時減少讀取容量，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”](#)。

如需範例，讓我們來看看下列查詢。

```
SELECT * FROM my_table WHERE partition_key=1234 LIMIT 1;
```

如果您未設定頁面大小，Amazon Keyspaces 會讀取 1MB 的資料，即使只傳回 1 列給您。若要讓 Amazon Keyspaces 只讀取一行，您可以將此查詢的頁面大小設定為 1。在此情況下，如果根據 Time-to-live 設定或用戶端時間戳記沒有過期的資料列，Amazon Keyspaces 只會讀取一行。

PAGE SIZE 參數會決定 Amazon Keyspaces 從磁碟掃描每個請求的資料列數，而不是 Amazon Keyspaces 傳回用戶端的資料列數。Amazon Keyspaces 會套用您提供的篩選條件，例如非金鑰資料欄上的不等式，或掃描磁碟上的資料 LIMIT 之後的。如果您未明確設定 PAGE SIZE，Amazon Keyspaces 會在套用篩選條件之前讀取最多 1MB 的資料。例如，如果您使用 LIMIT 1 而不指定 PAGE SIZE，Amazon Keyspaces 可以在套用限制子句並僅傳回單一資料列之前，從磁碟讀取數千個資料列。

為了避免過度讀取，請減少 PAGE SIZE，以減少 Amazon Keyspaces 掃描每個擷取的資料列數。例如，如果您在查詢 LIMIT 5 中定義，請將 PAGE SIZE 設定為介於 5 - 10 之間的值，以便 Amazon Keyspaces 在每個分頁呼叫上只會掃描 5 - 10 列。您可以修改此數字以減少擷取的數量。對於大於頁面大小的限制，Amazon Keyspaces 會以分頁狀態維持總結果計數。在 10,000 個資料列 LIMIT 的中，Amazon Keyspaces 可以在兩頁各 5,000 個資料列中擷取這些結果。1MB 限制是任何頁面大小集的上限。

## 估算資料表掃描的讀取容量耗用量

導致完整資料表掃描的查詢，例如使用 ALLOW FILTERING 選項的查詢，是另一個處理比其傳回的結果更多讀取的查詢範例。讀取容量消耗是根據資料讀取，而不是傳回的資料。

對於資料表掃描範例，我們在隨需容量模式中使用以下範例資料表。

```
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 預設會在隨需容量模式中建立具有四個分割區的資料表。在此範例資料表中，所有資料都存放在一個分割區中，其餘三個分割區是空的。

現在請在 資料表上執行下列查詢。

```
SELECT * from amazon_keyspaces.example_table_2;
```

此查詢會產生資料表掃描操作，其中 Amazon Keyspaces 會掃描資料表的所有四個分割區，並在 LOCAL\_QUORUM 一致性模式中耗用 6 RRUs。首先，Amazon Keyspaces 會使用 3 RRUs 來讀取具有的三列 `pk='pk'`。然後，Amazon Keyspaces 會使用額外的 3 RRUs 來掃描資料表的三個空白分割區。由於此查詢會導致資料表掃描，Amazon Keyspaces 會掃描資料表中的所有分割區，包括不含資料的分割區。

## 預估 Amazon Keyspaces 中輕量型交易的容量消耗

輕量型交易 (LWT) 可讓您針對資料表資料執行條件式寫入操作。條件式更新操作在根據評估目前狀態的條件插入、更新和刪除記錄時非常有用。

在 Amazon Keyspaces 中，所有寫入操作都需要 LOCAL\_QUORUM 一致性，而且使用 LWTs 無需額外付費。LWTs 的差異在於，當 LWT 條件檢查導致時 FALSE，Amazon Keyspaces 會使用寫入容量單位 WCUs) 或寫入請求單位 WRUs)。使用的 WCUs/WRUs 數量取決於資料列的大小。

例如，如果資料列大小為 2 KB，則失敗的條件式寫入會耗用兩個 WCUs/WRUs。如果資料列目前不存在於資料表中，則操作會耗用一個 WCUs/WRUs。

若要判斷導致條件檢查失敗的請求數量，您可以在 CloudWatch 中監控 ConditionalCheckFailed 指標。

### 預估存留時間 (TTL) 資料表的 LWT 成本

對於使用不使用用戶端時間戳記的 TTL 設定的資料表，LWTs 可能需要額外的讀取容量單位 (RCUs) 或讀取請求單位 (RRUs)。在 中使用 IF EXISTS 或 IF NOT EXISTS 關鍵字條件檢查結果時 FALSE，會耗用下列容量單位：

- RCUs/RRUs 如果資料列存在，使用的 RCUs/RRUs 會根據現有資料列的大小而定。
- RCUs/RRUs 如果資料列不存在，則會使用單一 RCU/RRU。

如果評估的條件導致寫入操作成功，WCUs/WRUs 會根據新資料列的大小耗用。

## 預估 Amazon Keyspaces 中靜態資料欄的容量耗用

在具有叢集資料欄的 Amazon Keyspaces 資料表中，您可以使用 STATIC 關鍵字來建立靜態資料欄。儲存在靜態資料欄中的值會在邏輯分割區中的所有資料列之間共用。當您更新此欄的值時，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欄時計算資料的編碼大小。此程序與將資料寫入資料至資料列非靜態資料欄的程序分開處理。除了靜態資料的大小配額之外，靜態資料欄上的讀取和寫入操作也會獨立影響資

料表的計量和輸送量容量。如需使用靜態資料欄和分頁範圍讀取結果時 Apache Cassandra 的功能差異，請參閱 [the section called “分頁”](#)。

## 主題

- [計算 Amazon Keyspaces 中每個邏輯分割區的靜態資料欄大小](#)
- [預估 Amazon Keyspaces 中靜態資料的讀取/寫入操作的容量輸送量需求](#)

## 計算 Amazon Keyspaces 中每個邏輯分割區的靜態資料欄大小

本節提供如何估計 Amazon Keyspaces 中靜態資料欄編碼大小的詳細資訊。當您計算帳單和配額用量時，會使用編碼大小。當您計算資料表的佈建輸送量容量需求時，也應該使用編碼的大小。若要計算 Amazon Keyspaces 中靜態資料欄的編碼大小，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欄最多需要 3 個位元組的中繼資料。這些中繼資料位元組會計入每個分割區 1 MB 的靜態資料大小配額。計算靜態資料的大小時，您應該假設每個分割區索引鍵資料欄使用完整 3 個位元組的中繼資料。
- 根據資料類型使用靜態資料欄資料值的原始大小。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。
- 叢集資料欄和一般、非主索引鍵資料欄不會計入靜態資料的大小。若要了解如何預估資料列中非靜態資料的大小，請參閱 [the section called “估算資料列大小”](#)。

靜態資料欄的總編碼大小是根據下列公式：

```
partition key columns + static columns + metadata = total encoded size of static data
```

請考慮下列資料表範例，其中所有資料欄的類型皆為整數。資料表有兩個分割區索引鍵資料欄、兩個叢集資料欄、一個規則資料欄和一個靜態資料欄。

```
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));
```

在此範例中，我們會計算下列陳述式的靜態資料大小：

```
INSERT INTO mykeyspace.mytable (pk_col1, pk_col2, static_col1) values(1,2,6);
```

若要估計此寫入操作所需的總位元組數，您可以使用下列步驟。

1. 透過新增存放在資料欄中的資料類型位元組和中繼資料位元組，來計算分割區索引鍵資料欄的大小。對所有分割區索引鍵資料欄重複此操作。

- a. 計算分割區索引鍵 (pk\_col1) 第一欄的大小：

```
4 bytes for the integer data type + 3 bytes for partition key metadata = 7 bytes
```

- b. 計算分割區索引鍵 (pk\_col2) 第二欄的大小：

```
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總預估大小：

```
7 bytes + 7 bytes = 14 bytes for the partition key columns
```

2. 新增靜態資料欄的大小。在此範例中，我們只有一個靜態資料欄可存放整數（需要 4 個位元組）。
3. 最後，若要取得靜態資料欄資料的總編碼大小，請為主索引鍵資料欄和靜態資料欄新增位元組，並為中繼資料新增額外的 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。若要估計寫入操作的總大小，您必須先計算非靜態資料更新的大小。然後計算資料列更新的大小，如的範例所示[the section called “估算資料列大小”](#)，並新增結果。

在這種情況下，您可以寫入總計 2 MB，1 MB 是資料列大小配額上限，而 1 MB 是每個邏輯分割區靜態資料大小上限的配額。

若要計算相同陳述式中靜態和非靜態資料更新的總大小，您可以使用下列公式：

```
(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
```



請考慮下列資料表範例，其中所有資料欄的類型皆為整數。資料表有兩個分割區索引鍵資料欄、兩個叢集資料欄、一個規則資料欄和一個靜態資料欄。

```
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));
```

在此範例中，我們在將資料列寫入資料表時計算資料的大小，如下列陳述式所示：

```
INSERT INTO mykeyspace.mytable (pk_col1, pk_col2, ck_col1, ck_col2, reg_col1,
static_col1) values(2,3,4,5,6,7);
```

若要估計此寫入操作所需的總位元組數，您可以使用下列步驟。

1. 計算靜態資料的總編碼大小，如先前所示。在此範例中，它是 122 個位元組。
2. 遵循 [中的步驟](#)，根據非靜態資料的更新，新增資料列的總編碼大小大小 [the section called “估算資料列大小”](#)。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.
```

## 預估 Amazon Keyspaces 中靜態資料的讀取/寫入操作的容量輸送量需求

靜態資料與 Cassandra 中的邏輯分割區相關聯，而不是與個別資料列相關聯。透過跨越多個實體儲存分割區，Amazon Keyspaces 中的邏輯分割區幾乎可以不受大小限制。因此，Amazon Keyspaces 會分別測量靜態和非靜態資料的寫入操作。此外，包含靜態和非靜態資料的寫入需要額外的基礎操作才能提供資料一致性。

如果您同時執行靜態和非靜態資料的混合寫入操作，這會產生兩個不同的寫入操作，一個用於非靜態資料，另一個用於靜態資料。這同時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訊，說明如何在計算具有靜態資料欄之 Amazon Keyspaces 中資料表的佈建輸送量需求時，預估所需的讀取容量單位 (RCUs) 和寫入容量單位 (WCUs)。您可以使用下列公式，預估資料表處理包含靜態和非靜態資料的寫入所需的容量：

```
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 KBs 的資料，且每次寫入包含 25.5 KBs 的非靜態資料和 1.5 KBs 的靜態資料，則資料表需要 56 個 WCUs (2 x 26 WCUs + 2 x 2 WCUs)。



Amazon Keyspaces 會測量靜態和非靜態資料的讀取，與多個資料列的讀取相同。因此，相同操作中讀取靜態和非靜態資料的價格，取決於為執行讀取而處理的資料彙總大小。

若要了解如何使用 Amazon CloudWatch 監控無伺服器資源，請參閱 [the section called “使用 CloudWatch 進行監控”](#)。

## Amazon Keyspaces 中多區域資料表的估計和佈建容量

您可以透過兩種方式之一設定多區域資料表的輸送量容量：

- 隨需容量模式，以寫入請求單位 (WRUs) 測量
- 具有自動擴展的佈建容量模式，以寫入容量單位 (WCUs) 測量

您可以使用佈建容量模式搭配自動擴展或隨需容量模式，以協助確保多區域資料表有足夠的容量來執行複寫寫入至所有 AWS 區域。

### Note

變更其中一個區域中資料表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需模式。使用隨需模式時，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 會在工作負載上升或下降到先前達到的任何流量層級時，立即滿足您的工作負載。如果工作負載的流量達到新的峰值，Amazon Keyspaces 會快速調整以適應工作負載。

如果您選擇資料表的佈建容量模式，則必須設定應用程式每秒所需的讀取容量單位 (RCUs) 和寫入容量單位 (WCUs) 數量。

若要規劃多區域資料表的輸送量容量需求，您應該先預估每個區域每秒所需的 WCUs 數量。然後，從複寫資料表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，因為在某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，則會發生容量例外狀況。此外，區域間複寫等待時間也會增加。

例如，如果您有一個多區域資料表，其中在美國東部（維吉尼亞北部）預期每秒 5 次寫入、美國東部（俄亥俄）預期每秒 10 次寫入，而歐洲（愛爾蘭）預期該資料表在每個區域中使用 20 個 WCUs：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，您需要

為每個資料表的複本佈建 20 WCUs。您可以使用 Amazon CloudWatch 監控資料表的容量消耗。如需詳細資訊，請參閱[the section called “使用 CloudWatch 進行監控”](#)。

每次寫入都會以 1 個 WCU 計費，因此您會在此範例中看到總共 60 WCUs 計費。如需定價的詳細資訊，請參閱 [Amazon Keyspaces \(適用於 Apache Cassandra\) 定價](#)。

如需使用 Amazon Keyspaces 自動擴展佈建容量的詳細資訊，請參閱 [the section called “使用自動擴展管理輸送量容量”](#)。

#### Note

如果資料表在具有自動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自動擴展設定內浮動。

## 使用 Amazon Keyspaces 中的 Amazon CloudWatch 估計讀取和寫入容量消耗

若要估計和監控讀取和寫入容量消耗，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”](#)。

若要使用 CloudWatch 監控特定陳述式使用的讀取和寫入容量單位，您可以遵循下列步驟。

1. 使用範例資料建立新的資料表
2. 設定資料表的 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. 執行 CQL 陳述式，例如使用 ALLOW FILTERING 選項，並在儀表板中檢查用於完整資料表掃描的讀取容量單位。

## 在 Amazon Keyspaces 中設定讀取/寫入容量模式

Amazon Keyspaces 有兩個讀取/寫入容量模式，用於處理資料表上的讀取和寫入：

- 隨需（預設）
- 佈建

您所選擇的讀取/寫入容量模式會控制讀寫輸送量的收費方式，以及資料表輸送容量的管理方式。

## 主題

- [設定隨需容量模式](#)
- [設定佈建容量模式](#)
- [在 Amazon Keyspaces 中檢視資料表的容量模式](#)
- [變更 Amazon Keyspaces 中資料表的容量模式](#)
- [為 Amazon Keyspaces 中的資料表設定預熱](#)

## 設定隨需容量模式

Amazon Keyspaces (適用於 Apache Cassandra) 隨需容量模式是一種靈活的計費選項，無需規劃容量即可每秒處理數千個請求。此選項提供讀取和寫入請求pay-per-request定價，因此您只需支付使用量的費用。

當您選擇隨需模式時，Amazon Keyspaces 可以立即將資料表的輸送量容量擴展到任何先前達到的流量層級，然後在應用程式流量減少時返回。如果工作負載的流量層級達到新的峰值，服務會快速調整以增加資料表的輸送量容量。您可以為新的和現有的資料表啟用隨需容量模式。

如果以下任何一項成立，隨需模式是不錯的選擇：

- 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。

若要開始使用隨需模式，您可以使用主控台或幾行 Cassandra 查詢語言 (CQL) 程式碼建立新的資料表，或更新現有的資料表以使用隨需容量模式。如需詳細資訊，請參閱[the section called “表格”](#)。

## 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

使用隨需容量模式資料表，您不需要指定預期應用程式預先使用的讀取和寫入輸送量。Amazon Keyspaces 會根據讀取請求單位 (RRUs) 和寫入請求單位 (WRUs)，向您收取您在資料表上執行的讀取和寫入費用。

- 一個 RRU 代表大小上限為 4 KB 的資料列一個 LOCAL\_QUORUM 讀取請求或兩個 LOCAL\_ONE 讀取請求。如果您需要讀取大於 4 KB 的資料列，讀取操作會使用其他 RRUs。所需的 RRUs 總數取決於資料列大小，以及您想要使用或 LOCAL\_QUORUM/LOCAL\_ONE 讀取一致性。例如，讀取 8 KB 的資料列需要 2 個使用 LOCAL\_QUORUM 讀取一致性 RRUs，如果您選擇 LOCAL\_ONE 讀取一致性，則需要 1 個 RRU。
- 一個 WRU 代表大小上限為 1 KB 的資料列一個寫入。所有寫入都使用 LOCAL\_QUORUM 一致性，而且使用輕量型交易 (LWTs) 無需額外付費。如果您需要寫入大於 1 KB 的資料列，則寫入操作會使用額外的 WRUs。所需的 WRUs 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 WRUs 來執行一個寫入請求。

如需支援的一致性層級的相關資訊，請參閱 [the section called “支援的 Cassandra 一致性層級”](#)。

## 峰值流量與擴展屬性

使用隨需容量模式的 Amazon Keyspaces 資料表會自動適應應用程式的流量。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，您應用程式的流量模式可能介於每秒 5,000 到 10,000 次 LOCAL\_QUORUM 讀取之間，其中每秒 10,000 次讀取是先前的流量峰值。

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，該峰值會成為您先前的新峰值，讓後續流量達到每秒 40,000 次讀取。

如果您在資料表上需要超過先前峰值的兩倍，Amazon Keyspaces 會在流量增加時自動配置更多容量。這有助於確保您的資料表有足夠的輸送量容量來處理其他請求。不過，如果您在 30 分鐘內超過先前峰值的兩倍，您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 個強式一致讀取之間有所不同，其中每秒 20,000 個讀取是先前達到的流量峰值。在此情況下，服務建議您將流量成長至少間隔 30 分鐘，然後每秒最多 40,000 次讀取。

若要了解如何預估資料表的讀取和寫入容量耗用，請參閱 [the section called “估計容量耗用量”](#)。

若要進一步了解帳戶的預設配額以及如何增加這些配額，請參閱 [配額](#)。

## 隨需容量模式的初始輸送量

如果您第一次建立啟用隨需容量模式的新資料表，或將現有資料表切換為隨需容量模式，則資料表會有下列先前的峰值設定，即使先前尚未使用隨需容量模式為流量提供服務：

- 使用隨需容量模式新建立的資料表：先前的峰值為 2,000 WRUs 和 6,000 RRUs。您可以立即驅動高達先前峰值的兩倍。這樣做可讓新建立的隨需資料表提供高達 4,000 WRUs 和 12,000 RRUs。
- 現有資料表切換到隨需容量模式：先前的峰值是為資料表佈建的先前 WCUs 和 RCUs 的一半，或為具有隨需容量模式的新建立資料表設定，以較高者為準。

## 設定佈建容量模式

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。這可協助您管理 Amazon Keyspaces 用量，以維持或低於定義的請求率，以維持可預測性。若要進一步了解佈建輸送量的自動擴展，請參閱 [the section called “使用自動擴展管理輸送量容量”](#)。

如果符合下列任一條件，佈建的輸送量容量模式是不錯的選擇：

- 您有可預期的應用程式流量。
- 您執行的應用程式有一致或逐漸增多的流量。
- 您可以預測容量需求。

## 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資料表，您可以指定讀取容量單位 (RCUs) 和寫入容量單位 (WCUs) 的輸送量容量：

- 一個 RCU 代表每秒一個 LOCAL\_QUORUM 讀取，或每秒兩個 LOCAL\_ONE 讀取，資料列大小上限為 4 KB。如果您需要讀取大於 4 KB 的資料列，則讀取操作會使用其他 RCUs。

所需的 RCUs 總數取決於資料列大小，以及您想要 LOCAL\_QUORUM 還是 LOCAL\_ONE 讀取。例如，如果您的資料列大小為 8 KB，則需要 2 RCUs 來維持每秒一個 LOCAL\_QUORUM 讀取，如果您選擇 LOCAL\_ONE 讀取，則需要 1 個 RCU。

- 一個 WCU 代表最大 1 KB 的資料列每秒一個寫入。所有寫入都使用 LOCAL\_QUORUM 一致性，而且使用輕量型交易 (LWTs) 無需額外付費。如果您需要寫入大於 1 KB 的資料列，則寫入操作會使用其他 WCUs。

所需的 WCUs 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 WCUs 來維持每秒一個寫入請求。如需如何預估資料表讀取和寫入容量耗用的詳細資訊，請參閱 [the section called “估計容量耗用量”](#)。

如果您的應用程式讀取或寫入較大的資料列（最大 Amazon Keyspaces 的資料列大小上限為 1 MB），則會耗用更多容量單位。若要進一步了解如何估計資料列大小，請參閱 [the section called “估算資料列大小”](#)。例如，假設您建立具有 6 個 RCUs 和 6 個 WCUs 佈建資料表。透過這些設定，您的應用程式可執行下列項目：

- 執行每秒高達 24 KB 的 LOCAL\_QUORUM 讀取 (4 KB × 6 RCUs)。
- 執行每秒高達 48 KB 的 LOCAL\_ONE 讀取（兩倍的讀取輸送量）。
- 每秒寫入最多 6 KB (1 KB × 6 WCUs)。

佈建輸送量是應用程式可以從資料表取用的最大輸送量。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。

例如，沒有足夠輸送量容量的讀取請求會失敗，但有 ReadTimeout 例外狀況，並會發佈到 ReadThrottleEvents 指標。沒有足夠輸送量的寫入請求會失敗，但有 WriteTimeout 例外狀況，並會發佈到 WriteThrottleEvents 指標。

您可以使用 Amazon CloudWatch 來監控佈建和實際輸送量指標，以及容量不足的事件。如需這些指標的詳細資訊，請參閱 [the section called “指標與維度”](#)。

#### Note

由於容量不足而重複的錯誤可能會導致用戶端驅動程式特定的例外狀況，例如 DataStax Java 驅動程式因而失敗 NoHostAvailableException。

若要變更資料表的輸送量容量設定，您可以使用 AWS 管理主控台 或使用 CQL 的 ALTER TABLE 陳述式，如需詳細資訊，請參閱 [the section called “ALTER TABLE”](#)。

若要進一步了解您帳戶的預設配額以及如何增加這些配額，請參閱 [配額](#)。

## 在 Amazon Keyspaces 中檢視資料表的容量模式

您可以在 Amazon Keyspaces 系統金鑰空間中查詢系統資料表，以檢閱資料表的容量模式資訊。您也可以查看資料表是否使用隨需或佈建的輸送量容量模式。如果資料表設定為佈建的輸送量容量模式，您可以查看為資料表佈建的輸送量容量。

您也可以使用 AWS CLI 來檢視資料表的容量模式。

若要變更資料表的佈建輸送量，請參閱 [the section called “變更容量模式”](#)。

## Cassandra Query Language (CQL)

### 範例

```
SELECT * from system_schema_mcs.tables where keyspace_name = 'mykeyspace' and
table_name = 'mytable';
```

使用隨需容量模式設定的資料表會傳回下列項目。

```
{
 "capacity_mode":{
 "last_update_to_pay_per_request_timestamp":"1579551547603",
 "throughput_mode":"PAY_PER_REQUEST"
 }
}
```

使用佈建輸送量容量模式設定的資料表會傳回下列項目。

```
{
 "capacity_mode":{
 "last_update_to_pay_per_request_timestamp":"1579048006000",
 "read_capacity_units":"5000",
 "throughput_mode":"PROVISIONED",
 "write_capacity_units":"6000"
 }
}
```

`last_update_to_pay_per_request_timestamp` 值的測量單位為毫秒。

### CLI

使用 檢視資料表的輸送量容量模式 AWS CLI

```
aws keyspaces get-table --keyspace-name myKeyspace --table-name myTable
```

對於佈建容量模式中的資料表，命令的輸出可能看起來像這樣。

```
"capacitySpecification": {
 "throughputMode": "PROVISIONED",
 "readCapacityUnits": 4000,
 "writeCapacityUnits": 2000
}
```



隨需模式中資料表的輸出如下所示。

```
"capacitySpecification": {
 "throughputMode": "PAY_PER_REQUEST",
 "lastUpdateToPayPerRequestTimestamp": "2024-10-03T10:48:19.092000+00:00"
}
```

## 變更 Amazon Keyspaces 中資料表的容量模式

當您將資料表從佈建容量模式切換到隨需容量模式時，Amazon Keyspaces 會對資料表和分割區的結構進行數項變更。此程序需要幾分鐘的時間。在切換期間，您的資料表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，您的資料表會提供與資料表設定為隨需容量模式時先前達到的尖峰一致的輸送量。

當您切換容量模式時，適用下列等待期間：

- 您可以隨時將新建立的隨需模式資料表切換為佈建容量模式。不過，您只能在資料表建立時間戳記的 24 小時後將其切換回隨需模式。
- 您可以隨時將隨需模式中的現有資料表切換為佈建容量模式。不過，您只能在 24 小時期間內將容量模式從佈建切換到隨需。

### Cassandra Query Language (CQL)

使用 CQL 變更資料表的輸送量容量模式

- 若要將資料表的容量模式變更為 PROVISIONED，您必須根據工作負載預期的峰值來設定讀取容量和寫入容量單位。以下陳述式為範例。您也可以執行此陳述式來調整資料表的讀取容量或寫入容量單位。

```
ALTER TABLE catalog.book_awards WITH CUSTOM_PROPERTIES={'capacity_mode':
{'throughput_mode': 'PROVISIONED', 'read_capacity_units': 6000,
'write_capacity_units': 3000}};
```

若要使用自動調整規模設定佈建容量模式，請參閱 [the section called “在現有資料表上設定自動擴展”](#)。



- 若要將資料表的容量模式變更為隨需模式，請將輸送量模式設定為 PAY\_PER\_REQUEST。下列陳述式為範例。

```
ALTER TABLE catalog.book_awards WITH CUSTOM_PROPERTIES={'capacity_mode':
{'throughput_mode': 'PAY_PER_REQUEST'}};
```

- 您可以使用下列陳述式來確認資料表的容量模式。

```
SELECT * from system_schema_mcs.tables where keyspace_name = 'catalog' and
table_name = 'book_awards';
```

使用隨需容量模式設定的資料表會傳回下列項目。

```
{
 "capacity_mode":{
 "last_update_to_pay_per_request_timestamp":"1727952499092",
 "throughput_mode":"PAY_PER_REQUEST"
 }
}
```

last\_update\_to\_pay\_per\_request\_timestamp 值的測量單位為毫秒。

## CLI

使用 變更資料表的輸送量容量模式 AWS CLI

- 若要將資料表的容量模式變更為 PROVISIONED，您必須根據工作負載的預期尖峰值來設定讀取容量和寫入容量單位。下列命令是此範例。您也可以執行此命令來調整 資料表的讀取容量或寫入容量單位。

```
aws keyspaces update-table --keyspace-name catalog --table-name book_awards
 \--capacity-specification
 throughputMode=PROVISIONED,readCapacityUnits=6000,writeCapacityUnits=3000
```

若要使用自動調整規模設定佈建容量模式，請參閱 [the section called “在現有資料表上設定自動擴展”](#)。

- 若要將資料表的容量模式變更為隨需模式，請將輸送量模式設定為 PAY\_PER\_REQUEST。下列陳述式為範例。

```
aws keyspaces update-table --keyspace-name catalog --table-name book_awards
 \--capacity-specification
 throughputMode=PAY_PER_REQUEST
```

3. 您可以使用下列命令來檢閱為資料表設定的容量模式。

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

隨需模式中資料表的輸出如下所示。

```
"capacitySpecification": {
 "throughputMode": "PAY_PER_REQUEST",
 "lastUpdateToPayPerRequestTimestamp": "2024-10-03T10:48:19.092000+00:00"
}
```

## 為 Amazon Keyspaces 中的資料表設定預熱

Amazon Keyspaces 會根據隨需或佈建的輸送量自動擴展儲存分割區，但對於新資料表或突然的輸送量峰值，配置所需的儲存分割區可能需要更長的時間。若要確保新的或現有的資料表有足夠的容量來支援預期的尖峰輸送量，您可以手動設定特定的暖輸送量值來預先暖機資料表。

暖輸送量是指 Amazon Keyspaces 資料表可立即支援的讀取和寫入操作數量。這些值預設為適用於所有新的和現有的資料表。如果您使用隨需模式，或更新佈建輸送量，Amazon Keyspaces 會確保您的應用程式能夠立即發出高達這些值的請求。

Amazon Keyspaces 會在用量增加時自動調整暖輸送量值。若要調整即將到來的尖峰事件的輸送量容量，例如當您從另一個資料庫遷移資料時，可能需要在短時間內載入 TB 的資料時，您可以手動增加資料表的暖輸送量值。這適用於計劃的尖峰事件，其中請求率可能會增加 10 倍、100 倍或更多。首先，評估目前的暖輸送量是否足以處理預期的流量。然後，如果您需要為計劃的尖峰工作負載預先暖機資料表，您可以手動增加暖輸送量值，而無需變更輸送量設定或[容量模式](#)。

您可以預熱讀取操作、寫入操作或兩者的資料表。您可以為新的和現有的單一區域資料表和多區域資料表增加此值，而且您設定的暖輸送量設定會自動套用至多區域資料表的所有複本。您可以隨時預先暖機的 Amazon Keyspaces 資料表數量沒有限制。完成預熱的時間取決於您設定的值和資料表的大小。您可以同時提交預熱請求，這些請求不會干擾任何資料表操作。您可以將資料表預熱到該區域中帳戶的資料表配額限制。使用 [Service Quotas 主控台](#) 來檢查您目前的配額，並視需要增加配額。

Amazon Keyspaces 根據隨需用量或佈建容量調整的暖輸送量值，預設適用於所有資料表，無需額外費用。不過，如果您手動將預設暖輸送量值增加到尖峰流量事件的預暖資料表，則需支付額外費用。如需詳細資訊，請參閱 [Amazon Keyspaces 定價](#)。

以下是預暖 Amazon Keyspaces 資料表時可能會考慮的一些不同案例和最佳實務。

## 暖輸送量與非均勻存取模式

資料表的暖輸送量可能是每秒 30,000 個讀取單位和每秒 10,000 個寫入單位，但在達到這些值之前，您仍然可能會在讀取或寫入時遇到超過事件的容量。這通常是因為存在熱分割區造成。雖然 Amazon Keyspaces 可以繼續擴展以支援幾乎無限制的輸送量，但每個分割區每秒僅限 1,000 個寫入單位和每秒 3,000 個讀取單位。如果您的應用程式對資料表的一小部分分割區驅動太多流量，即使達到資料表的暖輸送量值，容量也會超過事件。我們建議您遵循 [Amazon Keyspaces 最佳實務](#)，以確保無縫的可擴展性並避免熱分割區。

## 佈建資料表的暖輸送量

假設佈建資料表的暖輸送量為每秒 30,000 個讀取單位和每秒 10,000 個寫入單位，但目前佈建輸送量為 4,000 RCUs 和 8,000 WCUs。您可以透過更新佈建的輸送量設定，立即將資料表的佈建輸送量擴展到 30,000 RCUs 或 10,000 個 WCUs。當您增加佈建輸送量超過這些值時，暖輸送量會自動調整為新的更高值，因為您已建立新的尖峰輸送量。例如，如果您將佈建的輸送量設定為 50,000 個 RCU，暖輸送量會提高到每秒 50,000 個讀取單位。

```
"ProvisionedThroughput":
{
 "ReadCapacityUnits": 4000,
 "WriteCapacityUnits": 8000
}
"WarmThroughput":
{
 "ReadUnitsPerSecond": 30000,
 "WriteUnitsPerSecond": 10000
}
```

## 隨需資料表的暖輸送量

新建立的隨需資料表初始暖輸送量為每秒 12,000 個讀取單位與每秒 4,000 個寫入單位。資料表可立即支援持續流量，最高可達上述層級。當您的請求超過每秒 12,000 個讀取單位或每秒 4,000 個寫入單位時，暖輸送量會自動調整為較高的值。

```
"WarmThroughput":
```

```
{
 "ReadUnitsPerSecond": 12000,
 "WriteUnitsPerSecond": 4000
}
```

## 預熱 Amazon Keyspaces 資料表的最佳實務

為 Amazon Keyspaces 資料表實作預熱時，請遵循下列最佳實務：

### 準確估計所需的容量

由於預熱會產生一次性成本，請根據預期的工作負載仔細計算所需的輸送量，以避免過度佈建。

### 考慮資料表的結構描述

資料列較大的資料表可能需要更多分割區才能達到相同的輸送量。估算預熱前需求時，請考量平均資料列大小。

### 監控資料表效能

預熱後，請使用 CloudWatch 指標來驗證您的資料表是否如預期處理負載。如需詳細資訊，請參閱 [the section called “使用 CloudWatch 監控預暖”](#)。

### 管理配額

如果您的應用程式需要高於預設配額允許 (40,000 個 RCUs/WCUs 或 2,000 個分割區) 的輸送量，則請求配額會在高流量事件之前大幅增加。若要請求提升配額，您可以使用 [Service Quotas 主控台](#)。

### 最佳化成本

對於暫時高流量事件，請考慮使用預熱，而不是切換到具有高容量的佈建模式，因為它在短期事件中可能更具成本效益。如需定價的詳細資訊，請參閱 [Amazon Keyspaces 定價](#)。

#### Note

在測試階段監控應用程式的效能指標，以驗證預暖組態是否充分支援工作負載需求。

## 主題

- [建立暖輸送量較高的新 Amazon Keyspaces 資料表](#)
- [提高現有 Amazon Keyspaces 資料表的暖輸送量](#)
- [檢視 Amazon Keyspaces 資料表的暖輸送量](#)

- [使用 Amazon CloudWatch 監控預暖資料表的效能](#)

## 建立暖輸送量較高的新 Amazon Keyspaces 資料表

您可以使用主控台、CQL 或 `awscli`，在建立 Amazon Keyspaces 資料表時調整暖輸送量值 AWS CLI。

### Console

如何使用熱輸送量設定建立新的資料表

1. 登入 AWS 管理主控台，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. 在資料表詳細資訊區段的建立資料表頁面上，選取金鑰空間，並提供新資料表的名稱。
4. 在資料欄區段中，建立資料表的結構描述。
5. 在主索引鍵區段中，定義資料表的主索引鍵，然後選取選用的叢集資料欄。
6. 在資料表設定區段中，選擇自訂設定。
7. 繼續讀取/寫入容量設定。
8. 對於容量模式，您可以選擇隨需或佈建。
9. 在資料表的預熱前區段中，您可以視需要增加每秒讀取單位和每秒寫入單位的值，以準備資料表來處理規劃的尖峰事件。

Amazon Keyspaces 根據隨需用量或佈建容量調整的暖輸送量值，預設適用於所有資料表，無需額外費用。請注意，如果您手動增加預設暖傳輸量值來為尖峰流量事件預熱資料表，則需支付額外費用。

10. 視需要設定其他選用的資料表功能。然後選擇建立資料表。

### Cassandra Query Language (CQL)

- 使用下列其中一種方法建立具有暖輸送量的資料表：
  - 對於佈建模式，請建立資料表，並使用下列 CQL 語法指定讀取和寫入的預期尖峰容量：

```
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))
WITH CUSTOM_PROPERTIES = {
 'capacity_mode': {
 'throughput_mode': 'PROVISIONED',
 'read_capacity_units': 20000,
 'write_capacity_units': 10000
 },
 'warm_throughput': {
 'read_units_per_second': 40000,
 'write_units_per_second': 20000
 }
};

```

- 對於隨需模式，請建立資料表，並使用下列 CQL 語法指定預期的讀取和寫入尖峰容量：

```

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))
WITH CUSTOM_PROPERTIES = {
 'capacity_mode': {
 'throughput_mode': 'PAY_PER_REQUEST'
 },
 'warm_throughput': {
 'read_units_per_second': 40000,
 'write_units_per_second': 20000
 }
};

```

若要確認資料表的容量設定，請參閱 [the section called “檢視資料表的暖輸送量”](#)。

## CLI

1. 使用下列其中一種方法，使用 建立具有暖輸送量的資料表 AWS CLI

- 在佈建模式中建立新的資料表，並指定新資料表讀取和寫入的預期尖峰容量值。下列陳述式是此範例。

```
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=book_title,type=text},{name=author,type=text},
{name=publisher,type=text}],partitionKeys=[{name=year},
{name=award}],clusteringKeys=[{name=category,orderBy=ASC},
{name=rank,orderBy=ASC}]' \
--capacity-specification
 throughputMode=PROVISIONED,readCapacityUnits=20000,writeCapacityUnits=10000
\
--warm-throughput-specification
 readUnitsPerSecond=40000,writeUnitsPerSecond=20000
```

- 在隨需模式下建立新的資料表，並指定新資料表的讀取和寫入的預期尖峰容量值。下列陳述式是此範例。

```
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=book_title,type=text},{name=author,type=text},
{name=publisher,type=text}],partitionKeys=[{name=year},
{name=award}],clusteringKeys=[{name=category,orderBy=ASC},
{name=rank,orderBy=ASC}]' \
--warmThroughputSpecification
 readUnitsPerSecond=40000,writeUnitsPerSecond=20000
```

- 命令的輸出會傳回資料表的 ARN，如下列範例所示。

```
{
 "resourceArn": "arn:aws::cassandra:us-east-1:111122223333:/keyspace/catalog/
table/book_awards>"
}
```

若要確認資料表的容量設定，請參閱 [the section called “檢視資料表的暖輸送量”](#)。

## Java

使用適用於 Java 的 開發套件建立新資料表。

- 在佈建模式中建立新的資料表，並指定新資料表讀取和寫入的預期尖峰容量值。下列程式碼範例是此範例。

```
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.*;

public class PreWarmingExample {
 public static void main(String[] args) {
 KeyspacesClient keyspacesClient = KeyspacesClient.builder().build();

 // Define schema
 List<ColumnDefinition> columns = Arrays.asList(
 ColumnDefinition.builder().name("year").type("int").build(),
 ColumnDefinition.builder().name("award").type("text").build(),
 ColumnDefinition.builder().name("rank").type("int").build(),
 ColumnDefinition.builder().name("category").type("text").build(),
 ColumnDefinition.builder().name("book_title").type("text").build(),
 ColumnDefinition.builder().name("author").type("text").build(),
 ColumnDefinition.builder().name("publisher").type("text").build()
);

 List<PartitionKey> partitionKeys = Arrays.asList(
 PartitionKey.builder().name("year").build(),
 PartitionKey.builder().name("award").build()
);

 List<ClusteringKey> clusteringKeys = Arrays.asList(
 ClusteringKey.builder().name("category").orderBy("ASC").build(),
 ClusteringKey.builder().name("rank").orderBy("ASC").build()
);

 SchemaDefinition schema = SchemaDefinition.builder()
 .allColumns(columns)
 .partitionKeys(partitionKeys)
 .clusteringKeys(clusteringKeys)
 .build();

 // Define capacity specification
 CapacitySpecification capacitySpec = CapacitySpecification.builder()
```



```

 .throughputMode(ThroughputMode.PROVISIONED)
 .readCapacityUnits(20000)
 .writeCapacityUnits(10000)
 .build();

 // Define warm throughput specification
 WarmThroughputSpecification warmThroughput =
 WarmThroughputSpecification.builder()
 .readUnitsPerSecond(40000L)
 .writeUnitsPerSecond(20000L)
 .build();

 // Create table with PreWarming
 CreateTableRequest request = CreateTableRequest.builder()
 .keyspaceName("catalog")
 .tableName("book_awards")
 .schemaDefinition(schema)
 .capacitySpecification(capacitySpec)
 .warmThroughputSpecification(warmThroughput)
 .build();

 CreateTableResponse response = keyspacesClient.createTable(request);
 System.out.println("Table created with ARN: " + response.resourceArn());
}
}

```

## 提高現有 Amazon Keyspaces 資料表的暖輸送量

您可以使用主控台、CQL 或 來增加 Amazon Keyspaces 資料表目前的暖輸送量值 AWS CLI。

### Console

如何使用主控台增加資料表的預熱設定

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇資料表，然後選擇您要更新的資料表。
3. 在資料表的容量索引標籤上，繼續為資料表預熱。
4. 在資料表的預熱區段中，選擇編輯。
5. 在編輯資料表預暖頁面上，您可以更新每秒讀取單位和每秒寫入單位的值。

6. 選擇儲存變更。您的資料表正在以指定的預熱前設定進行更新。

## Cassandra Query Language (CQL)

使用 CQL 增加資料表的暖輸送量設定

- 使用 ALTER TABLE 陳述式來增加資料表的暖輸送量。下列陳述式是此範例。

```
ALTER TABLE catalog.book_awards
WITH CUSTOM_PROPERTIES = {
 'warm_throughput': {
 'read_units_per_second': 60000,
 'write_units_per_second': 30000
 }
};
```

若要確認資料表的更新容量設定，請參閱 [the section called “檢視資料表的暖輸送量”](#)。

## CLI

使用 增加資料表的預熱前設定 AWS CLI

- 若要增加資料表的暖通量，您可以使用 update-table 命令。下列陳述式是此範例。

```
aws keyspaces update-table \
--keyspace-name 'catalog' \
--table-name 'book_awards' \
--warmThroughputSpecification readUnitsPerSecond=60000,writeUnitsPerSecond=30000
```

若要確認資料表的更新容量設定，請參閱 [the section called “檢視資料表的暖輸送量”](#)。

## Java

使用適用於 Java 的 SDK 更新資料表的預熱前設定。

- 更新資料表的暖輸送量設定。下列程式碼範例是此範例。

```
import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.*;
```

```
public class UpdatePreWarmingExample {
 public static void main(String[] args) {
 KeyspacesClient keyspacesClient = KeyspacesClient.builder().build();

 // Define new warm throughput specification
 WarmThroughputSpecification warmThroughput =
 WarmThroughputSpecification.builder()
 .readUnitsPerSecond(60000L)
 .writeUnitsPerSecond(30000L)
 .build();

 // Update table with new PreWarming settings
 UpdateTableRequest request = UpdateTableRequest.builder()
 .keyspaceName("catalog")
 .tableName("book_awards")
 .warmThroughputSpecification(warmThroughput)
 .build();

 UpdateTableResponse response = keyspacesClient.updateTable(request);
 System.out.println("Table update requested: " + response.resourceArn());
 }
}
```

## 檢視 Amazon Keyspaces 資料表的暖輸送量

您可以使用主控台、CQL 或 檢視 Amazon Keyspaces 資料表目前的暖輸送量值 AWS CLI。

### Console

如何使用主控台檢視資料表的預熱前設定。

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇資料表，然後選擇您要檢閱的資料表。
3. 在資料表的容量索引標籤上，繼續為資料表預熱。

## Cassandra Query Language (CQL)

### 使用 CQL 檢視資料表的暖通量設定

- 若要檢視資料表的暖輸送量設定，您可以使用下列 CQL 陳述式。

```
SELECT custom_properties
FROM system_schema_mcs.tables
WHERE keyspace_name='catalog' and table_name='book_awards';

// Output:
...
custom_properties

{
 'warm_throughput':
 {
 'read_units_per_second': '40000',
 'write_units_per_second': '20000',
 'status': 'AVAILABLE'
 }
}
...
```

## CLI

### 使用 檢視資料表的暖通量設定 AWS CLI

- 您可以使用 `get-table` 命令檢視資料表的暖輸送量設定，如下列範例所示。

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

以下顯示佈建模式中單一區域資料表的 `get-table` 命令範例輸出。

```
{
 "keyspaceName": "catalog",
 "tableName": "book_awards",
 ... Existing Fields ...,
 "capacitySpecificationSummary": {
```

```

 "throughputMode": "PROVISIONED",
 "readCapacityUnits": 20000,
 "writeCapacityUnits": 10000
 },
 "warmThroughputSpecificationSummary": {
 "readUnitsPerSecond": 40000,
 "writeUnitsPerSecond": 20000,
 "status": "AVAILABLE"
 }
}

```

以下顯示隨需模式下單一區域資料表的範例輸出。

```

{
 "keyspaceName": "catalog",
 "tableName": "book_awards_ondemand",
 ... Existing Fields ...,
 "capacitySpecification": {
 "throughputMode": "PAY_PER_REQUEST"
 },
 "warmThroughputSpecificationSummary": {
 "readUnitsPerSecond": 40000,
 "writeUnitsPerSecond": 20000,
 "status": "AVAILABLE"
 }
}

```

## Java

使用適用於 Java 的 SDK 讀取資料表的預熱前設定。

- 使用 讀取資料表的暖通量值 `get-table`。下列程式碼範例是此範例。

```

import software.amazon.awssdk.services.keyspaces.KeyspacesClient;
import software.amazon.awssdk.services.keyspaces.model.*;

public class GetTableWithPreWarmingExample {
 public static void main(String[] args) {
 KeyspacesClient keyspacesClient = KeyspacesClient.builder().build();

 // Get table details including PreWarming specification
 GetTableRequest request = GetTableRequest.builder()

```

```

 .keyspaceName("catalog")
 .tableName("book_awards")
 .build();

 GetTableResponse response = keyspacesClient.getTable(request);

 // Access PreWarming details
 if (response.warmThroughputSpecification() != null) {
 WarmThroughputSpecificationSummary warmThroughputSummary =
response.warmThroughputSpecification();
 System.out.println("PreWarming Status: " +
warmThroughputSummary.status());
 System.out.println("Read Units: " +
warmThroughputSummary.readUnitsPerSecond());
 System.out.println("Write Units: " +
warmThroughputSummary.writeUnitsPerSecond());

 // Check if PreWarming is active
 if (warmThroughputSummary.status().equals("AVAILABLE")) {
 System.out.println("Table is fully pre-warmed and ready for high
throughput");
 } else if (warmThroughputSummary.status().equals("UPDATING")) {
 System.out.println("Table PreWarming is currently being
updated");
 }
 } else {
 System.out.println("Table does not have PreWarming enabled");
 }
 }
}

```

## 使用 Amazon CloudWatch 監控預暖資料表的效能

Amazon Keyspaces 預暖不會引入新的 CloudWatch 指標，但您可以使用現有的 Amazon Keyspaces 指標來監控預暖資料表的效能：

### SuccessfulRequestLatency

監控此指標，以確認預熱的資料表正在處理具有預期延遲的請求。

## WriteThrottleEvents 和 ReadThrottleEvents

對於正確預暖的資料表，這些指標應保持低。如果您在預熱前看到容量不足錯誤，您可能需要調整暖輸送量值。

## ConsumedReadCapacityUnits 和 ConsumedWriteCapacityUnits

這些指標顯示實際的容量使用量，這有助於驗證您的預暖組態是否適當。

## ProvisionedReadCapacityUnits 和 ProvisionedWriteCapacityUnits

對於佈建的資料表，這些指標會顯示目前配置的容量。

您可以在 CloudWatch 主控台中檢視這些指標，或使用 CloudWatch API 進行查詢。如需詳細資訊，請參閱[the section called “使用 CloudWatch 進行監控”](#)。

# 使用 Amazon Keyspaces 自動擴展自動管理輸送量容量

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。如果遊戲變得非常熱門，它可能會超過可用的資料庫資源，這會導致效能緩慢和客戶不滿意。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。

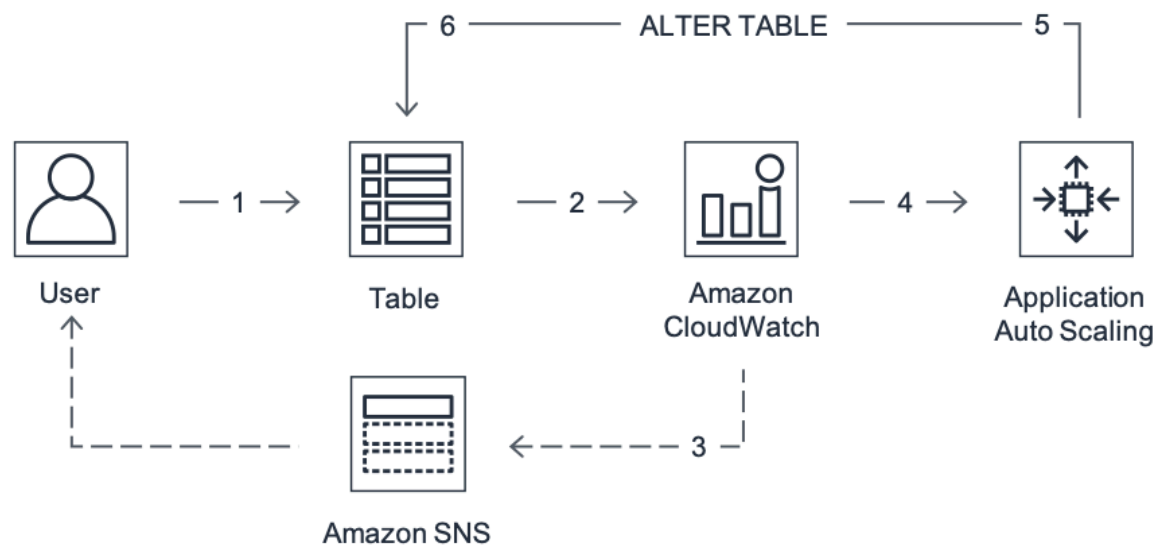
Amazon Keyspaces (適用於 Apache Cassandra) 透過自動調整輸送量容量以回應實際的應用程式流量，協助您有效率地佈建可變工作負載的輸送量容量。Amazon Keyspaces 使用 Application Auto Scaling 服務來代表您增加和減少資料表的讀取和寫入容量。如需 Application Auto Scaling 的詳細資訊，請參閱 [Application Auto Scaling 使用者指南](#)。

### Note

若要快速開始使用 Amazon Keyspaces 自動擴展，請參閱 [the section called “設定和更新自動擴展政策”](#)。

## Amazon Keyspaces 自動擴展的運作方式

下圖提供 Amazon Keyspaces 自動擴展如何管理資料表輸送量容量的高階概觀。



若要啟用資料表的自動擴展，您可以建立擴展政策。擴展政策會指定您要擴展讀取容量或寫入容量 (或兩者)，也可為資料表或索引指定最大與最小佈建容量單位設定。

擴展政策也會定義目標使用率。「目標使用率」是在某時間點使用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來向上或向下調整資料表的佈建輸送量，以回應實際工作負載。這樣做可讓實際容量使用率保持在或接近您的目標使用率。

您可以為讀取和寫入容量設定介於 20% 到 90% 之間的自動擴展目標使用率值。預設目標使用率為 70%。如果您的流量快速變更，而且您希望容量更快開始擴展，您可以將目標使用率設定為較低的百分比。如果您的應用程式流量變更較慢，而且您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。

如需擴展政策的詳細資訊，請參閱《[Application Auto Scaling 使用者指南](#)》中的 [Application Auto Scaling 的目標追蹤擴展政策](#)。 [Auto Scaling](#)

當您建立擴展政策時，Amazon Keyspaces 會代表您建立兩對 Amazon CloudWatch 警示。每對代表佈建和耗用輸送量設定的上限和下限。當資料表的實際使用率持續一段時間偏離您的目標使用率時，就會觸發這些 CloudWatch 警示。若要進一步了解 Amazon CloudWatch，請參閱 [Amazon CloudWatch 使用者指南](#)。

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)。CloudWatch 警示接著會叫用 Application Auto Scaling 來評估您的擴展政策。這反過來會向 Amazon Keyspaces 發出 Alter Table 請求，以視需要向上或向下調整資料表的佈建容量。若要進一步了解 Amazon SNS 通知，請參閱[設定 Amazon SNS 通知](#)。



Amazon Keyspaces 會透過增加（或減少）資料表的佈建輸送量容量來處理 Alter Table 請求，以便接近您的目標使用率。

#### Note

Amazon Keyspaces Auto Scaling 只有在實際工作負載持續幾分鐘保持升高（或降低）時，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。

## 自動擴展如何適用於多區域資料表

為了確保佈建容量模式中 AWS 區域 所有多區域資料表的所有資料表複本一律有足夠的讀取和寫入容量，建議您設定 Amazon Keyspaces 自動擴展。

當您在佈建模式中使用多區域資料表搭配自動擴展時，您無法停用單一資料表複本的自動擴展。但是，您可以調整不同區域的資料表讀取自動擴展設定。例如，您可以為複寫資料表的每個區域指定不同的讀取容量和讀取自動擴展設定。

您在指定區域中為資料表複本設定的讀取自動擴展設定會覆寫資料表的一般自動擴展設定。不過，寫入容量必須在所有資料表複本中保持同步，以確保有足夠的容量可在所有區域中複寫寫入。

Amazon Keyspaces 自動擴展 AWS 區域 會根據該區域中的用量，獨立更新每個 中資料表的佈建容量。因此，當自動擴展處於作用中狀態時，多區域資料表的每個區域中的佈建容量可能會不同。

您可以使用 Amazon Keyspaces 主控台 AWS CLI、API 或 CQL 來設定多區域資料表及其複本的自動擴展設定。如需如何建立和更新多區域資料表自動擴展設定的詳細資訊，請參閱 [the section called “更新多區域資料表的佈建容量和自動擴展設定”](#)。

#### Note

如果您針對多區域資料表使用自動擴展，則必須一律使用 Amazon Keyspaces API 操作來設定自動擴展設定。如果您直接使用 Application Auto Scaling API 操作來設定自動擴展設定，則無法指定多區域資料表 AWS 區域 的。這可能會導致不支援的組態。

## 使用須知

開始使用 Amazon Keyspaces 自動擴展之前，您應該注意下列事項：

- Amazon Keyspaces 自動擴展不適用於中東（阿拉伯聯合大公國）區域。
- Amazon Keyspaces 自動擴展可根據您的擴展政策，視需要增加讀取容量或寫入容量。所有 Amazon Keyspaces 配額都會保持有效，如中所述[配額](#)。
- Amazon Keyspaces 自動擴展不會阻止您手動修改佈建輸送量設定。這些手動調整不會影響連接到擴展政策的任何現有 CloudWatch 警示。
- 如果您使用主控台建立具有佈建輸送量容量的資料表，則預設會啟用 Amazon Keyspaces 自動擴展。您可以隨時修改自動擴展設定。如需詳細資訊，請參閱[the section called “關閉資料表的 Amazon Keyspaces 自動擴展”](#)。
- 如果您使用 CloudFormation 建立擴展政策，您應該從 管理擴展政策，CloudFormation 以便堆疊與堆疊範本同步。如果您從 Amazon Keyspaces 變更擴展政策，則會在堆疊重設時，以 CloudFormation 堆疊範本的原始值覆寫這些政策。
- 如果您使用 CloudTrail 監控 Amazon Keyspaces 自動擴展，您可能會在組態驗證程序中看到 Application Auto Scaling 發出的呼叫提醒。您可以使用 欄位來篩選掉這些提醒，該invokedBy欄位包含這些驗證檢查application-autoscaling.amazonaws.com的。

## 設定和更新 Amazon Keyspaces 自動擴展政策

您可以使用 主控台、CQL 或 AWS Command Line Interface (AWS CLI) 來設定新資料表和現有資料表的 Amazon Keyspaces 自動擴展。您也可以修改自動擴展設定或停用自動擴展。

如需設定向內擴展和向外擴展冷卻時間等更進階的功能，建議您使用 CQL 或 AWS CLI 來管理 Amazon Keyspaces 擴展政策。

### 主題

- [設定 Amazon Keyspaces 自動擴展的許可](#)
- [使用自動擴展建立新的資料表](#)
- [在現有資料表上設定自動擴展](#)
- [檢視資料表的 Amazon Keyspaces 自動擴展組態](#)
- [關閉資料表的 Amazon Keyspaces 自動擴展](#)
- [在 Amazon CloudWatch 中檢視 Amazon Keyspaces 資料表的自動擴展活動](#)

## 設定 Amazon Keyspaces 自動擴展的許可

若要開始使用，請確認主體具有適當的許可來建立和管理自動擴展設定。在 AWS Identity and Access Management (IAM) 中，管理 Amazon Keyspaces 擴展政策 `AmazonKeyspacesFullAccess` 需要 AWS 受管政策。

### Important

`application-autoscaling:*` 需要許可才能停用資料表上的自動擴展。您必須關閉資料表的自動擴展，才能將其刪除。

若要設定 Amazon Keyspaces 主控台存取和 Amazon Keyspaces 自動擴展的 IAM 使用者或角色，請新增下列政策。

### 連接 `AmazonKeyspacesFullAccess` 政策

1. 登入 AWS 管理主控台 並開啟位於 <https://console.aws.amazon.com/iam/> 的 IAM 主控台。
2. 在 IAM 主控台儀表板上，選擇使用者，然後從清單中選擇您的 IAM 使用者或角色。
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. 從政策清單中，選擇 `AmazonKeyspacesFullAccess`，然後選擇下一步：檢閱。
6. 選擇新增許可。

## 使用自動擴展建立新的資料表

當您建立新的 Amazon Keyspaces 資料表時，您可以為資料表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，將資料表註冊為可擴展的目標，並調整佈建的寫入或讀取容量。

如需如何建立多區域資料表和為資料表複本設定不同自動擴展設定的詳細資訊，請參閱 [the section called “在佈建模式中建立多區域資料表”](#)。

**Note**

Amazon Keyspaces 自動擴展需要存在可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”](#)。

## Console

使用主控台建立已啟用自動擴展的新資料表

1. 登入 AWS 管理主控台，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. 在資料表詳細資訊區段的建立資料表頁面上，選取金鑰空間，並提供新資料表的名稱。
4. 在資料欄區段中，建立資料表的結構描述。
5. 在主索引鍵區段中，定義資料表的主索引鍵，然後選取選用的叢集資料欄。
6. 在資料表設定區段中，選擇自訂設定。
7. 繼續讀取/寫入容量設定。
8. 針對容量模式，選擇佈建。
9. 在讀取容量區段中，確認已自動選取擴展。

在此步驟中，您會選取資料表的最小和最大讀取容量單位，以及目標使用率。

- 容量單位下限 – 輸入資料表應隨時準備好支援的輸送量下限值。此值必須介於 1 和您帳戶的每秒最大輸送量配額之間（預設為 40,000）。
- 最大容量單位 – 輸入您要為資料表佈建的最大輸送量。此值必須介於 1 和您帳戶的每秒最大輸送量配額之間（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，容量會自動擴展。當流量低於定義的目標時，會自動再次縮減規模。

**Note**

若要進一步了解帳戶的預設配額以及如何增加這些配額，請參閱 [配額](#)。

10. 在寫入容量區段中，選擇與上一個步驟中為讀取容量定義的相同設定，或手動設定容量值。

11. 選擇 Create Table (建立資料表)。您的資料表是使用指定的自動擴展參數建立的。

## Cassandra Query Language (CQL)

使用 CQL 使用 Amazon Keyspaces 自動擴展建立新的資料表

若要以程式設計方式設定資料表的自動擴展設定，您可以使用包含 Amazon Keyspaces 自動擴展參數的 AUTOSCALING\_SETTINGS 陳述式。這些參數定義了指示 Amazon Keyspaces 調整資料表佈建輸送量的條件，以及要採取哪些額外的選用動作。在此範例中，您會定義 mytable 的自動擴展設定。

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。需要下列值：
  - provisioned\_write\_capacity\_autoscaling\_update:
    - minimum\_units
    - maximum\_units
  - provisioned\_read\_capacity\_autoscaling\_update:
    - minimum\_units
    - maximum\_units
- scaling\_policy – Amazon Keyspaces 支援目標追蹤政策。若要定義目標追蹤政策，請設定下列參數。
  - target\_value – Amazon Keyspaces 自動擴展可確保耗用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: (選用) boolean 指定資料表 scale-in 是否已停用或已啟用的。此參數預設為停用。若要開啟 scale-in，請將 boolean 值設定為 FALSE。這表示會自動為您縮減資料表的容量。
  - scale\_out\_cooldown – 橫向擴展活動會增加資料表的佈建輸送量。若要為橫向擴展活動新增冷卻時間，請為指定以秒為單位的值 scale\_out\_cooldown。如果您未指定值，預設值為 0。如需目標追蹤和冷卻時間的詳細資訊，請參閱《Application Auto Scaling [Scaling 使用者指南](#)》中的 [目標追蹤擴展政策](#)。
  - scale\_in\_cooldown – 縮減活動可減少資料表的佈建輸送量。若要為縮減活動新增冷卻時間，請為指定以秒為單位的值 scale\_in\_cooldown。如果您未指定值，預設值為 0。如需目標追蹤和冷卻時間的詳細資訊，請參閱《Application Auto Scaling [Scaling 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。耗用與佈建比率現在為  $(150/200)$ ，也就是 75%。此比率超過您的目標，因此自動擴展會將佈建的寫入容量增加到 215，使得比率為  $(150 / 215)$  或 69.77%，`target_value` 盡可能接近您的，但不會超過它。

對於 `mytable`，您可以將讀取和寫入容量 `TargetValue` 設定為 50%。Amazon Keyspaces 自動擴展會在 5–10 個容量單位的範圍內調整資料表的佈建輸送量，以便 `consumed-to-provisioned` 比率維持在 50% 或接近 50%。對於讀取容量，您可以將 `ScaleOutCooldown` 和 `ScaleInCooldown` 的值設定為 60 秒。

您可以使用下列陳述式，在啟用自動擴展的情況下建立新的 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
 }
}
};

```

## CLI

使用 建立具有 Amazon Keyspaces 自動擴展的新資料表 AWS CLI

若要以程式設計方式設定資料表的自動擴展設定，您可以使用 `autoScalingSpecification` 動作來定義 Amazon Keyspaces 自動擴展的參數。這些參數定義了指示 Amazon Keyspaces 調整資料表佈建輸送量的條件，以及要採取哪些額外的選用動作。在此範例中，您可以定義 `mytable` 的自動擴展設定。

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自動擴展。然後，您必須為指定下列參數 `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 支援目標追蹤政策。若要定義目標追蹤政策，請設定下列參數。
    - `targetValue` – Amazon Keyspaces 自動擴展可確保耗用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`：(選用) `boolean` 指定資料表 `scale-in` 是否已停用或已啟用的。此參數預設為停用。若要開啟 `scale-in`，請將 `boolean` 值設定為 `FALSE`。這表示會自動為您縮減資料表的容量。
    - `scaleOutCooldown` – 橫向擴展活動會增加資料表的佈建輸送量。若要為向外擴展活動新增冷卻時間，請為指定以秒為單位的值 `ScaleOutCooldown`。預設值為 0。如需目標追蹤和冷卻期間的詳細資訊，請參閱《Application Auto Scaling [Scaling 使用者指南](#)》中的 [目標追蹤擴展政策](#)。
    - `scaleInCooldown` – 縮減活動可減少資料表的佈建輸送量。若要為縮減活動新增冷卻時間，請為指定以秒為單位的值 `ScaleInCooldown`。預設值為 0。如需目標追蹤和冷卻期間的詳細資訊，請參閱《Application Auto Scaling [Scaling 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。耗用與佈建比率現在為 (150/200)，也就是 75%。此比率超過您的目標，因此自動擴展會將佈建的寫入容量增加到 215，因此比率為 (150 / 215) 或 69.77%，TargetValue 盡可能接近您的，但不會超過它。

對於 mytable，您可以將讀取和寫入容量 TargetValue 設定為 50%。Amazon Keyspaces 自動擴展會在 5–10 個容量單位的範圍內調整資料表的佈建輸送量，以便 consumed-to-provisioned 比率維持在 50% 或接近 50%。對於讀取容量，您可以將 ScaleOutCooldown 和 的值設定為 ScaleInCooldown 60 秒。

建立具有複雜自動擴展設定的資料表時，從 JSON 檔案載入自動擴展設定會很有幫助。針對下列範例，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
```

## 在現有資料表上設定自動擴展

您可以更新現有的 Amazon Keyspaces 資料表，以開啟資料表寫入或讀取容量的自動擴展。如果您要更新目前處於隨需容量模式的資料表，則必須先將資料表的容量模式變更為佈建的容量模式。

如需如何更新多區域資料表自動擴展設定的詳細資訊，請參閱 [the section called “更新多區域資料表的佈建容量和自動擴展設定”](#)。

Amazon Keyspaces 自動擴展需要存在可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”](#)。



## Console

為現有資料表設定 Amazon Keyspaces 自動擴展

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 選擇您要使用的資料表，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，確定資料表使用佈建容量模式。
5. 選取自動擴展，並參閱 中的步驟 6 [the section called “使用自動擴展建立新的資料表”](#)以編輯讀取和寫入容量。
6. 定義自動擴展設定時，請選擇儲存。

## Cassandra Query Language (CQL)

使用 CQL 設定具有 Amazon Keyspaces 自動擴展的現有資料表

您可以使用現有 Amazon Keyspaces 資料表的 ALTER TABLE 陳述式，為資料表的寫入或讀取容量設定自動擴展。如果您要更新目前處於隨需容量模式的資料表，您必須將 `capacity_mode` 設定為已佈建。如果您的資料表已處於佈建容量模式，則可以省略此欄位。

在下列範例中，陳述式會更新資料表 `mytable`，其處於隨需容量模式。陳述式會將資料表的容量模式變更為已啟用自動擴展的佈建模式。

寫入容量設定在 5–10 個容量單位的範圍內，目標值為 50%。讀取容量也會設定在 5–10 個容量單位的範圍內，目標值為 50%。對於讀取容量，您可以將 `scale_out_cooldown` 和 `scale_in_cooldown` 的值設定為 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
 }
 }
 }
};

```

## CLI

使用 設定具有 Amazon Keyspaces 自動擴展的現有資料表 AWS CLI

對於現有的 Amazon Keyspaces 資料表，您可以使用 UpdateTable 操作開啟資料表寫入或讀取容量的自動擴展。

您可以使用下列命令來開啟現有資料表的 Amazon Keyspaces 自動擴展。資料表的自動擴展設定會從 JSON 檔案載入。針對下列範例，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自動擴展設定的詳細資訊，請參閱 [the section called “使用自動擴展建立新的資料表”](#)。

```

aws keyspace update-table --keyspace-name mykeyspace --table-name mytable
 \ --capacity-specification
 throughputMode=PROVISIONED,readCapacityUnits=1,writeCapacityUnits=1
 \ --auto-scaling-specification file://auto-scaling.json

```

## 檢視資料表的 Amazon Keyspaces 自動擴展組態

您可以使用 主控台、CQL 或 AWS CLI 來檢視和更新資料表的 Amazon Keyspaces 自動擴展設定。

## Console

使用主控台檢視自動擴展設定

1. 選擇您要檢視的資料表，然後前往容量索引標籤。
2. 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 [the section called “使用自動擴展建立新的資料表”](#)。

## Cassandra Query Language (CQL)

使用 CQL 檢視資料表的 Amazon Keyspaces 自動擴展政策

若要檢視資料表自動擴展組態的詳細資訊，請使用下列命令。

```
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}}}
```

## CLI

使用 檢視資料表的 Amazon Keyspaces 自動擴展政策 AWS CLI

若要檢視資料表的自動擴展組態，您可以使用 `get-table-auto-scaling-settings` 操作。下列 CLI 命令為範例。

```
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-1:111122223333:/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
 }
 }
 }
 }
}
```

## 關閉資料表的 Amazon Keyspaces 自動擴展

您可以隨時關閉資料表的 Amazon Keyspaces 自動擴展。如果您不再需要擴展資料表的讀取或寫入容量，您應該考慮關閉自動擴展，以便 Amazon Keyspaces 不會繼續修改資料表的讀取或寫入容量設定。您可以使用 主控台、CQL 或 更新資料表 AWS CLI。

關閉自動擴展也會刪除代表您建立的 CloudWatch 警示。

若要刪除 Application Auto Scaling 用來存取 Amazon Keyspaces 資料表的服務連結角色，請遵循中的步驟 [the section called “刪除 Amazon Keyspaces 的服務連結角色”](#)。

### 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擴展 AWS 區域。

## Console

使用主控台關閉資料表的 Amazon Keyspaces 自動擴展

使用 Amazon Keyspaces 主控台

1. 登入 AWS 管理主控台，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，然後前往容量索引標籤。
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。停用自動調整規模會將資料表取消註冊為 Application Auto Scaling 的可擴展目標。

## Cassandra Query Language (CQL)

使用 CQL 關閉資料表的 Amazon Keyspaces 自動擴展

下列陳述式會關閉資料表 mytable 寫入容量的自動擴展。

```
ALTER TABLE mykeyspace.mytable
WITH AUTOSCALING_SETTINGS = {
 'provisioned_write_capacity_autoscaling_update': {
```

```
 'autoscaling_disabled': true
 }
};
```

## CLI

使用 關閉資料表的 Amazon Keyspaces 自動擴展 AWS CLI

下列命令會關閉資料表讀取容量的自動擴展。它也會刪除代表您建立的 CloudWatch 警示。

```
aws keyspaces update-table --keyspace-name mykeyspace --table-name mytable
 \ --auto-scaling-specification
 readCapacityAutoScaling={autoScalingDisabled=true}
```

## 在 Amazon CloudWatch 中檢視 Amazon Keyspaces 資料表的自動擴展活動

您可以使用 Amazon CloudWatch 來監控 Amazon Keyspaces 自動擴展如何使用 資源，這會產生有關您的用量和效能的指標。遵循 [Application Auto Scaling 使用者指南](#) 中的步驟來建立 CloudWatch 儀表板。

## 在 Amazon Keyspaces 中有效使用高載容量

Amazon Keyspaces 透過提供高載容量，為您的每個分割區輸送量佈建提供一些靈活性。每當您未完全使用分割區的輸送量時，Amazon Keyspaces 會保留一部分未使用的容量，以供後續輸送量暴增處理用量激增。

Amazon Keyspaces 目前保留最多 5 分鐘 (300 秒) 未使用的讀取和寫入容量。在偶爾高載的讀取或寫入活動期間，這些額外的容量單位可以快速耗用，甚至比您為資料表定義的每秒佈建輸送量更快。

Amazon Keyspaces 也可以使用高載容量進行背景維護和其他任務，而無需事先通知。

請注意，高載容量的詳細資訊可能會在將來有所變更。

# 使用 Amazon Keyspaces (適用於 Apache Cassandra) 功能

本章提供有關使用 Amazon Keyspaces 和各種資料庫功能的詳細資訊，例如備份和還原、存留時間和多區域複寫。

- 存留時間 – Amazon Keyspaces 會根據您設定的存留時間值自動過期資料表中的資料。了解如何設定 TTL，以及如何在資料表中使用它。
- PITR – 透過建立資料表資料的連續備份，保護您的 Amazon Keyspaces 資料表免於意外寫入或刪除操作。了解如何在資料表上設定 PITR，以及如何將資料表還原至特定時間點，或如何還原意外刪除的資料表。
- 使用多區域資料表 – Amazon Keyspaces 中的多區域資料表必須在隨需或佈建容量模式中設定寫入輸送量容量，並搭配自動擴展。透過估計每個區域所需的寫入容量單位 (WCUs) 來規劃輸送量容量需求，並佈建所有區域的寫入總和，以確保有足夠的容量供複寫寫入使用。
- Amazon Keyspaces 變更資料擷取 (CDC) – Amazon Keyspaces CDC 會以近乎即時的方式從您的資料表串流記錄資料列層級變更事件。了解如何使用 Kinesis Client Library (KCL) 來取用和處理來自 Amazon Keyspaces CDC 串流的資料。
- 查詢和分頁 – Amazon Keyspaces 支援進階查詢功能，例如使用 IN 運算子搭配 SELECT 陳述式、使用排序結果 ORDER BY，以及自動分頁大型結果集。本節說明 Amazon Keyspaces 如何處理這些查詢並提供範例。
- 分割區 – Amazon Keyspaces 提供三個分割區：Murmur3Partitioner (預設)、RandomPartitioner、和 DefaultPartitioner。您可以使用 AWS 管理主控台 或 Cassandra 查詢語言 (CQL)，在帳戶層級變更每個區域的分割器。
- 用戶端時間戳記 – 用戶端時間戳記是 Amazon Keyspaces 為資料表中每個儲存格保留的 Cassandra 相容時間戳記。使用用戶端時間戳記來解決衝突，並讓用戶端應用程式判斷寫入順序。
- 使用者定義類型 (UDTs) – 使用 UDTs，您可以在應用程式中定義代表實際資料階層的資料結構。
- 標記資源 – 您可以使用標籤來標記 Amazon Keyspaces 資源，例如金鑰空間和資料表。標籤有助於分類資源、啟用成本追蹤，以及讓您根據標籤設定存取控制。本節涵蓋 Amazon Keyspaces 的標記限制、操作和最佳實務。
- CloudFormation 範本 – CloudFormation 協助您建立模型和設定 Amazon Keyspaces 金鑰空間和資料表，以減少建立和管理資源和基礎設施的時間。

## 主題

- [Amazon Keyspaces 中的系統金鑰空間](#)
- [Amazon Keyspaces 中使用者定義的類型 \(UDT\)](#)

- [在 Amazon Keyspaces 中使用 CQL 查詢](#)
- [在 Amazon Keyspaces 中使用變更資料擷取 \(CDC\) 串流](#)
- [在 Amazon Keyspaces 中使用分割區](#)
- [Amazon Keyspaces 中的用戶端時間戳記](#)
- [Amazon Keyspaces 的多區域複寫 \(適用於 Apache Cassandra\)](#)
- [使用 point-in-time 復原來備份和還原資料](#)
- [使用 Amazon Keyspaces 的存留時間 \(TTL\) 過期資料 \(適用於 Apache Cassandra\)](#)
- [搭配 AWS SDK 使用此服務](#)
- [使用 Amazon Keyspaces 資源的標籤和標籤](#)
- [使用 建立 Amazon Keyspaces 資源 AWS CloudFormation](#)
- [搭配 Amazon Keyspaces 使用 NoSQL Workbench \(適用於 Apache Cassandra\)](#)

## Amazon Keyspaces 中的系統金鑰空間

Amazon Keyspaces 中的系統金鑰空間和系統資料表 (適用於 Apache Cassandra) 是唯讀資源，可存放 Amazon Keyspaces 資源的中繼資料。系統金鑰空間存在於每個 AWS 帳戶，無論您是否已建立任何金鑰空間或資料表。它們是 Apache Cassandra 的相容性功能，而且免費提供。

您無法修改或刪除系統金鑰空間。Amazon Keyspaces 主控台只會顯示使用者建立的金鑰空間。系統金鑰空間可透過 CQL 以程式設計方式存取，並出現在 AWS CloudFormation 和 等服務中 AWS Config。

Amazon Keyspaces 使用四個系統金鑰空間：

- system
- system\_schema
- system\_schema\_mcs
- system\_multiregion\_info

下列各節提供 Amazon Keyspaces 中支援之系統金鑰空間和系統資料表的詳細資訊。

### system

這是 Cassandra 金鑰空間。Amazon Keyspaces 使用以下資料表。



| 資料表名稱          | 欄位名稱                                                                                                                                                                                                                                                          | 說明                                                                                                                                                                                                                    |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| local          | key, bootstrap_address, 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 | 本機金鑰空間的相關資訊。                                                                                                                                                                                                          |
| 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，如果您透過公有端點連線，您會看到九個可用 IP 地址的清單。如果您是透過 FIPS 端點連線，您會看到三個 IP 地址的清單。如果您是透過 AWS PrivateLink VPC 端點連線，您會看到已設定的 IP 地址清單。如需詳細資訊，請參閱 <a href="#">the section called “使用界面 VPC 端點資訊填入 system.peers 資料表項目”</a> 。 |
| size_estimates | keyspace_name, table_name, range_start, range_end, mean_partition_size, partitions_count                                                                                                                                                                      | 此資料表定義每個資料表的每個字符範圍的總大小和分割區數量。Apache Cassandra Spark Connector 需要此操                                                                                                                                                    |

| 資料表名稱               | 欄位名稱                                             | 說明                   |
|---------------------|--------------------------------------------------|----------------------|
|                     |                                                  | 作，這會使用預估的分割區大小來分配工作。 |
| prepared_statements | prepared_id,<br>logged_keyspace,<br>query_string | 此資料表包含已儲存查詢的相關資訊。    |

## system\_schema

這是 Cassandra 金鑰空間。Amazon Keyspaces 使用以下資料表。

| 資料表名稱     | 欄位名稱                                                                                                                                                                                                                                                                                                        | 說明           |
|-----------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------|
| keyspaces | keyspace_name,<br>durable_writes,<br>replication                                                                                                                                                                                                                                                            | 特定金鑰空間的相關資訊。 |
| tables    | keyspace_name,<br>table_name, bloom_filter_fp_chance,<br>caching, comment,<br>compaction, compression, crc_check_chance, dclocal_read_repair_chance,<br>default_time_to_live, extensions,<br>flags, gc_grace_seconds, id,<br>max_index_interval, memtable_flush_period_in_ms, min_index_interval, read_repa | 特定資料表的相關資訊。  |

| 資料表名稱   | 欄位名稱                                                                                              | 說明                     |
|---------|---------------------------------------------------------------------------------------------------|------------------------|
|         | ir_chance, speculative_retry                                                                      |                        |
| types   | keyspace_name, type_name, field_names, field_types                                                | 特定使用者定義類型 (UDT) 的相關資訊。 |
| columns | keyspace_name, table_name, column_name, clustering_order, column_name_bytes, kind, position, type | 特定資料欄的相關資訊。            |

## system\_schema\_mcs

這是存放 AWS 或 Amazon Keyspaces 特定設定相關資訊的 Amazon Keyspaces 金鑰空間。

| 資料表名稱     | 欄位名稱                                                                                                                                                   | 說明                                                                                                                                                   |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------|
| keyspaces | keyspace_name, durable_writes, replication                                                                                                             | 查詢此表格，以程式設計方式找出是否已建立金鑰空間。如需詳細資訊，請參閱 <a href="#">the section called “檢查金鑰空間建立狀態”</a> 。                                                                |
| tables    | keyspace_name, creation_time, speculative_retry, cdc, gc_grace_seconds, crc_check_chance, min_index_interval, bloom_filter_fp_chance, flags, custom_pr | 查詢此資料表以了解特定資料表的狀態。如需詳細資訊，請參閱 <a href="#">the section called “檢查資料表建立狀態”</a> 。<br><br>您也可以查詢此表格，列出 Amazon Keyspaces 特定的設定，並儲存為 custom_properties 。例如： |

| 資料表名稱          | 欄位名稱                                                                                                                                                                                                                                                | 說明                                                                                                                                                                                                                    |
|----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                | operties, dclocal_read_repair_chance, table_name, caching, default_time_to_live, read_repair_chance, max_index_interval, extensions, compaction, comment, id, compression, memtable_flush_period_in_ms, cdc_specification, latest_stream_arn status | <ul style="list-style-type: none"> <li>• capacity_mode</li> <li>• cdc_specification</li> <li>• client_side_timestamps</li> <li>• encryption_specification</li> <li>• point_in_time_recovery</li> <li>• ttl</li> </ul> |
| tables_history | keyspace_name, table_name, event_time, creation_time, custom_properties, event                                                                                                                                                                      | 查詢此表格以了解特定資料表的結構描述變更。                                                                                                                                                                                                 |
| columns        | keyspace_name, table_name, column_name, clustering_order, column_name_bytes, kind, position, type                                                                                                                                                   | 此資料表與 system_schema 金鑰空間中的 Cassandra 資料表相同。                                                                                                                                                                           |
| tags           | resource_id, keyspace_name, resource_name, resource_type, tags                                                                                                                                                                                      | 查詢此表格以了解金鑰空間是否有標籤。如需詳細資訊，請參閱 <a href="#">the section called “檢視資料表標籤”</a> 。                                                                                                                                           |

| 資料表名稱       | 欄位名稱                                                                                                                                                 | 說明                                                                                                                                                                                        |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| types       | keyspace_name, type_name, field_names, field_types, max_nesting_depth, last_modified_timestamp, status, direct_referring_tables, direct_parent_types | 查詢此表格以了解使用者定義類型 (UDTs) 的相關資訊。例如，您可以查詢此資料表，列出指定金鑰空間的所有 UDTs。如需詳細資訊，請參閱 <a href="#">the section called “使用者定義的類型 (UDTs)”</a> 。                                                              |
| autoscaling | keyspace_name, table_name, provisioned_read_capacity_autoscaling_update, provisioned_write_capacity_autoscaling_update                               | 查詢此資料表以取得佈建資料表的自動擴展設定。請注意，在資料表處於作用中狀態之前，這些設定將無法使用。若要查詢此資料表，您必須在 WHERE 子句 table_name 中指定 keyspace_name 和 。如需詳細資訊，請參閱 <a href="#">the section called “檢視資料表的 Amazon Keyspaces 自動擴展組態”</a> 。 |

## system\_multiregion\_info

這是存放多區域複寫相關資訊的 Amazon Keyspaces 金鑰空間。

| 資料表名稱  | 欄位名稱                                      | 說明                                                                                     |
|--------|-------------------------------------------|----------------------------------------------------------------------------------------|
| tables | keyspace_name, table_name, region, status | 此資料表包含多區域資料表的相關資訊，例如，複寫資料表 AWS 區域 所在的 和 資料表的狀態。您也可以查詢此表格，列出存放為 之 Amazon Keyspaces 特有的設 |

| 資料表名稱                    | 欄位名稱                                                                                                                                                                                          | 說明                                                                                                                                                                                                                                                                                                                  |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                          |                                                                                                                                                                                               | <p>定 <code>custom_properties</code> 。</p> <p>例如：</p> <ul style="list-style-type: none"> <li><code>capacity_mode</code></li> </ul> <p>若要查詢此資料表，您必須在 WHERE 子句 <code>table_name</code> 中指定 <code>keyspace_name</code> 和 <code>table_name</code>。</p> <p>如需詳細資訊，請參閱 <a href="#">the section called “建立多區域金鑰空間”</a>。</p> |
| <code>keyspaces</code>   | <code>keyspace_name</code> ,<br><code>region</code> , <code>status</code> ,<br><code>tables_replication_progress</code>                                                                       | <p>此資料表包含 ALTER KEYSPACE 操作的進度資訊，該操作會將複本新增至金鑰空間，例如，在新區域中已建立多少個資料表，以及仍在進行多少個資料表。如需範例，請參閱 <a href="#">the section called “檢查複寫進度”</a>。</p>                                                                                                                                                                            |
| <code>autoscaling</code> | <code>keyspace_name</code> ,<br><code>table_name</code> , <code>provisioned_read_capacity_update</code> ,<br><code>provisioned_write_capacity_autoscaling_update</code> , <code>region</code> | <p>查詢此資料表以取得多區域佈建資料表的自動擴展設定。請注意，在資料表處於作用中狀態之前，這些設定將無法使用。若要查詢此資料表，您必須在 WHERE 子句 <code>table_name</code> 中指定 <code>keyspace_name</code> 和 <code>table_name</code>。</p> <p>如需詳細資訊，請參閱 <a href="#">the section called “更新多區域資料表的佈建容量和自動擴展設定”</a>。</p>                                                                   |

| 資料表名稱 | 欄位名稱                                                                                                                                                         | 說明                                                                                                                                                           |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------|
| types | keyspace_name, type_name, field_names, field_types, max_nesting_depth, last_modified_timestamp, status, direct_referring_tables, direct_parent_types, region | 查詢此表格，以了解多區域金鑰空間中使用使用者定義類型 (UDTs) 的相關資訊。例如，您可以查詢此資料表以列出所有資料表複本，以及其對指定金鑰空間使用 UDTs 個別 AWS 區域。如需詳細資訊，請參閱 <a href="#">the section called “使用者定義的類型 (UDTs)”</a> 。 |

## Amazon Keyspaces 中使用使用者定義的類型 (UDT)

使用者定義類型 (UDT) 是欄位和資料類型的群組，可用於在 Amazon Keyspaces 中定義單一資料欄。UDTs 的有效資料類型都是支援的 Cassandra 資料類型，包括您已在相同金鑰空間中建立的集合和其他 UDTs。如需支援的 Cassandra 資料類型的詳細資訊，請參閱 [the section called “Cassandra 資料類型支援”](#)。

您可以在 Amazon Keyspaces 中使用使用者定義類型 (UDTs)，以更有效率的方式組織資料。例如，您可以使用巢狀集合建立 UDTs，以便在應用程式中實作更複雜的資料建模。您也可以使用凍結關鍵字來定義 UDTs。

UDTs 繫結至金鑰空間，並可供相同金鑰空間中的所有資料表和 UDTs 使用。您可以在單一區域和多區域金鑰空間中建立 UDTs。

您可以建立新的資料表或修改現有的資料表，並新增使用 UDT 的新資料欄。若要使用巢狀 UDT 建立 UDT，必須將巢狀 UDT 凍結。

若要檢閱每個金鑰空間支援多少 UDTs、支援的巢狀層級，以及與 UDTs 相關的其他預設值和配額，請參閱 [the section called “Amazon Keyspaces 中使用使用者定義類型 \(UDTs\) 的配額和預設值”](#)。

如需如何計算 UDTs 編碼大小的資訊，請參閱 [the section called “根據資料類型估計資料值的編碼大小”](#)。

如需 CQL 語法的詳細資訊，請參閱 [the section called “類型”](#)。

若要進一步了解 UDTs 和時間點還原，請參閱 [the section called “PITR UDTs”](#)。

## 主題

- [設定許可以在 Amazon Keyspaces 中使用使用者定義類型 \(UDTs\)](#)
- [在 Amazon Keyspaces 中建立使用者定義類型 \(UDT\)](#)
- [檢視 Amazon Keyspaces 中的使用者定義類型 \(UDTs\)](#)
- [在 Amazon Keyspaces 中刪除使用者定義類型 \(UDT\)](#)

## 設定許可以在 Amazon Keyspaces 中使用使用者定義類型 (UDTs)

如同資料表，UDTs 會繫結至特定金鑰空間。但與資料表不同，您無法直接定義 UDTs 許可。UDTs 不會被視為 中的資源 AWS，而且它們沒有 Amazon Resource Name (ARN) 格式的唯一識別符。相反地，若要授予 IAM 主體在 UDT 上執行特定動作的許可，您必須定義 UDT 所繫結之金鑰空間的許可。若要在多區域金鑰空間中使用 UDTs，需要額外許可。

若要能夠建立、檢視或刪除 UDTs，委託人，例如 IAM 使用者或角色，需要對 UDT 繫結的金鑰空間執行相同動作所需的相同許可。

如需的詳細資訊 AWS Identity and Access Management，請參閱 [the section called “AWS Identity and Access Management”](#)。

### 建立 UDT 的許可

若要在單一區域金鑰空間中建立 UDT，委託人需要金鑰空間的 Create 許可。

下列 IAM 政策為範例。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "cassandra:Create",
 "Resource": [
 "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/"
]
 }
]
}
```

若要在多區域金鑰空間中建立 UDT，除了 Create 許可之外，委託人還需要指定金鑰空間之 動作 CreateMultiRegionResource 的許可。



下列 IAM 政策為範例。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": ["cassandra:Create", "cassandra:CreateMultiRegionResource"],
 "Resource": [
 "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/"
]
 }
]
}
```

## 檢視 UDT 的許可

若要在單一區域金鑰空間中檢視或列出 UDTs，委託人需要系統金鑰空間的讀取許可。如需詳細資訊，請參閱[the section called “system\\_schema\\_mcs”](#)。

下列 IAM 政策為範例。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "cassandra:Select",
 "Resource": [
 "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
]
 }
]
}
```

若要檢視或列出多區域金鑰空間UDTs，委託人需要動作SELECT和系統金鑰空間SelectMultiRegionResource的許可。如需詳細資訊，請參閱[the section called “system\\_multiregion\\_info”](#)。

下列 IAM 政策為範例。

```
{
```

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": ["cassandra:Select", "cassandra:SelectMultiRegionResource"],
 "Resource": [
 "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
]
 }
]
}

```

## 刪除 UDT 的許可

若要從單一區域金鑰空間刪除 UDT，委託人需要指定金鑰空間之 Drop 動作的許可。

下列 IAM 政策為範例。

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "cassandra:Drop",
 "Resource": [
 "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/"
]
 }
]
}

```

若要從多區域金鑰空間刪除 UDT，委託人需要 Drop 動作的許可，以及指定金鑰空間的 DropMultiRegionResource 動作的許可。

下列 IAM 政策為範例。

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": ["cassandra:Drop", "cassandra:DropMultiRegionResource"],

```

```

 "Resource": [
 "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/"
]
 }
]
}

```

## 在 Amazon Keyspaces 中建立使用者定義類型 (UDT)

若要在單一區域金鑰空間中建立 UDT，您可以在 CQL、`create-type` 命令與 AWS CLI 或 主控台中使用 `CREATE TYPE` 陳述式。

UDT 名稱不得超過 48 個字元、開頭必須是英文字母字元，且只能包含英數字元和底線。Amazon Keyspaces 會自動將大寫字元轉換成小寫字元。

或者，您可以使用雙引號宣告 UDT 名稱。在 UDT 名稱前後加上雙引號來宣告名稱時，Amazon Keyspaces 會保留大寫並允許特殊字元。

您也可以在建 UDT 時使用雙引號作為名稱的一部分，但必須使用額外的雙引號字元來逸出每個雙引號字元。

下表顯示允許的 UDT 名稱範例。第一欄顯示如何在建立類型時輸入名稱，第二欄顯示 Amazon Keyspace 如何在內部格式化名稱。Amazon Keyspaces 預期 `GetType` 這類操作使用格式化的名稱。

| 輸入的名稱                     | 格式化名稱                   | 注意                                                   |
|---------------------------|-------------------------|------------------------------------------------------|
| MY_UDT                    | my_udt                  | 若沒有雙引號，Amazon Keyspaces 就會將所有大寫字元轉換成小寫字元。            |
| "MY_UDT"                  | MY_UDT                  | 若有雙引號，Amazon Keyspaces 就會遵循大寫字元，並從格式化名稱中移除雙引號。       |
| "1234"                    | 1234                    | 若有雙引號，名稱開頭就可以是數字，而且 Amazon Keyspaces 會從格式化名稱中移除雙引號。  |
| "Special_Ch@r@cter s<>!!" | Special_Ch@r@cter s<>!! | 若有雙引號，名稱就可以包含特殊字元，而且 Amazon Keyspaces 會從格式化名稱中移除雙引號。 |

| 輸入的名稱                       | 格式化名稱              | 注意                                        |
|-----------------------------|--------------------|-------------------------------------------|
| "nested"<br>""""quote<br>s" | nested""<br>quotes | Amazon Keyspaces 會從格式化名稱中移除最外層的雙引號及逸出雙引號。 |

## Console

使用 Amazon Keyspaces 主控台建立使用者定義類型 (UDT)

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇金鑰空間，然後從清單中選擇金鑰空間。
3. 選擇 UDTs 標籤。
4. 選擇建立 UDT
5. 在 UDT 詳細資訊下，輸入 UDT 的名稱。在 UDT 欄位下，您可以定義 UDT 的結構描述。
6. 若要完成，請選擇建立 UDT。

## Cassandra Query Language (CQL)

使用 CQL 建立使用者定義類型 (UDT)

在此範例中，我們建立了中使用的新版本書籍獎勵表 [the section called “建立資料表”](#)。在此表格中，我們會存放作者為指定書籍收到的所有獎勵。我們建立兩個巢狀 UDTs，其中包含有關獲得獎勵之書籍的資訊。

1. 建立名為的金鑰空間 catalog。

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

2. 建立第一個類型。此類型存放 BISAC 代碼，用於定義書籍類型。BISAC 代碼由英數字元代碼組成，最多四個主題區域。

```
CREATE TYPE catalog.bisac (
 bisac_code text,
 subject1 text,
 subject2 text,
```

```
subject3 text,
subject4 text
);
```

3. 為使用第一個 UDT 的書籍獎勵建立第二個類型。巢狀 UDT 必須凍結。

```
CREATE TYPE catalog.book (
 award_title text,
 book_title text,
 publication_date date,
 page_count int,
 ISBN text,
 genre FROZEN <bisac>
);
```

4. 使用作者名稱的資料欄建立資料表，並使用書籍獎勵的清單類型。請注意，清單中使用的 UDT 必須凍結。

```
CREATE TABLE catalog.authors (
 author_name text PRIMARY KEY,
 awards list <FROZEN <book>>
);
```

5. 在此步驟中，我們會將一系列資料插入新資料表。

```
CONSISTENCY LOCAL_QUORUM;
```

```
INSERT INTO catalog.authors (author_name, awards) VALUES (
'John Stiles' ,
[
 {
 award_title: 'Wolf',
 book_title: 'Yesterday',
 publication_date: '2020-10-10',
 page_count: 345,
 ISBN: '026204630X',
 genre: { bisac_code: 'FIC014090', subject1: 'FICTION', subject2:
'Historical', subject3: '20th Century', subject4: 'Post-World War II'}
 },
 {award_title: 'Richard Roe',
 book_title: 'Who ate the cake?',
 publication_date: '2019-05-13',
 page_count: 193,
```

```
ISBN: '9780262046305',
genre: { bisac_code: 'FIC022130', subject1: 'FICTION', subject2: 'Mystery &
Detective', subject3: 'Cozy', subject4: 'Culinary'}
}]
);
```

6. 在最後一個步驟中，我們從資料表讀取資料。

```
SELECT * FROM catalog.authors;
```

命令的輸出應如下所示。

```
author_name | awards

+-----+
John Stiles | [{award_title: 'Wolf', book_title: 'Yesterday', publication_date:
2020-10-10, page_count: 345, isbn: '026204630X', genre: {bisac_code:
'FIC014090', subject1: 'FICTION', subject2: 'Historical', subject3: '20th
Century', subject4: 'Post-World War II'}}, {award_title: 'Richard Roe',
book_title: 'Who ate the cake?', publication_date: 2019-05-13, page_count: 193,
isbn: '9780262046305', genre: {bisac_code: 'FIC022130', subject1: 'FICTION',
subject2: 'Mystery & Detective', subject3: 'Cozy', subject4: 'Culinary'}}]
(1 rows)
```

如需 CQL 語法的詳細資訊，請參閱 [the section called “CREATE TYPE”](#)。

## CLI

### 使用 建立使用者定義類型 (UDT) AWS CLI

1. 若要建立類型，您可以使用下列語法。

```
aws keyspaces create-type
--keyspace-name 'my_keyspace'
--type-name 'my_udt'
--field-definitions
'['
 {"name" : "field1", "type" : "int"},
 {"name" : "field2", "type" : "text"}
']'
```

2. 該命令的輸出看起來與此範例類似。請注意，會 `typeName` 傳回 UDT 的格式化名稱。

```
{
 "keyspaceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
my_keyspace/",
 "typeName": "my_udt"
}
```

## 檢視 Amazon Keyspaces 中的使用者定義類型 (UDTs)

若要檢視或列出單一區域金鑰空間中的所有 UDTs，您可以使用 CQL `system_schema_mcs.types` 中的陳述式查詢系統金鑰空間中的資料表，或搭配 AWS CLI 或 主控台使用 `get-type` 和 `list-type` 命令。

對於任一選項，IAM 主體都需要系統金鑰空間的讀取許可。如需詳細資訊，請參閱 [the section called “設定 許可”](#)。

### Console

使用 Amazon Keyspaces 主控台檢視使用者定義類型 (UDT)

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇金鑰空間，然後從清單中選擇金鑰空間。
3. 選擇 UDTs 索引標籤，以檢閱金鑰空間中所有 UDTs 的清單。
4. 若要詳細檢閱一個 UDT，請從清單中選擇 UDT。
5. 在結構描述索引標籤上，您可以檢閱結構描述。在用於 索引標籤上，您可以查看此 UDT 是否用於資料表或其他 UDTs。請注意，您只能刪除資料表或其他 UDTs 未使用的 UDTs。

### Cassandra Query Language (CQL)

使用 CQL 檢視單一區域金鑰空間的使用者定義類型 (UDTs)

1. 若要查看指定金鑰空間中可用的類型，您可以使用下列陳述式。

```
SELECT type_name
FROM system_schema_mcs.types
WHERE keyspace_name = 'my_keyspace';
```

- 若要檢視特定類型的詳細資訊，您可以使用下列陳述式。

```
SELECT
 keyspace_name,
 type_name,
 field_names,
 field_types,
 max_nesting_depth,
 last_modified_timestamp,
 status,
 direct_referring_tables,
 direct_parent_types
FROM system_schema_mcs.types
WHERE keyspace_name = 'my_keyspace' AND type_name = 'my_udt';
```

- 您可以使用 `DESC TYPE` 列出帳戶中存在的所有 UDTs。

```
DESC TYPES;

Keyspace my_keyspace

my_udt1 my_udt2

Keyspace my_keyspace2

my_udt1
```

- 您可以使用 `DESC TYPE` 列出目前所選金鑰空間中的所有 UDTs。

```
USE my_keyspace;
my_keyspace DESC TYPES;

my_udt1 my_udt2
```

- 若要列出多區域金鑰空間中的所有 UDTs，您可以在 `system_multiregion_info` 金鑰空間 `types` 中查詢系統資料表。下列查詢是此範例。

```
SELECT keyspace_name, type_name, region, status FROM
system_multiregion_info.types WHERE keyspace_name = 'mykeyspace' AND table_name
= 'mytable';
```

此命令的輸出看起來與此類似。



| keyspace_name | table_name | region         | status |
|---------------|------------|----------------|--------|
| mykeyspace    | mytable    | us-east-1      | ACTIVE |
| mykeyspace    | mytable    | ap-southeast-1 | ACTIVE |
| mykeyspace    | mytable    | eu-west-1      | ACTIVE |

## CLI

### 使用 檢視使用者定義類型 (UDTs) AWS CLI

- 若要列出金鑰空間中可用的類型，您可以使用 `list-types` 命令。

```
aws keyspaces list-types
--keyspace-name 'my_keyspace'
```

該命令的輸出看起來與此範例類似。

```
{
 "types": [
 "my_udt",
 "parent_udt"
]
}
```

- 若要檢視指定類型的詳細資訊，您可以使用 `get-type` 命令。

```
aws keyspaces get-type
--type-name 'my_udt'
--keyspace-name 'my_keyspace'
```

此命令的輸出看起來與此範例類似。

```
{
 "keyspaceName": "my_keyspace",
 "typeName": "my_udt",
 "fieldDefinitions": [
 {
 "name": "a",
 "type": "int"
 },
],
}
```

```
{
 "name": "b",
 "type": "text"
},
"lastModifiedTimestamp": 1721328225776,
"maxNestingDepth": 3
"status": "ACTIVE",
"directReferringTables": [],
"directParentTypes": [
 "parent_udt"
],
"keyspaceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
my_keyspace/"
}
```

## 在 Amazon Keyspaces 中刪除使用者定義類型 (UDT)

若要刪除金鑰空間中的 UDT，您可以在 CQL、`delete-type` 命令與 AWS CLI 或主控台中使用 `DROP TYPE` 陳述式。

### Console

使用 Amazon Keyspaces 主控台刪除使用者定義類型 (UDT)

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇金鑰空間，然後從清單中選擇金鑰空間。
3. 選擇 UDTs 標籤。
4. 選擇您要刪除的 UDT。在 中使用的 上，您可以確認資料表或其他 UDT 目前未使用您要刪除的類型。
5. 選擇摘要上方的刪除。
6. Delete 在顯示的對話方塊中輸入，然後選擇刪除 UDT。

### Cassandra Query Language (CQL)

使用 CQL 刪除使用者定義類型 (UDT)

- 若要刪除類型，您可以使用下列陳述式。

```
DROP TYPE my_keyspace.my_udt;
```

如需 CQL 語法的詳細資訊，請參閱 [the section called “DROP TYPE”](#)。

## CLI

使用 刪除使用者定義類型 (UDT) AWS CLI

1. 若要刪除類型，您可以使用下列命令。

```
aws keyspaces delete-type
--keyspace-name 'my_keyspace'
--type-name 'my_udt'
```

2. 命令的輸出看起來與此範例類似。

```
{
 "keyspaceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
my_keyspace/",
 "typeName": "my_udt"
}
```

## 在 Amazon Keyspaces 中使用 CQL 查詢

本節介紹如何在 Amazon Keyspaces (適用於 Apache Cassandra) 中使用查詢。可用於查詢、轉換和管理資料的 CQL 陳述式為 SELECT、UPDATE、INSERT 和 DELETE。下列主題概述使用查詢時一些較複雜的可用選項。如需範例的完整語言語法，請參閱 [the section called “DML 陳述式”](#)。

### 主題

- [在 Amazon Keyspaces 的查詢中使用 IN 運算子搭配 SELECT 陳述式](#)
- [在 Amazon Keyspaces 中使用批次陳述式](#)
- [在 Amazon Keyspaces ORDER BY 中使用 排序結果](#)
- [在 Amazon Keyspaces 中分頁結果](#)

## 在 Amazon Keyspaces 的查詢中使用 IN 運算子搭配 SELECT 陳述式

### SELECT IN

您可以使用 SELECT 陳述式從資料表查詢資料，該陳述式會讀取資料表中一或多個資料列的一或多個資料欄，並傳回結果集，其中包含符合請求的資料列。SELECT 陳述式包含 `select_clause`，可決定要讀取哪些資料欄，以及在結果集中傳回哪些資料欄。子句可以包含在傳回資料之前轉換資料的指示。選用 WHERE 的子句指定哪些資料列必須查詢，並由屬於主索引鍵之資料欄上的關係組成。Amazon Keyspaces 支援 WHERE 子句中的 IN 關鍵字。本節使用範例來示範 Amazon Keyspaces 如何使用 IN 關鍵字處理 SELECT 陳述式。

此範例示範 Amazon Keyspaces 如何將具有 IN 關鍵字的 SELECT 陳述式分解為子查詢。在此範例中，我們使用名為 `my_keyspace.customers` 的資料表。資料表有一個主索引鍵資料欄 `department_id`、兩個叢集資料欄 `sales_region_id` 和 `sales_representative_id`，以及一個包含資料欄中客戶名稱的資料欄 `customer_name`。

```
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             |

使用此資料表，您可以執行下列 SELECT 陳述式，使用 WHERE 子句中的 IN 關鍵字來尋找您感興趣的部門和銷售區域中的客戶。下列陳述式是此範例。

```
SELECT * FROM my_keyspace.customers WHERE department_id IN (0, 1) AND sales_region_id IN (0, 1);
```

Amazon Keyspaces 將此陳述式分成四個子查詢，如下列輸出所示。

```
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 會在下列任何情況下自動分頁結果：
- 每處理第 10 個子查詢之後。
  - 處理 1MB 的邏輯 IO 之後。
  - 如果您已設定 PAGE SIZE，Amazon Keyspaces 會在讀取查詢數目後分頁，以根據集合 進行處理PAGE SIZE。
  - 當您使用 LIMIT關鍵字來減少傳回的資料列數時，Amazon Keyspaces 會在根據集合 讀取要處理的查詢數之後進行分頁LIMIT。

下表用於說明 範例。

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 |

您可以在此表格上執行下列陳述式，以查看分頁的運作方式。

```
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 將此陳述式處理為 24 個子查詢，因為此查詢中包含之所有IN詞彙的笛卡兒產品基數為 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         |

此範例示範如何在具有 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             |

子查詢的處理順序是分割區索引鍵和叢集索引鍵資料欄在查詢中呈現的順序。在下列範例中，會先處理分割區索引鍵值 "2" 的子查詢，接著處理分割區索引鍵值 "3" 和 "1" 的子查詢。如果存在，則會根據查詢的排序子句或資料表建立期間定義的資料表叢集順序，來排序指定子查詢的結果。

```
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 中使用批次陳述式

您可以將多個 INSERT、UPDATE 和 DELETE 操作合併為 BATCH 陳述式。LOGGED 批次為預設值。

```
batch_statement ::= BEGIN [UNLOGGED] BATCH
 [USING update_parameter(AND update_parameter)*]
 modification_statement (';' modification_statement)*
```

## APPLY BATCH

```
modification_statement ::= insert_statement | update_statement | delete_statement
```

當您執行批次陳述式時，驅動程式會將批次中的所有陳述式合併為單一批次操作。

若要決定要使用的批次操作類型，您可以考慮下列準則。

在下列情況下使用記錄的批次：

- 您需要原子交易保證。
- 稍微較高的延遲是可接受的取捨。

在下列情況下使用未記錄的批次：

- 您需要最佳化單一分割區操作。
- 您想要減少網路額外負荷。
- 您有高輸送量需求。

如需批次陳述式配額的詳細資訊，請參閱 [配額](#)。

## 未記錄的批次

使用未記錄的批次時，Amazon Keyspaces 會將多個操作視為單一請求來處理，而無需維護批次日誌。透過未記錄的批次操作，有些動作可能會成功，有些則失敗。當您想要：

- 最佳化單一分割區中的操作。
- 透過將相關請求分組來減少網路流量。

未記錄批次的語法類似於已記錄批次的語法，並新增 UNLOGGED 關鍵字。

```
BEGIN UNLOGGED BATCH
 INSERT INTO users (id, firstname, lastname) VALUES (1, 'John', 'Doe');
 INSERT INTO users (id, firstname, lastname) VALUES (2, 'Jane', 'Smith');
APPLY BATCH;
```

## 已記錄的批次

記錄的批次會將多個寫入動作合併為單一原子操作。當您執行記錄的批次時：

- 所有動作會一起成功或一起失敗。



- 操作是同步且等冪的。
- 您可以寫入多個 Amazon Keyspaces 資料表，只要它們位於相同的 AWS 帳戶和 AWS 區域。

記錄的批次可能會有稍微較高的延遲。對於高輸送量應用程式，請考慮使用未記錄的批次。

在 Amazon Keyspaces 中使用記錄批次無需額外費用。您只需為屬於批次操作的寫入付費。Amazon Keyspaces 會執行批次中每一列的兩個基礎寫入：一個用於準備批次的資料列，另一個用於遞交批次。為使用記錄批次的資料表規劃容量時，請記住，批次中的每一列都需要標準寫入操作的兩倍容量。例如，如果您的應用程式每秒執行一個記錄批次，其中包含三個 1KB 資料列，則您需要佈建六個寫入容量單位 (WCUs)，而個別寫入或未記錄批次只需要三個 WCUs。

如需定價的相關資訊，請參閱 [Amazon Keyspaces \(適用於 Apache Cassandra\) 定價](#)。

### 批次操作的最佳實務

使用 Amazon Keyspaces 批次操作時，請考慮下列建議實務。

- 啟用自動擴展，讓您的資料表有足夠的輸送量容量來處理批次操作，以及記錄批次的額外輸送量需求。
- 當操作可以獨立執行時，請使用個別操作或未記錄的批次，而不會影響應用程式的正確性。
- 設計您的應用程式，將相同資料列的並行更新降至最低，因為同時批次操作可能會衝突和失敗。
- 對於沒有原子需求的高輸送量大量資料擷取，請使用個別寫入操作或未記錄的批次。

### 一致性和並行

Amazon Keyspaces 會針對記錄的批次強制執行下列一致性和並行規則：

- 所有批次操作都使用LOCAL\_QUORUM一致性層級。
- 影響不同資料列的並行批次可以同時執行。
- 持續批次中涉及之資料列的並行 INSERTUPDATE、或 DELETE操作會失敗並產生衝突。

### 支援的運算子和條件

支援的WHERE子句運算子：

- 等式 (=)

不支援的運算子：

- 範圍運算子 (>、<、>=、<=)

- IN 運算子
- LIKE 運算子
- BETWEEN 運算子

記錄的批次中不支援：

- 影響相同資料列的多個陳述式
- 計數器操作
- 範圍刪除

已記錄批次陳述式的失敗條件

在下列任何情況下，記錄的批次操作可能會失敗：

- 條件表達式（例如 IF NOT EXISTS 或 IF）會評估為 false。
- 一或多個操作包含無效的參數。
- 請求與在相同資料列上執行的另一個批次操作衝突。
- 資料表缺少足夠的佈建容量。
- 資料列超過大小上限。
- 輸入資料格式無效。

批次陳述式和多區域複寫

在多區域部署中：

- 來源區域操作是同步和原子的。
- 目的地區域操作是非同步的。
- 所有批次操作都會複寫到目的地區域，但可能無法在應用程式期間維持隔離。

監控批次操作

您可以使用 Amazon CloudWatch 指標監控批次操作，以追蹤效能、錯誤和用量模式。Amazon Keyspaces 提供下列 CloudWatch 指標，用於監控每個資料表的批次操作：

- SuccessfulRequestCount – 追蹤成功的批次操作。
- Latency – 測量批次操作效能。

- ConsumedWriteCapacityUnits – 監控批次操作的容量耗用。

如需詳細資訊，請參閱[the section called “Amazon Keyspaces 指標”](#)。

除了 CloudWatch 指標之外，您還可以使用 AWS CloudTrail 記錄所有 Amazon Keyspaces API 動作。批次中的每個 API 動作都會記錄在 CloudTrail 中，讓您更輕鬆地追蹤和稽核 Amazon Keyspaces 資料表中的批次操作。

### 批次操作範例

以下是基本記錄批次陳述式的範例。

```
BEGIN BATCH
 INSERT INTO users (id, firstname, lastname) VALUES (1, 'John', 'Doe');
 INSERT INTO users (id, firstname, lastname) VALUES (2, 'Jane', 'Smith');
APPLY BATCH;
```

這是包含 INSERT、UPDATE 和 DELETE 陳述式的批次範例。

```
BEGIN BATCH
 INSERT INTO users (id, firstname, lastname) VALUES (1, 'John', 'Doe');
 UPDATE users SET firstname = 'Johnny' WHERE id = 2;
 DELETE FROM users WHERE id = 3;
APPLY BATCH;
```

這是使用用戶端時間戳記的批次範例。

```
BEGIN BATCH
 INSERT INTO users (id, firstname, lastname) VALUES (1, 'John', 'Stiles') USING
TIMESTAMP 1669069624;
 INSERT INTO users (id, firstname, lastname) VALUES (2, 'Jane', 'Doe') USING
TIMESTAMP 1669069624;
APPLY BATCH;

BEGIN BATCH
 UPDATE users USING TIMESTAMP 1669069624 SET firstname = 'Carlos' WHERE id = 1;
 UPDATE users USING TIMESTAMP 1669069624 SET firstname = 'Diego' WHERE id = 2;
APPLY BATCH;
```

這是條件式批次的範例。

```
BEGIN BATCH
 INSERT INTO users (id, firstname, lastname) VALUES (1, 'Jane', 'Doe') IF NOT
 EXISTS;
 INSERT INTO users (id, firstname, lastname) VALUES (2, 'John', 'Doe') IF NOT
 EXISTS;
APPLY BATCH;
```

```
BEGIN BATCH
 UPDATE users SET lastname = 'Stiles' WHERE id = 1 IF lastname = 'Doe';
 UPDATE users SET lastname = 'Stiles' WHERE id = 2 IF lastname = 'Doe';
APPLY BATCH;
```

這是使用存留時間 (TTL) 的批次範例。

```
BEGIN BATCH
 INSERT INTO users (id, firstname, lastname) VALUES (1, 'John', 'Doe') USING TTL
 3600;
 INSERT INTO users (id, firstname, lastname) VALUES (2, 'Jane', 'Smith') USING TTL
 7200;
APPLY BATCH;
```

這是更新多個資料表的批次陳述式範例。

```
BEGIN BATCH
 INSERT INTO users (id, firstname) VALUES (1, 'John');
 INSERT INTO user_emails (user_id, email) VALUES (1, 'john@example.com');
APPLY BATCH;
```

這是使用使用者定義類型 (UDTs) 的批次操作範例。此範例假設 UDT address 存在。

```
BEGIN BATCH
 INSERT INTO users (id, firstname, address)
 VALUES (1, 'John', {street: '123 Main St', city: 'NYC', zip: '10001'});
 INSERT INTO users (id, firstname, address)
 VALUES (2, 'Jane', {street: '456 Oak Ave', city: 'LA', zip: '90210'});
APPLY BATCH;

BEGIN BATCH
 UPDATE users SET address.zip = '10002' WHERE id = 1;
 UPDATE users SET address.city = 'Boston' WHERE id = 2;
```

```
APPLY BATCH;
```

## 在 Amazon Keyspaces **ORDER BY**中使用 排序結果

**ORDER BY** 子句指定SELECT陳述式中傳回之結果的排序順序。陳述式會以引數的形式取得資料欄名稱清單，而且您可以為每個資料欄指定資料的排序順序。您只能依子句的順序指定叢集資料欄，不允許非叢集資料欄。

傳回結果的兩個可用排序順序選項ASC適用於遞增和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    |

如果您未在查詢陳述式中指定排序順序，則會使用叢集資料欄的預設排序。

您可以在排序子句中使用的可能排序順序取決於資料表建立時指派給每個叢集欄的排序順序。查詢結果只能依照資料表建立時為所有叢集資料欄定義的順序，或反向定義的排序順序進行排序。不允許其他可能的組合。

例如，如果資料表的 **CLUSTERING ORDER**是 (col1 ASC、col2 DESC、col3 ASC)，則的有效參數**ORDER BY**是 (col1 ASC、col2 DESC、col3 ASC) 或 (col1 DESC、col2 ASC、col3 DESC)。如

需的詳細資訊 CLUSTERING ORDER，請參閱 `table_options` 下的 [the section called “CREATE TABLE”](#)。

## 在 Amazon Keyspaces 中分頁結果

當資料讀取處理 SELECT 陳述式超過 1 MB 時，Amazon Keyspaces 會自動從陳述式分頁結果。SELECT 使用分頁時，SELECT 陳述式結果會分為大小為 1 MB（或更少）的「頁面」資料。應用程式可以處理結果的第一頁、第二頁，以此類推。用戶端在處理傳回多列的 SELECT 查詢時，應一律檢查分頁字符。

如果用戶端提供的 PAGE SIZE 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成多個頁面。

例如，如果資料列的平均大小為 100 KB，而您指定的 PAGE SIZE 為 20，Amazon Keyspaces 會在讀取 10 列（1000 KB 的資料讀取）之後自動分頁資料。

由於 Amazon Keyspaces 會根據讀取處理請求的資料列數來分頁結果，而不是結果集中傳回的資料列數，因此如果您執行篩選的查詢，某些頁面可能不包含任何資料列。

例如，如果您將 PAGE SIZE 設定為 10，且 Keyspaces 評估 30 列來處理 SELECT 查詢，Amazon Keyspaces 將傳回三個頁面。如果只有一部分的資料列符合您的查詢，有些頁面可能少於 10 個資料列。如需 LIMIT 查詢 PAGE SIZE 如何影響讀取容量的範例，請參閱 [the section called “預估限制查詢的讀取容量耗用量”](#)。

如需 Apache Cassandra 分頁的比較，請參閱 [the section called “分頁”](#)。

## 在 Amazon Keyspaces 中使用變更資料擷取 (CDC) 串流

Amazon Keyspaces 變更資料擷取 (CDC) 會以近乎即時的方式記錄來自 Amazon Keyspaces 資料表的資料列層級變更事件。

Amazon Keyspaces CDC 可啟用事件驅動的使用案例，例如工業 IoT 和詐騙偵測，以及資料處理使用案例，例如全文搜尋和資料封存。Amazon Keyspaces CDC 在串流中擷取的變更事件可供執行關鍵業務功能的下游應用程式取用，例如資料分析、文字搜尋、ML 訓練/推論，以及封存的持續資料備份。例如，您可以將串流資料傳輸到 Amazon OpenSearch Service、Amazon Redshift 和 Amazon S3 等 AWS 分析和儲存服務，以進行進一步處理。

Amazon Keyspaces CDC 為資料表提供按時間排序和已取消重複的變更記錄，資料輸送量的自動擴展和保留時間最多可達 24 小時。

Amazon Keyspaces CDC 串流完全無伺服器，您不需要管理資料基礎設施來擷取變更事件。此外，Amazon Keyspaces CDC 不會使用運算或儲存體的任何資料表容量。如需詳細資訊，請參閱[the section called “CDC 串流的運作方式”](#)。

您可以使用 Amazon Keyspaces Streams API 來建置使用 Amazon Keyspaces CDC 串流的應用程式，並根據內容採取行動。如需可用的端點，請參閱 [the section called “存取 CDC 串流端點”](#)。

如需串流 API 中可用於 Amazon Keyspaces 的所有操作的完整清單，請參閱 [Amazon Keyspaces Streams API 參考](#)。

## 主題

- [變更資料擷取 \(CDC\) 串流如何在 Amazon Keyspaces 中運作](#)
- [如何在 Amazon Keyspaces 中使用變更資料擷取 \(CDC\) 串流](#)

## 變更資料擷取 (CDC) 串流如何在 Amazon Keyspaces 中運作

本節概述變更資料擷取 (CDC) 串流如何在 Amazon Keyspaces 中運作。

Amazon Keyspaces 變更資料擷取 (CDC) 會在 Amazon Keyspaces 資料表中記錄一系列列層級修改，並將此資訊存放在稱為串流的日誌中長達 24 小時。每個資料列層級修改都會產生新的 CDC 記錄，其中包含主索引鍵資料欄資訊，以及包含所有資料欄的資料列的「之前」和「之後」狀態。應用程式可以近乎即時地存取串流並檢視變動。

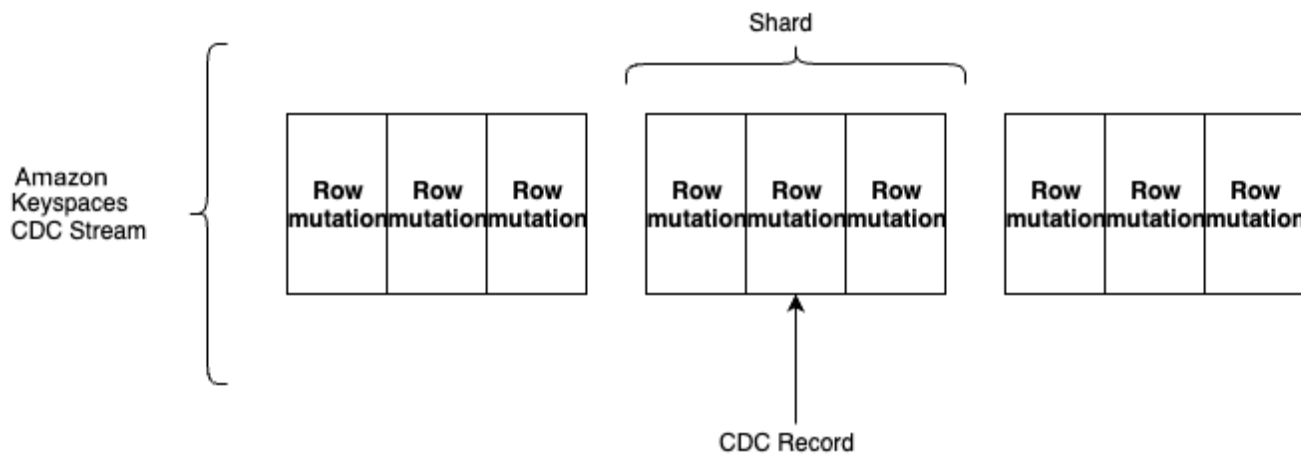
當您在資料表上啟用 CDC 時，Amazon Keyspaces 會建立新的 CDC 串流，並開始擷取資料表中每個修改的相關資訊。CDC 串流具有下列格式的 Amazon Resource Name (ARN)：

```
arn:${Partition}:cassandra:${Region}:${Account}:/keyspace/${keyspaceName}/table/${tableName}/stream/${streamLabel}
```

您可以在第一次啟用 CDC 串流時，選取 CDC 串流為每個記錄收集的資訊類型或檢視類型。您之後無法變更串流的檢視類型。Amazon Keyspaces 支援下列檢視類型：

- NEW\_AND\_OLD\_IMAGES – 擷取變動前後的資料列版本。這是預設值。
- NEW\_IMAGE – 擷取變動之後的資料列版本。
- OLD\_IMAGE – 擷取變動前的資料列版本。
- KEYS\_ONLY – 擷取已變動資料列的分割區和叢集索引鍵。

每個 CDC 串流都包含記錄。每個記錄代表 Amazon Keyspaces 資料表中的單一資料列修改。記錄會以邏輯方式組織成稱為碎片的群組。這些群組會依主索引鍵的範圍（分割區索引鍵的組合、叢集索引鍵範圍）進行邏輯組織，並且是 Amazon Keyspaces 的內部建構。每個碎片都充當多個記錄的容器，並包含透過這些記錄存取和迭代所需的資訊。



每個 CDC 記錄都會獲指派序號，以反映記錄在碎片中發佈的順序。序列號保證在每個碎片中增加且是唯一的。

Amazon Keyspaces 會自動建立和刪除碎片。根據流量負載，Amazon Keyspaces 也可以隨著時間分割或合併碎片。例如，Amazon Keyspaces 可以將一個碎片分割成多個新碎片，或將碎片合併成新的單一碎片。Amazon Keyspaces APIs 會發佈碎片和 CDC 串流資訊，以允許取用應用程式存取碎片的整個譜系圖，以正確順序處理記錄。

Amazon Keyspaces CDC 是以下列原則為基礎，您可以在建置應用程式時倚賴這些原則：

- 每個資料列層級變動記錄只會在 CDC 串流中出現一次。
- 當您依歷程順序使用碎片時，每個資料列層級變動記錄會以與主索引鍵上實際變動順序相同的順序顯示。

## 主題

- [Amazon Keyspaces 中 CDC 串流的資料保留運作方式](#)
- [存留時間 \(TTL\) 資料過期如何與 Amazon Keyspaces 中的 CDC 串流搭配使用](#)
- [Amazon Keyspaces 中 CDC 串流的批次操作運作方式](#)
- [靜態資料欄如何在 Amazon Keyspaces 的 CDC 串流中運作](#)
- [靜態加密對 Amazon Keyspaces 中 CDC 串流的運作方式](#)



- [多區域複寫如何適用於 Amazon Keyspaces 中的 CDC 串流](#)
- [CDC 串流和與 AWS 服務的整合](#)

## Amazon Keyspaces 中 CDC 串流的資料保留運作方式

Amazon Keyspaces 會保留 CDC 串流中的記錄 24 小時。您無法變更保留期間。如果您在資料表上停用 CDC，串流中的資料會繼續讀取 24 小時。在此期間之後，資料會過期，並自動刪除記錄。

## 存留時間 (TTL) 資料過期如何與 Amazon Keyspaces 中的 CDC 串流搭配使用

Amazon Keyspaces 會顯示資料欄/儲存格層級的過期時間，以及 CDC 變更記錄中稱為 `expirationTime` 的中繼資料欄位中的資料列層級。當 Amazon Keyspaces TTL 偵測到儲存格過期時，CDC 會建立新的變更記錄，顯示 TTL 為變更的來源。如需 TTL 的詳細資訊，請參閱 [the section called “使用存留時間過期資料”](#)。

## Amazon Keyspaces 中 CDC 串流的批次操作運作方式

批次操作在內部分為個別的資料列層級修改。Amazon Keyspaces 會在資料列層級保留 CDC 串流中的所有記錄，即使修改發生在批次操作中。Amazon Keyspaces 會以與資料列層級或主索引鍵上發生的變動順序相同的順序，維護 CDC 串流內的記錄順序。

## 靜態資料欄如何在 Amazon Keyspaces 的 CDC 串流中運作

靜態資料欄值會在 Cassandra 中分割區中的所有資料列之間共用。由於此行為，Amazon Keyspaces 會將靜態資料欄的任何更新擷取為 CDC 串流中的個別記錄。下列範例摘要說明靜態資料欄變動的行為：

- 當僅更新靜態資料欄時，CDC 串流包含靜態資料欄的資料列修改作為資料列中的唯一資料欄。
- 更新資料列而不變更靜態資料欄時，CDC 串流會包含資料列修改，其中包含靜態資料欄以外的所有資料欄。
- 當資料列與靜態資料欄一起更新時，CDC 串流包含兩個不同的資料列修改，一個用於靜態資料欄，另一個用於其餘資料列。

## 靜態加密對 Amazon Keyspaces 中 CDC 串流的運作方式

若要加密 CDC 排序日誌中的靜態資料，Amazon Keyspaces 會使用已用於資料表的相同加密金鑰。如需靜態加密的詳細資訊，請參閱 [the section called “靜態加密”](#)。

## 多區域複寫如何適用於 Amazon Keyspaces 中的 CDC 串流

您可以使用 `update-table` API 或 `ALTER TABLE CQL` 命令，為多區域資料表的個別複本啟用和停用 CDC 串流。由於非同步複寫和衝突解決，多區域資料表的 CDC 串流跨區域不一致 AWS 區域。因此，Amazon Keyspaces 在串流中擷取的記錄可能會在不同的區域中以不同的順序顯示。

如需多區域複寫的詳細資訊，請參閱 [the section called “多區域複製”](#)。

## CDC 串流和與 AWS 服務的整合

### 如何在 Amazon Keyspaces 中使用 CDC 串流的 VPC 端點

您可以使用 VPC 端點來存取 Amazon Keyspaces CDC 串流。如需如何建立和存取串流 VPC 端點的詳細資訊，請參閱 [the section called “搭配串流使用介面 VPC 端點”](#)。

### 使用 CloudWatch 監控 如何在 Amazon Keyspaces 中用於 CDC 串流

您可以使用 Amazon CloudWatch 來監控對 Amazon Keyspaces CDC 端點發出的 API 呼叫。如需可用指標的詳細資訊，請參閱 [the section called “Amazon Keyspaces 變更資料擷取 \(CDC\) 的指標”](#)。

### 使用 CloudTrail 記錄日誌 如何在 Amazon Keyspaces 中用於 CDC 串流

Amazon Keyspaces CDC 已與服務整合 AWS CloudTrail，此服務提供由 Amazon Keyspaces AWS 中的使用者、角色或服務所採取之動作的記錄。CloudTrail 會將 Amazon Keyspaces 的 Data Definition Language (DDL) API 呼叫和 Data Manipulation Language (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 操作的程式設計呼叫。

如需 CloudTrail 擷取之 CDC 事件的詳細資訊，請參閱 [the section called “使用 CloudTrail 進行記錄”](#)。

### 標記在 Amazon Keyspaces 中如何適用於 CDC 串流

Amazon Keyspaces CDC 串流是可標記的資源。您可以在使用 CQL、AWS SDK 或以程式設計方式建立資料表時標記串流 AWS CLI。您也可以標記現有的串流、刪除標籤或檢視串流的標籤。如需詳細資訊，請參閱 [the section called “標記鍵空間、資料表和串流”](#)。

## 如何在 Amazon Keyspaces 中使用變更資料擷取 (CDC) 串流

### 主題

- [設定許可以在 Amazon Keyspaces 中使用 CDC 串流](#)

- [如何在 Amazon Keyspaces 中存取 CDC 串流端點](#)
- [在 Amazon Keyspaces 中建立新資料表時啟用 CDC 串流](#)
- [為 Amazon Keyspaces 中的現有資料表啟用 CDC 串流](#)
- [在 Amazon Keyspaces 中停用 CDC 串流](#)
- [在 Amazon Keyspaces 中檢視 CDC 串流](#)
- [在 Amazon Keyspaces 中存取 CDC 串流中的記錄](#)
- [使用 Kinesis Client Library \(KCL\) 來處理 Amazon Keyspaces 串流](#)

## 設定許可以在 Amazon Keyspaces 中使用 CDC 串流

若要啟用 CDC 串流，委託人，例如 IAM 使用者或角色，需要下列許可。

如需的詳細資訊 AWS Identity and Access Management，請參閱 [the section called “AWS Identity and Access Management”](#)。

### 為資料表啟用 CDC 串流的許可

若要啟用 Amazon Keyspaces 資料表的 CDC 串流，主體首先需要建立或修改資料表的許可，然後需要第二個許可來建立服務連結角色 [AWSServiceRoleForAmazonKeyspacesCDC](#)。Amazon Keyspaces 會使用服務連結角色，代表您將 CloudWatch 指標發佈至您的帳戶

下列 IAM 政策為範例。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "cassandra:Create",
 "cassandra:CreateMultiRegionResource",
 "cassandra:Alter",
 "cassandra:AlterMultiRegionResource"
],
 "Resource": [
 "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/*",
 "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
]
 }
]
}
```

```

 },
 {
 "Sid": "KeyspacesCDCServiceLinkedRole",
 "Effect": "Allow",
 "Action": "iam:CreateServiceLinkedRole",
 "Resource": "arn:aws:iam::*:role/aws-service-role/cassandra-
streams.amazonaws.com/AWSServiceRoleForAmazonKeyspacesCDC",
 "Condition": {
 "StringLike": {
 "iam:AWSServiceName": "cassandra-streams.amazonaws.com"
 }
 }
 }
]
}

```

若要停用串流，只需要ALTER TABLE許可。

### 檢視 CDC 串流的許可

若要檢視或列出 CDC 串流，主體需要系統金鑰空間的讀取許可。如需詳細資訊，請參閱[the section called “system\\_schema\\_mcs”](#)。

下列 IAM 政策為範例。

```

{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": "cassandra:Select",
 "Resource": [
 "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
]
 }
]
}

```

若要使用或 Amazon Keyspaces API 檢視 AWS CLI 或列出 CDC 串流，委託人需要動作 `cassandra:ListStreams` 和 的額外許可 `cassandra:GetStream`。

下列 IAM 政策為範例。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "cassandra:Select",
 "cassandra:ListStreams",
 "cassandra:GetStream"
],
 "Resource": "*"
 }
]
}
```

## 讀取 CDC 串流的許可

若要讀取 CDC 串流，委託人需要下列許可。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "cassandra:GetStream",
 "cassandra:GetShardIterator",
 "cassandra:GetRecords"
],
 "Resource": [
 "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/
table/my_table/stream/stream_label"
]
 }
]
}
```

使用 Kinesis Client Library (KCL) 處理 Amazon Keyspaces CDC 串流的許可

若要使用 KCL 處理 Amazon Keyspaces CDC 串流，IAM 主體需要下列許可。

- Amazon Keyspaces – 對指定 Amazon Keyspaces CDC 串流的唯讀存取。

- DynamoDB – 建立 shard lease 資料表、讀取和寫入資料表存取權，以及視需要讀取索引以進行 KCL 串流處理的許可。
- CloudWatch – 將 Amazon Keyspaces CDC 串流處理中的指標資料發佈到 CloudWatch 帳戶中 KCL 用戶端應用程式命名空間的許可。如需監控的詳細資訊，請參閱[使用 Amazon CloudWatch 監控 Kinesis 用戶端程式庫](#)。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "cassandra:GetStream",
 "cassandra:GetShardIterator",
 "cassandra:GetRecords"
],
 "Resource": [
 "arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/
table/my_table/stream/stream_label"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "dynamodb:CreateTable",
 "dynamodb:DescribeTable",
 "dynamodb:UpdateTable",
 "dynamodb:GetItem",
 "dynamodb:UpdateItem",
 "dynamodb:PutItem",
 "dynamodb>DeleteItem",
 "dynamodb:Scan"
],
 "Resource": [
 "arn:aws:dynamodb:us-east-1:111122223333:table/KCL_APPLICATION_NAME"
]
 },
 {
 "Effect": "Allow",
 "Action": [
 "dynamodb:CreateTable",
```

```

 "dynamodb:DescribeTable",
 "dynamodb:GetItem",
 "dynamodb:UpdateItem",
 "dynamodb:PutItem",
 "dynamodb>DeleteItem",
 "dynamodb:Scan"
],
 "Resource": [
 "arn:aws:dynamodb:us-east-1:111122223333:table/KCL_APPLICATION_NAME-WorkerMetricStats",
 "arn:aws:dynamodb:us-east-1:111122223333:table/KCL_APPLICATION_NAME-CoordinatorState"
]
},
{
 "Effect": "Allow",
 "Action": [
 "dynamodb:Query"
],
 "Resource": [
 "arn:aws:dynamodb:us-east-1:111122223333:table/KCL_APPLICATION_NAME/index/*"
]
},
{
 "Effect": "Allow",
 "Action": [
 "cloudwatch:PutMetricData"
],
 "Resource": "*"
}
]
}

```

## 如何在 Amazon Keyspaces 中存取 CDC 串流端點

Amazon Keyspaces 會在提供 Amazon Keyspaces 的每個 AWS 區域 中維護金鑰空間/資料表和 CDC 串流的個別[端點](#)。若要存取 CDC 串流，請選取資料表的區域，並在端點名稱cassandra-streams中將cassandra字首取代為，如下列範例所示：

| 區域名稱             | 區域        | 金鑰空間和資料表的端點                         | 串流的端點                                       |
|------------------|-----------|-------------------------------------|---------------------------------------------|
| 美國東部<br>(維吉尼亞北部) | us-east-1 | <i>cassandra</i> .us-east-1.api.aws | <i>cassandra-streams</i> .us-east-1.api.aws |

下表包含適用於 的可用公有端點的完整清單 Amazon Keyspaces change data capture streams。同時 Amazon Keyspaces CDC streams 支援 IPv4 和 IPv6。所有公有端點，例如 *cassandra-streams.us-east-1.api.aws*，都是可針對 IPv4 和 IPv6 設定的雙堆疊端點。

| 區域名稱             | 區域         | 端點                                   | 通訊協定  |
|------------------|------------|--------------------------------------|-------|
| 美國東部<br>(俄亥俄)    | us-east-2  | cassandra-streams.us-east-2.api.aws  | HTTPS |
| 美國東部<br>(維吉尼亞北部) | us-east-1  | cassandra-streams.us-east-1.api.aws  | HTTPS |
| 美國西部<br>(加州北部)   | us-west-1  | cassandra-streams.us-west-1.api.aws  | HTTPS |
| 美國西部<br>(奧勒岡)    | us-west-2  | cassandra-streams.us-west-2.api.aws  | HTTPS |
| 非洲 (開普敦)         | af-south-1 | cassandra-streams.af-south-1.api.aws | HTTPS |
| 亞太區域<br>(香港)     | ap-east-1  | cassandra-streams.ap-east-1.api.aws  | HTTPS |
| 亞太區域<br>(孟買)     | ap-south-1 | cassandra-streams.ap-south-1.api.aws | HTTPS |



| 區域名稱          | 區域             | 端點                                       | 通訊協定  |  |
|---------------|----------------|------------------------------------------|-------|--|
| 亞太區域<br>(首爾)  | ap-northeast-2 | cassandra-streams.ap-northeast-2.api.aws | HTTPS |  |
| 亞太區域<br>(新加坡) | ap-southeast-1 | cassandra-streams.ap-southeast-1.api.aws | HTTPS |  |
| 亞太區域<br>(雪梨)  | ap-southeast-2 | cassandra-streams.ap-southeast-2.api.aws | HTTPS |  |
| 亞太區域<br>(東京)  | ap-northeast-1 | cassandra-streams.ap-northeast-1.api.aws | HTTPS |  |
| 加拿大<br>(中部)   | ca-central-1   | cassandra-streams.ca-central-1.api.aws   | HTTPS |  |
| 歐洲 (法蘭克福)     | eu-central-1   | cassandra-streams.eu-central-1.api.aws   | HTTPS |  |
| 歐洲 (愛爾蘭)      | eu-west-1      | cassandra-streams.eu-west-1.api.aws      | HTTPS |  |
| 歐洲 (倫敦)       | eu-west-2      | cassandra-streams.eu-west-2.api.aws      | HTTPS |  |
| 歐洲 (巴黎)       | eu-west-3      | cassandra-streams.eu-west-3.api.aws      | HTTPS |  |
| 歐洲 (斯德哥爾摩)    | eu-north-1     | cassandra-streams.eu-north-1.api.aws     | HTTPS |  |
| 中東 (巴林)       | me-south-1     | cassandra-streams.me-south-1.api.aws     | HTTPS |  |

| 區域名稱                | 區域            | 端點                                      | 通訊協定  |
|---------------------|---------------|-----------------------------------------|-------|
| 中東 (阿拉伯聯合大公國)       | me-central-1  | cassandra-streams.me-central-1.api.aws  | HTTPS |
| 南美洲 (聖保羅)           | sa-east-1     | cassandra-streams.sa-east-1.api.aws     | HTTPS |
| AWS GovCloud (美國東部) | us-gov-east-1 | cassandra-streams.us-gov-east-1.api.aws | HTTPS |
| AWS GovCloud (美國西部) | us-gov-west-1 | cassandra-streams.us-gov-west-1.api.aws | HTTPS |

## 在 Amazon Keyspaces 中建立新資料表時啟用 CDC 串流

若要在建立資料表時啟用 CDC 串流，您可以在 CQL 或 `create-table` 命令中使用 `CREATE TABLE` 陳述式與 AWS CLI。

對於資料表中每個變更的資料列，Amazon Keyspaces 可以根據您 `cdc_specification` 選取的 `view_type` 的 擷取下列變更：

- `NEW_AND_OLD_IMAGES` – 變更前後的兩個資料列版本。這是預設值。
- `NEW_IMAGE` – 變更後的列版本。
- `OLD_IMAGE` – 變更前的資料列版本。
- `KEYS_ONLY` – 已變更之資料列的分割區和叢集索引鍵。

如需如何標記串流的詳細資訊，請參閱 [the section called “使用新資料表建立具有標籤的串流”](#)。

**Note**

Amazon Keyspaces CDC 需要存在服務連結角色 (AWSServiceRoleForAmazonKeyspacesCDC)，以代表您將指標資料從 Amazon Keyspaces CDC 串流發佈到 CloudWatch 帳戶中 "cloudwatch:namespace": "AWS/Cassandra" 的。系統會自動建立此角色。如需詳細資訊，請參閱 [the section called “CDC 串流”](#)。

## Cassandra Query Language (CQL)

## 使用 CQL 建立資料表時啟用 CDC 串流

1. 

```
CREATE TABLE mykeyspace.mytable (a text, b text, PRIMARY KEY(a))
WITH CUSTOM_PROPERTIES={'cdc_specification': {'view_type': 'NEW_IMAGE'}} AND CDC
= TRUE;
```
2. 若要確認串流設定，您可以使用下列陳述式。

```
SELECT keyspace_name, table_name, cdc, custom_properties FROM
system_schema_mcs.tables WHERE keyspace_name = 'mykeyspace' AND table_name =
'mytable';
```

該陳述式的輸出看起來應該與此類似。

```
SELECT keyspace_name, table_name, cdc, custom_properties FROM
system_schema_mcs.tables WHERE keyspace_name = 'mykeyspace' AND table_name =
'mytable';keyspace_name | table_name | cdc | custom_properties
-----+-----+-----+-----
+-----+-----+-----+-----
mykeyspace | mytable | True | {'capacity_mode':
{'last_update_to_pay_per_request_timestamp': '1741383893782',
'throughput_mode': 'PAY_PER_REQUEST'}, 'cdc_specification':
{'latest_stream_arn': 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/
mykeyspace/table/mytable/stream/2025-03-07T21:44:53.783', 'status': 'ENABLED',
'view_type': 'NEW_IMAGE'}, 'encryption_specification': {'encryption_type':
'AWS_OWNED_KMS_KEY'}, 'point_in_time_recovery': {'status': 'disabled'}}>
```

## CLI

當您使用 建立資料表時，啟用 CDC 串流 AWS CLI

1. 若要建立串流，您可以使用下列語法。

```
aws keyspaces create-table \
--keyspace-name 'mykeyspace' \
--table-name 'mytable' \
--schema-definition 'allColumns=[{name=a,type=text},
{name=b,type=text}],partitionKeys=[{name=a}]' \
--cdc-specification status=ENABLED,viewType=NEW_IMAGE
```

2. 該命令的輸出會顯示標準create-table回應，看起來與此範例類似。

```
{ "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/
table/mytable" }
```

## 為 Amazon Keyspaces 中的現有資料表啟用 CDC 串流

若要為現有資料表啟用 CDC 串流，您可以在 CQL 中使用 ALTER TABLE 陳述式、將 update-table 命令與 搭配使用 AWS CLI，也可以使用 主控台。

對於資料表中每個變更的資料列，Amazon Keyspaces 可以根據您cdc\_specification選取的view\_type的 擷取下列變更：

- NEW\_AND\_OLD\_IMAGES – 變更前後的兩個資料列版本。這是預設值。
- NEW\_IMAGE – 變更後的列版本。
- OLD\_IMAGE – 變更前的資料列版本。
- KEYS\_ONLY – 已變更之資料列的分割區和叢集索引鍵。

如需如何標記串流的詳細資訊，請參閱 [the section called “將新標籤新增至串流”](#)。

### Note

Amazon Keyspaces CDC 需要存在服務連結角色 (AWSServiceRoleForAmazonKeyspacesCDC)，以代表您將指標資料從 Amazon Keyspaces CDC 串流發佈到 CloudWatch 帳戶中"cloudwatch:namespace": "AWS/

Cassandra"的。系統會自動建立此角色。如需詳細資訊，請參閱[the section called “CDC 串流”](#)。

## Cassandra Query Language (CQL)

### 使用 CQL 啟用串流 (CDC 串流)

您可以使用 ALTER TABLE 為現有資料表啟用串流。

1. 下列範例會建立僅擷取變更資料列之分割區和叢集索引鍵變更的串流。

```
ALTER TABLE mykeyspace.mytable
WITH cdc = TRUE
AND CUSTOM_PROPERTIES={'cdc_specification': {'view_type': 'KEYS_ONLY'}};
```

2. 若要驗證串流設定，您可以使用下列陳述式。

```
SELECT keyspace_name, table_name, cdc, custom_properties FROM
system_schema_mcs.tables WHERE keyspace_name = 'mykeyspace' AND table_name =
'mytable';
```

陳述式的輸出看起來與此類似。

```
keyspace_name | table_name | cdc | custom_properties
-----+-----+-----+-----
+-----+-----+-----+-----
mykeyspace | mytable | True | {'capacity_mode':
{'last_update_to_pay_per_request_timestamp': '1741385897045',
'throughput_mode': 'PAY_PER_REQUEST'}, 'cdc_specification':
{'latest_stream_arn': 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/
mykeyspace/table/mytable/stream/2025-03-07T22:20:10.454', 'status': 'ENABLED',
'view_type': 'KEYS_ONLY'}, 'encryption_specification': {'encryption_type':
'AWS_OWNED_KMS_KEY'}, 'point_in_time_recovery': {'status': 'disabled'}}
```

## CLI

### 使用 建立 CDC 串流 AWS CLI

1. 若要為現有資料表建立串流，您可以使用下列語法。

```
aws keyspaces update-table \
--keyspace-name 'mykeyspace' \
--table-name 'mytable' \
--cdc-specification status=ENABLED,viewType=NEW_AND_OLD_IMAGES
```

2. 該命令的輸出會顯示標準create-table回應，看起來與此範例類似。

```
{ "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/
table/mytable" }
```

## Console

使用 Amazon Keyspaces 主控台啟用 CDC 串流

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇資料表，然後從清單中選擇資料表。
3. 選擇串流索引標籤。
4. 選擇編輯以啟用串流。
5. 選取開啟串流。
6. 選擇串流的檢視類型。以下是可用的選項。請注意，您無法在建立串流之後變更串流的檢視類型。
  - 新舊映像 – Amazon Keyspaces 會擷取變更前後的兩個資料列版本。這是預設值。
  - 新映像 – Amazon Keyspaces 只會擷取變更後的資料列版本。
  - 舊映像 – Amazon Keyspaces 只會擷取變更前的資料列版本。
  - 僅限主索引鍵 – Amazon Keyspaces 只會擷取已變更資料列的分割區和叢集索引鍵資料欄。
7. 若要完成，請選擇儲存變更。

## 在 Amazon Keyspaces 中停用 CDC 串流

若要停用金鑰空間中的 CDC 串流，您可以在 CQL、update-table 命令與 AWS CLI 或 主控台中使用 ALTER TABLE 陳述式。

## Cassandra Query Language (CQL)

### 使用 CQL 停用串流 (CDC 串流)

- 若要停用串流，您可以使用下列陳述式。

```
ALTER TABLE mykeyspace.mytable
WITH cdc = FALSE;
```

- 若要確認串流已停用，您可以使用下列陳述式。

```
SELECT keyspace_name, table_name, cdc, custom_properties FROM
system_schema_mcs.tables WHERE keyspace_name = 'mykeyspace' AND table_name =
'mytable';
```

該陳述式的輸出看起來與此類似。

```
keyspace_name | table_name | cdc | custom_properties
-----+-----+-----+-----
+-----+-----+-----+-----
mykeyspace | mytable | False | {'capacity_mode':
{'last_update_to_pay_per_request_timestamp': '1741385668642',
'throughput_mode': 'PAY_PER_REQUEST'}, 'encryption_specification':
{'encryption_type': 'AWS_OWNED_KMS_KEY'}, 'point_in_time_recovery': {'status':
'disabled'}}
```

## CLI

### 使用 停用串流 (CDC 串流) AWS CLI

- 若要停用串流，您可以使用下列命令。

```
aws keyspaces update-table \
--keyspace-name 'mykeyspace' \
--table-name 'mytable' \
--cdc-specification status=DISABLED
```

- 命令的輸出看起來與此範例類似。

```
{
```

```
"keyspaceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
my_keyspace/",
"streamName": "my_stream"
}
```

## Console

使用 Amazon Keyspaces 主控台停用串流 (CDC 串流 )

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇資料表，然後從清單中選擇資料表。
3. 選擇串流索引標籤。
4. 選擇編輯。
5. 取消選取開啟串流。
6. 選擇儲存變更以停用串流。

## 在 Amazon Keyspaces 中檢視 CDC 串流

若要檢視或列出金鑰空間中的所有串流，您可以使用 CQL `system_schema_mcs.streams` 中的陳述式查詢系統金鑰空間中的資料表，或使用 `get-stream` 和 `list-stream` 命令搭配 AWS CLI、或 主控台。

如要了解必要的許可，請參閱 [the section called “設定 許可”](#)。

## Cassandra Query Language (CQL)

使用 CQL 檢視 CDC 串流

- 若要監控資料表的 CDC 狀態，您可以使用下列陳述式。

```
SELECT custom_properties
FROM system_schema_mcs.tables
WHERE keyspace_name='my_keyspace' and table_name='my_table';
```

命令的輸出看起來與此類似。

...



```
custom_properties

{'cdc_specification':{'status': 'Enabled', 'view_type': 'NEW_IMAGE',
 'latest_stream_arn': 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/
my_keyspace/table/my_table/stream/stream_label'}}
...
```

## CLI

### 使用 檢視 CDC 串流 AWS CLI

1. 此範例說明如何查看資料表的串流資訊。

```
aws keyspaces get-table \
--keyspace-name 'my_keyspace' \
--table-name 'my_table'
```

命令的輸出如下所示。

```
{
 "keyspaceName": "my_keyspace",
 "tableName": "my_table",
 ... Other fields ...,
 "latestStreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
my_keyspace/table/my_table/stream/stream_label",
 "cdcSpecification": {
 "status": "ENABLED",
 "viewType": "NEW_AND_OLD_IMAGES"
 }
}
```

2. 您可以在指定的 中列出帳戶中的所有串流 AWS 區域。下列命令是此範例。

```
aws keyspacestreams list-streams --region us-east-1
```

命令的輸出可能看起來像這樣。

```
{
 "Streams": [
 {
```

```

 "StreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
ks_1/table/t1/stream/2023-05-11T21:21:33.291",
 "StreamLabel": "2023-05-11T21:21:33.291",
 "KeyspaceName": "ks_1"
 "TableName": "t1",
 },
 {
 "StreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
ks_1/table/t2/stream/2023-05-11T21:21:33.291",
 "StreamLabel": "2023-05-11T21:21:33.291",
 "KeyspaceName": "ks_1"
 "TableName": "t2",
 },
 {
 "StreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
ks_2/table/t1/stream/2023-05-11T21:21:33.291",
 "StreamLabel": "2023-05-11T21:21:33.291",
 "KeyspaceName": "ks_3"
 "TableName": "t1",
 }
]
}

```

Create a keyspace with the name catalog. Note that streams are not supported in multi-Region keyspaces.

3. 您也可以使用下列參數列出指定金鑰空間的 CDC 串流。

```
aws keyspacestreams list-streams --keyspace-name ks_1 --region us-east-1
```

命令的輸出看起來與此類似。

```

{
 "Streams": [
 {
 "StreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
ks_1/table/t1/stream/2023-05-11T21:21:33.291",
 "StreamLabel": "2023-05-11T21:21:33.291",
 "KeyspaceName": "ks_1"
 "TableName": "t1",
 },
 {
 "StreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
ks_1/table/t2/stream/2023-05-11T21:21:33.291",

```

```
 "StreamLabel": "2023-05-11T21:21:33.291",
 "KeyspaceName": "ks_1"
 "TableName": "t2",
 }
]
}
```

4. 您也可以使用下列參數列出指定資料表的 CDC 串流。

```
aws keyspacesstreams list-streams --keyspace-name ks_1 --table-name t2 --
region us-east-1
```

命令的輸出看起來與此類似。

```
{
 "Streams": [
 {
 "StreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
ks_1/table/t2/stream/2023-05-11T21:21:33.291",
 "StreamLabel": "2023-05-11T21:21:33.291",
 "KeyspaceName": "ks_1"
 "TableName": "t2",
 }
]
}
```

## Console

在 Amazon Keyspaces 主控台中檢視 CDC 串流

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇資料表，然後從清單中選擇資料表。
3. 選擇串流索引標籤以檢閱串流詳細資訊。

## 在 Amazon Keyspaces 中存取 CDC 串流中的記錄

若要存取串流中的記錄，請使用 [Amazon Keyspaces Streams API](#)。下一節包含如何使用 存取記錄的範例 AWS CLI。

要了解必要的許可，請參閱 [the section called “設定 許可”](#)。

## 使用 存取串流中的記錄 AWS CLI

1. 您可以使用 Amazon Keyspaces Streams API 來存取串流的變更記錄。如需詳細資訊，請參閱 [Amazon Keyspaces Streams API 參考](#)。若要擷取串流中的碎片，您可以使用 `get-stream` API，如下列範例所示。

```
aws keyspacesstreams get-stream \
--stream-arn 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/
mytable/stream/STREAM_LABEL '
```

以下為輸出範例。

```
{
 "StreamArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/
table/mytable/stream/2023-05-11T21:21:33.291",
 "StreamStatus": "ENABLED",
 "StreamViewType": "NEW_AND_OLD_IMAGES",
 "CreationRequestDateTime": "<CREATION_TIME>",
 "KeyspaceName": "mykeyspace",
 "TableName": "mytable",
 "StreamLabel": "2023-05-11T21:21:33.291",
 "Shards": [
 {
 "SequenceNumberRange": {
 "EndingSequenceNumber": "<END_SEQUENCE_NUMBER>",
 "StartingSequenceNumber": "<START_SEQUENCE_NUMBER>"
 },
 "ShardId": "<SHARD_ID>"
 },
]
}
```

2. 若要從串流擷取記錄，請先取得迭代器，以提供您存取記錄的起點。若要這樣做，您可以使用上一個步驟中 API 傳回的 CDC 串流中的碎片。若要收集迭代器，您可以使用 `get-shard-iterator` API。在此範例中，您會使用從碎片的最後一個裁剪點或開頭 `TRIM_HORIZON` 擷取的類型反覆運算器 )。

```
aws keyspacesstreams get-shard-iterator \

```

```
--stream-arn 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/
mytable/stream/STREAM_LABEL' \
--shard-id 'SHARD_ID' \
--shard-iterator-type 'TRIM_HORIZON'
```

命令的輸出看起來與下列範例類似。

```
{
 "ShardIterator": "<SHARD_ITERATOR>"
}
```

- 若要使用 `get-records` API 擷取 CDC 記錄，您可以使用最後一個步驟中傳回的迭代器。下列命令是此範例。

```
aws keyspacesstreams get-records \
--shard-iterator 'SHARD_ITERATOR' \
--limit 100
```

## 使用 Kinesis Client Library (KCL) 來處理 Amazon Keyspaces 串流

本主題說明如何使用 Kinesis Client Library (KCL) 來取用和處理來自 Amazon Keyspaces 變更資料擷取 (CDC) 串流的資料。

使用 Kinesis Client Library (KCL) 提供許多好處，而不是直接使用 Amazon Keyspaces Streams API，例如：

- 內建碎片歷程追蹤和迭代器處理。
- 跨工作者的自動負載平衡。
- 工作者故障的容錯能力和復原能力。
- 檢查點以追蹤處理進度。
- 調整串流容量的變更。
- 簡化處理 CDC 記錄的分散式運算。

下一節概述為什麼和如何使用 Kinesis Client Library (KCL) 來處理串流，並提供使用 KCL 處理 Amazon Keyspaces CDC 串流的範例。

如需定價的相關資訊，請參閱 [Amazon Keyspaces \(適用於 Apache Cassandra\) 定價](#)。

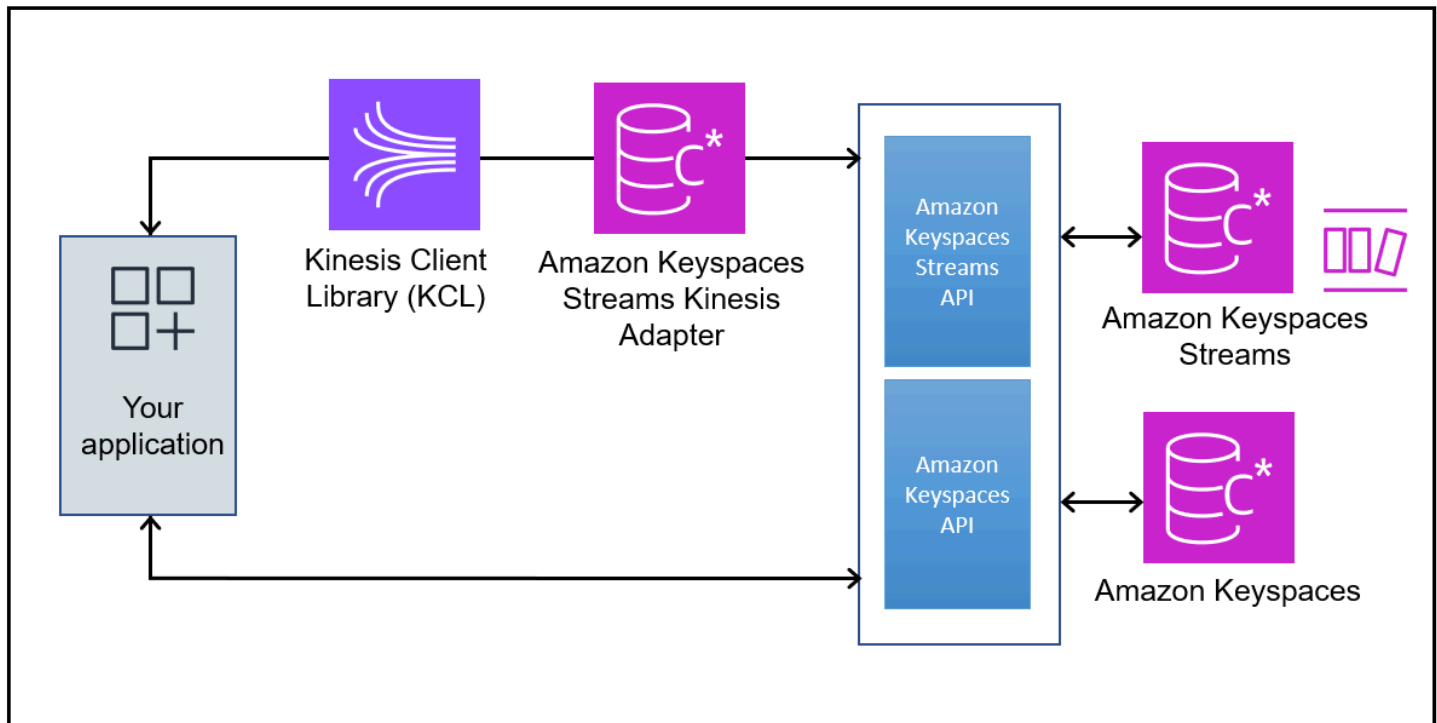
## 什麼是 Kinesis Client Library ?

Kinesis Client Library (KCL) 是獨立的 Java 軟體程式庫，旨在簡化從串流取用和處理資料的程式。KCL 會處理許多與分散式運算相關的複雜任務，讓您專注於在處理串流資料時實作商業邏輯。KCL 會管理各種活動，例如跨多個工作者的負載平衡、回應工作者失敗、檢查點處理過的記錄，以及回應串流中碎片數量的變更。

若要處理 Amazon Keyspaces CDC 串流，您可以使用 KCL 中找到的設計模式來處理串流碎片和串流記錄。KCL 會在低階 Kinesis Data Streams API 上提供有用的抽象，可以簡化程式碼。如需 KCL 的詳細資訊，請參閱《Amazon Kinesis Data Streams 開發人員指南》中的[使用 KCL 開發消費者](#)。

若要使用 KCL 撰寫應用程式，請使用 Amazon Keyspaces Streams Kinesis Adapter。Kinesis Adapter 實作 Kinesis Data Streams 介面，讓您可以使用 KCL 來取用和處理來自 Amazon Keyspaces 串流的記錄。如需如何設定和安裝 Amazon Keyspaces 串流 Kinesis 轉接器的指示，請造訪 [GitHub](#) 儲存庫。

下圖顯示這些程式庫如何彼此互動。



KCL 經常更新，以納入基礎程式庫的較新版本、安全性改善和錯誤修正。我們建議您使用最新版本的 KCL，以避免已知問題並從所有最新的改進中獲益。若要尋找最新的 KCL 版本，請參閱 [KCL GitHub 儲存庫](#)。

## KCL 概念

在使用 KCL 實作消費者應用程式之前，您應該了解下列概念：

## KCL 消費者應用程式

KCL 取用者應用程式是處理來自 Amazon Keyspaces CDC 串流資料的程式。KCL 可做為取用者應用程式程式碼與 Amazon Keyspaces CDC 串流之間的媒介。

### 工作者

工作者是 KCL 取用者應用程式的執行單位，可處理來自 Amazon Keyspaces CDC 串流的資料。您的應用程式可以執行分散在多個執行個體的多個工作者。

### 記錄處理器

記錄處理器是應用程式中的邏輯，可處理 Amazon Keyspaces CDC 串流中碎片的資料。記錄處理器會由工作者針對其管理的每個碎片進行執行個體化。

### 租用

租用代表碎片的處理責任。工作者使用租用來協調哪個工作者正在處理哪個碎片。KCL 會將租用資料存放在 Amazon DynamoDB 的資料表中。

### 檢查點

檢查點是碎片中位置的記錄，直到記錄處理器成功處理記錄為止。檢查點可讓您的應用程式在工作者失敗時，從停止的地方繼續處理。

使用 Amazon Keyspaces Kinesis 轉接器，您可以開始針對 KCL 介面進行開發，並將 API 呼叫無縫導向 Amazon Keyspaces 串流端點。如需可用端點的清單，請參閱 [the section called “存取 CDC 串流端點”](#)。

當應用程式啟動後，其會呼叫 KCL 以將工作者執行個體化。您必須向工作者提供應用程式的組態資訊，例如串流描述項和 AWS 登入資料，以及您提供的記錄處理器類別名稱。當其在記錄處理器中執行程式碼時，工作者會執行下列任務：

- 連線到串流
- 列舉串流內的碎片
- 與其他工作者 (若有) 協調碎片關聯性
- 為其所管理的每個碎片執行個體化記錄處理器
- 從串流提取紀錄
- 將記錄推送至對應的記錄處理器
- 對已處理的記錄執行檢查點作業

- 當工作者執行個體數目變更時，平衡碎片與工作者的關聯
- 當碎片進行分割時，平衡碎片與工作者的關聯

## 實作 Amazon Keyspaces CDC 串流的 KCL 取用者應用程式

本主題提供step-by-step指南，以處理 Amazon Keyspaces CDC 串流。

### 1. 先決條件：開始之前，請確定您已：

- 具有 CDC 串流的 Amazon Keyspaces 資料表
- IAM 主體需要 IAM 許可才能存取 Amazon Keyspaces CDC 串流、建立和存取 DynamoDB 資料表以進行 KCL 串流處理，以及將指標發佈至 CloudWatch 的許可。如需詳細資訊和政策範例，請參閱 [the section called “KCL 的許可”](#)。
- 請確定已在本機組態中設定有效的AWS登入資料。如需詳細資訊，請參閱[the section called “管理存取金鑰”](#)。
- Java 開發套件 (JDK) 8 或更新版本
- Github 上的 [Readme](#) 中列出的需求。

### 2. 在此步驟中，您將 KCL 相依性新增至您的專案。對於 Maven，請將下列項目新增至您的 pom.xml：

```
<dependencies>
 <dependency>
 <groupId>software.amazon.kinesis</groupId>
 <artifactId>amazon-kinesis-client</artifactId>
 <version>3.1.0</version>
 </dependency>
 <dependency>
 <groupId>software.amazon.keyspaces</groupId>
 <artifactId>keyspaces-streams-kinesis-adapter</artifactId>
 <version>1.0.0</version>
 </dependency>
</dependencies>
```

#### Note

一律在 KCL [GitHub 儲存庫檢查 KCL](#) 的最新版本。

### 3. 建立產生記錄處理器執行個體的工廠類別：



```

import software.amazon.awssdk.services.keyspacesstreams.model.Record;
import
 software.amazon.keyspaces.streamsadapter.adapter.KeyspacesStreamsClientRecord;
import
 software.amazon.keyspaces.streamsadapter.model.KeyspacesStreamsProcessRecordsInput;
import
 software.amazon.keyspaces.streamsadapter.processor.KeyspacesStreamsShardRecordProcessor;
import software.amazon.kinesis.lifecycle.events.InitializationInput;
import software.amazon.kinesis.lifecycle.events.LeaseLostInput;
import software.amazon.kinesis.lifecycle.events.ShardEndedInput;
import software.amazon.kinesis.lifecycle.events.ShutdownRequestedInput;
import software.amazon.kinesis.processor.RecordProcessorCheckpoint;

public class RecordProcessor implements KeyspacesStreamsShardRecordProcessor {
 private String shardId;

 @Override
 public void initialize(InitializationInput initializationInput) {
 this.shardId = initializationInput.shardId();
 System.out.println("Initializing record processor for shard: " + shardId);
 }

 @Override
 public void processRecords(KeyspacesStreamsProcessRecordsInput
processRecordsInput) {
 try {
 for (KeyspacesStreamsClientRecord record :
processRecordsInput.records()) {
 Record keyspacesRecord = record.getRecord();
 System.out.println("Received record: " + keyspacesRecord);
 }

 if (!processRecordsInput.records().isEmpty()) {
 RecordProcessorCheckpoint checkpoint =
processRecordsInput.checkpointer();
 try {
 checkpoint.checkpoint();
 System.out.println("Checkpoint successful for shard: " +
shardId);
 } catch (Exception e) {
 System.out.println("Error while checkpointing for shard: " +
shardId + " " + e);
 }
 }
 }
 }
}

```

```

 }
 } catch (Exception e) {
 System.out.println("Error processing records for shard: " + shardId + "
" + e);
 }
}

@Override
public void leaseLost(LeaseLostInput leaseLostInput) {
 System.out.println("Lease lost for shard: " + shardId);
}

@Override
public void shardEnded(ShardEndedInput shardEndedInput) {
 System.out.println("Shard ended: " + shardId);
 try {
 // This is required. Checkpoint at the end of the shard
 shardEndedInput.checkpointer().checkpoint();
 System.out.println("Final checkpoint successful for shard: " +
shardId);
 } catch (Exception e) {
 System.out.println("Error while final checkpointing for shard: " +
shardId + " " + e);
 throw new RuntimeException("Error while final checkpointing", e);
 }
}

@Override
public void shutdownRequested(ShutdownRequestedInput shutdownRequestedInput) {
 System.out.println("Shutdown requested for shard " + shardId);
 try {
 shutdownRequestedInput.checkpointer().checkpoint();
 } catch (Exception e) {
 System.out.println("Error while checkpointing on shutdown for shard: "
+ shardId + " " + e);
 }
}
}

```

#### 4. 建立記錄工廠，如下列範例所示。

```

import software.amazon.kinesis.processor.ShardRecordProcessor;
import software.amazon.kinesis.processor.ShardRecordProcessorFactory;

```

```
import java.util.Queue;
import java.util.concurrent.ConcurrentLinkedQueue;

public class RecordProcessorFactory implements ShardRecordProcessorFactory {
 private final Queue<RecordProcessor> processors = new
 ConcurrentLinkedQueue<>();

 @Override
 public ShardRecordProcessor shardRecordProcessor() {
 System.out.println("Creating new RecordProcessor");
 RecordProcessor processor = new RecordProcessor();
 processors.add(processor);
 return processor;
 }
}
```

5. 在此步驟中，您會建立基本類別來設定 KCLv3 和 Amazon Keyspaces 轉接器。

```
import com.example.KCLExample.utils.RecordProcessorFactory;
import
 software.amazon.keyspaces.streamsadapter.AmazonKeyspacesStreamsAdapterClient;
import software.amazon.keyspaces.streamsadapter.StreamsSchedulerFactory;
import java.util.Arrays;
import java.util.List;
import java.util.concurrent.ExecutionException;

import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cloudwatch.CloudWatchAsyncClient;
import software.amazon.awssdk.services.dynamodb.DynamoDbAsyncClient;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableRequest;
import software.amazon.awssdk.services.dynamodb.model.DeleteTableResponse;
import software.amazon.awssdk.services.keyspacesstreams.KeyspacesStreamsClient;
import software.amazon.awssdk.services.kinesis.KinesisAsyncClient;
import software.amazon.kinesis.common.ConfigsBuilder;
import software.amazon.kinesis.common.InitialPositionInStream;
import software.amazon.kinesis.common.InitialPositionInStreamExtended;
import software.amazon.kinesis.coordinator.CoordinatorConfig;
import software.amazon.kinesis.coordinator.Scheduler;
import software.amazon.kinesis.leases.LeaseManagementConfig;
import software.amazon.kinesis.processor.ProcessorConfig;
import software.amazon.kinesis.processor.StreamTracker;
import software.amazon.kinesis.retrieval.polling.PollingConfig;

public class KCLTestBase {
```

```
protected KeyspacesStreamsClient streamsClient;
protected KinesisAsyncClient adapterClient;
protected DynamoDbAsyncClient dynamoDbAsyncClient;
protected CloudWatchAsyncClient cloudWatchClient;
protected Region region;
protected RecordProcessorFactory recordProcessorFactory;
protected Scheduler scheduler;
protected Thread schedulerThread;

public void baseSetUp() {
 recordProcessorFactory = new RecordProcessorFactory();
 setupKCLBase();
}

protected void setupKCLBase() {
 region = Region.US_EAST_1;

 streamsClient = KeyspacesStreamsClient.builder()
 .region(region)
 .build();
 adapterClient = new AmazonKeyspacesStreamsAdapterClient(
 streamsClient,
 region);
 dynamoDbAsyncClient = DynamoDbAsyncClient.builder()
 .region(region)
 .build();
 cloudWatchClient = CloudWatchAsyncClient.builder()
 .region(region)
 .build();
}

protected void startScheduler(Scheduler scheduler) {
 this.scheduler = scheduler;
 schedulerThread = new Thread(() -> scheduler.run());
 schedulerThread.start();
}

protected void shutdownScheduler() {
 if (scheduler != null) {
 scheduler.shutdown();
 try {
 schedulerThread.join(30000);
 } catch (InterruptedException e) {
```

```

 System.out.println("Error while shutting down scheduler " + e);
 }
}

protected Scheduler createScheduler(String streamArn, String leaseTableName) {
 String workerId = "worker-" + System.currentTimeMillis();

 // Create ConfigsBuilder
 ConfigsBuilder configsBuilder = createConfigsBuilder(streamArn, workerId,
leaseTableName);

 // Configure retrieval config for polling
 PollingConfig pollingConfig = new PollingConfig(streamArn, adapterClient);

 // Create the Scheduler
 return StreamsSchedulerFactory.createScheduler(
 configsBuilder.checkpointConfig(),
 configsBuilder.coordinatorConfig(),
 configsBuilder.leaseManagementConfig(),
 configsBuilder.lifecycleConfig(),
 configsBuilder.metricsConfig(),
 configsBuilder.processorConfig(),

configsBuilder.retrievalConfig().retrievalSpecificConfig(pollingConfig),
 streamsClient,
 region
);
}

private ConfigsBuilder createConfigsBuilder(String streamArn, String workerId,
String leaseTableName) {
 ConfigsBuilder configsBuilder = new ConfigsBuilder(
 streamArn,
 leaseTableName,
 adapterClient,
 dynamoDbAsyncClient,
 cloudWatchClient,
 workerId,
 recordProcessorFactory);

 configureCoordinator(configsBuilder.coordinatorConfig());
 configureLeaseManagement(configsBuilder.leaseManagementConfig());
 configureProcessor(configsBuilder.processorConfig());
}

```

```

 configureStreamTracker(configsBuilder, streamArn);

 return configsBuilder;
 }

 private void configureCoordinator(CoordinatorConfig config) {
 config.skipShardSyncAtWorkerInitializationIfLeasesExist(true)
 .parentShardPollIntervalMillis(1000)
 .shardConsumerDispatchPollIntervalMillis(500);
 }

 private void configureLeaseManagement(LeaseManagementConfig config) {
 config.shardSyncIntervalMillis(0)
 .leasesRecoveryAuditorInconsistencyConfidenceThreshold(0)
 .leasesRecoveryAuditorExecutionFrequencyMillis(5000)
 .leaseAssignmentIntervalMillis(1000L);
 }

 private void configureProcessor(ProcessorConfig config) {
 config.callProcessRecordsEvenForEmptyRecordList(true);
 }

 private void configureStreamTracker(ConfigsBuilder configsBuilder, String
streamArn) {
 StreamTracker streamTracker =
StreamsSchedulerFactory.createSingleStreamTracker(
 streamArn,

InitialPositionInStreamExtended.newInitialPosition(InitialPositionInStream.TRIM_HORIZON)
);
 configsBuilder.streamTracker(streamTracker);
 }

 public void deleteAllDdbTables(String baseTableName) {
 List<String> tablesToDelete = Arrays.asList(
 baseTableName,
 baseTableName + "-CoordinatorState",
 baseTableName + "-WorkerMetricStats"
);

 for (String tableName : tablesToDelete) {
 deleteTable(tableName);
 }
 }
}

```

```

private void deleteTable(String tableName) {
 DeleteTableRequest deleteTableRequest = DeleteTableRequest.builder()
 .tableName(tableName)
 .build();

 try {
 DeleteTableResponse response =
dynamoDbAsyncClient.deleteTable(deleteTableRequest).get();
 System.out.println("Table deletion response " + response);
 } catch (InterruptedException | ExecutionException e) {
 System.out.println("Error deleting table: " + tableName + " " + e);
 }
}
}

```

6. 在此步驟中，您會為應用程式實作記錄處理器類別，以開始處理變更事件。

```

import software.amazon.kinesis.coordinator.Scheduler;

public class KCLTest {

 private static final int APP_RUNTIME_SECONDS = 1800;
 private static final int SLEEP_INTERNAL_MS = 60*1000;

 public static void main(String[] args) {
 KCLTestBase kclTestBase;

 kclTestBase = new KCLTestBase();
 kclTestBase.baseSetUp();

 // Create and start scheduler
 String leaseTableName = generateUniqueApplicationName();

 // Update below to your Stream ARN
 String streamArn = "arn:aws:cassandra:us-east-1:759151643516:/keyspace/
cdc_sample_test/table/test_kcl_bool/stream/2025-07-01T15:52:57.529";
 Scheduler scheduler = kclTestBase.createScheduler(streamArn,
leaseTableName);
 kclTestBase.startScheduler(scheduler);

 // Wait for specified time before shutting down - KCL applications are
designed to run forever, however in this
 // example we will shut it down after APP_RUNTIME_SECONDS
 }
}

```

```

 long startTime = System.currentTimeMillis();
 long endTime = startTime + (APP_RUNTIME_SECONDS * 1000);
 while (System.currentTimeMillis() < endTime) {
 try {
 // Print and sleep every minute
 Thread.sleep(SLEEP_INTERNAL_MS);
 System.out.println("Application is running");
 } catch (InterruptedException e) {
 System.out.println("Interrupted while waiting for records");
 Thread.currentThread().interrupt();
 break;
 }
 }

 // Stop the scheduler
 kclTestBase.shutdownScheduler();
 kclTestBase.deleteAllDdbTables(leaseTableName);
 }

 public static String generateUniqueApplicationName() {
 String timestamp = String.valueOf(System.currentTimeMillis());
 String randomString = java.util.UUID.randomUUID().toString().substring(0,
8);
 return String.format("KCL-App-%s-%s", timestamp, randomString);
 }
}

```

## 最佳實務

搭配 Amazon Keyspaces CDC 串流使用 KCL 時，請遵循下列最佳實務：

### 錯誤處理

在記錄處理器中實作強大的錯誤處理，以正常處理例外狀況。考慮實作暫時性故障的重試邏輯。

### 檢查點頻率

平衡檢查點頻率，將重複處理降至最低，同時確保合理的進度追蹤。過於頻繁的檢查點可能會影響效能，而過於頻繁的檢查點可能會在工作者失敗時導致更多的重新處理。

### 工作者擴展

根據 CDC 串流中的碎片數量來擴展工作者數量。一個很好的起點是每個碎片有一個工作者，但您可能需要根據您的處理需求進行調整。



## 監控

使用 KCL 提供的 CloudWatch 指標來監控消費者應用程式的運作狀態和效能。關鍵指標包括處理延遲、檢查點存留期和租用計數。

## 測試

徹底測試您的消費者應用程式，包括工作者失敗、串流重新分片和各種載入條件等案例。

## 使用 KCL 搭配非 Java 語言

雖然 KCL 主要是 Java 程式庫，但您可以透過 MultiLangDaemon 將其與其他程式設計語言搭配使用。MultiLangDaemon 是以 Java 為基礎的協助程式，可管理非 Java 記錄處理器與 KCL 之間的互動。

KCL 支援下列語言：

- Python
- Ruby
- Node.js
- .NET

如需使用 KCL 搭配非 Java 語言的詳細資訊，請參閱 [KCL MultiLangDaemon 文件](#)。

## 疑難排解

本節提供將 KCL 與 Amazon Keyspaces CDC 串流搭配使用時可能遇到的常見問題的解決方案。

### 處理緩慢

如果您的消費者應用程式處理記錄的速度緩慢，請考慮：

- 增加工作者執行個體的數量
- 最佳化您的記錄處理邏輯
- 檢查下游系統中是否有瓶頸

### 重複處理

如果您看到重複處理記錄，請檢查您的檢查點邏輯。確定您在成功處理記錄後進行檢查點。

## 工作者故障

如果工作者經常失敗，請檢查：

- 資源限制 (CPU、記憶體 )
- 網路連線問題
- 許可問題

## 租用資料表問題

如果您遇到 KCL 租用資料表的問題：

- 檢查您的應用程式是否具有存取 Amazon Keyspaces 資料表的適當許可
- 確認資料表具有足夠的佈建輸送量

# 在 Amazon Keyspaces 中使用分割區

在 Apache Cassandra 中，分割區控制哪些節點資料存放在叢集中。分割區會使用分割區索引鍵的雜湊值來建立數值字符。Cassandra 使用此權杖來跨節點分配資料。用戶端也可以在 SELECT 操作和 WHERE 子句中使用這些字符來最佳化讀取和寫入操作。例如，用戶端可以透過在每個平行任務中指定要查詢的不同字符範圍，有效率地對大型資料表執行平行查詢。

Amazon Keyspaces 提供三種不同的分割區。

### Murmur3Partitioner (預設)

Apache Cassandra 相容 Murmur3Partitioner。Murmur3Partitioner 是 Amazon Keyspaces 和 Cassandra 1.2 及更新版本中的預設 Cassandra 分割區。

### RandomPartitioner

Apache Cassandra 相容 RandomPartitioner。RandomPartitioner 是 Cassandra 1.2 之前版本的預設 Cassandra 分割區。

### Keyspaces 預設分割區

DefaultPartitioner 會傳回與相同的 token 函數結果 RandomPartitioner。

分割器設定會在帳戶層級的每個區域套用。例如，如果您在美國東部（維吉尼亞北部）變更分割區，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分割器。請注意，組態變更

大約需要 10 分鐘才能完成。當您變更分割區設定時，不需要重新載入 Amazon Keyspaces 資料。用戶端下次連線時，會自動使用新的分割區設定。

## 如何在 Amazon Keyspaces 中變更分割區

您可以使用 AWS 管理主控台 或 Cassandra 查詢語言 (CQL) 來變更分割區。

### AWS 管理主控台

使用 Amazon Keyspaces 主控台變更分割區

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇組態。
3. 在組態頁面上，移至編輯分割區。
4. 選取與您的 Cassandra 版本相容的分割區器。分割區變更大約需要 10 分鐘才能套用。

#### Note

組態變更完成後，您必須中斷連線並重新連線至 Amazon Keyspaces，請求才能使用新的分割區。

### Cassandra Query Language (CQL)

1. 若要查看已為帳戶設定哪些分割區，您可以使用下列查詢。

```
SELECT partitioner from system.local;
```

如果分割區器尚未變更，查詢會有下列輸出。

```
partitioner

com.amazonaws.cassandra.DefaultPartitioner
```

2. 若要將分割區器更新為Murmur3分割區器，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。若要確認分割區已設定，您可以再次執行 SELECT 查詢。請注意，由於最終讀取一致性，回應可能無法反映最近完成的分割區變更的結果。如果您在短時間內再次重複 SELECT 操作，回應應該會傳回最新的資料。

```
SELECT partitioner from system.local;
```

 Note

您必須中斷連線並重新連線至 Amazon Keyspaces，讓請求使用新的分割區。

## Amazon Keyspaces 中的用戶端時間戳記

在 Amazon Keyspaces 中，用戶端時間戳記是 Cassandra 相容的時間戳記，會針對資料表中的每個儲存格保留這些時間戳記。您可以讓用戶端應用程式決定寫入順序，藉此使用用戶端時間戳記來解決衝突。例如，當全域分佈應用程式的用戶端更新相同的資料時，用戶端時間戳記會保留在用戶端上進行更新的順序。Amazon Keyspaces 使用這些時間戳記來處理寫入。

Amazon Keyspaces 用戶端時間戳記受到完整管理。您不需要管理低階系統設定，例如清除和壓縮策略。

當您刪除資料時，資料列會標示為使用 tombstone 刪除。Amazon Keyspaces 會自動（通常在 10 天內）移除斷層資料，而不會影響您的應用程式效能或可用性。Tombstoned 資料不適用於資料處理語言 (DML) 陳述式。當您繼續對包含斷層資料的列執行讀取和寫入時，斷層資料會繼續計入儲存體、讀取容量單位 (RCUs) 和寫入容量單位 WCUs)，直到從儲存體中刪除為止。

在資料表的用戶端時間戳記開啟後，您可以在 Data Manipulation Language (DML) CQL 查詢中指定具有 USING TIMESTAMP 子句的時間戳記。如需詳細資訊，請參閱 [the section called “在查詢中使用用戶端時間戳記”](#)。如果您未在 CQL 查詢中指定時間戳記，Amazon Keyspaces 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不提供時間戳記，Amazon Keyspaces 會自動指派儲存格層級時間戳記，因為時間戳記不能是 NULL。若要查詢時間戳記，您可以在 DML 陳述式中使用 WRITETIME 函數。

Amazon Keyspaces 不會額外收費以開啟用戶端時間戳記。不過，使用用戶端時間戳記，您可以為資料列中的每個值存放和寫入其他資料。這可能會導致額外的儲存體用量，在某些情況下可能會導致額外的輸送量用量。如需 Amazon Keyspaces 定價的詳細資訊，請參閱 [Amazon Keyspaces \(適用於 Apache Cassandra\) 定價](#)。

在 Amazon Keyspaces 中開啟用戶端時間戳記時，每一列的每一欄都會儲存時間戳記。這些時間戳記大約需要 20-40 個位元組（取決於您的資料），並對資料列的儲存和輸送量成本做出貢獻。這些中繼資料位元組也會計入 1-MB 的資料列大小配額。若要判斷儲存空間的整體增加（以確保資料列大小保持在 1 MB 以下），請考慮資料表中的資料欄數目和每一列中的集合元素數目。例如，如果資料表有 20 個資料欄，每欄存放 40 個位元組的資料，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 “估算資料列大小”](#)。除了額外 400 個位元組的儲存，在此範例中，每次寫入消耗的寫入容量單位 (WCUs) 數量從 1 個 WCU 增加到 2 WCUs。如需如何計算讀取和寫入容量的詳細資訊，請參閱 [the section called “設定讀取/寫入容量模式”](#)。

資料表的用戶端時間戳記開啟後，您就無法將其關閉。

若要進一步了解如何在查詢中使用用戶端時間戳記，請參閱 [the section called “在查詢中使用用戶端時間戳記”](#)。

## 主題

- [Amazon Keyspaces 用戶端時間戳記如何與 AWS 服務整合](#)
- [在 Amazon Keyspaces 中建立具有用戶端時間戳記的新資料表](#)
- [在 Amazon Keyspaces 中設定資料表的用戶端時間戳記](#)
- [在 Amazon Keyspaces 中的查詢中使用用戶端時間戳記](#)

## Amazon Keyspaces 用戶端時間戳記如何與 AWS 服務整合

下列用戶端時間戳記指標可在 Amazon CloudWatch 中使用，以啟用持續監控。

- SystemReconciliationDeletes – 移除斷層資料所需的刪除操作數目。

如需如何監控 CloudWatch 指標的詳細資訊，請參閱 [the section called “使用 CloudWatch 進行監控”](#)。

使用時 CloudFormation，您可以在建立 Amazon Keyspaces 資料表時啟用用戶端時間戳記。如需詳細資訊，請參閱 [「AWS CloudFormation 使用者指南」](#)。

## 在 Amazon Keyspaces 中建立具有用戶端時間戳記的新資料表

請依照這些範例，使用 Amazon Keyspaces AWS 管理主控台、Cassandra 查詢語言 (CQL) 或 建立啟用用戶端時間戳記的新 Amazon Keyspaces 資料表 AWS Command Line Interface

## Console

使用用戶端時間戳記建立新的資料表（主控台）

1. 登入 AWS 管理主控台，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. 在資料表詳細資訊區段的建立資料表頁面上，選取金鑰空間，並提供新資料表的名稱。
4. 在結構描述區段中，建立資料表的結構描述。
5. 在資料表設定區段中，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，以開啟資料表的用戶端時間戳記。

7. 選擇 Create Table (建立資料表)。您的資料表會在用戶端時間戳記開啟的情況下建立。

## Cassandra Query Language (CQL)

使用 CQL 建立新資料表

1. 若要使用 CQL 建立啟用用戶端時間戳記的新資料表，您可以使用下列範例。

```
CREATE TABLE my_keyspace.my_table (
 userid uuid,
 time timeuuid,
 subject text,
 body text,
 user inet,
 PRIMARY KEY (userid, time)
) WITH CUSTOM_PROPERTIES = {'client_side_timestamps': {'status': 'enabled'}};
```

2. 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 custom\_properties，如下列範例所示。

```
SELECT custom_properties from system_schema_mcs.tables where keyspace_name =
 'my_keyspace' and table_name = 'my_table';
```

此陳述式的輸出會顯示用戶端時間戳記的狀態。

```
'client_side_timestamps': {'status': 'enabled'}
```

## AWS CLI

### 使用 建立新的資料表 AWS CLI

1. 若要在啟用用戶端時間戳記的情況下建立新的資料表，您可以使用下列範例。

```
./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}]'
```

2. 若要確認新資料表的用戶端時間戳記已開啟，請執行下列程式碼。

```
./aws keyspaces get-table \
--keyspace-name my_keyspace \
--table-name my_table
```

輸出看起來應該與此範例類似。

```
{
 "keyspaceName": "my_keyspace",
 "tableName": "my_table",
 "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/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 Amazon Keyspaces 中設定資料表的用戶端時間戳記

請依照這些範例，使用 Amazon Keyspaces AWS 管理主控台、Cassandra Query Language (CQL) 或開啟現有資料表的用戶端時間戳記 AWS Command Line Interface。



## Console

開啟現有資料表的用戶端時間戳記（主控台）

1. 登入 AWS 管理主控台，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，然後選擇其他設定索引標籤。
3. 在其他設定索引標籤上，前往修改用戶端時間戳記，然後選取開啟用戶端時間戳記
4. 選擇儲存變更以變更資料表的設定。

## Cassandra Query Language (CQL)

使用 CQL 陳述式

1. 使用 ALTER TABLE CQL 陳述式開啟現有資料表的用戶端時間戳記。

```
ALTER TABLE my_table WITH custom_properties = {'client_side_timestamps':
{'status': 'enabled'}};;
```

2. 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 custom\_properties，如下列範例所示。

```
SELECT custom_properties from system_schema.tables where keyspace_name =
'my_keyspace' and table_name = 'my_table';
```

此陳述式的輸出會顯示用戶端時間戳記的狀態。

```
'client_side_timestamps': {'status': 'enabled'}
```

## AWS CLI

使用 AWS CLI

1. 您可以使用 AWS CLI 下列範例，使用 開啟現有資料表的用戶端時間戳記。

```
./aws keyspaces update-table \
--keyspace-name my_keyspace \
--table-name my_table \
--enable-client-side-timestamps
```

```
--client-side-timestamps 'status=ENABLED'
```

- 若要確認資料表的用戶端時間戳記已開啟，請執行下列程式碼。

```
./aws keyspace get-table \
--keyspace-name my_keyspace \
--table-name my_table
```

輸出看起來應該類似於此範例，並將用戶端時間戳記的狀態顯示為 ENABLED。

```
{
 "keyspaceName": "my_keyspace",
 "tableName": "my_table",
 "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/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 Amazon Keyspaces 中的查詢中使用用戶端時間戳記

開啟用戶端時間戳記之後，您可以使用 `USING TIMESTAMP` 子句在 `INSERT`、`UPDATE` 和 `DELETE` 陳述式中傳遞時間戳記。

時間戳記值 `bigint` 表示自稱為 的標準基準時間以來的微秒數 `epoch`：1970 年 1 月 1 日，格林威治標準時間 00:00:00。由用戶端提供的時間戳記必須介於過去 2 天到未來 5 分鐘之間，從目前的牆壁時鐘時間起算。

Amazon Keyspaces 會在資料生命週期內保留時間戳記中繼資料。您可以使用 `WRITETIME` 函數來查詢過去幾年發生的時間戳記。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 的範例 `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 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 會為您的寫入操作指派伺服器端時間戳記。

若要查看針對特定資料欄存放的時間戳記值，您可以在 SELECT 陳述式中使用 WRITETIME 函數，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;
```

## Amazon Keyspaces 的多區域複寫（適用於 Apache Cassandra）

您可以使用 Amazon Keyspaces 多區域複寫，透過 AWS 區域 您選擇的自動化、全受管、主動-主動複寫來複寫資料。透過主動-主動複寫，每個區域都可以獨立執行讀取和寫入。您可以改善區域降級的可用性和彈性，同時受益於全域應用程式的低延遲本機讀取和寫入。

使用多區域複寫時，Amazon Keyspaces 會在區域之間非同步複寫資料，資料通常會在一秒內跨區域傳播。此外，透過多區域複寫，您不再難以解決衝突和修正資料差異問題，因此您可以專注於應用程式。

根據預設，Amazon Keyspaces 會在相同 內的三個[可用區域中](#)複寫資料，AWS 區域 以提供耐用性和高可用性。透過多區域複寫，您可以建立多區域金鑰空間，以 AWS 區域 您選擇的不同地理位置複寫資料表。

### 主題

- [使用多區域複寫的優點](#)
- [容量模式和定價](#)
- [多區域複寫如何在 Amazon Keyspaces 中運作](#)
- [Amazon Keyspaces 多區域複寫用量備註](#)
- [設定 Amazon Keyspaces 的多區域複寫（適用於 Apache Cassandra）](#)

## 使用多區域複寫的優點

多區域複寫提供下列優點。

- 具有單一位數毫秒延遲的全域讀取和寫入 – 在 Amazon Keyspaces 中，複寫是主動的。您可以從最接近您客戶的區域，以任何規模的單一位數毫秒延遲提供本機讀取和寫入。您可以將 Amazon Keyspaces 多區域資料表用於需要快速回應時間的全域應用程式。
- 改善單一區域降級的業務持續性和保護 – 透過多區域複寫，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 AWS 區域，以從單一區域降級中復原。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 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。區域恢復上線後，Amazon Keyspaces 會自動同步任何遺失的變更，讓您可以復原而不會有任何應用程式影響。

- 跨區域高速複寫 – 多區域複寫使用跨區域的快速儲存型資料實體複寫，複寫延遲通常不到 1 秒。

Amazon Keyspaces 中的複寫對您的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。這表示您可以解決高寫入輸送量使用案例，或輸送量突然遽增或爆增的使用案例，而不會影響應用程式。

- 一致性和衝突解決 – 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，則可能會發生衝突。

為了協助提供最終一致性，Amazon Keyspaces 使用儲存格層級時間戳記，而最後一個寫入器會在並行更新之間取得對帳。衝突解決是完全受管的，並在背景發生，不會影響任何應用程式。

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”](#)。

## 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容量模式。如需詳細資訊，請參閱 [the section called “設定讀取/寫入容量模式”](#)。

對於隨需模式，您需要支付 1 個寫入請求單位 (WRU) 的費用，以寫入每列最多 1 KB 的資料，方式與單一區域資料表相同。但是，您需要為多區域金鑰空間的每個區域中的寫入付費。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 6 WRUs： $3 * 2 = 6$  WRUs。此外，包含靜態和非靜態資料的寫入需要額外的寫入操作。

對於佈建模式，您需支付 1 個寫入容量單位 (WCU) 的費用，每一列最多寫入 1 KB 的資料，與單一區域資料表相同。但是，您需要為多區域金鑰空間的每個區域中的寫入付費。例如，在具有兩個區域的多區域金鑰空間中每秒寫入 3 KB 的資料列需要 6 WCUs： $3 * 2 = 6$  WCUs。此外，包含靜態和非靜態資料的寫入需要額外的寫入操作。

如需定價的詳細資訊，請參閱 [Amazon Keyspaces \(適用於 Apache Cassandra\) 定價](#)。

## 多區域複寫如何在 Amazon Keyspaces 中運作

本節概述 Amazon Keyspaces 多區域複寫的運作方式。如需定價的詳細資訊，請參閱 [Amazon Keyspaces \(適用於 Apache Cassandra\) 定價](#)。

### 主題

- [多區域複寫如何在 Amazon Keyspaces 中運作](#)
- [多區域複寫衝突解決](#)
- [多區域複寫災難復原](#)
- [預設 AWS 區域 停用 中的多區域複寫](#)
- [多區域複寫和與point-in-time的整合](#)
- [多區域複寫和與 AWS 服務的整合](#)

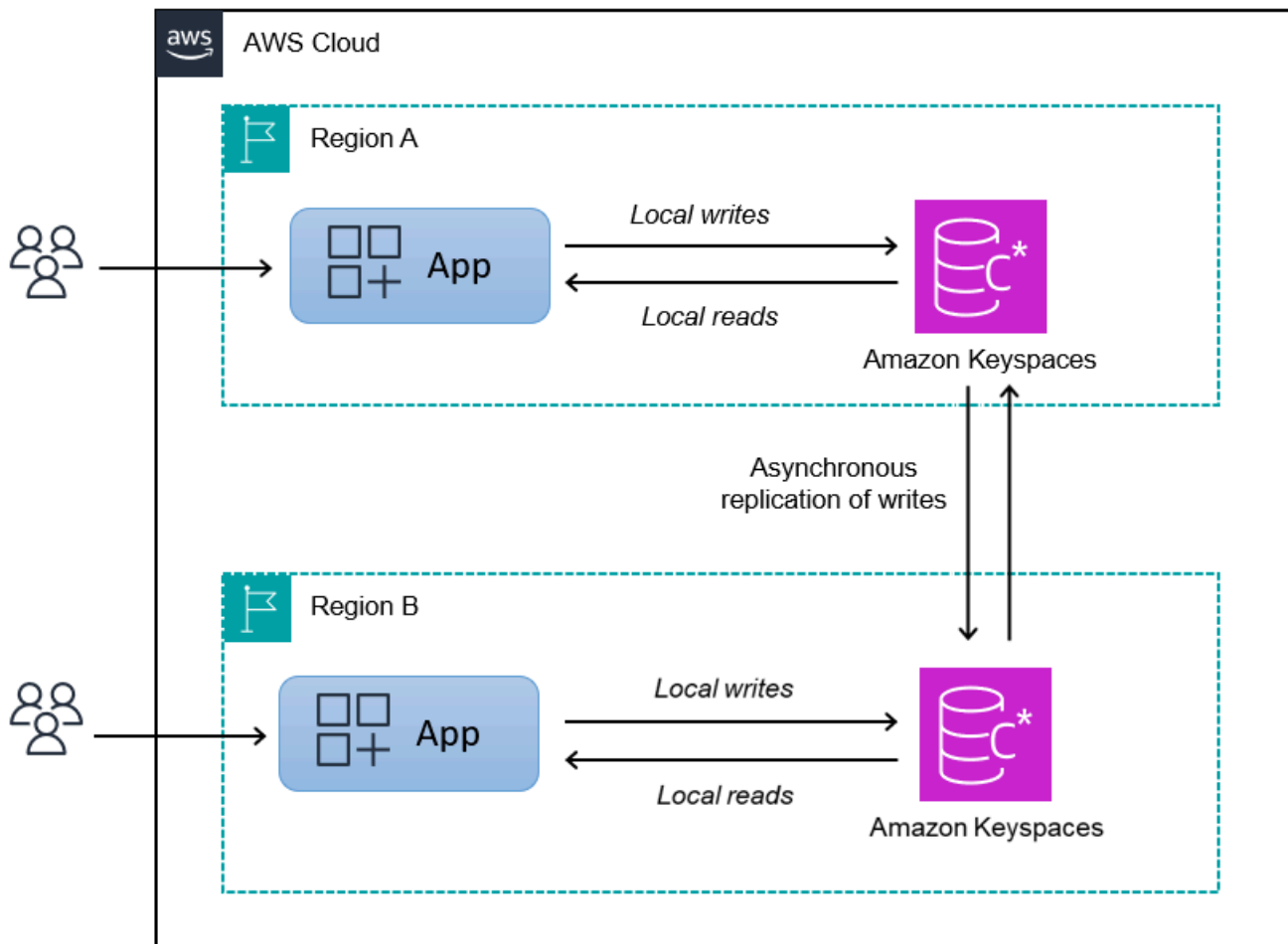
## 多區域複寫如何在 Amazon Keyspaces 中運作

Amazon Keyspaces 多區域複寫實作資料彈性架構，將您的資料分散到獨立和地理分佈 AWS 區域。它使用主動-主動複寫，可提供本機低延遲，讓每個區域能夠獨立執行讀取和寫入。

當您建立 Amazon Keyspaces 多區域金鑰空間時，您可以選擇要複寫資料的其他區域。您在多區域金鑰空間中建立的每個資料表都包含 Amazon Keyspaces 視為單一單位的多個複本資料表（每個區域一個）。

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，會使用LOCAL\_QUORUM一致性層級長期寫入資料。Amazon Keyspaces 會自動將資料非同步複寫到其他複寫區域。跨區域的複寫延遲通常不到一秒，而且不會影響您應用程式的效能或輸送量。

寫入資料後，您可以從具有LOCAL\_ONE/LOCAL\_QUORUM一致性層級的另一個複寫區域中的多區域資料表讀取資料。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”](#)。



## 多區域複寫衝突解決

Amazon Keyspaces 多區域複寫是完全受管的，這表示您不需要執行複寫任務，例如定期執行修復操作以清除資料同步問題。Amazon Keyspaces 透過偵測和修復衝突 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會贏得資料對帳方法。透過此衝突解決機制，多區域金鑰空間中的所有區域都會同意最新的更新，並收斂至具有相同資料的狀態。對帳程序不會影響應用程式效能。為了支援解決衝突，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[the section called “用戶端時間戳記”](#)。

## 多區域複寫災難復原

使用 Amazon Keyspaces 多區域複寫時，寫入會跨每個區域非同步複寫。在極少數的單一區域降級或故障情況下，多區域複寫可協助您從災難中復原，而對您的應用程式幾乎沒有影響。從災難復原通常使用復原時間目標 (RTO) 和復原點目標 (RPO) 的值進行測量。



復原時間目標 – 系統在災難後返回工作狀態所需的時間。RTO 會測量工作負載可容忍的停機時間量，並按時間測量。對於使用多區域複寫容錯移轉到不受影響區域的災難復原計劃，RTO 可以接近零。RTO 受限於您的應用程式偵測故障條件並將流量重新導向至另一個區域的速度。

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到不受影響區域的災難復原計劃，RPO 通常為單位數秒。RPO 受限於容錯移轉目標複本的複寫延遲。

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 中的複寫是作用中的。反之，您可以使用 Amazon Route 53 將應用程式路由到最接近的運作狀態區域。若要進一步了解 Route 53，請參閱[什麼是 Amazon Route 53？](#)。

如果單一 AWS 區域變得隔離或降級，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，以針對不同的複本資料表執行讀取和寫入。您也可以套用自訂商業邏輯，以判斷何時將請求重新導向至其他區域。例如，讓您的應用程式了解多個可用的端點。

當區域恢復上線時，Amazon Keyspaces 會繼續將任何擱置的寫入從該區域傳播到其他區域中的複本資料表。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。

## 預設 AWS 區域 停用 中的多區域複寫

根據預設 AWS 區域，下列支援 Amazon Keyspaces 多區域複寫：

- 非洲 (開普敦) 區域
- 中東 (阿拉伯聯合大公國) 區域
- 亞太地區 (香港) 區域
- Middle East (Bahrain) Region

您必須先啟用區域，才能將預設停用的區域與 Amazon Keyspaces 多區域複寫搭配使用。如需詳細資訊，請參閱[AWS Organizations 《使用者指南》AWS 區域 中的在帳戶中啟用或停用](#)。

啟用區域之後，您可以在區域中建立新的 Amazon Keyspaces 資源，並將該區域新增至多區域金鑰空間。

當您停用 Amazon Keyspaces 多區域複寫所使用的區域時，Amazon Keyspaces 會啟動 24 小時寬限期。在此期間，您可以預期下列行為：

- Amazon Keyspaces 會持續在已啟用的區域中執行資料處理語言 (DML) 操作。
- Amazon Keyspaces 會暫停將資料更新從已啟用的區域複寫到已停用的區域。
- Amazon Keyspaces 會封鎖已停用區域中的所有資料定義語言 (DDL) 請求。



如果您錯誤地停用區域，您可以在 24 小時內重新啟用該區域。如果您在 24 小時寬限期內重新啟用區域，Amazon Keyspaces 將採取下列動作：

- 自動恢復所有複寫至重新啟用的區域。
- 在停用區域時複寫在已啟用區域中發生的任何資料更新，以確保資料一致性。
- 自動繼續所有其他多區域複寫操作。

如果區域在 24 小時時段關閉後仍處於停用狀態，Amazon Keyspaces 會採取下列動作，從多區域複寫中永久移除區域：

- 從所有多區域複寫金鑰空間中移除已停用的區域。
- 將已停用區域中的多區域複寫資料表複本轉換為單一區域金鑰空間和資料表。
- Amazon Keyspaces 不會從已停用的區域刪除任何資源。

Amazon Keyspaces 從多區域金鑰空間永久移除已停用的區域之後，您就無法新增已停用的區域。

## 多區域複寫和與point-in-time的整合

多區域資料表支援Point-in-time復原。若要使用 PITR 成功還原多區域資料表，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。
- 來源資料表的金鑰空間和目標資料表的金鑰空間的複寫區域必須相同。
- 必須在來源資料表的所有複本上啟用 PITR。

您可以從來源資料表可用的任何區域執行還原陳述式。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”](#)。

當您建立多區域資料表時，您在建立程序期間定義的 PITR 設定會自動套用至所有區域中的所有資料表。當您使用 變更 PITR 設定時ALTER TABLE，Amazon Keyspaces 只會將更新套用至本機資料表，而不會套用至其他區域中的複本。若要啟用現有多區域資料表的 PITR，您必須為所有複本重複 ALTER TABLE陳述式。

## 多區域複寫和與 AWS 服務的整合

您可以使用 Amazon CloudWatch AWS 區域 指標監控不同資料表之間的複寫效能。下列指標提供多區域金鑰空間的持續監控。

- ReplicationLatency – 此指標測量deletes從一個複本資料表複寫 updates、inserts或 到多區域金鑰空間中另一個複本資料表所需的時間。

如需如何監控 CloudWatch 指標的詳細資訊，請參閱 [the section called “使用 CloudWatch 進行監控”](#)。

## Amazon Keyspaces 多區域複寫用量備註

當您搭配 Amazon Keyspaces 使用多區域複寫時，請考慮下列事項。

- 您可以選取任何[可用的公](#) AWS 區域有。如需 AWS 區域 [預設停用之](#) 的詳細資訊，請參閱 [the section called “預設停用的區域”](#)。
- AWS GovCloud (US) Regions 不支援 和中國區域。
- 請考慮下列解決方法，直到功能可用為止：

建立多區域資料表時設定存留時間 (TTL)。您將無法啟用和停用 TTL，或稍後調整 TTL 值。如需詳細資訊，請參閱[the section called “使用存留時間過期資料”](#)。

- 對於靜態加密，請使用 AWS 擁有的金鑰。多區域資料表目前不支援客戶受管金鑰。如需詳細資訊，請參閱

[the section called “運作方式”](#)。

- 您可以使用 ALTER KEYSPACE將區域新增至單一區域或多區域金鑰空間。如需詳細資訊，請參閱[the section called “將區域新增至金鑰空間”](#)。
  - 將區域新增至單一區域金鑰空間之前，請確定金鑰空間下的資料表未設定客戶受管金鑰。
  - 針對金鑰空間或資料表設定的任何現有標籤都不會複寫到新區域。
- 當您搭配 Amazon Keyspaces 自動擴展使用佈建容量管理時，請務必使用 Amazon Keyspaces API 操作來建立和設定多區域資料表。Amazon Keyspaces 代您呼叫的基礎 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 “更新多區域資料表的佈建容量和自動擴展設定”](#)。如需如何預估佈建多區域資料表寫入容量輸送量的詳細資訊，請參閱 [the section called “多區域資料表的預估容量”](#)。

- 雖然資料會自動跨多區域資料表的所選區域複寫，但當用戶端連線到某個區域中的端點並查詢system.peers資料表時，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久保留對資料列的更新。然後，將寫入傳播到複寫的區域（或區域）會以非同步方式執行。

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資料表時，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資料表。當您使用 ALTER KEYSPACE或 變更現有標籤時ALTER TABLE，更新只會套用至您進行變更之區域中的鍵空間或資料表。
- Amazon CloudWatch ReplicationLatency 為每個複寫區域提供指標。它透過追蹤到達資料列、比較其到達時間與其初始寫入時間，以及計算平均值來計算此指標。計時會儲存在來源區域的CloudWatch 當中。如需詳細資訊，請參閱[the section called “使用 CloudWatch 進行監控”](#)。

檢視平均和最大時間，以判斷平均和最壞情況的複寫延遲會很有用。此延遲沒有 SLA。

- 在隨需模式下使用多區域資料表時，如果資料表複本遇到新的流量峰值，您可能會發現非同步複寫寫入的延遲增加。與 Amazon Keyspaces 自動將單一區域隨需資料表的容量調整為其接收的應用程式流量類似，Amazon Keyspaces 會自動將多區域隨需資料表複本的容量調整為其接收的流量。增加複寫延遲是暫時性的，因為 Amazon Keyspaces 會在流量增加時自動配置更多容量。一旦所有複本都適應您的流量，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”](#)。
- 在佈建模式中使用多區域資料表時，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和複寫延遲增加。為了確保所有 AWS 區域 多區域資料表中的所有資料表複本都有足夠的讀取和寫入容量，建議您設定 Amazon Keyspaces 自動擴展。Amazon Keyspaces Auto Scaling 透過自動調整輸送量容量以回應實際的應用程式流量，協助您有效率地佈建可變工作負載的輸送量容量。如需詳細資訊，請參閱[the section called “自動擴展如何適用於多區域資料表”](#)。

## 設定 Amazon Keyspaces 的多區域複寫（適用於 Apache Cassandra）

您可以使用 主控台、Cassandra 查詢語言 (CQL) 或 AWS Command Line Interface，在 Amazon Keyspaces 中建立和管理多區域金鑰空間和資料表。

本節提供如何建立和管理多區域金鑰空間和資料表的範例。您在多區域金鑰空間中建立的所有資料表會自動從金鑰空間繼承多區域設定。

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”](#)。

### 主題

- [設定建立多區域金鑰空間和資料表所需的 IAM 許可](#)

- [設定將 新增至金鑰空間所需的 IAM AWS 區域 許可](#)
- [在 Amazon Keyspaces 中建立多區域金鑰空間](#)
- [將 AWS 區域 新增至 Amazon Keyspaces 中的金鑰空間](#)
- [將新區域新增至金鑰空間時，請檢查複寫進度](#)
- [在 Amazon Keyspaces 中使用預設設定建立多區域資料表](#)
- [在 Amazon Keyspaces 中使用自動擴展的佈建模式中建立多區域資料表](#)
- [更新 Amazon Keyspaces 中多區域資料表的佈建容量和自動擴展設定](#)
- [在 Amazon Keyspaces 中檢視多區域資料表的佈建容量和自動擴展設定](#)
- [關閉 Amazon Keyspaces 中資料表的自動擴展](#)
- [在 Amazon Keyspaces 中手動設定多區域資料表的佈建容量](#)

## 設定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資料表，IAM 主體需要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。它包含 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 “多區域複寫”](#)。

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` – 委託人可執行的動作。
- `arn:aws:iam::*:role/aws-service-role/replication.cassandra.amazonaws.com/AWSServiceRoleForKeyspacesReplication` – 可以對其執行動作的資源。
- `iam:AWSServiceName`: `"replication.cassandra.amazonaws.com"` – 此角色可連接的唯一 AWS 服務是 Amazon Keyspaces。

以下是將最低必要許可授予委託人以建立多區域金鑰空間和資料表的政策範例。

```
{
 "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"}}
}
```

如需多區域金鑰空間和資料表的其他 IAM 許可，請參閱《服務授權參考》中的 [Amazon Keyspaces \(適用於 Apache Cassandra\) 的動作、資源和條件金鑰](#)。

## 設定將 新增至金鑰空間所需的 IAM AWS 區域 許可

若要將區域新增至金鑰空間，IAM 主體需要下列許可：

- `cassandra:Alter`
- `cassandra:AlterMultiRegionResource`
- `cassandra:Create`
- `cassandra:CreateMultiRegionResource`
- `cassandra:Select`
- `cassandra:SelectMultiRegionResource`
- `cassandra:Modify`
- `cassandra:ModifyMultiRegionResource`

如果資料表是在佈建模式中設定並啟用自動擴展，則需要下列額外許可。

- `application-autoscaling:RegisterScalableTarget`
- `application-autoscaling:DeregisterScalableTarget`
- `application-autoscaling:DescribeScalableTargets`
- `application-autoscaling:PutScalingPolicy`
- `application-autoscaling:DescribeScalingPolicies`

若要成功將區域新增至單一區域金鑰空間，IAM 主體也需要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。它包含 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 “多區域複寫”](#)。

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` – 委託人可執行的動作。
- `arn:aws:iam::*:role/aws-service-role/replication.cassandra.amazonaws.com/AWSServiceRoleForKeyspacesReplication` – 可以對其執行動作的資源。
- `iam:AWSServiceName`: "replication.cassandra.amazonaws.com – 此角色可連接的唯一 AWS 服務是 Amazon Keyspaces。

以下是將最低必要許可授予委託人，以將區域新增至金鑰空間的政策範例。

```
{
 "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"}}
}
```

## 在 Amazon Keyspaces 中建立多區域金鑰空間

本節提供如何建立多區域金鑰空間的範例。您可以使用 CQL 或在 Amazon Keyspaces 主控台上執行此操作 AWS CLI。您在多區域金鑰空間中建立的所有資料表會自動從金鑰空間繼承多區域設定。

### Note

建立多區域金鑰空間時，Amazon Keyspaces 會使用 `AWSServiceRoleForAmazonKeyspacesReplication` 帳戶中的名稱建立服務連結角色。此角色允許 Amazon Keyspaces 代表您將寫入複寫到多區域資料表的所有複本。如需詳細資訊，請參閱 [the section called “多區域複寫”](#)。

## Console

### 建立多區域金鑰空間（主控台）

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇金鑰空間，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. 在多區域複寫區段中，您可以新增清單中可用的其他區域。
5. 若要完成，請選擇建立金鑰空間。

## Cassandra Query Language (CQL)

### 使用 CQL 建立多區域金鑰空間

1. 若要建立多區域金鑰空間，請使用 `NetworkTopologyStrategy` 指定要複寫金鑰空間 AWS 區域的。您必須包含目前的區域和至少一個額外的區域。

金鑰空間中的所有資料表都會從金鑰空間繼承複寫策略。您無法在資料表層級變更複寫策略。

`NetworkTopologyStrategy` – 每個區域的複寫因素為三個 AWS 區域，因為 Amazon Keyspaces 預設會將資料複寫到相同區域內[的三個可用區域](#)。

下列 CQL 陳述式為範例。

```
CREATE KEYSPACE mykeyspace
WITH REPLICATION = {'class': 'NetworkTopologyStrategy', 'us-east-1': '3', 'ap-southeast-1': '3', 'eu-west-1': '3' };
```

2. 您可以使用 CQL 陳述式查詢 `system_multiregion_info` 金鑰空間中的 `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狀態。下列程式碼為範例。

```
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 |

## CLI

### 使用 建立新的多區域金鑰空間 AWS CLI

- 若要建立多區域金鑰空間，您可以使用下列 CLI 陳述式。在 中指定您目前的區域和至少一個額外的區域 `regionList`。

```
aws keyspaces create-keyspace --keyspace-name mykeyspace \
```



```
--replication-specification replicationStrategy=MULTI_REGION,regionList=us-east-1,eu-west-1
```

若要建立多區域資料表，請參閱 [the section called “使用預設設定建立多區域資料表”](#)和 [the section called “在佈建模式中建立多區域資料表”](#)。

## 將 AWS 區域 新增至 Amazon Keyspaces 中的金鑰空間

您可以將新的 AWS 區域 新增至單一或多區域金鑰空間的金鑰空間。新的複本區域會套用至金鑰空間中的所有資料表。

若要將單一區域變更為多區域金鑰空間，您必須為金鑰空間中的所有資料表啟用用戶端時間戳記。如需詳細資訊，請參閱[the section called “用戶端時間戳記”](#)。

如果您要將其他區域新增至多區域金鑰空間，Amazon Keyspaces 必須使用每個現有資料表的一次性跨區域還原，將現有資料表複寫至新區域。每個資料表的還原費用按 GB 計費，如需詳細資訊，請參閱 Amazon Keyspaces (適用於 Apache Cassandra) 定價頁面上的[備份和還原](#)。針對此還原操作，跨區域傳輸資料無需付費。除了資料之外，除了標籤之外的所有資料表屬性都會複寫到新的區域。

您可以在 CQL、update-keyspace 命令與 AWS CLI 或 主控台中使用 ALTER KEYSPACE 陳述式，將新區域新增至 Amazon Keyspaces 中的單一或多區域金鑰空間。若要成功執行 陳述式，您使用的帳戶必須位於已提供 金鑰空間的其中一個區域中。正在新增複本時，您無法對正在更新和複寫的資源執行任何其他資料定義語言 (DDL) 操作。

如需新增區域所需許可的詳細資訊，請參閱 [the section called “設定 IAM 許可以新增區域”](#)。

### Note

將其他區域新增至單一區域金鑰空間時，Amazon Keyspaces 會使用 `AWSServiceRoleForAmazonKeyspacesReplication` 您帳戶中的名稱建立服務連結角色。此角色允許 Amazon Keyspaces 將資料表複寫至新區域，並代表您將寫入從一個資料表複寫至多區域資料表的所有複本。如需詳細資訊，請參閱 [the section called “多區域複寫”](#)。

## Console

請依照下列步驟，使用 Amazon Keyspaces 主控台將區域新增至金鑰空間。



## 將區域新增至金鑰空間（主控台）

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇金鑰空間，然後從清單中選擇金鑰空間。
3. 選擇 (AWS 區域) 索引標籤。
4. 在 AWS 區域索引標籤上，選擇新增區域。
5. 在新增區域對話方塊中，選擇您要新增至金鑰空間的其他區域。
6. 若要完成，請選擇新增。

## Cassandra Query Language (CQL)

### 使用 CQL 將區域新增至金鑰空間

- 若要將新區域新增至金鑰空間，您可以使用下列陳述式。在此範例中，美國東部（維吉尼亞北部）區域和美國西部（奧勒岡）區域已提供金鑰空間，而 CQL 陳述式正在新增美國西部（加利佛尼亞北部）區域。

```
ALTER KEYSPACE my_keyspace
WITH REPLICATION = {
 'class': 'NetworkTopologyStrategy',
 'us-east-1': '3',
 'us-west-2': '3',
 'us-west-1': '3'
} AND CLIENT_SIDE_TIMESTAMPS = {'status': 'ENABLED'};
```

## CLI

### 使用 將區域新增至金鑰空間 AWS CLI

- 若要使用 CLI 將新區域新增至金鑰空間，您可以使用下列範例。請注意，client-side-timestamps 預設值為 DISABLED。使用 update-keyspace 命令，您必須將值變更為 ENABLED。

```
aws keyspaces update-keyspace \
--keyspace-name my_keyspace \
--replication-specification '{"replicationStrategy": "MULTI_REGION",
"regionList": ["us-east-1", "eu-west-1", "eu-west-3"] }' \
```

```
--client-side-timestamps '{"status": "ENABLED"}'
```

## 將新區域新增至金鑰空間時，請檢查複寫進度

將新區域新增至 Amazon Keyspaces 金鑰空間是一項長時間執行的操作。若要追蹤進度，您可以使用本節中顯示的查詢。

### Cassandra Query Language (CQL)

#### 使用 CQL 驗證新增區域進度

- 若要驗證在指定金鑰空間中建立新資料表複本的進度，您可以查詢 `system_multiregion_info.keyspaces` 資料表。下列 CQL 陳述式為範例。

```
SELECT keyspace_name, region, status, tables_replication_progress
FROM system_multiregion_info.keyspaces
WHERE keyspace_name = 'my_keyspace';
```

當複寫操作正在進行時，狀態會顯示新區域中資料表建立的進度。這是 10 個資料表中有 5 個已複寫到新區域的範例。

| keyspace_name | region    | status   | tables_replication_progress |
|---------------|-----------|----------|-----------------------------|
| my_keyspace   | us-east-1 | Updating |                             |
| my_keyspace   | us-west-2 | Updating |                             |
| my_keyspace   | eu-west-1 | Creating | 50%                         |

複寫程序成功完成後，輸出應如下所示。

| keyspace_name | region    | status |
|---------------|-----------|--------|
| my_keyspace   | us-east-1 | Active |
| my_keyspace   | us-west-2 | Active |
| my_keyspace   | eu-west-1 | Active |

## CLI

### 使用 AWS CLI 驗證新增區域進度

- 若要確認指定金鑰空間的資料表複本建立狀態，您可以使用下列範例。

```
aws keyspaces get-keyspace \
--keyspace-name my_keyspace
```

輸出看起來應該與此範例類似。

```
{
 "keyspaceName": "my_keyspace",
 "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
my_keyspace/",
 "replicationStrategy": "MULTI_REGION",
 "replicationRegions": [
 "us-east-1",
 "eu-west-1"
]
 "replicationGroupStatus": [
 {
 "RegionName": "us-east-1",
 "KeyspaceStatus": "Active"
 },
 {
 "RegionName": "eu-west-1",
 "KeyspaceStatus": "Creating",
 "TablesReplicationProgress": "50.0%"
 }
]
}
```

## 在 Amazon Keyspaces 中使用預設設定建立多區域資料表

本節提供如何在隨需模式下使用所有預設設定建立多區域資料表的範例。您可以使用 CQL 或在 Amazon Keyspaces 主控台上執行此操作 AWS CLI。您在多區域金鑰空間中建立的所有資料表會自動從金鑰空間繼承多區域設定。

若要建立多區域金鑰空間，請參閱 [the section called “建立多區域金鑰空間”](#)。

## Console

使用預設設定建立多區域資料表（主控台）

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 選擇多區域金鑰空間。
3. 在資料表索引標籤上，選擇建立資料表。
4. 針對資料表名稱，輸入資料表的名稱。正在複寫 AWS 區域 此資料表的 會顯示在資訊方塊中。
5. 繼續執行資料表結構描述。
6. 在資料表設定下，繼續執行預設設定選項。請注意多區域資料表的下列預設設定。
  - 容量模式 – 預設容量模式為隨需。如需設定佈建模式的詳細資訊，請參閱 [the section called “在佈建模式中建立多區域資料表”](#)。
  - 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。
- 如果您需要開啟資料表及其所有複本的存留時間 (TTL)，請選擇自訂設定。

### Note

您無法變更現有多區域資料表上的 TTL 設定。

7. 若要完成，請選擇建立資料表。

## Cassandra Query Language (CQL)

使用預設設定在隨需模式下建立多區域資料表

- 若要建立具有預設設定的多區域資料表，您可以使用下列 CQL 陳述式。

```
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'
 }
 }
};

```

## CLI

### 使用 AWS CLI

1. 若要使用預設設定建立多區域資料表，您只需要指定結構描述。您可以使用下列範例。

```
aws keyspaces create-table --keyspace-name mykeyspace --table-name mytable \
--schema-definition 'allColumns=[{name=pk,type=int}],partitionKeys={name= pk}'
```

命令的輸出為：

```

{
 "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
mykeyspace/table/mytable"
}

```

2. 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
```

輸出會顯示多區域資料表的所有預設設定。

```

{
 "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
 }
 }
]
}
```

```
}
 }
] }
}
```

## 在 Amazon Keyspaces 中使用自動擴展的佈建模式中建立多區域資料表

本節提供如何使用自動擴展在佈建模式中建立多區域資料表的範例。您可以使用 CQL 或在 Amazon Keyspaces 主控台上執行此操作 AWS CLI。

如需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”](#)。

若要建立多區域金鑰空間，請參閱 [the section called “建立多區域金鑰空間”](#)。

當您使用自動擴展設定在佈建模式中建立新的多區域資料表時，您可以指定資料表的一般設定，這些設定對複寫 AWS 區域 資料表的所有項目都有效。然後，您可以覆寫每個複本的讀取容量設定和讀取自動擴展設定。不過，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跨所有區域複寫寫入。

### Note

Amazon Keyspaces 自動擴展需要存在可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”](#)。

## Console

### 建立啟用自動擴展的新多區域資料表

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 選擇多區域金鑰空間。
3. 在資料表索引標籤上，選擇建立資料表。
4. 在資料表詳細資訊區段的建立資料表頁面上，選取金鑰空間，並提供新資料表的名稱。
5. 在資料欄區段中，建立資料表的結構描述。
6. 在主索引鍵區段中，定義資料表的主索引鍵，然後選取選用的叢集資料欄。
7. 在資料表設定區段中，選擇自訂設定。

8. 繼續讀取/寫入容量設定。
9. 針對容量模式，選擇佈建。
10. 在讀取容量區段中，確認已自動選取擴展。

您可以選擇為複寫資料表的所有 AWS 區域 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，您可以為每個資料表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 – 輸入資料表應隨時準備好支援的最小輸送量層級的值。該值必須介於 1 和您帳戶的每秒最大輸送量配額之間（預設為 40,000）。
- 最大容量單位 – 輸入您要為資料表佈建的最大輸送量。該值必須介於 1 和您帳戶的每秒最大輸送量配額之間（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，容量會自動擴展。當流量低於定義的目標時，會自動再次縮減規模。
- 如果您想要手動佈建資料表的讀取容量，請清除自動調整規模核取方塊。此設定適用於資料表的所有複本。

 Note

為了確保所有複本有足夠的讀取容量，我們建議針對佈建的多區域資料表進行 Amazon Keyspaces 自動擴展。

 Note

若要進一步了解您帳戶的預設配額以及如何增加這些配額，請參閱 [配額](#)。

11. 在寫入容量區段中，確認已自動選取擴展。然後設定資料表的容量單位。寫入容量單位會保持同步，AWS 區域 以確保有足夠的容量跨區域複寫寫入事件。
  - 如果您想要手動佈建資料表的寫入容量，請自動清除擴展。此設定適用於資料表的所有複本。



 Note

為了確保所有複本有足夠的寫入容量，我們建議針對佈建的多區域資料表進行 Amazon Keyspaces 自動擴展。

12. 選擇 Create Table (建立資料表)。您的資料表是使用指定的自動擴展參數建立的。

## Cassandra Query Language (CQL)

使用 CQL 建立具有佈建容量模式和自動擴展的多區域資料表

- 若要使用自動擴展在佈建模式中建立多區域資料表，您必須先定義資料表CUSTOM\_PROPERTIES的 來指定容量模式。指定佈建容量模式之後，您可以使用 來設定資料表的自動擴展設定AUTOSCALING\_SETTINGS。

如需自動擴展設定、目標追蹤政策、目標值和選用設定的詳細資訊，請參閱 [the section called “使用自動擴展建立新的資料表”](#)。

若要定義特定區域中資料表複本的讀取容量，您可以將下列參數設定為資料表 的一部分replica\_updates：

- 區域
- 佈建的讀取容量單位（選用）
- 讀取容量的自動擴展設定（選用）

下列範例顯示佈建模式中多區域資料表的CREATE TABLE陳述式。一般寫入和讀取容量自動擴展設定相同。不過，讀取自動擴展設定會指定額外的冷卻時間 60 秒，然後再將資料表的讀取容量向上或向下擴展。此外，美國東部區域（維吉尼亞北部）的讀取容量自動擴展設定高於其他複本的讀取容量自動擴展設定。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
 }
 }
 }
 }
 }
};

```

## CLI

使用 在佈建模式下建立具有自動擴展的新多區域資料表 AWS CLI

- 若要使用自動擴展組態在佈建模式中建立多區域資料表，您可以使用 AWS CLI。請注意，您必須使用 Amazon Keyspaces CLI `create-table` 命令來設定多區域自動擴展設定。這是因為

Application Auto Scaling 是 Amazon Keyspaces 用來代表您執行自動擴展的服務，不支援多個區域。

如需自動擴展設定、目標追蹤政策、目標值和選用設定的詳細資訊，請參閱 [the section called “使用自動擴展建立新的資料表”](#)。

若要定義特定區域中資料表複本的讀取容量，您可以將下列參數設定為資料表的一部分 replicaSpecifications：

- 區域
- 佈建的讀取容量單位（選用）
- 讀取容量的自動擴展設定（選用）

當您使用複雜的自動擴展設定和資料表複本的不同組態建立佈建多區域資料表時，從 JSON 檔案載入資料表的自動擴展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，並擷取 auto-scaling.json 和 replication.json。請記下檔案的路徑。

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 Amazon Keyspaces 中多區域資料表的佈建容量和自動擴展設定

本節包含如何使用主控台、CQL 和 AWS CLI 來管理佈建多區域資料表之 Amazon Keyspaces 自動擴展設定的範例。如需一般自動擴展組態選項及其運作方式的詳細資訊，請參閱 [the section called “使用自動擴展管理輸送量容量”](#)。

請注意，如果您使用多區域資料表的佈建容量模式，您必須一律使用 Amazon Keyspaces API 呼叫來設定自動擴展。這是因為基礎 Application Auto Scaling API 操作不熟悉區域。

如需如何預估佈建多區域資料表寫入容量輸送量的詳細資訊，請參閱 [the section called “多區域資料表的預估容量”](#)。

如需 Amazon Keyspaces API 的詳細資訊，請參閱 [Amazon Keyspaces API 參考](#)。

當您更新多區域資料表的佈建模式或自動擴展設定時，您可以更新資料表每個複本的讀取容量設定和讀取自動擴展組態。

不過，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跨所有區域複寫寫入。

## Cassandra Query Language (CQL)

### 使用 CQL 更新多區域資料表的佈建容量和自動擴展設定

- 您可以使用 ALTER TABLE 更新現有資料表的容量模式和自動擴展設定。如果您要更新目前處於隨需容量模式的資料表，capacity\_mode 則需要。如果您的資料表已處於佈建容量模式，則可以省略此欄位。

如需自動擴展設定、目標追蹤政策、目標值和選用設定的詳細資訊，請參閱 [the section called “使用自動擴展建立新的資料表”](#)。

在相同的陳述式中，您也可以更新資料表的 replica\_updates 屬性，以更新特定區域中資料表複本的讀取容量和自動擴展設定。下列陳述式為範例。

```
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
 }
 }
 }
 }
 }
}
};

```

## CLI

### 使用 更新多區域資料表的佈建容量和自動擴展設定 AWS CLI

- 若要更新現有資料表的佈建模式和自動擴展組態，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自動擴展設定。這是因為 Application Auto Scaling 是 Amazon Keyspaces 用來代表您執行資料表容量自動擴展的服務，不支援多個 AWS 區域。

若要更新特定區域中資料表複本的讀取容量，您可以變更資料表的下列其中一個選用參數 `replicaSpecifications`：

- 佈建的讀取容量單位（選用）
- 讀取容量的自動擴展設定（選用）

當您使用複雜的自動擴展設定和資料表複本的不同組態更新多區域資料表時，從 JSON 檔案載入資料表的自動擴展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，並解壓縮 `auto-scaling.json` 和 `replication.json`。請記下檔案的路徑。

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
```

## 在 Amazon Keyspaces 中檢視多區域資料表的佈建容量和自動擴展設定

您可以使用 CQL 或在 Amazon Keyspaces 主控台上檢視多區域資料表的佈建容量和自動擴展設定 AWS CLI。本節提供如何使用 CQL 和執行此操作的範例 AWS CLI。

### Cassandra Query Language (CQL)

使用 CQL 檢視多區域資料表的佈建容量和自動擴展設定

- 若要檢視多區域資料表的自動擴展組態，請使用下列命令。

```
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}}}

```

## CLI

使用 檢視多區域資料表的佈建容量和自動擴展設定 AWS CLI

- 若要檢視多區域資料表的自動擴展組態，您可以使用 `get-table-auto-scaling-settings` 操作。下列 CLI 命令為範例。

```
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 Amazon Keyspaces 中資料表的自動擴展

本節提供如何在佈建容量模式中關閉多區域資料表自動擴展的範例。您可以使用 CQL 或在 Amazon Keyspaces 主控台上執行此操作 AWS CLI。

### Important

對於使用佈建容量模式的多區域資料表，我們建議您使用自動擴展。如需詳細資訊，請參閱[the section called “多區域資料表的預估容量”](#)。

### Note

若要刪除 Application Auto Scaling 使用的服務連結角色，您必須在帳戶中所有 的所有資料表上停用自動擴展 AWS 區域。

## Console

關閉主控台上現有多區域資料表的 Amazon Keyspaces 自動擴展

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 選擇您要使用的資料表，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。停用自動擴展會將資料表取消註冊為 Application Auto Scaling 的可擴展目標。若要刪除 Application Auto Scaling 用來存取 Amazon Keyspaces 資料表的服務連結角色，請遵循中的步驟[the section called “刪除 Amazon Keyspaces 的服務連結角色”](#)。
5. 定義自動擴展設定時，請選擇儲存。

## Cassandra Query Language (CQL)

### 使用 CQL 關閉多區域資料表的自動擴展

- 您可以使用 ALTER TABLE 關閉現有資料表的自動擴展。請注意，您無法關閉個別資料表複本的自動擴展。

在下列範例中，資料表讀取容量的自動擴展已關閉。

```
ALTER TABLE mykeyspace.mytable
WITH AUTOSCALING_SETTINGS = {
 'provisioned_read_capacity_autoscaling_update': {
 'autoscaling_disabled': true
 }
};
```

## CLI

### 使用 關閉多區域資料表的自動擴展 AWS CLI

- 您可以使用 AWS CLI update-table 命令來關閉現有資料表的自動擴展。請注意，您無法關閉個別資料表複本的自動擴展。

在下列範例中，資料表讀取容量的自動擴展已關閉。

```
aws keyspaces update-table --keyspace-name mykeyspace --table-name mytable
 \ --auto-scaling-specification
 readCapacityAutoScaling={autoScalingDisabled=true}
```

## 在 Amazon Keyspaces 中手動設定多區域資料表的佈建容量

如果您必須關閉多區域資料表的自動擴展，則可以使用 CQL 或 手動佈建複本資料表的讀取容量 AWS CLI。

### Note

對於使用佈建容量模式的多區域資料表，我們建議您使用自動擴展。如需詳細資訊，請參閱[the section called “多區域資料表的預估容量”](#)。

## Cassandra Query Language (CQL)

### 使用 CQL 手動設定多區域資料表的佈建容量

- 您可以使用 ALTER TABLE 手動佈建複本資料表的資料表讀取容量。

```
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
 }
 }
};
```

## CLI

### 使用 手動設定多區域資料表的佈建容量 AWS CLI

- 如果您必須關閉多區域資料表的自動擴展，您可以使用 update-table 手動佈建複本資料表的讀取容量。

```
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
```

## 使用 point-in-time 復原來備份和還原資料

Point-in-time 復原 (PITR) 透過提供資料表資料的連續備份，協助保護您的 Amazon Keyspaces 資料表免於意外寫入或刪除操作。

例如，假設測試指令碼意外寫入生產 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至過去 35 天內啟用 PITR 後的任何一秒。如果您刪除已啟用 point-in-time 復原的資

料表，您可以查詢已刪除資料表的資料 35 天（無需額外費用），並將其還原至剛好在刪除點之前的狀態。

您可以使用 主控台、AWS SDK 和 AWS Command Line Interface (AWS CLI) 或 Cassandra 查詢語言 (CQL)，將 Amazon Keyspaces 資料表還原至某個時間點。如需詳細資訊，請參閱[在 point-in-time 復原](#)。

Point-in-time 操作對基礎資料表沒有效能或可用性影響，還原資料表不會消耗額外的輸送量。

如需 PITR 配額的詳細資訊，請參閱 [配額](#)。

如需定價的相關資訊，請參閱 [Amazon Keyspaces（適用於 Apache Cassandra）定價](#)。

## 主題

- [point-in-time 復原如何在 Amazon Keyspaces 中運作](#)
- [在 point-in-time 復原](#)

## point-in-time 復原如何在 Amazon Keyspaces 中運作

本節概述 Amazon Keyspaces point-in-time 復原 (PITR) 的運作方式。如需定價的詳細資訊，請參閱 [Amazon Keyspaces（適用於 Apache Cassandra）定價](#)。

## 主題

- [PITR 連續備份的時間範圍](#)
- [PITR 還原設定](#)
- [加密資料表的 PITR 還原](#)
- [多區域資料表的 PITR 還原](#)
- [使用使用者定義類型 \(UDTs\) 還原資料表的 PITR](#)
- [PITR 的資料表還原時間](#)
- [Amazon Keyspaces PITR 和 AWS 服務整合](#)

## PITR 連續備份的時間範圍

Amazon Keyspaces PITR 使用兩個時間戳記來維護資料表可使用可還原備份的時間範圍。

- 最早可還原時間 – 標記最早可還原備份的時間。最早的可還原備份最多可備份 35 天，或啟用 PITR 時，以較新者為準。35 天的最大備份時段無法修改。

- 目前時間 – 最新可還原備份的時間戳記是目前時間。如果在還原期間未提供時間戳記，則會使用目前時間。

啟用 PITR 時，您可以還原至 `EarliestRestorableDateTime` 和 之間的任何時間點 `CurrentTime`。您只能將資料表資料還原至啟用 PITR 的時間。

如果您停用 PITR 之後再次啟用，您可以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 會清除您的備份歷史記錄。

#### Note

資料表上的資料定義語言 (DDL) 操作，例如結構描述變更，會以非同步方式執行。您只能在還原的資料表資料中查看已完成的操作，但如果來源資料表在還原時正在進行，您可能會在來源資料表上看到其他動作。如需 DDL 陳述式的清單，請參閱 [the section called “DDL 陳述式”](#)。

資料表不需要處於作用中狀態即可還原。如果在刪除的資料表上啟用 PITR，且刪除發生在備份時段（或過去 35 天內），您也可以還原刪除的資料表。

#### Note

如果使用與先前刪除的資料表相同的合格名稱（例如 `mykeyspace.mytable`）建立新資料表，則刪除的資料表將無法再還原。如果您嘗試從主控台執行此操作，則會顯示警告。

## PITR 還原設定

當您使用 PITR 還原資料表時，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至 狀態。PITR 不會覆寫現有的資料表。

除了資料表的結構描述和資料之外，PITR 還會 `custom_properties` 從來源資料表還原。與根據最早還原時間和目前時間之間選取的時間戳記還原的資料表資料不同，自訂屬性一律會根據截至目前時間的資料表設定還原。

還原資料表的設定會比對來源資料表的設定，以及啟動還原時的時間戳記。如果您想要在還原期間覆寫這些設定，您可以使用 來執行此操作 `WITH custom_properties`。自訂屬性包括下列設定。

- 讀取/寫入容量模式

- 佈建的輸送量容量設定
- PITR 設定

如果資料表處於佈建容量模式且已啟用自動擴展，則還原操作也會還原資料表的自動擴展設定。您可以使用 CQL 中的 `autoscaling_settings` 參數或使用 CLI `autoScalingSpecification` 覆寫它們。如需自動擴展設定的詳細資訊，請參閱 [the section called “使用自動擴展管理輸送量容量”](#)。

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，您將資料表的狀態還原至三週前。目前，其佈建的輸送量設定為 100 個讀取容量單位和 100 個寫入容量單位。在此情況下，Amazon Keyspaces 會將資料表資料還原到該時間點，但會使用目前的佈建輸送量設定 (50 個讀取容量單位和 50 個寫入容量單位)。

不會還原下列設定，您必須手動為新資料表設定這些設定。

- Amazon Keyspaces 變更資料擷取 (CDC) 串流
- AWS Identity and Access Management (IAM) 政策
- Amazon CloudWatch 指標和警示
- 標籤 (可使用新增至 CQL RESTORE 陳述式 WITH TAGS)

## 加密資料表的 PITR 還原

當您使用 PITR 還原資料表時，Amazon Keyspaces 會還原來源資料表的加密設定。如果資料表使用 AWS 擁有的金鑰 (預設) 加密，資料表會自動以相同的設定還原。如果您要還原的資料表是使用客戶受管金鑰加密的，Amazon Keyspaces 需要存取相同的客戶受管金鑰才能還原資料表資料。

您可以在還原時變更資料表的加密設定。若要從 AWS 擁有的金鑰變更為客戶受管金鑰，您需要在還原時提供有效且可存取的客戶受管金鑰。

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，以使用還原資料表 AWS 擁有的金鑰。如需資料表靜態加密設定的詳細資訊，請參閱 [the section called “運作方式”](#)。

### Note

如果因為 Amazon Keyspaces 無法存取客戶受管金鑰而刪除資料表，您需要確保 Amazon Keyspaces 可以存取客戶受管金鑰，然後再嘗試還原資料表。如果 Amazon Keyspaces 無法

存取該金鑰，則無法還原使用客戶受管金鑰加密的資料表。如需詳細資訊，請參閱 [《開發人員指南》中的對金鑰存取進行故障診斷](#)。AWS Key Management Service

## 多區域資料表的 PITR 還原

您可以使用 PITR 還原多區域資料表。若要讓還原操作成功，必須在來源資料表的所有複本上啟用 PITR，且來源和目的地資料表都必須複寫至相同的複本 AWS 區域。

Amazon Keyspaces 會在屬於金鑰空間的每個複寫區域中還原來源資料表的設定。您也可以在這還原操作期間覆寫設定。如需可在還原期間變更之設定的詳細資訊，請參閱 [the section called “還原設定”](#)。

如需多區域複寫的詳細資訊，請參閱 [the section called “運作方式”](#)。

## 使用使用者定義類型 (UDTs) 還原資料表的 PITR

您可以還原使用 UDTs 資料表。若要讓還原操作成功，參考 UDTs 必須在 金鑰空間中存在且有效。

如果您在嘗試還原資料表時遺失任何必要的 UDT，Amazon Keyspaces 會嘗試自動還原 UDT 結構描述，然後繼續還原資料表。

如果您移除並重新建立 UDT，Amazon Keyspaces 會使用 UDT 的新結構描述還原 UDT，並拒絕使用原始 UDT 結構描述還原資料表的請求。在此情況下，如果您想要使用舊的 UDT 結構描述還原資料表，您可以將資料表還原至新的金鑰空間。當您刪除並重新建立 UDT 時，即使重新建立 UDT 的結構描述與已刪除 UDT 的結構描述相同，重新建立的 UDT 也會被視為新的 UDT。在此情況下，Amazon Keyspaces 會拒絕使用舊 UDT 結構描述還原資料表的請求。

如果 UDT 遺失，且 Amazon Keyspaces 嘗試還原 UDT，則如果您已達到區域中帳戶的 UDTs 數目上限，則嘗試會失敗。

如需 UDT 配額和預設值的詳細資訊，請參閱 [the section called “Amazon Keyspaces 中使用者定義類型 \(UDTs\) 的配額和預設值”](#)。如需使用 UDTs 的詳細資訊，請參閱 [the section called “使用者定義的類型 \(UDTs\)”](#)。

## PITR 的資料表還原時間

還原資料表所需的時間是以多個因素為基礎，且不一定與資料表的大小直接相關。

以下是還原時間的一些考量。



- 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)。
- 對於具有分佈良好的資料模型的大型資料表，還原時間可以是數小時或更長時間。
- 如果您的來源資料表包含大幅扭曲的資料，還原時間可能會增加。例如，如果您資料表的主索引鍵使用當月做為分割區索引鍵，而且所有資料都是從 12 月開始，表示您資料偏斜。

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點目標。

## Amazon Keyspaces PITR 和 AWS 服務整合

使用 記錄下列 PITR 操作 AWS CloudTrail，以啟用持續監控和稽核。

- 建立啟用或停用 PITR 的新資料表。
- 在現有資料表上啟用或停用 PITR。
- 還原作用中或已刪除的資料表。

如需詳細資訊，請參閱[使用 記錄 Amazon Keyspaces API 呼叫 AWS CloudTrail](#)。

您可以使用 執行下列 PITR 動作 CloudFormation。

- 建立啟用或停用 PITR 的新資料表。
- 在現有資料表上啟用或停用 PITR。

如需詳細資訊，請參閱[CloudFormation 《使用者指南》](#)中的 [Cassandra 資源類型參考](#)。

## 在 point-in-time 復原

使用 Amazon Keyspaces (適用於 Apache Cassandra)，您可以使用時間點還原 (PITR) Point-in-Time 將資料表還原至特定時間點。PITR 可讓您將資料表還原至過去 35 天內的狀態，提供資料保護和復原功能。此功能在意外刪除資料、應用程式錯誤或用於測試等情況下非常有用。您可以快速有效地復原資料，將停機時間和資料遺失降到最低。以下各節會引導您完成在 Amazon Keyspaces 中使用 PITR 還原資料表的程序，以確保資料完整性和業務連續性。

### 主題

- [設定 Amazon Keyspaces PITR 的還原資料表 IAM 許可](#)
- [在 Amazon Keyspaces 中設定資料表的 PITR](#)

- [關閉 Amazon Keyspaces 資料表的 PITR](#)
- [將資料表從備份還原到 Amazon Keyspaces 中指定的時間點](#)
- [使用 Amazon Keyspaces PITR 還原已刪除的資料表](#)

## 設定 Amazon Keyspaces PITR 的還原資料表 IAM 許可

本節摘要說明如何設定 AWS Identity and Access Management (IAM) 主體的許可來還原 Amazon Keyspaces 資料表。在 IAM 中，AWS 受管政策AmazonKeyspacesFullAccess包含還原 Amazon Keyspaces 資料表的許可。若要實作具有最低必要許可的自訂政策，請考慮下一節中概述的需求。

若要成功還原資料表，IAM 主體需要下列最低許可：

- `cassandra:Restore` – 目標資料表需要還原動作才能還原。
- `cassandra:Select` – 從來源資料表讀取時需要選取動作。
- `cassandra:TagResource` – 標籤動作是選用的，只有在還原操作新增標籤時才需要。

這是將最低必要許可授予使用者以還原金鑰空間 中的資料表的政策範例mykeyspace。

```
{
 "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，可能需要其他許可才能還原資料表。例如，如果來源資料表使用客戶受管金鑰進行靜態加密，Amazon Keyspaces 必須具有存取來源資料表之客戶受管金鑰的許可，才能成功還原資料表。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，您必須確保 Amazon Keyspaces 具有代表您委託人執行還原操作的許可。如果您的政策將傳入流量限制為下列任何項目，您必須將 `aws:ViaAWSService` 條件金鑰新增至 IAM 政策：

- 使用的 VPC 端點 `aws:SourceVpce`
- 使用的 IP 範圍 `aws:SourceIp`
- 使用 VPCs `aws:SourceVpc`

當任何 AWS 服務使用委託人的登入資料提出請求時，`aws:ViaAWSService` 條件金鑰允許存取。如需詳細資訊，請參閱《[IAM 使用者指南](#)》中的 [IAM JSON 政策元素：條件索引鍵](#)。

以下是將來源流量限制為特定 IP 地址，並允許 Amazon Keyspaces 代表委託人還原資料表的政策範例。

```
{
 "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` 全域條件金鑰的範例政策，請參閱 [the section called “VPC 端點政策和 Amazon Keyspaces point-in-time\(PITR\)”](#)。

## 在 Amazon Keyspaces 中設定資料表的 PITR

您可以在 Amazon Keyspaces 中設定資料表，以搭配主控台、CQL 和 使用 PITR 進行備份和還原操作 AWS CLI。

使用 CQL 或 建立新資料表時 AWS CLI，您必須在建立資料表陳述式中明確啟用 PITR。當您使用主控台建立新資料表時，預設會啟用 PITR。

若要了解如何還原資料表，請參閱 [the section called “將資料表還原至某個時間點”](#)。

### Console

#### 使用主控台設定資料表的 PITR

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇資料表，然後選取您要編輯的資料表。
3. 在備份索引標籤上，選擇編輯。
4. 在編輯point-in-time復原設定區段中，選取啟用Point-in-time復原。
5. 選擇儲存變更。

### Cassandra Query Language (CQL)

#### 使用 CQL 設定資料表的 PITR

1. 您可以使用 `point_in_time_recovery` 自訂屬性來管理資料表的 PITR 設定。

若要在建立新資料表時啟用 PITR，您必須將狀態設定為 `point_in_time_recovery enabled`。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point-in-time復原自訂屬性，則時間點point-in-time復原預設為停用。

- 若要使用 CQL 為現有資料表啟用 PITR，請執行下列 CQL 命令。

```
ALTER TABLE mykeyspace.mytable
WITH custom_properties = {'point_in_time_recovery': {'status': 'enabled'}}
```

## CLI

### 使用 設定資料表的 PITR AWS CLI

- 您可以使用 UpdateTable API 管理資料表的 PITR 設定。

若要在建立新資料表時啟用 PITR，您必須在建立資料表命令point-in-time-recovery 'status=ENABLED'中包含。您可以使用下列 AWS CLI 命令做為範例。命令已分成不同的行來改善可讀性。

```
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，則會預設停用point-in-time復原。

- 若要確認資料表的point-in-time復原設定，您可以使用下列 AWS CLI 命令。

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

- 若要使用 為現有資料表啟用 PITR AWS CLI，請執行下列命令。

```
aws keyspaces update-table --keyspace-name 'myKeyspace' --table-name 'myTable'
--point-in-time-recovery 'status=ENABLED'
```

## 關閉 Amazon Keyspaces 資料表的 PITR

您可以隨時使用主控台、CQL 或 來關閉 Amazon Keyspaces 資料表的 PITR AWS CLI。

### Important

停用 PITR 會立即刪除備份歷史記錄，即使您在 35 天內在資料表上重新啟用 PITR。

若要了解如何還原資料表，請參閱 [the section called “將資料表還原至某個時間點”](#)。

## Console

使用主控台停用資料表的 PITR

- 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
- 在導覽窗格中，選擇資料表，然後選取您要編輯的資料表。
- 在備份索引標籤上，選擇編輯。
- 在編輯point-in-time復原設定區段中，清除啟用Point-in-time復原核取方塊。
- 選擇儲存變更。

## Cassandra Query Language (CQL)

使用 CQL 停用資料表的 PITR

- 若要停用現有資料表的 PITR，請執行下列 CQL 命令。

```
ALTER TABLE mykeyspace.mytable
WITH custom_properties = {'point_in_time_recovery': {'status': 'disabled'}}
```

## CLI

### 使用 停用資料表的 PITR AWS CLI

- 若要停用現有資料表的 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'
```

## 將資料表從備份還原到 Amazon Keyspaces 中指定的時間點

下一節示範如何將現有的 Amazon Keyspaces 資料表還原至指定的時間點。

### Note

此程序假設您使用的資料表已設定point-in-time復原。若要啟用資料表的 PITR，請參閱 [the section called “設定 PITR”](#)。

### Important

還原進行中時，請勿修改或刪除授予 IAM 主體（例如使用者、群組或角色）執行還原許可的 AWS Identity and Access Management (IAM) 政策。否則，可能會造成意外行為。例如，如果您在資料表還原時移除資料表的寫入許可，基礎RestoreTableToPointInTime操作就無法將任何還原的資料寫入資料表。  
您只能在還原操作完成後修改或刪除許可。

## Console

### 使用主控台將資料表還原至指定的時間點

- 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
- 在主控台左側的導覽窗格中，選擇 Tables (資料表)。
- 在資料表清單中，選擇您要還原的資料表。
- 在資料表的備份索引標籤的Point-in-time復原區段中，選擇還原。

5. 針對新的資料表名稱，輸入還原資料表的新名稱，例如 **mytable\_restored**。
6. 若要定義還原操作的時間點，您可以選擇兩個選項：
  - 選取預先設定的最早時間。
  - 選取指定日期和時間，然後輸入您要還原新資料表的日期和時間。

 Note

您可以在最早時間和目前時間還原至任何時間點。Amazon Keyspaces 會根據選取的日期和時間（日：小時：分鐘：秒）將資料表資料還原至狀態。

7. 選擇還原以開始還原程序。

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，還原資料表的狀態會變更為作用中。

## Cassandra Query Language (CQL)

使用 CQL 將資料表還原至某個時間點

1. 您可以將作用中資料表還原至介於 point-in-time。 `earliest_restorable_timestamp` 目前時間是預設值。

若要確認資料表已啟用 point-in-time 復原，請查詢 `system_schema_mcs.tables`，如本範例所示。

```
SELECT custom_properties
FROM system_schema_mcs.tables
WHERE keyspace_name = 'mykeyspace' AND table_name = 'mytable';
```

Point-in-time 復原已啟用，如下列範例輸出所示。

```
custom_properties

{
 ...,
 "point_in_time_recovery": {
 "earliest_restorable_timestamp": "2020-06-30T19:19:21.175Z"
 "status": "enabled"
 }
}
```



```
}
```

2. 將資料表還原至目前時間。當您省略 `WITH restore_timestamp = ...` 子句時，會使用目前的時間戳記。

```
RESTORE TABLE mykeyspace.mytable_restored
FROM TABLE mykeyspace.mytable;
```

- 您也可以還原至特定時間點，由 ISO 8601 格式 `restore_timestamp` 的定義。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，請參閱語言參考[the section called “RESTORE 資料表”](#)中的。

3. 若要驗證資料表的還原是否成功，請查詢 `system_schema_mcs.tables` 以確認資料表的狀態。

```
SELECT status
FROM system_schema_mcs.tables
WHERE keyspace_name = 'mykeyspace' AND table_name = 'mytable_restored'
```

查詢會顯示下列輸出。

```
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，資料表的狀態會變更為作用中。

## CLI

使用 將資料表還原至某個時間點 AWS CLI

1. 建立名為 `myTable` 且已啟用 PITR 的簡單資料表。命令已分成不同的行以方便閱讀。

```
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. 確認新資料表的屬性，並檢閱 PITR `earliestRestorableTimestamp` 的。

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

此命令的輸出會傳回下列項目。

```
{
 "keyspaceName": "myKeyspace",
 "tableName": "myTable",
 "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/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": ""
 }
}

```

3. • 若要將資料表還原至某個時間點，`restore_timestamp`請以 ISO 8601 格式指定。您可以選擇過去 35 天內的任何時間點，每隔一秒鐘。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"
```

此命令的輸出會傳回還原資料表的 ARN。

```
{
 "restoredTableARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/myKeyspace/table/myTable_restored"
}
```

- 若要將資料表還原至目前時間，您可以省略 `restore-timestamp` 參數。

```
aws keyspaces restore-table --source-keyspace-name 'myKeyspace' --source-table-name 'myTable' --target-keyspace-name 'myKeyspace' --target-table-name 'myTable_restored1'
```

## 使用 Amazon Keyspaces PITR 還原已刪除的資料表

下列程序說明如何將已刪除的資料表從備份還原至刪除時間。您可以使用 CQL 或來執行此操作 AWS CLI。

**Note**

此程序假設刪除的資料表上已啟用 PITR。

## Cassandra Query Language (CQL)

### 使用 CQL 還原已刪除的資料表

- 若要確認已刪除資料表已啟用point-in-time復原，請查詢系統資料表。只會顯示已啟用point-in-time復原的資料表。

```
SELECT custom_properties
FROM system_schema_mcs.tables_history
WHERE keyspace_name = 'mykeyspace' AND table_name = 'my_table';
```

查詢會顯示下列輸出。

```
custom_properties

{
 ...,
 "point_in_time_recovery":{
 "restorable_until_time":"2020-08-04T00:48:58.381Z",
 "status":"enabled"
 }
}
```

- 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;
```

## CLI

### 使用 還原已刪除的資料表 AWS CLI

- 刪除您先前建立且已啟用 PITR 的資料表。下列是範例命令。

```
aws keyspaces delete-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

2. 使用下列命令將已刪除的資料表還原至刪除時間。

```
aws keyspaces restore-table --source-keyspace-name 'myKeyspace' --source-table-name 'myTable' --target-keyspace-name 'myKeyspace' --target-table-name 'myTable_restored2'
```

此命令的輸出會傳回還原資料表的 ARN。

```
{
 "restoredTableARN": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/myKeyspace/table/myTable_restored2"
}
```

## 使用 Amazon Keyspaces 的存留時間 (TTL) 過期資料 (適用於 Apache Cassandra)

Amazon Keyspaces (適用於 Apache Cassandra) 存留時間 (TTL) 可協助您簡化應用程式邏輯，並透過自動從資料表中過期資料來最佳化儲存體的價格。您不再需要的資料會根據您設定的存留時間值，自動從資料表中刪除。

這可讓您更輕鬆地根據業務、產業或法規要求遵循資料保留政策，以定義資料需要保留多久，或指定何時必須刪除資料。

例如，您可以在 AdTech 應用程式中使用 TTL 來排程特定廣告的資料過期且用戶端不再可見的時間。您也可以使用 TTL 自動淘汰較舊的資料，並節省儲存成本。

您可以為整個資料表設定預設 TTL 值，並覆寫個別資料列和資料欄的該值。TTL 操作不會影響您應用程式的效能。此外，標記為 TTL 過期的資料列和資料欄數量不會影響您資料表的可用性。

Amazon Keyspaces 會自動篩選掉過期的資料，這樣就不會在查詢結果中傳回過期的資料，或可用於資料處理語言 (DML) 陳述式。Amazon Keyspaces 通常會在過期日期的 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來保護可用性，Amazon Keyspaces 可能無法在 10 天內刪除資料。在這些情況下，當分割區上的流量減少時，Amazon Keyspaces 會繼續嘗試刪除過期的資料。

從儲存體中永久刪除資料後，您會停止產生儲存費用。

您可以使用主控台、Cassandra 查詢語言 (CQL) 或 `awscli` 來設定、修改或停用新資料表和現有資料表的預設 TTL 設定 AWS CLI。

在已設定預設 TTL 的資料表上，您可以使用 CQL 陳述式覆寫資料表的預設 TTL 設定，並將自訂 TTL 值套用至資料列和資料欄。如需詳細資訊，請參閱 [the section called “使用 INSERT 設定新資料列的自訂 TTL”](#) 及 [the section called “使用 UPDATE 設定資料列和資料欄的自訂 TTL”](#)。

TTL 定價是根據使用存留時間刪除或更新的資料列大小。TTL 操作的計量單位為 TTL deletes。刪除或更新的每列每 KB 資料會耗用一個 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欄，需要刪除三個 TTL。或者，若要刪除包含 3.5 KB 資料的整個資料列，需要刪除四個 TTL。

每列每 KB 的已刪除資料會耗用一個 TTL 刪除。如需定價的詳細資訊，請參閱 [Amazon Keyspaces \(適用於 Apache Cassandra\) 定價](#)。

## 主題

- [Amazon Keyspaces 存留時間以及與 AWS 服務的整合](#)
- [使用預設存留時間 \(TTL\) 設定建立新的資料表](#)
- [更新資料表的預設存留時間 \(TTL\) 值](#)
- [建立已啟用自訂存留時間 \(TTL\) 設定的資料表](#)
- [使用自訂存留時間 \(TTL\) 更新資料表](#)
- [使用 INSERT 陳述式來設定新資料列的自訂存留時間 \(TTL\) 值](#)
- [使用 UPDATE 陳述式來編輯資料列和資料欄的自訂存留時間 \(TTL\) 設定](#)

## Amazon Keyspaces 存留時間以及與 AWS 服務的整合

下列 TTL 指標可在 Amazon CloudWatch 中使用，以啟用持續監控。

- `TTLDeletes` – 使用存留時間 (TTL) 刪除或更新資料的資料所消耗的單位。

如需如何監控 CloudWatch 指標的詳細資訊，請參閱 [the section called “使用 CloudWatch 進行監控”](#)。

使用 `awscli` 時 CloudFormation，您可以在建立 Amazon Keyspaces 資料表時開啟 TTL。如需詳細資訊，請參閱 [「AWS CloudFormation 使用者指南」](#)。

## 使用預設存留時間 (TTL) 設定建立新的資料表

在 Amazon Keyspaces 中，您可以為建立資料表時資料表中的所有資料列設定預設 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，則會將過期時間戳記新增至每一列。

TTL 值以秒為單位設定，最大可設定值為 630,720,000 秒，相當於 20 年。

建立資料表後，您可以使用 CQL DML 陳述式覆寫特定資料列或資料欄的資料表預設 TTL 設定。如需詳細資訊，請參閱[the section called “使用 INSERT 設定新資料列的自訂 TTL”](#)及[the section called “使用 UPDATE 設定資料列和資料欄的自訂 TTL”](#)。

當您在資料表上啟用 TTL 時，Amazon Keyspaces 會開始儲存每個資料列的其他 TTL 相關中繼資料。此外，TTL 會使用過期時間戳記來追蹤資料列或資料欄何時過期。時間戳記會儲存為資料列中繼資料，並造成資料列的儲存成本。

啟用 TTL 功能之後，您就無法停用資料表的 TTL 功能。將資料表 `default_time_to_live` 設為 0 會停用新資料的預設過期時間，但不會停用 TTL 功能，或將資料表還原為原始 Amazon Keyspaces 儲存中繼資料或寫入行為。

下列範例示範如何使用預設 TTL 值建立新的資料表。

### Console

使用主控台建立具有存留時間預設值的新資料表。

1. 登入 AWS 管理主控台，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. 在資料表詳細資訊區段的建立資料表頁面上，選取金鑰空間，並提供新資料表的名稱。
4. 在結構描述區段中，建立資料表的結構描述。
5. 在資料表設定區段中，選擇自訂設定。
6. 繼續執行存留時間 (TTL)。

在此步驟中，您會選取資料表的預設 TTL 設定。

針對預設 TTL 期間，輸入過期時間，然後選擇您輸入的時間單位，例如秒、天或年。Amazon Keyspaces 會以秒為單位存放值。

7. 選擇 Create Table (建立資料表)。您的資料表會以指定的預設 TTL 值建立。

## Cassandra Query Language (CQL)

使用 CQL 建立具有預設 TTL 值的新資料表

1. 下列陳述式會建立新的資料表，並將預設 TTL 值設為 3,024,000 秒，代表 35 天。

```
CREATE TABLE my_table (
 userid uuid,
 time timeuuid,
 subject text,
 body text,
 user inet,
 PRIMARY KEY (userid, time)
) WITH default_time_to_live = 3024000;
```

2. 若要確認新資料表的 TTL 設定，請使用 `cqlshDESCRIBE` 陳述式，如下列範例所示。輸出會將資料表的預設 TTL 設定顯示為 `default_time_to_live`。

```
DESC TABLE my_table;
```

```
CREATE TABLE my_keyspace.my_table (
 userid uuid,
 time timeuuid,
 body text,
 subject text,
 user inet,
 PRIMARY KEY (userid, time)
) WITH CLUSTERING ORDER BY (time ASC)
 AND bloom_filter_fp_chance = 0.01
 AND caching = {'class': 'com.amazonaws.cassandra.DefaultCaching'}
 AND comment = ''
 AND compaction = {'class': 'com.amazonaws.cassandra.DefaultCompaction'}
 AND compression = {'class': 'com.amazonaws.cassandra.DefaultCompression'}
 AND crc_check_chance = 1.0
 AND dclocal_read_repair_chance = 0.0
 AND default_time_to_live = 3024000
 AND gc_grace_seconds = 7776000
 AND max_index_interval = 2048
 AND memtable_flush_period_in_ms = 3600000
 AND min_index_interval = 128
 AND read_repair_chance = 0.0
 AND speculative_retry = '99PERCENTILE';
```



## CLI

## 使用 建立具有預設 TTL 值的新資料表 AWS CLI

1. 您可以使用下列命令來建立新的資料表，並將預設 TTL 值設為一年。

```
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}]' \
 --default-time-to-live '31536000'
```

2. 若要確認資料表的 TTL 狀態，您可以使用下列命令。

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

命令的輸出看起來與下列範例類似

```
{
 "keyspaceName": "myKeyspace",
 "tableName": "myTable",
 "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
myKeyspace/table/myTable",
 "creationTimestamp": "2024-09-02T10:52:22.190000+00:00",
 "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": "2024-09-02T10:52:22.190000+00:00"
},
"encryptionSpecification": {
 "type": "AWS_OWNED_KMS_KEY"
},
"pointInTimeRecovery": {
 "status": "DISABLED"
},
"ttl": {
 "status": "ENABLED"
},
"defaultTimeToLive": 31536000,
"comment": {
 "message": ""
},
"replicaSpecifications": []
}

```

## 更新資料表的預設存留時間 (TTL) 值

您可以使用新的預設 TTL 值更新現有的資料表。TTL 值以秒為單位設定，最大可設定值為 630,720,000 秒，相當於 20 年。

當您在資料表上啟用 TTL 時，Amazon Keyspaces 會開始儲存每個資料列的其他 TTL 相關中繼資料。此外，TTL 會使用過期時間戳記來追蹤資料列或資料欄何時過期。時間戳記會儲存為資料列中繼資料，並造成資料列的儲存成本。

資料表啟用 TTL 後，您可以使用 CQL DML 陳述式覆寫特定資料列或資料欄的資料表預設 TTL 設定。如需詳細資訊，請參閱[the section called “使用 INSERT 設定新資料列的自訂 TTL”](#)及[the section called “使用 UPDATE 設定資料列和資料欄的自訂 TTL”](#)。

啟用 TTL 功能之後，您就無法停用資料表的 TTL 功能。將資料表 `default_time_to_live` 設為 0 會停用新資料的預設過期時間，但不會停用 TTL 功能，或將資料表還原為原始 Amazon Keyspaces 儲存中繼資料或寫入行為。

請依照下列步驟，使用主控台、CQL 或 更新現有資料表的預設存留時間設定 AWS CLI。

## Console

### 使用主控台更新資料表的預設 TTL 值

1. 登入 AWS 管理主控台，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，然後選擇其他設定索引標籤。
3. 繼續執行存留時間 (TTL)，然後選擇編輯。
4. 針對預設 TTL 期間，輸入過期時間，然後選擇時間單位，例如秒、天或年。Amazon Keyspaces 會以秒為單位存放值。這不會變更現有資料列的 TTL 值。
5. 定義 TTL 設定時，選擇儲存變更。

## Cassandra Query Language (CQL)

### 使用 CQL 更新資料表的預設 TTL 值

1. 您可以使用 ALTER TABLE 編輯資料表的預設存留時間 (TTL) 設定。若要將資料表的預設 TTL 設定更新為 2,592,000 秒，這代表 30 天，您可以使用下列陳述式。

```
ALTER TABLE my_table WITH default_time_to_live = 2592000;
```

2. 若要確認更新資料表的 TTL 設定，請使用 cqlsh DESCRIBE 陳述式，如下列範例所示。輸出會將資料表的預設 TTL 設定顯示為 default\_time\_to\_live。

```
DESC TABLE my_table;
```

陳述式的輸出看起來應該與此範例類似。

```
CREATE TABLE my_keyspace.my_table (
 id int PRIMARY KEY,
 date timestamp,
 name text
) WITH bloom_filter_fp_chance = 0.01
 AND caching = {'class': 'com.amazonaws.cassandra.DefaultCaching'}
 AND comment = ''
 AND compaction = {'class': 'com.amazonaws.cassandra.DefaultCompaction'}
 AND compression = {'class': 'com.amazonaws.cassandra.DefaultCompression'}
```

```
AND crc_check_chance = 1.0
AND dclocal_read_repair_chance = 0.0
AND default_time_to_live = 2592000
AND gc_grace_seconds = 7776000
AND max_index_interval = 2048
AND memtable_flush_period_in_ms = 3600000
AND min_index_interval = 128
AND read_repair_chance = 0.0
AND speculative_retry = '99PERCENTILE';
```

## CLI

### 使用 更新資料表的預設 TTL 值 AWS CLI

1. 您可以使用 `update-table` 編輯資料表的預設 TTL 值。若要將資料表的預設 TTL 設定更新為 2,592,000 秒，這代表 30 天，您可以使用下列陳述式。

```
aws keyspaces update-table --keyspace-name 'myKeyspace' --table-name 'myTable'
--default-time-to-live '2592000'
```

2. 若要確認更新的預設 TTL 值，您可以使用下列陳述式。

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

陳述式的輸出看起來應該如下例所示。

```
{
 "keyspaceName": "myKeyspace",
 "tableName": "myTable",
 "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
myKeyspace/table/myTable",
 "creationTimestamp": "2024-09-02T10:52:22.190000+00:00",
 "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": "2024-09-02T10:52:22.190000+00:00"
 },
 "encryptionSpecification": {
 "type": "AWS_OWNED_KMS_KEY"
 },
 "pointInTimeRecovery": {
 "status": "DISABLED"
 },
 "ttl": {
 "status": "ENABLED"
 },
 "defaultTimeToLive": 2592000,
 "comment": {
 "message": ""
 },
 "replicaSpecifications": []
}

```

## 建立已啟用自訂存留時間 (TTL) 設定的資料表

若要使用可套用至資料列和資料欄的存留時間自訂設定建立新的資料表，而不啟用整個資料表的 TTL 預設設定，您可以使用下列命令。

**Note**

如果在啟用ttl自訂設定的情況下建立資料表，您稍後無法停用設定。

## Cassandra Query Language (CQL)

使用 CQL 建立具有自訂 TTL 設定的新資料表

- ```
CREATE TABLE my_keyspace.my_table (id int primary key) WITH  
CUSTOM_PROPERTIES={'ttl':{'status': 'enabled'}};
```

CLI

使用 建立具有自訂 TTL 設定的新資料表 AWS CLI

- 您可以使用下列命令，在啟用 TTL 的情況下建立新的資料表。

```
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}]' \  
--ttl 'status=ENABLED'
```

- 若要確認資料表已啟用 TTL，您可以使用下列陳述式。

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

陳述式的輸出看起來應該如下例所示。

```
{  
  "keyspaceName": "myKeyspace",  
  "tableName": "myTable",  
  "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/  
myKeyspace/table/myTable",  
  "creationTimestamp": "2024-09-02T10:52:22.190000+00:00",  
  "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": "2024-09-02T11:18:55.796000+00:00"
},
"encryptionSpecification": {
    "type": "AWS_OWNED_KMS_KEY"
},
"pointInTimeRecovery": {
    "status": "DISABLED"
},
"ttl": {
    "status": "ENABLED"
},
"defaultTimeToLive": 0,
"comment": {
    "message": ""
},
"replicaSpecifications": []
}
```

使用自訂存留時間 (TTL) 更新資料表

若要啟用資料表的存留時間自訂設定，以便將 TTL 值套用至個別資料列和資料欄，而無需設定整個資料表的 TTL 預設值，您可以使用下列命令。

Note

ttl 啟用 之後，您就無法為資料表停用它。

Cassandra Query Language (CQL)

使用 CQL 啟用資料表的自訂 TTL 設定

- ```
ALTER TABLE my_table WITH CUSTOM_PROPERTIES={'ttl':{'status': 'enabled'}};
```

### CLI

使用 啟用資料表的自訂 TTL 設定 AWS CLI

- 您可以使用下列命令來更新資料表的自訂 TTL 設定。

```
aws keyspaces update-table --keyspace-name 'myKeyspace' --table-name 'myTable'
--ttl 'status=ENABLED'
```

- 若要確認資料表現在已啟用 TTL，您可以使用下列陳述式。

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

陳述式的輸出看起來應該如下例所示。

```
{
 "keyspaceName": "myKeyspace",
 "tableName": "myTable",
 "resourceArn": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/
myKeyspace/table/myTable",
 "creationTimestamp": "2024-09-02T11:32:27.349000+00:00",
 "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": "2024-09-02T11:32:27.349000+00:00"
},
"encryptionSpecification": {
 "type": "AWS_OWNED_KMS_KEY"
},
"pointInTimeRecovery": {
 "status": "DISABLED"
},
"ttl": {
 "status": "ENABLED"
},
"defaultTimeToLive": 0,
"comment": {
 "message": ""
},
"replicaSpecifications": []
}
```

## 使用 **INSERT** 陳述式來設定新資料列的自訂存留時間 (TTL) 值

### Note

您必須先在資料表上啟用自訂 TTL，才能使用 INSERT 陳述式設定資料列的自訂 TTL 值。如需詳細資訊，請參閱[the section called “更新資料表自訂 TTL”](#)。

若要透過設定個別資料列的過期日期來覆寫資料表的預設 TTL 值，您可以使用 INSERT 陳述式：

- INSERT – 插入設定 TTL 值的新資料列。

使用 INSERT 陳述式設定新列的 TTL 值優先於資料表的預設 TTL 設定。

下列 CQL 陳述式會將資料列插入資料表，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 **UPDATE** 陳述式來編輯資料列和資料欄的自訂存留時間 (TTL) 設定

### Note

您必須先在資料表上啟用 TTL，才能設定資料列和資料欄的自訂 TTL 值。如需詳細資訊，請參閱[the section called “更新資料表自訂 TTL”](#)。

您可以使用 UPDATE 陳述式來覆寫資料表的預設 TTL 值，方法是設定個別資料列和資料欄的過期日期：

- 資料列 – 您可以使用自訂 TTL 值更新現有的資料列。

- 資料欄 – 您可以使用自訂 TTL 值更新現有資料列中的資料欄子集。

設定資料列和資料欄的 TTL 值優先於資料表的預設 TTL 設定。

若要將先前插入的「主體」資料欄的 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;
```

您可以執行簡單的選取查詢，在過期時間之前查看更新的記錄。

```
SELECT * from my_table;
```

查詢會顯示下列輸出。

| userid                               | subject         | user            | time                                 | body  |
|--------------------------------------|-----------------|-----------------|--------------------------------------|-------|
| b79cb3ba-745e-5d9a-8903-4a02327a7e09 | Updated Message | 205.212.123.123 | 96a29100-5e25-11ec-90d7-b5d91eceda0a | Hello |
| 50554d6e-29bb-11e5-b345-feff819cdc9f | Message         | 205.212.123.123 | cf03fb21-59b5-11ec-b371-dff626ab9620 | Hello |

若要確認過期成功，請在設定的過期時間之後再次執行相同的查詢。

```
SELECT * from my_table;
```

查詢會在「主旨」資料欄過期後顯示下列輸出。

| userid                               | subject | user            | time                                 | body  |
|--------------------------------------|---------|-----------------|--------------------------------------|-------|
| b79cb3ba-745e-5d9a-8903-4a02327a7e09 | null    | 205.212.123.123 | 96a29100-5e25-11ec-90d7-b5d91eceda0a | Hello |
| 50554d6e-29bb-11e5-b345-feff819cdc9f | Message | 205.212.123.123 | cf03fb21-59b5-11ec-b371-dff626ab9620 | Hello |

## 搭配 AWS SDK 使用此服務

AWS 軟體開發套件 (SDKs) 適用於許多熱門的程式設計語言。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                                               |
|----------------------------------------------|----------------------------------------------------|
| <a href="#">適用於 C++ 的 AWS SDK</a>            | <a href="#">適用於 C++ 的 AWS SDK 程式碼範例</a>            |
| <a href="#">AWS CLI</a>                      | <a href="#">AWS CLI 程式碼範例</a>                      |
| <a href="#">適用於 Go 的 AWS SDK</a>             | <a href="#">適用於 Go 的 AWS SDK 程式碼範例</a>             |
| <a href="#">適用於 Java 的 AWS SDK</a>           | <a href="#">適用於 Java 的 AWS SDK 程式碼範例</a>           |
| <a href="#">適用於 JavaScript 的 AWS SDK</a>     | <a href="#">適用於 JavaScript 的 AWS SDK 程式碼範例</a>     |
| <a href="#">適用於 Kotlin 的 AWS SDK</a>         | <a href="#">適用於 Kotlin 的 AWS SDK 程式碼範例</a>         |
| <a href="#">適用於 .NET 的 AWS SDK</a>           | <a href="#">適用於 .NET 的 AWS SDK 程式碼範例</a>           |
| <a href="#">適用於 PHP 的 AWS SDK</a>            | <a href="#">適用於 PHP 的 AWS SDK 程式碼範例</a>            |
| <a href="#">AWS Tools for PowerShell</a>     | <a href="#">AWS Tools for PowerShell 程式碼範例</a>     |
| <a href="#">適用於 Python (Boto3) 的 AWS SDK</a> | <a href="#">適用於 Python (Boto3) 的 AWS SDK 程式碼範例</a> |
| <a href="#">適用於 Ruby 的 AWS SDK</a>           | <a href="#">適用於 Ruby 的 AWS SDK 程式碼範例</a>           |
| <a href="#">適用於 Rust 的 AWS SDK</a>           | <a href="#">適用於 Rust 的 AWS SDK 程式碼範例</a>           |
| <a href="#">適用於 SAP ABAP 的 AWS SDK</a>       | <a href="#">適用於 SAP ABAP 的 AWS SDK 程式碼範例</a>       |
| <a href="#">適用於 Swift 的 AWS SDK</a>          | <a href="#">適用於 Swift 的 AWS SDK 程式碼範例</a>          |

### 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 Amazon Keyspaces 資源的標籤和標籤

您可以使用標籤來標記 Amazon Keyspaces (適用於 Apache Cassandra) 資源。標籤可讓您以不同的方式分類資源，例如，依用途、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。
- 請參閱依標籤細分的 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 “以 Amazon Keyspaces 標籤為基礎的授權”](#)。

Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Simple Storage Service (Amazon S3)、Amazon Keyspaces 等 AWS 服務支援標記。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 [使用標籤在 Amazon Keyspaces 中標記資源的限制](#)。
2. 使用 [在 Amazon Keyspaces 中標記金鑰空間、資料表和串流](#) 建立標籤。
3. 使用 [使用 Amazon Keyspaces 的標籤建立成本分配報告](#) 追蹤每個作用中標籤 AWS 的成本。

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](#)。

## 使用標籤在 Amazon Keyspaces 中標記資源的限制

每個標記皆包含由您定義的索引鍵和值。將適用以下限制：

- 每個 Amazon Keyspaces 金鑰空間、資料表或串流只能有一個具有相同金鑰的標籤。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資料表。若要將相同的標籤套用至金鑰空間及其所有資料表，每個資源都必須個別加上標籤。
- 套用至資料表的標籤不會自動套用至該資料表的串流。若要在建立資料表期間將相同的標籤套用至資料表和串流，您可以在建立資料表時使用 `PropagateTagsOnEnable` 旗標。使用此旗標，Amazon Keyspaces 會在串流建立期間將資料表的標籤套用至串流。當串流處於作用中狀態時，資料表標籤的變更不會套用至串流。

- 當您建立多區域金鑰空間或資料表時，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資料表。當您使用 ALTER KEYSPACE 或 變更現有標籤時 ALTER TABLE，更新只會套用至您進行變更之區域中的 金鑰空間或資料表。
- 值就像標籤類別 (索引鍵) 內的描述項。在 Amazon Keyspaces 中，值不能為空或 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 指派的標籤名稱和值會自動指派前 aws: 綴，您無法指派。AWS 指派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。

## 在 Amazon Keyspaces 中標記金鑰空間、資料表和串流

您可以使用 Amazon Keyspaces 主控台、或 Cassandra 查詢語言 (CQL)，新增、列出 AWS CLI、編輯或刪除金鑰空間、資料表和串流的標籤。然後，您可以啟用這些使用者定義的標籤，讓它們出現在 AWS 帳單與成本管理 主控台上以進行成本分配追蹤。如需詳細資訊，請參閱[使用 Amazon Keyspaces 的標籤建立成本分配報告](#)。

對於大量編輯，您也可以在主控台上使用標籤編輯器。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](#)。

如需標籤結構的相關資訊，請參閱[使用標籤在 Amazon Keyspaces 中標記資源的限制](#)。

### 主題

- [在建立新的金鑰空間時新增標籤](#)
- [將標籤新增至金鑰空間](#)
- [從金鑰空間刪除標籤](#)
- [檢視金鑰空間的標籤](#)
- [建立新資料表時新增標籤](#)
- [將標籤新增至資料表](#)
- [從資料表刪除標籤](#)
- [檢視資料表的標籤](#)

- [建立資料表時，將標籤新增至新串流](#)
- [將標籤新增至現有資料表的新串流](#)
- [將新標籤新增至串流](#)
- [從串流刪除標籤](#)
- [檢視串流的標籤](#)

## 在建立新的金鑰空間時新增標籤

您可以在建立新的金鑰空間時 AWS CLI，使用 Amazon Keyspaces 主控台、CQL 或 新增標籤。

### Console

使用主控台建立新金鑰空間時設定標籤

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 在導覽窗格中，選擇金鑰空間，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。
4. 在標籤下選擇新增標籤，然後輸入索引鍵和值。
5. 選擇建立金鑰空間。

### Cassandra Query Language (CQL)

使用 CQL 建立新金鑰空間時設定標籤

- 下列範例會使用標籤建立新的金鑰空間。

```
CREATE KEYSPACE mykeyspace WITH TAGS = {'key1':'val1', 'key2':'val2'};
```

### CLI

使用 建立新金鑰空間時設定標籤 AWS CLI

- 下列陳述式會使用標籤建立新的金鑰空間。

```
aws keyspaces create-keyspace --keyspace-name 'myKeyspace' --tags
'key=key1,value=val1' 'key=key2,value=val2'
```

## 將標籤新增至金鑰空間

下列範例示範如何將標籤新增至 Amazon Keyspaces 中的金鑰空間。

### Console

使用主控台將標籤新增至現有的金鑰空間

1. 登入 AWS 管理主控台，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 金鑰空間。
3. 從清單中選擇金鑰空間。然後選擇標籤索引標籤，您可以在其中檢視金鑰空間的標籤。
4. 選擇管理標籤以新增、編輯或刪除標籤。
5. 選擇儲存變更。

### Cassandra Query Language (CQL)

使用 CQL 將標籤新增至現有的金鑰空間

- ```
ALTER KEYSPACE mykeyspace ADD TAGS {'key1':'val1', 'key2':'val2'};
```

CLI

使用 將標籤新增至現有的金鑰空間 AWS CLI

- 下列範例示範如何將新標籤新增至現有的 金鑰空間。

```
aws keyspaces tag-resource --resource-arn 'arn:aws:cassandra:us-  
east-1:111122223333:/keyspace/myKeyspace/' --tags 'key=key3,value=val3'  
'key=key4,value=val4'
```


從金鑰空間刪除標籤

Console

使用主控台從現有金鑰空間刪除標籤

1. 登入 AWS 管理主控台，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 金鑰空間。
3. 從清單中選擇金鑰空間。然後選擇標籤索引標籤，您可以在其中檢視金鑰空間的標籤。
4. 選擇管理標籤並刪除不再需要的標籤。
5. 選擇儲存變更。

Cassandra Query Language (CQL)

使用 CQL 從現有的金鑰空間刪除標籤

- ```
ALTER KEYSPACE mykeyspace DROP TAGS {'key1':'val1', 'key2':'val2'};
```

### CLI

使用 從現有金鑰空間刪除標籤 AWS CLI

- 下列陳述式會從金鑰空間移除指定的標籤。

```
aws keyspaces untag-resource --resource-arn 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/myKeyspace/' --tags 'key=key3,value=val3' 'key=key4,value=val4'
```

## 檢視金鑰空間的標籤

下列範例示範如何使用 主控台、CQL 或 讀取標籤 AWS CLI。

## Console

使用 Amazon Keyspaces 主控台檢視金鑰空間的標籤

1. 登入 AWS 管理主控台，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 金鑰空間。
3. 從清單中選擇金鑰空間。然後選擇標籤索引標籤，您可以在其中檢視金鑰空間的標籤。

## Cassandra Query Language (CQL)

使用 CQL 檢視金鑰空間的標籤

若要讀取連接至金鑰空間的標籤，請使用下列 CQL 陳述式。

```
SELECT * FROM system_schema_mcs.tags WHERE valid_where_clause;
```

子WHERE句是必要項目，且必須使用下列其中一種格式：

- `keyspace_name = 'mykeyspace' AND resource_type = 'keyspace'`
- `resource_id = arn`
- 下列陳述式顯示金鑰空間是否具有標籤。

```
SELECT * FROM system_schema_mcs.tags WHERE keyspace_name = 'mykeyspace' AND
resource_type = 'keyspace';
```

查詢的輸出如下所示。

```
resource_id | keyspace_name
| resource_name | resource_type | tags

+-----+-----+-----+-----
arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/ | mykeyspace
| mykeyspace | keyspace | {'key1': 'val1', 'key2': 'val2'}
```

## CLI

### 使用 檢視金鑰空間的標籤 AWS CLI

- 此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-1:111122223333:/keyspace/myKeyspace/'
```

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"
 }
]
}
```

### 建立新資料表時新增標籤

您可以使用 Amazon Keyspaces 主控台、CQL 或 AWS CLI，在建立新資料表時新增標籤。

## Console

使用 建立新資料表時新增標籤 (主控台)

1. 登入 AWS 管理主控台，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. 在資料表詳細資訊區段的建立資料表頁面上，選取金鑰空間，並提供資料表的名稱。
4. 在結構描述區段中，建立資料表的結構描述。
5. 在資料表設定區段中，選擇自訂設定。
6. 繼續前往資料表標籤 – 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。

## Cassandra Query Language (CQL)

使用 CQL 建立新資料表時新增標籤

- 下列範例會使用標籤建立新的資料表。

```
CREATE TABLE mytable(...) WITH TAGS = {'key1':'val1', 'key2':'val2'};
```

## CLI

使用 建立新資料表時新增標籤 AWS CLI

- 下列範例示範如何建立具有標籤的新資料表。命令會在現有的金鑰空間 myKeyspace 中建立資料表 myTable。myKeyspace 請注意，命令已分成不同的行，以協助提高可讀性。

```
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'
```

## 將標籤新增至資料表

您可以使用主控台、CQL 或 將標籤新增至 Amazon Keyspaces 中的現有資料表 AWS CLI。

## Console

使用 Amazon Keyspaces 主控台將標籤新增至資料表

1. 登入 AWS 管理主控台，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)。
3. 從清單中選擇資料表，然後選擇標籤索引標籤。
4. 選擇管理標籤，將標籤新增至資料表。
5. 選擇儲存變更。

## Cassandra Query Language (CQL)

使用 CQL 將標籤新增至資料表

- 下列陳述式說明如何將標籤新增至現有資料表。

```
ALTER TABLE mykeyspace.mytable ADD TAGS {'key1':'val1', 'key2':'val2'};
```

## CLI

使用 將標籤新增至資料表 AWS CLI

- 下列範例顯示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-1:11112223333:/keyspace/myKeyspace/table/myTable' --tags 'key=key3,value=val3' 'key=key4,value=val4'
```

## 從資料表刪除標籤

### Console

使用 Amazon Keyspaces 主控台從資料表刪除標籤

1. 登入 AWS 管理主控台，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)。
3. 從清單中選擇資料表，然後選擇標籤索引標籤。
4. 選擇管理標籤以從資料表中刪除標籤。
5. 選擇儲存變更。

## Cassandra Query Language (CQL)

使用 CQL 從資料表刪除標籤

- 下列陳述式說明如何從現有資料表中刪除標籤。

```
ALTER TABLE mytable DROP TAGS {'key3':'val3', 'key4':'val4'};
```

## CLI

使用 將標籤新增至資料表 AWS CLI

- 下列陳述式會從金鑰空間移除指定的標籤。

```
aws keyspaces untag-resource --resource-arn 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/myKeyspace/table/myTable' --tags 'key=key3,value=val3' 'key=key4,value=val4'
```

## 檢視資料表的標籤

下列範例示範如何使用主控台、CQL 或 在 Amazon Keyspaces 中檢視資料表的標籤 AWS CLI。

### Console

使用主控台檢視資料表的標籤

1. 登入 AWS 管理主控台，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)。
3. 從清單中選擇資料表，然後選擇標籤索引標籤。

## Cassandra Query Language (CQL)

使用 CQL 檢視資料表的標籤

若要讀取連接至資料表的標籤，請使用下列 CQL 陳述式。

```
SELECT * FROM system_schema_mcs.tags WHERE valid_where_clause;
```

子WHERE句是必要項目，且必須使用下列其中一種格式：

- `keyspace_name = 'mykeyspace' AND resource_name = 'mytable'`
- `resource_id = arn`
- 下列查詢會傳回指定資料表的標籤。

```
SELECT * FROM system_schema_mcs.tags WHERE keyspace_name = 'mykeyspace' AND
resource_name = 'mytable';
```

該查詢的輸出如下所示。

```
resource_id |
keyspace_name | resource_name | resource_type | tags

+-----+-----+-----+-----
arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/mytable |
mykeyspace | mytable | table | {'key1': 'val1', 'key2':
'val2'}
```

## CLI

使用 檢視資料表的標籤 AWS CLI

- 此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:arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/
```

```
table/my_table/stream/2025-05-11T21:21:33.291cassandra:us-east-1:111122223333:/
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"
 }
]
}
```

建立資料表時，將標籤新增至新串流

當您使用 CQL 或 建立具有串流的新資料表 AWS CLI 來標記串流時，您可以將標籤新增至串流。

#### Note

Amazon Keyspaces CDC 需要存在服務連結角色 (AWSServiceRoleForAmazonKeyspacesCDC)，以代表您將指標資料從 Amazon Keyspaces CDC 串流發佈到 CloudWatch 帳戶中 "cloudwatch:namespace": "AWS/Cassandra" 的。系統會自動建立此角色。如需詳細資訊，請參閱 [the section called "CDC 串流"](#)。



## Cassandra Query Language (CQL)

使用 CQL 建立新資料表時，將標籤新增至串流

1. 若要使用串流建立新的資料表，並將資料表標籤自動套用至串流，您可以使用 'propagate\_tags': 'TABLE' 旗標。下列陳述式是此範例。

```
CREATE TABLE mytable (pk int, ck text, PRIMARY KEY(pk))
WITH TAGS={'key1':'val1', 'key2':'val2'}
AND cdc = TRUE
AND CUSTOM_PROPERTIES={
 'cdc_specification': {
 'view_type': 'NEW_IMAGE',
 'propagate_tags': 'TABLE'
 }
};
```

2. 若要將新標籤套用至串流，您可以使用下列範例。

```
CREATE TABLE mytable (pk int, ck text, PRIMARY KEY(pk))
WITH TAGS={'key1':'val1', 'key2':'val2'}
AND cdc = TRUE
AND CUSTOM_PROPERTIES={
 'cdc_specification': {
 'view_type': 'NEW_IMAGE',
 'tags': { 'key': 'string', 'value': 'string' },
 }
};
```

## CLI

使用 建立新資料表時，將標籤新增至串流 AWS CLI

1. 若要建立具有串流的資料表，並將資料表標籤自動套用至串流，您可以使用 propagateTags=Table 旗標。下列程式碼為範例。

```
aws keyspaces create-table \
--keyspace-name 'my_keyspace' \
--table-name 'my_table' \
```

```
--schema-definition 'allColumns=[{name=pk,type=int},
{name=ck,type=text}],clusteringKeys=[{name=ck,orderBy=ASC}],partitionKeys=[{name=pk}]' \
--tags key=tag_key,value=tag_value
--cdc-specification propagateTags=TABLE,status=ENABLED,viewType=NEW_IMAGE
```

- 若要將不同的標籤套用至串流，您可以使用下列範例。

```
aws keyspaces create-table \
--keyspace-name 'my_keyspace' \
--table-name 'my_table' \
--schema-definition 'allColumns=[{name=pk,type=int},
{name=ck,type=text}],clusteringKeys=[{name=ck,orderBy=ASC}],partitionKeys=[{name=pk}]' \
--tags key=tag_key,value=tag_value
--cdc-specification 'status=ENABLED,viewType=NEW_IMAGE,tags=[{key=tag_key,
value=tag_value}]'
```

## 將標籤新增至現有資料表的新串流

您可以在為現有資料表建立新串流時新增標籤。您可以使用 `PropagateTags` 旗標將資料表標籤套用至串流，或指定串流的新標籤。您可以使用 CQL 或 AWS CLI 來標記新的串流。

### Note

Amazon Keyspaces CDC 需要存在服務連結角色 (AWS `ServiceRoleForAmazonKeyspacesCDC`)，以代表您將指標資料從 Amazon Keyspaces CDC 串流發佈到 CloudWatch 帳戶中 "cloudwatch:namespace": "AWS/Cassandra" 的。系統會自動建立此角色。如需詳細資訊，請參閱 [the section called "CDC 串流"](#)。

## Console

使用 建立新串流時新增標籤 (主控台)

- 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
- 在導覽窗格中，選擇資料表，然後選擇您要為其新增串流的資料表。

3. 選擇串流索引標籤。
4. 在串流詳細資訊區段中，選擇編輯。
5. 選取開啟串流。
6. 選取檢視類型並繼續前往標籤以建立串流的標籤。
7. 您可以選擇以下其中一個選項：
  - 無標籤 – 如果您不想為串流建立任何標籤，請使用此選項。
  - 從資料表複製標籤 – 如果您想要將標籤從資料表複製到串流，請使用此選項。複製標籤之後，您可以編輯串流的標籤。請注意，只有在資料表具有標籤時，才能使用此選項。
  - 新增標籤 – 您可以選擇新增標籤，為串流新增最多 50 個標籤。
8. 選擇儲存變更。

## Cassandra Query Language (CQL)

### 建立新串流時新增標籤

1. 若要為現有資料表建立新的串流，並將資料表的標籤套用至串流，您可以使用 'propagate\_tags': 'TABLE' 旗標。下列陳述式為範例。

```
ALTER TABLE mytable WITH cdc = TRUE AND CUSTOM_PROPERTIES={ 'cdc_specification':
 { 'view_type': 'NEW_IMAGE', 'propagate_tags': 'TABLE' } };
```

2. 若要為現有資料表建立新的串流並指定新標籤，您可以使用下列範例。

```
ALTER TABLE mytable WITH cdc = TRUE AND CUSTOM_PROPERTIES={ 'cdc_specification':
 { 'view_type': 'NEW_IMAGE', 'tags': { 'key': 'string', 'value': 'string' } } };
```

## CLI

### 使用 建立新串流時新增標籤 AWS CLI

1. 若要使用標籤建立新的串流，您可以使用 propagateTags=TABLE 旗標將資料表的標籤自動套用至串流。下列程式碼為範例。

```
aws keyspaces update-table \
--keyspace-name 'my_keyspace' \
--table-name 'my_table' \
--propagate-tags TABLE
```

```
--cdc-specification propagateTags=TABLE,status=ENABLED,viewType=NEW_IMAGE
```

- 若要為現有資料表建立新的串流並指定新標籤，您可以使用下列範例。

```
aws keyspaces update-table \
--keyspace-name 'my_keyspace' \
--table-name 'my_table' \
--cdc-specification 'status=ENABLED,viewType=NEW_IMAGE,tags=[{key=tag_key,
value=tag_value}]'
```

## 將新標籤新增至串流

您可以使用 CQL 或 將新標籤新增至 Amazon Keyspaces 中的現有串流 AWS CLI。您只能將標籤新增至最新的串流。

### Console

將標籤新增至現有串流（主控台）

- 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
- 在導覽窗格中，選擇資料表，然後選擇具有您要標記之串流的資料表。
- 選擇串流索引標籤。
- 在標籤區段中，選擇管理標籤。
- 選擇新增標籤以新增標籤。您可以重複此步驟，建立最多 50 個標籤。
- 選擇儲存變更。

### Cassandra Query Language (CQL)

使用 CQL 將標籤新增至串流

- 下列陳述式說明如何將標籤新增至現有的串流。

```
ALTER TABLE mykeyspace.mytable ADD TAGS_FOR_CDC {'key1':'val1', 'key2':'val2'};
```

## CLI

使用 將標籤新增至串流 AWS CLI

- 下列範例顯示如何將新標籤新增至現有串流。

```
aws keyspaces tag-resource --resource-arn 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/table/my_table/stream/2025-05-11T21:21:33.291' --tags 'key=key3,value=val3' 'key=key4,value=val4'
```

## 從串流刪除標籤

若要從串流刪除標籤，您可以使用 CQL 或 AWS CLI。您只能刪除最新串流的標籤。

### Console

使用 Amazon Keyspaces 主控台從資料表刪除標籤

1. 登入 AWS 管理主控台，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)。
3. 從清單中選擇資料表，然後選擇串流索引標籤。
4. 在標籤區段中，選擇管理標籤以從資料表中刪除標籤。
5. 在您要刪除的標籤之後，選擇移除。
6. 選擇儲存變更。

## Cassandra Query Language (CQL)

使用 CQL 從串流刪除標籤

- 下列陳述式示範如何從現有串流刪除標籤。

```
ALTER TABLE mytable DROP TAGS_FOR_CDC {'key3':'val3', 'key4':'val4'};
```

## CLI

### 使用 從串流刪除標籤 AWS CLI

- 下列陳述式會從串流中移除指定的標籤。

```
aws keyspaces untag-resource --resource-arn 'arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/table/my_table/stream/2025-05-11T21:21:33.291' --tags 'key=key3,value=val3' 'key=key4,value=val4'
```

## 檢視串流的標籤

下列範例示範如何使用 CQL 或在 Amazon Keyspaces 中檢視串流的標籤 AWS CLI。

### Console

#### 使用主控台檢視串流的標籤

- 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
- 在導覽窗格中，選擇 Tables (資料表)。
- 從清單中選擇資料表，然後選擇串流索引標籤。
- 您可以在標籤區段中檢視串流的標籤。

### Cassandra Query Language (CQL)

#### 使用 CQL 檢視串流的標籤

若要讀取連接至串流的標籤，您必須在 WHERE 子句中指定串流的資源 ARN。下列 CQL 語法為範例。

```
SELECT * FROM system_schema_mcs.tags WHERE resource_id = stream_arn;
```

- 下列查詢會傳回指定串流的標籤。

```
SELECT tags FROM system_schema_mcs.tags WHERE resource_id =
'arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/table/my_table/
stream/2025-05-06T17:17:39.800';
```

該查詢的輸出如下所示。

```
resource_id
resource_type | tags
-----+-----+-----
+-----+-----+-----
arn:aws:cassandra:us-east-1:111122223333:/keyspace/my_keyspace/table/my_table/
stream/2025-04-02T23:00:07.052 | singleks | 2025-04-02T23:00:07.052 |
stream | {'tagkey': 'tagval'}
```

## CLI

### 使用 檢視串流的標籤 AWS CLI

- 此範例說明如何列出指定金鑰空間下所有串流的標籤。

```
aws keyspaces list-tags-for-resource --resource-arn 'arn:aws:cassandra:us-
east-1:111122223333:/keyspace/my_keyspace/table/my_table/
stream/2025-05-11T21:21:33.291'
```

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 Amazon Keyspaces 的標籤建立成本分配報告

AWS 使用標籤來整理成本分配報告中的資源成本。AWS 提供兩種類型的成本分配標籤：

- AWS產生的 tag。AWS defines、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。

若要啟用 AWS產生的標籤：

1. 登入 AWS 管理主控台 並開啟 Billing and Cost Management 主控台，網址為 <https://console.aws.amazon.com/billing/home#/>。
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 管理主控台 並開啟 Billing and Cost Management 主控台，網址為 <https://console.aws.amazon.com/billing/home#/>。
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)。

在您建立和啟用標籤之後，AWS 會產生成本分配報告，其中包含依作用中標籤分組的用量和成本。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依成本分配報告上的標籤細分。

如需詳細資訊，請參閱[使用成本分配標籤](#)。

## 使用 建立 Amazon Keyspaces 資源 AWS CloudFormation

Amazon Keyspaces 已與 整合 AWS CloudFormation，這項服務可協助您建立和設定 AWS 金鑰空間和資料表，以減少建立和管理資源和基礎設施的時間。您可以建立範本來描述所需的金鑰空間和資料表，並 CloudFormation 負責為您佈建和設定這些資源。

使用 時 CloudFormation，您可以重複使用範本來一致且重複地設定 Amazon Keyspaces 資源。只需描述您的資源一次，然後在多個 AWS 帳戶 和 區域中逐一佈建相同的資源。

### Amazon Keyspaces 和 CloudFormation 範本

若要佈建和設定 Amazon Keyspaces 的資源，您必須了解 [CloudFormation 範本](#)。範本是以 JSON 或 YAML 格式化的文本檔案。這些範本說明您要在 CloudFormation 堆疊中佈建的資源。如果您不熟悉 JSON 或 YAML，您可以使用 CloudFormation 設計工具來協助您開始使用 CloudFormation 範本。如需詳細資訊，請參閱AWS CloudFormation 《使用者指南》中的[什麼是 CloudFormation 設計工具？](#)。

Amazon Keyspaces 支援在其中建立金鑰空間和資料表 CloudFormation。對於您使用 CloudFormation 範本建立的資料表，您可以指定結構描述、讀取/寫入模式、佈建的輸送量設定和其他支援的功能。如需詳細資訊，包括金鑰空間和資料表的 JSON 和 YAML 範本範例，請參閱 範本[參考中的 Amazon Keyspaces \(適用於 Apache Cassandra\) 資源類型](#)AWS CloudFormation 參考。

### 進一步了解 CloudFormation

若要進一步了解 CloudFormation，請參閱下列資源：

- [AWS CloudFormation](#)
- [AWS CloudFormation 使用者指南](#)
- [AWS CloudFormation 命令列界面使用者指南](#)

# 搭配 Amazon Keyspaces 使用 NoSQL Workbench (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 用戶端可用於 Windows、macOS 和 Linux。

## 設計資料模型並自動建立資源

NoSQL Workbench point-and-click 式界面，以設計和建立 Amazon Keyspaces 資料模型。您可以透過定義鍵空間、資料表和資料欄，從頭開始輕鬆建立新的資料模型。您也可以匯入現有的資料模型並進行修改（例如新增、編輯或移除資料欄），以調整新應用程式的資料模型。然後 NoSQL Workbench 可讓您將資料模型遞交至 Amazon Keyspaces 或 Apache Cassandra，並自動建立金鑰空間和資料表。若要了解如何建置資料模型，請參閱 [the section called “建立資料模型”](#) 和 [the section called “編輯資料模型”](#)。

## 視覺化資料模型

使用 NoSQL Workbench，您可以視覺化資料模型，以協助確保資料模型可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，以進行協同合作、文件和簡報。如需詳細資訊，請參閱 [the section called “視覺化資料模型”](#)。

## 主題

- [下載 NoSQL Workbench](#)
- [NoSQL Workbench 入門](#)
- [使用 NoSQL Workbench 視覺化資料模型](#)
- [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](#)
- [如何將資料模型遞交至 Amazon Keyspaces 和 Apache Cassandra](#)
- [NoSQL Workbench 中的範例資料模型](#)
- [NoSQL Workbench 的版本歷史記錄](#)

## 下載 NoSQL Workbench

請依照這些指示下載並安裝 NoSQL Workbench。

## 下載並安裝 NoSQL Workbench

1. 使用下列連結之一，免費下載 NoSQL Workbench。

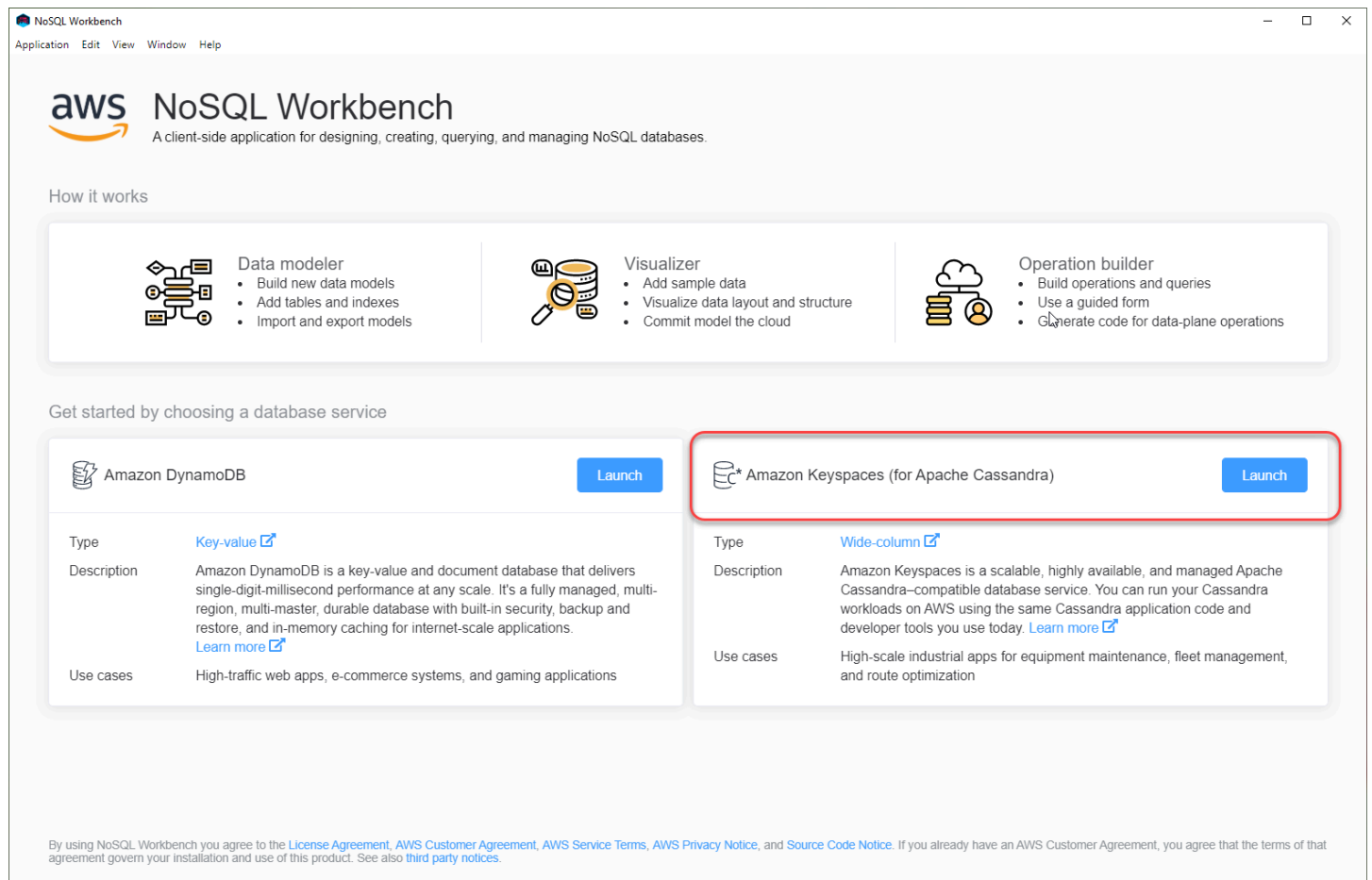
| 作業系統    | 下載連結                            |
|---------|---------------------------------|
| macOS   | <a href="#">適用於 macOS 的下載</a>   |
| Linux*  | <a href="#">下載 Linux 版本</a>     |
| Windows | <a href="#">適用於 Windows 的下載</a> |

\* 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，啟動應用程式並依照畫面上的指示完成安裝。

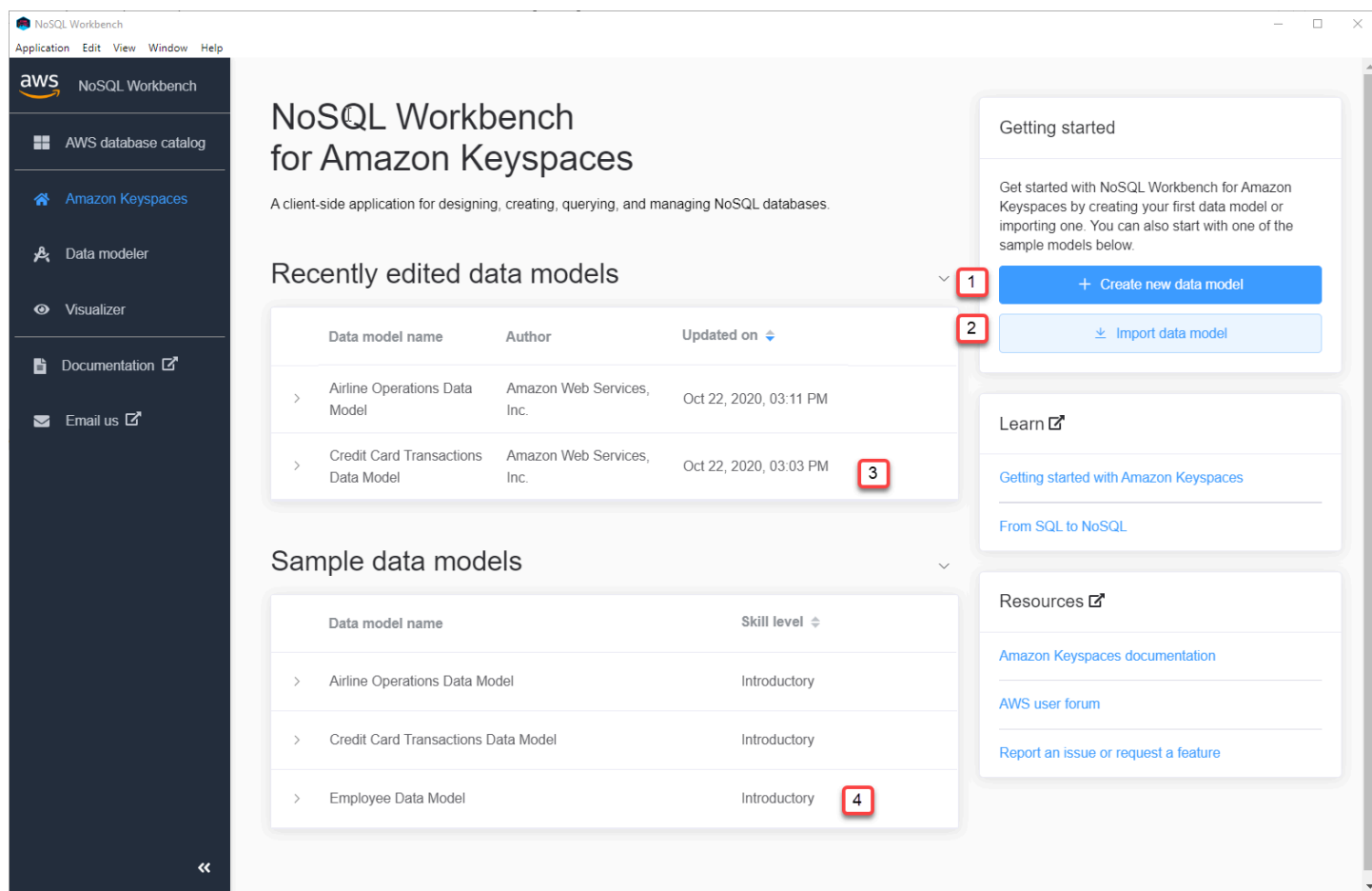
## NoSQL Workbench 入門

若要開始使用 NoSQL Workbench，請在 NoSQL Workbench 的資料庫目錄頁面上，選擇 Amazon Keyspaces，然後選擇啟動。



這會開啟 Amazon Keyspaces 的 NoSQL Workbench 首頁，您可以在其中使用下列選項開始：

1. 建立新的資料模型。
2. 匯入 JSON 格式的現有資料模型。
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 Workbench 資料模型器。若要繼續建立新的資料模型，請參閱 [the section called “建立資料模型”](#)。若要編輯現有的資料模型，請參閱 [the section called “編輯資料模型”](#)。

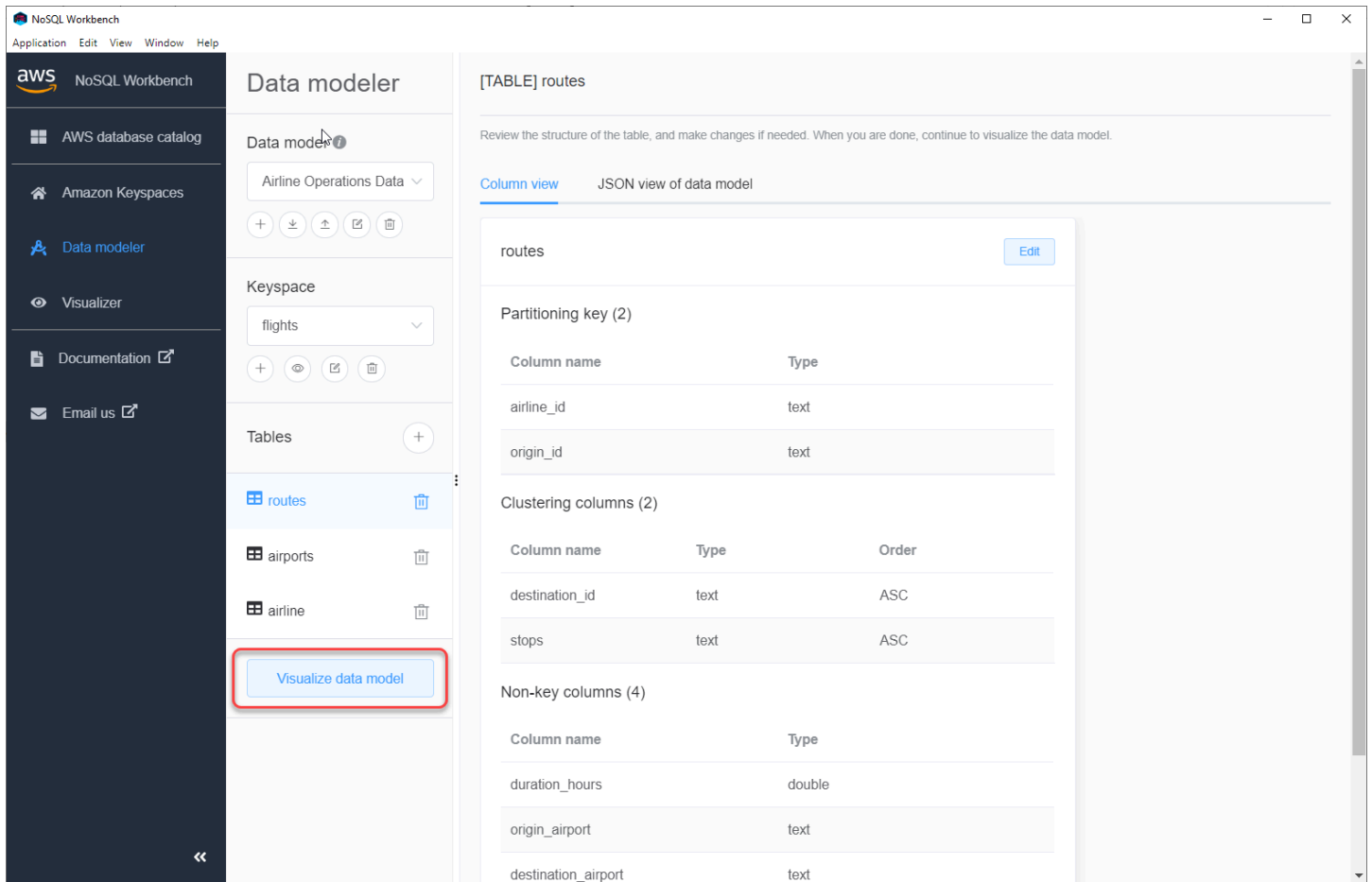
## 使用 NoSQL Workbench 視覺化資料模型

使用 NoSQL Workbench，您可以視覺化資料模型，以協助確保資料模型可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，以進行協同合作、文件和簡報。

建立新資料模型或編輯現有資料模型之後，您可以視覺化模型。

## 使用 NoSQL Workbench 視覺化資料模型

當您在資料模型器中完成資料模型時，請選擇視覺化資料模型。



這會帶您前往 NoSQL Workbench 中的資料視覺化工具。資料視覺化工具提供資料表結構描述的視覺化呈現，並可讓您新增範例資料。若要將範例資料新增至資料表，請從模型中選擇資料表，然後選擇編輯。若要新增資料列，請選擇畫面底部的新增資料列。完成後，請選擇 Save (儲存)。

The screenshot shows the AWS NoSQL Workbench Visualizer interface. The left sidebar contains navigation options: AWS database catalog, Amazon Keyspaces, Data modeler, Visualizer (selected), Documentation, and Email us. The main area is titled 'Visualizer' and shows the 'Data model' for 'Airline Operations'. The 'Keyspace' is set to 'flights'. The 'Tables' section lists 'routes', 'airports', and 'airline'. The 'routes' table is selected, and its schema is displayed. The schema includes a 'Primary key' (airline\_id) and 'Non-key columns' (duration\_hours, origin\_airport). The 'Clustering columns' are destination\_id and stops. The data grid shows three rows of sample data. A red box highlights the '+ Add new row' button at the bottom.

| Primary key       |                       |              | Non-key columns         |                       |
|-------------------|-----------------------|--------------|-------------------------|-----------------------|
| Partition key     | Clustering columns    |              | duration_hours (double) | origin_airport (text) |
| airline_id (text) | destination_id (text) | stops (text) |                         |                       |
| acme_airlines     | MCI                   | 1            | 0.33333333              | Newark Liberty        |
| trusted_airlines  | MIC                   | 2            | 0.33333333              | Phoenix Sky Ha        |
| freedom_airline   | EWB                   | 1            | 0.33333333              | San Francisco         |

## 彙總檢視

確認資料表的結構描述後，您可以彙總資料模型視覺化效果。

The screenshot shows the AWS NoSQL Workbench Visualizer interface. On the left sidebar, the 'Visualizer' tab is selected. The main area displays a data model for a table named 'routes'. The model is structured as follows:

| Primary key       |                  |                       |              | Non-key columns                  |                              |           |
|-------------------|------------------|-----------------------|--------------|----------------------------------|------------------------------|-----------|
| Partition key     |                  | Clustering columns    |              |                                  |                              |           |
| airline_id (text) | origin_id (text) | destination_id (text) | stops (text) | origin_airport (text)            | destination_airport (text)   | equipment |
| acme_airlines     | EWR              | MCI                   | 1            | Newark Liberty International     | Kansas City International    | 737       |
| trusted_airlines  | PHX              | MIC                   | 2            | Phoenix Sky Harbor International | Kansas City International    | 737       |
| freedom_airlines  | SFO              | EWR                   | 1            | San Francisco International      | Newark Liberty International | 747       |

In the left sidebar, the 'Aggregate view' button is highlighted with a red box. Below it are buttons for 'Commit to Amazon Keyspaces' and 'Commit to Apache Cassandra'.

彙總資料模型的檢視之後，您可以將檢視匯出至 PNG 檔案。若要將資料模型匯出至 JSON 檔案，請選擇資料模型名稱下的上傳簽署。

#### Note

您可以在設計過程中隨時以 JSON 格式匯出資料模型。



Visualizer

Data model ⓘ

Airline Operations Data

Keyspaces

flights

Tables

- routes
- airports
- airline

Aggregate view

Commit to Amazon Keyspaces

Commit to Apache Cassandra

Aggregate view

[TABLE] routes

| Primary key       |                  |                       |              | Non-key columns                  |                              |                  |
|-------------------|------------------|-----------------------|--------------|----------------------------------|------------------------------|------------------|
| Partition key     |                  | Clustering columns    |              |                                  |                              |                  |
| airline_id (text) | origin_id (text) | destination_id (text) | stops (text) | origin_airport (text)            | destination_airport (text)   | equipment (text) |
| acme_airlines     | EWR              | MCI                   | 1            | Newark Liberty International     | Kansas City International    | 737              |
| trusted_airlines  | PHX              | MIC                   | 2            | Phoenix Sky Harbor International | Kansas City International    | 737              |
| freedom_airlines  | SFO              | EWR                   | 1            | San Francisco International      | Newark Liberty International | 747              |

[TABLE] airports

| Primary key       |  | Non-key columns |             |                  |                          |
|-------------------|--|-----------------|-------------|------------------|--------------------------|
| Partition key     |  |                 |             |                  |                          |
| airport_id (text) |  |                 |             |                  |                          |
| EWR               |  | name (text)     | city (text) | iata_code (text) | details (map<text,text>) |

您有下列選項可遞交變更：

- 向 Amazon Keyspaces 遞交
- 遞交至 Apache Cassandra 叢集

若要進一步了解如何遞交變更，請參閱 [the section called “遞交資料模型”](#)。

## 使用 NoSQL Workbench 建立新的資料模型

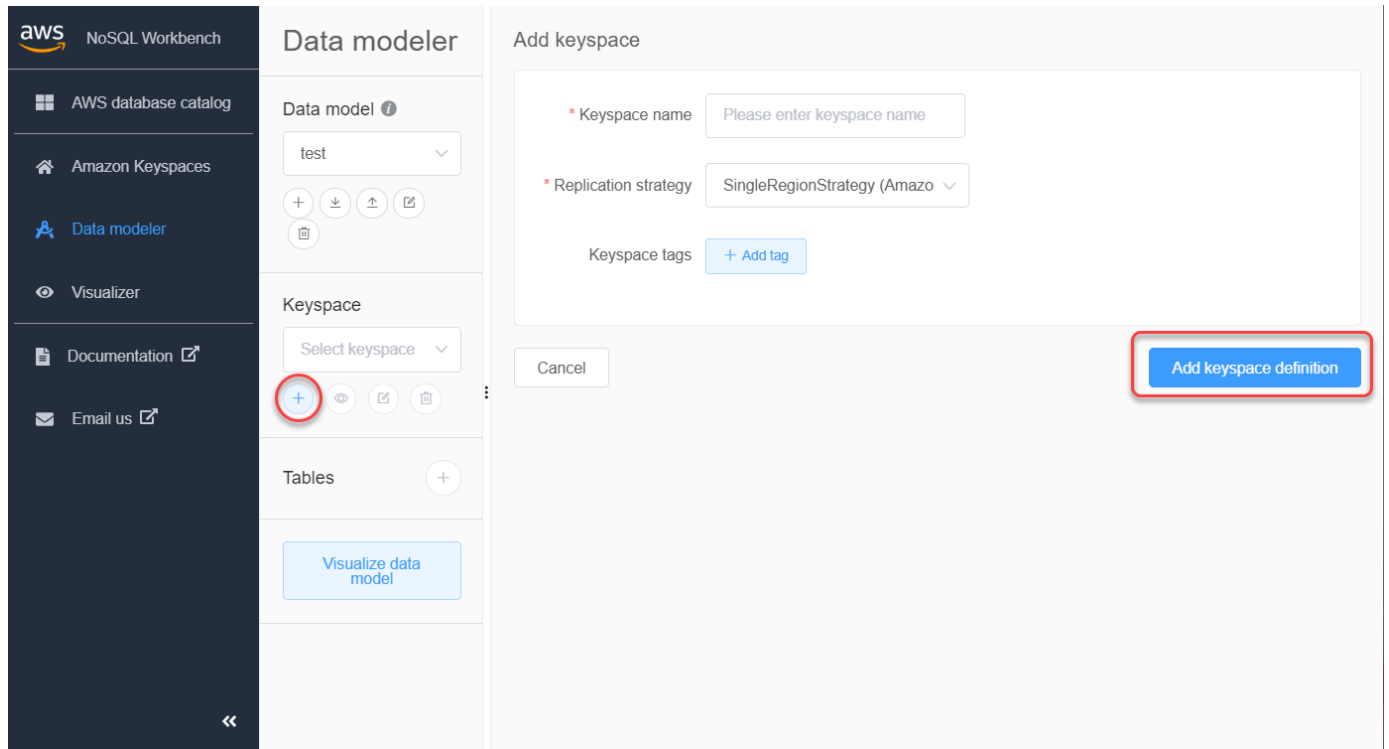
您可以使用 NoSQL Workbench 資料模型器，根據應用程式的資料存取模式設計新的資料模型。若要為 Amazon Keyspaces 建立新的資料模型，您可以使用 NoSQL Workbench 資料模型器來建立金鑰空間、資料表和資料欄。請依照下列步驟建立新的資料模型。

1. 若要建立新的金鑰空間，請選擇金鑰空間下的加號。

在此步驟中，選擇下列屬性和設定。

- 金鑰空間名稱 – 輸入新金鑰空間的名稱。
- 複寫策略 – 選擇金鑰空間的複寫策略。Amazon Keyspaces 使用 SingleRegionStrategy 在多個 AWS 可用區域中自動複寫資料三次。如果您打算將資料模型遞交至 Apache Cassandra 叢集，您可以選擇 SimpleStrategy 或 NetworkTopologyStrategy。
- 鍵空間標籤 – 資源標籤是選用的，可讓您以不同的方式分類資源，例如，依用途、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”](#)。

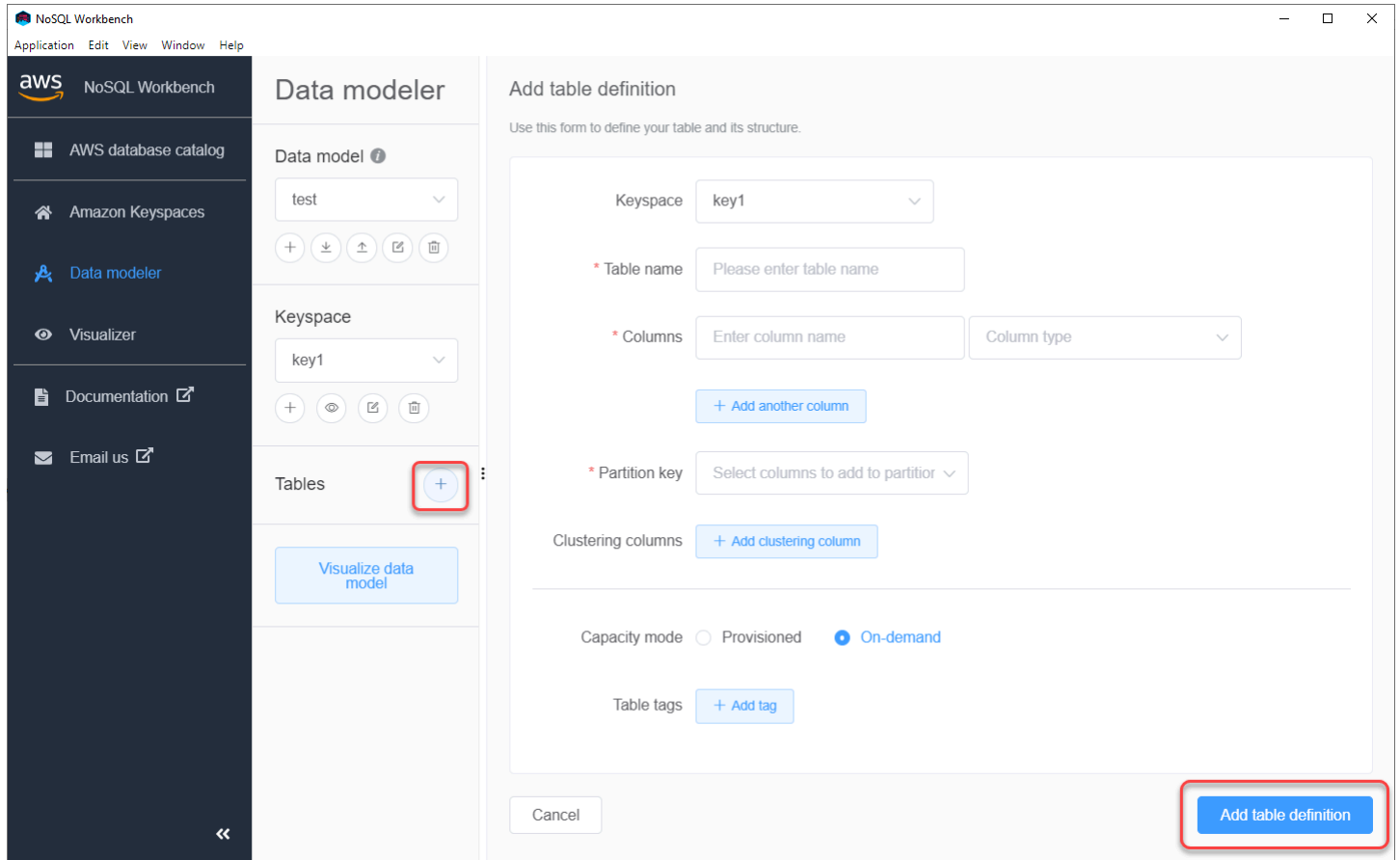
## 2. 選擇新增金鑰空間定義以建立金鑰空間。



## 3. 若要建立新的資料表，請選擇資料表旁的加號。在此步驟中，您會定義下列屬性和設定。

- 資料表名稱 – 新資料表的名稱。
- 資料欄 – 新增資料欄名稱，然後選擇資料類型。針對結構描述中的每個資料欄重複這些步驟。
- 分割區索引鍵 – 選擇分割區索引鍵的資料欄。
- 叢集資料欄 – 選擇叢集資料欄（選用）。
- 容量模式 – 選擇資料表的讀取/寫入容量模式。您可以選擇佈建或隨需容量。若要進一步了解容量模式，請參閱 [the section called “設定讀取/寫入容量模式”](#)。

- 資料表標籤 – 資源標籤是選用的，可讓您以不同的方式分類資源，例如，依用途、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”](#)。
- 選擇新增資料表定義以建立新的資料表。
  - 重複這些步驟來建立其他資料表。
  - 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

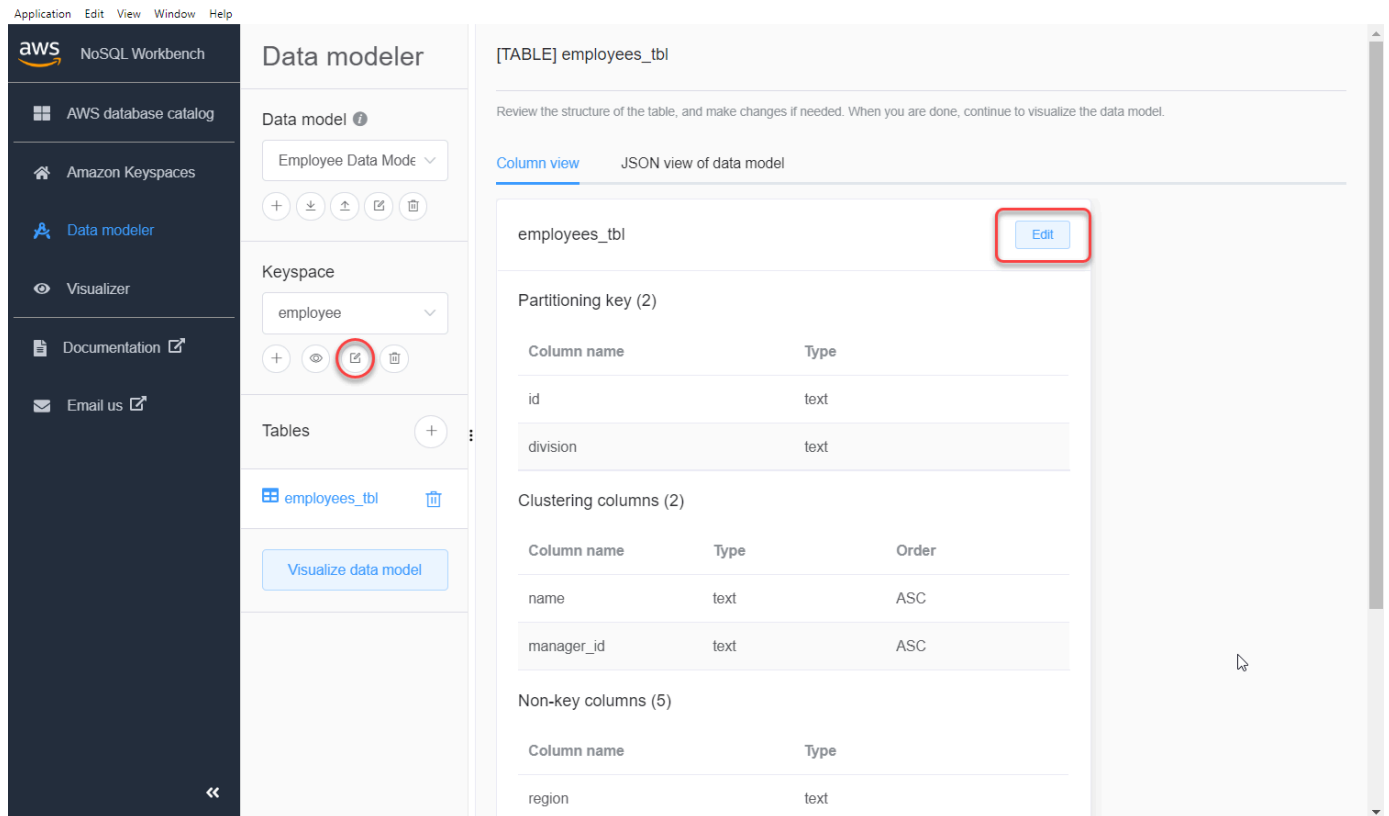
您可以使用資料模型器來匯入和修改使用 NoSQL Workbench 建立的現有資料模型。資料模型器也包含幾個範例資料模型，協助您開始使用資料模型。您可以使用 NoSQL Workbench 編輯的資料模型可以是從檔案、提供的範例資料模型或您先前建立的資料模型匯入的資料模型。

- 若要編輯金鑰空間，請選擇金鑰空間下的編輯符號。

在此步驟中，您可以編輯下列屬性和設定。

- 金鑰空間名稱 – 輸入新金鑰空間的名稱。
- 複寫策略 – 選擇金鑰空間的複寫策略。Amazon Keyspaces 使用 SingleRegionStrategy 在多個 AWS 可用區域中自動複寫資料三次。如果您打算將資料模型遞交至 Apache Cassandra 叢集，您可以選擇 SimpleStrategy 或 NetworkTopologyStrategy。
- 鍵空間標籤 – 資源標籤是選用的，可讓您以不同的方式分類資源，例如，依用途、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，請選擇資料表名稱旁的編輯。在此步驟中，您可以更新下列屬性和設定。

- 資料表名稱 – 新資料表的名稱。
- 資料欄 – 新增資料欄名稱，然後選擇資料類型。針對結構描述中的每個資料欄重複這些步驟。
- 分割區索引鍵 – 選擇分割區索引鍵的資料欄。
- 叢集資料欄 – 選擇叢集資料欄（選用）。
- 容量模式 – 選擇資料表的讀取/寫入容量模式。您可以選擇佈建或隨需容量。若要進一步了解容量模式，請參閱 [the section called “設定讀取/寫入容量模式”](#)。

- 資料表標籤 – 資源標籤是選用的，可讓您以不同的方式分類資源，例如，依用途、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”](#)。
- 選擇儲存編輯以更新資料表。
  - 繼續[the section called “視覺化資料模型”](#)以視覺化方式呈現您更新的資料模型。

## 如何將資料模型遞交至 Amazon Keyspaces 和 Apache Cassandra

本節說明如何將完成的資料模型遞交至 Amazon Keyspaces 和 Apache Cassandra 叢集。此程序會根據您在資料模型中定義的設定，自動建立金鑰空間和資料表的伺服器端資源。

The screenshot shows the AWS NoSQL Workbench Visualizer interface. On the left sidebar, the 'Visualizer' tab is selected. The main panel displays a visual representation of a table schema for 'routes'. The schema includes a primary key with partition key columns (airline\_id, origin\_id) and clustering column (stops), and non-key columns (origin\_airport, destination\_airport, equipment). The 'Commit to Amazon Keyspaces' button is highlighted with a red box.

| Primary key       |                  |                       |              | Non-key columns                  |                              |           |
|-------------------|------------------|-----------------------|--------------|----------------------------------|------------------------------|-----------|
| Partition key     |                  | Clustering columns    |              |                                  |                              |           |
| airline_id (text) | origin_id (text) | destination_id (text) | stops (text) | origin_airport (text)            | destination_airport (text)   | equipment |
| acme_airlines     | EWR              | MCI                   | 1            | Newark Liberty International     | Kansas City International    | 737       |
| trusted_airlines  | PHX              | MIC                   | 2            | Phoenix Sky Harbor International | Kansas City International    | 737       |
| freedom_airlines  | SFO              | EWR                   | 1            | San Francisco International      | Newark Liberty International | 747       |

### 主題

- [開始之前](#)
- [使用服務特定的登入資料連線至 Amazon Keyspaces](#)
- [使用 AWS Identity and Access Management \(IAM\) 登入資料連線至 Amazon Keyspaces](#)

- [使用已儲存的連線](#)
- [遞交至 Apache Cassandra](#)

## 開始之前

Amazon Keyspaces 需要使用 Transport Layer Security (TLS) 來協助保護與用戶端的連線。若要使用 TLS 連線至 Amazon Keyspaces，您必須先完成下列任務，才能開始。

- 下載下列數位憑證，並將檔案儲存在本機或主目錄中。
  1. AmazonRootCA1
  2. AmazonRootCA2
  3. AmazonRootCA3
  4. AmazonRootCA4
  5. Starfield Class 2 根目錄（選用 – 用於回溯相容性）

若要下載憑證，您可以使用下列命令。

```
curl -O https://www.amazontrust.com/repository/AmazonRootCA1.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA2.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA3.pem
curl -O https://www.amazontrust.com/repository/AmazonRootCA4.pem
curl -O https://certs.secureserver.net/repository/sf-class2-root.crt
```

### Note

Amazon Keyspaces 先前使用以 Starfield 類別 2 CA 為基礎的 TLS 憑證。AWS 正在將所有遷移 AWS 區域 至根據 Amazon Trust Services (Amazon 根 CAs 發行的憑證。在此轉換期間，請將用戶端設定為信任 Amazon 根 CAs1–4 和 Starfield 根，以確保所有區域的相容性。

使用範例中的名稱 *keyspaces-bundle.pem*，將所有下載的憑證合併成單一 pem 檔案。您可以執行下列命令，即可進行新增：請記下 檔案的路徑，您稍後需要此操作。

```
cat AmazonRootCA1.pem \
 AmazonRootCA2.pem \
 AmazonRootCA3.pem \
 AmazonRootCA4.pem \
```

```
sf-class2-root.crt \
> keyspaces-bundle.pem
```

儲存憑證檔案之後，您可以連線至 Amazon Keyspaces。其中一個選項是使用服務特定的登入資料進行連線。服務特定的登入資料是與特定 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。第二個選項是與使用 [AWS Signature 第 4 版程序 \(SigV4\)](#) 的 IAM 登入資料連線。若要進一步了解這兩個選項，請參閱 [the section called “建立程式設計存取憑證”](#)。

若要與服務特定的登入資料連線，請參閱 [the section called “使用服務特定的登入資料連線”](#)。

若要使用 IAM 登入資料連線，請參閱 [the section called “透過 IAM 憑證連線”](#)。

## 使用服務特定的登入資料連線至 Amazon Keyspaces

本節說明如何使用服務特定的登入資料來遞交您使用 NoSQL Workbench 建立或編輯的資料模型。

1. 若要使用服務特定的登入資料建立新的連線，請使用使用者名稱和密碼索引標籤選擇連線。
  - 開始之前，您必須使用 中記載的程序來建立服務特定的登入資料[the section called “建立服務特定的登入資料”](#)。

取得服務特定的登入資料後，您可以繼續設定連線。繼續執行下列其中一項操作：

- 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”](#)。
- 連接埠 – Amazon Keyspaces 使用連接埠 9142。

或者，您可以從檔案匯入已儲存的登入資料。

2. 選擇遞交以使用資料模型更新 Amazon Keyspaces。

## 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) 登入資料連線至 Amazon Keyspaces

本節說明如何使用 IAM 憑證遞交使用 NoSQL Workbench 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來建立 IAM 登入資料。
  - 對於主控台存取權，請使用您的 IAM 使用者名稱和密碼從 IAM 登入頁面登入 [AWS 管理主控台](#)。如需有關 AWS 安全登入資料的資訊，包括程式設計存取和長期登入資料的替代方案，請參閱《IAM 使用者指南》中的[AWS 安全登入](#)資料。如需登入的詳細資訊 AWS 帳戶，請參閱AWS 登入 《使用者指南》中的[如何登入 AWS](#)。
  - 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[搭配 AWS 資源使用臨時憑證](#)。
  - 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。
- 私密存取金鑰 – 輸入私密存取金鑰。
- 連接埠 – Amazon Keyspaces 使用連接埠 9142。
- AWS 公有憑證 – 指向第一個步驟中下載的憑證。AWS
- 持續連線 – 如果您想要在本機儲存 AWS 連線秘密，請選取此核取方塊。

2. 選擇遞交以使用資料模型更新 Amazon Keyspaces。

**i** 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. **i**

Persist connection



If you select this check box, AWS connection secrets will be persisted in

C:\Users\awsuser\aws\credentials

Cancel

Reset

Commit

## 使用已儲存的連線

如果您先前已設定與 Amazon Keyspaces 的連線，您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並繼續遞交更新。

### 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 : >

\* Saved connections

default



\* 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.

Cancel

Reset

Commit

## 遞交至 Apache Cassandra

本節會逐步引導您與 Apache Cassandra 叢集建立連線，以遞交使用 NoSQL Workbench 建立或編輯的資料模型。

 Note

只有使用 SimpleStrategy 或 建立的資料模型 NetworkTopologyStrategy 才能遞交至 Apache Cassandra 叢集。若要變更複寫策略，請在資料模型器中編輯金鑰空間。

1.
  - 使用者名稱 – 如果在叢集上啟用身分驗證，請輸入使用者名稱。
  - 密碼 – 如果在叢集上啟用身分驗證，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. 選擇遞交以使用資料模型更新 Apache Cassandra 叢集。

## 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

Password

\* Contact points

[+ Add another contact point](#)

\* Local data center

\* Port

Cancel

Reset

Commit

## NoSQL Workbench 中的範例資料模型

模型師和視覺化工具的首頁會顯示一些隨附於 NoSQL Workbench 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](#)
- [信用卡交易資料模型](#)
- [航空公司營運資料模型](#)

### 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 結構描述。

存取指定公司員工資訊的應用程式可以使用此資料模型。

此資料模型支援的存取模式為：

- 擷取具有指定 ID 的員工記錄。
- 擷取具有指定 ID 和部門的員工記錄。
- 擷取具有指定 ID 和名稱的員工記錄。

### 信用卡交易資料模型

此資料模型代表零售商店信用卡交易的 Amazon Keyspaces 結構描述。

信用卡交易的儲存不僅可協助存放區進行簿記，也有助於存放區經理分析購買趨勢，進而協助他們預測和規劃。

此資料模型支援的存取模式為：

- 依信用卡號碼、月份和年份以及日期擷取交易。
- 依信用卡號碼、類別和日期擷取交易。
- 依類別、位置和信用卡號碼擷取交易。
- 依信用卡號碼和爭端狀態擷取交易。

## 航空公司營運資料模型

此資料模型會顯示有關飛機航班的資料，包括機場、航空公司和航班路線。

示範的 Amazon Keyspaces 建模的關鍵元件是鍵值對、廣欄資料存放區、複合索引鍵和複雜的資料類型，例如地圖，以示範常見的 NoSQL 資料存取模式。

此資料模型支援的存取模式為：

- 擷取來自指定機場之指定航空公司的路由。
- 擷取具有指定目的地機場的路由。
- 使用直飛航班擷取機場。
- 擷取機場詳細資訊和航空公司詳細資訊。

## NoSQL Workbench 的版本歷史記錄

下表說明每個 NoSQL Workbench 用戶端應用程式版本的重要變更。

| 變更                                       | 描述                                                                                                                                                                              | 日期               |
|------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| Amazon Keyspaces 的 NoSQL Workbench – GA。 | Amazon Keyspaces 的 NoSQL Workbench 已全面推出。                                                                                                                                       | 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 用戶端可用於 Windows、macOS 和 Linux。如需詳細資訊，請參閱 <a href="#">Amazon Keyspaces 的 NoSQL Workbench</a> 。 | 2020 年 10 月 5 日  |

# 使用 AWS SDKs 程式碼範例

下列程式碼範例示範如何使用 Amazon Keyspaces 搭配 AWS 軟體開發套件 (SDK)。

基本概念是程式碼範例，這些範例說明如何在服務內執行基本操作。

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中查看內容中的動作。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 程式碼範例

- [使用 AWS SDKs 基本範例](#)
  - [Hello Amazon Keyspaces](#)
  - [使用 AWS SDK 了解 Amazon Keyspaces 的基本概念](#)
  - [使用 AWS SDKs 動作](#)
    - [CreateKeyspace 搭配 AWS SDK 使用](#)
    - [CreateTable 搭配 AWS SDK 使用](#)
    - [DeleteKeyspace 搭配 AWS SDK 使用](#)
    - [DeleteTable 搭配 AWS SDK 使用](#)
    - [GetKeyspace 搭配 AWS SDK 使用](#)
    - [GetTable 搭配 AWS SDK 使用](#)
    - [ListKeyspaces 搭配 AWS SDK 使用](#)
    - [ListTables 搭配 AWS SDK 使用](#)
    - [RestoreTable 搭配 AWS SDK 使用](#)
    - [UpdateTable 搭配 AWS SDK 使用](#)

## 使用 AWS SDKs 基本範例

下列程式碼範例示範如何搭配 AWS SDK 使用 Amazon Keyspaces (適用於 Apache Cassandra) 的基本功能。

### 範例

- [Hello Amazon Keyspaces](#)



- [使用 AWS SDK 了解 Amazon Keyspaces 的基本概念](#)
- [使用 AWS SDKs 動作](#)
  - [CreateKeyspace 搭配 AWS SDK 使用](#)
  - [CreateTable 搭配 AWS SDK 使用](#)
  - [DeleteKeyspace 搭配 AWS SDK 使用](#)
  - [DeleteTable 搭配 AWS SDK 使用](#)
  - [GetKeyspace 搭配 AWS SDK 使用](#)
  - [GetTable 搭配 AWS SDK 使用](#)
  - [ListKeyspaces 搭配 AWS SDK 使用](#)
  - [ListTables 搭配 AWS SDK 使用](#)
  - [RestoreTable 搭配 AWS SDK 使用](#)
  - [UpdateTable 搭配 AWS SDK 使用](#)

## Hello Amazon Keyspaces

下列程式碼範例示範如何開始使用 Amazon Keyspaces。

.NET

適用於 .NET 的 SDK

### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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《適用於 .NET 的 AWS SDK API 參考》中的 [ListKeyspaces](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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》中的 [ListKeyspaces](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
 val response = keyClient.listKeyspaces(keyspacesRequest)
 response.keyspaces?.forEach { keyspace ->
 println("The name of the keyspace is ${keyspace.keyspaceName}")
 }
 }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [ListKeyspaces](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [ListKeyspaces](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 使用 AWS SDK 了解 Amazon Keyspaces 的基本概念

下列程式碼範例示範如何：

- 建立鍵空間和資料表。資料表結構描述會保留電影資料，並啟用時間點復原。
- 使用具有 SigV4 驗證的安全 TLS 連線，以連接至鍵空間。
- 查詢資料表。新增、擷取和更新電影資料。
- 更新資料表。新增資料欄以追蹤觀看的電影。
- 將資料表還原至其先前的狀態，並清除資源。

### .NET

適用於 .NET 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.WriteLine($"Created {keyspaceName}. Waiting for it to become
available. ");
 do
 {
 getKeySpaceArn = await
keyspacesWrapper.GetKeyspace(keyspaceName);
 Console.WriteLine(". ");
 } while (getKeySpaceArn != keyspaceArn);
}
catch (ResourceNotFoundException)
{
 Console.WriteLine("Waiting for keyspace to be created.");
}

Console.WriteLine($"\\n\\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.WriteLine("Waiting for the new table to be active. ");
 do
 {
 try
 {
 resp = await keyspacesWrapper.GetTable(keyspaceName,
tableName);

 Console.WriteLine(".");
 }
 catch (ResourceNotFoundException)
 {
 Console.WriteLine(".");
 }
 } 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 driver 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《適用於 .NET 的 AWS SDK 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

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
 * 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 keyspace 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)
 .addColumnns(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.bindStatementBuilder()
 .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

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/**
```

```
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](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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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 collList = ArrayList<ColumnDefinition>()
collList.add(defTitle)
collList.add(defYear)
collList.add(defReleaseDate)
collList.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 schemaDefinition0b =
 SchemaDefinition {
 partitionKeys = keyList
 allColumns = collList
 }

val timeRecovery =
 PointInTimeRecovery {
 status = PointInTimeRecoveryStatus.Enabled
 }

val tableRequest =
 CreateTableRequest {
 keyspaceName = keySpaceVal
 tableName = tableNameVal
 schemaDefinition = schemaDefinition0b
 pointInTimeRecovery = timeRecovery
 }

KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
 val response = keyClient.createTable(tableRequest)
 println("The table ARN is ${response.resourceArn}")
}
```

```
}

suspend fun listKeyspacesPaginator() {
 KeyspacesClient.fromEnvironment { 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.fromEnvironment { 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.fromEnvironment { 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 SDK for Kotlin API 參考》中的下列主題。
  - [CreateKeyspace](#)
  - [CreateTable](#)

- [DeleteKeyspace](#)
- [DeleteTable](#)
- [GetKeyspace](#)
- [GetTable](#)
- [ListKeyspaces](#)
- [ListTables](#)
- [RestoreTable](#)
- [UpdateTable](#)

## Python

適用於 Python 的 SDK (Boto3)

### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

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 keyspace in your account.
 """
 print("Let's create a keyspace.")
 ks_name = q.ask(
```

```

 "Enter a name for your new keyspace.\n\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 keyspace 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.")
```

定義包裝鍵空間和資料表動作的類別。

```
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

```

定義一種類別，其能建立鍵空間的 TLS 連線、使用 SigV4 驗證，並可將 CQL 查詢傳送至鍵空間中的資料表。

```

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 詳細資訊，請參閱《適用於 Python (Boto3) 的 AWS SDK API 參考》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 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## 使用 AWS SDKs動作

下列程式碼範例示範如何使用 AWS SDKs 執行個別 Amazon Keyspaces 動作。每個範例均包含 GitHub 的連結，您可以在連結中找到設定和執行程式碼的相關說明。

下列範例僅包含最常使用的動作。如需完整清單，請參閱《[Amazon Keyspaces \(適用於 Apache Cassandra\) API 參考](#)》。

### 範例

- [CreateKeyspace 搭配 AWS SDK 使用](#)
- [CreateTable 搭配 AWS SDK 使用](#)
- [DeleteKeyspace 搭配 AWS SDK 使用](#)
- [DeleteTable 搭配 AWS SDK 使用](#)
- [GetKeyspace 搭配 AWS SDK 使用](#)
- [GetTable 搭配 AWS SDK 使用](#)
- [ListKeyspaces 搭配 AWS SDK 使用](#)
- [ListTables 搭配 AWS SDK 使用](#)
- [RestoreTable 搭配 AWS SDK 使用](#)
- [UpdateTable 搭配 AWS SDK 使用](#)

## CreateKeyspace 搭配 AWS SDK 使用

下列程式碼範例示範如何使用 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：

- [了解基本概念](#)

## .NET

適用於 .NET 的 SDK

### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [CreateKeyspace](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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 [CreateKeyspace](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun createKeySpace(keyspaceNameVal: String) {
 val keySpaceRequest =
 CreateKeyspaceRequest {
 keyspaceName = keyspaceNameVal
 }

 KeyspacesClient.fromEnvironment { 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《適用於 Kotlin 的 AWS SDK API 參考》中的 [CreateKeyspace](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [CreateKeyspace](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

TRY.



```

oo_result = lo_kys->createkeyspace(
 iv_keyspacename = iv_keyspace_name).
MESSAGE 'Keyspace created successfully.' TYPE 'I'.
CATCH /aws1/cx_kysconflictexception.
MESSAGE 'Keyspace already exists.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [CreateKeyspace](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## CreateTable 搭配 AWS SDK 使用

下列程式碼範例示範如何使用 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：

- [了解基本概念](#)

### .NET

適用於 .NET 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

/// <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 的詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [CreateTable](#)。

## Java

### 適用於 Java 2.x 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

 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》中的 [CreateTable](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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 schemaDefinition0b =
 SchemaDefinition {
 partitionKeys = keyList
 allColumns = collist
 }
```

```

val timeRecovery =
 PointInTimeRecovery {
 status = PointInTimeRecoveryStatus.Enabled
 }

val tableRequest =
 CreateTableRequest {
 keyspaceName = keySpaceVal
 tableName = tableNameVal
 schemaDefinition = schemaDefinitionObj
 pointInTimeRecovery = timeRecovery
 }

KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
 val response = keyClient.createTable(tableRequest)
 println("The table ARN is ${response.resourceArn}")
}
}

```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [CreateTable](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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《適用於 Python (Boto3) 的 AWS SDK API 參考》中的 [CreateTable](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
TRY.
 " Define schema with columns
 DATA(lt_columns) = VALUE /aws1/
cl_kyscolumndefinition=>tt_columndefinitionlist(
 (NEW /aws1/cl_kyscolumndefinition(iv_name = 'title' iv_type =
 'text'))
 (NEW /aws1/cl_kyscolumndefinition(iv_name = 'year' iv_type =
 'int'))
 (NEW /aws1/cl_kyscolumndefinition(iv_name = 'release_date' iv_type =
 'timestamp'))
 (NEW /aws1/cl_kyscolumndefinition(iv_name = 'plot' iv_type =
 'text'))
).

 " Define partition keys
 DATA(lt_partition_keys) = VALUE /aws1/
cl_kyspartitionkey=>tt_partitionkeylist(
 (NEW /aws1/cl_kyspartitionkey(iv_name = 'year'))
 (NEW /aws1/cl_kyspartitionkey(iv_name = 'title'))
).

 " Create schema definition
 DATA(lo_schema) = NEW /aws1/cl_kysschemadefinition(
 it_allcolumns = lt_columns
 it_partitionkeys = lt_partition_keys).
```



```

 " Enable point-in-time recovery
 DATA(lo_pitr) = NEW /aws1/cl_kyspointintimerec(
 iv_status = 'ENABLED').

 oo_result = lo_kys->createtable(
 iv_keyspacename = iv_keyspace_name
 iv_tablename = iv_table_name
 io_schemadefinition = lo_schema
 io_pointintimerecovery = lo_pitr).
 MESSAGE 'Table created successfully.' TYPE 'I'.
 CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
 DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
 MESSAGE lv_error TYPE 'E'.
 ENDTRY.

```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的 [CreateTable](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## DeleteKeyspace 搭配 AWS SDK 使用

下列程式碼範例示範如何使用 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：

- [了解基本概念](#)

### .NET

適用於 .NET 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [DeleteKeyspace](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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》中的 [DeleteKeyspace](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun deleteKeyspace(keyspaceNameVal: String?) {
 val deleteKeyspaceRequest =
 DeleteKeyspaceRequest {
 keyspaceName = keyspaceNameVal
 }

 KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
 keyClient.deleteKeyspace(deleteKeyspaceRequest)
 }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [DeleteKeyspace](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [DeleteKeyspace](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
TRY.
 lo_kys->deletekeyspace(
 iv_keyspacename = iv_keyspace_name).
 MESSAGE 'Keyspace deleted successfully.' TYPE 'I'.
 CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
 DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.
 MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [DeleteKeyspace](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## DeleteTable 搭配 AWS SDK 使用

下列程式碼範例示範如何使用 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：

- [了解基本概念](#)

## .NET

### 適用於 .NET 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <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 的詳細資訊，請參閱《[適用於 .NET 的 AWS SDK API 參考](#)》中的 DeleteTable。

## Java

### 適用於 Java 2.x 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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》中的 [DeleteTable](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun deleteTable(
 keyspaceNameVal: String?,
 tableNameVal: String?,
) {
 val tableRequest =
 DeleteTableRequest {
 keyspaceName = keyspaceNameVal
 tableName = tableNameVal
 }

 KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
 keyClient.deleteTable(tableRequest)
 }
}
```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [DeleteTable](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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《適用於 Python (Boto3) 的 AWS SDK API 參考》中的 [DeleteTable](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
TRY.
 lo_kys->deletetable(
 iv_keyspacename = iv_keyspace_name
 iv_tablename = iv_table_name).
 MESSAGE 'Table deleted successfully.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
 DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
 MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的 [DeleteTable](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## GetKeyspace 搭配 AWS SDK 使用

下列程式碼範例示範如何使用 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：

- [了解基本概念](#)

### .NET

適用於 .NET 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [GetKeyspace](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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》中的 [GetKeyspace](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun checkKeyspaceExistence(keyspaceNameVal: String?) {
 val keyRequest =
 GetKeyspaceRequest {
 keyspaceName = keyspaceNameVal
 }
 KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
 val response: GetKeyspaceResponse =
 keyClient.getKeyspace(keyRequest)
 val name = response.keyspaceName
 println("The $name KeySpace is ready")
 }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [GetKeyspace](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [GetKeyspace](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

TRY.
 oo_result = lo_kys->getkeyspace(
 iv_keyspacename = iv_keyspace_name).
 MESSAGE 'Keyspace retrieved successfully.' TYPE 'I'.
CATCH /aws1/cx_kysresourcenotfoundex.
 MESSAGE 'Keyspace does not exist.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
 DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
 MESSAGE lv_error TYPE 'E'.
ENDTRY.

```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [GetKeyspace](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## GetTable 搭配 AWS SDK 使用

下列程式碼範例示範如何使用 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：

- [了解基本概念](#)

### .NET

適用於 .NET 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [GetTable](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

 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》中的 [GetTable](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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.fromEnvironment { 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}")
 }
 }
}
}

```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [GetTable](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [GetTable](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
TRY.
 oo_result = lo_kys->gettable(
 iv_keyspacename = iv_keyspace_name
 iv_tablename = iv_table_name).
 MESSAGE 'Table information retrieved successfully.' TYPE 'I'.
CATCH /aws1/cx_kysresourcenotfoundex.
 MESSAGE 'Table does not exist.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
 DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.
 MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [GetTable](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## ListKeyspaces 搭配 AWS SDK 使用

下列程式碼範例示範如何使用 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：

- [了解基本概念](#)

## .NET

### 適用於 .NET 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [ListKeyspaces](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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》中的 [ListKeyspaces](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
suspend fun listKeyspacesPaginator() {
 KeyspacesClient.fromEnvironment { region = "us-east-1" }.use { keyClient ->
 keyClient
 .listKeyspacesPaginated(ListKeyspacesRequest {})
 .transform { it.keyspaces?.forEach { obj -> emit(obj) } }
 .collect { obj ->
 println("Name: ${obj.keyspaceName}")
 }
 }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [ListKeyspaces](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [ListKeyspaces](#)。

## SAP ABAP

適用於 SAP ABAP 的開發套件

### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
TRY.
 oo_result = lo_kys->listkeyspaces(
 iv_maxresults = iv_max_results).
 MESSAGE 'Keyspaces listed successfully.' TYPE 'I'.
 CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
 DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception->av_err_msg }|.
 MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [ListKeyspaces](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## ListTables 搭配 AWS SDK 使用

下列程式碼範例示範如何使用 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：

- [了解基本概念](#)



## .NET

### 適用於 .NET 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <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 的詳細資訊，請參閱《適用於 .NET 的 AWS SDK API 參考》中的 [ListTables](#)。

## Java

### 適用於 Java 2.x 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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 參考中的 [ListTables](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

suspend fun listTables(keyspaceNameVal: String?) {
 val tablesRequest =
 ListTablesRequest {
 keyspaceName = keyspaceNameVal
 }

 KeyspacesClient.fromEnvironment { 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}")
 }
 }
}

```

- 如需 API 的詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [ListTables](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

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《適用於 Python (Boto3) 的 AWS SDK API 參考》中的 [ListTables](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

TRY.
 oo_result = lo_kys->listtables(
 iv_keyspacename = iv_keyspace_name).
 MESSAGE 'Tables listed successfully.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
 DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
 MESSAGE lv_error TYPE 'E'.

```

```
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 SAP ABAP 的 AWS SDK API 參考》中的 [ListTables](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## RestoreTable 搭配 AWS SDK 使用

下列程式碼範例示範如何使用 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：

- [了解基本概念](#)

### .NET

適用於 .NET 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [RestoreTable](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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》中的 [RestoreTable](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
 val response = keyClient.restoreTable(restoreTableRequest)
 println("The ARN of the restored table is ${response.restoredTableArn}")
 }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [RestoreTable](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [RestoreTable](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

TRY.
 oo_result = lo_kys->restoretale(
 iv_sourcekeyspacename = iv_source_keyspace_name
 iv_sourcetablename = iv_source_table_name
 iv_targetkeyspacename = iv_target_keyspace_name
 iv_targettablename = iv_target_table_name
 iv_restoretimestamp = iv_restore_timestamp).

```

```

 MESSAGE 'Table restore initiated successfully.' TYPE 'I'.
 CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).
 DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
 MESSAGE lv_error TYPE 'E'.
 ENDTRY.

```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [RestoreTable](#)。

如需 AWS SDK 開發人員指南和程式碼範例的完整清單，請參閱 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

## UpdateTable 搭配 AWS SDK 使用

下列程式碼範例示範如何使用 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：

- [了解基本概念](#)

### .NET

適用於 .NET 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

/// <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《適用於 .NET 的 AWS SDK API 參考》中的 [UpdateTable](#)。

## Java

### SDK for Java 2.x

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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》中的 [UpdateTable](#)。

## Kotlin

### 適用於 Kotlin 的 SDK

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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.fromEnvironment { region = "us-east-1" }.use { keyClient ->
 keyClient.updateTable(tableRequest)
 }
}
```

- 如需 API 詳細資訊，請參閱《適用於 Kotlin 的 AWS SDK API 參考》中的 [UpdateTable](#)。

## Python

### 適用於 Python 的 SDK (Boto3)

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```
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 詳細資訊，請參閱《AWS SDK for Python (Boto3) API 參考》中的 [UpdateTable](#)。

## SAP ABAP

### 適用於 SAP ABAP 的開發套件

#### Note

GitHub 上提供更多範例。尋找完整範例，並了解如何在 [AWS 程式碼範例儲存庫](#) 中設定和執行。

```

TRY.
 " Add a new column to track watched movies
 DATA(lt_add_columns) = VALUE /aws1/
cl_kyscolumndefinition=>tt_columndefinitionlist(
 (NEW /aws1/cl_kyscolumndefinition(iv_name = 'watched' iv_type =
'boolean'))
).

 oo_result = lo_kys->updatetable(
 iv_keyspacename = iv_keyspace_name
 iv_tablename = iv_table_name
 it_addcolumns = lt_add_columns).
 MESSAGE 'Table updated successfully.' TYPE 'I'.
CATCH /aws1/cx_rt_service_generic INTO DATA(lo_exception).

```

```
DATA(lv_error) = |"{ lo_exception->av_err_code }" - { lo_exception-
>av_err_msg }|.
MESSAGE lv_error TYPE 'E'.
ENDTRY.
```

- 如需 API 詳細資訊，請參閱《適用於 AWS SAP ABAP 的 SDK API 參考》中的 [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 [搭配 AWS SDK 使用此服務](#)。此主題也包含有關入門的資訊和舊版 SDK 的詳細資訊。

# Amazon Keyspaces (適用於 Apache Cassandra) 程式庫和工具

本節提供有關 Amazon Keyspaces (適用於 Apache Cassandra) 程式庫、程式碼範例和工具的資訊。

## 主題

- [程式庫和範例](#)
- [反白的範例和開發人員工具庫](#)

## 程式庫和範例

您可以在 [和 AWS 範例](#) 儲存庫的 GitHub 中找到 Amazon Keyspaces 開放原始碼程式庫 [AWS](#) 和開發人員工具。

## Amazon Keyspaces (適用於 Apache Cassandra) 開發人員工具組

此儲存庫為 Amazon Keyspaces 提供具有實用開發人員工具的 Docker 映像。例如，它包含具有最佳實務的 CQLSHRC 檔案、cqlsh 的選用 AWS 身分驗證擴展，以及執行常見任務的協助工具。工具組已針對 Amazon Keyspaces 最佳化，但也適用於 Apache Cassandra 叢集。

<https://github.com/aws-samples/amazon-keyspaces-toolkit>.

## Amazon Keyspaces (適用於 Apache Cassandra) 範例

此儲存庫是我們 Amazon Keyspaces 範例程式碼的官方清單。儲存庫會依語言細分為區段（請參閱 [範例](#)）。每個語言都有自己的範例子區段。這些範例示範常見的 Amazon Keyspaces 服務實作和模式，您可以在建置應用程式時使用。

<https://github.com/aws-samples/amazon-keyspaces-examples/>.

## AWS Signature 第 4 版 (SigV4) 身分驗證外掛程式

外掛程式可讓您使用 AWS Identity and Access Management (IAM) 使用者和角色來管理對 Amazon Keyspaces 的存取。

Java : <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>。

Go : <https://github.com/aws/aws-sigv4-auth-cassandra-gocql-driver-plugin>。

## 反白的範例和開發人員工具庫

以下是適用於 Amazon Keyspaces (適用於 Apache Cassandra) 的一系列實用社群工具。

### Amazon Keyspaces 通訊協定緩衝區

您可以搭配 Amazon Keyspaces 使用 Protocol Buffers (Protobuf)，提供 Apache Cassandra 使用者定義類型 (UDTs) 的替代方案。Protobuf 是一種免費且開放原始碼的跨平台資料格式，用於序列化結構化資料。您可以使用 CQL BLOB 資料類型和重構 UDTs 來存放 Protobuf 資料，同時跨應用程式和程式設計語言保留結構化資料。

此儲存庫提供程式碼範例，可連線至 Amazon Keyspaces、建立新的資料表，以及插入包含 Protobuf 訊息的資料列。然後，會以強烈的一致性讀取資料列。

<https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/java/datastax-v4/protobuf-user-defined-types>

### CloudFormation 為 Amazon Keyspaces (適用於 Apache Cassandra) 指標建立 Amazon CloudWatch 儀表板的 範本

此儲存庫提供 CloudFormation 範本，可快速設定 Amazon Keyspaces 的 CloudWatch 指標。使用此範本可讓您透過提供可部署的預先建置 CloudWatch 儀表板與常用的指標，更輕鬆地開始。

<https://github.com/aws-samples/amazon-keyspaces-cloudwatch-cloudformation-templates>。

### 搭配 使用 Amazon Keyspaces (適用於 Apache Cassandra) AWS Lambda

儲存庫包含示範如何從 Lambda 連線至 Amazon Keyspaces 的範例。以下是一些範例。

C#/。NET : <https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/dotnet/datastax-v3/connection-lambda>。

Java : <https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/java/datastax-v4/connection-lambda>。

另一個顯示如何從 Python Lambda 部署和使用 Amazon Keyspaces 的 Lambda 範例可從下列儲存庫取得。

<https://github.com/aws-samples/aws-keyspaces-lambda-python>

## 搭配 Spring 使用 Amazon Keyspaces (適用於 Apache Cassandra)

這是示範如何搭配 Spring Boot 使用 Amazon Keyspaces 的範例。

<https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/java/datastax-v4/spring>

## 搭配 Scala 使用 Amazon Keyspaces (適用於 Apache Cassandra)

這是示範如何使用 SigV4 身分驗證外掛程式搭配 Scala 連線至 Amazon Keyspaces 的範例。

<https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/scala/datastax-v4/connection-sigv4>

## 搭配 使用 Amazon Keyspaces (適用於 Apache Cassandra) AWS Glue

這是示範如何使用 Amazon Keyspaces 的範例 AWS Glue。

<https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/scala/datastax-v4/aws-glue>

## 要 CloudFormation 轉換的 Amazon Keyspaces (適用於 Apache Cassandra) Cassandra 查詢語言 (CQL)

此套件實作命令列工具，將 Apache Cassandra 查詢語言 (CQL) 指令碼 CloudFormation 轉換為 (CloudFormation) 範本，這可讓 Amazon Keyspaces 結構描述在 CloudFormation 堆疊中輕鬆管理。

<https://github.com/aws/amazon-keyspaces-cql-to-cfn-converter>.

## 適用於 Java 的 Apache Cassandra 驅動程式的 Amazon Keyspaces (適用於 Apache Cassandra) 協助程式

此儲存庫包含搭配 Amazon Keyspaces (適用於 Apache Cassandra) 使用 DataStax Java 驅動程式時的驅動程式政策、範例和最佳實務。

<https://github.com/aws-samples/amazon-keyspaces-java-driver-helpers>.

## Amazon Keyspaces (適用於 Apache Cassandra) 快照壓縮示範

此儲存庫示範如何壓縮、存放和讀取/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>.

## Amazon Keyspaces (適用於 Apache Cassandra) 和 Amazon S3 轉碼器示範

自訂 Amazon S3 Codec 支援透明、使用者可設定的 UUID 指標映射至 Amazon S3 物件。

<https://github.com/aws-samples/amazon-keyspaces-large-object-s3-demo>.

# 使用 Amazon Keyspaces 設計和架構的最佳實務（適用於 Apache Cassandra）

使用本節可快速尋找在使用 Amazon Keyspaces 時，將效能最大化並盡可能降低輸送量成本的建議。

## 內容

- [NoSQL 設計的主要差異和設計原則](#)
  - [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](#)
- [最佳化無伺服器環境的用戶端驅動程式連線](#)
  - [連線在 Amazon Keyspaces 中的運作方式](#)
  - [如何在 Amazon Keyspaces 中設定連線](#)
  - [如何在 Amazon Keyspaces 中設定連線的重試政策](#)
  - [如何在 Amazon Keyspaces 中透過 VPC 端點設定連線](#)
  - [如何在 Amazon Keyspaces 中監控連線](#)
  - [如何處理 Amazon Keyspaces 中的連線錯誤](#)
- [資料建模最佳實務：設計資料模型的建議](#)
  - [如何在 Amazon Keyspaces 中有效使用分割區金鑰](#)
    - [使用寫入碎片跨分割區平均分配工作負載](#)
    - [使用複合分割區索引鍵和隨機值的碎片](#)
    - [使用複合分割區索引鍵和計算值的碎片](#)
- [最佳化 Amazon Keyspaces 資料表的成本](#)
  - [在資料表層級評估您的成本](#)
    - [如何檢視單一 Amazon Keyspaces 資料表的成本](#)
    - [Cost Explorer 的預設檢視](#)
    - [如何在 Cost Explorer 中使用和套用資料表標籤](#)
  - [評估資料表的容量模式](#)
    - [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](#)
- [評估資料表的 Application Auto Scaling 設定](#)
  - [了解 Application Auto Scaling 設定](#)
  - [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](#)
- [識別未使用的資源，以最佳化 Amazon Keyspaces 中的成本](#)
  - [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，以最佳化效能和成本](#)
  - [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 資料表擷取取用指標](#)
  - [如何識別佈建不足的 Amazon Keyspaces 資料表](#)
  - [如何識別過度佈建的 Amazon Keyspaces 資料表](#)

## NoSQL 設計的主要差異和設計原則

Amazon Keyspaces 等 NoSQL 資料庫系統使用替代模型進行資料管理，例如鍵/值對或文件儲存。當您從關聯式資料庫管理系統切換到 Amazon Keyspaces 等 NoSQL 資料庫系統時，請務必了解主要差異和特定設計方法。

### 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 Amazon Keyspaces 等 NoSQL 資料庫中，可以透過有限數量的方式有效率地查詢資料，在其中查詢可能昂貴且緩慢。

這些差異導致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，讓最常見和重要的查詢盡可能快速且經濟實惠。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

- 相反地，在您知道需要回答的問題之前，您不應該開始設計 Amazon Keyspaces 的結構描述。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 應用程式中盡可能維持最少的資料表。擁有較少的資料表可讓物件更具可擴展性、需要較少的許可管理，並減少 Amazon Keyspaces 應用程式的額外負荷。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：了解一次要儲存和請求的資料量，有助於判斷分割資料的最有效方式。

- **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割區數量，以及有效率地將資料分散到這些分割區來擴展。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 應用程式中盡可能維持最少的資料表。

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。
- **發布佇列。** 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」。

這些一般原則會轉換為一些常見的設計模式，您可以用來在 Amazon Keyspaces 中有效率地建立資料模型。

## 最佳化無伺服器環境的用戶端驅動程式連線

若要與 Amazon Keyspaces 通訊，您可以使用您選擇的任何現有 Apache Cassandra 用戶端驅動程式。由於 Amazon Keyspaces 是無伺服器服務，我們建議您最佳化用戶端驅動程式的連線組態，以滿足應用程式的輸送量需求。本主題介紹最佳實務，包括如何計算應用程式需要多少連線，以及監控和錯誤處理連線。

### 主題

- [連線在 Amazon Keyspaces 中的運作方式](#)
- [如何在 Amazon Keyspaces 中設定連線](#)
- [如何在 Amazon Keyspaces 中設定連線的重試政策](#)

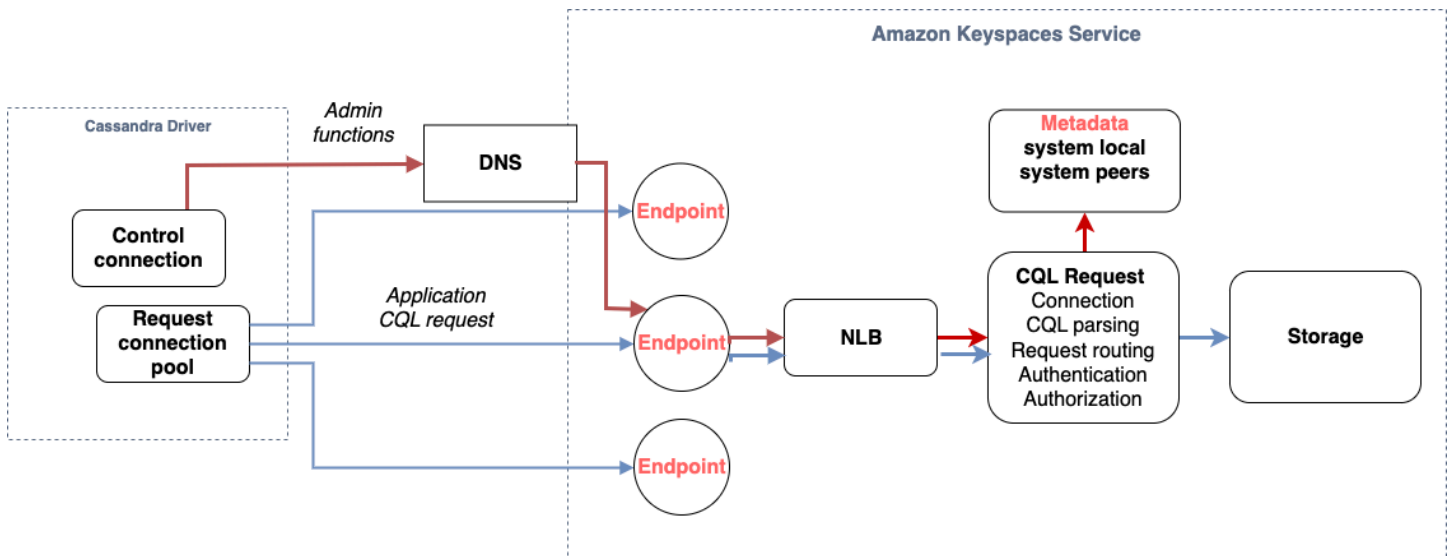


- [如何在 Amazon Keyspaces 中透過 VPC 端點設定連線](#)
- [如何在 Amazon Keyspaces 中監控連線](#)
- [如何處理 Amazon Keyspaces 中的連線錯誤](#)

## 連線在 Amazon Keyspaces 中的運作方式

本節概述用戶端驅動程式連線如何在 Amazon Keyspaces 中運作。由於 Cassandra 用戶端驅動程式設定錯誤可能會導致 Amazon Keyspaces 中的 `PerConnectionRequestExceeded` 事件，因此需要在用戶端驅動程式組態中設定正確數量的連線，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才能建立初始連線。Amazon Keyspaces 使用 DNS 將初始連線路由到許多可用端點之一。端點會連接到網路負載平衡器，進而建立與機群中其中一個請求處理常式的連線。建立初始連線後，用戶端驅動程式會從 `system.peers` 資料表收集所有可用端點的相關資訊。透過此資訊，用戶端驅動程式可以建立與所列端點的額外連線。用戶端驅動程式可以建立的連線數目會受限於用戶端驅動程式設定中指定的本機連線數目。根據預設，大多數用戶端驅動程式會為每個端點建立一個連線，並建立連至 Cassandra 的連線集區，以及透過該連線集區的負載平衡查詢。雖然可以建立與相同端點的多個連線，但在網路負載平衡器後方，它們可能會連接到許多不同的請求處理常式。透過公有端點連線時，建立與 `system.peers` 資料表中所列九個端點的連線，會產生與不同請求處理常式的九個連線。



## 如何在 Amazon Keyspaces 中設定連線

Amazon Keyspaces 每秒支援每個 TCP 連線最多 3,000 個 CQL 查詢。由於驅動程式可以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 請求，以允許額外負荷、流量爆增和更好的負載平衡。請遵循下列步驟，以確保您驅動程式的連線已針對應用程式的需求正確設定。



增加驅動程式在其連線集區中維護的每個 IP 地址的連線數量。

- 大多數 Cassandra 驅動程式會建立連至 Cassandra 的連線集區，並針對該連線集區進行負載平衡查詢。大多數驅動程式的預設行為是建立與每個端點的單一連線。Amazon Keyspaces 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 每秒支援每個 TCP 連線最多 3,000 個 CQL 查詢，因此使用預設設定的驅動程式 CQL 查詢輸送量上限為每秒 27,000 個 CQL 查詢。如果您使用驅動程式的預設設定，則單一連線可能必須處理超過每秒 3,000 個 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 `PerConnectionRequestExceeded` 事件，您必須設定驅動程式為每個端點建立額外的連線，以分配輸送量。
- 作為 Amazon Keyspaces 的最佳實務，假設每個連線每秒可以支援 500 個 CQL 查詢。
- 這表示對於需要支援分佈在九個可用端點上每秒 27,000 個 CQL 查詢的生產應用程式，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 個請求。

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 地址的連線數。

若要判斷應用程式每個端點需要設定的連線數，請考慮下列範例。您有一個應用程式需要支援每秒 20,000 個 CQL 查詢，其中包含 10,000 `INSERT`、5,000 `SELECT` 和 5,000 個 `DELETE` 操作。Java 應用程式正在 Amazon Elastic Container Service (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 Amazon Keyspaces 的單一工作階段。您可以使用計算來估計您需要為驅動程式設定多少連線使用以下輸入。

1. 您的應用程式需要支援的每秒請求數。
2. 將可用執行個體的數量減去一個執行個體，以考慮維護或故障。
3. 可用端點的數量。如果您要透過公有端點連線，您有九個可用的端點。如果您使用的是 VPC 端點，則視區域而定，您有兩到五個可用的端點。
4. 使用每個連線每秒 500 個 CQL 查詢做為 Amazon Keyspaces 的最佳實務。
5. 將結果四捨五入。

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，每個端點只能設定一個連線。

## 如何在 Amazon Keyspaces 中設定連線的重試政策

設定 Amazon Keyspaces 連線的重試政策時，建議您實作 Amazon Keyspaces 重試政策 `AmazonKeyspacesExponentialRetryPolicy`。此重試政策更適合在與驅動程式 不同的 Amazon Keyspaces 連線之間重試 `DefaultRetryPolicy`。

使用 `AmazonKeyspacesExponentialRetryPolicy`，您可以為符合您需求的連線設定重試嘗試次數。根據預設，的重試嘗試次數 `AmazonKeyspacesExponentialRetryPolicy` 設定為 3。

另一個優點是 Amazon Keyspaces 重試政策會傳回 服務傳回的原始例外狀況，這表示請求嘗試失敗的原因。預設重試政策只會傳回一般 `NoHostAvailableException`，這可能會隱藏對請求失敗的洞察。

若要使用 設定請求重試政策 `AmazonKeyspacesExponentialRetryPolicy`，建議您設定少量重試，並處理應用程式程式碼中傳回的任何例外狀況。

如需實作重試政策的程式碼範例，請參閱 Github 上的 [Amazon Keyspaces 重試政策](#)。

## 如何在 Amazon Keyspaces 中透過 VPC 端點設定連線

透過私有 VPC 端點連線時，您很可能有 3 個端點可用。每個區域的 VPC 端點數量可以不同，取決於可用區域的數量，以及指派的 VPC 中的子網路數量。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，您最多可以有兩個 Amazon Keyspaces 端點。端點數量不會影響擴展，但確實會增加您需要在驅動程式組態中建立的連線數量。請考量下列範例。您的應用程式需要支援 20,000 個 CQL 查詢，並在 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 Amazon Keyspaces 的單一工作階段。唯一的差別是不同 中有多少個端點可用 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 需要額外的許可，才能動態探索可用的 VPC 端點並填入 `system.peers` 資料表。如需詳細資訊，請參閱 [the section called “使用界面 VPC 端點資訊填入 `system.peers` 資料表項目”](#)。

使用不同的私有 VPC 端點存取 Amazon Keyspaces 時 AWS 帳戶，您可能只會看到單一 Amazon Keyspaces 端點。同樣地，這不會影響 Amazon Keyspaces 可能輸送量的規模，但可能需要增加驅動程式組態中的連線數。此範例顯示單一可用端點的相同計算。

```
20,000 CQL queries / (3 instances - 1 failure) / 1 private VPC endpoints / 500 CQL queries per second = 20 local connections
```

若要進一步了解如何使用共用 VPC 跨帳戶存取 Amazon Keyspaces，請參閱 [the section called “在共用 VPC 中設定跨帳戶存取”](#)。

## 如何在 Amazon Keyspaces 中監控連線

為了協助識別應用程式連線的端點數量，您可以記錄 `system.peers` 資料表中探索的對等數目。下列範例是 Java 程式碼的範例，其會在建立連線後列印對等數目。

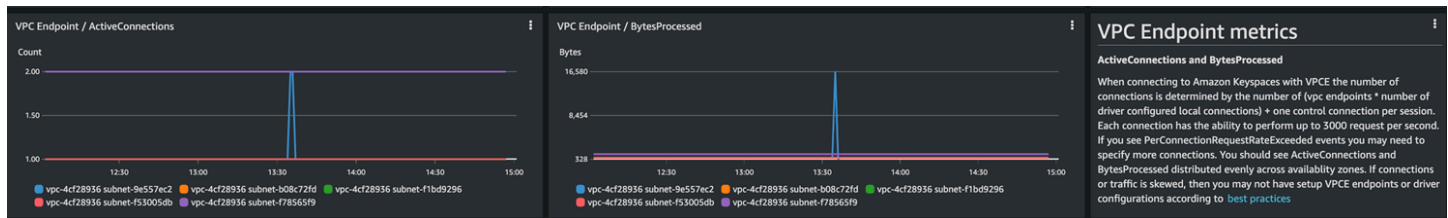
```
ResultSet result = session.execute(new SimpleStatement("SELECT * FROM system.peers"));

logger.info("number of Amazon Keyspaces endpoints:" + result.all().stream().count());
```

### 📘 Note

CQL 主控台或 AWS 主控台不會部署在 VPC 內，因此會使用公有端點。因此，從位於 VPCE 外部的應用程式執行 `system.peers` 查詢通常會產生 9 個對等。列印每個對等的 IP 地址也很有幫助。

您也可以設定 VPCE Amazon CloudWatch 指標，觀察使用 VPC 端點時的對等數量。在 CloudWatch 中，您可以查看與 VPC 端點建立的連線數。Cassandra 驅動程式會為每個端點建立連線，以傳送 CQL 查詢和控制連線，以收集系統資料表資訊。下圖顯示透過驅動程式設定中設定的 1 個連線連線至 Amazon Keyspaces 後的 VPC 端點 CloudWatch 指標。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 圖形監控連線數量，您可以在 Amazon Keyspaces CloudFormation 範本儲存庫的 GitHub 上部署此範本。<https://github.com/aws-samples/amazon-keyspaces-cloudwatch-cloudformation-templates>

## 如何處理 Amazon Keyspaces 中的連線錯誤

超過每個連線配額 3,000 個請求時，Amazon Keyspaces 會傳回 `PerConnectionRequestExceeded` 事件，且 Cassandra 驅動程式會收到 `WriteTimeout` 或 `ReadTimeout` 例外狀況。您應該在 Cassandra 重試政策或應用程式中以指數退避重試此例外狀況。您應該提供指數退避，以避免傳送其他請求。

預設重試政策會嘗試查詢計畫 `try next host` 中的。由於 Amazon Keyspaces 在連線至 VPC 端點時可能會有一到三個可用的端點，因此除了應用程式日誌中的 `WriteTimeout` 和 `ReadTimeout` 例外狀況 `NoHostAvailableException` 之外，您也可以看到。您可以使用 Amazon Keyspaces 提供的重試政策，在相同的端點上重試，但在不同的連線之間重試。

您可以在 Amazon Keyspaces Java 程式碼範例儲存庫中找到 GitHub 上 Java 的指數重試政策範例。<https://github.com/aws-samples/amazon-keyspaces-java-driver-helpers/blob/main/src/main/java/com/aws/ssa/keyspaces/retry/AmazonKeyspacesExponentialRetryPolicy.java> 您可以在 [Amazon Keyspaces 程式碼範例儲存庫](#) 的 Github 上找到其他語言範例。

## 資料建模最佳實務：設計資料模型的建議

使用 Amazon Keyspaces (適用於 Apache Cassandra) 時，有效的資料建模對於最佳化效能和降低成本至關重要。本主題涵蓋設計適合您應用程式資料存取模式之資料模型的重要考量和建議。

- 分割區索引鍵設計 – 分割區索引鍵在判斷資料如何在 Amazon Keyspaces 中的分割區之間分佈時扮演重要角色。選擇適當的分割區索引鍵可能會大幅影響查詢效能和輸送量成本。本節討論設計分割區索引鍵的策略，以促進跨分割區均勻分佈讀取和寫入活動。
- 重要考量事項：
  - 統一活動分佈 – 旨在統一所有分割區的讀取和寫入活動，以將輸送量成本降至最低，並有效利用高載容量。

- 存取模式 – 將您的分割區金鑰設計與應用程式的主要資料存取模式保持一致。
- 分割區大小 – 避免建立過大的分割區，因為這可能會影響效能並增加成本。

若要更輕鬆地視覺化和設計資料模型，您可以使用 [NoSQL Workbench](#)。

## 主題

- [如何在 Amazon Keyspaces 中有效使用分割區金鑰](#)

## 如何在 Amazon Keyspaces 中有效使用分割區金鑰

可唯一識別 Amazon Keyspaces 資料表中每一列的主索引鍵可以包含一或多個分割區索引鍵資料欄，這些索引鍵欄決定要存放資料的分割區，以及一或多個選用的叢集資料欄，定義如何在分割區中叢集和排序資料。

由於分割區索引鍵會建立資料存放於 中的分割區數量，以及資料在這些分割區間的分佈方式，因此您選擇分割區索引鍵的方式可能會對查詢的效能產生重大影響。一般而言，您應該針對磁碟上所有分割區的統一活動來設計應用程式。

將應用程式的讀取和寫入活動平均分散到所有分割區，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，如果您使用的是佈建容量模式，您可以判斷應用程式所需的存取模式，並預估每個資料表所需的總讀取容量單位 (RCU) 和寫入容量單位 (WCU)。只要針對特定分割區的流量不超過 3,000 RCUs 和 1,000 個 WCUs，Amazon Keyspaces 就會使用您佈建的輸送量來支援您的存取模式。

當分割區經歷持續的高讀取或寫入輸送量時，根據流量模式，Amazon Keyspaces 可能會自動將分割區分割成兩個新的分割區。每個新分割區都包含原始分割區列的子集，將輸送量平均分散到兩個分割區。

Amazon Keyspaces 透過提供高載容量，為您的每個分割區輸送量佈建提供額外的彈性，如需詳細資訊，請參閱 [the section called “使用高載容量”](#)。

## 主題

- [使用寫入碎片跨分割區平均分配工作負載](#)

## 使用寫入碎片跨分割區平均分配工作負載

在 Amazon Keyspaces 中跨分割區更好地分配寫入的一個方法是擴展空間。您可以數種不同的方式來執行此動作：您可以新增額外的分割區索引鍵資料欄，您可以將隨機數字寫入其中，以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。

### 使用複合分割區索引鍵和隨機值的碎片

跨分割區更平均地分配負載的一個策略是新增一個額外的分割區索引鍵資料欄，您可以將其寫入隨機數字。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，其具有代表日期的單一分割區索引鍵。

```
CREATE TABLE IF NOT EXISTS tracker.blogs (
 publish_date date,
 title text,
 description int,
 PRIMARY KEY (publish_date));
```

若要跨分割區更平均地分配此資料表，您可以包含存放shard隨機數字的額外分割區索引鍵資料欄。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 1 和 之間200為資料shard欄選擇隨機數字。這會透過 產生複合分割區索引鍵值，例如 (2020-07-09, 1)、(2020-07-09, 2)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輸送量。

不過，若要讀取指定日期的所有資料列，您必須查詢所有碎片的資料列，然後合併結果。例如，您會先發出分割區索引鍵值 的SELECT陳述式(2020-07-09, 1)。然後(2020-07-09, 2)，透過 為發出另一個SELECT陳述式，以此類推(2020-07-09, 200)。最後，您的應用程式必須合併所有這些SELECT陳述式的結果。



## 使用複合分割區索引鍵和計算值的碎片

隨機化的策略可大幅改善寫入輸送量。但是，讀取特定資料列並不容易，因為您不知道在資料列寫入時將哪個值寫入資料shard欄。若要更輕鬆地讀取個別資料列，您可以使用不同的策略。不是使用隨機數字在分割區之間分配資料列，而是使用您可以根據要查詢的內容計算的數字。

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。現在假設每一列都有可存取的資料title欄，而且除了日期之外，您通常還需要依標題尋找資料列。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，並使用它來填入資料shard欄。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。

簡單計算可能就足夠了，例如標題模數 200、+ 1 中字元的 UTF-8 程式碼點值乘積。然後，複合分割區索引鍵值會是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。您可以輕鬆為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割區索引鍵title值。

若要讀取指定日期的所有資料列，您仍然必須SELECT每個(2020-07-09, N)金鑰（其中 N 為 1–200），然後您的應用程式必須合併所有結果。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。

## 最佳化 Amazon Keyspaces 資料表的成本

本節涵蓋如何最佳化現有 Amazon Keyspaces 資料表成本的最佳實務。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。每個策略都提供可能對成本造成影響的概觀、如何尋找最佳化成本的機會，以及如何實作這些最佳實務以協助您節省成本的規範性指導。

### 主題

- [在資料表層級評估您的成本](#)
- [評估資料表的容量模式](#)
- [評估資料表的 Application Auto Scaling 設定](#)
- [識別未使用的資源，以最佳化 Amazon Keyspaces 中的成本](#)
- [評估您的資料表用量模式，以最佳化效能和成本](#)
- [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 管理主控台 可讓您查看依類型細分的成本，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鬆檢閱一個特定資料表的成本，因為 Cost Explorer 不會讓您依特定資料表的成本篩選或分組。您可以在資料表的監控索引標籤的 Amazon Keyspaces 主控台中檢視每個資料表的指標計費資料表大小 (位元組)。如果您需要更多每個資料表的成本相關資訊，本節說明如何使用[標記](#)在 Cost Explorer 中執行個別資料表的成本分析。

### 主題

- [如何檢視單一 Amazon Keyspaces 資料表的成本](#)
- [Cost Explorer 的預設檢視](#)
- [如何在 Cost Explorer 中使用和套用資料表標籤](#)

## 如何檢視單一 Amazon Keyspaces 資料表的成本

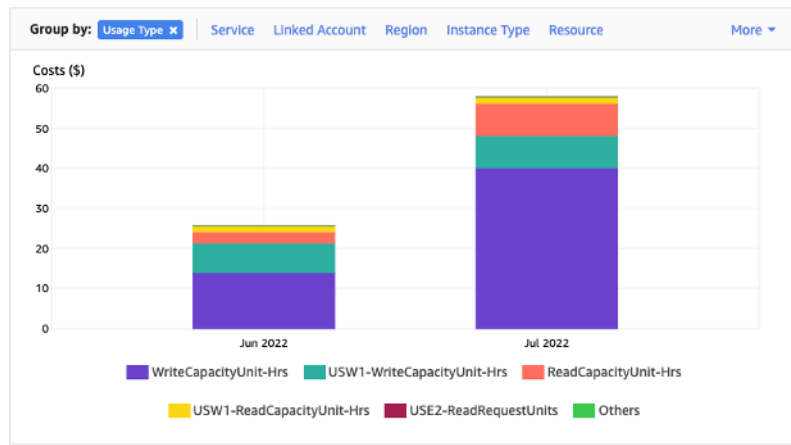
您可以在 主控台中查看 Amazon Keyspaces 資料表的基本資訊，包括主索引鍵結構描述、計費資料表大小和容量相關指標。您可以使用資料表的大小來計算資料表的每月儲存成本。例如，美國東部 (維吉尼亞北部) 每 GB 0 AWS 區域.25 美元。

如果資料表使用佈建容量模式，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，這些成本可能會變更，特別是如果您已使用 Amazon Keyspaces 自動擴展設定資料表時。

## Cost Explorer 的預設檢視

Cost Explorer 中的預設檢視提供圖表，顯示耗用資源的成本，例如輸送量和儲存。您可以選擇按期間分組這些成本，例如按月或按天的總計。儲存、讀取、寫入和其他類別的成本也可以分割和比較。





## 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。標籤是索引鍵/值組，可用於各種用途，例如識別屬於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，我們使用名為 MyTable 的資料表。

1. 設定一項具有「table\_name」索引鍵和「MyTable」值的標籤。
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 AWS CLI、或 AWS SDK 以程式設計方式設定中繼資料標籤。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 “建立成本分配報告”](#)。

## 評估資料表的容量模式

本節提供如何為 Amazon Keyspaces 資料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。在做出決策時，必須平衡考量這些因素。

### 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 資料表時，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 “設定讀取/寫入容量模式”](#)。

### 隨需容量模式

隨需容量模式旨在消除規劃或佈建 Amazon Keyspaces 資料表容量的需求。在此模式中，您的資料表會立即容納請求，而不需要擴展或縮減任何資源（高達資料表先前尖峰輸送量的兩倍）。

隨需資料表的計費方式是針對資料表計算實際請求數，因此您只需支付您使用的項目，而非已佈建的項目。

### 佈建容量模式

佈建容量模式是一種較傳統的模型，您可以在其中定義資料表可直接或在 Application Auto Scaling 的協助下可用於請求的容量。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。超過配置的容量也可能導致資料表拒絕請求，並減少應用程式使用者的體驗。

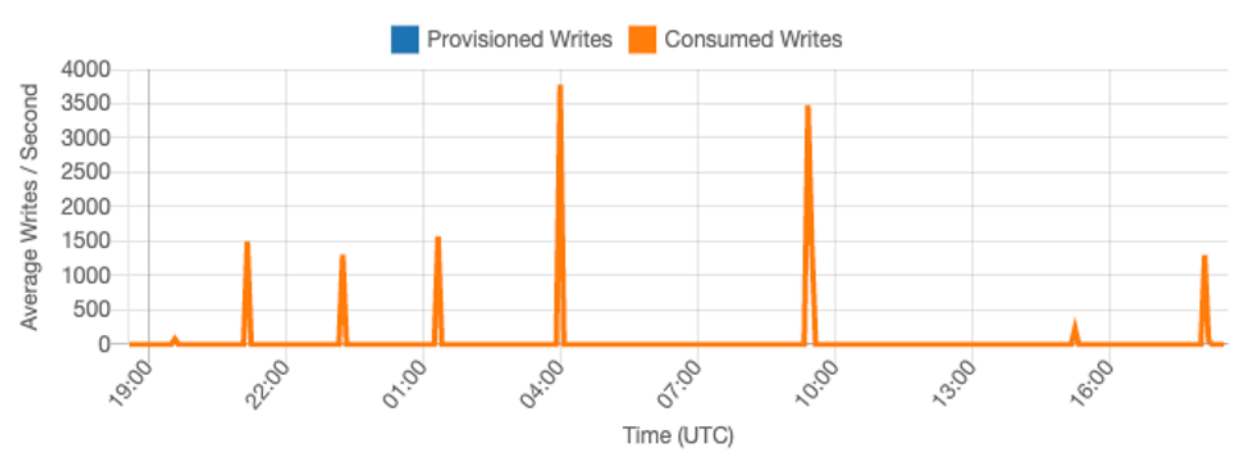
佈建容量模式需要在未過度佈建或佈建資料表下之間取得平衡，以實現低輸送量容量不足錯誤和最佳化成本。

## 何時選取隨需容量模式

當您有類似下圖所示的無法預測工作負載時，最佳化成本時，隨需模式是您的最佳選擇。

這些因素會導致這種類型的工作負載：

- 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)
- 下降到指定小時峰值的零或低於峰值的 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 來維持足夠容量讓資料表回應流量激增，可能會導致不良結果。資料表可能過度佈建且成本超過必要，或者資料表可能佈建不足，且請求導致不必要的低容量輸送量錯誤。在這種情況下，隨需資料表是更好的選擇。

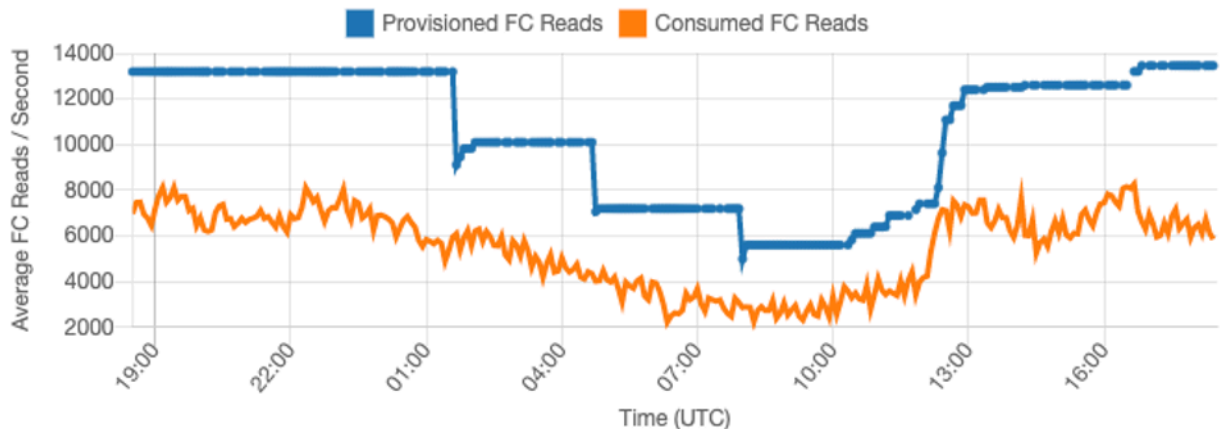
由於隨需資料表是依請求計費，因此您不需要再在資料表層級執行任何操作來最佳化成本。您應該定期評估隨需資料表，以確認工作負載仍具有上述特性。如果工作負載已穩定，請考慮變更為佈建模式，以維持成本最佳化。

## 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用量模式的工作負載，如下圖所示。

下列因素會導致可預測的工作負載：

- 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更為穩定，因此您可以設定相對接近資料表實際耗用容量的佈建容量。將佈建容量資料表進行成本最佳化，最終是讓佈建容量（藍線）盡可能接近耗用容量（橘色線），而不會增加資料表ThrottledRequests的事件。兩行之間的空間同時是浪費的容量，以及因為輸送容量不足錯誤而導致不良使用者體驗的保險。

Amazon Keyspaces 為佈建容量資料表提供 Application Auto Scaling，可代表您自動平衡。您可以追蹤一整天的耗用容量，並根據少量變數來設定資料表的佈建容量。

### 容量單位下限

您可以設定資料表的最小容量，以限制輸送量容量不足錯誤的發生，但不會降低資料表的成本。如果您的資料表具有低用量期間，然後突然高用量暴增，則設定最小值可防止 Application Auto Scaling 設定太低的資料表容量。

### 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。考慮對開發或測試資料表套用最大值，其中不需要大規模負載測試。您可以為任何資料表設定最大值，但在生產環境中使用時，請務必定期根據資料表基準評估此設定，以防止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資料表過度佈建的程度，增加成本，但可降低輸送量容量不足錯誤的風險。設定較高的百分比值會減少資料表過度佈建的程度，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決定時，有一些其他因素值得考慮。

在兩個資料表模式之間進行決策時，請考慮此額外折扣對資料表成本的影響程度。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佈建容量資料表上執行也更具成本效益。

### 改善工作負載的可預測性

在某些情況下，工作負載可能同時具有可預測和不可預測的模式。雖然隨需資料表可以輕鬆支援，但如果工作負載中無法預測的模式可以改善，成本可能會更低。

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以超過資料表的基準容量，達到如果執行就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：

- 如果批次在排程時間發生，您可以在執行之前排程增加應用程式自動擴展容量。

- 如果批次隨機發生，請考慮嘗試延長執行所需的時間，而不是盡快執行。
- 將漸進測試期間新增至匯入，匯入速度從小開始，但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。

## 評估資料表的 Application Auto Scaling 設定

本節概述如何評估 Amazon Keyspaces 資料表的 Application Auto Scaling 設定。[Amazon Keyspaces Application Auto Scaling](#) 是一項功能，可根據您的應用程式流量和目標使用率指標來管理資料表輸送量。這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資料表使用率，並將其與目標使用率值進行比較：TargetValue。如果該增加或減少配置的容量，它會通知您。

### 主題

- [了解 Application Auto Scaling 設定](#)
- [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](#)

## 了解 Application Auto Scaling 設定

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。這可讓您根據歷史應用程式用量正確定義值，以用來觸發 Application Auto Scaling 政策。使用率目標是在套用 Application Auto Scaling 規則前一段時間內需要達到的總容量百分比。

當您設定高使用率目標（約 90% 的目標）時，這表示您的流量需要高於 90% 一段時間，才能啟用 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。

當您設定極低的使用率（低於 50% 的目標）時，表示您的應用程式需要達到佈建容量的 50%，才能觸發 Application Auto Scaling 政策。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%)

您可以使用 AWS CLI 或 AWS 管理主控台 來監控和識別 Amazon Keyspaces 資源中 Application Auto Scaling 政策TargetValues的。

**Note**

當您在佈建容量模式中使用多區域資料表搭配 Amazon Keyspaces 自動擴展時，請務必使用 Amazon Keyspaces API 操作來設定自動擴展。Amazon Keyspaces 代您呼叫的基礎 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 “檢視多區域資料表的佈建容量和自動擴展設定”](#)。

## 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 資源的完整 Application Auto Scaling 政策清單。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，僅傳回特定資料表的自動擴展政策

```
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。您需要確保目標值大於 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:us-
east-1:111122223333:scalingPolicy:<uuid>:resource/keyspaces/table/table-name-
scaling-policy",
 "PolicyName": "${full-table-name}",
 "ServiceNamespace": "cassandra",
 "ResourceId": "keyspace/keyspace-name/table/table-name",
 "ScalableDimension": "cassandra:table:WriteCapacityUnits",
 "PolicyType": "TargetTrackingScaling",
 }
]
}
```

```

 "TargetTrackingScalingPolicyConfiguration": {
 "TargetValue": 70.0,
 "PredefinedMetricSpecification": {
 "PredefinedMetricType": "KeyspacesWriteCapacityUtilization"
 }
 },
 "Alarms": [
 ...
],
 "CreationTime": "2022-03-04T16:23:48.641000+10:00"
 },
 {
 "PolicyARN": "arn:aws:autoscaling:us-
east-1:11112223333:scalingPolicy:<uuid>:resource/keyspaces/table/table-name-
scaling-policy",
 "PolicyName": "${<full-table-name>}",
 "ServiceNamespace": "cassandra",
 "ResourceId": "keyspace/keyspace-name/table/table-name",
 "ScalableDimension": "cassandra:table:ReadCapacityUnits",
 "PolicyType": "TargetTrackingScaling",
 "TargetTrackingScalingPolicyConfiguration": {
 "TargetValue": 70.0,
 "PredefinedMetricSpecification": {
 "PredefinedMetricType": "KeyspacesReadCapacityUtilization"
 }
 },
 "Alarms": [
 ...
],
 "CreationTime": "2022-03-04T16:23:47.820000+10:00"
 }
]
}

```

## AWS 管理主控台

1. 登入 AWS 管理主控台，並在 [入門 AWS 管理主控台](#) 中導覽至 CloudWatch 服務頁面。AWS 區域 視需要選取適當的。
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. 在容量索引標籤的資料表詳細資訊頁面上，檢閱資料表的 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。由於某些使用者會在上午 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，以在尖峰時間達到所需的容量。

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[排程應用程式自動擴展](#)，其中您的資料表仍然可以設定 Application Auto Scaling 規則，但具有較不積極的目標使用率，只會以您所需的特定間隔佈建額外的容量。

您可以使用 AWS CLI 來執行下列步驟，以建立排程的自動擴展規則，該規則會根據一天中的時間和星期幾來執行。

1. 向將 Amazon Keyspaces 資料表註冊為可擴展的目標 Application Auto Scaling。可擴展的目標是 Application Auto Scaling 可橫向擴充和縮減的資源。

```
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案例：一個向上擴展，另一個向下擴展。擴展排程動作的第一個規則會顯示在下列範例中。



```
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"
```

此範例中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：

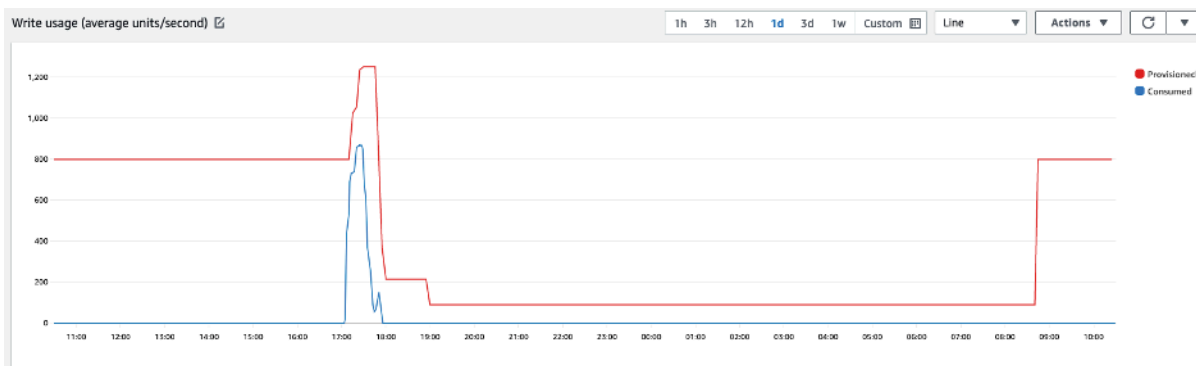
```
{
 "ScheduledActions": [
 {
 "ScheduledActionName": "my-5-8-scheduled-down-action",
 "ScheduledActionARN": "arn:aws:autoscaling:us-
east-1:111122223333:scheduledAction:<uuid>:resource/keyspaces/table/table-
name:scheduledActionName/my-5-8-scheduled-down-action",
 "ServiceNamespace": "cassandra",
 "Schedule": "cron(15 17 ? * MON-FRI *)",
 "Timezone": "Australia/Brisbane",
 "ResourceId": "keyspace/keyspace-name/table/table-name",
 "ScalableDimension": "cassandra:table:WriteCapacityUnits",
 "ScalableTargetAction": {
 "MinCapacity": 90,
```

```

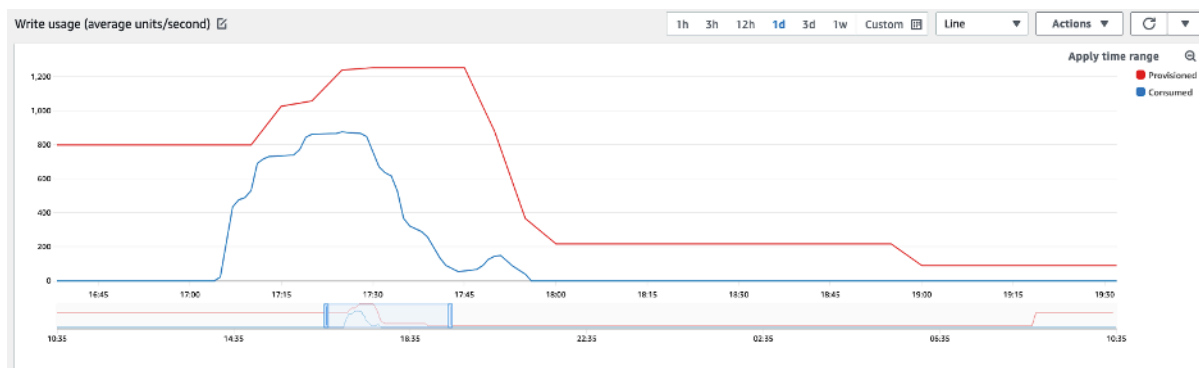
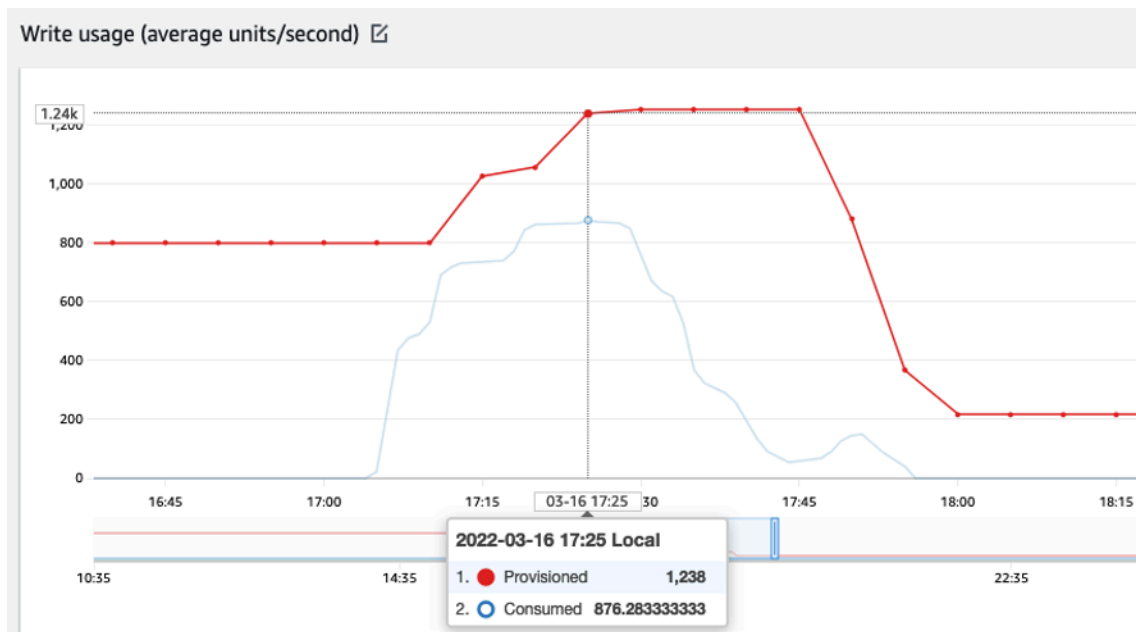
 "MaxCapacity": 1500
 },
 "CreationTime": "2022-03-15T17:30:25.100000+10:00"
},
{
 "ScheduledActionName": "my-8-5-scheduled-action",
 "ScheduledActionARN": "arn:aws:autoscaling:us-east-1:111122223333:scheduledAction:<uuid>:resource/keyspaces/table/table-name:scheduledActionName/my-8-5-scheduled-action",
 "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，自動擴展規則如何仍然套用，且輸送量不會降低。



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。排程的自動擴展動作會影響最大值和最小值，而設定這些值是您的責任。



## 如何處理具有未知模式的尖峰工作負載

在此案例中，應用程式使用非常低的使用率目標，因為您尚不知道應用程式模式，而且您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的讀取和寫入輸送量，因為 Amazon Keyspaces 會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。

考慮開發自訂應用程式自動調整規模邏輯，以對您可以增加資料表容量並 TargetValues 根據您的特定需求做出反應的事件做出反應。您可以受益於 `Amazon EventBridge` 並使用 `Lambda` 和 `Step Functions` 等 AWS 服務組合來回應您的特定應用程式需求。

## 識別未使用的資源，以最佳化 Amazon Keyspaces 中的成本

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演進，您應該確保沒有未使用的資源，並造成不必要的 Amazon Keyspaces 成本。以下所述的程序使用 Amazon CloudWatch 指標來識別未使用的資源，並採取行動以降低成本。

您可以使用 CloudWatch 監控 Amazon Keyspaces，其會收集來自 Amazon Keyspaces 的原始資料，並將其處理為可讀且近乎即時的指標。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 指標資料會自動傳送至 CloudWatch。如需詳細資訊，請參閱《Amazon CloudWatch 使用者指南》中的 [什麼是 Amazon CloudWatch？](#) 以及 [指標保留](#)。

### 主題

- [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](#)

## 如何識別未使用資源

若要識別未使用的資料表，您可以查看 30 天內的下列 CloudWatch 指標，以了解特定資料表上是否有任何作用中的讀取或寫入：

### 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。您可以擷取資料表的總耗用讀取容量。

### 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 是一種監控和可觀測性服務，提供可用於識別未使用資源的 Amazon Keyspaces 資料表指標。您可以透過 AWS 管理主控台 和 AWS Command Line Interface 來檢視 CloudWatch 指標。

### AWS Command Line Interface

若要透過 檢視資料表指標 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

如果資料表名稱在帳戶中不是唯一的，您也必須指定金鑰空間的名稱。

```
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 管理主控台

下列步驟可讓您透過 評估資源使用率 AWS 管理主控台。

1. 登入 AWS 管理主控台 並導覽至位於 <https://console.aws.amazon.com/cloudwatch/> 的 CloudWatch 服務頁面。如有必要，請在主控台的 AWS 區域 右上角選取適當的。
2. 在左側導覽列上，找到指標區段，然後選擇所有指標。
3. 上述動作會開啟具有兩個面板的儀表板。在頂端面板中，您可以看到目前繪製的指標。在底部，您可以選取可用於繪製圖形的指標。在底部面板中選擇 Amazon Keyspaces。
4. 在 Amazon Keyspaces 指標選擇面板中，選擇資料表指標類別，以顯示目前區域中資料表的指標。
5. 向下捲動選單來識別資料表名稱，然後 ConsumedWriteCapacityUnits 為您的資料表選擇指標 ConsumedReadCapacityUnits 和。
6. 選擇圖形化指標 (2) 索引標籤，並將統計資料欄調整為總和。
7. 若要避免錯誤地將資料表識別為未使用，請評估較長期間內的資料表指標。在圖形面板頂端，選擇適當的時間範圍，例如 1 個月，以評估您的資料表。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。讀取和寫入的平面圖形為 0，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，您可以考慮刪除資料表。

## 容量模式

Amazon Keyspaces 會收取讀取、寫入和存放 Amazon Keyspaces 資料表中資料的費用。

Amazon Keyspaces [有兩種容量模式](#)，隨附處理資料表上的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輸送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，收取讀取請求單位和寫入請求單位的費用。如果您的資料表上沒有活動，您不需要支付輸送量費用，但仍需支付儲存費用。

## 刪除資料表

如果您發現未使用的資料表並想要將其刪除，請考慮先進行備份或匯出資料。

透過 取得的備份 AWS Backup 可以利用冷儲存分層，進一步降低成本。如需如何使用生命週期將備份移至冷儲存體的資訊，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 管理主控台 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Point-in-time復原，可提供 35 天的連續備份，協助您防止意外寫入或刪除。PITR 備份有相關聯的成本。

請參閱 的文件[the section called “使用point-in-time復原進行備份和還原”](#)，以判斷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，以最佳化效能和成本

本節提供如何評估您是否有效率地使用 Amazon Keyspaces 資料表的概觀。有些使用模式不適用於 Amazon Keyspaces，而且從效能和成本角度都允許最佳化的空間。

### 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。最終一致讀取會以 0.5 RCU 收費，最多 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。高度一致的讀取會以 1 個 RCU 收費，最多 4 KB 的資料，基本上是讀取成本的兩倍。Amazon Keyspaces 可讓您靈活地在同一資料表上使用兩個一致性模型。

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)

[存留時間 \(TTL\)](#) 可協助您簡化應用程式邏輯，並透過自動從資料表過期資料來最佳化儲存體的價格。您不再需要的資料會根據您設定的存留時間值，自動從資料表中刪除。

## 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 資料表上是否有適當大小的佈建的概觀。隨著工作負載的演進，您應該適當地修改操作程序，尤其是當您的 Amazon Keyspaces 資料表在佈建模式中設定，而且您有過度佈建或佈建不足資料表的風險時。

本節所述的程序需要從支援生產應用程式的 Amazon Keyspaces 資料表中擷取的統計資訊。若要了解您的應用程式行為，您應該定義足夠重要的時段，以擷取應用程式的資料季節性。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。

在評估容量時，對於 Amazon Keyspaces 資料表，您可以獨立設定讀取容量單位 (RCUs) 和寫入容量單位 (WCU)。

### 主題

- [如何從 Amazon Keyspaces 資料表擷取取用指標](#)
- [如何識別佈建不足的 Amazon Keyspaces 資料表](#)
- [如何識別過度佈建的 Amazon Keyspaces 資料表](#)

## 如何從 Amazon Keyspaces 資料表擷取取用指標

若要評估資料表容量，請監控下列 CloudWatch 指標，然後選取適當的維度以擷取資料表資訊：

| 讀取容量單位                    | 寫入容量單位                     |
|---------------------------|----------------------------|
| ConsumedReadCapacityUnits | ConsumedWriteCapacityUnits |

| 讀取容量單位                       | 寫入容量單位                        |
|------------------------------|-------------------------------|
| ProvisionedReadCapacityUnits | ProvisionedWriteCapacityUnits |
| ReadThrottleEvents           | WriteThrottleEvents           |

您可以透過 AWS CLI 或 執行此操作 AWS 管理主控台。

## AWS CLI

擷取資料表取用指標之前，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`。這些檔案代表資料表的計算。您需要更新一些欄位，如下表所示，以符合您的環境。

### Note

如果資料表名稱在帳戶中不是唯一的，您也必須指定金鑰空間的名稱。

| 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指定日期範圍期間內佈建和取用的 WCU 數量。它也會產生可用於分析的使用率百分比。`write-calc.json` 檔案的完整內容看起來應該如下例所示。

```
{
 "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
}
```

讀取計算檔案使用類似的指標。此檔案會擷取在指定日期範圍的期間內佈建和取用的 RCUs 數量。在此範例中，`read-calc.json` 檔案的內容看起來應該類似。

```
{
 "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 RCU",
 "ReturnData": false
},
{
 "Id": "utilizationPercentage",
 "Expression": "100*(m1/provisionedRCU)",
 "Label": "Utilization Percentage",
 "ReturnData": true
}
],
"StartTime": "<start-time>",
"EndTime": "<end-time>",
"ScanBy": "TimestampDescending",
"MaxDatapoints": 24
}

```

建立檔案之後，您就可以開始擷取使用率資料。

- 若要擷取寫入使用率資料，請發出下列命令：

```
aws cloudwatch get-metric-data --cli-input-json file://write-calc.json
```

- 若要擷取讀取使用率資料，請發出下列命令：

```
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。您的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。在以下範例中可能看起來像。

```
{
 "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,
 40.947256944444445,
 25.618194444444444,
 0.0
],
 "StatusCode": "Complete"
 }
],
 "Messages": []
}
```

#### Note

如果您指定短期間和長時間範圍，您可能需要修改值，該MaxDatapoints值預設為指令碼中的 24。這代表每小時產生一個資料點，每天 24 個資料點。

## AWS 管理主控台

1. 登入 AWS 管理主控台，並在 [入門 AWS 管理主控台](#) 中導覽至 CloudWatch 服務頁面。AWS 區域 視需要選取適當的。

2. 找到左側導覽列上的指標區段，然後選擇所有指標。
3. 這會開啟具有兩個面板的儀表板。頂端面板會顯示圖形，而底部面板具有您要繪製圖形的指標。選擇 Amazon Keyspaces 面板。
4. 從子面板中選擇資料表指標類別。這會顯示目前中的資料表 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，但您也可以使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。根據預設，CloudWatch 會為圖形選擇統計函數平均。
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。

第一次選取百分比函數。

第二次選取百分比函數。

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 區段中使用的名稱相符。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名為 ANOMALY\_DETECTION\_BAND 的指標。對於此程序的範圍，您可以透過移除新產生的 ad1 指標上的核取方塊來忽略這一點。
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 Expression1 標籤，並將值更新為 m1，並將標籤更新為已使用 WCUs。

 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。此步驟也可能產生另一個 ANOMALY\_DETECTION\_BAND 指標，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：一個表示資料表上的佈建 WCUs 另一個表示耗用的 WCUs。
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  $100 * (m1 / \text{provisionedWCU})$ 。
14. 從 utilizationPercentage 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，但您可以視需要修改間隔。

您取得的結果取決於工作負載的實際資料。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 提供[高載容量](#)，但一旦高載容量用盡，超過 100% 的任何內容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 資料表

對於大多數工作負載，資料表在持續耗用其佈建容量的 80% 以上時，會被視為佈建不足。

[爆量容量](#)是一項 Amazon Keyspaces 功能，可讓客戶暫時使用比原始佈建更多的 RCUs/WCUs（超過為資料表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。此高載容量有限，如需詳細資訊，請參閱 [the section called “使用高載容量”](#)。一旦未使用的 RCUs 和 WCUs 用盡，如果您嘗試使用比佈建更多的容量，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。

在某些情況下，當您每天或每週分析資料時，資料會扭曲。例如，如果季節性應用程式在工作時間內（但在工作時間之外降到幾乎零）的使用量遽增，您可以受益於[排程應用程式自動擴展](#)，您可以在其中指定一天的時數（和一週的天數）來增加佈建的容量，以及何時減少它。如果您的季節性較不明顯，則您也可以受益於 [Amazon Keyspaces 資料表自動調整規模](#) 組態，而不是為了涵蓋忙碌時間而瞄準更高的容量。



## 如何識別過度佈建的 Amazon Keyspaces 資料表

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。

當您的資料表包含數個低用量間隔時，您可以透過排程 Application Auto Scaling 或僅為根據使用率的資料表設定預設 Application Auto Scaling 政策，從使用 Application Auto Scaling 政策中獲益。

如果您的工作負載在間隔中具有低使用率和高限流率 ( $\text{Max(ThrottleEvents)}/\text{Min(ThrottleEvents)}$ )，則當您的工作負載非常尖峰，其中流量在特定日期（或一天中的時間）顯著增加，但在其他情況下一直很低時，可能會發生這種情況。在這些情況下，使用[排定的 Application Auto Scaling](#)可能會有所幫助。

# Amazon Keyspaces 疑難排解 (適用於 Apache Cassandra)

本指南涵蓋使用 Amazon Keyspaces (適用於 Apache Cassandra) 時各種案例的疑難排解步驟。它包含解決一般錯誤、連線問題、容量管理問題和資料定義語言 (DDL) 錯誤的資訊。

- 一般錯誤
  - 疑難排解最上層例外狀況 `NoHostAvailableException`，例如 `NoNodeAvailableException`、和 `AllNodesFailedException`。
  - 隔離基礎錯誤與 Java 驅動程式例外狀況。
  - 實作重試政策和正確設定連線。
- 連線問題
  - 解決使用 `cqlsh` 或 Apache Cassandra 用戶端驅動程式連線至 Amazon Keyspaces 端點時的錯誤。
  - 故障診斷 VPC 端點連線、Cassandra-stress 連線和 IAM 組態錯誤。
  - 在資料匯入期間處理連線中斷。
- 容量管理錯誤
  - 識別和解決與資料表、分割區和連線相關的容量不足錯誤。
  - 監控 Amazon CloudWatch Logs 中的相關 Amazon Keyspaces 指標。
  - 最佳化連線和輸送量以提升效能。
- 資料定義語言 (DDL) 錯誤
  - 建立、存取或還原金鑰空間和資料表時發生錯誤的故障診斷。
  - 處理與自訂存留時間 (TTL) 設定、資料欄限制和範圍刪除相關的失敗。
  - 大量刪除工作負載的考量事項。

如需 IAM 存取特定的疑難排解指引，請參閱 [the section called “疑難排解”](#)。如需安全最佳實務的詳細資訊，請參閱 [the section called “安全最佳實務”](#)。

## 主題

- [對 Amazon Keyspaces 中的一般錯誤進行故障診斷](#)
- [對 Amazon Keyspaces 中的連線錯誤進行故障診斷](#)
- [對 Amazon Keyspaces 中的容量管理錯誤進行故障診斷](#)
- [對 Amazon Keyspaces 中的資料定義語言錯誤進行故障診斷](#)

# 對 Amazon Keyspaces 中的一般錯誤進行故障診斷

取得一般錯誤？以下是一些常見問題，以及如何解決這些問題。

## 一般錯誤

您會收到下列其中一個可能因許多不同原因而發生的頂層例外狀況。

- `NoNodeAvailableException`
- `NoHostAvailableException`
- `AllNodesFailedException`

這些例外狀況是由用戶端驅動程式產生，而且可能會在您建立控制連線或執行`read/write/prepare/執行/批次請求`時發生。

當您建立控制連線時發生錯誤時，表示應用程式中指定的所有聯絡點都無法連線。當執行`read/write/prepare/執行查詢`時發生錯誤時，表示該請求的所有重試都已用盡。當您使用預設重試政策時，會在不同的節點上嘗試每次重試。

## 如何隔離基礎錯誤與最上層 Java 驅動程式例外狀況

這些一般錯誤可能是由連線問題或執行`read/write/prepare/執行操作`時造成。必須在分散式系統中預期暫時性失敗，並且應該透過重試請求來處理。遇到連線錯誤時，Java 驅動程式不會自動重試，因此建議在應用程式中建立驅動程式連線時實作重試政策。如需連線最佳實務的詳細概觀，請參閱 [the section called “連線”](#)。

根據預設，Java 驅動程式會將所有請求`idempotence`設為 `false`，這表示 Java 驅動程式不會自動重試失敗的`read/write/prepare`請求。若要`idempotence`將設定為 `true` 並告知驅動程式重試失敗的請求，您可以透過幾種不同的方式執行此操作。以下範例說明如何以程式設計方式為 Java 應用程式中的單一請求設定冪等性。

```
Statement s = new SimpleStatement("SELECT * FROM my_table WHERE id = 1");
s.setIdempotent(true);
```

或者，您可以透過程式設計方式設定整個 Java 應用程式的預設冪等性，如下列範例所示。

```
// 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政策。在此情況下，應用程式需要擷取 `NoNodeAvailableException` 並重試請求。我們建議從 10 毫秒開始，以指數退避進行 10 次重試，並對所有重試進行最多 100 毫秒的總時間為 1 秒。

另一個選項是在建立 [Github](#) 上可用的 Java 驅動程式連線時套用 Amazon Keyspaces 指數重試政策。

確認您在使用預設重試政策時已建立與多個節點的連線。您可以在 Amazon Keyspaces 中使用下列查詢來執行此操作。

```
SELECT * FROM system.peers;
```

如果此查詢的回應是空的，這表示您正在使用 Amazon Keyspaces 的單一節點。如果您使用的是預設重試政策，則不會重試，因為預設重試一律發生在不同的節點上。若要進一步了解如何透過 VPC 端點建立連線，請參閱 [the section called “VPC 端點連線”](#)。

如需 step-by-step 教學課程，說明如何使用 Datastax 4.x Cassandra 驅動程式建立與 Amazon Keyspaces 的連線，請參閱 [the section called “Java 4.x 的身分驗證外掛程式”](#)。

## 對 Amazon Keyspaces 中的連線錯誤進行故障診斷

連線時發生問題？以下是一些常見問題，以及如何解決這些問題。

### 連線至 Amazon Keyspaces 端點時發生錯誤

失敗的連線和連線錯誤可能會導致不同的錯誤訊息。下一節涵蓋最常見的案例。

#### 主題

- [我無法使用 cqlsh 連線到 Amazon Keyspaces](#)
- [我無法使用 Cassandra 用戶端驅動程式連線至 Amazon Keyspaces](#)

### 我無法使用 cqlsh 連線到 Amazon Keyspaces

您嘗試使用 cqlsh 連線至 Amazon Keyspaces 端點，但連線與的連線失敗 **Connection error**。

如果您嘗試連線至 Amazon Keyspaces 資料表，但尚未正確設定 `cqlsh`，則連線會失敗。下節提供在您嘗試使用 `cqlsh` 建立連線時，導致連線錯誤的最常見組態問題範例。

#### Note

如果您嘗試從 VPC 連線至 Amazon Keyspaces，則需要額外的許可。若要使用 VPC 端點成功設定連線，請遵循中的步驟[the section called “與 VPC 端點連線”](#)。

您嘗試使用 `cqlsh` 連線至 Amazon Keyspaces，但發生連線 **timed out** 錯誤。

如果您未提供正確的連接埠，則可能會發生這種情況，這會導致下列錯誤。

```
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")})
```

若要解決此問題，請確認您使用連接埠 9142 進行連線。

您嘗試使用 `cqlsh` 連線至 Amazon Keyspaces，但發生錯誤 **Name or service not known**。

如果您使用的端點拼寫錯誤或不存在，則可能會發生這種情況。在下列範例中，端點的名稱拼錯。

```
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
```

若要在使用公有端點進行連線時解決此問題，請從 選取可用的端點[the section called “服務端點”](#)，並確認端點的名稱沒有任何錯誤。如果您使用 VPC 端點進行連線，請確認 `cqlsh` 組態中的 VPC 端點資訊正確無誤。

您嘗試使用 `cqlsh` 連線至 Amazon Keyspaces，但收到 **OperationTimedOut** 錯誤。

Amazon Keyspaces 需要為連線啟用 SSL，以確保強大的安全性。如果您收到下列錯誤，則 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',)})
#
```

若要解決此問題，請將下列旗標新增至 cqlsh 連線命令。

```
--ssl
```

您嘗試使用 cqlsh 連線至 Amazon Keyspaces，並收到 **SSL transport factory requires a valid certfile to be specified** 錯誤。

在此情況下，缺少 SSL/TLS 憑證的路徑，這會導致下列錯誤。

```
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 certfile。如需詳細資訊，請參閱[the section called “如何手動設定 TLS 的 cqlsh 連線”](#)。

```
certfile = path_to_file/keysaces-bundle.pem
```

您嘗試使用 cqlsh 連線至 Amazon Keyspaces，但收到 **No such file or directory** 錯誤。

如果電腦上憑證檔案的路徑錯誤，這可能會導致以下錯誤。

```
cat .cassandra/cqlshrc
[connection]
port = 9142
factory = cqlshlib.ssl.ssl_transport_factory

[ssl]
validate = true
certfile = /root/wrong_path/keysaces-bundle.pem
#

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 certfile 的路徑正確無誤。如需詳細資訊，請參閱[the section called “如何手動設定 TLS 的 cqlsh 連線”](#)。

您嘗試使用 cqlsh 連線至 Amazon Keyspaces，但收到 **[X509] PEM lib** 錯誤。

如果 SSL/TLS 憑證 pem 檔案無效，則可能會發生這種情況，這會導致下列錯誤。

```
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，請確定您已下載必要的數位憑證。如需詳細資訊，請參閱[the section called “如何手動設定 TLS 的 cqlsh 連線”](#)。

您嘗試使用 cqlsh 連線至 Amazon Keyspaces，但收到 **unknown SSL** 錯誤。

如果 SSL/TLS 憑證 pem 檔案是空的，則可能會發生這種情況，這會導致下列錯誤。

```
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，請確定您已下載必要的數位憑證。您可以使用下列主題 中的步驟來確認這一點[the section called “如何手動設定 TLS 的cqlsh連線”](#)。

您嘗試使用 cqlsh 連線至 Amazon Keyspaces，但收到 **SSL: CERTIFICATE\_VERIFY\_FAILED** 錯誤。

如果無法驗證 SSL/TLS 憑證檔案，則可能會發生這種情況，這會導致下列錯誤。

```
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，請確定您已下載必要的數位憑證。您可以使用下列主題 中的步驟來確認這一點[the section called “如何手動設定 TLS 的cqlsh連線”](#)。

您嘗試使用 cqlsh 連線至 Amazon Keyspaces，但收到 **Last error: timed out** 錯誤。

如果您未在 Amazon EC2 安全群組中為 Amazon Keyspaces 設定傳出規則，這可能會導致以下錯誤。

```
cqlsh cassandra.us-east-1.amazonaws.com 9142 -u "USERNAME" -p "PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.206': 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，而非 cqlsh，您可以嘗試使用 連線到您的金鑰空間 AWS CLI，例如使用下列命令。

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

如果此命令也逾時，表示 Amazon EC2 執行個體未正確設定。

若要確認您有足夠的許可來存取 Amazon Keyspaces，您可以使用 AWS CloudShell 來與 連線 cqlsh。如果連線已建立，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至 Amazon Keyspaces 的傳出規則。如果不是這種情況，您需要為 EC2 執行個體建立新的安全群組，並新增允許傳出流量到 Amazon Keyspaces 資源的規則。若要更新傳出規則以允許流量到 Amazon Keyspaces，請從類型下拉式選單中選擇 CQLSH/CASSANDRA。

使用傳出流量規則建立新的安全群組之後，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，然後選擇變更安全群組。使用傳出規則新增安全群組，但請確定預設群組也保持可用。



如需如何檢視和編輯 EC2 傳出規則的詳細資訊，請參閱 [《Amazon EC2 使用者指南》中的將規則新增至安全群組](#)。

您嘗試使用 cqlsh 連線至 Amazon Keyspaces，但收到 **Unauthorized** 錯誤。

如果您在 IAM 使用者政策中缺少 Amazon Keyspaces 許可，這可能會導致以下錯誤。

```
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，請確定 IAM 使用者 `testuser-at-12345678910` 具有存取 Amazon Keyspaces 的許可。如需授予 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 連線至 Amazon Keyspaces，但收到 **Bad credentials** 錯誤。

如果使用者名稱或密碼錯誤，則可能會發生這種情況，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', {'3.234.248.248':
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"',))
#
```

若要解決此問題，請確認程式碼中的 **USERNAME** 和 **PASSWORD** 與您產生 [服務特定登入](#) 資料時取得的使用者名稱和密碼相符。

#### 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，並在聯絡時包含詳細輸出 支援。

## 我無法使用 Cassandra 用戶端驅動程式連線至 Amazon Keyspaces

下列各節顯示與 Cassandra 用戶端驅動程式連線時最常見的錯誤。

您嘗試使用 DataStax Java 驅動程式連線至 Amazon Keyspaces 資料表，但收到 **NodeUnavailableException** 錯誤。

如果嘗試請求的連線中斷，則會導致下列錯誤。

```
[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，如果訊號值較高，請將其降至 30 秒。

```
advanced.heartbeat.interval = 30 seconds
```

然後尋找相關聯的逾時，並確保值設定為至少 5 秒。

```
advanced.connection.init-query-timeout = 5 seconds
```

您嘗試使用驅動程式和 SigV4 外掛程式連線至 Amazon Keyspaces 資料表，但收到 **AttributeError** 錯誤。

如果登入資料未正確設定，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，傳遞與您的 IAM 使用者或角色相關聯的登入資料。SigV4 外掛程式需要下列登入資料。

- **AWS\_ACCESS\_KEY\_ID** – 指定與 IAM 使用者或角色相關聯的 AWS 存取金鑰。
- **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」。

若要進一步了解存取金鑰和 SigV4 外掛程式，請參閱 [the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

您嘗試使用驅動程式連線至 Amazon Keyspaces 資料表，但收到 **PartialCredentialsError** 錯誤。

如果遺失 `AWS_SECRET_ACCESS_KEY`，可能會導致下列錯誤。

```
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，請確認您使用 SigV4 外掛程式 `AWS_SECRET_ACCESS_KEY` 時同時傳遞 `AWS_ACCESS_KEY_ID` 和 `AWS_SECRET_ACCESS_KEY`。若要進一步了解存取金鑰和 SigV4 外掛程式，請參閱 [the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

您嘗試使用驅動程式連線至 Amazon Keyspaces 資料表，但收到 **Invalid signature** 錯誤。

如果簽章所需的任何元件錯誤或未正確定義工作階段，則可能會發生這種情況。

- `AWS_ACCESS_KEY_ID`
- `AWS_SECRET_ACCESS_KEY`
- `AWS_DEFAULT_REGION`

下列錯誤是無效的存取金鑰範例。

```
cassandra.cluster.NoHostAvailable: ('Unable to connect to any servers',
{'11.234.11.234:9142':
AuthenticationFailed('Failed to authenticate to 11.234.11.234:9142: Error from server:
code=0100
[Bad credentials] message="Authentication failure: Invalid signature"')})
```

若要解決此問題，請確認存取金鑰和 AWS 區域 已正確設定供 SigV4 外掛程式存取 Amazon Keyspaces。若要進一步了解存取金鑰和 SigV4 外掛程式，請參閱 [the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

我的 VPC 端點連線無法正常運作

您嘗試使用 VPC 端點連線至 Amazon Keyspaces，但收到字符映射錯誤或輸送量低。

如果 VPC 端點連線未正確設定，則可能會發生這種情況。

若要解決這些問題，請確認下列組態詳細資訊。若要遵循step-by-step教學，了解如何透過 Amazon Keyspaces 的介面 VPC 端點設定連線，請參閱 [the section called “與 VPC 端點連線”](#)。

1. 確認用於連線至 Amazon Keyspaces 的 IAM 實體具有使用者資料表的讀取/寫入存取權，以及對系統資料表的讀取存取權，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. 確認用於連線至 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並AmazonKeyspacesFullAccess包含必要許可，讓 Amazon Keyspaces 存取 Amazon EC2 執行個體，以讀取可用介面 VPC 端點的相關資訊。

如需 VPC 端點的詳細資訊，請參閱 [the section called “使用 Amazon Keyspaces 的介面 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，如本範例所示。

```
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 VPC 端點已正確設定，您可以從 VPC 內執行下列陳述式。

**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;
```

輸出看起來應該類似此範例，並在從 IPv4 網路連線時傳回 2 到 6 個具有私有 IPv4 地址的節點，視您的 VPC 設定和 AWS 區域而定。

```
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)

## 我無法使用 連線 **cassandra-stress**

您嘗試使用 **cassandra-stress** 命令連線至 Amazon Keyspaces，但收到 **SSL context** 錯誤。

如果您嘗試連線至 Amazon Keyspaces，但您未正確設定 `trustStore`，就會發生這種情況。Amazon Keyspaces 需要使用 Transport Layer Security (TLS) 來協助保護與用戶端的連線。

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，請依照指示設定 `trustStore`，如本主題 所示 [the section called “開始之前”](#)。

設定 `trustStore` 後，您應該能夠使用下列命令進行連線。

```
./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 身分連線

您嘗試使用 IAM 身分連線至 Amazon Keyspaces 資料表，但收到 **Unauthorized** 錯誤。

如果您嘗試使用 IAM 身分（例如 IAM 使用者）連線到 Amazon Keyspaces 資料表，但未實作政策並先提供使用者所需的許可，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。若要與標準驅動程式連線，使用者必須至少 SELECT 存取系統資料表，因為大多數驅動程式會在建立連線時讀取系統金鑰空間/資料表。

如需授予 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 “存取 Amazon Keyspaces 資料表”](#)。

若要檢閱 IAM 特定的疑難排解區段，請參閱 [the section called “疑難排解”](#)。

我嘗試使用 cqlsh 匯入資料，且與 Amazon Keyspaces 資料表的連線遺失

您嘗試使用 cqlsh 將資料上傳至 Amazon Keyspaces，但收到連線錯誤。

cqlsh 用戶端從伺服器收到任何類型的連續三次錯誤後，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符合 Amazon Keyspaces 中的資料表結構描述。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。

用戶端會自動嘗試重新建立連線。

## 對 Amazon Keyspaces 中的容量管理錯誤進行故障診斷

無法使用無伺服器容量？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

本節概述如何識別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，您可能會發現容量事件不足。

由於 Apache Cassandra 是叢集型軟體，專為在節點機群上執行而設計，因此沒有與無伺服器功能相關的例外狀況訊息，例如輸送量容量。大多數驅動程式只了解 Apache Cassandra 中可用的錯誤代碼，因此 Amazon Keyspaces 會使用相同的一組錯誤代碼來維持相容性。

若要將 Cassandra 錯誤對應至基礎容量事件，您可以使用 Amazon CloudWatch 來監控相關的 Amazon Keyspaces 指標。導致用戶端錯誤的容量不足事件，可根據導致事件的資源分類為這三個群組：

- 資料表 – 如果您選擇資料表的佈建容量模式，且應用程式超過佈建的輸送量，則可能會發現容量不足錯誤。如需詳細資訊，請參閱[the section called “設定讀取/寫入容量模式”](#)。
- 分割區 – 如果針對指定分割區的流量超過 3,000 RCUs 或 1,000 個 WCUs，您可能會遇到容量不足的事件。最佳實務是建議在分割區之間均勻分配流量。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最大操作數的配額，可能會遇到輸送量不足的情況。若要提高輸送量，您可以在設定與驅動程式的連線時增加預設連線數。

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”](#)。如需透過 VPC 端點最佳化連線的詳細資訊，請參閱 [the section called “VPC 端點連線”](#)。

若要判斷哪個資源造成傳回用戶端錯誤的容量不足事件，您可以在 Amazon Keyspaces 主控台中檢查儀表板。根據預設，主控台會在資料表的容量索引標籤的容量和相關指標區段中，提供最常見容量和流量相關 CloudWatch 指標的彙總檢視。

若要使用 Amazon CloudWatch 建立您自己的儀表板，請檢查下列 Amazon Keyspaces 指標。

- `PerConnectionRequestRateExceeded` – 對 Amazon Keyspaces 的請求超過每個連線請求率的配額。每個客戶端到 Amazon Keyspaces 的連線最多可支援每秒 3000 個 CQL 請求。透過建立多個連線，可以每秒執行 3000 多個請求。
- `ReadThrottleEvents` – 請求超過資料表讀取容量的 Amazon Keyspaces。
- `StoragePartitionThroughputCapacityExceeded` – 請求超過分割區輸送量容量的 Amazon Keyspaces 儲存分割區。Amazon Keyspaces 儲存分割區每秒最多可支援 1000 WCU/WRU 和 3000 RCU/RRU。為了減少這些例外情況，建議您查看資料模型，以便在更多分區之間分配讀取/寫入流量。
- `WriteThrottleEvents` – 請求超過資料表寫入容量的 Amazon Keyspaces。

若要進一步了解 CloudWatch，請參閱 [the section called “使用 CloudWatch 進行監控”](#)。如需 Amazon Keyspaces 所有可用 CloudWatch 指標的清單，請參閱 [the section called “指標與維度”](#)。

#### Note

若要開始使用顯示 Amazon Keyspaces 所有常用指標的自訂儀表板，您可以使用 [AWS 範例](#) 儲存庫中 GitHub 上可用的預先建置 CloudWatch 範本。

## 主題

- [我從用戶端驅動程式收到容量NoHostAvailable不足的錯誤](#)
- [我在資料匯入期間收到寫入逾時錯誤](#)
- [我看不到金鑰空間或資料表的實際儲存大小](#)



## 我從用戶端驅動程式收到容量NoHostAvailable不足的錯誤

您正在查看資料表的 **Read\_Timeout**或 **Write\_Timeout**例外狀況。

重複嘗試寫入或讀取容量不足的 Amazon Keyspaces 資料表，可能會導致驅動程式特有的用戶端錯誤。

使用 CloudWatch 監控您的佈建和實際輸送量指標，以及資料表的容量不足事件。例如，沒有足夠輸送量容量的讀取請求會失敗，但有Read\_Timeout例外狀況，並會發佈到 ReadThrottleEvents 指標。輸送量容量不足的寫入請求會失敗，但有Write\_Timeout例外狀況，並會發佈到 WriteThrottleEvents 指標。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。

- 增加資料表的佈建輸送量，這是應用程式可使用的最大輸送量。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”](#)。
- 讓服務透過自動擴展代表您管理輸送量容量。如需詳細資訊，請參閱[the section called “使用自動擴展管理輸送量容量”](#)。
- 選擇資料表的隨需容量模式。如需詳細資訊，請參閱[the section called “設定隨需容量模式”](#)。

如果您需要增加帳戶的預設容量配額，請參閱 [配額](#)。

您看到與超過分割區容量相關的錯誤。

當您看到錯誤時StoragePartitionThroughputCapacityExceeded，會暫時超過分割區容量。這可能會由自適應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/寫入流量分散到更多分割區，以減少這些錯誤。Amazon Keyspaces 儲存分割區每秒最多可支援 1000 WCU/WRU 和 3000 RCU/RRU。若要進一步了解如何改善資料模型，以將讀取/寫入流量分散到更多分割區，請參閱 [the section called “建立資料模型”](#)。

Write\_Timeout 例外狀況也可能是由並行寫入操作的速率提高所造成，包括相同邏輯分割區中的靜態和非靜態資料。如果預期流量會執行多個並行寫入操作，其中包含相同邏輯分割區內的靜態和非靜態資料，建議您分別寫入靜態和非靜態資料。分別寫入資料也有助於最佳化輸送量成本。

您看到與超出連線請求率相關的錯誤。

由於下列其中一個原因PerConnectionRequestRateExceeded，您看到。

- 您可能沒有為每個工作階段設定足夠的連線。

- 您取得的連線可能少於可用的對等，因為您未正確設定 VPC 端點許可。如需 VPC 端點政策的詳細資訊，請參閱 [the section called “使用 Amazon Keyspaces 的介面 VPC 端點”](#)。
- 如果您使用的是 4.x 驅動程式，請檢查您是否已啟用主機名稱驗證。根據預設，驅動程式會啟用 TLS 主機名稱驗證。此組態會導致 Amazon Keyspaces 顯示為驅動程式的單一節點叢集。建議您關閉主機名稱驗證。

我們建議您遵循這些最佳實務，以確保您的連線和輸送量已最佳化：

- 設定 CQL 查詢輸送量調校。

Amazon Keyspaces 每秒支援每個 TCP 連線最多 3,000 個 CQL 查詢，但驅動程式可以建立的連線數量沒有限制。

大多數開放原始碼 Cassandra 驅動程式會建立連至 Cassandra 的連線集區，並透過該連線集區進行負載平衡查詢。Amazon Keyspaces 向驅動程式公開 9 個對等 IP 地址。大多數驅動程式的預設行為是建立與每個對等 IP 地址的單一連線。因此，使用預設設定的驅動程式 CQL 查詢輸送量上限為每秒 27,000 個 CQL 查詢。

若要增加此數量，建議您增加驅動程式在其連線集區中維護的每個 IP 地址的連線數量。例如，將每個 IP 地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。

- 最佳化您的單一節點連線。

根據預設，大多數開放原始碼 Cassandra 驅動程式會在建立工作階段時，建立與 `system.peers` 資料表中公告之每個 IP 地址的一或多個連線。不過，某些組態可能會導致驅動程式連線到單一 Amazon Keyspaces IP 地址。如果驅動程式嘗試對等節點進行 SSL 主機名稱驗證（例如 DataStax Java 驅動程式），或透過 VPC 端點連線時，可能會發生這種情況。

若要取得與多個 IP 地址連線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。
- 建立自訂重試政策，以確保重試在相同的節點上執行。如需詳細資訊，請參閱

[the section called “如何設定重試政策”](#)。

- 如果您使用 VPC 端點，請授予用於連線至 Amazon Keyspaces 的 IAM 實體存取許可，以查詢 VPC 以取得端點和網路介面資訊。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** 命令上傳資料時，您會收到逾時錯誤。

```
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 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，請降低每個工作者的寫入速率或總擷取速率，然後重試上傳資料列。如需詳細資訊，請參閱 [the section called “步驟 4：設定 cqlsh COPY FROM 設定”](#)。如需更強大的資料上傳選項，請考慮使用 DSBulk，可從 [GitHub 儲存庫](#) 取得。如需逐步說明，請參閱 [the section called “使用 DSBulk 載入資料”](#)。

## 我看不到金鑰空間或資料表的實際儲存大小

您無法查看金鑰空間或資料表的實際儲存大小。

若要進一步了解資料表的儲存體大小，請參閱 [the section called “在資料表層級評估您的成本”](#)。您也可以開始計算資料表中的資料列大小，來估計儲存體大小。如需計算資料列大小的詳細指示，請參閱 [the section called “估算資料列大小”](#)。

## 對 Amazon Keyspaces 中的資料定義語言錯誤進行故障診斷

建立資源時遇到問題？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 會以非同步方式執行資料定義語言 (DDL) 操作，例如建立和刪除金鑰空間和資料表。如果應用程式在準備好之前嘗試使用資源，則操作會失敗。

您可以在 中監控新金鑰空間和資料表的建立狀態 AWS 管理主控台，這表示金鑰空間或資料表何時處於待定或作用中狀態。您也可以透過查詢系統結構描述資料表，以程式設計方式監控新金鑰空間或資料表的建立狀態。金鑰空間或資料表準備就緒時，會顯示在系統結構描述中。

#### Note

若要使用 最佳化金鑰空間的建立 CloudFormation，您可以使用此公用程式將 CQL 指令碼轉換為 CloudFormation 範本。此工具可從 [GitHub 儲存庫](#)取得。

## 主題

- [我建立了新的金鑰空間，但我無法檢視或存取它](#)
- [我建立了新的資料表，但我無法檢視或存取它](#)
- [我嘗試使用 Amazon Keyspaces point-in-time復原 \(PITR\) 還原資料表，但還原失敗](#)
- [我嘗試使用 INSERT/UPDATE 來編輯自訂存留時間 \(TTL\) 設定，但操作失敗](#)
- [我嘗試將資料上傳至 Amazon Keyspaces 資料表，並收到超過資料欄數量的錯誤](#)
- [我嘗試刪除 Amazon Keyspaces 資料表中的資料，但該範圍的刪除失敗](#)

## 我建立了新的金鑰空間，但我無法檢視或存取它

您從應用程式收到嘗試存取新金鑰空間的錯誤。

如果您嘗試存取仍在非同步建立的新建立 Amazon Keyspaces 金鑰空間，您會收到錯誤。下列錯誤為範例。

```
InvalidRequest: Error from server: code=2200 [Invalid query] message="unconfigured keyspace mykeyspace"
```

檢查新金鑰空間何時可供使用的建議設計模式是輪詢 Amazon Keyspaces 系統結構描述表 (system\_schema\_mcs.\*)。

如需詳細資訊，請參閱[the section called “檢查金鑰空間建立狀態”](#)。

## 我建立了新的資料表，但我無法檢視或存取它

您從應用程式收到嘗試存取新資料表的錯誤。

如果您嘗試存取仍在非同步建立的新建立 Amazon Keyspaces 資料表，您會收到錯誤。例如，嘗試查詢尚無法使用的資料表會失敗並顯示 `unconfigured table` 錯誤。

```
InvalidRequest: Error from server: code=2200 [Invalid query] message="unconfigured
table mykeyspace.mytable"
```

嘗試使用檢視資料表時，會 `sync_table()` 失敗 `KeyError`。

```
KeyError: 'mytable'
```

檢查新資料表何時可供使用的建議設計模式是輪詢 Amazon Keyspaces 系統結構描述資料表 (`system_schema_mcs.*`)。

這是所建立資料表的範例輸出。

```
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 “檢查資料表建立狀態”](#)。

我嘗試使用 Amazon Keyspaces point-in-time復原 (PITR) 還原資料表，但還原失敗

如果您嘗試使用point-in-time復原 (PITR) 還原 Amazon Keyspaces 資料表，而且您看到還原程序開始但未成功完成，您可能尚未設定此特定資料表還原程序所需的所有必要許可。

除了使用者許可之外，Amazon Keyspaces 可能需要許可，才能代表您委託人在還原程序期間執行動作。如果資料表使用客戶受管金鑰加密，或者您使用限制傳入流量的 IAM 政策，就會發生這種情況。

例如，如果您在 IAM 政策中使用條件索引鍵來限制來源流量至特定端點或 IP 範圍，則還原操作會失敗。若要允許 Amazon Keyspaces 代表您委託人執行資料表還原操作，您必須在 IAM 政策中新增aws:ViaAWSService全域條件金鑰。

如需還原資料表之許可的詳細資訊，請參閱 [the section called “設定還原的 IAM 許可”](#)。

我嘗試使用 INSERT/UPDATE 來編輯自訂存留時間 (TTL) 設定，但操作失敗

如果您嘗試插入或更新自訂 TTL 值，操作可能會失敗，並出現下列錯誤。

```
TTL is not yet supported.
```

若要使用 `INSERT` 或 `UPDATE` 操作指定資料列或資料欄的自訂 TTL 值，您必須先為資料表啟用 TTL。您可以使用 `ttl` 自訂屬性為資料表啟用 TTL。

如需啟用資料表自訂 TTL 設定的詳細資訊，請參閱 [the section called “更新資料表自訂 TTL”](#)。

我嘗試將資料上傳至 Amazon Keyspaces 資料表，並收到超過資料欄數量的錯誤

您正在上傳資料，並已超過可同時更新的欄數。

當您的資料表結構描述超過 350 KB 的大小上限時，就會發生此錯誤。如需詳細資訊，請參閱[配額](#)。

我嘗試刪除 Amazon Keyspaces 資料表中的資料，但該範圍的刪除失敗

您嘗試依分割區索引鍵刪除資料，並收到範圍刪除錯誤。

當您嘗試在一個刪除操作中刪除超過 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 “刪除範圍”](#)。

若要在單一分割區中刪除超過 1,000 個資料列，請考慮下列選項。

- 依分割區刪除 – 如果大部分分割區少於 1,000 個資料列，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，請改為嘗試由叢集資料欄刪除。
- 透過叢集資料欄刪除 – 如果您的模型包含多個叢集資料欄，您可以使用資料欄階層來刪除多個資料列。叢集資料欄是一種巢狀結構，您可以透過對頂層資料欄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（分割區資料欄和叢集資料欄）刪除每一列。
- 最佳實務是考慮跨分割區分割資料列 – 在 Amazon Keyspaces 中，建議您將輸送量分散到資料表分割區。這會將資料和存取平均分配到實體資源，以提供最佳輸送量。如需詳細資訊，請參閱[the section called “建立資料模型”](#)。

當您規劃大量工作負載的刪除操作時，也請考慮下列建議。

- 使用 Amazon Keyspaces，分割區可以包含幾乎沒有限制的資料列數目。這可讓您擴展分割區「比傳統的 100 MB Cassandra 指引更寬」。時間序列或分類帳隨著時間經過 1 GB 的資料成長並不罕見。
- 使用 Amazon Keyspaces，當您必須為繁重的工作負載執行刪除操作時，無需考慮任何壓縮策略或墓地。您可以視需要刪除任意數量的資料，而不會影響讀取效能。



# 監控 Amazon Keyspaces (適用於 Apache Cassandra)

監控是維護 Amazon Keyspaces 及其他 AWS 解決方案可靠性、可用性和效能的重要部分。AWS 提供下列監控工具來監看 Amazon Keyspaces、在發生錯誤時回報，以及適時採取自動動作：

- Amazon Keyspaces 在 中提供預先設定的儀表板，AWS 管理主控台 顯示帳戶內所有資料表彙總的延遲和錯誤。
- Amazon CloudWatch AWS 會即時監控您的 AWS 資源和您在 上執行的應用程式。您可以使用自訂儀表板收集和追蹤指標。例如，您可以在各種時間和不同的負載條件下測量效能，為環境中的一般 Amazon Keyspaces 效能建立基準。當您監控 Amazon Keyspaces 時，請存放歷史監控資料，以便將其與目前的效能資料進行比較，識別正常效能模式和效能異常，並設計方法來解決問題。若要建立基準，您至少應該監控系統錯誤。如需更多資訊，請參閱 [Amazon CloudWatch 使用者指南](#)。
- Amazon CloudWatch 警示會在您指定的期間內監控單一指標，並根據相對於指定閾值的指標值，在多個期間內執行一或多個動作。例如，如果您在佈建模式中使用 Amazon Keyspaces 搭配應用程式自動擴展，則動作是由 Amazon Simple Notification Service (Amazon SNS) 傳送的通知，以評估 Application Auto Scaling 政策。

CloudWatch 警示不會只因為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[使用 Amazon CloudWatch 監控 Amazon Keyspaces](#)。

- Amazon CloudWatch Logs 可讓您監控、存放和存取來自 Amazon Keyspaces 資料表、CloudTrail 和其他來源的日誌檔案。CloudWatch Logs 可監控日誌檔案中的資訊，並在達到特定閾值時通知您。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 Logs 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，並傳送日誌檔案至您指定的 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 為無伺服器事件匯流排服務，可讓您輕鬆將應用程式與來自各種來源的資料互相連線。EventBridge 可從您自己的應用程式、Software-as-a-Service(SaaS) 應用程式 AWS 和服務提供即時資料串流，並將該資料路由到 Lambda 等目標。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

- [使用 Amazon CloudWatch 監控 Amazon Keyspaces](#)
- [使用 記錄 Amazon Keyspaces API 呼叫 AWS CloudTrail](#)



# 使用 Amazon CloudWatch 監控 Amazon Keyspaces

您可以使用 Amazon CloudWatch 監控 Amazon Keyspaces，這會收集原始資料並將其處理為可讀且近乎即時的指標。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

若要快速開始使用顯示 Amazon Keyspaces 常見指標的預先設定 CloudWatch 儀表板，您可以使用提供的 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 指標和維度](#)
- [在 CloudWatch 中檢視 Amazon Keyspaces 指標](#)
- [建立 CloudWatch 警示以監控 Amazon Keyspaces](#)

## 如何使用 Amazon Keyspaces 指標？

Amazon Keyspaces 報告的指標提供您可以以不同方式分析的資訊。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。如需指標和保留的詳細資訊，請參閱 [指標](#)。

| 如何？                 | 相關指標                                                                         |
|---------------------|------------------------------------------------------------------------------|
| 如何判斷是否發生任何系統錯誤？     | 您可以監控 <code>SystemErrors</code> 來判斷是否有任何請求導致伺服器錯誤碼。這項指標通常應該等於零。如果不是，您可能想要調查。 |
| 如何比較平均佈建讀取與已耗用讀取容量？ | 監控平均佈建的讀取容量和耗用的讀取容量                                                          |

| 如何？                       | 相關指標                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                           | <ol style="list-style-type: none"> <li>1. 將 ConsumedReadCapacityUnits 和 的期間ProvisionedReadCapacityUnits 設定為您要監控的間隔。</li> <li>2. 將 的統計資料ConsumedReadCapacityUnits 從 Average變更為 Sum。</li> <li>3. 建立新的空白數學表達式。</li> <li>4. 在新數學表達式的詳細資訊區段中，輸入 的 ID , ConsumedReadCapacityUnits 並將指標除以指標的 CloudWatch PERIOD 函數 (metric_id/(PERIOD(metric_id)))。</li> <li>5. 取消選取 ConsumedReadCapacityUnits 。</li> </ol> <p>您現在可以將平均消耗的讀取容量與佈建的容量進行比較。如需基本算術函數以及如何建立時間序列的詳細資訊，請參閱<a href="#">使用指標數學</a>。</p>                               |
| <p>如何比較平均佈建寫入與耗用寫入容量？</p> | <p>監控平均佈建的寫入容量和耗用的寫入容量</p> <ol style="list-style-type: none"> <li>1. 將 ConsumedWriteCapacityUnits 和 的期間ProvisionedWriteCapacityUnits 設定為您要監控的間隔。</li> <li>2. 將 的統計資料ConsumedWriteCapacityUnits 從 Average變更為 Sum。</li> <li>3. 建立新的空白數學表達式。</li> <li>4. 在新數學表達式的詳細資訊區段中，輸入 的 ID , ConsumedWriteCapacityUnits 並將指標除以指標的 CloudWatch PERIOD 函數 (metric_id/(PERIOD(metric_id)))。</li> <li>5. 取消選取 ConsumedWriteCapacityUnits 。</li> </ol> <p>您現在可以將平均耗用寫入容量與佈建容量進行比較。如需基本算術函數以及如何建立時間序列的詳細資訊，請參閱<a href="#">使用指標數學</a>。</p> |

## Amazon Keyspaces 指標和維度

當您與 Amazon Keyspaces 互動時，它會將指標和維度傳送至 Amazon CloudWatch。所有指標都會每分鐘彙總和報告。下列各節概述 Amazon Keyspaces 的所有可用指標。

### 主題

- [Amazon Keyspaces 指標](#)
- [Amazon Keyspaces 指標的維度](#)
- [Amazon Keyspaces 變更資料擷取 \(CDC\) 的指標](#)
- [Amazon Keyspaces 變更資料擷取 \(CDC\) 指標的維度](#)

### Amazon Keyspaces 指標

Amazon CloudWatch 每隔一分鐘彙總 Amazon Keyspaces 指標。

並非所有統計數字，例如 Average 或 Sum，皆適用於所有指標。不過，所有這些值都可以透過 Amazon Keyspaces 主控台，或使用 CloudWatch 主控台 AWS CLI，或為所有指標提供 AWS SDKs。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  | <p>帳戶資料表可使用的讀取容量單位數目上限。對於隨需資料表，此限制會限制資料表可以使用的最大讀取請求單位。</p> <p>單位：Count</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>• Maximum – 帳戶資料表可使用的讀取容量單位數目上限。</li> </ul> |
| AccountMaxTableLevelWrites | <p>帳戶資料表可使用的最大寫入容量單位數量。對於隨需資料表，此限制會限制資料表可以使用的最大寫入請求單位。</p> <p>單位：Count</p> <p>有效的統計數字：</p>                                                                                    |

| 指標                                         | 描述                                                                                                                                                                                                                                                                      |
|--------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                            | <ul style="list-style-type: none"> <li>Maximum – 帳戶資料表可使用的寫入容量單位數目上限。</li> </ul>                                                                                                                                                                                        |
| AccountProvisionedReadCapacityUtilization  | <p>帳戶可以使用的佈建讀取容量單元百分比。</p> <p>單位：Percent</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>Maximum：帳戶使用的佈建讀取容量單元百分比上限。</li> <li>Minimum：帳戶使用的佈建讀取容量單元百分比下限。</li> <li>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。</li> </ul> |
| AccountProvisionedWriteCapacityUtilization | <p>帳戶使用的佈建寫入容量單元百分比。</p> <p>單位：Percent</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>Maximum：帳戶使用的佈建寫入容量單元百分比上限。</li> <li>Minimum：帳戶使用的佈建寫入容量單元百分比下限。</li> <li>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。</li> </ul>   |

| 指標                             | 描述                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|--------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BillableTableSizeInBytes       | <p>資料表的計費大小，以位元組為單位。這是資料表中所有資料列的編碼大小總和。此指標可協助您追蹤一段時間內的資料表儲存成本。</p> <p>單位：Bytes</p> <p>維度：Keyspace, TableName</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>• Maximum – 資料表的儲存大小上限。</li> <li>• Minimum – 資料表的儲存大小下限。</li> <li>• Average – 資料表的平均儲存大小。此指標的計算間隔為 4 - 6 小時。</li> </ul>                                                                                                                                                 |
| ConditionalCheckFailedRequests | <p>失敗的輕量型交易 (LWT) 寫入請求數目。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 會以 1 遞增。評估為 false 的條件檢查會根據資料列的大小耗用寫入容量單位。如需詳細資訊，請參閱<a href="#">the section called “LWT 的預估容量使用量”</a>。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>• Minimum</li> <li>• Maximum</li> <li>• Average</li> <li>• SampleCount</li> <li>• Sum</li> </ul> |

| 指標                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 描述                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ConsumedReadCapacityUnits                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | <p>在指定期間內耗用的讀取容量單位數量。如需詳細資訊，請參閱<a href="#">讀取/寫入容量模式</a>。</p> <div><p> <b>Note</b></p><p>若要了解您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內消耗的輸送量。然後，將總和除以一分鐘內的秒數 (60)，以計算 ConsumedReadCapacityUnits 每秒平均數（認識到此平均數不會反白顯示該分鐘內讀取活動中發生任何大型但短暫的峰值）。如需比較平均耗用讀取容量與佈建讀取容量的詳細資訊，請參閱 <a href="#">the section called “使用指標”</a></p></div> |
| <p>單位：Count</p> <p>維度：Keyspace, TableName</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"><li>• Minimum – 任何個別請求對資料表所耗用的讀取容量單位數目下限。</li><li>• Maximum – 任何個別請求對資料表所耗用的讀取容量單位數目上限。</li><li>• Average：每個請求消耗的平均讀取容量。量。</li></ul> <div><p> <b>Note</b></p><p>此 Average 數值受樣本數值會為零的閒置時段所影響。</p></div> <ul style="list-style-type: none"><li>• 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。</li><li>• SampleCount – Amazon Keyspaces 的請求數量，即使沒有消耗讀取容量。</li></ul> |                                                                                                                                                                                                                                                                                                                                                                                                    |

| 指標 | 描述                                                                                                                                                       |
|----|----------------------------------------------------------------------------------------------------------------------------------------------------------|
|    | <div><div> Note</div><div>此 SampleCount 數值受樣本數值會為零的閒置時段所影響。</div></div> |

| 指標                         | 描述                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
|----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ConsumedWriteCapacityUnits | <p>在指定期間內耗用的寫入容量單位數量。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<a href="#">讀取/寫入容量模式</a>。</p> <div> <p> <b>Note</b></p> <p>若要了解您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內消耗的輸送量。然後，將總和除以一分鐘內的秒數 (60)，以計算 ConsumedWriteCapacityUnits 每秒平均數（認識到此平均值不會反白顯示在該分鐘內發生的任何大型但短暫的寫入活動峰值）。如需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 <a href="#">the section called “使用指標”</a></p> </div> <p>單位：Count</p> <p>維度：Keyspace, TableName</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>• Minimum – 任何個別請求對資料表所耗用的寫入容量單位數目下限。</li> <li>• Maximum – 任何個別請求對資料表所耗用的寫入容量單位數目上限。</li> <li>• Average：每個請求消耗的平均寫入容量。</li> </ul> <div> <p> <b>Note</b></p> <p>此 Average 數值受樣本數值會為零的閒置時段所影響。</p> </div> <ul style="list-style-type: none"> <li>• 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。</li> </ul> |



| 指標                                                | 描述                                                                                                                                                                                                                                                                                                                                    |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|                                                   | <ul style="list-style-type: none"><li>• <b>SampleCount</b> – Amazon Keyspaces 的請求數量，即使沒有消耗寫入容量。</li></ul> <div> <b>Note</b><br/>此 <b>SampleCount</b> 數值受樣本數值會為零的閒置時段所影響。</div>                                                                       |
| <b>MaxProvisionedTableReadCapacityUtilization</b> | <p>帳戶的最高佈建讀取資料表所使用的佈建讀取容量單位百分比。</p> <p>單位：Percent</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"><li>• <b>Maximum</b>：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。</li><li>• <b>Minimum</b>：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。</li><li>• <b>Average</b>：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。</li></ul> |

| 指標                                          | 描述                                                                                                                                                                                                                                                                                                                                                                                                 |
|---------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| MaxProvisionedTableWriteCapacityUtilization | <p>帳戶的最高佈建寫入資料表所使用的佈建寫入容量百分比。</p> <p>單位：Percent</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>Maximum – 帳戶的最高佈建寫入資料表所使用的佈建寫入容量單位百分比上限。</li> <li>Minimum – 帳戶的最高佈建寫入資料表所使用的佈建寫入容量單位百分比下限。</li> <li>Average – 帳戶最高佈建寫入資料表使用之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。</li> </ul>                                                                               |
| PerConnectionRequestRateExceeded            | <p>對超過每個連線請求速率配額的 Amazon Keyspaces 的請求。Amazon Keyspaces 的每個用戶端連線每秒最多可支援 3000 個 CQL 請求。用戶端可以建立多個連線來增加輸送量。</p> <p>當您使用多區域複寫時，每個複寫的寫入也會造成此配額。根據最佳實務，建議您增加資料表的連線數，以避免PerConnectionRequestRateExceeded 發生錯誤。您可以在 Amazon Keyspaces 中擁有的連線數目沒有限制。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName, Operation</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>SampleCount</li> <li>Sum</li> </ul> |

| 指標                           | 描述                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ProvisionedReadCapacityUnits | <p>資料表的佈建讀取容量單位數目。</p> <p>TableName 維度會傳回資料表ProvisionedReadCapacityUnits 的。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"><li>• 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。</li><li>• 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。</li><li>• 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。</li></ul> |

| 指標                            | 描述                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ProvisionedWriteCapacityUnits | <p>資料表的佈建寫入容量單位數目。</p> <p>TableName 維度會傳回資料表ProvisionedWriteCapacityUnits 的。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>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。</li> <li>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。</li> <li>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。</li> </ul> |
| ReadThrottleEvents            | <p>對 Amazon Keyspaces 的請求超過資料表的佈建讀取容量、帳戶層級配額、每個連線配額的請求或分割區層級配額。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName, Operation</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>SampleCount</li> <li>Sum</li> </ul>                                                                                                                                                                                                                                                                        |

| 指標                        | 描述                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
|---------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReplicationLatency        | <p>此指標僅適用於多區域金鑰空間inserts，並測量deletes從一個複本資料表複寫 updates、或 到多區域金鑰空間中另一個複本資料表所花費的時間。</p> <p>單位：Millisecond</p> <p>維度：TableName, ReceivingRegion</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>• Average</li> <li>• Maximum</li> <li>• Minimum</li> </ul>                                                                                                                                                                                               |
| ReturnedItemCountBySelect | <p>在指定期間內多列SELECT查詢傳回的資料列數。多列SELECT查詢是不包含完整主索引鍵的查詢，例如完整資料表掃描和範圍查詢。</p> <p>傳回的資料列數不一定與評估的資料列數相同。例如，假設您在具有 100 個資料列的資料表ALLOW FILTERING 上請求SELECT *具有的，但指定的WHERE子句縮小了結果範圍，因此只傳回 15 個資料列。在此情況下，的回應SELECT將包含 100 ScanCount 的 和 15 個傳回資料列Count的。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName, Operation</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>• Minimum</li> <li>• Maximum</li> <li>• Average</li> <li>• SampleCount</li> <li>• Sum</li> </ul> |

| 指標                                         | 描述                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
|--------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| StoragePartitionThroughputCapacityExceeded | <p>請求超過分割區輸送量容量的 Amazon Keyspaces 儲存分割區。Amazon Keyspaces 儲存分割區每秒最多可支援 1000 WCU/WRU 和 3000 RCU/RRU。我們建議您檢閱資料模型，將讀取/寫入流量分散到更多分割區，以減輕這些例外狀況。</p> <div> <b>Note</b><br/>邏輯 Amazon Keyspaces 分割區可以跨越多個儲存分割區，而且大小幾乎沒有限制。</div> <p>單位：Count</p> <p>維度：Keyspace, TableName, Operation</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"><li>• SampleCount</li><li>• Sum</li></ul> |
| SuccessfulRequestCount                     | <p>在指定期間內處理的成功請求數量。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName, Operation</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"><li>• SampleCount</li></ul>                                                                                                                                                                                                                                                                                                 |

| 指標                       | 描述                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SuccessfulRequestLatency | <p>在指定期間內向 Amazon Keyspaces 提出的成功請求。SuccessfulRequestLatency 可以提供兩種不同類型的資訊：</p> <ul style="list-style-type: none"><li>成功請求的經過時間 (Minimum、Maximum、Sum 或 Average)。</li><li>成功請求的數量 (SampleCount )。</li></ul> <p>SuccessfulRequestLatency 只會反映 Amazon Keyspaces 內的活動，不會考慮網路延遲或用戶端活動。</p> <p>單位：Milliseconds</p> <p>維度：Keyspace, TableName, Operation</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"><li>Minimum</li><li>Maximum</li><li>Average</li><li>SampleCount</li></ul> |
| SystemErrors             | <p>在指定期間內產生的 Amazon Keyspaces ServerError 請求。ServerError 通常表示內部服務錯誤。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName, Operation</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"><li>Sum</li><li>SampleCount</li></ul>                                                                                                                                                                                                                                                         |

| 指標                          | 描述                                                                                                                                                                                                                                                                                                                                                                                            |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SystemReconciliationDeletes | <p>啟用用戶端時間戳記時，用於刪除斷層資料的單位。每個 SystemReconciliationDelete 都提供足夠的容量來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欄，則需要 3 SystemReconciliationDeletes。或者，若要刪除包含 3.5 KB 資料的整個資料列，需要 4 個 SystemReconciliationDeletes。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>Sum – 期間內 SystemReconciliationDeletes 消耗的總數。</li> </ul> |
| TTLDeletes                  | <p>使用存留時間 (TTL) 刪除或更新資料的資料所消耗的單位。每個 TTLDelete 都提供足夠的容量來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欄，則需要刪除 3 個 TTL。或者，若要刪除包含 3.5 KB 資料的整個資料列，需要 4 個 TTL 刪除。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>Sum – 期間內 TTLDeletes 消耗的總數。</li> </ul>                                                                     |



| 指標                  | 描述                                                                                                                                                                                                                                                                                                                              |
|---------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| UserErrors          | <p>在指定期間內產生InvalidRequest 錯誤的 Amazon Keyspaces 請求。</p> <p>InvalidRequest 錯誤通常代表用戶端錯誤，例如無效的參數組合、嘗試更新不存在的資料表，或不正確的請求簽章。</p> <p>UserErrors 代表目前 AWS 區域 和目前 無效請求的彙總 AWS 帳戶。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName, Operation</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>Sum</li> <li>SampleCount</li> </ul> |
| WriteThrottleEvents | <p>對 Amazon Keyspaces 的請求超過資料表的佈建寫入容量、帳戶層級配額、每個連線配額的請求或分割區層級配額。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName, Operation</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>SampleCount</li> <li>Sum</li> </ul>                                                                                                         |

## Amazon Keyspaces 指標的維度

Amazon Keyspaces 的指標取決於帳戶、資料表名稱或操作的值。您可以使用 CloudWatch 主控台，沿著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。此值可以是目前 區域和目前 中的任何鍵空間 AWS 帳戶。                                 |
| Operation | 此維度會將資料限制在其中一個 Amazon Keyspaces CQL 操作，例如 INSERT或 SELECT操作。                     |
| TableName | 此維度將資料限制為特定資料表。此值可以是目前區域和目前區域中的任何資料表名稱 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。 |

## Amazon Keyspaces 變更資料擷取 (CDC) 的指標

CloudWatch 帳戶中提供下列指標，可監控對 Amazon Keyspaces 變更資料擷取 (CDC) 端點的 API 呼叫。

| 指標                       | 描述                                                                                                                                                                                                                                                    |
|--------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ReturnedRecordsCount     | <p>GetRecords 操作在指定期間內傳回的串流記錄數目。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName, Operation</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>• Minimum</li> <li>• Maximum</li> <li>• Average</li> <li>• SampleCount</li> <li>• Sum</li> </ul> |
| SuccessfulRequestLatency | <p>在指定期間內向 Amazon Keyspaces 提出的成功請求。</p> <p>SuccessfulRequestLatency 可以提供兩種不同類型的資訊：</p>                                                                                                                                                               |

| 指標           | 描述                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
|              | <ul style="list-style-type: none"> <li>成功請求的經過時間 (Minimum、Maximum、Sum 或 Average)。</li> <li>成功請求的數量 (SampleCount )。</li> </ul> <p>SuccessfulRequestLatency 僅反映 Amazon Keyspaces CDC APIs 內的活動，且不考慮網路延遲或用戶端活動。</p> <p>單位：Milliseconds</p> <p>維度：Keyspace, TableName, Operation</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>Minimum</li> <li>Maximum</li> <li>Average</li> <li>SampleCount</li> </ul> |
| SystemErrors | <p>在指定期間內產生HTTP 500狀態碼的 Amazon Keyspaces CDC 請求。HTTP 500 狀態碼通常表示內部服務錯誤。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName, Operation</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>Sum</li> <li>SampleCount</li> </ul>                                                                                                                                                                                   |

| 指標         | 描述                                                                                                                                                                                                                                                                                    |
|------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| UserErrors | <p>在指定期間內產生HTTP 400狀態碼的 Amazon Keyspaces CDC 請求。</p> <p>UserErrors 代表目前 AWS 區域 和目前 Amazon Keyspaces CDC 的HTTP 400錯誤彙總 AWS 帳戶。</p> <p>單位：Count</p> <p>維度：Keyspace, TableName, Operation</p> <p>有效的統計數字：</p> <ul style="list-style-type: none"> <li>Sum</li> <li>SampleCount</li> </ul> |

## Amazon Keyspaces 變更資料擷取 (CDC) 指標的維度

Amazon Keyspaces 的指標取決於帳戶、資料表名稱或操作的值。您可以使用 CloudWatch 主控台，沿著下表中的任何維度擷取 Amazon Keyspaces 資料。

| 維度          | 描述                                                                                                   |
|-------------|------------------------------------------------------------------------------------------------------|
| StreamLabel | 指定 KeyspaceName 和 TableName 時，此維度會將資料限制在特定變更資料擷取 (CDC) 串流。                                           |
| Keyspace    | 此維度會將資料限制在特定的金鑰空間。此值可以是目前 區域和目前 中的任何鍵空間 AWS 帳戶。                                                      |
| TableName   | 此維度將資料限制為特定資料表。此值可以是目前區域和目前區域中的任何資料表名稱 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。                      |
| Operation   | 此維度將資料限制為下列其中一個 Amazon Keyspaces Streams API 操作：GetRecords、GetStream、GetShardIterator 和 ListStreams。 |

## 在 CloudWatch 中檢視 Amazon Keyspaces 指標

當您與 Amazon Keyspaces 互動時，它會將下列指標和維度傳送至 Amazon CloudWatch。所有指標都會每分鐘彙總和報告。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. 在 <https://console.aws.amazon.com/cloudwatch/> 開啟 CloudWatch 主控台。
2. 如有需要，請變更 AWS 區域。在導覽列上，選擇 AWS 資源所在的區域。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. 在所有指標索引標籤下，選擇 AWS/Cassandra。

使用 AWS CLI 檢視指標

- 在命令提示中，使用下列命令。

```
aws cloudwatch list-metrics --namespace "AWS/Cassandra"
```

## 建立 CloudWatch 警示以監控 Amazon Keyspaces

您可以為 Amazon Keyspaces 建立 Amazon CloudWatch 警示，在警示變更狀態時傳送 Amazon Simple Notification Service (Amazon SNS) 訊息。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。

當您以佈建模式搭配 Application Auto Scaling 使用 Amazon Keyspaces 時，此服務會代表您建立兩對 CloudWatch 警示。每對代表佈建和耗用輸送量設定的上限和下限。當資料表的實際使用率持續一段時間偏離您的目標使用率時，就會觸發這些 CloudWatch 警示。若要進一步了解 Application Auto Scaling 建立的 CloudWatch 警示，請參閱 [the section called “Amazon Keyspaces 自動擴展的運作方式”](#)。

警示僅會針對持續狀態變更調用動作。CloudWatch 警示不會只因為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。

如需建立 CloudWatch 警示的詳細資訊，請參閱 [《Amazon CloudWatch 使用者指南》中的使用 Amazon CloudWatch 警示](#)。Amazon CloudWatch

## 使用 記錄 Amazon Keyspaces API 呼叫 AWS CloudTrail

Amazon Keyspaces 已與 服務整合 AWS CloudTrail，此服務提供由 Amazon Keyspaces AWS 中的使用者、角色或服務所採取之動作的記錄。CloudTrail 會將 Amazon Keyspaces 的 Data Definition Language (DDL) API 呼叫和 Data Manipulation Language (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 操作的程式設計呼叫。

如果您建立線索，則可以將 CloudTrail 事件持續交付至 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 CloudTrail 主控台的事件歷史記錄中檢視最新的支援事件。您可以使用 CloudTrail 所收集的資訊，判斷對 Amazon Keyspaces 提出的請求、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 [「AWS CloudTrail 使用者指南」](#)。

### 主題

- [在 CloudTrail 中設定 Amazon Keyspaces 日誌檔案項目](#)
- [CloudTrail 中的 Amazon Keyspaces 資料定義語言 \(DDL\) 資訊](#)
- [CloudTrail 中的 Amazon Keyspaces 資料處理語言 \(DML\) 資訊](#)
- [了解 Amazon Keyspaces 日誌檔案項目](#)

## 在 CloudTrail 中設定 Amazon Keyspaces 日誌檔案項目

在 CloudTrail 中記錄的每個 Amazon Keyspaces API 動作都包含以 CQL 查詢語言表示的請求參數。如需更多資訊，請參閱[CQL 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 事件歷史記錄檢視事件](#)。

若要坚持記錄中的事件 AWS 帳戶，包括 Amazon Keyspaces 的事件，請建立追蹤。線索能讓 CloudTrail 將日誌檔案交付至 Amazon S3 儲存貯體。根據預設，當您在主控台中建立線索時，線索會套用至所有 AWS 區域。線索會記錄 AWS 分割區中所有區域的事件，並將日誌檔案交付至您指定的

Amazon 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。

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 支援的服務和整合](#)
- [設定 CloudTrail 的 Amazon SNS 通知](#)
- [從多個區域接收 CloudTrail 日誌檔案](#)
- [從多個帳戶接收 CloudTrail 日誌檔案](#)

每一筆事件或日誌項目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：

- 是否使用根或 AWS Identity and Access Management (IAM) 使用者登入資料提出請求。
- 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 userIdentity 元素](#)。

## CloudTrail 中的 Amazon Keyspaces 資料定義語言 (DDL) 資訊

建立帳戶 AWS 帳戶時，您的上會啟用 CloudTrail。當 Amazon Keyspaces 中發生 DDL 活動時，該活動會自動記錄為 CloudTrail 事件，以及事件歷史記錄中的其他 AWS 服務事件。下表顯示為 Amazon Keyspaces 記錄的 DDL 陳述式。

| CloudTrail<br>eventName | 陳述式 | CQL 動作          | AWS SDK 動作     |
|-------------------------|-----|-----------------|----------------|
| CreateKeyspace          | DDL | CREATE KEYSPACE | CreateKeyspace |
| AlterKeyspace           | DDL | ALTER KEYSPACE  | UpdateKeyspace |
| DropKeyspace            | DDL | DROP KEYSPACE   | DeleteKeyspace |
| CreateTable             | DDL | CREATE TABLE    | CreateTable    |
| DropTable               | DDL | DROP TABLE      | DeleteTable    |

| CloudTrail<br>eventName | 陳述式 | CQL 動作       | AWS SDK 動作                                      |
|-------------------------|-----|--------------|-------------------------------------------------|
| AlterTable              | DDL | ALTER TABLE  | UpdateTable ,<br>TagResource ,<br>UntagResource |
| CreateUdt               | DDL | CREATE TYPE  | CreateType                                      |
| DropUdt                 | DDL | DROP TYPE    | DeleteType                                      |
| GetStream               | DDL | 沒有可用的 CQL 動作 | GetStream                                       |
| ListStreams             | DDL | 沒有可用的 CQL 動作 | ListStreams                                     |

## CloudTrail 中的 Amazon Keyspaces 資料處理語言 (DML) 資訊

若要使用 CloudTrail 啟用記錄 Amazon Keyspaces DML 陳述式，您必須先在 CloudTrail 中啟用資料平面 API 活動的記錄。您可以選擇使用 CloudTrail 主控台記錄資料事件類型 Cassandra 資料表的活動，或使用 CLI 或 CloudTrail API 操作將值設定為 `resources.type AWS::Cassandra::Table`，以開始在新的或現有的線索中記錄 Amazon Keyspaces DML 事件。如需詳細資訊，請參閱[記錄資料事件](#)。

如需詳細資訊和示範如何為資料事件建立警示的範例，請參閱 資料庫 AWS 部落格上的下列文章：[使用 Amazon Keyspaces 的 DML 稽核 \(適用於 Apache Cassandra\)](#)。

下表顯示 CloudTrail 為記錄的資料事件 Cassandra table。

| CloudTrail<br>eventName | 陳述式 | CQL 動作 | AWS SDK 動作                                                                                          |
|-------------------------|-----|--------|-----------------------------------------------------------------------------------------------------|
| Select                  | DML | SELECT | GetKeyspace ,<br>GetTable, GetType,<br>ListKeyspaces , ListTables , ListTypes , ListTagsForResource |



| CloudTrail eventName | 陳述式 | CQL 動作       | AWS SDK 動作       |
|----------------------|-----|--------------|------------------|
| 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 |
| GetShardIterator     | DML | 沒有可用的 CQL 動作 | GetShardIterator |
| GetRecords           | DML | 沒有可用的 CQL 動作 | GetRecords       |

## 了解 Amazon Keyspaces 日誌檔案項目

CloudTrail 日誌檔案包含一或多個日誌專案。一個事件為任何來源提出的單一請求，並包含請求動作、請求的日期和時間、請求參數等資訊。CloudTrail 日誌檔並非依公有 API 呼叫的堆疊追蹤排序，因此不會以任何特定順序出現。

下列範例顯示示範 CreateKeyspace、CreateTable、和 DropTable動作的 CloudTrail DropKeyspace日誌項目：

```
{
 "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"
 }
 },
 "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": "e5aebac-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": "1a5cbcdc-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"
 }
]
}

```

下列日誌檔案顯示 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"
 }
}

```

```
}
}
```

下列日誌檔案顯示 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"
}
}

```

下列日誌檔案顯示 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"
 }
 }
}

```

下列日誌檔案顯示 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"
 }
 }
}

```

# Amazon Keyspaces 的安全性（適用於 Apache Cassandra）

的雲端安全性 AWS 是最高優先順序。身為 AWS 客戶，您可以受益於資料中心和網路架構，該架構專為滿足最安全敏感組織的需求而建置。

安全性是 AWS 與您之間共同責任。[共同責任模型](#)將其描述為雲端的安全性和雲端中的安全性：

- 雲端的安全性 – AWS 負責保護在 中執行 AWS 服務的基礎設施 AWS 雲端。AWS 也為您提供可安全使用的服務。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[AWS 合規計劃範圍內的服務](#)。
- 雲端的安全性 – 您的責任取決於您使用 AWS 的服務。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。

本文件將協助您了解如何在使用 Amazon Keyspaces 時套用共同責任模型。下列主題說明如何設定 Amazon Keyspaces 以符合您的安全與合規目標。您也將了解如何使用其他 AWS 服務，協助您監控和保護 Amazon Keyspaces 資源。

## 主題

- [Amazon Keyspaces 中的資料保護](#)
- [AWS Identity and Access Management 適用於 Amazon Keyspaces](#)
- [Amazon Keyspaces 的合規驗證（適用於 Apache Cassandra）](#)
- [Amazon Keyspaces 中的彈性和災難復原](#)
- [Amazon Keyspaces 中的基礎設施安全性](#)
- [Amazon Keyspaces 的組態和漏洞分析](#)
- [Amazon Keyspaces 的安全最佳實務](#)

## Amazon Keyspaces 中的資料保護

AWS [共同責任模型](#)適用於 Amazon Keyspaces 中的資料保護（適用於 Apache Cassandra）。如此模型所述，AWS 負責保護執行所有 的全域基礎設施 AWS 雲端。您負責維護在此基礎設施上託管內容的控制權。您也同時負責所使用 AWS 服務 的安全組態和管理任務。如需資料隱私權的詳細資訊，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，我們建議您保護 AWS 帳戶 登入資料，並使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 設定個別使用者。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 設定 API 和使用使用者活動記錄 AWS CloudTrail。如需有關使用 CloudTrail 追蹤擷取 AWS 活動的資訊，請參閱AWS CloudTrail 《使用者指南》中的[使用 CloudTrail 追蹤](#)。
- 使用 AWS 加密解決方案，以及其中的所有預設安全控制 AWS 服務。
- 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 AWS 透過命令列界面或 API 存取 時需要 FIPS 140-3 驗證的密碼編譯模組，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 \(FIPS\) 140-3](#)。

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，例如名稱欄位。這包括當您使用 Amazon Keyspaces 或其他 AWS 服務 使用主控台、API AWS CLI或 AWS SDKs。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 Keyspaces 中的靜態加密](#)
- [在 Amazon Keyspaces 中傳輸中的加密](#)
- [Amazon Keyspaces 中的網際網路流量隱私權](#)

## Amazon Keyspaces 中的靜態加密

Amazon Keyspaces (適用於 Apache Cassandra) 靜態加密使用 [AWS Key Management Service \(AWS KMS\)](#) 中存放的加密金鑰來加密所有靜態資料，以提供增強的安全性。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，您可以建置安全敏感的應用程式，以滿足嚴格的資料保護合規和法規要求。

Amazon Keyspaces 靜態加密會使用 256 位元進階加密標準 (AES-256) 來加密您的資料。這有助於保護您的資料免於未經授權存取基礎儲存體。

Amazon Keyspaces 會以透明的方式加密和解密資料表中的資料和串流。Amazon Keyspaces 使用信封加密和金鑰階層來保護資料加密金鑰。它與 AWS KMS 整合，用於存放和管理根加密金鑰。如需加

密金鑰階層的詳細資訊，請參閱 [the section called “運作方式”](#)。如需信封加密等 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](#)。

建立新資料表時，您可以選擇下列其中一個 AWS KMS 金鑰 (KMS 金鑰)：

- AWS 擁有的金鑰 – 這是預設加密類型。金鑰由 Amazon Keyspaces 擁有（不收取額外費用）。
- 客戶受管金鑰 – 此金鑰會存放在您的帳戶中，並由您建立、擁有和管理。您可以完全控制客戶受管金鑰（需支付 AWS KMS 費用）。

Amazon Keyspaces 會使用與基礎資料表相同的金鑰自動加密變更資料擷取 (CDC) 串流。如需 CDC 的詳細資訊，請參閱 [the section called “使用 CDC 串流”](#)。

您可以隨時在 AWS 擁有的金鑰 和客戶受管金鑰之間切換。您可以在建立新資料表時指定客戶受管金鑰，或使用主控台或以程式設計方式使用 CQL 陳述式變更現有資料表的 KMS 金鑰。如要瞭解如何作業，請參閱 [靜態加密：如何使用客戶自管金鑰來加密 Amazon Keyspaces 中的表格](#)。

使用的預設選項 AWS 擁有的金鑰 進行靜態加密無需額外費用。不過，客戶受管金鑰會 AWS KMS 收取費用。如需定價的詳細資訊，請參閱 [AWS KMS 定價](#)。

Amazon Keyspaces 靜態加密適用於所有 AWS 區域，包括 AWS 中國（北京）和 AWS 中國（寧夏）區域。如需詳細資訊，請參閱 [靜態加密：在 Amazon Keyspaces 中運作的方式](#)。

## 主題

- [靜態加密：在 Amazon Keyspaces 中運作的方式](#)
- [靜態加密：如何使用客戶自管金鑰來加密 Amazon Keyspaces 中的表格](#)

## 靜態加密：在 Amazon Keyspaces 中運作的方式

Amazon Keyspaces（適用於 Apache Cassandra）靜態加密會使用 256 位元進階加密標準 (AES-256) 加密您的資料。這有助於保護您的資料免於未經授權存取基礎儲存體。根據預設，Amazon Keyspaces 資料表中的所有客戶資料都會靜態加密，且伺服器端加密是透明的，這表示不需要變更應用程式。

靜態加密與 AWS Key Management Service (AWS KMS) 整合，用於管理用來加密資料表的加密金鑰。建立新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 – 這是預設加密類型。金鑰由 Amazon Keyspaces 擁有（不收取額外費用）。
- 客戶受管金鑰 – 此金鑰存放在您的帳戶中，由您建立、擁有和管理。您可以完全控制客戶受管金鑰（需支付 AWS KMS 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 Keyspaces 資料。根據預設，Amazon Keyspaces 會使用 [AWS 擁有的金鑰](#)，這是在 Amazon Keyspaces 服務帳戶中建立和管理的多租戶加密金鑰。

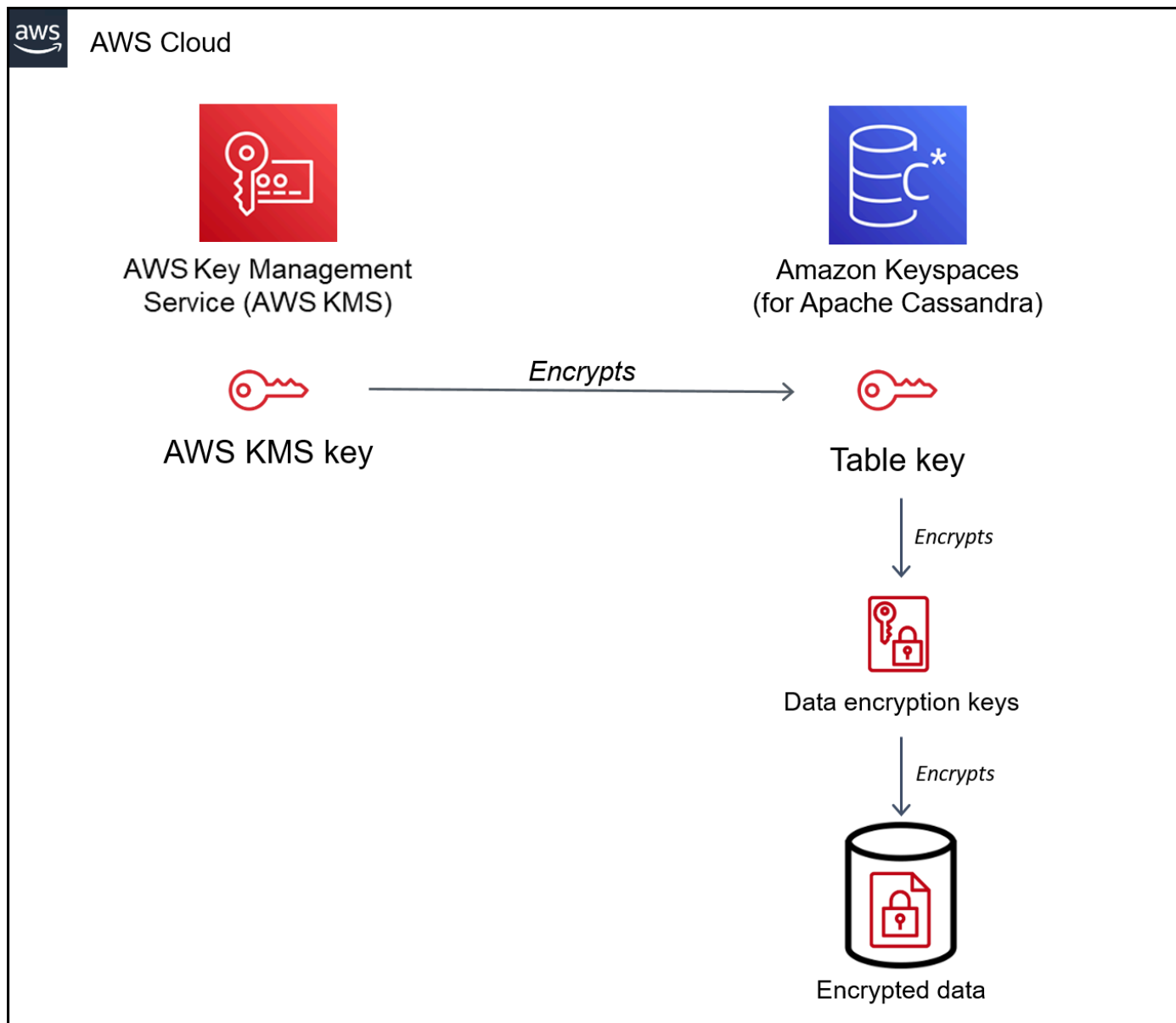
不過，您可以使用中的 [客戶受管金鑰](#) 來加密 Amazon Keyspaces 資料表 AWS 帳戶。您可以為金鑰空間中的每個資料表選取不同的 KMS 金鑰。您為資料表選取的 KMS 金鑰也會用來加密其所有中繼資料和可還原的備份。

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 Keyspaces 主控台或使用 [ALTER TABLE](#) 陳述式來變更資料表的 KMS 金鑰。切換 KMS 金鑰的程序是無縫的，不需要停機時間或導致服務降級。

### 金鑰階層

Amazon 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。它用於加密和解密 Amazon Keyspaces 資料表加密金鑰。資料表加密金鑰用於加密 Amazon Keyspaces 在內部使用的加密金鑰，以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，您可以對 KMS 金鑰進行變更，而無需重新加密資料或影響應用程式和持續的資料操作。



## 資料表索引鍵

Amazon 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內部資料加密金鑰，這些金鑰用於加密存放在資料表、日誌檔案和可還原備份中的資料。Amazon Keyspaces 會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。AWS KMS 資料金鑰是指 Amazon Keyspaces 中的資料表金鑰。



當您存取加密的資料表時，Amazon Keyspaces 會傳送請求至 AWS KMS，以使用 KMS 金鑰解密資料表金鑰。然後，它會使用純文字資料表金鑰來解密 Amazon Keyspaces 資料加密金鑰，並使用純文字資料加密金鑰來解密資料表資料。

Amazon Keyspaces 使用並將資料表金鑰和資料加密金鑰存放在外部 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，它會將加密的金鑰與加密的資料一起存放，以便他們可以隨需解密資料表資料。

## 資料表金鑰快取

為了避免 AWS KMS 呼叫每個 Amazon Keyspaces 操作，Amazon Keyspaces 會快取記憶體中每個連線的純文字資料表金鑰。如果 Amazon Keyspaces 在閒置五分鐘後收到快取資料表金鑰的請求，則會將新的請求傳送至 AWS KMS 以解密資料表金鑰。此呼叫會擷取自上次請求解密資料表金鑰以來，對 AWS KMS 或 AWS Identity and Access Management (IAM) 中 KMS 金鑰的存取政策所做的任何變更。

## 封套加密

如果您變更資料表的客戶受管金鑰，Amazon Keyspaces 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。它也會使用新的資料表金鑰來加密先前用於保護可還原備份的資料表金鑰。此程序稱為信封加密。這可確保即使您輪換客戶受管金鑰，也可以存取可還原的備份。如需信封加密的詳細資訊，請參閱《AWS Key Management Service 開發人員指南》中的[信封加密](#)。

## 主題

- [AWS 擁有的金鑰](#)
- [客戶受管金鑰](#)
- [靜態加密用量備註](#)

## AWS 擁有的金鑰

AWS 擁有的金鑰不會存放在您的 AWS 帳戶中。它們是 KMS 金鑰集合的一部分，這些金鑰 AWS 擁有和管理用於 multiple AWS 帳戶。AWS services 可用來 AWS 擁有的金鑰保護您的資料。

您無法檢視、管理或使用 AWS 擁有的金鑰或稽核其使用。不過，您不需要進行任何工作或變更任何程式，即可保護加密資料的金鑰。

您不需要支付每月費用或使用費 AWS 擁有的金鑰，也不會計入您帳戶的 AWS KMS 配額。

## 客戶受管金鑰

客戶受管金鑰是您 AWS 帳戶 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[自動金鑰輪換](#)，並在不再使用時[排定刪除客戶受管金鑰](#)。您可以為您管理的客戶受管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 AWS KMS 代表您傳送到 的請求。如需詳細資訊，請參閱[the section called “步驟 6：使用 設定監控 AWS CloudTrail”](#)。

客戶受管金鑰會針對每個 API 呼叫[產生費用](#)，而配額適用於這些 KMS AWS KMS 金鑰。如需詳細資訊，請參閱[AWS KMS 資源或請求配額](#)。

當您將客戶受管金鑰指定為資料表的根加密金鑰時，可還原備份會使用建立備份時為資料表指定的相同加密金鑰進行加密。如果輪換資料表的 KMS 金鑰，金鑰包圍可確保最新的 KMS 金鑰可存取所有可還原的備份。

Amazon Keyspaces 必須能夠存取您的客戶受管金鑰，才能讓您存取資料表資料。如果加密金鑰的狀態設定為停用或排定刪除，Amazon Keyspaces 將無法加密或解密資料。因此，您無法在資料表上執行讀取和寫入操作。一旦服務偵測到您的加密金鑰無法存取，Amazon Keyspaces 就會傳送電子郵件通知，提醒您。

您必須在七天內還原加密金鑰的存取權，否則 Amazon Keyspaces 會自動刪除您的資料表。基於預防措施，Amazon Keyspaces 會在刪除資料表之前建立資料表資料的可還原備份。Amazon Keyspaces 會維護可還原備份 35 天。35 天後，您無法再還原資料表資料。您不需要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備份，將資料還原至新的資料表。若要啟動還原，必須啟用用於資料表的最後一個客戶受管金鑰，Amazon Keyspaces 必須具有其存取權。

### Note

當您建立使用客戶受管金鑰加密的資料表時，在建立程序完成之前無法存取或排程刪除，會發生錯誤。建立資料表操作失敗，且您會收到電子郵件通知。

## 靜態加密用量備註

當您在 Amazon Keyspaces 中使用靜態加密時，請考慮下列事項。

- 所有 Amazon Keyspaces 資料表上都會啟用靜態伺服器端加密，且無法停用。整個資料表會靜態加密，您無法選取特定資料欄或資料列進行加密。
- 根據預設，Amazon 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。如果此金鑰不存在，則會為您建立。無法停用服務預設金鑰。
- 靜態加密只會在資料在持久性儲存媒體上靜態（靜態）時加密資料。如果資料安全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：您在 Amazon Keyspaces 中的所有資料都會在傳輸中加密。根據預設，Amazon Keyspaces 的通訊會使用 Secure Sockets Layer (SSL)/Transport Layer Security (TLS) 加密進行保護。
  - 使用中的資料：使用用戶端加密，在將資料傳送到 Amazon Keyspaces 之前保護資料。
  - 客戶受管金鑰：資料表中的靜態資料一律使用客戶受管金鑰加密。不過，在處理 AWS 擁有的金鑰期間執行多列原子更新的操作會暫時使用來加密資料。這包括範圍刪除操作和同時存取靜態和非靜態資料的操作。
- 單一客戶受管金鑰最多可有 50,000 個[授予](#)。與客戶受管金鑰相關聯的每個 Amazon Keyspaces 資料表都會耗用 2 個授與。刪除資料表時，會釋出一個授予。第二個授權用於建立資料表的自動快照，以防止資料遺失，以防 Amazon Keyspaces 意外遺失對客戶受管金鑰的存取。此授與會在刪除資料表後 42 天發行。

## 靜態加密：如何使用客戶自管金鑰來加密 Amazon 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 Keyspaces 中現有資料表的加密金鑰。下列主題概述如何實作新資料表和現有資料表的客戶受管金鑰。

### 主題

- [先決條件：使用 建立客戶受管金鑰，AWS KMS 並將許可授予 Amazon Keyspaces](#)
- [步驟 3：指定新資料表的客戶受管金鑰](#)
- [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 設定監控 AWS CloudTrail](#)

先決條件：使用 建立客戶受管金鑰，AWS KMS 並將許可授予 Amazon Keyspaces

您必須先在 AWS Key Management Service (AWS KMS) 中建立金鑰，然後授權 Amazon Keyspaces 使用該金鑰，才能使用[客戶受管金鑰](#)來保護 Amazon Keyspaces 資料表。

步驟 1：使用 建立客戶受管金鑰 AWS KMS

若要建立用於保護 Amazon Keyspaces 資料表的客戶受管金鑰，您可以遵循使用主控台或 AWS API [建立對稱加密 KMS 金鑰](#)中的步驟。

步驟 2：授權使用客戶受管金鑰

在您可以選擇[客戶受管金鑰](#)來保護 Amazon Keyspaces 資料表之前，該客戶受管金鑰上的政策必須授予 Amazon Keyspaces 代表您使用它的許可。您可以完全控制客戶受管金鑰上的政策和授予。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，即可使用預設值[AWS 擁有的金鑰](#)來保護您 AWS 帳戶中的 Amazon Keyspaces 資料表。

下列主題說明如何使用 IAM 政策和授予來設定必要的許可，以允許 Amazon Keyspaces 資料表使用客戶受管金鑰。

#### 主題

- [客戶受管金鑰的金鑰政策](#)
- [範例金鑰政策](#)
- [使用授予授權 Amazon Keyspaces](#)

#### 客戶受管金鑰的金鑰政策

當您選取[客戶受管金鑰](#)來保護 Amazon Keyspaces 資料表時，Amazon Keyspaces 會取得代表進行選取之主體使用客戶受管金鑰的許可。該委託人、使用者或角色必須擁有 Amazon Keyspaces 所需客戶受管金鑰的許可。

Amazon Keyspaces 至少需要客戶受管金鑰的下列許可：

- [kms:Encrypt](#)
- [kms:Decrypt](#)
- [kms:ReEncrypt\\*](#) (適用於 [kms:ReEncryptFrom](#) and [kms:ReEncryptTo](#))
- [kms:GenerateDataKey\\*](#) (適用於 [kms:GenerateDataKey](#) 和 [kms:GenerateDataKeyWithoutPlaintext](#))

- [kms:DescribeKey](#)
- [kms: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 在密碼編譯操作中使用客戶受管金鑰並建立授予，但僅限於代表帳戶中有權使用 Amazon Keyspaces 的主體時。如果政策陳述式中指定的委託人沒有使用 Amazon Keyspaces 的許可，呼叫會失敗，即使它來自 Amazon Keyspaces 服務。
- [kms:ViaService](#) 條件金鑰只有在請求來自 Amazon Keyspaces 時，才允許許可，這些請求代表政策陳述式中列出的委託人。這些主體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，才能獨立於任何特定項目 AWS 區域。
- 授予客戶受管金鑰管理員（可擔任該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 Keyspaces 保護資料表所需的授予](#)。
- 授予 Amazon Keyspaces 對客戶受管金鑰的唯讀存取權。在這種情況下，Amazon Keyspaces 可以直接呼叫這些操作。它不必代表帳戶委託人採取行動。

使用範例金鑰政策之前，請將範例主體取代為來自您的實際主體 AWS 帳戶。

```
{
 "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": "*"
 }
]
}

```

## 使用授予授權 Amazon Keyspaces

除了金鑰政策之外，Amazon Keyspaces 還使用授予來設定客戶受管金鑰的許可。若要檢視您帳戶中客戶受管金鑰的授予，請使用 [ListGrants](#) 操作。Amazon Keyspaces 不需要授予或任何其他許可，即可使用 [AWS 擁有的金鑰](#) 來保護您的資料表。

Amazon Keyspaces 會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。如果帳戶包含以相同客戶受管金鑰加密的多個資料表，則會授予每個資料表的每種類型。[Amazon Keyspaces 加密內容](#) 會限制授予，其中包含資料表名稱和 AWS 帳戶 ID。授予包含不再需要時 [淘汰授予](#) 的許可。

若要建立授予，Amazon Keyspaces 必須具有 CreateGrant 代表建立加密資料表的使用者呼叫 的許可。

金鑰政策也會允許帳戶 [撤銷客戶受管金鑰上的授予](#)。不過，如果您撤銷作用中加密資料表的授予，Amazon Keyspaces 將無法保護和維護資料表。



### 步驟 3：指定新資料表的客戶受管金鑰

請依照下列步驟，使用 Amazon Keyspaces 主控台或 CQL 在新資料表上指定客戶受管金鑰。

使用客戶受管金鑰建立加密的資料表（主控台）

1. 登入 AWS 管理主控台，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. 在資料表詳細資訊區段的建立資料表頁面上，選取金鑰空間，並提供新資料表的名稱。
4. 在結構描述區段中，建立資料表的結構描述。
5. 在資料表設定區段中，選擇自訂設定。
6. 繼續至加密設定。

在此步驟中，您會選取資料表的加密設定。

在選擇 AWS KMS key 下的靜態加密區段中，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 Resource Name (ARN)。

#### Note

如果您選取的金鑰無法存取或缺少必要的許可，請參閱 [《開發人員指南》中的對金鑰存取進行故障診斷](#)。AWS Key Management Service

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，您可以使用 CREATE TABLE 陳述式，如下列範例所示。請務必使用授予 Amazon Keyspaces 之許可的有效金鑰的 ARN 取代金鑰 ARN。

```
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-1:555555555555: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有效，且 Amazon 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》中的對[金鑰存取進行故障診斷](#)。

#### 步驟 4：更新現有資料表的加密金鑰

您也可以使用 Amazon Keyspaces 主控台或 CQL 隨時變更 AWS 擁有的金鑰 和客戶受管 KMS 金鑰之間現有資料表的加密金鑰。

使用新的客戶受管金鑰更新現有的資料表（主控台）

1. 登入 AWS 管理主控台，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)。
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資料表的加密設定。

在選擇 AWS KMS key 下，選擇選項 選擇不同的 KMS 金鑰（進階），然後在搜尋欄位中選擇 AWS KMS key 或輸入 Amazon Resource Name (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[對金鑰存取進行故障診斷](#)。

或者，您可以 AWS 擁有的金鑰 為使用客戶受管金鑰加密的資料表選擇。

5. 選擇儲存變更以儲存您對資料表的變更。

#### 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受管金鑰。請務必使用授予 Amazon Keyspaces 之許可的有效金鑰的 ARN 取代金鑰 ARN。

```
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-1:555555555555:key/11111111-1111-1111-1111-111111111111'
 }
}
```



```
 }
};
```

如果您收到 `Invalid Request Exception`，則需要確認客戶受管金鑰有效，且 Amazon 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》中的對[金鑰存取進行故障診斷](#)。

若要使用 將加密金鑰變更回預設靜態加密選項 AWS 擁有的金鑰，您可以使用 `ALTER TABLE` 陳述式，如下列範例所示。

```
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。當您在加密資料的請求中包含加密內容時，會以 AWS KMS 加密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 AWS KMS 密碼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 Amazon Keyspaces 資料表，您可以使用加密內容來識別稽核記錄和日誌中客戶受管金鑰的使用。它也會以純文字顯示在日誌中，例如 [AWS CloudTrail](#)和 [Amazon CloudWatch Logs](#) 的日誌中。

在對的請求中 AWS KMS，Amazon Keyspaces 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"
}
```

- 金鑰空間 – 第一個金鑰/值對會識別包含 Amazon Keyspaces 正在加密之資料表的金鑰空間。金鑰為 `aws:cassandra:keyspaceName`。值是金鑰空間的名稱。

```
"aws:cassandra:keyspaceName": "<keyspace-name>"
```

例如：

```
"aws:cassandra:keyspaceName": "my_keyspace"
```

- 資料表 – 第二個金鑰/值對可識別 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 設定監控 AWS CloudTrail

如果您使用[客戶受管金鑰](#)來保護 Amazon Keyspaces 資料表，您可以使用 AWS CloudTrail 日誌來追蹤 Amazon Keyspaces AWS KMS 代表您傳送給 的請求。

本節會討論 `GenerateDataKey`、`Decrypt`、`DescribeKey`和 `CreateGrant`請求。此外，Amazon Keyspaces 會使用 [RetireGrant](#) 操作，在刪除資料表時移除授予。

### Note

當您使用 Amazon Keyspaces 時，某些操作可能會產生具有 `invokedBy` 欄位的 CloudTrail 事件 `dynamodb.amazonaws.com`。這是預期會發生的，因為 Amazon Keyspaces 與 Amazon DynamoDB 整合以提供其服務。

## GenerateDataKey

Amazon Keyspaces 會建立唯一的資料表金鑰來加密靜態資料。它會將 [GenerateDataKey](#) 請求傳送至 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 Keyspaces 服務帳戶。這些參數包括客戶受管金鑰的 Amazon Resource Name (ARN)、需要 256 位元金鑰的金鑰指標，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-1:555555555555: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-1:555555555555:key/11111111-1111-111-1111-111111111111"
 }
],
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "eventCategory": "Management",
 "recipientAccountId": "123SAMPLE012",
 "sharedEventID": "84fbaaf0-9641-4e32-9147-57d2cb08792e"
}
```

```
}

```

## DescribeKey

Amazon Keyspaces 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中。

記錄 DescribeKey 操作的事件類似於以下範例事件。使用者是 Amazon Keyspaces 服務帳戶。這些參數包括客戶受管金鑰的 ARN，以及需要 256 位元金鑰的金鑰指標。

```
{
 "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-1:555555555555: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-1:555555555555:key/11111111-1111-111-1111-111111111111"
 }
],
"eventType": "AwsApiCall",
"managementEvent": true,
"eventCategory": "Management",
"recipientAccountId": "123SAMPLE012"
}

```

## 解密

當您存取 Amazon Keyspaces 資料表時，Amazon Keyspaces 需要解密資料表金鑰，以便在階層中解密其下方的金鑰。然後解密資料表中的資料。若要解密資料表金鑰，Amazon Keyspaces 會將[解密](#)請求傳送至 AWS KMS，以指定資料表的 KMS 金鑰。

記錄 Decrypt 操作的事件類似於以下範例事件。使用者是中存取資料表 AWS 帳戶的主體。這些參數包括加密的資料表金鑰（做為加密文字 Blob），以及識別資料表和的[加密內容](#) AWS 帳戶。會從加密文字 AWS KMS 衍生客戶受管金鑰的 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-1:555555555555: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 Amazon Keyspaces 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和維護和耐久性任務。在上不需要這些授權[AWS 擁有的金鑰](#)。

Amazon Keyspaces 建立的授予專屬於資料表。[CreateGrant](#) 請求中的主體是建立資料表的使用者。

記錄 CreateGrant 操作的事件類似於以下範例事件。這些參數包括資料表的客戶受管金鑰 ARN、承授者委託人和淘汰委託人 (Amazon Keyspaces 服務)，以及授予涵蓋的操作。它還包含需要所有加密操作使用指定[加密內容](#)的限制。

```

{
 "eventVersion": "1.08",
 "userIdentity": {
 "type": "IAMUser",
 "principalId": "AIDAZ3FNIIVIZZ6H7CFQG",
 "arn": "arn:aws:iam::arn:aws:kms:eu-west-1:555555555555:key/11111111-1111-111-1111-111111111111:user/admin",
 "accountId": "arn:aws:kms:eu-west-1:555555555555: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-1:555555555555:key/11111111-1111-111-1111-111111111111"
 },
 "eventType": "AwsApiCall",
 "managementEvent": true,
 "eventCategory": "Management",
 "recipientAccountId": "123SAMPLE012"
}
```

## 在 Amazon Keyspaces 中傳輸中的加密

Amazon Keyspaces 只接受使用 Transport Layer Security (TLS) 的安全連線。傳輸中加密可在資料往返 Amazon Keyspaces 時加密資料，以提供多一層的資料保護。組織政策、產業或政府法規以及合規要求通常需要使用傳輸中的加密，以在應用程式透過網路傳輸資料時提高其資料安全性。

若要了解如何使用 TLS 加密 Amazon Keyspaces 的 `cqlsh` 連線，請參閱 [the section called “如何手動設定 TLS 的 `cqlsh` 連線”](#)。若要了解如何搭配用戶端驅動程式使用 TLS 加密，請參閱 [the section called “使用 Cassandra 用戶端驅動程式”](#)。

## Amazon Keyspaces 中的網際網路流量隱私權

本主題說明 Amazon Keyspaces (適用於 Apache Cassandra) 如何保護從內部部署應用程式到 Amazon Keyspaces 以及 Amazon Keyspaces 和相同 內其他 AWS 資源之間的連線 AWS 區域。

服務和內部部署用戶端與應用程式之間的流量。

您的私有網路與 之間有兩個連線選項 AWS：

- 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 ?](#)。
- Direct Connect 連線。如需詳細資訊，請參閱《Direct Connect 使用者指南》中的 [什麼是 Direct Connect ?](#)。

Amazon Keyspaces (適用於 Apache Cassandra) 是受管服務，受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及如何 AWS 保護基礎設施的資訊，請參閱 [AWS 雲端安全](#)。若要使用基礎設施安



全的最佳實務設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的[基礎設施保護](#)。

您可以使用 AWS 發佈的 API 呼叫，透過網路存取 Amazon Keyspaces。使用者端必須支援下列專案：

- 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。

Amazon Keyspaces 支援兩種驗證用戶端請求的方法。第一個方法使用服務特定的登入資料，這是針對特定 IAM 使用者產生的密碼型登入資料。您可以使用 IAM 主控台、AWS CLI或 AWS API 來建立和管理密碼。如需詳細資訊，請參閱[搭配 Amazon Keyspaces 使用 IAM](#)。

第二個方法針對 Cassandra 的開放原始碼 DataStax Java 驅動程式使用身分驗證外掛程式。此外掛程式可讓 [IAM 使用者、角色和聯合身分](#) 使用 [AWS Signature 第 4 版程序 \(SigV4\)](#)，將身分驗證資訊新增至 [Amazon Keyspaces \(適用於 Apache Cassandra\) API](#) 請求。如需詳細資訊，請參閱[the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

## 相同區域中 AWS 資源之間的流量

介面 VPC 端點可讓您在 Amazon VPC 和 Amazon Keyspaces 中執行的虛擬私有雲端 (VPC) 之間進行私有通訊。介面 VPC 端點由提供支援 AWS PrivateLink，該 AWS 服務可啟用 VPCs和 AWS 服務之間的私有通訊。透過在您的 VPC 中使用具有私有 IPs彈性網路介面來 AWS PrivateLink 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 連線或 Direct Connect 連線。如需詳細資訊，請參閱 [Amazon Virtual Private Cloud](#) 和[界面 VPC 端點 \(AWS PrivateLink\)](#)。如需範例政策，請參閱 [the section called “使用 Amazon Keyspaces 的介面 VPC 端點”](#)。

## AWS Identity and Access Management 適用於 Amazon Keyspaces

AWS Identity and Access Management (IAM) 是一種 AWS 服務，可協助管理員安全地控制對 AWS 資源的存取。IAM 管理員可控制誰可以進行身分驗證（登入）和授權（具有許可），以使用 Amazon Keyspaces 資源。IAM 是您可以免費使用 AWS 服務的。

### 主題

- [目標對象](#)

- [使用身分驗證](#)
- [使用政策管理存取權](#)
- [Amazon Keyspaces 如何與 IAM 搭配使用](#)
- [Amazon Keyspaces 身分型政策範例](#)
- [AWS Amazon Keyspaces 的 受管政策](#)
- [對 Amazon Keyspaces 身分和存取進行故障診斷](#)
- [使用 Amazon Keyspaces 的服務連結角色](#)

## 目標對象

您的使用方式 AWS Identity and Access Management (IAM) 會根據您的角色而有所不同：

- 服務使用者 — 若無法存取某些功能，請向管理員申請所需許可 (請參閱 [對 Amazon Keyspaces 身分和存取進行故障診斷](#))
- 服務管理員 — 負責設定使用者存取權並提交相關許可請求 (請參閱 [Amazon Keyspaces 如何與 IAM 搭配使用](#))
- IAM 管理員 — 撰寫政策以管理存取控制 (請參閱 [Amazon Keyspaces 身分型政策範例](#))

## 使用身分驗證

身分驗證是您 AWS 使用身分憑證登入 的方式。您必須以 AWS 帳戶根使用者、IAM 使用者或擔任 IAM 角色身分進行身分驗證。

您可以使用身分來源的登入資料，例如 AWS IAM Identity Center (IAM Identity Center)、單一登入身分驗證或 Google/Facebook 登入資料，以聯合身分的形式登入。如需有關登入的詳細資訊，請參閱《AWS 登入 使用者指南》中的[如何登入您的 AWS 帳戶](#)。

對於程式設計存取，AWS 提供 SDK 和 CLI 以密碼編譯方式簽署請求。如需詳細資訊，請參閱《IAM 使用者指南》中的[API 請求的AWS 第 4 版簽署程序](#)。

## AWS 帳戶 根使用者

當您建立 時 AWS 帳戶，您會從一個名為 AWS 帳戶 theroot 使用者的登入身分開始，該身分具有對所有 AWS 服務 和 資源的完整存取權。強烈建議不要使用根使用者來執行日常任務。有關需要根使用者憑證的任務，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[https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_users.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_users.html)是一種身分具備單人或應用程式的特定許可權。建議以臨時憑證取代具備長期憑證的 IAM 使用者。如需詳細資訊，請參閱《IAM 使用者指南》中的[要求人類使用者使用聯合身分提供者來 AWS 使用臨時憑證存取](#)。

[IAM 群組](#)會指定 IAM 使用者集合，使管理大量使用者的許可權更加輕鬆。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM 使用者的使用案例](#)。

## IAM 角色

IAM 角色[https://docs.aws.amazon.com/IAM/latest/UserGuide/id\\_roles.html](https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles.html)的身分具有特定許可權，其可以提供臨時憑證。您可以透過[從使用者切換到 IAM 角色（主控台）](#)或呼叫 AWS CLI 或 AWS API 操作來擔任角色。如需詳細資訊，請參閱《IAM 使用者指南》中的[擔任角色的方法](#)。

IAM 角色適用於聯合身分使用者存取、臨時 IAM 使用者許可、跨帳戶存取權與跨服務存取，以及在 Amazon EC2 執行的應用程式。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM 中的快帳戶資源存取](#)。

## 使用政策管理存取權

您可以透過建立政策並將其連接到身分或資源 AWS 來控制 AWS 中的存取。政策定義與身分或資源相關聯的許可。當委託人提出請求時 AWS，會評估這些政策。大多數政策會以 JSON 文件 AWS 形式存放在中。如需進一步了解 JSON 政策文件，請參閱《IAM 使用者指南》中的[JSON 政策概觀](#)。

管理員會使用政策，透過定義哪些主體可在哪些條件下對哪些資源執行動作，以指定可存取的範圍。

預設情況下，使用者和角色沒有許可。IAM 管理員會建立 IAM 政策並將其新增至角色，供使用者後續擔任。IAM 政策定義動作的許可，無論採用何種方式執行。

## 身分型政策

身分型政策是附加至身分 (使用者、使用者群組或角色) 的 JSON 許可政策文件。這類政策控制身分可對哪些資源執行哪些動作，以及適用的條件。如需了解如何建立身分型政策，請參閱《IAM 使用者指南》中的[透過客戶管理政策定義自訂 IAM 許可](#)。

身分型政策可分為內嵌政策 (直接內嵌於單一身分) 與受管政策 (可附加至多個身分的獨立政策)。如需了解如何在受管政策及內嵌政策之間做選擇，請參閱《IAM 使用者指南》中的[在受管政策與內嵌政策之間選擇](#)。

## 資源型政策

資源型政策是附加到資源的 JSON 政策文件。範例包括 IAM 角色信任政策與 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。您必須在資源型政策中[指定主體](#)。

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來自 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 AWS WAF 和 Amazon VPC 是支援 ACLs 的服務範例。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 支援其他政策類型，可設定更多常見政策類型授予的最大許可：

- 許可界限 — 設定身分型政策可授與 IAM 實體的最大許可。如需詳細資訊，請參閱《IAM 使用者指南》中的[IAM 實體許可界限](#)。
- 服務控制政策 (SCP) — 為 AWS Organizations 中的組織或組織單位指定最大許可。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的[服務控制政策](#)。
- 資源控制政策 (RCP) — 設定您帳戶中資源可用許可的上限。如需詳細資訊，請參閱《AWS Organizations 使用者指南》中的[資源控制政策 \(RCP\)](#)。
- 工作階段政策 — 在以程式設計方式為角色或聯合身分使用者建立臨時工作階段時，以參數形式傳遞的進階政策。如需詳細資訊，請參《IAM 使用者指南》中的[工作階段政策](#)。

## 多種政策類型

當多種類型的政策適用於請求時，產生的許可會更複雜而無法理解。若要了解如何 AWS 在涉及多個政策類型時決定是否允許請求，請參閱《IAM 使用者指南》中的[政策評估邏輯](#)。

## Amazon Keyspaces 如何與 IAM 搭配使用

在您使用 IAM 管理對 Amazon Keyspaces 的存取之前，您應該了解哪些 IAM 功能可與 Amazon Keyspaces 搭配使用。若要全面了解 Amazon Keyspaces 和其他 AWS 服務如何與 IAM 搭配使用，請參閱《[AWS IAM 使用者指南](#)》中的[與 IAM 搭配使用的服務](#)。

## 主題

- [Amazon Keyspaces 身分型政策](#)
- [Amazon Keyspaces 資源型政策](#)
- [以 Amazon Keyspaces 標籤為基礎的授權](#)
- [Amazon Keyspaces IAM 角色](#)

## Amazon Keyspaces 身分型政策

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 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 Keyspaces 服務特定的資源和動作，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 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。

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。政策會使用動作來授予執行相關聯動作的許可。

Amazon Keyspaces 中的政策動作在動作之前使用下列字首：cassandra:。例如，若要授予某人使用 Amazon Keyspaces CREATE CQL 陳述式建立 Amazon Keyspaces 金鑰空間的許可，請在其政策中包含 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，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"
]
```

若要查看 Amazon Keyspaces 動作的清單，請參閱《服務授權參考》中的 [Amazon Keyspaces \(適用於 Apache Cassandra\) 定義的動作](#)。

## Resources

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。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。最佳實務是使用其 [Amazon Resource Name \(ARN\)](#) 來指定資源。若動作不支援資源層級許可，使用萬用字元 (\*) 表示該陳述式適用於所有資源。

```
"Resource": ""
```

在 Amazon Keyspaces 中，金鑰空間、資料表和串流可用於 IAM 許可的 Resource 元素。

### Note

若要存取 Amazon Keyspaces 中的使用者金鑰空間和資料表，您的 IAM 政策必須包含系統資料表的 `cassandra:Select` 許可：

```
arn:${Partition}:cassandra:${Region}:${Account}:/keyspace/system*
```

這適用於下列案例：

- AWS 管理主控台存取
- SDK 資源操作，例如 `GetKeyspace`、`ListKeyspaces`、`GetTable` 和 `ListTables`
- 標準 Apache Cassandra 用戶端驅動程式連線，因為驅動程式會在連線初始化期間自動讀取系統資料表

系統資料表為唯讀且無法修改。

Amazon Keyspaces 金鑰空間資源具有下列 ARN：

```
arn:${Partition}:cassandra:${Region}:${Account}:/keyspace/${keyspaceName}/
```

Amazon Keyspaces 資料表資源具有下列 ARN：

```
arn:${Partition}:cassandra:${Region}:${Account}:/keyspace/${keyspaceName}/table/${tableName}
```

Amazon Keyspaces 串流資源具有下列 ARN：



```
arn:${Partition}:cassandra:{Region}:${Account}:/keyspace/${keyspaceName}/table/
${tableName}/stream/${streamLabel}
```

如需 ARNs 格式的詳細資訊，請參閱 [Amazon Resource Name \(ARNs AWS 和服務命名空間\)](#)。

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-1:111122223333:/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-1:111122223333:/keyspace/*"
```

有些 Amazon Keyspaces 動作無法在特定資源上執行，例如用於建立資源的動作。在這些情況下，您必須使用萬用字元 (\*)。

```
"Resource": "*"
```

例如，若要將SELECT許可授予 mytable中的 IAM 委託人mykeyspace，委託人必須具有讀取 mytable和 的許可keyspace/system\*。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*"
```

若要查看 Amazon Keyspaces 資源類型及其 ARNs，請參閱《服務授權參考》中的 [Amazon Keyspaces \(適用於 Apache Cassandra\) 定義的資源](#)。若要了解您可以使用哪些動作指定每個資源的 ARN，請參閱 [Amazon Keyspaces 定義的動作 \(適用於 Apache Cassandra\)](#)。

## 條件索引鍵

管理員可以使用 AWS JSON 政策來指定誰可以存取內容。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 元素會根據定義的條件，指定陳述式的執行時機。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。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的[AWS 全域條件內容索引鍵](#)。

Amazon Keyspaces 會定義自己的一組條件金鑰，也支援使用一些全域條件金鑰。若要查看所有 AWS 全域條件索引鍵，請參閱《IAM 使用者指南》中的[AWS 全域條件內容索引鍵](#)。

所有 Amazon Keyspaces 動作都支援 `aws:RequestTag/${TagKey}`、`aws:ResourceTag/${TagKey}` 和 `aws:TagKeys` 條件金鑰。如需詳細資訊，請參閱[the section called “根據標籤的 Amazon Keyspaces 資源存取”](#)。

若要查看 Amazon Keyspaces 條件金鑰清單，請參閱服務授權參考中的 [Amazon Keyspaces \(適用於 Apache Cassandra\) 的條件金鑰](#)。若要了解您可以使用條件金鑰的動作和資源，請參閱 [Amazon Keyspaces 定義的動作 \(適用於 Apache Cassandra\)](#)。

## 範例

若要檢視 Amazon Keyspaces 身分型政策的範例，請參閱 [Amazon Keyspaces 身分型政策範例](#)。

## 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>。

## 以 Amazon Keyspaces 標籤為基礎的授權

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 `aws:RequestTag/key-name`、`cassandra:ResourceTag/key-name` 或 [條件索引鍵，在政策的條件元素](#) 中提供標籤資訊。aws:TagKeys 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「[根據標籤的 Amazon Keyspaces 資源存取](#)」。

## Amazon Keyspaces IAM 角色

[IAM 角色](#) 是具有特定許可 AWS 帳戶 的實體。

### 搭配 Amazon Keyspaces 使用臨時憑證

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 [GetFederationToken](#) 等 AWS STS API 操作來取得臨時安全登入資料。

Amazon Keyspaces 支援使用臨時憑證搭配 Github 儲存庫提供的 AWS Signature 第 4 版 (SigV4) 身分驗證外掛程式，適用於下列語言：



- Java : <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>。
- Go : <https://github.com/aws/aws-sigv4-auth-cassandra-gocql-driver-plugin>。

如需實作身分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 Cassandra 用戶端驅動程式”](#)。

## 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 身分型政策範例

根據預設，IAM 使用者和角色沒有建立或修改 Amazon Keyspaces 資源的許可。他們也無法使用 主控台 AWS CLI、CQLSH 或 AWS API 來執行任務。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 主控台](#)
- [允許使用者檢視他們自己的許可](#)
- [存取 Amazon Keyspaces 資料表](#)
- [根據標籤的 Amazon Keyspaces 資源存取](#)

## 政策最佳實務

身分型政策會判斷您帳戶中的某個人員是否可以建立、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 受管政策並邁向最低權限許可 – 若要開始將許可授予您的使用者和工作負載，請使用將許可授予許多常見使用案例的 AWS 受管政策。它們可在您的 中使用 AWS 帳戶。我們建議您定義特定於使用案例 AWS 的客戶受管政策，以進一步減少許可。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。如果透過特定 等使用服務動作 AWS 服務，您也可以使用條件來授予其存取權 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，可協助您撰寫安全且實用的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 [使用 IAM Access Analyzer 驗證政策](#)。
- 需要多重要素驗證 (MFA) – 如果您的案例需要 IAM 使用者或 中的根使用者 AWS 帳戶，請開啟 MFA 以提高安全性。如需在呼叫 API 操作時請求 MFA，請將 MFA 條件新增至您的政策。如需詳細資訊，請參閱《IAM 使用者指南》中的 [透過 MFA 的安全 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 主控台

Amazon Keyspaces 不需要特定許可即可存取 Amazon Keyspaces 主控台。您需要至少 個唯讀許可，才能列出和檢視 中 Amazon Keyspaces 資源的詳細資訊 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 Amazon Keyspaces 主控台存取的實體使用。

- [AmazonKeyspacesReadOnlyAccess\\_v2](#) – 此政策授予 Amazon Keyspaces 的唯讀存取權。
- [AmazonKeyspacesFullAccess](#) – 此政策授予許可，以使用具有所有功能完整存取權的 Amazon Keyspaces。

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。此政策包含在主控台或使用或 AWS CLI AWS API 以程式設計方式完成此動作的許可。

```
{
 "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": "*"
 }
]
}
```

## 存取 Amazon Keyspaces 資料表

### Note

若要存取 Amazon Keyspaces 中的使用者金鑰空間和資料表，您的 IAM 政策必須包含系統資料表的 `cassandra:Select` 許可：

```
arn:${Partition}:cassandra:${Region}:${Account}:/keyspace/system*
```

這適用於下列案例：

- AWS 管理主控台存取
- SDK 資源操作，例如 `GetKeyspace`、`ListKeyspaces`、`GetTable` 和 `ListTables`
- 標準 Apache Cassandra 用戶端驅動程式連線，因為驅動程式會在連線初始化期間自動讀取系統資料表

系統資料表為唯讀且無法修改。

以下是授予 Amazon Keyspaces 系統資料表唯讀 (SELECT) 存取權的範例政策。對於所有範例，請將 Amazon Resource Name (ARN) 中的區域和帳戶 ID 取代為您自己的 ID。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "cassandra:Select"
],
 "Resource": [
 "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
]
 }
]
}
```

下列範例政策會將唯讀存取權新增至金鑰空間 `mytable` 中的使用者資料表 `mykeyspace`。

```
{
 "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*"
]
 }
]
}
```

下列範例政策會將讀取/寫入存取權指派給使用者資料表，並將讀取存取權指派給系統資料表。

```
{
 "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*"
]
 }
]
}
```

下列範例政策允許使用者在金鑰空間 中建立資料表mykeyspace。

```
{
 "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*"
]
}
```

下列範例政策會指派系統資料表的讀取存取權，但會限制 SELECT（讀取）和 MODIFY（寫入）對使用者資料表的存取權mytable。

```
{
 "Version": "2012-10-17",
 "Statement": [
 {
 "Effect": "Allow",
 "Action": [
 "cassandra:Select"
],
 "Resource": [
 "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
]
 },
 {
 "Effect": "Deny",
 "Action": [
 "cassandra:Select",
 "cassandra:Modify"
],
 "Resource": [
 "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/mytable"
]
 }
]
}
```

## 根據標籤的 Amazon Keyspaces 資源存取

您可以在身分型政策中使用條件，根據標籤控制對 Amazon Keyspaces 資源的存取。這些政策會控制帳戶中金鑰空間和資料表的可見性。請注意，與透過 Cassandra 驅動程式和開發人員工具進行的 Cassandra 查詢語言 (CQL) API 呼叫相比，使用 AWS SDK 提出請求時，系統資料表的標籤型許可會有不同的行為。

- 若要在使用標籤型存取時透過 AWS SDK 提出 Get List 和資源請求，呼叫者需要具有系統資料表的讀取存取權。例如，需要 Select 動作許可，才能透過 GetTable 操作從系統資料表讀取資料。如果發起人只有對特定資料表的標籤型存取，則需要額外存取系統資料表的操作將會失敗。
- 為了相容於已建立的 Cassandra 驅動程式行為，使用 Cassandra 查詢語言 (CQL) API 呼叫透過 Cassandra 驅動程式和開發人員工具在系統資料表上執行操作時，不會強制執行標籤型授權政策。

下列範例示範如何建立政策，在資料表的 Owner 包含該使用者名稱的值時，將檢視資料表的許可授予使用者。在此範例中，您也提供系統資料表的讀取存取權。

```
{
 "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，則該資料表必須加上標籤 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](#)。

如果資料表的 Owner 包含該使用者的使用者名稱值，則下列政策會授予使用者使用標籤建立資料表的許可。

```
{
 "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 Amazon Keyspaces 的 受管政策

AWS 受管政策是由 AWS 受管政策建立和管理的獨立政策旨在為許多常用案例提供許可，以便您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予特定使用案例的最低權限許可，因為這些許可可供所有 AWS 客戶使用。我們建議您定義特定於使用案例的[客戶管理政策](#)，以便進一步減少許可。

您無法變更 AWS 受管政策中定義的許可。如果 AWS 更新受 AWS 管政策中定義的許可，則更新會影響政策連接的所有主體身分（使用者、群組和角色）。當新的 AWS 服務 啟動或新的 API 操作可用於現有服務時，AWS 最有可能更新 AWS 受管政策。

如需詳細資訊，請參閱 IAM 使用者指南中的 [AWS 受管政策](#)。



## AWS 受管政策：AmazonKeyspacesReadOnlyAccess\_v2

您可將 AmazonKeyspacesReadOnlyAccess\_v2 政策連接到 IAM 身分。

此政策授予 Amazon 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 Amazon Keyspaces 的唯讀存取權。
- Amazon Keyspaces CDC streams – 允許主體檢視 Amazon Keyspaces CDC 串流。
- Application Auto Scaling – 允許主體檢視來自 Application Auto Scaling 的設定。這是必要的，讓使用者能檢視附加於資料表的自動擴展政策。
- CloudWatch – 允許主體檢視於 CloudWatch 中設定的指標資料與警示。這是必要的，以便使用者檢視資料表的計費大小與 CloudWatch 警示。
- AWS KMS – 允許主體檢視其中設定的金鑰 AWS KMS。這是必要的，因此使用者可以檢視在其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，以查詢 Amazon EC2 執行個體上的 VPC 以取得端點和網路介面資訊。需要對 Amazon EC2 執行個體的唯讀存取權，以便 Amazon Keyspaces 可以在用於連線負載平衡的 `system.peers` 資料表中查詢和存放可用的介面 VPC 端點。

若要檢閱 JSON 格式的政策，請參閱 [AmazonKeyspacesReadOnlyAccess\\_v2](#)。

## AWS 受管政策：AmazonKeyspacesReadOnlyAccess

您可將 AmazonKeyspacesReadOnlyAccess 政策連接到 IAM 身分。

此政策授予 Amazon 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 Amazon Keyspaces 的唯讀存取權。
- Amazon Keyspaces CDC streams – 允許主體檢視 Amazon Keyspaces CDC 串流。
- Application Auto Scaling – 允許主體檢視來自 Application Auto Scaling 的設定。這是必要的，讓使用者能檢視附加於資料表的自動擴展政策。
- CloudWatch – 允許主體檢視於 CloudWatch 中設定的指標資料與警示。這是必要的，以便使用者檢視資料表的計費大小與 CloudWatch 警示。
- AWS KMS – 允許主體檢視其中設定的金鑰 AWS KMS。這是必要的，因此使用者可以檢視在其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。

若要以 JSON 格式檢閱政策，請參閱 [AmazonKeyspacesReadOnlyAccess](#)。

## AWS 受管政策：AmazonKeyspacesFullAccess

您可將 AmazonKeyspacesFullAccess 政策連接到 IAM 身分。

此政策會授予管理許可，讓您的管理員不受限制地存取 Amazon 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 資料表的自動擴展政策。這是必要的，以便管理員可以管理 Amazon Keyspaces 資料表的自動擴展政策。
- CloudWatch – 允許主體查看計費資料表大小，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計費資料表大小並建立 CloudWatch 儀表板。

- IAM – 允許 Amazon Keyspaces 在開啟下列功能時，使用 IAM 自動建立服務連結角色：
  - Amazon Keyspaces CDC streams – 當管理員為資料表啟用串流時，Amazon Keyspaces 會建立服務連結角色 [AWSServiceRoleForAmazonKeyspacesCDC](#)，以代表您將 CloudWatch 指標發佈至您的帳戶。
  - Application Auto Scaling – 當管理員為資料表啟用 Application Auto Scaling 時，Amazon Keyspaces 會建立服務連結角色 [AWSServiceRoleForApplicationAutoScaling\\_CassandraTable](#)，以代表您執行自動擴展動作。
  - Amazon Keyspaces multi-Region replication – 當管理員建立新的多區域金鑰空間，或將新的 新增至現有的單一區域金鑰空間 AWS 區域 時，Amazon Keyspaces 會建立服務連結角色 [AWSServiceRoleForAmazonKeyspacesReplication](#)，代表您將資料表、資料和中繼資料複寫至選取的區域。
- AWS KMS – 允許主體檢視於 AWS KMS 中設定的金鑰。這是必要的，以便使用者可以檢視在其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，以查詢 Amazon EC2 執行個體上的 VPC 以取得端點和網路介面資訊。需要對 Amazon EC2 執行個體的唯一讀存取權，以便 Amazon Keyspaces 可以在用於連線負載平衡的 `system.peers` 資料表中查詢和存放可用的介面 VPC 端點。

若要以 JSON 格式檢閱政策，請參閱 [AmazonKeyspacesFullAccess](#)。

## AWS 受管政策：KeyspacesCDCServiceRolePolicy

您不得將 `KeyspacesCDCServiceRolePolicy` 連接到 IAM 實體。此政策會連接到服務連結角色，允許 Amazon Keyspaces 代表您執行動作。如需詳細資訊，請參閱 [the section called “CDC 串流”](#)。

此政策會將必要的許可授予服務連結角色 `AWSServiceRoleForAmazonKeyspacesCDC`，以代表您將 Amazon Keyspaces CDC 串流指標資料發佈至 CloudWatch。

### 許可詳細資訊

此政策包含以下許可。

- CloudWatch – 允許service-linked-role [AWSServiceRoleForAmazonKeyspacesCDC](#) 代表您將指標資料從 Amazon Keyspaces CDC 串流發佈到 CloudWatch 帳戶中"cloudwatch:namespace": "AWS/Cassandra"的。

若要以 JSON 格式檢閱政策，請參閱 [KeyspacesCDCServiceRolePolicy](#)。

## Amazon Keyspaces 受 AWS 管政策的更新

檢視自此服務開始追蹤 Amazon Keyspaces AWS 受管政策更新以來的詳細資訊。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                                                                                                                                                                                                                | Date           |
|------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| <a href="#">KeyspacesCDCServiceRolePolicy</a> – 新政策        | Amazon Keyspaces 新增了新的受管政策KeyspacesCDCServiceRolePolicy，將必要的許可授予服務連結角色AWSServiceRoleForAmazonKeyspacesCDC，以代表您將Amazon Keyspaces CDC 串流指標資料發佈至 CloudWatch。如需詳細資訊，請參閱 <a href="#">the section called “CDC 串流”</a> 。 | 2025 年 7 月 2 日 |
| <a href="#">AmazonKeyspacesReadOnlyAccess_v2</a> – 更新至現有政策 | Amazon Keyspaces 新增了允許 IAM 主體檢視 Amazon Keyspaces CDC 串流的許可。如需詳細資訊，請參閱 <a href="#">the section called “檢視 CDC 串流”</a> 。                                                                                            | 2025 年 7 月 2 日 |
| <a href="#">AmazonKeyspacesReadOnlyAccess</a> – 更新現有政策     | Amazon Keyspaces 新增了允許 IAM 主體檢視 Amazon                                                                                                                                                                            | 2025 年 7 月 2 日 |

| 變更                                                    | 描述                                                                                                                                                                                                                                                                                                  | Date           |
|-------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
|                                                       | Keyspaces CDC 串流的許可。如需詳細資訊，請參閱 <a href="#">the section called “檢視 CDC 串流”</a> 。                                                                                                                                                                                                                     |                |
| <a href="#">AmazonKeyspacesFullAccess</a><br>– 更新現有政策 | <p>Amazon Keyspaces 為服務連結角色 <a href="#">AWSServiceRoleForAmazonKeyspacesCDC</a> 建立KeyspacesCDCServiceRolePolicy 受管政策，以新增管理員為資料表啟用串流時所需的許可。</p> <p>Amazon Keyspaces 會使用服務連結角色AWSServiceRoleForAmazonKeyspacesCDC，代表您將CloudWatch 指標發佈至您的帳戶。如需詳細資訊，請參閱<a href="#">the section called “CDC 串流”</a>。</p> | 2025 年 7 月 2 日 |

| 變更                                                    | 描述                                                                                                                                                                                                                                                                                                                             | Date             |
|-------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#">AmazonKeyspacesFullAccess</a><br>– 更新現有政策 | <p>Amazon Keyspaces 更新了服務連結角色 <a href="#">AWSServiceRoleForAmazonKeyspacesReplication</a> Keyspaces ReplicationService RolePolicy 的，以新增管理員將新 AWS 區域 新增至單一或多區域金鑰空間時所需的許可。</p> <p>Amazon Keyspaces 使用服務連結角色代表您AWSServiceRoleForAmazonKeyspacesReplication 複寫資料表、其設定和資料。如需詳細資訊，請參閱<a href="#">the section called “多區域複寫”</a>。</p> | 2024 年 11 月 19 日 |
| <a href="#">AmazonKeyspacesFullAccess</a><br>– 更新現有政策 | <p>Amazon Keyspaces 新增了許可，以允許 Amazon Keyspaces 在管理員將新區域新增至單一或多區域金鑰空間時建立服務連結角色。</p> <p>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<a href="#">the section called “多區域複寫”</a>。</p>                                                                                                                                           | 2023 年 10 月 3 日  |

| 變更                                                     | 描述                                                                                                                                                                                                                                                     | Date            |
|--------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| <a href="#">AmazonKeyspacesReadOnlyAccess_v2</a> – 新政策 | <p>Amazon Keyspaces 建立新的政策，為透過介面 VPC 端點連線至 Amazon Keyspaces 的用戶端新增唯讀許可，以存取 Amazon EC2 執行個體來查詢網路資訊。</p> <p>Amazon Keyspaces 會將可用的介面 VPC 端點存放在 <code>system.peers</code> 資料表中，以進行連線負載平衡。如需詳細資訊，請參閱<a href="#">the section called “使用 介面 VPC 端點”</a>。</p> | 2023 年 9 月 12 日 |
| <a href="#">AmazonKeyspacesFullAccess</a> – 更新現有政策     | <p>Amazon Keyspaces 新增了許可，允許 Amazon Keyspaces 在管理員建立多區域金鑰空間時建立服務連結角色。</p> <p>Amazon Keyspaces 使用服務連結角色 <code>AWSServiceRoleForAmazonKeyspacesReplication</code> 代表您執行資料複寫任務。如需詳細資訊，請參閱<a href="#">the section called “多區域複寫”</a>。</p>                  | 2023 年 6 月 5 日  |

| 變更                                                     | 描述                                                                                                                                                                                             | Date           |
|--------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| <a href="#">AmazonKeyspacesReadOnlyAccess</a> – 更新現有政策 | <p>Amazon Keyspaces 新增了新的許可，允許使用者使用 CloudWatch 檢視資料表的計費大小。</p> <p>Amazon Keyspaces 與 Amazon CloudWatch 整合，可讓您監控計費資料表大小。如需詳細資訊，請參閱<a href="#">the section called “Amazon Keyspaces 指標”</a>。</p> | 2022 年 7 月 7 日 |
| <a href="#">AmazonKeyspacesFullAccess</a> – 更新現有政策     | <p>Amazon Keyspaces 新增了新的許可，允許使用者使用 CloudWatch 檢視資料表的計費大小。</p> <p>Amazon Keyspaces 與 Amazon CloudWatch 整合，可讓您監控計費資料表大小。如需詳細資訊，請參閱<a href="#">the section called “Amazon Keyspaces 指標”</a>。</p> | 2022 年 7 月 7 日 |
| <a href="#">AmazonKeyspacesReadOnlyAccess</a> – 更新現有政策 | <p>Amazon Keyspaces 新增了新的許可，允許使用者檢視已為 Amazon Keyspaces 靜態加密設定的 AWS KMS 金鑰。</p> <p>Amazon Keyspaces 靜態加密與整合 AWS KMS，以保護和管理用於加密靜態資料的加密金鑰。若要檢視為 Amazon Keyspaces 設定的 AWS KMS 金鑰，已新增唯讀許可。</p>      | 2021 年 6 月 1 日 |



| 變更                                                    | 描述                                                                                                                                                                                 | Date           |
|-------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------|
| <a href="#">AmazonKeyspacesFullAccess</a><br>– 更新現有政策 | Amazon Keyspaces 新增了新的許可，允許使用者檢視已為 Amazon Keyspaces 靜態加密設定的 AWS KMS 金鑰。<br><br>Amazon Keyspaces 靜態加密與整合 AWS KMS，以保護和管理用於加密靜態資料的加密金鑰。若要檢視為 Amazon Keyspaces 設定的 AWS KMS 金鑰，已新增唯讀許可。 | 2021 年 6 月 1 日 |
| Amazon Keyspaces 開始追蹤變更                               | Amazon Keyspaces 開始追蹤其 AWS 受管政策的變更。                                                                                                                                                | 2021 年 6 月 1 日 |

## 對 Amazon Keyspaces 身分和存取進行故障診斷

使用以下資訊來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

- [我未獲授權在 Amazon Keyspaces 中執行動作](#)
- [我修改了 IAM 使用者或角色，變更未立即生效](#)
- [我無法使用 Amazon Keyspaces point-in-time復原 \(PITR\) 還原資料表](#)
- [我未獲授權執行 iam:PassRole](#)
- [我是管理員，想要允許其他人存取 Amazon Keyspaces](#)
- [我想要允許以外的人員 AWS 帳戶 存取我的 Amazon Keyspaces 資源](#)

### 我未獲授權在 Amazon Keyspaces 中執行動作

如果 AWS 管理主控台 告知您無權執行 動作，則必須聯絡您的管理員尋求協助。您的管理員是為您提供使用者名稱和密碼的人員。

當 mateojackson IAM 使用者嘗試使用主控台檢視###的詳細資訊，但沒有資料表的cassandra:*Select*許可時，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，變更未立即生效

對於具有現有、已建立的 Amazon Keyspaces 連線的應用程式，IAM 政策變更最多可能需要 10 分鐘才會生效。當應用程式建立新的連線時，IAM 政策變更會立即生效。如果您已修改現有的 IAM 使用者或角色，但尚未立即生效，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復原 (PITR) 還原資料表

如果您嘗試使用 point-in-time復原 (PITR) 還原 Amazon Keyspaces 資料表，而您看到還原程序開始，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，以允許您在 Amazon Keyspaces 中還原資料表。

除了使用者許可之外，Amazon Keyspaces 可能需要許可，才能代表您委託人在還原程序期間執行動作。如果資料表使用客戶受管金鑰加密，或者您使用限制傳入流量的 IAM 政策，就會發生這種情況。例如，如果您在 IAM 政策中使用條件索引鍵來限制來源流量至特定端點或 IP 範圍，則還原操作會失敗。若要允許 Amazon Keyspaces 代表您委託人執行資料表還原操作，您必須在 IAM 政策中新增 `aws:ViaAWSService` 全域條件金鑰。

如需還原資料表之許可的詳細資訊，請參閱 [the section called “設定還原的 IAM 許可”](#)。

## 我未獲授權執行 iam:PassRole

如果您收到錯誤，告知您無權執行 `iam:PassRole` 動作，您的政策必須更新，以允許您將角色傳遞給 Amazon Keyspaces。

有些 AWS 服務可讓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 `marymajor` 的 IAM 使用者嘗試使用主控台在 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 沒有將角色傳遞給服務的許可。

```
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 管理員。您的管理員提供您的簽署憑證。

## 我是管理員，想要允許其他人存取 Amazon Keyspaces

若要允許其他人存取 Amazon Keyspaces，您必須將許可授予需要存取的人員或應用程式。如果您使用 AWS IAM Identity Center 管理人員和應用程式，您可以將許可集指派給使用者或群組，以定義其存取層級。許可集會自動建立 IAM 政策，並將其指派給與該人員或應用程式相關聯的 IAM 角色。如需詳細資訊，請參閱 AWS IAM Identity Center 《使用者指南》中的[許可集](#)。

如果您不是使用 IAM Identity Center，則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）。然後，您必須將政策連接到實體，以授予他們在 Amazon Keyspaces 中的正確許可。授予許可後，請將登入資料提供給使用者或應用程式開發人員。他們將使用這些登入資料來存取 AWS。若要進一步了解如何建立 IAM 使用者、群組、政策和許可，請參閱《IAM [使用者指南](#)》中的 [IAM 身分](#)和[政策和許可](#)。

## 我想要允許 以外的人員 AWS 帳戶 存取我的 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：

- 若要了解 Amazon Keyspaces 是否支援這些功能，請參閱 [Amazon Keyspaces 如何與 IAM 搭配使用](#)。
- 若要了解如何 AWS 帳戶 在您擁有的 資源之間提供存取權，請參閱《[IAM 使用者指南](#)》中的[在您擁有 AWS 帳戶 的另一個 IAM 使用者中提供存取權](#)。
- 若要了解如何將資源的存取權提供給第三方 AWS 帳戶，請參閱《IAM 使用者指南》中的[將存取權提供給第三方 AWS 帳戶 擁有](#)。
- 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 \(聯合身分\)](#)。
- 如需了解使用角色和資源型政策進行跨帳戶存取之間的差異，請參閱《IAM 使用者指南》中的 [IAM 中的跨帳戶資源存取](#)。

## 使用 Amazon Keyspaces 的服務連結角色

Amazon Keyspaces (適用於 Apache Cassandra) 使用 AWS Identity and Access Management (IAM) [服務連結角色](#)。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 預先定義，並包含該服務代表您呼叫其他 AWS 服務所需的所有許可。

如需有關支援服務連結角色的其他服務的資訊，請參閱[AWS 使用 IAM 的服務](#)，並在服務連結角色欄中尋找具有是的服務。選擇具有連結的是，以檢視該服務的服務連結角色文件。

### 主題

- [使用 Amazon Keyspaces 應用程式自動擴展的角色](#)
- [使用 Amazon Keyspaces 多區域複寫的角色](#)
- [使用 Amazon Keyspaces CDC 串流的角色](#)

## 使用 Amazon Keyspaces 應用程式自動擴展的角色

Amazon Keyspaces (適用於 Apache Cassandra) 使用 AWS Identity and Access Management (IAM) [服務連結角色](#)。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 預先定義，並包含該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。這可保護您的 Amazon Keyspaces 資源，因為您不會不小心移除存取資源的許可。

### 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，以允許 Application Auto Scaling 代表您呼叫 Amazon Keyspaces 和 Amazon CloudWatch。

`AWSServiceRoleForApplicationAutoScaling_CassandraTable` 服務連結角色信任下列服務擔任該角色：

- `cassandra.application-autoscaling.amazonaws.com`

角色許可政策允許 Application Auto Scaling 對指定的 Amazon Keyspaces 資源完成下列動作：

- 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 建立服務連結角色

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 管理主控台、CQL、AWS CLI 或 AWS API 在資料表上啟用 Amazon Keyspaces 自動擴展時，Application Auto Scaling 會為您建立服務連結角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您為資料表啟用 Amazon Keyspaces 自動擴展時，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[我的 中出現的新角色 AWS 帳戶](#)。

## 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 的服務連結角色

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，您必須先在帳戶中所有 的所有資料表上停用自動擴展 AWS 區域，才能手動刪除服務連結角色。若要在 Amazon Keyspaces 資料表上停用自動擴展，請參閱 [the section called “關閉資料表的 Amazon Keyspaces 自動擴展”](#)。

**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、AWS CLI、或 AWS API 來刪除

AWSServiceRoleForApplicationAutoScaling\_CassandraTable 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](#)。

**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資料表的自動擴展。

## Amazon Keyspaces 服務連結角色支援的區域

Amazon Keyspaces 支援在提供服務的所有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 Amazon Keyspaces 多區域複寫的角色

Amazon Keyspaces (適用於 Apache Cassandra) 使用 AWS Identity and Access Management (IAM) [服務連結角色](#)。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 預先定義，並包含該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。這可保護您的 Amazon Keyspaces 資源，因為您不會不小心移除存取資源的許可。

## Amazon Keyspaces 的服務連結角色許可

Amazon Keyspaces 使用名為 AWSServiceRoleForAmazonKeyspacesReplication 的服務連結角色，以允許 Amazon Keyspaces 代表您將新 AWS 區域新增至金鑰空間，並將資料表及其所有資料和設定複寫至新區域。此角色也允許 Amazon Keyspaces 代表您複寫寫入至所有區域中的資料表。



AWSServiceRoleForAmazonKeyspacesReplication 服務連結角色信任下列服務擔任該角色：

- replication.cassandra.amazonaws.com

名為 KeyspacesReplicationServiceRolePolicy 的角色許可政策允許 Amazon Keyspaces 完成下列動作：

- 動作：cassandra:Select
- 動作：cassandra:SelectMultiRegionResource
- 動作：cassandra:Modify
- 動作：cassandra:ModifyMultiRegionResource
- 動作：cassandra:AlterMultiRegionResource
- 動作：application-autoscaling:RegisterScalableTarget 當您在佈建模式中將複本新增至單一區域資料表並啟用自動擴展時，Amazon Keyspaces 會使用應用程式自動擴展許可。
- 動作：application-autoscaling:DeregisterScalableTarget
- 動作：application-autoscaling:DescribeScalableTargets
- 動作：application-autoscaling:PutScalingPolicy
- 動作：application-autoscaling:DescribeScalingPolicies
- 動作：cassandra:Alter
- 動作：cloudwatch>DeleteAlarms
- 動作：cloudwatch:DescribeAlarms
- 動作：cloudwatch:PutMetricAlarm

雖然 Amazon Keyspaces 服務連結角色 AWSServiceRoleForAmazonKeyspacesReplication 提供政策中指定 Amazon Resource Name (ARN) "arn: \*" 的許可："Action："，但 Amazon Keyspaces 會提供您帳戶的 ARN。

AmazonKeyspacesFullAccess 受管政策中包含建立服務連結角色

AWSServiceRoleForAmazonKeyspacesReplication 的許可。如需詳細資訊，請參閱[the section called "AmazonKeyspacesFullAccess"](#)。

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 建立服務連結角色

您無法手動建立服務連結角色。當您在 AWS 管理主控台、AWS CLI 或 AWS API 中建立多區域金鑰空間時，Amazon Keyspaces 會為您建立服務連結角色。

若您刪除此服務連結角色，之後需要再次建立，您可以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 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 的服務連結角色

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資料表，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。

刪除 `AWSServiceRoleForAmazonKeyspacesReplication` 使用的 Amazon Keyspaces 資源（主控台）

1. 登入 AWS 管理主控台，並在 <https://console.aws.amazon.com/keyspaces/home> 開啟 Amazon Keyspaces 主控台。
2. 從左側面板中選擇金鑰空間。
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，然後選擇刪除金鑰空間。

您也可以使用下列任一方法，以程式設計方式刪除多區域金鑰空間。



- Cassandra 查詢語言 (CQL) [???陳述式](#)。
- CLI 的 [delete-keyspace](#) AWS 操作。
- 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](#)。

## Amazon Keyspaces 服務連結角色支援的區域

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 AWSServiceRoleForAmazonKeyspacesReplication 角色。

| 區域名稱           | 區域身分           | Amazon Keyspaces 的支援 |
|----------------|----------------|----------------------|
| 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                    |
| 歐洲 (法蘭克福)      | eu-central-1   | 是                    |

| 區域名稱                | 區域身分          | Amazon Keyspaces 的支援 |
|---------------------|---------------|----------------------|
| 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                    |
| Africa (Cape Town)  | af-south-1    | 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                    |

## 使用 Amazon Keyspaces CDC 串流的角色

Amazon Keyspaces (適用於 Apache Cassandra) 使用 AWS Identity and Access Management (IAM) [服務連結角色](#)。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 預先定義，並包含該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。

您無法刪除服務連結角色。

### Amazon Keyspaces 的服務連結角色許可

Amazon Keyspaces 使用名為 `AWSServiceRoleForAmazonKeyspacesCDC` 的服務連結角色，以允許 Amazon Keyspaces CDC 串流代表您將 CloudWatch 指標發佈至您的帳戶。

`AWSServiceRoleForAmazonKeyspacesCDC` 服務連結角色信任下列服務擔任該角色：

- `cassandra-streams.amazonaws.com`

名為 [KeyspacesCDCServiceRolePolicy](#) 的角色許可政策允許 Amazon Keyspaces 對 CloudWatch 命名空間中的資源完成下列動作AWS/Cassandra：

- 動作：\* 上的 `cloudwatch:PutMetricData`

`AWSServiceRoleForAmazonKeyspacesCDC` 會在符合下列條件的所有資源上提供許可：Action：  
`cloudwatch:PutMetricData` "cloudwatch:namespace": "AWS/Cassandra"。

如需 `KeyspacesCDCServiceRolePolicy` 的詳細資訊，請參閱 [the section called “KeyspacesCDCServiceRolePolicy”](#)。

若要為會自動建立服務連結角色 `AWSServiceRoleForAmazonKeyspacesCDC` 的資料表啟用 CDC 串流，IAM 主體需要下列許可。

```
{
 "Sid": "KeyspacesCDCServiceLinkedRole",
 "Effect": "Allow",
 "Action": "iam:CreateServiceLinkedRole",
 "Resource": "arn:aws:iam::*:role/aws-service-role/cassandra-streams.amazonaws.com/AWSServiceRoleForAmazonKeyspacesCDC",
 "Condition": {
 "StringLike": {
 "iam:AWSServiceName": "cassandra-streams.amazonaws.com"
 }
 }
}
```

`AmazonKeyspacesFullAccess` 受管政策中包含建立服務連結角色 `AWSServiceRoleForAmazonKeyspacesCDC` 的許可。如需詳細資訊，請參閱[the section called “AmazonKeyspacesFullAccess”](#)。

為 Amazon Keyspaces 建立服務連結角色

您不需要手動為 Amazon Keyspaces CDC 串流建立服務連結角色。當您在具有 AWS 管理主控台、CQL、AWS CLI 或 AWS API 的資料表上啟用 Amazon Keyspaces CDC 串流時，Amazon Keyspaces 會為您建立服務連結角色。

若您刪除此服務連結角色，之後需要再次建立，您可以在帳戶中使用相同程序重新建立角色。當您為資料表啟用 Amazon Keyspaces CDC 串流時，Amazon Keyspaces 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 `AWSServiceRoleForAmazonKeyspacesCDC` 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](#)。

## Amazon Keyspaces 服務連結角色支援的區域

Amazon Keyspaces 支援在提供服務的所有區域中使用服務連結角色。如需詳細資訊，請參閱 [AWS Regions and endpoints](#)。

## Amazon Keyspaces 的合規驗證 (適用於 Apache Cassandra)

在多個合規計畫中，第三方稽核人員會評估 Amazon Keyspaces (適用於 Apache Cassandra) 的安全性和 AWS 合規性。其中包含：

- ISO/IEC 27001 : 2013、27017 : 2015、27018 : 2019 和 ISO/IEC 9001 : 2015。如需詳細資訊，請參閱 [AWS ISO 和 CSA STAR 認證和服務](#)。
- 系統和組織控制 (SOC)
- 支付卡產業 (PCI)
- 聯邦風險與授權管理計畫 (FedRAMP) 高
- 美國健康保險流通與責任法案 (HIPAA)

若要了解 是否 AWS 服務 在特定合規計劃範圍內，請參閱[AWS 服務 合規計劃範圍內](#)然後選擇您感興趣的合規計劃。如需一般資訊，請參閱[AWS 合規計劃](#)。

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[下載報告下載 AWS Artifact](#)。

您使用 時的合規責任 AWS 服務 取決於資料的機密性、您公司的合規目標，以及適用的法律和法規。如需使用 時合規責任的詳細資訊 AWS 服務，請參閱 [AWS 安全文件](#)。

## Amazon Keyspaces 中的彈性和災難復原

AWS 全球基礎設施是以 AWS 區域 和 可用區域為基礎建置。AWS 區域 提供多個實體分隔和隔離的可用區域，這些可用區域與低延遲、高輸送量和高備援聯網連接。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 會在相同區域內的多個 AWS 可用區域中自動複寫資料三次，AWS 區域 以提供耐用性和高可用性。

如需 AWS 區域 和可用區域的詳細資訊，請參閱 [AWS 全球基礎設施](#)。

除了 AWS 全球基礎設施之外，Amazon Keyspaces 還提供數種功能，以協助支援您的資料彈性和備份需求。

## 多區域複寫

如果您需要在更大的地理距離上複寫資料或應用程式，Amazon Keyspaces 會提供多區域複寫。您可以將 Amazon Keyspaces 資料表複寫到 AWS 區域 您選擇的不同位置。如需詳細資訊，請參閱[the section called “多區域複製”](#)。

## Point-in-time(PITR)

PITR 透過提供資料表資料的連續備份，協助保護您的 Amazon Keyspaces 資料表免於意外寫入或刪除操作。如需詳細資訊，請參閱[Point-in-time復原](#)。

# Amazon Keyspaces 中的基礎設施安全性

Amazon Keyspaces (適用於 Apache Cassandra) 是受管服務，受到 AWS 全球網路安全的保護。如需 AWS 安全服務以及 如何 AWS 保護基礎設施的相關資訊，請參閱[AWS 雲端安全](#)。若要使用基礎設施安全的最佳實務設計您的 AWS 環境，請參閱安全支柱 AWS Well-Architected Framework 中的[基礎設施保護](#)。

您可以使用 AWS 發佈的 API 呼叫，透過網路存取 Amazon Keyspaces。使用者端必須支援下列專案：

- 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。

Amazon Keyspaces 支援兩種驗證用戶端請求的方法。第一個方法使用服務特定的登入資料，這是針對特定 IAM 使用者產生的密碼型登入資料。您可以使用 IAM 主控台、AWS CLI或 AWS API 來建立和管理密碼。如需詳細資訊，請參閱[搭配 Amazon Keyspaces 使用 IAM](#)。

第二個方法針對 Cassandra 的開放原始碼 DataStax Java 驅動程式使用身分驗證外掛程式。此外掛程式可讓 [IAM 使用者、角色和聯合身分](#)使用 [AWS Signature 第 4 版程序 \(SigV4\)](#)，將身分驗證資訊新增至 [Amazon Keyspaces \(適用於 Apache Cassandra\) API](#) 請求。如需詳細資訊，請參閱[the section called “建立 IAM 登入資料以進行 AWS 身分驗證”](#)。

您可以從任何網路位置呼叫這些 API 操作，但 Amazon Keyspaces 支援以資源為基礎的存取政策，其中可能包含根據來源 IP 地址的限制。您也可以使用 Amazon Keyspaces 政策來控制來自特定 Amazon

Virtual Private Cloud (Amazon VPC) 端點或特定 VPCs 存取。實際上，這只會隔離網路中特定 VPC 對指定 Amazon Keyspaces 資源 AWS 的網路存取。

您可以使用介面 VPC 端點來防止 Amazon VPC 和 Amazon Keyspaces 之間的流量離開 Amazon 網路。介面 VPC 端點採用一種 AWS 技術 AWS PrivateLink，可讓您在 Amazon VPC 中使用具有私有 IPs 彈性網路介面，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”](#)。

## 搭配介面 VPC 端點使用 Amazon Keyspaces

介面 VPC 端點可在 Amazon VPC 和 Amazon Keyspaces 中執行的虛擬私有雲端 (VPC) 之間啟用私有通訊。介面 VPC 端點採用技術 AWS PrivateLink，這是一種 AWS 可在 VPCs 和服務 AWS 之間進行私有通訊的服務。

AWS PrivateLink 透過在您的 VPC 中使用具有私有 IP 地址的彈性網路介面來啟用此功能，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 Direct Connect 連線。如需詳細資訊，請參閱 [Amazon Virtual Private Cloud](#) 和 [介面 VPC 端點 \(AWS PrivateLink\)](#)。

### 主題

- [使用 Amazon Keyspaces 的介面 VPC 端點](#)
- [使用介面 VPC 端點資訊填入 system.peers 資料表項目](#)
- [控制對 Amazon Keyspaces 介面 VPC 端點的存取](#)
- [可用性](#)
- [VPC 端點政策和 Amazon Keyspaces point-in-time \(PITR\)](#)
- [常見錯誤和警告](#)

## 使用 Amazon Keyspaces 的介面 VPC 端點

您可以建立介面 VPC 端點，讓 Amazon Keyspaces 和 Amazon VPC 資源之間的流量開始流經介面 VPC 端點。若要開始使用，請依照步驟 [建立介面端點](#)。接著，編輯與您在上一個步驟中建立的端點相關聯的安全群組，並設定連接埠 9142 的傳入規則。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 Amazon Keyspaces 連線的 step-by-step 教學課程，請參閱 [the section called “與 VPC 端點連線”](#)。若要了解如何為 Amazon Keyspaces 資源設定跨帳戶存取，這些資源與 VPC AWS 帳戶中不同的應用程式分開，請參閱 [the section called “設定跨帳戶存取”](#)。

## 使用介面 VPC 端點資訊填入 `system.peers` 資料表項目

Apache Cassandra 驅動程式使用 `system.peers` 資料表來查詢叢集的節點資訊。Cassandra 驅動程式使用節點資訊來負載平衡連線和重試操作。Amazon Keyspaces 會自動為透過公有端點連線的用戶端填入 `system.peers` 資料表中的九個項目。

為了為透過介面 VPC 端點連線的用戶端提供類似的功能，Amazon Keyspaces 會使用 VPC 端點可用的每個可用區域的項目填入您帳戶中的 `system.peers` 資料表。若要在 `system.peers` 資料表中查詢和存放可用的介面 VPC 端點，Amazon Keyspaces 會要求您授予 IAM 實體用於連線至 Amazon Keyspaces 的存取許可，以查詢 VPC 以取得端點和網路介面資訊。

### Important

使用可用的介面 VPC 端點填入 `system.peers` 資料表可改善負載平衡並提高讀取/寫入輸送量。建議使用介面 VPC 端點存取 Amazon Keyspaces 的所有用戶端，Apache Spark 需要此端點。

若要授予用於連線至 Amazon Keyspaces 的 IAM 實體許可，以查詢必要的介面 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 並 AmazonKeyspacesFullAccess 包含必要許可，讓 Amazon Keyspaces 存取 Amazon EC2 執行個體，以讀取可用介面 VPC 端點的相關資訊。

若要確認政策已正確設定，請查詢 `system.peers` 資料表以查看聯網資訊。如果 `system.peers` 資料表為空，則可能表示政策未成功設定，或您已超過 `DescribeNetworkInterfaces` 和 `DescribeVPCEndpoints` API 動作的請求率配額。`DescribeVPCEndpoints` 屬於 `Describe*` 類別，並視為非變動動作。`DescribeNetworkInterfaces` 屬於未篩選和未分頁的非變動動作子集，且適用不同的配額。如需詳細資訊，請參閱《Amazon EC2 API 參考》中的[請求字符儲存貯體大小和重新填充率](#)。

如果您確實看到空白資料表，請在幾分鐘後再試一次，以排除請求率配額問題。若要驗證您已正確設定 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的結果，表示您的政策已正確設定。

## 控制對 Amazon Keyspaces 介面 VPC 端點的存取

使用 VPC 端點政策，您可以透過兩種方式控制對資源的存取：

- IAM 政策 – 您可以透過特定 VPC 端點控制允許存取 Amazon Keyspaces 的請求、使用者或群組。您可以在連接到 IAM 使用者、群組或角色的政策中使用[條件索引鍵](#)來執行此操作。
- VPC 政策 – 您可以將政策連接至 Amazon Keyspaces 資源，以控制哪些 VPC 端點可存取這些資源。若要限制對特定金鑰空間或資料表的存取，只允許流經特定 VPC 端點的流量，請編輯限制資源存取的現有 IAM 政策，並新增該 VPC 端點。

以下是存取 Amazon Keyspaces 資源的範例端點政策。

- IAM 政策範例：限制對特定 Amazon Keyspaces 資料表的所有存取，除非流量來自指定的 VPC 端點 – 此範例政策可以連接到 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 資料表的存取。

```
{
 "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

若要限制對特定資料表的存取，您還必須包含對系統資料表的存取。系統資料表為唯讀。

- VPC 政策範例：唯讀存取 – 此範例政策可以連接到 VPC 端點。(如需詳細資訊，請參閱[控制對 Amazon VPC 資源的存取](#))。它限制動作透過其連接的 VPC 端點唯讀存取 Amazon Keyspaces 資源。

```

{
 "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 資料表的存取 – 此範例政策可以連接到 VPC 端點。它透過其連接的 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

若要限制對特定資料表的存取，您還必須包含對系統資料表的存取。系統資料表為唯讀。

## 可用性

Amazon Keyspaces 支援在所有提供服務 AWS 區域 的 中使用介面 VPC 端點。如需詳細資訊，請參閱[???。](#)

## VPC 端點政策和 Amazon Keyspaces point-in-time(PITR)

如果您使用 IAM 政策搭配[條件索引鍵](#)來限制傳入流量，資料表還原操作可能會失敗。例如，如果您使用aws:SourceVpce條件索引鍵將來源流量限制在特定 VPC 端點，資料表還原操作會失敗。若要允許 Amazon Keyspaces 代表委託人執行還原操作，您必須將 aws:ViaAWSService 條件金鑰新增至 IAM 政策。當任何 AWS 服務使用委託人的登入資料提出請求時，aws:ViaAWSService條件金鑰允許存取。如需詳細資訊，請參閱《[IAM 使用者指南](#)》中的 [IAM JSON 政策元素：條件索引鍵](#)。下列政策為範例。

```

{
 "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 Amazon Virtual Private Cloud 並連線至 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.

```

發生此警告是因為 `system.peers` 資料表包含 Amazon Keyspaces 有權檢視之所有 Amazon VPC 端點的項目，包括您連線的 Amazon VPC 端點。您可以安全地忽略此警告。

如需其他錯誤，請參閱 [the section called “VPC 端點連線錯誤”](#)。

## 搭配介面 VPC 端點使用 Amazon Keyspaces CDC 串流

介面 VPC 端點可讓您在 Amazon VPC 和 Amazon Keyspaces 中執行的虛擬私有雲端 (VPC) 之間進行私有通訊。介面 VPC 端點採用技術 AWS PrivateLink，這是一種 AWS 可在 VPCs 和服務 AWS 之間進行私有通訊的服務。

AWS PrivateLink 會透過在您的 VPC 中使用具有私有 IP 地址的彈性網路介面來啟用此功能，讓網路流量不會離開 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 Direct Connect 連線。如需詳細資訊，請參閱 [Amazon Virtual Private Cloud](#) 和 [介面 VPC 端點 \(AWS PrivateLink\)](#)。

### 主題

- [使用 Amazon Keyspaces CDC 串流的介面 VPC 端點](#)
- [Amazon Keyspaces CDC 串流介面 VPC 端點](#)
- [建立 Amazon Keyspaces CDC 串流介面 VPC 端點](#)
- [更新 Amazon Keyspaces CDC 串流介面 VPC 端點](#)
- [使用 Amazon Keyspaces CDC 串流介面 VPC 端點列出串流](#)
- [為 Amazon Keyspaces CDC 串流介面 VPC 端點建立政策](#)

### 使用 Amazon Keyspaces CDC 串流的介面 VPC 端點

您可以使用介面 VPC 端點，讓 Amazon Keyspaces CDC 串流與 Amazon VPC 資源之間的流量開始流經介面 VPC 端點。您可以使用 VPC 端點政策來限制對 CDC 串流的存取。

如需 Amazon Keyspaces CDC 串流的詳細資訊，請參閱 [the section called “使用 CDC 串流”](#)。

### Amazon Keyspaces CDC 串流介面 VPC 端點

當您建立介面端點時，Amazon Keyspaces CDC 串流會為串流產生兩種類型的端點特定 DNS 名稱：區域和區域。

#### 區域性

區域 DNS 名稱包含下列資訊：

- 唯一的 Amazon VPC 端點 ID
- 服務識別符
- 的 AWS 區域

- `vpce.amazonaws.com` 尾碼

對於 ID 為的 Amazon VPC 端點 `vpce-1a2b3c4d`，產生的 DNS 名稱可能看起來類似下列範例：`vpce-1a2b3c4d-5e6f.cassandra-streams.us-east-1.vpce.amazonaws.com`。

## 區域

除了區域 DNS 名稱中的資訊之外，區域 DNS 名稱還包含[可用區域](#)。具有 ID 的 Amazon VPC 端點所產生的 DNS 名稱 `vpce-1a2b3c4d` 看起來如下範例所示，請注意，AWS 區域現在包含可用區域：`vpce-1a2b3c4d-5e6f-us-east-1a.cassandra-streams.us-east-1.vpce.amazonaws.com`

如果您的架構隔離可用區域，您可以使用此選項。例如，您可以將其用於故障遏止或降低區域資料傳輸成本。

### Note

為了獲得最佳可靠性，建議您將服務部署到至少三個可用區域。

## 建立 Amazon Keyspaces CDC 串流介面 VPC 端點

您可以使用 AWS CLI 或 AWS SDK，透過 Amazon Keyspaces CDC Streams 介面端點存取 Amazon Keyspaces CDC Streams API 操作。如需所有可用 API 操作的完整清單，請參閱 [Amazon Keyspaces Streams API 參考](#)。

如需如何建立 VPC 端點的詳細資訊，請參閱《Amazon VPC 使用者指南》中的[建立介面端點](#)。

若要建立 VPC 端點，您可以在下列範例中使用語法。

```
aws ec2 create-vpc-endpoint \
 --region us-east-1 \
 --service-name api.aws.us-east-1.cassandra-streams \
 --vpc-id client-vpc-id \
 --subnet-ids client-subnet-id \
 --vpc-endpoint-type Interface \
 --security-group-ids client-sg-id
```

## 更新 Amazon Keyspaces CDC 串流介面 VPC 端點

若要更新 VPC 端點，您可以在下列範例中使用語法。

```
aws ec2 modify-vpc-endpoint \
 --region us-east-1 \
 --vpc-endpoint-id client-vpc-id \
 --policy-document policy-document \ #example optional parameter
 --add-security-group-ids security-group-ids \ #example optional parameter
```

## 使用 Amazon Keyspaces CDC 串流介面 VPC 端點列出串流

若要列出使用 VPC 端點的串流，您可以使用下列範例中的語法。請務必使用您自己的資訊取代 VPC 端點 ID 的區域和 DNS 名稱。

```
aws keyspacesstreams \
 --endpoint https://vpce-1a2b3c4d-5e6f.cassandra-streams.us-east-1.vpce.amazonaws.com
 \
 --region us-east-1 \
 list-streams
```

## 為 Amazon Keyspaces CDC 串流介面 VPC 端點建立政策

您可以將端點政策連接至 Amazon VPC 端點，以控制對 Amazon Keyspaces CDC 串流的存取。此政策會指定下列資訊：

- 可執行動作的 AWS Identity and Access Management (IAM) 委託人
- 可執行的動作
- 可在其中執行動作的資源

若要限制對特定 Amazon Keyspaces CDC 串流的存取，使其僅允許 Amazon VPC 存取中的特定 AWS 服務，您可以使用下列範例。

下列串流政策會針對 動作 `cassandra:GetStream` 和 `2025-02-20T11:22:33.444` 連接至 `/keyspace/mykeyspace/table/mytable/` 屬於帳戶之資源 `cassandra:GetRecords` 的指定串流，授予任何 IAM 主體的存取權 `123456788901`。若要使用此端點政策，請務必使用串流標籤取代區域、帳戶 ID 和資源。

```
{
 "Version": "2012-10-17",
 "Id": "Policy1216114807515",
 "Statement": [
 { "Sid": "Access-to-specific-stream-only",
 "Principal": "*",
```

```
"Action": [
 "cassandra:GetStream",
 "cassandra:GetRecords"
],
"Effect": "Allow",
"Resource": ["arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/
table/mytable/stream/2025-02-20T11:22:33.444"]
}
]
```

#### Note

Amazon Keyspaces 不支援 CDC 串流的閘道端點。

## Amazon Keyspaces 的組態和漏洞分析

AWS 處理基本安全任務，例如訪客作業系統 (OS) 和資料庫修補、防火牆組態和災難復原。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：

- [共同責任模型](#)
- [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) 提供許多安全功能，供您在開發和實作自己的安全政策時考慮。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](#)
- [Amazon Keyspaces 的偵測安全最佳實務](#)

## Amazon Keyspaces 的預防性安全最佳實務

下列安全最佳實務被視為預防性的，因為它們可協助您預測和防止 Amazon Keyspaces 中的安全事件。

## 使用靜態加密

Amazon Keyspaces 會使用存放在 [AWS Key Management Service \(AWS KMS\)](#) 中的加密金鑰，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 會使用 AWS 擁有的金鑰來加密所有資料表。如果此金鑰不存在，則會為您建立。無法停用服務預設金鑰。

或者，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 [Amazon Keyspaces 靜態加密](#)。

## 使用 IAM 角色來驗證對 Amazon Keyspaces 的存取

對於存取 Amazon Keyspaces 的使用者、應用程式和其他 AWS 服務，他們必須在其 AWS API 請求中包含有效的 AWS 登入資料。您不應將 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憑證，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](#)。

## 針對 Amazon Keyspaces 基本授權使用 IAM 政策

授予許可時，您可以決定取得許可的人員、他們取得許可的 Amazon Keyspaces APIs，以及您想要在這些資源上允許的特定動作。實作最低權限是降低安全風險和可能因錯誤或惡意意圖而產生之影響的關鍵。

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](#)
- [客戶管理政策](#)

## 使用 IAM 政策條件進行精細定義存取控制

當您在 Amazon Keyspaces 中授予許可時，您可以指定決定許可政策如何生效的條件。實作最低權限是降低安全風險和可能因錯誤或惡意意圖而產生之影響的關鍵。

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，以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，授予許可以允許使用者寫入存取特定資料表。



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

如果您在 Amazon Keyspaces 中存放敏感或機密資料，建議您盡可能將資料加密至接近其原始伺服器的位置，以便在整個生命週期內保護您的資料。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 的偵測安全最佳實務

下列安全最佳實務被視為偵測性，因為它們可協助您偵測潛在的安全弱點和事件。

使用 AWS CloudTrail 來監控 AWS Key Management Service (AWS KMS) AWS KMS 金鑰用量

如果您使用[客戶受管 AWS KMS 金鑰](#)進行靜態加密，則會登入此金鑰的使用 AWS CloudTrail。CloudTrail 透過記錄對您帳戶所採取的動作，來提供使用者活動的可見性。CloudTrail 會記錄每個動作的重要資訊，包括提出請求的人員、使用的服務、執行的動作、動作的參數，以及 AWS 服務傳回的回應元素。此資訊可協助您追蹤對 AWS 資源所做的變更，並疑難排解操作問題。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 來稽核金鑰的使用情況。CloudTrail 會建立日誌檔案，其中包含您帳戶的 AWS API 呼叫和相關事件的歷史記錄。這些日誌檔案包括使用主控台、AWS SDKs 和命令列工具提出的所有 AWS KMS API 請求，以及透過整合 AWS 服務提出的請求。您可以使用這些日誌檔案來取得有關何時使用 AWS KMS 金鑰、請求的操作、請求者的身分、請求的 IP 地址等資訊。如需詳細資訊，請參閱《[AWS CloudTrail 使用者指南](#)》中的[使用 AWS CloudTrail 記錄 AWS Key Management Service API 呼叫](#)。

使用 CloudTrail 監控 Amazon Keyspaces 資料定義語言 (DDL) 操作

CloudTrail 透過記錄對您帳戶所採取的動作，來提供使用者活動的可見性。CloudTrail 會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、執行過的動作、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 資源所做的變更，並針對操作問題進行疑難排解。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 Amazon Keyspaces [DDL 操作](#)都會自動記錄在 CloudTrail 中。DDL 操作可讓您建立和管理 Amazon Keyspaces 金鑰空間和資料表。

當 Amazon Keyspaces 中發生活動時，該活動會與事件歷史記錄中的其他服務 AWS 事件一起記錄在 CloudTrail 事件中。如需詳細資訊，請參閱[使用 記錄 Amazon Keyspaces 操作 AWS CloudTrail](#)。您可以在 中檢視、搜尋和下載最近的事件 AWS 帳戶。如需詳細資訊，請參閱 AWS CloudTrail 使用者指南中的 [使用 CloudTrail 事件歷史記錄檢視事件](#)。

若要持續記錄中的事件 AWS 帳戶，包括 Amazon Keyspaces 的事件，請建立[追蹤](#)。線索能讓 CloudTrail 將日誌檔案交付至 Amazon Simple Storage Service (Amazon S3) 儲存貯體。根據預設，當您在主控台上建立線索時，線索會套用至所有 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。

## 標記 Amazon Keyspaces 資源以進行識別和自動化

您可以將中繼資料以標籤形式指派給 AWS 資源。每個標籤都是簡單的標籤，其中包含客戶定義的金鑰和選用值，可讓您更輕鬆地管理、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 – 用於根據標籤控制對 Amazon Keyspaces 資源的存取。如需詳細資訊，請參閱[the section called “以 Amazon Keyspaces 標籤為基礎的授權”](#)。
- 安全性 – 用來判斷資料保護設定等需求。
- 機密性 – 資源支援的特定資料機密性層級的識別符。
- 環境：用來區分開發、測試和正式作業基礎結構。

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 (適用於 Apache Cassandra)

連線至 Amazon Keyspaces 端點後，您可以使用 Cassandra 查詢語言 (CQL) 來使用資料庫。CQL 在許多方面與結構化查詢語言 (SQL) 類似。

- CQL 元素 – 本節涵蓋 Amazon Keyspaces 中支援的 CQL 基本元素，包括識別符、常數、術語和資料類型。它說明字串類型、數值類型、集合類型等概念。
- 資料定義語言 (DDL) – DDL 陳述式用於管理 Amazon Keyspaces 中的金鑰空間和資料表等資料結構。本節涵蓋建立、變更和捨棄金鑰空間和資料表，以及從point-in-time備份還原資料表的陳述式。
- 資料處理語言 (DML) – DML 陳述式用於管理資料表中的資料。本節涵蓋選取、插入、更新和刪除資料的陳述式。它還說明了進階查詢功能，例如使用 IN 運算子、排序結果和分頁。
- 內建函數 – Amazon Keyspaces 支援各種內建純量函數，可用於 CQL 陳述式。本節提供這些函數的概觀，包括其用量的範例。

在本主題中，您會找到在 Amazon Keyspaces 中有效使用 CQL 的詳細語法、範例和最佳實務。

## 主題

- [Amazon Keyspaces 中的 Cassandra 查詢語言 \(CQL\) 元素](#)
- [Amazon Keyspaces 中的 DDL 陳述式 \(資料定義語言\)](#)
- [Amazon Keyspaces 中的 DML 陳述式 \(資料處理語言\)](#)
- [Amazon Keyspaces 中的內建函數](#)

## Amazon Keyspaces 中的 Cassandra 查詢語言 (CQL) 元素

了解 Amazon Keyspaces 支援的 Cassandra 查詢語言 (CQL) 元素，包括識別符、常數、術語和資料類型。

## 主題

- [識別碼](#)
- [常數](#)
- [條款](#)

- [資料類型](#)
- [Amazon Keyspaces 資料類型的 JSON 編碼](#)

## 識別碼

識別符（或名稱）用於識別資料表、資料欄和其他物件。可以引用或不引用識別符。下列適用。

```
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

已定義下列常數。

```
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

一詞表示支援的 值類型。術語定義如下。

```
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    | Description                        |
|---------|------------------------------------|
| ascii   | 代表 ASCII 字元字串。                     |
| text    | 代表 UTF-8 編碼字串。                     |
| varchar | 代表 UTF-8 編碼字串 (varchar 是 的別名text)。 |

## 數值類型

| 資料類型    | Description                                                              |
|---------|--------------------------------------------------------------------------|
| bigint  | 代表 64 位元簽署長。                                                             |
| counter | 代表 64 位元帶正負號的整數計數器。如需詳細資訊，請參閱 <a href="#">the section called “計數器”</a> 。 |
| decimal | 代表變數精確度小數位數。                                                             |
| double  | 代表 64 位元 IEEE 754 浮點。                                                    |
| float   | 代表 32 位元 IEEE 754 浮點。                                                    |
| int     | 代表 32 位元帶正負號的整數。                                                         |
| varint  | 代表 +/-10 <sup>38</sup> 範圍內的整數值。                                          |

## 計數器

資料counter欄包含 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，且無法直接設定。這可讓counter資料欄有助於追蹤計數。例如，您可以使用計數器來追蹤日誌檔案中的項目數量，或在社交網路上檢視文章的次數。下列限制適用於資料counter欄：

- 類型 的資料欄counter不能是資料表primary key的一部分。

- 在包含一或多個 類型資料欄的資料表中counter，該資料表中的所有資料欄都必須是 類型counter。

如果計數器更新失敗（例如，由於逾時或失去與 Amazon Keyspaces 的連線），用戶端不知道計數器值是否已更新。如果重試更新，對計數器值的更新可能會再次套用。

## Blob 類型

| 資料類型 | Description |
|------|-------------|
| blob | 代表任意位元組。    |

## 布林值 (Boolean) 類型

| 資料類型    | Description     |
|---------|-----------------|
| boolean | 代表 true或 false。 |

## 時間相關類型

| 資料類型      | Description                                                                  |
|-----------|------------------------------------------------------------------------------|
| date      | 格式為 的字串<yyyy>-<mm>-<dd> 。                                                    |
| timestamp | 64 位元帶正負號整數，代表自 epoch 起的日期和時間 (1970 年 1 月 1 日，格林威治標準時間 00 : 00 : 00)，以毫秒為單位。 |
| timeuuid  | 代表 <a href="#">版本 1 UUID</a> 。                                               |

## 集合類型

| 資料類型 | Description  |
|------|--------------|
| list | 代表常值元素的排序集合。 |

| 資料類型 | Description       |
|------|-------------------|
| map  | 代表索引鍵/值對的未排序集合。   |
| set  | 代表一或多個常值元素的未排序集合。 |

您可以在角括號中使用集合類型後接另一個資料類型（例如，TEXT或INT）來宣告集合資料欄。您可以使用 SET 的 建立資料欄TEXT，也可以建立 MAP TEXT和 INT鍵值對的，如下列範例所示。

```
SET <TEXT>
MAP <TEXT, INT>
```

非凍結集合可讓您更新每個個別集合元素。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的值，Amazon Keyspaces 會將它們視為 BLOB。這是凍結集合。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，而不是個別元素。Frozen集合資料欄可以是資料表PRIMARY KEY的一部分。

您可以巢狀凍結集合。例如，SET如果 MAP 使用 FROZEN關鍵字，您可以在 MAP中定義，如下列範例所示。

```
SET <FROZEN> <MAP <TEXT, INT>>>
```

Amazon Keyspaces 預設支援最多 8 個層級的凍結集合巢狀化。如需詳細資訊，請參閱[the section called “Amazon Keyspaces 服務配額”](#)。如需 Apache Cassandra 功能差異的詳細資訊，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”](#)。

## 雙組類型

tuple 資料類型代表常值元素的邊界群組。您可以使用元組做為 的替代方案user defined type。您不需要針對元組使用 FROZEN關鍵字。這是因為元組一律凍結，您無法個別更新元素。

## 其他類型

| 資料類型 | Description                    |
|------|--------------------------------|
| inet | 代表 IP 地址的字串，採用 IPv4 或 IPv6 格式。 |

## 靜態

在具有叢集資料欄的 Amazon Keyspaces 資料表中，您可以使用 `STATIC` 關鍵字來建立任何類型的靜態資料欄。

下列陳述式為範例。

```
my_column INT STATIC
```

如需使用靜態資料欄的詳細資訊，請參閱 [the section called “估算靜態資料欄的容量耗用量”](#)。

## 使用者定義的類型 (UDTs)

Amazon Keyspaces 支援使用者定義的類型 (UDTs)。您可以使用任何有效的 Amazon Keyspaces 資料類型來建立 UDT，包括集合和其他現有的 UDTs。您可以在金鑰空間中建立 UDTs，並使用它們來定義金鑰空間中任何資料表中的資料欄。

如需 CQL 語法的詳細資訊，請參閱 [the section called “類型”](#)。如需使用 UDTs 的詳細資訊，請參閱 [the section called “使用者定義的類型 \(UDTs\)”](#)。

若要檢閱每個金鑰空間支援的 UDTs 數量、支援的巢狀層級，以及其他與 UDTs 相關的預設值和配額，請參閱 [the section called “Amazon Keyspaces 中使用者定義類型 \(UDTs\) 的配額和預設值”](#)。

## Amazon Keyspaces 資料類型的 JSON 編碼

Amazon Keyspaces 提供與 Apache Cassandra 相同的 JSON 資料類型映射。下表說明 Amazon Keyspaces 在 `INSERT` JSON 陳述式中接受的資料類型，以及 Amazon Keyspaces 使用 `SELECT` JSON 陳述式傳回資料時使用的資料類型。

對於 `float`、`int`、`UUID` 和 `date` 等單一欄位資料類型，您也可以將資料插入為 `string`。對於複合資料類型和集合，例如 `tuple`、`map` 和 `list`，您也可以將資料插入為 JSON 或編碼的 JSON `string`。



| JSON 資料類型 | INSERT JSON 陳述式中接受的資料類型 | SELECT JSON 陳述式中傳回的資料類型 | 備註                                         |
|-----------|-------------------------|-------------------------|--------------------------------------------|
| ascii     | string                  | string                  | 使用 JSON 字元逸出 \u。                           |
| bigint    | integer, string         | integer                 | 字串必須是有效的 64 位元整數。                          |
| blob      | string                  | string                  | 字串應以 0x 開頭，0x 後面接著偶數的十六進位數字。               |
| boolean   | boolean, string         | boolean                 | 字串必須是 true 或 false。                        |
| date      | string                  | string                  | 日期格式為 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 浮點精確度。 |
| double    | integer, float, string  | float                   | 字串必須是有效的整數或浮點數。                            |
| float     | integer, float, string  | float                   | 字串必須是有效的整數或浮點數。                            |
| inet      | string                  | string                  | IPv4 或 IPv6 地址。                            |
| 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。                           |

| JSON 資料類型 | INSERT JSON 陳述式中接受的資料類型 | SELECT JSON 陳述式中傳回的資料類型 | 備註                                                          |
|-----------|-------------------------|-------------------------|-------------------------------------------------------------|
| map       | map, string             | map                     | 使用原生 JSON 映射表示法。                                            |
| 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 字元逸出 \u。                                            |
| time      | string                  | string                  | 一天中的時間，格式為 HH-MM-SS[.ffffffffffff] 。                        |
| timestamp | integer, string         | string                  | 時間戳記。字串常數可讓您將時間戳記儲存為日期。YYYY-MM-DD HH:MM:SS.SSS 傳回格式為 的日期戳記。 |
| timeuuid  | string                  | string                  | 輸入 1 UUID。如需 UUID 格式 <a href="#">constants</a> ，請參閱 。       |
| tinyint   | integer, string         | integer                 | 字串必須是有效的 8 位元整數。                                            |
| tuple     | list, string            | list                    | 使用原生 JSON 清單表示法。                                            |
| UDT       | map, string             | map                     | 使用原生 JSON 映射表示法搭配欄位名稱做為索引鍵。                                 |

| JSON 資料類型 | INSERT JSON 陳述式中接受的資料類型 | SELECT JSON 陳述式中傳回的資料類型 | 備註                                         |
|-----------|-------------------------|-------------------------|--------------------------------------------|
| uuid      | string                  | string                  | 如需 UUID 格式 <a href="#">constants</a> ，請參閱。 |
| varchar   | string                  | string                  | 使用 JSON 字元逸出 \u。                           |
| varint    | integer, string         | integer                 | 可變長度；可能會溢出用戶端解碼器中的 32 位元或 64 位元整數。         |

## Amazon Keyspaces 中的 DDL 陳述式（資料定義語言）

資料定義語言 (DDL) 是一組 Cassandra 查詢語言 (CQL) 陳述式，用於管理 Amazon Keyspaces（適用於 Apache Cassandra）中的資料結構，例如金鑰空間和資料表。您可以使用 DDL 建立這些資料結構、在建立後修改它們，並在它們不再使用時將其移除。Amazon Keyspaces 會以非同步方式執行 DDL 操作。如需如何確認非同步操作已完成的詳細資訊，請參閱 [the section called “非同步建立和刪除金鑰空間和資料表”](#)。

支援下列 DDL 陳述式：

- [建立 KEYSPACE](#)
- [ALTER KEYSPACE](#)
- [DROP KEYSPACE](#)
- [使用](#)
- [建立資料表](#)
- [ALTER TABLE](#)
- [RESTORE 資料表](#)
- [DROP 資料表](#)
- [建立類型](#)
- [DROP 類型](#)

## 主題

- [Keyspaces](#)
- [表格](#)
- [使用者定義的類型 \(UDTs\)](#)

## Keyspaces

與一或多個應用程式相關的鍵空間群組相關資料表。在關聯式資料庫管理系統 (RDBMS) 方面，金鑰空間大致類似於資料庫、資料表空間或類似建構。

### Note

在 Apache Cassandra 中，金鑰空間會決定如何在多個儲存節點之間複寫資料。不過，Amazon Keyspaces 是一項全受管服務：其儲存層的詳細資訊會代表您管理。因此，Amazon Keyspaces 中的鍵空間僅為邏輯建構，與基礎實體儲存無關。

如需 Amazon Keyspaces 金鑰空間配額限制和限制的相關資訊，請參閱 [配額](#)。

### 金鑰空間的陳述式

- [建立 KEYSPACE](#)
- [ALTER KEYSPACE](#)
- [DROP KEYSPACE](#)
- [USE](#)

## 建立 KEYSPACE

使用 CREATE KEYSPACE 陳述式建立新的金鑰空間。

### 語法

```
create_keyspace_statement ::=
 CREATE KEYSPACE [IF NOT EXISTS] keyspace_name
 WITH options
```

其中：

- *keyspace\_name* 是要建立的金鑰空間名稱。
- 選項是下列一或多個項目：
  - REPLICATION – 表示金鑰空間複寫策略的映射：
    - SingleRegionStrategy – 適用於單一區域金鑰空間。(必要)
    - NetworkTopologyStrategy – 指定至少兩個 AWS 區域。每個區域的複寫係數為 3。(選用)
  - DURABLE\_WRITES – 寫入 Amazon Keyspaces 一律耐用，因此不需要此選項。不過，如果指定，則值必須為 true。
  - TAGS – 建立時要附加至資源的鍵/值對標籤清單。(選用)

## 範例

建立金鑰空間，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 AWS 區域。每個區域的複寫係數為 3。

```
CREATE KEYSPACE my_keyspace
 WITH REPLICATION = {'class': 'NetworkTopologyStrategy', 'us-east-1': '3', 'ap-
southeast-1': '3', 'eu-west-1': '3'};
```

## ALTER KEYSPACE

您可以使用下列選項的 ALTER KEYSPACE WITH 陳述式

- REPLICATION – 使用此選項將新 AWS 區域 複本新增至金鑰空間。您可以將新區域新增至單一區域或多區域金鑰空間。
- TAGS – 使用此選項從鍵空間新增或移除標籤。

## 語法

```
alter_keyspace_statement ::=
 ALTER KEYSPACE keyspace_name
 WITH options
```

其中：

- *keyspace\_name* 是要變更的金鑰空間名稱。
- 選項為下列其中一項：
  - ADD | DROP TAGS – 要從鍵空間新增或移除的鍵值對標籤清單。
  - REPLICATION – 表示金鑰空間複寫策略的映射；
    - class– 將金鑰空間NetworkTopologyStrategy定義為多區域金鑰空間。
    - region– AWS 區域 為此金鑰空間指定一個額外的。每個區域的複寫係數為 3。
    - CLIENT\_SIDE\_TIMESTAMPS – 預設值為 DISABLED。您只能將狀態變更為 ENABLED。

## 範例

修改金鑰空間，如下列範例所示，以新增標籤。

```
ALTER KEYSPACE my_keyspace ADD TAGS {'key1':'val1', 'key2':'val2'};
```

若要將第三個區域新增至多區域金鑰空間，您可以使用下列陳述式。

```
ALTER KEYSPACE my_keyspace
WITH REPLICATION = {
 'class': 'NetworkTopologyStrategy',
 'us-east-1': '3',
 'us-west-2': '3',
 'us-west-1': '3'
} AND CLIENT_SIDE_TIMESTAMPS = {'status': 'ENABLED'};
```

## DROP KEYSPACE

使用 DROP KEYSPACE陳述式移除金鑰空間，包括其所有內容，例如資料表。

### 語法

```
drop_keyspace_statement ::=
 DROP KEYSPACE [IF EXISTS] keyspace_name
```

其中：

- *keyspace\_name* 是要捨棄的金鑰空間名稱。

## 範例

```
DROP KEYSPACE my_keyspace;
```

## USE

使用 USE 陳述式來定義目前的金鑰空間。這可讓您參考繫結至特定金鑰空間的物件，例如資料表和類型，而無需使用包含金鑰空間字首的完整名稱。

## 語法

```
use_statement ::=
 USE keyspace_name
```

其中：

- `keyspace_name` 是要使用的金鑰空間名稱。

## 範例

```
USE my_keyspace;
```

## 表格

資料表是 Amazon Keyspaces 中的主要資料結構。資料表中的資料會組織成資料列和資料欄。這些資料欄的子集用於透過分割區索引鍵的規格來判斷分割區（最終是資料置放）。

另一組資料欄可以定義成叢集資料欄，這表示它們可以參與做為查詢執行的述詞。

根據預設，會使用隨需輸送量容量建立新的資料表。您可以變更新資料表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 “設定讀取/寫入容量模式”](#)。

對於佈建模式中的資料表，您可以設定選用的 AUTOSCALING\_SETTINGS。如需 Amazon Keyspaces 自動擴展和可用選項的詳細資訊，請參閱 [the section called “在現有資料表上設定自動擴展”](#)。

如需 Amazon Keyspaces 資料表配額限制和限制的相關資訊，請參閱 [配額](#)。

資料表的陳述式

- [CREATE TABLE](#)

- [ALTER TABLE](#)
- [RESTORE 資料表](#)
- [DROP TABLE](#)

## CREATE TABLE

使用 CREATE TABLE 陳述式建立新的資料表。

### 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]
 | cdc
 | 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。完整名稱包含金鑰空間字首。或者，您可以使用 金鑰空間陳述式來設定目前的USE金鑰空間。



- *column\_definition* 包含下列項目：
  - *column\_name* – 資料欄的名稱。
  - *cql\_type* – Amazon Keyspaces 資料類型 ( 請參閱 [資料類型](#))。
  - *FROZEN* – 將使用者定義或類型 *collection* ( 例如 *LIST*、*SET*或 *MAP*) 的此欄指定為凍結。凍結集合會序列化為單一不可變值，並視為 *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。靜態資料欄會存放相同分割區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* 包含下列項目：
  - *partition\_key*
  - *clustering\_columns*
- *partition\_key*:
  - 分割區索引鍵可以是單一資料欄，也可以是由兩個或多個資料欄組成的複合值。主索引鍵的分割區索引鍵部分是必要的，並決定 Amazon Keyspaces 如何存放您的資料。
- *clustering\_columns*:
  - 主索引鍵的選用叢集資料欄部分會決定如何在每個分割區中叢集和排序資料。
- *table\_options* 包含下列項目：
  - *CLUSTERING ORDER BY* – 資料表上的預設 *CLUSTERING ORDER* 由 *ASC* ( 遞增 ) 排序方向的叢集索引鍵組成。指定它來覆寫預設排序行為。
  - *cdc* – 布林值，指定 Amazon Keyspaces 是否為資料表建立變更資料擷取 (CDC) 串流。預設值為 *false*。若要在啟用串流 *view type*時指定，*cdc\_specification*請使用 設定 *CUSTOM\_PROPERTIES*。
  - *CUSTOM\_PROPERTIES* – Amazon Keyspaces 特有的設定映射。
    - *capacity\_mode*：指定資料表的讀取/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*和 *write\_capacity\_units*做為輸入。預設值為 *throughput\_mode:PAY\_PER\_REQUEST*。
    - *cdc\_specification*：指定 CDC 串流 *view\_type*的。選項為：
      - *NEW\_AND\_OLD\_IMAGES* – 變更前後的兩個資料列版本。這是預設值。
      - *NEW\_IMAGE* – 變更後的列版本。
      - *OLD\_IMAGE* – 變更前的資料列版本。
    - *KEYS\_ONLY* – 已變更資料列的分割區和叢集索引鍵。

如需 CDC 串流的詳細資訊，請參閱 [the section called “使用 CDC 串流”](#)。如需程式碼範例，請參閱 [the section called “為新資料表啟用 CDC 串流”](#)。

- `client_side_timestamps`：指定資料表的用戶端時間戳記是啟用或停用。選項包括 `{'status': 'enabled'}` 和 `{'status': 'disabled'}`。如果未指定，則預設值為 `status:disabled`。為資料表啟用用戶端時間戳記後，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`。加密選項客戶受管金鑰需要 Amazon Resource Name (ARN) 格式的 AWS KMS 金鑰做為輸入：`kms_key_identifier:ARN:kms_key_identifier:ARN`。
- `point_in_time_recovery`：指定資料表的point-in-time還原是啟用或停用。選項包括 `status:enabled` 和 `status:disabled`。如果未指定，則預設值為 `status:disabled`。
- `replica_updates`：指定專屬的多區域資料表設定 AWS 區域。對於多區域資料表，您可以為每個以不同的方式設定資料表的讀取容量 AWS 區域。您可以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 “在佈建模式中建立多區域資料表”](#)。
- `region` – 具有下列設定的資料表複本 AWS 區域的：
  - `read_capacity_units`
- `TTL`：啟用資料表的存留時間自訂設定。若要啟用，請使用 `status:enabled`。預設值為 `status:disabled`。TTL 啟用之後，您就無法為資料表停用它。
- `AUTOSCALING_SETTINGS` 包含下列佈建模式中資料表的選用設定。如需詳細資訊和範例，請參閱 [the section called “使用自動擴展建立新的資料表”](#)。
- `provisioned_write_capacity_autoscaling_update`:
  - `autoscaling_disabled` – 若要啟用寫入容量的自動擴展，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小寫入輸送量層級。此值必須介於 1 和您帳戶的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寫入輸送量上限。此值必須介於 1 和您帳戶的每秒最大輸送量配額之間（預設為 40,000）。
  - `scaling_policy` – Amazon Keyspaces 支援目標追蹤政策。自動擴展目標是資料表的佈建寫入容量。
  - `target_tracking_scaling_policy_configuration` – 若要定義目標追蹤政策，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《Application Auto Scaling [Scaling 使用者指南](#)》中的[目標追蹤擴展政策](#)。

- `target_value` – 資料表的目標使用率。Amazon Keyspaces 自動擴展可確保耗用容量與佈建容量的比率保持在或接近此值。您能以百分比的形式定義 `target_value`。介於 20 到 90 之間的兩倍。(必要)
- `scale_in_cooldown` – 擴展活動之間的冷卻時間，以秒為單位，可讓資料表在另一個擴展活動開始之前穩定。如果未提供任何值，則預設值為 0。(選用)
- `scale_out_cooldown` – 擴展活動之間的冷卻時間，以秒為單位，可讓資料表在另一個橫向擴展活動開始之前穩定。如果未提供任何值，則預設值為 0。(選用)
- `disable_scale_in` : `boolean` 指定資料表 `scale-in` 是否已停用或已啟用的。此參數預設為停用。若要開啟 `scale-in`，請將 `boolean` 值設定為 `FALSE`。這表示會自動為您縮減資料表的容量。(選用)
- `provisioned_read_capacity_autoscaling_update`:
  - `autoscaling_disabled` – 若要啟用讀取容量的自動擴展，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輸送量最低層級。此值必須介於 1 和您帳戶的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輸送量上限。此值必須介於 1 和您帳戶的每秒最大輸送量配額之間（預設為 40,000）。
  - `scaling_policy` – Amazon Keyspaces 支援目標追蹤政策。自動擴展目標是資料表的佈建讀取容量。
  - `target_tracking_scaling_policy_configuration` – 若要定義目標追蹤政策，您必須定義目標值。如需目標追蹤和冷卻期間的詳細資訊，請參閱 [《Application Auto Scaling Scaling 使用者指南》中的目標追蹤擴展政策](#)。
  - `target_value` – 資料表的目標使用率。Amazon Keyspaces 自動擴展可確保耗用容量與佈建容量的比率保持在或接近此值。您能以百分比的形式定義 `target_value`。介於 20 到 90 之間的兩倍。(必要)
  - `scale_in_cooldown` – 擴展活動之間的冷卻時間，以秒為單位，可讓資料表在另一個擴展活動開始之前穩定。如果未提供任何值，則預設值為 0。(選用)
  - `scale_out_cooldown` – 擴展活動之間的冷卻時間，以秒為單位，可讓資料表在另一個橫向擴展活動開始之前穩定。如果未提供任何值，則預設值為 0。(選用)
  - `disable_scale_in` : `boolean` 指定資料表 `scale-in` 是否已停用或已啟用的。此參數預設為停用。若要開啟 `scale-in`，請將 `boolean` 值設定為 `FALSE`。這表示會自動為您縮減資料表的容量。(選用)

- `replica_updates` : 指定多區域資料表 AWS 區域 的特定自動擴展設定。對於多區域資料表，您可以為每個以不同的方式設定資料表的讀取容量 AWS 區域。您可以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 “更新多區域資料表的佈建容量和自動擴展設定”](#)。
- `region` – 具有下列設定的資料表複本 AWS 區域 的 :
  - `provisioned_read_capacity_autoscaling_update`
  - `autoscaling_disabled` – 若要啟用資料表讀取容量的自動擴展，請將值設定為 `false`。預設值為 `true`。(選用)

 Note

必須為資料表的所有複本啟用或停用多區域資料表的自動擴展。

- `minimum_units` – 資料表應隨時準備好支援的讀取輸送量最低層級。此值必須介於 1 和您帳戶的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讀取輸送量上限。此值必須介於 1 和您帳戶的每秒最大輸送量配額之間（預設為 40,000）。
- `scaling_policy` – Amazon Keyspaces 支援目標追蹤政策。自動擴展目標是資料表的佈建讀取容量。
- `target_tracking_scaling_policy_configuration` – 若要定義目標追蹤政策，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《Application Auto Scaling [Scaling 使用者指南](#)》中的 [目標追蹤擴展政策](#)。
  - `target_value` – 資料表的目標使用率。Amazon Keyspaces 自動擴展可確保耗用讀取容量與佈建讀取容量的比率保持在或接近此值。您能以百分比的形式定義 `target_value`。介於 20 到 90 之間的兩倍。(必要)
  - `scale_in_cooldown` – 擴展活動之間的冷卻時間，以秒為單位，可讓資料表在另一個擴展活動開始之前穩定。如果未提供任何值，則預設值為 0。(選用)
  - `scale_out_cooldown` – 擴展活動之間的冷卻時間，以秒為單位，可讓資料表在另一個橫向擴展活動開始之前穩定。如果未提供任何值，則預設值為 0。(選用)
  - `disable_scale_in` : `boolean` 指定資料表 `scale-in` 是否已停用或已啟用的。此參數預設為停用。若要開啟 `scale-in`，請將 `boolean` 值設定為 `FALSE`。這表示系統會自動為您縮減資料表的讀取容量。(選用)
- `default_time_to_live` – 資料表的預設存留時間設定，以秒為單位。
- `TAGS` – 建立索引鍵/值對標籤時要附加至資源的清單。

- *clustering\_order* 包含下列項目：
  - *column\_name* – 資料欄的名稱。
  - *ASC* / *DESC* – 設定上階 (ASC) 或下階 (DESC) 順序修改器。如果未指定，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-1:555555555555:key/11111111-1111-111-1111-111111111111'
 }
}
AND CLUSTERING ORDER BY (division ASC)
AND TAGS={'key1':'val1', 'key2':'val2'}
AND default_time_to_live = 3024000;
```

在使用叢集資料欄的資料表中，非叢集資料欄可以在資料表定義中宣告為靜態資料欄。如需靜態資料欄的詳細資訊，請參閱 [the section called “估算靜態資料欄的容量耗用量”](#)。

## 範例

```
CREATE TABLE my_keyspace.my_table (
 id int,
 name text,
 region text,
 division text,
 project text STATIC,
 PRIMARY KEY (id,division));
```

您可以使用使用者定義類型 (UDT) 的欄來建立資料表。範例中的第一個陳述式會建立 類型，第二個陳述式會使用 類型的資料欄來建立資料表。

### 範例

```
CREATE TYPE my_keyspace."udt""N@ME" (my_field int);
CREATE TABLE my_keyspace.my_table (my_col1 int pri key, my_col2 "udt""N@ME");
```

## ALTER TABLE

使用 ALTER TABLE 陳述式來新增資料欄、新增標籤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* 是要修改的資料表名稱。
- *column\_definition* 是要新增的資料欄名稱和資料類型。
- *column\_definition\_list* 是放在括號內以逗號分隔的資料欄清單。
- *table\_options* 包含下列項目：



- *AUTOSCALING\_SETTINGS* 包含佈建資料表的選用自動擴展設定。如需語法和詳細說明，請參閱 [the section called “CREATE TABLE”](#)。如需範例，請參閱 [the section called “在現有資料表上設定自動擴展”](#)。
- *cdc* – 布林值，指定 Amazon Keyspaces 是否為資料表建立變更資料擷取 (CDC) 串流。預設值為 `false`。若要在啟用串流 `view type` 時指定，*cdc\_specification* 請使用 設定 `CUSTOM_PROPERTIES`。
- *CUSTOM\_PROPERTIES* – Amazon Keyspaces 特有的設定映射。
  - *capacity\_mode*：指定資料表的讀取/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` 和 `write_capacity_units` 做為輸入。預設值為 `throughput_mode:PAY_PER_REQUEST`。
  - *cdc\_specification*：指定 CDC 串流 `view_type` 的。選項為：
    - `NEW_AND_OLD_IMAGES` – 變更前後的兩個資料列版本。這是預設值。
    - `NEW_IMAGE` – 變更後的列版本。
    - `OLD_IMAGE` – 變更前的資料列版本。
    - `KEYS_ONLY` – 已變更資料列的分割區和叢集索引鍵。

如需 CDC 串流的詳細資訊，請參閱 [the section called “使用 CDC 串流”](#)。如需程式碼範例，請參閱 [the section called “為現有資料表啟用 CDC 串流”](#)。

- *client\_side\_timestamps*：指定資料表的用戶端時間戳記是啟用或停用。選項包括 `{'status': 'enabled'}` 和 `{'status': 'disabled'}`。如果未指定，則預設值為 `status:disabled`。為資料表啟用用戶端時間戳記後，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`。加密選項客戶受管金鑰需要 Amazon Resource Name (ARN) 格式的 AWS KMS 金鑰做為輸入：`kms_key_identifier:ARN`。
- *point\_in\_time\_recovery*：指定資料表的 point-in-time 還原是啟用或停用。選項包括 `status:enabled` 和 `status:disabled`。預設值為 `status:disabled`。
- *replica\_updates*：指定多區域資料表 AWS 區域 的特定設定。對於多區域資料表，您可以為每個資料表設定不同的讀取容量 AWS 區域。您可以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 “更新多區域資料表的佈建容量和自動擴展設定”](#)。
  - *region* – 具有下列設定的資料表複本 AWS 區域 的：
    - `read_capacity_units`

- `ttl`：啟用資料表的存留時間自訂設定。若要啟用，請使用 `status:enabled`。預設值為 `status:disabled`。`ttl` 啟用 之後，您就無法為資料表停用它。
- `default_time_to_live`：資料表的預設存留時間設定，以秒為單位。
- `TAGS` 是要連接到資源的鍵/值對標籤清單。

#### Note

使用 `ALTER TABLE`，您只能變更單一自訂屬性。您不能在同一陳述式中結合多個 `ALTER TABLE` 命令。

## 範例

下列陳述式說明如何將資料欄新增至現有資料表。

```
ALTER TABLE mykeyspace.mytable ADD (ID int);
```

此陳述式說明如何將兩個集合資料欄新增至現有資料表：

- 包含巢狀凍結集合 `col_frozen_list` 的凍結集合欄
- `col_map` 包含巢狀凍結集合的非凍結集合欄

```
ALTER TABLE my_Table ADD(col_frozen_list FROZEN<LIST<FROZEN<SET<TEXT>>>>, col_map
MAP<INT, FROZEN<SET<INT>>>>);
```

下列範例示範如何將使用使用者定義類型 (UDT) 的資料欄新增至資料表。

```
ALTER TABLE my_keyspace.my_table ADD (my_column, my_udt);
```

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}};
```

下列陳述式指定資料表的客戶受管 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-1:555555555555: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'}};
```

若要設定資料表的預設存留時間值，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'}};
```

## RESTORE 資料表

使用 RESTORE TABLE陳述式將資料表還原至某個時間點。此陳述式需要在point-in-time復原。如需詳細資訊，請參閱[the section called “使用point-in-time復原進行備份和還原”](#)。

### 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* 包含下列項目：
  - *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 特有的設定映射。
  - **capacity\_mode** : 指定資料表的讀取/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**和 **write\_capacity\_units**做為輸入。預設值是來源資料表的目前設定。
  - **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**. 加密選項客戶受管金鑰需要 Amazon Resource Name (ARN) 格式的 AWS KMS 金鑰做為輸入 : **kms\_key\_identifier:ARN**。若要將以客戶受管金鑰加密的資料表還原至以 加密的資料表 AWS 擁有的金鑰 , Amazon Keyspaces 需要存取來源資料表的 AWS KMS 金鑰。
  - **point\_in\_time\_recovery** : 指定資料表的point-in-time還原是啟用或停用。選項包括 **status:enabled** 和 **status:disabled**. 與建立新資料表時不同 , 還原資料表的預設狀態是 , **status:enabled**因為設定繼承自來源資料表。若要停用還原資料表的 PITR , 您必須**status:disabled**明確設定 。
  - **replica\_updates** : 指定多區域資料表 AWS 區域 的特定設定。對於多區域資料表 , 您可以為每個資料表設定不同的讀取容量 AWS 區域。您可以設定下列參數來執行此操作。
    - **region** – 具有下列設定的資料表複本 AWS 區域 的 :
      - **read\_capacity\_units**
  - **AUTOSCALING\_SETTINGS** 包含佈建資料表的選用自動擴展設定。如需詳細的語法和說明 , 請參閱 [the section called “CREATE TABLE”](#)。
  - **TAGS** 是要連接到資源的鍵/值對標籤清單。

#### Note

刪除的資料表只能還原至刪除時間。

## 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 DROP TABLE 陳述式從金鑰空間移除資料表。

### 語法

```
drop_table_statement ::=
 DROP TABLE [IF EXISTS] table_name
```

其中：

- IF EXISTS 如果資料表不存在，DROP TABLE 會防止失敗。(選用)
- *table\_name* 是要捨棄的資料表名稱。

### 範例

```
DROP TABLE my_keyspace.my_table;
```

## 使用者定義的類型 (UDTs)

UDT – 欄位和資料類型的分組，可用於在 Amazon Keyspaces 中定義單一資料欄。UDTs 的有效資料類型都是支援的 Cassandra 資料類型，包括您已在相同金鑰空間中建立的集合和其他 UDTs。如需支援的 Cassandra 資料類型的詳細資訊，請參閱 [the section called “Cassandra 資料類型支援”](#)。

```
user_defined_type::= udt_name
udt_name::= [keyspace_name '.'] identifier
```

### 類型的陳述式

- [CREATE TYPE](#)
- [DROP TYPE](#)

## CREATE TYPE

使用 CREATE TYPE 陳述式來建立新的類型。

### 語法

```
create_type_statement ::= CREATE TYPE [IF NOT EXISTS] udt_name
 '(' field_definition (',' field_definition) * ')'
 field_definition ::= identifier cql_type
```

其中：

- IF NOT EXISTS 如果類型已存在，會CREATE TYPE防止 失敗。(選用)
- udt\_name 是類型格式 UDT 的完整名稱，例如 my\_keyspace.my\_type。如果您使用 USE陳述式 定義目前的金鑰空間，則不需要指定金鑰空間名稱。
- field\_definition 包含名稱和類型。

下表顯示允許的 UDT 名稱範例。第一欄顯示如何在建立 類型時輸入名稱，第二欄顯示 Amazon Keyspaces 如何在內部格式化名稱。Amazon Keyspaces 預期 等操作的格式化名稱GetType。

| 輸入的名稱                     | 格式化名稱                   | 注意                                                 |
|---------------------------|-------------------------|----------------------------------------------------|
| MY_UDT                    | my_udt                  | 如果沒有雙引號，Amazon Keyspaces 會將所有大寫字元轉換為小寫。            |
| "MY_UDT"                  | MY_UDT                  | 使用雙引號時，Amazon Keyspaces 會遵守大寫字元，並從格式化名稱中移除雙引號。     |
| "1234"                    | 1234                    | 使用雙引號時，名稱可以以數字開頭，Amazon Keyspaces 會從格式化的名稱中移除雙引號。  |
| "Special_Ch@r@cter s<>!!" | Special_Ch@r@cter s<>!! | 使用雙引號時，名稱可以包含特殊字元，Amazon Keyspaces 會從格式化的名稱中移除雙引號。 |
| "nested""quote s"         | nested""quotes          | Amazon Keyspaces 會從格式化名稱中移除外部雙引號和逸出雙引號。            |

範例

```
CREATE TYPE my_keyspace.phone (
 country_code int,
```

```
number text
);
```

如果巢狀 UDTs 凍結，您可以巢狀 UDT。如需 類型預設值和配額的詳細資訊，請參閱 [the section called “Amazon Keyspaces UDT 配額和預設值”](#)。

```
CREATE TYPE my_keyspace.user (
 first_name text,
 last_name text,
 phones FROZEN<phone>
);
```

如需示範如何建立 UDTs 更多程式碼範例，請參閱 [the section called “使用者定義的類型 \(UDTs\)”](#)。

## DROP TYPE

使用 DROP TYPE 陳述式刪除 UDT。您只能刪除其他類型或資料表未使用的類型。

### 語法

```
drop_type_statement ::= DROP TYPE [IF EXISTS] udt_name
```

其中：

- IF EXISTS 如果類型不存在，會 DROP TYPE 防止失敗。(選用)
- *udt\_name* 是類型格式 UDT 的完整名稱，例如 my\_keyspace.my\_type。如果您使用 USE 陳述式定義目前的金鑰空間，則不需要指定金鑰空間名稱。

### 範例

```
DROP TYPE udt_name;
```

## Amazon Keyspaces 中的 DML 陳述式（資料處理語言）

資料處理語言 (DML) 是一組 Cassandra 查詢語言 (CQL) 陳述式，用於管理 Amazon Keyspaces (適用於 Apache Cassandra) 資料表中的資料。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 不支援聯結或子查詢。)

## 主題

- [SELECT](#)
- [INSERT](#)
- [UPDATE](#)
- [DELETE](#)

## SELECT

使用 SELECT 陳述式查詢資料。

### 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 編碼資料類型映射至 Amazon Keyspaces 資料類型的資料表，請參閱 [the section called “Amazon Keyspaces 資料類型的 JSON 編碼”](#)。

### 使用 IN 關鍵字

IN 關鍵字指定一或多個值的相等性。它可以套用至分割區索引鍵和叢集資料欄。結果會依索引鍵在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IN關鍵字以及 Amazon Keyspaces 如何處理陳述式的詳細資訊，請參閱 [the section called “使用 IN SELECT”](#)。

### 排序結果

ORDER BY 子句指定傳回結果的排序順序。它需要資料欄名稱清單以及每個資料欄的排序順序做為引數。您只能依子句順序指定叢集資料欄。不允許使用非叢集資料欄。排序順序選項ASC適用於遞增和DESC遞減排序。如果省略排序順序，則會使用叢集資料欄的預設順序。如需可能的排序順序，請參閱 [the section called “訂單結果”](#)。

### 範例

```
SELECT name, id, division, manager_id FROM "myGSGKeyspace".employees_tbl WHERE id =
'012-34-5678' ORDER BY division;
```

ORDER BY 搭配 IN關鍵字使用時，結果會在頁面中排序。不支援使用已停用分頁進行完整重新排序。

### TOKEN

您可以將TOKEN函數套用至 SELECT和 WHERE子句中的資料PARTITION KEY欄。使用 TOKEN函數時，Amazon Keyspaces 會根據 的映射字符值傳回資料列，PARTITION\_KEY而不是根據 的值傳回資料列PARTITION KEY。

TOKEN IN關鍵字不支援關聯。

### 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

您可以使用 TTL 函數搭配 SELECT 陳述式，以秒為單位擷取為資料欄存放的過期時間。如果未設定 TTL 任何值，函數會傳回 null。

### 範例

```
SELECT TTL(my_column) from my_table;
```

TTL 函數無法用於多儲存格資料欄，例如集合。

## WRITETIME 函式

只有在資料表使用用戶端時間戳記時，您才能使用 WRITETIME 函數搭配 SELECT 陳述式來擷取存放為資料欄值中繼資料的時間戳記。如需詳細資訊，請參閱[the section called “用戶端時間戳記”](#)。

```
SELECT WRITETIME(my_column) from my_table;
```

WRITETIME 函數無法用於多儲存格資料欄，例如集合。

### Note

為了相容於已建立的 Cassandra 驅動程式行為，當您透過 Cassandra 驅動程式和開發人員工具使用 Cassandra 查詢語言 (CQL) API 呼叫對系統資料表執行操作時，不會強制執行標籤型授權政策。如需詳細資訊，請參閱[the section called “根據標籤的 Amazon Keyspaces 資源存取”](#)。

## INSERT

使用 INSERT 陳述式將資料列新增至資料表。

### 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 支援下列值做為 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 – 代表自稱為 的標準基準時間以來微秒數bigint的值epoch：1970 年 1 月 1 日 00：00：00 GMT。Amazon Keyspaces 中的時間戳記必須介於過去 2 天到未來的 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;

```

## JSON 支援

如需將 JSON 編碼資料類型映射至 Amazon Keyspaces 資料類型的資料表，請參閱 [the section called “Amazon Keyspaces 資料類型的 JSON 編碼”](#)。

您可以使用 JSON 關鍵字，將 JSON 編碼的映射插入為單一資料列。對於資料表中存在但 JSON 插入陳述式中省略的資料欄，請使用 DEFAULT UNSET 保留現有的值。使用 DEFAULT NULL 將 NULL 值寫入省略的資料欄的每一列，並覆寫現有的值（需支付標準寫入費用）。DEFAULT NULL 是預設選項。

## 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 資料包含重複的金鑰，Amazon Keyspaces 會儲存金鑰的最後一個值（類似於 Apache Cassandra）。在下列範例中，如果重複的金鑰是 `id`，234-56-7890 則會使用 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 counter，請使用下列語法。如需詳細資訊，請參閱[the section called “計數器”](#)。

```
UPDATE ActiveUsers SET counter = counter + 1 WHERE user = A70FE1C0-5408-4AE3-
BE34-8733E5K09F14 AND action = 'click';
```

## 更新參數

UPDATE 支援下列值做為 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 – 代表自稱為 的標準基準時間以來微秒數bigint的值epoch：1970 年 1 月 1 日 00 : 00 : 00 GMT。Amazon Keyspaces 中的時間戳記必須介於過去 2 天到未來的 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 DELETE 陳述式從資料表中移除資料列。

## 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* 是包含您要刪除之資料列的資料表。

範例

```
DELETE manager_id FROM "myGSGKeyspace".employees_tbl WHERE id='789-01-2345' AND
division='Executive' ;
```

DELETE 支援以下值為 *update\_parameter*：

- **TIMESTAMP** – 代表自稱為 的標準基準時間以來微秒數 *bigint* 的值 *epoch*：1970 年 1 月 1 日 00：00：00 GMT。

## Amazon Keyspaces 中的內建函數

Amazon Keyspaces (適用於 Apache Cassandra) 支援可在 Cassandra 查詢語言 (CQL) 陳述式中使用的各種內建函數。

主題

- [純量函數](#)

### 純量函數

純量函數會對單一值執行計算，並將結果傳回為單一值。Amazon Keyspaces 支援下列純量函數。

| 函式                       | Description            |
|--------------------------|------------------------|
| <code>blobAsType</code>  | 傳回指定資料類型的值。            |
| <code>cast</code>        | 將一種原生資料類型轉換為另一種原生資料類型。 |
| <code>currentDate</code> | 傳回目前日期/時間做為日期。         |

| 函式                            | Description                                   |
|-------------------------------|-----------------------------------------------|
| <code>currentTime</code>      | 以時間的形式傳回目前的日期/時間。                             |
| <code>currentTimestamp</code> | 以時間戳記的形式傳回目前的日期/時間。                           |
| <code>currentTimeUUID</code>  | 以 傳回目前的日期/時間timeuuid。                         |
| <code>fromJson</code>         | 將 JSON 字串轉換為所選資料欄的資料類型。                       |
| <code>maxTimeuuid</code>      | 傳回時間戳記或日期字串的最大可能timeuuid值。                    |
| <code>minTimeuuid</code>      | 傳回時間戳記或日期字串的最小timeuuid可能。                     |
| <code>now</code>              | 傳回新的唯一 timeuuid。                              |
| <code>toDate</code>           | 將 timeuuid或時間戳記轉換為日期類型。                       |
| <code>toJson</code>           | 以 JSON 格式傳回所選資料欄的資料欄值。                        |
| <code>token</code>            | 傳回分割區索引鍵的雜湊值。                                 |
| <code>toTimestamp</code>      | 將 timeuuid或 日期轉換為時間戳記。                        |
| <code>TTL</code>              | 傳回資料欄的過期時間，以秒為單位。                             |
| <code>typeAsBlob</code>       | 將指定的資料類型轉換為 blob。                             |
| <code>toUnixTimestamp</code>  | 將 timeuuid或時間戳記轉換為 bigInt。                    |
| <code>uuid</code>             | 傳回隨機第 4 版 UUID。                               |
| <code>writetime</code>        | 傳回指定資料欄值的時間戳記。                                |
| <code>dateOf</code>           | ( 已棄用 ) 擷取 的時間戳記timeuuid，並將值傳回為日期。            |
| <code>unixTimestampOf</code>  | ( 已棄用 ) 擷取 的時間戳記timeuuid，並以原始的 64 位元整數時間戳記傳回。 |

# Amazon Keyspaces 配額 (適用於 Apache Cassandra)

本節說明 Amazon Keyspaces (適用於 Apache Cassandra) 目前的配額和預設值。

## 主題

- [Amazon Keyspaces 服務配額](#)
- [提高或降低輸送量 \(已佈建的資料表\)](#)
- [Amazon Keyspaces 靜態加密](#)
- [Amazon Keyspaces 中使用使用者定義類型 \(UDTs\) 的配額和預設值](#)

## Amazon Keyspaces 服務配額

下表包含 Amazon Keyspaces (適用於 Apache Cassandra) 配額和預設值。有關可以調整哪些配額的資訊可在 [Service Quotas](#) 主控台中取得，您也可以請求增加配額。如需配額的詳細資訊，請聯絡 AWS 支援。

| 配額                      | Description                                                             | Amazon Keyspaces 預設 |
|-------------------------|-------------------------------------------------------------------------|---------------------|
| 每個 的最大金鑰空間 AWS 區域       | 此訂閱用戶每個區域的金鑰空間數目上限。您可以在 <a href="#">Service Quotas</a> 主控台中調整此預設值。      | 256                 |
| 每個 的資料表上限 AWS 區域        | 每個區域此訂閱用戶所有金鑰空間的資料表數量上限。您可以在 <a href="#">Service Quotas</a> 主控台中調整此預設值。 | 256                 |
| 使用 ADD REGION操作還原的資料量上限 | ADD REGION 操作可同時還原的資料大小上限。若要增加要同時還原的資料量，請聯絡 支援。                         | 10 TB               |
| 資料表結構描述大小上限             | 資料表結構描述的大小上限。                                                           | 350 KB              |

| 配額                           | Description                                                                                                      | Amazon Keyspaces 預設 |
|------------------------------|------------------------------------------------------------------------------------------------------------------|---------------------|
| 並行 DDL 操作上限                  | 每個區域允許此訂閱者的並行 DDL 操作數目上限。                                                                                        | 50                  |
| 每個連線的最大查詢數                   | 每秒可由單一用戶端 TCP 連線處理的 CQL 查詢數目上限。                                                                                  | 3000                |
| 資料列大小上限                      | 資料列的大小上限，不包括靜態資料欄資料。如需詳細資訊，請參閱 <a href="#">the section called “估算資料列大小”</a> 。                                    | 1 MB                |
| INSERT 和 UPDATE 陳述式中的資料欄數目上限 | CQL INSERT 或 UPDATE 陳述式中允許的欄數上限。關閉存留時間 (TTL) 時，INSERT 或 UPDATE 陳述式最多支援 225 個一般資料欄。如果開啟 TTL，單一操作最多可修改 166 個一般資料欄。 | 225/166             |
| 每個邏輯分割區的最大靜態資料               | 邏輯分割區中靜態資料的彙總大小上限。如需詳細資訊，請參閱 <a href="#">the section called “計算每個邏輯分割區的靜態資料欄大小”</a> 。                            | 1 MB                |
| 每個 INSELECT 陳述式的子查詢上限        | 您可以在 SELECT 陳述式中用於 IN 關鍵字子查詢數目上限。您可以在 <a href="#">Service Quotas</a> 主控台中調整此預設值。                                 | 100                 |

| 配額                      | Description                                                                                                         | Amazon Keyspaces 預設 |
|-------------------------|---------------------------------------------------------------------------------------------------------------------|---------------------|
| 每個的巢狀凍結集合數量上限<br>AWS 區域 | 當您為具有集合資料類型的資料欄使用 FROZEN關鍵字時，支援的巢狀集合數量上限。如需凍結集合的詳細資訊，請參閱 <a href="#">the section called “集合類型”</a> 。若要增加巢狀層級，請聯絡支援。 | 8                   |
| 已記錄批次中的陳述式數目上限          | 當您使用記錄的批次統計資料時，支援的 CQL 陳述式數目上限。如需批次陳述式的詳細資訊，請參閱 <a href="#">the section called “使用批次陳述式”</a> 。                      | 100                 |
| 使用靜態資料欄記錄批次中的陳述式數目上限    | 當您在記錄的批次統計資料中使用靜態資料欄時，支援的 CQL 陳述式數目上限。如需批次陳述式的詳細資訊，請參閱 <a href="#">the section called “使用批次陳述式”</a> 。               | 50                  |
| 記錄批次陳述式的大小上限            | 已記錄批次統計資料的大小上限。如需批次陳述式的詳細資訊，請參閱 <a href="#">the section called “使用批次陳述式”</a> 。                                      | 4MB                 |
| 未記錄批次中的陳述式數目上限          | 當您使用未記錄的批次統計資料時，支援的 CQL 陳述式數目上限。如需批次陳述式的詳細資訊，請參閱 <a href="#">the section called “使用批次陳述式”</a> 。                     | 30                  |



| 配額                                       | Description                                                                                                    | Amazon Keyspaces 預設 |
|------------------------------------------|----------------------------------------------------------------------------------------------------------------|---------------------|
| 每秒最大讀取輸送量                                | 每秒最大讀取輸送量：讀取請求單位 (RRUs) 或讀取容量單位 RCUs)，可配置給每個區域的資料表。您可以在 <a href="#">Service Quotas</a> 主控台中調整此預設值。             | 40,000              |
| 每秒寫入輸送量上限                                | 每秒寫入輸送量上限：寫入請求單位 (WRUs) 或寫入容量單位 WCUs)，可配置給每個區域的資料表。您可以在 <a href="#">Service Quotas</a> 主控台中調整此預設值。             | 40,000              |
| 使用 change-data-capture (CDC) 的資料表最大寫入輸送量 | 啟用 CDC 的 Amazon Keyspaces 資料表的最大寫入容量。此預設配額僅適用於佈建讀取/寫入容量模式中的資料表。您可以在 <a href="#">Service Quotas</a> 主控台中調整此預設值。 | 40,000              |
| 帳戶層級讀取輸送量（佈建）                            | 每個區域為帳戶配置的彙總讀取容量單位 (RCUs) 數目上限。這僅適用於佈建讀取/寫入容量模式中的資料表。您可以在 <a href="#">Service Quotas</a> 主控台中調整此預設值。           | 80,000              |
| 帳戶層級寫入輸送量（佈建）                            | 每個區域為帳戶配置的彙總寫入容量單位 (WCU) 數目上限。這僅適用於佈建讀取/寫入容量模式中的資料表。您可以在 <a href="#">Service Quotas</a> 主控台中調整此預設值。            | 80,000              |

| 配額                                  | Description                                                                                                                                                                                            | Amazon Keyspaces 預設 |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------|
| 每個帳戶每個區域的可擴展目標數量上限                  | 每個區域帳戶的可擴展目標數量上限。如果針對讀取容量啟用自動擴展，則 Amazon Keyspaces 資料表會計為一個可擴展的目標，如果針對寫入容量啟用自動擴展，則會計為另一個可擴展的目標。您可以透過選擇 Amazon Keyspaces 的可擴展目標，在 <a href="#">Service Quotas</a> 主控台中調整 Application Auto Scaling 的此預設值。 | 1,500               |
| 分割區索引鍵大小上限                          | 複合分割區索引鍵的大小上限。最多 3 個位元組的額外儲存空間會新增至中繼資料分割區索引鍵中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                            | 所有叢集資料欄的合併大小上限。中繼資料的每個叢集資料欄的原始大小最多可新增 4 個位元組的額外儲存空間。                                                                                                                                                   | 850 個位元組            |
| 使用Point-in-time復原 (PITR) 的並行資料表還原上限 | 每個訂閱用戶使用 PITR 的並行資料表還原數目上限為 4。您可以在 <a href="#">Service Quotas</a> 主控台中調整此預設值。                                                                                                                          | 4                   |
| 使用point-in-time復原 (PITR) 還原的資料量上限   | 可在 24 小時內使用 PITR 還原的資料大小上限。您可以在 <a href="#">Service Quotas</a> 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

您可以使用 主控台 `ReadCapacityUnits` 或 `WriteCapacityUnits` `ALTER TABLE` 陳述式，視需要增加 或 。新的設定值要在 `ALTER TABLE` 操作完成後才會生效。

新增佈建容量時，您無法超過每個帳戶的配額。您也可以視需要增加資料表的佈建容量。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` 陳述式中的每個資料表，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。這有效地將一天內的最大減少次數增加到 27 個 ( 第一個小時減少 4 個，而後續一天中的每個 1 小時時段減少 1 個 )。

## Amazon Keyspaces 靜態加密

從建立資料表開始，您可以在 24 小時時段內變更 AWS 擁有的 AWS KMS 金鑰和客戶受管 AWS KMS 金鑰之間的加密選項，最多四次。如果過去 6 小時內沒有變化，則允許額外變更。這可有效地將一天中的變更數量上限提高到八 ( 前六小時內有四項變更，而後續六小時時段每天各有一項變更 )。

您可以視需要變更加密選項以使用 AWS 擁有的 AWS KMS 金鑰，即使先前的配額已用盡。

除非您請求較高的數量，否則配額如下。若要請求提高服務配額，請參閱 [支援](#)。

## Amazon Keyspaces 中使用者定義類型 (UDTs) 的配額和預設值

### Amazon Keyspaces UDT 配額和預設值

下表包含與 Amazon UDTs 相關的配額和預設值。有關可以調整哪些配額的資訊可在 [Service Quotas](#) 主控台中取得，您也可以請求增加配額。如需這些配額的詳細資訊，請聯絡 AWS 支援。

| 配額                   | Description                 | Amazon Keyspaces 預設 |
|----------------------|-----------------------------|---------------------|
| 每個 UDTs 數目上限 AWS 區域  | 每個區域此訂閱用戶所有金鑰空間的 UDTs 數目上限。 | 256                 |
| 每個 UDT 的資料表數目上限      | 可參考相同 UDT 的資料表數目上限。         | 100                 |
| 每個資料表UDTs 數目上限       | 資料表可參考UDTs 數目上限。            | 50                  |
| UDTs 巢狀化層級上限         | UDTs 支援的最大巢狀深度。             | 8                   |
| 每個 UDTs直接子 UDT 數量上限  | UDTs支援的子 UDT 數目上限。          | 10                  |
| 每個 UDTs直接父系 UDT 數量上限 | UDTs支援的父系 UDT 數目上限。         | 10                  |
| UDT 結構描述大小上限         | UDT 結構描述的大小上限。              | 25 KB               |
| UDT 名稱長度上限           | UDT 名稱中的字元數上限。              | 48                  |
| UDT 欄位名稱長度上限         | UDT 欄位名稱中的字元數上限。            | 128                 |

# Amazon Keyspaces 的文件歷史記錄（適用於 Apache Cassandra）

下表說明自上次發行 Amazon Keyspaces（適用於 Apache Cassandra）以來文件的重要變更。如需有關此文件更新的通知，您可以訂閱 RSS 訂閱源。

- 文件最近更新時間：2026 年 1 月 28 日

| 變更                                                          | 描述                                                                          | 日期               |
|-------------------------------------------------------------|-----------------------------------------------------------------------------|------------------|
| <a href="#">您現在可以在 Amazon Keyspaces 中預熱新的和現有的資料表</a>        | 您現在可以warm-throughput 針對 Amazon Keyspaces 中的新資料表和現有資料表進行設定，以針對尖峰容量的預期尖峰預先暖機。 | 2026 年 1 月 28 日  |
| <a href="#">Amazon Keyspaces 現在支援整合式 AWS CloudShell 體驗</a>  | 您現在可以直接從 Amazon Keyspaces 主控台啟動預先驗證的 Shell。                                 | 2026 年 1 月 26 日  |
| <a href="#">Amazon Keyspaces 現在支援記錄的批次</a>                  | 您現在可以將多個陳述式合併為 Amazon Keyspaces 中記錄的批次。                                     | 2025 年 11 月 11 日 |
| <a href="#">Amazon Keyspaces TLS 憑證的變更</a>                  | 更新指示，以納入在 Amazon Trust Services (ATS) 鏈下發行的憑證。                              | 2025 年 10 月 25 日 |
| <a href="#">Amazon Keyspaces 現在支援 IPv4 和 IPv6 請求</a>        | Amazon Keyspaces 雙堆疊端點支援 IPv4 和 IPv6 組態。                                    | 2025 年 9 月 30 日  |
| <a href="#">NOW Amazon Keyspaces 中的所有 CQL 陳述式UUID現在都支援和</a> | 您現在可以在 Amazon Keyspaces 中的所有 CQL 陳述式中使用 NOW和 UUID函數。                        | 2025 年 9 月 8 日   |

## [Amazon Keyspaces Streams Kinesis Adapter 現已推出](#)

您現在可以使用 Kinesis Client Library (KCL) 處理 Amazon Keyspaces CDC 串流。

2025 年 7 月 2 日

## [Amazon Keyspaces 新受管政策](#)

Amazon Keyspaces 新增了新的受管政策 `KeyspacesCDCSERVICE_ROLE_POLICY`，將必要的許可授予服務連結角色 `AWS_SERVICE_ROLE_FOR_AMAZON_KEYSPACES_CDC`，以代表您將 Amazon Keyspaces CDC 串流指標資料發佈至 CloudWatch。如需詳細資訊，請參閱[使用 Amazon Keyspaces CDC 串流的角色](#)。

2025 年 7 月 2 日

## [Amazon Keyspaces 受管政策更新](#)

Amazon Keyspaces 將新許可新增至 `AmazonKeyspacesReadOnlyAccess_v2` 和 `AmazonKeyspacesReadOnlyAccess` 受管政策，以允許 IAM 主體檢視串流。Amazon Keyspaces 更新 `AmazonKeyspacesFullAccess` 受管政策的許可，以建立代表委託人發佈 CloudWatch `AWS_SERVICE_ROLE_FOR_AMAZON_KEYSPACES_CDC` 指標的服務連結角色。

2025 年 7 月 2 日

## [Amazon Keyspaces 變更資料擷取 \(CDC\) 串流現已推出](#)

啟用 CDC 串流以記錄 Amazon Keyspaces 資料表中的變更。

2025 年 7 月 2 日

|                                                            |                                                                                                                                                             |                  |
|------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#">Amazon Keyspaces 多區域複寫現在支援非洲（開普敦）區域。</a>       | 您現在可以定義多區域金鑰空間的非洲（開普敦）區域。                                                                                                                                   | 2025 年 3 月 31 日  |
| <a href="#">Amazon Keyspaces 多區域複寫現在支援無限制 AWS 區域。</a>      | 您現在可以為多區域金鑰空間定義任意數量的區域。                                                                                                                                     | 2025 年 3 月 25 日  |
| <a href="#">Amazon Keyspaces 支援多區域金鑰空間中的使用者定義類型 (UDTs)</a> | 您現在可以在多區域金鑰空間中建立和使用 UDTs。                                                                                                                                   | 2025 年 3 月 24 日  |
| <a href="#">Amazon Keyspaces 受管政策更新</a>                    | Amazon Keyspaces 將新許可新增至AmazonKey spacesFullAccess 受管政策，以允許 IAM 主體將新區域新增至現有的金鑰空間。這包括服務連結角色 <a href="#">AWSServiceRoleForAmazonKeyspacesReplication</a> 的更新。 | 2024 年 11 月 19 日 |
| <a href="#">將複本新增至 Amazon Keyspaces 中的多區域資料表</a>           | 您現在可以將新的 AWS 區域新增至現有的單一和多區域金鑰空間。                                                                                                                            | 2024 年 11 月 19 日 |
| <a href="#">支援 Amazon Keyspaces 中的使用者定義類型 (UDTs)</a>       | 透過原生支援 Amazon Keyspaces 中的 UDTs，您現在可以在應用程式中定義代表實際資料階層的資料結構。                                                                                                 | 2024 年 10 月 30 日 |
| <a href="#">ADD COLUMN 支援 Amazon Keyspaces 多區域複寫</a>       | Amazon Keyspaces 現在支援多區域資料表的結構描述變更。                                                                                                                         | 2024 年 9 月 17 日  |

|                                                                          |                                                                                                                                                                        |                  |
|--------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#">已更新 Amazon Keyspaces 的遷移指引</a>                               | 更新後的遷移指引概述如何建立遷移計畫，以成功從 Apache Cassandra 遷移至 Amazon Keyspaces，包括離線和線上遷移的不同策略。                                                                                          | 2024 年 6 月 28 日  |
| <a href="#">從 Amazon Elastic Kubernetes Service 連線至 Amazon Keyspaces</a> | 您現在可以遵循 step-by-step 教學，從 Amazon EKS 連線至 Amazon Keyspaces。                                                                                                             | 2024 年 2 月 7 日   |
| <a href="#">Amazon Keyspaces 多區域複寫支援佈建資料表</a>                            | Amazon Keyspaces 現在支援多區域資料表的佈建容量模式。                                                                                                                                    | 2024 年 1 月 23 日  |
| <a href="#">佈建資料表的 Amazon Keyspaces 自動擴展 APIs</a>                        | Amazon Keyspaces 現在提供使用佈建容量模式設定自動擴展的 CQL 和 AWS API 支援。                                                                                                                 | 2024 年 1 月 23 日  |
| <a href="#">CloudTrail 日誌中包含的 Amazon Keyspaces DML 活動</a>                | 您現在可以稽核 Amazon Keyspaces Data Manipulation Language (DML) API 呼叫 AWS CloudTrail。                                                                                       | 2023 年 12 月 20 日 |
| <a href="#">Amazon Keyspaces 支援 FROZEN 關鍵字</a>                           | Amazon Keyspaces 現在支援集合資料類型的 FROZEN 關鍵字。                                                                                                                               | 2023 年 11 月 15 日 |
| <a href="#">Amazon Keyspaces 受管政策更新</a>                                  | Amazon Keyspaces 將新的許可新增至 AmazonKeyspacesFullAccess 受管政策，以允許用戶端透過介面 VPC 端點連線至 Amazon Keyspaces 存取 Amazon EC2 執行個體，以使用來自 VPC 的網路資訊更新 Amazon Keyspaces system.peers 資料表。 | 2023 年 10 月 3 日  |



|                                               |                                                                                                                                                                         |                 |
|-----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------|
| <a href="#">Amazon Keyspaces 受管政策更新</a>       | Amazon Keyspaces 建立新的AmazonKeyspacesReadOnlyAccess_v2 受管政策，允許用戶端透過介面 VPC 端點連線至 Amazon Keyspaces 存取 Amazon EC2 執行個體，以使用來自 VPC 的網路資訊更新 Amazon Keyspaces system.peers 資料表。 | 2023 年 9 月 12 日 |
| <a href="#">在 Amazon Keyspaces 中建立連線的最佳實務</a> | 了解如何改善和最佳化 Amazon Keyspaces 中的用戶端驅動程式組態。                                                                                                                                | 2023 年 6 月 30 日 |
| <a href="#">Amazon Keyspaces 現在已記錄系統金鑰空間</a>  | 了解系統金鑰空間中存放的內容，以及如何查詢它們以取得 Amazon Keyspaces 中的實用資訊。                                                                                                                     | 2023 年 6 月 21 日 |
| <a href="#">Amazon Keyspaces 現在支援多區域複寫</a>    | Amazon Keyspaces 多區域複寫透過為您提供改善的容錯能力、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  |
| <a href="#">Amazon Keyspaces 受管政策更新</a>       | Amazon Keyspaces 將新許可新增至 AmazonKeyspacesFullAccess 受管政策，以允許 Amazon Keyspaces 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  |
| <a href="#">Amazon Keyspaces 支援 IN 關鍵字</a>    | Amazon Keyspaces 現在支援 SELECT 陳述式中的 IN 關鍵字。                                                                                                                              | 2023 年 4 月 25 日 |

|                                                             |                                                                                                                                         |                  |
|-------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------|------------------|
| <a href="#">Amazon Keyspaces 和介面 VPC 端點的跨帳戶存取</a>           | 了解如何使用 VPC 端點實作 Amazon Keyspaces 的跨帳戶存取。                                                                                                | 2023 年 4 月 20 日  |
| <a href="#">Amazon Keyspaces 支援用戶端時間戳記</a>                  | Amazon Keyspaces 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判斷不同用戶端變更相同資料的寫入操作順序。                                                       | 2023 年 3 月 14 日  |
| <a href="#">Amazon Keyspaces 和介面 VPC 端點入門</a>               | 在此step-by-step教學中，了解如何從 VPC 連線至 Amazon Keyspaces。                                                                                       | 2023 年 3 月 1 日   |
| <a href="#">最佳化 Amazon Keyspaces 資料表的成本</a>                 | 最佳實務和指導可協助您識別最佳化現有 Amazon Keyspaces 資料表成本的策略。                                                                                           | 2023 年 2 月 17 日  |
| <a href="#">現在Murmur3Partitioner 是預設值</a>                   | 現在Murmur3Partitioner 是 Amazon Keyspaces 中的預設分割區。                                                                                        | 2022 年 11 月 17 日 |
| <a href="#">Amazon Keyspaces 現在支援 Murmur3Partitioner</a>    | Murmur3Partitioner 現已在 Amazon Keyspaces 中提供。                                                                                            | 2022 年 11 月 9 日  |
| <a href="#">支援空白字串和 Blob 值的更新</a>                           | Amazon Keyspaces 現在也支援空字串和 Blob 值做為叢集資料欄值。                                                                                              | 2022 年 10 月 19 日 |
| <a href="#">Amazon Keyspaces 現在可在 中使用 AWS GovCloud (US)</a> | Amazon Keyspaces 現在可在 中使用，AWS GovCloud (US) Region 且符合 FedRAMP-High 合規。如需可用端點的資訊，請參閱 <a href="#">AWS GovCloud (US) Region FIPS 端點</a> 。 | 2022 年 8 月 4 日   |

[使用 Amazon CloudWatch 監控 Amazon Keyspaces 資料表儲存成本](#)

Amazon Keyspaces 現在可協助您使用 `BillableTableSizeInBytes` CloudWatch 指標監控和追蹤一段時間內的資料表儲存成本。

2022 年 6 月 14 日

[Amazon Keyspaces 現在支援 Terraform](#)

您現在可以使用 Terraform 在 Amazon Keyspaces 中執行資料定義語言 (DDL) 操作。

2022 年 6 月 9 日

[Amazon Keyspaces token 函數支援](#)

Amazon Keyspaces 現在可協助您使用 token 函數來最佳化應用程式查詢。

2022 年 4 月 19 日

[Amazon Keyspaces 與 Apache Spark 整合](#)

Amazon Keyspaces 現在使用開放原始碼 Spark [Cassandra 連接器](#)，協助您更輕鬆地讀取和寫入 Apache Spark 中的資料。

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 和管理金鑰空間和資料表 AWS CLI。API 參考指南會詳細說明支援的控制平面操作。

2022 年 3 月 2 日

[如何在使用 Amazon Keyspaces 時疑難排解常見的組態問題。](#)

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 支援存留時間 \(TTL\)。](#)

Amazon Keyspaces 存留時間 (TTL) 可協助您簡化應用程式邏輯，並透過自動從資料表中過期資料來最佳化儲存體的價格。

2021 年 10 月 18 日

[使用 DSBulk 將資料遷移至 Amazon Keyspaces。](#)

使用 DataStax 大量載入器 (DSBulk) 將資料從 Apache Cassandra 遷移至 Amazon Keyspaces 的 Step-by-step 教學課程。

2021 年 8 月 9 日

[Amazon Keyspaces 支援 system.peers 資料表中的 VPC 端點項目。](#)

Amazon Keyspaces 可讓您將可用的介面 VPC 端點資訊填入 system.peers 資料表，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 Keyspaces 的 IAM 受管政策現在包含列出和檢視存放於之可用客戶受管 AWS KMS 金鑰的許可 AWS KMS。

2021 年 6 月 1 日

[Amazon Keyspaces 支援客戶受管 AWS KMS 金鑰。](#)

Amazon Keyspaces 可讓您控制存放在中的客戶受管 AWS KMS 金鑰，AWS KMS 以進行靜態加密。

2021 年 6 月 1 日

[Amazon Keyspaces 支援 JSON 語法](#)

Amazon Keyspaces 支援 INSERT 和 SELECT 操作的 JSON 語法，可協助您更輕鬆地讀取和寫入 JSON 文件。

2021 年 1 月 21 日

[Amazon Keyspaces 支援靜態資料欄](#)

Amazon Keyspaces 現在可協助您使用靜態資料欄，在多個資料列之間有效率地更新和存放常用資料。

2020 年 11 月 9 日

[Amazon Keyspaces 的 GA 版  
NoSQL Workbench 支援](#)

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 用戶端適用於 Windows、macOS 和 Linux。

2020 年 10 月 28 日

[Amazon Keyspaces 的 NoSQL  
Workbench 支援預覽版本](#)

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 用戶端適用於 Windows、macOS 和 Linux。

2020 年 10 月 5 日

[以程式設計方式存取 Amazon  
Keyspaces 的新程式碼範例](#)

我們會繼續新增程式碼範例，以程式設計方式存取 Amazon Keyspaces。範例現在適用於支援 Apache Cassandra 3.11.2 版的 Java、Python、Go、C# 和 Perl Cassandra 驅動程式。

2020 年 7 月 17 日

[Amazon Keyspaces point-in-  
time\(PITR\)](#)

Amazon Keyspaces 現在提供 point-in-time 復原 (PITR)，透過提供資料表資料的連續備份，協助保護您的資料表免於意外寫入或刪除操作。

2020 年 7 月 9 日

## [Amazon Keyspaces 一般可用性](#)

透過先前在預覽期間稱為 Amazon Managed Apache Cassandra Service (MCS) 的 Amazon Keyspaces，您可以使用 Cassandra Query Language (CQL) 程式碼、Apache 2.0 授權的 Cassandra 驅動程式，以及您目前已使用的開發人員工具。

2020 年 4 月 23 日

## [Amazon Keyspaces 自動擴展](#)

Amazon Keyspaces (適用於 Apache Cassandra) 與 Application Auto Scaling 整合，透過自動調整輸送量容量，協助您有效佈建可變工作負載的輸送量容量，以回應實際的應用程式流量。

2020 年 4 月 23 日

## [Amazon Keyspaces 的界面虛擬私有雲端 \(VPC\) 端點](#)

Amazon Keyspaces 會在服務與 VPC 之間提供私有通訊，讓網路流量不會離開 Amazon 網路。

2020 年 4 月 16 日

## [標籤型存取政策](#)

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來協調資料欄值的增量和遞減。

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

## [CloudFormation 支援](#)

Amazon Keyspaces 現在可協助您使用自動化資源的建立和管理 CloudFormation。

2020 年 3 月 25 日

### [支援 IAM 角色和政策以及 SigV4 身分驗證](#)

新增了有關如何使用 [AWS Identity and Access Management \(IAM\)](#) 來管理 Amazon Keyspaces 的存取許可和實作安全政策的資訊，以及如何使用 DataStax Java Driver for Cassandra 的身分驗證外掛程式，以程式設計方式使用 IAM 角色和聯合身分存取 Amazon Keyspaces。

2020 年 3 月 17 日

### [讀取/寫入容量模式](#)

Amazon Keyspaces 現在支援兩種讀取/寫入輸送量模式。讀取/寫入容量模式會控制如何向您收取讀取和寫入輸送量的費用，以及如何管理資料表輸送量容量。

2020 年 2 月 20 日

### [初始版本](#)

本文件涵蓋 Amazon Keyspaces 的初始版本（適用於 Apache Cassandra）。

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。