Amazon Keyspaces (for Apache Cassandra)

开发人员指南

亚马逊云科技

Amazon Keyspaces (for Apache Cassandra): 开发人员指南

Table of Contents

什么是 Amazon Keyspaces（针对 Apache Cassandra）？

Amazon Keyspaces（针对 Apache Cassandra）是一种可扩展、高可用、托管的 Apache Cassandra 兼容数据库服务。使用 Amazon Keyspaces，您无需预置、修补或管理服务器，并且无需安装、维护或操作软件。

Amazon Keyspaces 是无服务器服务器的，因此您只需为您使用的资源付费，并且该服务会根据应用程序流量自动扩展和缩减表。您可以构建每秒可处理数千个请求且吞吐量和存储空间几乎无限的应用程序。

> Note
>
> Apache Cassandra 是一个开源宽列数据存储，设计为处理海量数据。有关更多信息，请参阅 Apache Cassandra。

Amazon Keyspaces 使您能够轻松地在Amazon Web Services 云. 只需点击几下Amazon使用管理控制台或使用几行代码，您可以在 Amazon Keyspaces 中创建键空间和表，而无需部署任何基础设施或安装软件。

借助 Amazon Keyspaces，您可以在上运行现有的 Cassandra 工作负载Amazon使用与您今天使用的相同 Cassandra 应用程序代码和开发人员工具。

有关可用的列表Amazon Web Services 区域和终端节点，请参阅Amazon Keyspaces 的服务终端节点.

我们建议您首先阅读以下部分：

主题

Amazon Keyspaces：工作方式

Amazon Keyspaces 消除了 Cassandra 的管理开销。要了解为什么首先从 Cassandra 架构开始然后与 Amazon Keyspaces 进行比较是很有帮助的。

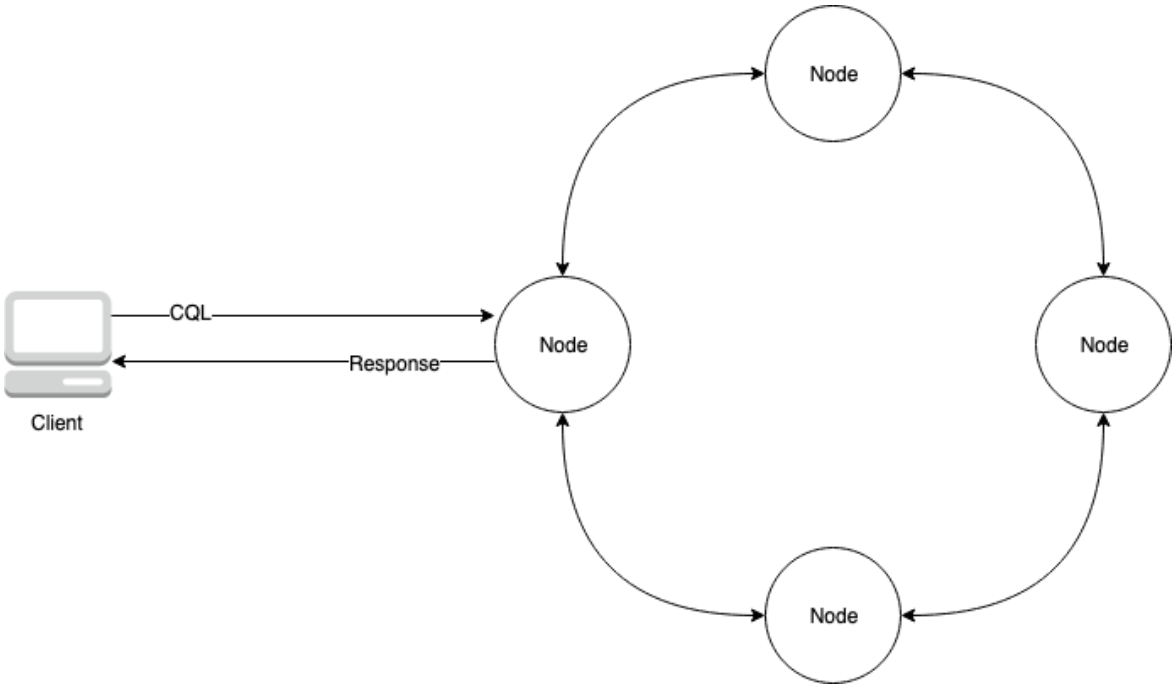
主题

高级别架构：Apache Cassandra vs 亚马逊 Keyspaces

传统的 Apache Cassandra 部署在由一个或多个节点组成的集群中。您负责管理各个节点，并随着集群的扩展添加和删除节点。

客户端程序通过连接到其中一个节点并发出 Cassandra 查询语言 (CQL) 语句，来访问 Cassandra。CQL 类似于 SQL，后者是关系数据库中常用的语言。虽然 Cassandra 并不是关系数据库，不过 CQL 提供了一个熟悉的界面，可用于查询和操作 Cassandra 中的数据。

下图显示了一个由四个节点构成的简单 Apache Cassandra 集群。



生产 Cassandra 部署可能由数百个节点构成，这些节点在一个或多个物理数据中心内的数百台物理计算机上运行。这会给应用程序开发人员带来运营负担，这些人员除了安装、维护和操作软件之外，还需要预置、修补和管理服务器。

使用 Amazon Keyspaces（针对 Apache Cassandra），您无需预置、修补或管理服务器，因此可以专注于构建更好的应用程序。Amazon Keyspaces 为读取和写入提供了两种吞吐容量模式：按需和预置。您可以根据工作负载的可预测性和可变性选择表的吞吐容量模式，以优化读取和写入价格。

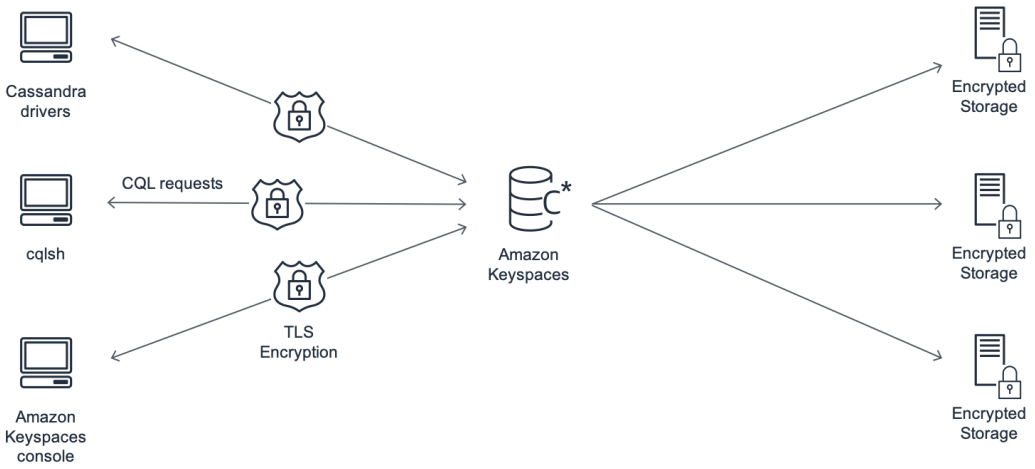
借助按需模式，您只需为您的应用程序实际执行的读取和写入付费。您无需事先指定表的吞吐容量。Amazon Keyspaces 几乎可以立即容纳应用程序上升或下降的流量，这使其成为流量不可预测的应用程序的理想选择。

如果您的应用程序流量可预测，并且可以提前预测表的容量需求，那么预置容量模式可以帮助您优化吞吐量的价格。使用预置容量模式时，请指定您的应用程序预计每秒需要执行的读取和写入次数。您可以通过启用自动扩展来自动增加和减少表的预置容量。

在以下情况下，您可以每天更改一次表的容量模式：您需要了解有关工作负载的流量模式的更多信息，或如果您预计流量会突增（例如，您预计会发生导致大量表流量的重大事件）。有关预置读取和写入容量的更多信息，请参阅the section called “读/写容量模式” (p. 88)。

Amazon Keyspaces（针对 Apache Cassandra）将您的三个数据副本存储在多个中可用区为了实现持久性和高可用性。此外，您可以从专为满足大多数安全敏感型组织的要求而打造的数据中心和网络架构中受益。在您创建新的 Amazon Keyspaces 表时，将自动启用静态加密，并且所有客户端连接都需要传输层安全性 (TLS)。其他Amazon安全功能包括监控、Amazon Identity and Access Management, 和Virtual Private Cloud (VPC) 终端节点. 有关所有可用安全功能的概述，请参阅安全性 (p. 145)。

下图演示了 Amazon Keyspaces 的架构。



客户端程序通过连接到预先确定的终端节点（主机名和端口号）并发出 CQL 语句来访问 Amazon Keyspaces。有关可用终端节点的列表，请参阅the section called “服务端点” (p. 21)。

Cassandra 数据模型

如何为业务案例建模数据，对于实现 Amazon Keyspaces 的最佳性能至关重要。较差的数据模型会显著地降低性能。

尽管 CQL 与 SQL 类似，但 Cassandra 后端与关系数据库后端存在明显的不同，必须采取不同的方法。下面是需要考虑的一些更重要的问题：

存储

可以在表中直观地显示您的 Cassandra 数据，其中每行表示一条记录，每列表示该记录中的一个字段。

桌设计：先查询

CQL 中没有 `JOIN`。因此，您应该根据数据的性质，以及需要如何访问数据来满足业务使用案例的要求，设计您的表。这可能会导致重复数据的逆规范化。您应该专门为特定的访问模式设计每个表。

分区

您的数据存储在磁盘分区中。存储数据的分区数以及数据在分区之间的分布方式由您的分区键 决定。分区键的定义方式会对查询的性能产生重大影响。

主键

在 Cassandra 中，数据以键/值对的形式存储。为此，每个 Cassandra 表必须有一个主键，这是表中每一行的键。主键是一个必需的分区键与多个可选的聚类列的组合。组成主键的数据在表中的所有记录之间必须是唯一的。

- 分区键— 主键的分区键部分是必需的，可确定数据存储在集群的哪个分区中。分区键可以是单个列，也可以是由两列或多个列组成的复合值。如果单列分区键会导致单个分区，或者导致极少数分区具有大部分数据，从而承担大部分磁盘 I/O 操作，则应使用复合分区键。
- 群集专栏— 主键的可选聚类列部分决定如何在每个分区中聚类和排序数据。如果您在主键中包含聚类列，聚类列可以有一个或多个列。如果聚类列中有多个列，则排序顺序由各列在聚类列中从左到右的列出顺序决定。

从应用程序访问亚马逊 Keyspaces

Amazon Keyspaces（针对 Apache Cassandra）实施 Apache Cassandra 查询语言 (CQL) API，因此您可以使用您已在使用的 CQL 和 Cassandra 驱动程序。更新应用程序与更新您的 Cassandra 驱动程序或`cqlsh`配置以指向 Amazon Keyspaces 服务终端节点。

Note

为了帮助您开始使用，您可以在上的 Amazon Keyspaces 代码示例存储库中使用各种 Cassandra 客户端驱动程序找到连接到 Amazon Keyspaces 的端到端代码示例GitHub.

考虑以下 Python 程序，该程序连接到 Cassandra 集群并查询表。

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

ksp = 'MyKeyspace'
tbl = 'WeatherData'

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

session.execute('USE ' + ksp)

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

要针对 Amazon Keyspaces 运行同一个程序，您需要：

- 添加群集终端节点和端口：例如，可以将主机替换为服务终端节点，例如`cassandra.us-east-2.amazonaws.com`带有以下端口号：`9142`.
- 添加TLS/SSL 配置：有关使用 Cassandra 客户端 Python 驱动程序添加 TLS/SSL 配置以连接到 Amazon Keyspaces 的更多信息，请参阅使用 Cassandra Python 客户端驱动程序以编程方式访问亚马逊 Keyspaces (p. 35).

Amazon Keyspaces 使用案例

下面仅是您可以使用 Amazon Keyspaces 的部分方法：

- 构建需要低延迟的应用— 为需要个位数毫秒级延迟的应用程序高速处理数据，此类应用程序包括工业设备维护、贸易监控、车队管理和路线优化等。
- 使用开源技术构建应用— 在上构建应用程序Amazon使用开源 Cassandra API 和驱动程序，包括支持多种编程语言的 API 和驱动程序，例如 Java、Python、Ruby、Microsoft .NET、Node.js、PHP、C++、Perl 和 Go。有关代码示例，请参阅 代码示例和工具 (p. 85)。
- 将 Cassandra 工作负载移动到云— 自行管理 Cassandra 桌非常耗时且成本高昂。借助 Amazon Keyspaces，您可以在Amazon Web Services 云而不管理基础设施。有关更多信息，请参阅无服务器资源管理 (p. 88)。

什么是 Cassandra 查询语言 (CQL)？

Cassandra 查询语言(CQL) 是与 Apache Cassandra 沟通的主要语言。Amazon Keyspaces（针对 Apache Cassandra）与 CQL 3.x API 兼容（与版本 2.x 向后兼容）兼容。

要运行 CQL 查询，可以执行以下操作之一：

- 在 上使用 CQL 编辑器Amazon Web Services Management Console.
- 在`cqlsh`客户端。
- 使用 Apache 2.0 许可的 Cassandra 客户端驱动程序以编程方式运行它们。

有关使用这些方法访问 Amazon Keyspaces 的更多信息，请参阅访问 Amazon Keyspaces（针对 Apache Cassandra）（p. 14).

有关 CQL 的更多信息，请参阅Amazon Keyspaces（针对 Apache Cassandra）（p. 203)。

与 Apache Cassandra 相比，Amazon Keyspaces（针对 Apache Cassandra）如何？

Amazon Keyspaces（针对 Apache Cassandra）显示为包含 9 个节点的 Apache Cassandra 3.11.2 集群，可支持与 Apache Cassandra 3.11.2 兼容的驱动程序和客户端。Amazon Keyspaces 支持 3.x Cassandra 查询语言 (CQL) API，并且与版本 2.x 向后兼容。借助 Amazon Keyspaces，您可以在上运行 Cassandra 工作负载Amazon使用当前的 Cassandra 应用程序代码、获得 Apache 2.0 许可的驱动程序以及工具。

Amazon Keyspaces 支持所有常用的 Cassandra 数据层面操作，例如创建键空间和表、读取数据和写入数据。Amazon Keyspaces 是无服务器的，因此您无需预置、修补或管理服务器。您也不必安装、维护或操作软件。因此，不需要用来管理集群和节点设置的 Cassandra 控制层面 API 操作即可使用 Amazon Keyspaces。

自动配置诸如复制系数和一致性级别之类的设置，可为您提供高可用性、持久性和个位数毫秒级性能。

主题

功能差异：亚马逊 Keyspaces 与 Apache Cassandra

下面是 Amazon Keyspaces 与 Apache Cassandra 之间的功能差异。

主题

- 系统表 (p. 8)

Apache Cassandra API、操作和数据类型

Amazon Keyspaces 支持所有常用的 Cassandra 数据层面操作，例如创建键空间和表、读取数据和写入数据。要查看当前支持的项，请参阅Amazon Keyspaces 中支持的 Cassandra API、操作、函数和数据类型 (p. 8)。

异步创建和删除键空间和表

Amazon Keyspaces 异步执行数据定义语言 (DDL) 操作，例如创建和删除键空间和表。要了解如何监控资源的创建状态，请参阅。the section called “CREATE 键空间” (p. 102)和the section called “创建表” (p. 103). 有关 CQL 语言参考中的 DDL 语句列表，请参阅the section called “DDL 语句” (p. 207).

身份验证和授权

Amazon Keyspaces (for Apache Cassandra) 使用Amazon Identity and Access Management(IAM) 用于用户身份验证和授权，并支持与 Apache Cassandra 等效的授权策略。因此，Amazon Keyspaces 不支持 Apache Cassandra 的安全配置命令。

Batch

Amazon Keyspaces 支持未记录的批处理命令，批处理中最多包含 30 条命令。批处理中仅允许无条件的 INSERT、UPDATE 或 DELETE 命令。不支持记录的批处理。

群集配置

Amazon Keyspaces 是无服务器的，因此没有要配置的集群、主机或 Java 虚拟机 (JVM)。Amazon Keyspaces Cassandra 的压缩、缓存、垃圾回收和绽放筛选设置不适用于 Amazon Keyspaces，如果指定，则将被忽略。

CQL 查询吞吐量调整

Amazon Keyspaces 支持每秒每个 TCP 连接 3,000 次 CQL 查询，而驱动程序可建立的连接数不受限。

大多数开源的 Cassandra 驱动程序都建立了一个与 Cassandra 的连接池，并对该连接池进行负载平衡查询。Amazon Keyspaces 向驱动程序公开 9 个对等 IP 地址，大多数驱动程序的默认行为是建立到每个对等 IP 地址的单个连接。因此，使用默认设置的驱动程序的最大 CQL 查询吞吐量将是每秒 27,000 次 CQL 查询。

要增大此数字，我们建议增加驱动程序在其连接池中维护的每个 IP 地址的连接数。例如，如果将每个 IP 地址的最大连接数设置为 2，则将使驱动程序的最大吞吐量增加一倍，达到每秒 54000 次 CQL 查询。

空字符串

Amazon Keyspaces 支持空字符串和 blob 值。但是，不支持空字符串和 blob 作为聚类列值。

轻量级事务

Amazon Keyspaces (for Apache Cassandra) 完全支持对比和设置功能INSERT、UPDATE, 和DELETE命令，称为轻量级事务（LWT）在阿帕奇·卡桑德拉。作为无服务器产品，Amazon Keyspaces（针对 Apache Cassandra）可提供任何规模（包括轻量级事务）的一致性能。使用 Amazon Keyspaces，使用轻量级事务不会造成性能损失。

负载均衡

这些区域有：`system.peers`表条目对应于 Amazon Keyspaces 负载均衡器。要获得最佳结果，我们建议使用轮询负载平衡策略并调整每个 IP 的连接数以满足应用程序的需求。

分页

Amazon Keyspaces 根据它读取的用于处理请求的行数而不是结果集中返回的行数对结果进行分页。因此，某些页面包含的行可能少于您在 PAGE SIZE 中为筛选查询指定的行数。此外，Amazon Keyspaces 在读取 1 MB 数据后自动对结果进行分页，以便为客户提供一致的单位数毫秒读取性能。有关更多信息，请参阅 the section called “分页结果” (p. 108)。

分区者

亚马逊 Keyspaces 为您提供了使用亚马逊 Keyspaces 的选择`DefaultPartitioner`或者 Cassandra 兼容`RandomPartitioner`. 使用亚马逊 Keyspaces，您可以安全地更改账户的分区程序，而无需重新加载您的亚马逊 Keyspaces 数据。客户下次连接时将自动看到新的分区程序设置。

预准备语句

Amazon Keyspaces 支持将准备好的语句用于数据操作语言 (DML) 操作，例如读取和写入数据。Amazon Keyspaces 目前不支持使用预准备语句执行数据定义语言 (DDL) 操作（如创建表和键空间）。DDL 操作必须在准备好的语句之外运行。

范围删除

亚马逊 Keyspaces 支持删除范围内的行。范围是分区内的一组连续行。您可以使用 WHERE 子句在 DELETE 操作中指定范围。您可以将范围指定为整个分区。

此外，您可以通过使用关系运算符（例如 “>”、“<”）或者通过包含分区键并省略一个或多个聚类列来指定范围作为分区内连续行的子集。使用 Amazon Keyspaces，您可以在单个操作中删除范围内最多 1,000 行。此外，范围删除是原子的，但不是孤立的。

系统表

Amazon Keyspaces 填充 Apache 2.0 开源 Cassandra 驱动程序所需的系统表。对客户端可见的系统表包含经过身份验证的用户所特有的信息。系统表完全由 Amazon Keyspaces 控制，并且是只读的。

需要对系统表进行只读访问，您可以使用 IAM 访问策略对其进行控制。有关更多信息，请参阅 the section called “使用策略管理访问” (p. 161)。您必须为系统表定义基于标签的访问控制策略，具体取决于是否使用 AmazonSDK 或 Cassandra 查询语言 (CQL) API 通过 Cassandra 驱动程序和开发人员工具调用。要了解有关系统表基于标记的访问控制的详细信息，请参阅the section called “ 基于标签的亚马逊Keyspaces 资源访问权限” (p. 170).

如果您使用访问亚马逊 KeyspacesAmazon VPC 终端节点 (p. 196)，你会看到`system.peers`Amazon Keyspaces 有权查看的每个 Amazon VPC 终端节点的表。因此，你的 Cassandra 司机可能会发出警告消息 (p. 199)关于控制节点本身`system.peers`表。您可以放心地忽略此警告。

Amazon Keyspaces 中支持的 Cassandra API、操作、函数和数据类型

Cassandra 查询语言 (CQL) 3.11 API 兼 Keyspaces（向后兼容版本 2.x）。

Amazon Keyspaces 支持所有常用的 Cassandra 数据层面操作，例如创建键空间和表、读取数据和写入数据。

以下部分列出了支持的功能。

主题

Cassandra API 支持

API 操作	支持
`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`	否
`SELECT`	是
`INSERT`	是
`DELETE`	是
`UPDATE`	是
`USE`	是
`CREATE TYPE`	否
`ALTER TYPE`	否
`DROP TYPE`	否
`CREATE TRIGGER`	否
`DROP TRIGGER`	否
`CREATE FUNCTION`	否
`DROP FUNCTION`	否

API 操作	支持
`CREATE AGGREGATE`	否
`DROP AGGREGATE`	否
`CREATE MATERIALIZED VIEW`	否
`ALTER MATERIALIZED VIEW`	否
`DROP MATERIALIZED VIEW`	否
`TRUNCATE`	否

Cassandra 控制层面 API 支持

由于托管了 Amazon Keyspaces，因此不需要用来管理集群和节点设置的 Cassandra 控制层面 API 操作。因此，以下 Cassandra 功能不适用。

功能	Reason
持久写入切换	所有写入都是持久性的
读取修复设置	不适用
GC 宽限期秒数	不适用
Bloom 筛选条件设置	不适用
压缩设置	不适用
Compression settings (压缩设置)	不适用
缓存设置	不适用
安全设置	被 IAM 取代

Cassandra 数据层面 API 支持

功能	支持
对 SELECT 和 INSERT 语句的 JSON 支持	是
静态列	是
生存时间 (TTL)	是

Cassandra 函数支持

有关支持的函数的更多信息，请参阅the section called “内置函数” (p. 217).

函数	支持
Aggregate 函数	否
Blob转换	是
Cast	是
Datetime 函数	是
时间转换函数	是
TimeUuid 函数	是
Token	是
User defined functions (UDF)	否
Uuid	是

卡桑德拉数据类型支持

数据类型	支持
ascii	是
bigint	是
blob	是
boolean	是
counter	是
date	是
decimal	是
double	是
float	是
frozen	否
inet	是
int	是
list	是
map	是
set	是
smallint	是
text	是
time	是

数据类型	支持
timestamp	是
timeuuid	是
tinyint	是
tuple	是
user-defined types (UDT)	否
uuid	是
varchar	是
varint	是

亚马逊 Keyspaces 中支持的 Apache Cassandra 一致性级别

此部分中的主题介绍了 Amazon Keyspaces 中的读取和写入操作支持哪些 Apache Cassandra 一致性级别。

主题

写入一致性级别

跨多个可用区 Keyspaces 复所有写入操作三次以实现持久性和高可用性。在使用 LOCAL_QUORUM 一致性级别确认写入之前，将持久存储写入。就每个 1 KB 的写入而言，对于使用预置容量模式的表，将向您收取 1 个写入容量单位 (WCU) 的费用，对于使用按需模式的表，将收取 1 个写入请求单位 (WRU) 的费用。

读取一致性级别

Amazon Keyspaces 支持三个读取一致性级别：ONE、LOCAL_ONE, 和LOCAL_QUORUM. 在一个LOCAL_QUORUM读取，Amazon Keyspaces 将返回一个响应来反映来自所有先前成功的写入操作的最近更新。通过使用一致性级别，ONE 或 LOCAL_ONE 可以提高读取请求的性能和可用性，但响应可能无法反映最近完成的写入操作的结果。

就每个使用 ONE 或 LOCAL_ONE 一致性的 4 KB 读取操作而言，对于使用预置容量模式的表，将向您收取 0.5 个读取容量单位 (RCU) 的费用，对于使用按需模式的表，将向您收取 0.5 个读取请求单位 (RRU) 的费用。就每个使用 LOCAL_QUORUM 一致性的 4 KB 读取操作而言，对于使用预置容量模式的表，将向您收取 1 个读取容量单位 (RCU) 的费用，对于使用按需模式的表，将向您收取 1 个读取请求单位 (RRU) 的费用。

根据每个表的读取一致性和读取容量吞吐量模式对每个 4 KB 的读取操作进行计费

一致性级别	已预置	按需
ONE	0.5 个 RCU	0.5 个 RRU
LOCAL_ONE	0.5 个 RCU	0.5 个 RRU

一致性级别	已预置	按需
LOCAL_QUORUM	1 个 RCU	1 个 RRU

不受支持的一致性级别

以下一致性级别不受 Keyspaces 支持，并且将导致异常。

不受支持的一致性级别

Apache Cassandra	Amazon Keyspaces
EACH_QUORUM	不支持
QUORUM	不支持
ALL	不支持
TWO	不支持
THREE	不支持
ANY	不支持
SERIAL	不支持
LOCAL_SERIAL	不支持

访问 Amazon Keyspaces（针对 Apache Cassandra）

您可以使用控制台访问 Amazon Keyspaces，或者通过运行编程方式通过运行`cqlsh`或者使用 Apache 2.0 许可的 Cassandra 驱动程序。亚马逊 Keyspaces 支持与 Apache Cassandra 3.11.2 兼容的驱动程序和客户端。在访问 Amazon Keyspaces 之前，您必须完成以下两个步骤：

1. 注册 Amazon (p. 14)
2. 设置 Amazon Identity and Access Management (p. 14)

注册 Amazon

要使用 Amazon Keyspaces 服务，您必须拥有Amazon Web Services 账户. 如果您还没有账户，系统会在您注册时提示您创建一个。如果您没有使用注册的任何 Amazon 产品，系统将不会针对它们向您收费。

注册 Amazon

1. 打开 https://portal.aws.amazon.com/billing/signup。
2. 按照屏幕上的说明进行操作。

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

设置 Amazon Identity and Access Management

使用以下方式管理对 Amazon Keyspaces 资源的访问Amazon Identity and Access Management(IAM). 使用 IAM，您可以将策略附加到 IAM 用户、角色和联合身份，从而授予对特定资源的读写权限。例如，您可以仅授予 IAM 用户对键空间和表子集的只读访问权限。

以下示例 IAM 策略授予对 Amazon Keyspaces 资源的完全读写访问权限，仅建议进行试用。有关遵循安全准则的示例策略，请参阅the section called “Amazon Keyspaces” (p. 167)。

1. 创建 Amazon Identity and Access Management 用户。
2. 创建以下策略并将其附加到刚创建的用户。

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

要在创建后访问亚马逊 KeyspacesAmazon Web Services 账户和 IAM 策略，请参阅以下章节：

使用控制台访问亚马逊 Keyspaces

您可以访问 Amazon Keyspaces 的控制台，请访问：https://console.amazonaws.cn/keyspaces/home.

您可以使用控制台在 Amazon Keyspaces 中执行以下操作：

- 创建、删除、描述和列出键空间和表。
- 插入、更新和删除数据。
- 使用 CQL 编辑器运行查询。

要了解如何创建 Amazon Keyspaces 键空间和表以及使用示例应用程序数据进行设置，请参阅开始使用 Amazon Keyspaces（针对 Apache Cassandra）(p. 46).

以编程方式连接到亚马逊 Keyspaces

本主题概述了以编程方式连接到 Amazon Keyspaces 所需的步骤。它将指导您创建特定于服务的凭据并列出可用Amazon服务终端节点。最后一节介绍如何使用 cqlsh 连接到 Amazon Keyspaces。有关使用不同的 Apache Cassandra 驱动程序连接到 Amazon Keyspaces 的分步教程，请参阅the section called “使用 Cassandra 客户端驱动程序” (p. 28).

> Note
>
> 为了帮助您开始使用，您可以找到 end-to-end 通过在上的 Amazon Keyspaces 代码示例存储库中使用各种 Cassandra 客户端驱动程序连接到 Amazon Keyspaces 的代码示例GitHub.

亚马逊 Keyspaces 支持与 Apache Cassandra 3.11.2 兼容的驱动程序和客户端。它假定您已完成Amazon中的设置说明访问 Amazon Keyspaces (p. 14).

如果您已有Amazon Web Services 账户要了解如何以编程方式使用 cqlsh 访问 Amazon Keyspaces，请参阅以下主题：

主题

创建凭据以编程方式访问 Amazon Keyspaces

要向用户和应用程序提供对 Amazon Keyspaces 资源进行编程访问的凭证，您可以执行以下任一操作：

- 创建与 Cassandra 用于身份验证和访问管理的传统用户名和密码类似的服务特定凭证。Amazon特定于服务的凭证与特定凭证关联Amazon Identity and Access Management(IAM) 用户，只能用于为其创建的服务。有关更多信息，请参阅 。将 IAM 与 Amazon Keyspaces 结合使用（针对 Apache Cassandra）(在 IAM 用户指南中)。
- 为了增强安全性，我们建议为所有用户和角色创建 IAM 访问密钥Amazon服务。使用适用于 Cassandra 客户端驱动程序的 Amazon Keyspaces Sigv4 身份验证插件，您可以使用 IAM 访问密钥而不是用户名和

密码对 Amazon Keyspaces 的呼叫进行身份验证。要了解 Amazon Keyspaces SigV4 插件如何启用，请执行以下操作：IAM 用户、角色和联合身份要在 Amazon Keyspaces API 请求中进行身份验证，请参阅Amazon签名版本 4 流程 (SigV4).

您可以从以下位置下载 SigV4 插件。

- Amazon 开发工具包: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.
- 转到：https://github.com/aws/aws-sigv4-auth-cassandra-gocql-driver-plugin.

有关演示如何使用 Sigv4 身份验证插件建立连接的代码示例，请参阅the section called “使用 Cassandra 客户端驱动程序” (p. 28).

主题

生成服务特定凭证

特定于服务的凭据类似于 Cassandra 用于身份验证和访问管理的传统用户名和密码。IAM 特定于服务的凭证允许 IAM 用户访问特定凭证Amazon服务。凭证不能用于访问其他凭证Amazon服务。它们与特定 IAM 用户关联，不能由其他 IAM 用户使用。

> **Important**
>
> 特定于服务的凭证与特定 IAM 用户关联，只能用于为其创建的服务。向 IAM 角色或联合身份授予访问您的所有权限Amazon资源，你应该创建 IAM 访问密钥Amazon身份验证并使用 Sigv4 身份验证插件。

使用以下过程之一生成服务特定凭证。

使用控制台生成服务特定凭证

使用控制台生成服务特定凭证

1. 通过以下网址登录 Amazon Web Services Management Console并打开 Amazon Identity and Access Management 控制台：https://console.amazonaws.cn/iam/home。
2. 在导航窗格中，选择导航窗格用户，然后选择之前创建的具有 Amazon Keyspaces 权限（已附加策略）的用户。
3. 选择 Security Credentials (安全凭证)。UNDERAmazon Keyspaces 凭证，选择生成凭证生成服务特定凭证。

   您的服务特定凭证现在可用。这是您下载或查看密码的唯一机会。以后您无法恢复它。不过，您可以随时重置密码。将用户和密码保存在安全的位置，因为随后您需要使用它们。

使用生成服务特定凭证Amazon CLI

要使用 Amazon CLI 生成服务特定凭证，请执行下方操作

生成服务特定凭证之前，您需要下载、安装和配置 Amazon Command Line Interface (Amazon CLI)：

1. 下载Amazon CLI在http://aws.amazon.com/cli.

   > **Note**
   >
   > Amazon CLI 在 Windows、macOS 或 Linux 上运行。

2. 按照 Amazon Command Line Interface 用户指南中安装 Amazon CLI 和配置 Amazon CLI 的说明进行操作。
3. 使用Amazon CLI，运行以下命令为用户生成服务特定凭证`alice`，以便她可以访问亚马逊Keyspaces。

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

输出如下所示。

```
{
    "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"
    }
}
```

在输出中，记下 `ServiceUserName` 和 `ServicePassword` 的值。将这些值保存在安全位置，因为稍后将需要它们。

> Important
>
> 这是您唯一能够看到 `ServicePassword` 的机会。

如何创建和配置AmazonAmazon Keyspaces 凭证

要以编程方式访问亚马逊Keyspaces，请使用Amazon CLI，AmazonSDK，或者使用 Cassandra 客户端驱动程序和 SigV4 插件，您需要一个拥有访问密钥的 IAM 用户或角色。当您以编程方式使用 Amazon 时，您需要提供您的 Amazon访问密钥，以便 Amazon 可以在编程调用中验证您的身份。您的访问密钥包含访问密钥 ID（例如 AKIAIOSFODNN7EXAMPLE）和秘密访问密钥（例如 wJalrXUtnFEMI/K7MDENG/AKIAIOSFODNN7EXAMPLEbPxRfiCYEXAMPLEKEY）。本主题说明该过程中的必填步骤。

主题

所需的证书Amazon CLI，AmazonSDK，或适用于 Cassandra 客户端驱动程序的 Amazon Keyspaces Sigv4 插件

对 IAM 用户或角色进行身份验证需要以下证书：

`AWS_ACCESS_KEY_ID`

指定与 IAM 用户或角色关联的 Amazon 访问密钥。

访问密钥aws_access_key_id需要通过编程方式连接到 Amazon Keyspaces。

AWS_SECRET_ACCESS_KEY

指定与访问密钥关联的私有密钥。这基本上是访问密钥的“密码”。

这些区域有：aws_secret_access_key需要通过编程方式连接到 Amazon Keyspaces。

AWS_SESSION_TOKEN— 可选

指定在使用您直接从 Amazon Security Token Service 操作中检索的临时安全凭证时需要的会话令牌值。有关更多信息，请参阅 the section called “使用临时证书连接到 Amazon Keyspaces” (p. 21)。

如果您正在与 IAM 用户连接，aws_session_token不是必填的。

创建 IAM 用户以编程方式访问您的亚马逊KeyspacesAmazon帐户

要获取以编程方式访问 Amazon Keyspaces 的证书Amazon CLI，AmazonSDK 或 SigV4 插件，您需要先创建 IAM 用户或角色。创建用户并配置该用户以编程方式访问 Amazon Keyspaces。

1. 在中创建用户Amazon Web Services Management Console，Amazon CLI，适用于 Windows 的 PowerShell，或者使用AmazonAPI 操作。如果你在中创建用户Amazon Web Services Management Console，然后自动创建证书。
2. 如果您以编程方式创建用户，则必须在附加步骤中为该用户创建访问密钥（访问密钥 ID 和秘密访问密钥）。
3. 授予用户访问亚马逊Keyspaces 的权限。

有关创建用户时需要的权限的信息，请参阅访问 IAM 资源所需的权限.

创建 IAM 用户（控制台）

您可以使用 Amazon Web Services Management Console 创建 IAM 用户。

要创建具有编程访问的 IAM 用户（控制台）

1. 登录Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Users（用户），然后选择 Add users（添加用户）。
3. 为新用户键入用户名。这是 Amazon 的登录名。

   Note

   用户名可以是一个最多由 64 个字母、数字和以下字符构成的组合：加号 (+)、等号 (=)、逗号 (,)、句点 (.)、at 符号 (@)、下划线 (_) 和连字符 (-)。账户中的名称必须唯一。名称不区分大小写。例如，您不能创建名为 TESTUSER 和 testuser 的两个用户。
4. Select访问密钥-编程访问为新用户创建访问密钥。您可以在转到决赛页面。

   选择 Next:。Permissions (下一步：权限)。
5. 在设置权限页面，选择直接附加现有策略向新用户分配权限。

   此选项显示以下列表Amazon您的账户中提供托管策略和客户托管策略。你可以输入keyspaces进入搜索字段，以仅显示与 Amazon Keyspaces。

   对于 Amazon Keyspaces，可用的托管策略是AmazonKeyspacesFullAccess和AmazonKeyspacesReadOnlyAccess. 有关各项策略的更多信息，请参阅the section called “Amazon 托管策略” (p. 171).

   出于测试目的和遵循连接教程，请选择AmazonKeyspacesReadOnlyAccess新用户的策略。注意：作为最佳实践，我们建议您遵循最低权限原则，创建自定义策略，限制对特定资源的访问并仅允许所需

操作。有关 IAM 策略的更多信息以及查看 Amazon Keyspaces 的示例策略，请参阅the section called “Amazon Keyspaces 基于身份的策略” (p. 163). 创建自定义策略后，将策略附加到组，然后使用户成为相应组的成员。

选择 Next:。标签。

6. 在添加标签（可选）页面你可以为用户添加标签，或者选择下一步：审核。
7. 在审核页面您可以看到您此时已做出的所有选择。如果您准备好继续，请选择创建用户.
8. 要查看用户的访问密钥（访问密钥 ID 和秘密访问密钥），请选择密码和访问密钥旁边的 Show (显示)。要保存访问密钥，请选择下载 .csv，然后将文件保存到安全位置。

   Important

   这是您查看或下载秘密访问密钥的唯一机会，他们才能使用SigV4插件的。将用户的新访问密钥 ID 和秘密访问密钥保存在安全的地方。完成此步骤后，您再也无法访问这些秘密访问密钥。

创建 IAM 用户 (Amazon CLI)

您可以使用 Amazon CLI 创建 IAM 用户。

创建具有编程访问权限的 IAM 用户 (Amazon CLI)

1. 使用以下内容创建用户Amazon CLI代码。
   - aws iam create-user
2. 向用户提供编程访问。这需要访问密钥，可以通过以下方式生成。
   - Amazon CLI: aws iam create-access-key
   - 适用于 Windows 的工具 PowerShell：New-IAMAccessKey
   - IAM API：CreateAccessKey

     Important

     这是您查看或下载秘密访问密钥的唯一机会，他们才能使用SigV4插件的。将用户的新访问密钥 ID 和秘密访问密钥保存在安全的地方。完成此步骤后，您再也无法访问这些秘密访问密钥。
3. 附上`AmazonKeyspacesReadOnlyAccess`策略适用于用户，此策略用于定义该用户的权限。注意：作为最佳实践，我们建议您通过将用户添加到一个组并向该组附加策略（而不是直接向用户附加策略）来管理用户权限。
   - Amazon CLI: aws iam attach-user-policy

为 IAM 用户创建新的访问密钥

如果您已有一个 IAM 用户，您可以随时创建新的访问密钥。有关密钥管理的更多信息，例如如何轮换访问密钥，请参阅管理 IAM 用户的访问密钥.

创建 IAM 用户的访问密钥（控制台）

1. 登录 Amazon Web Services Management Console，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择 Users（用户）。
3. 选择要为其创建访问密钥的用户的名称。
4. 在摘要用户页面，选择安全凭证选项卡。
5. 在 Access keys（访问密钥）部分中，选择 Create access key（创建访问密钥）。

要查看新访问密钥对，请选择 Show (显示)。您的凭证与下面类似：

- 访问密钥 ID：AKIAIOSFODNN7EXAMPLE
- 秘密访问密钥：wJalrXUtnFEM/K7MDENG/K7MDENG/bPxRfiCYEXAMPLEKE

Note

关闭此对话框后，您将无法再次访问该秘密访问密钥。

6. 要下载密钥对，请选择 Download .csv file（下载 .csv 文件）。将密钥存储在安全位置。
7. 下载 csv 格式文件之后，选择 Close（关闭）。

在创建访问密钥时，预设情况下，密钥对处于活动状态，并且您可以立即使用此密钥对。

如何管理 IAM 用户的访问密钥

作为最佳实践，我们建议您不要将访问密钥直接嵌入到代码中。这些区域有：Amazon开发工具包包包包包和Amazon使用 Command Line Tools，可以在已知位置放置访问密钥，这样就不必保留在代码中。在以下任一位置中放置访问密钥：

- 环境变量— 在多租户系统上，选择用户环境变量，而不是系统环境变量。
- CLI 凭证文件 – 在运行 `credentials` 命令时，将更新 `config` 和 `aws configure` 文件。这些区域有：`credentials`文件位于`~/.aws/credentials`在 Linux、macOS 或 Unix 上，或者在`C:\Users\USERNAME\.aws\credentials`在 Windows 上。该文件可以包含 `default` 配置文件和任何命名配置文件的凭证详细信息。
- CLI 配置文件 – 在运行 `credentials` 命令时，将更新 `config` 和 `aws configure` 文件。这些区域有：`config`文件位于`~/.aws/config`在 Linux、macOS 或 Unix 上，或者在`C:\Users\USERNAME\.aws\config`在 Windows 上。该文件包含原定设置配置文件和任何命名配置文件的配置设置。

将访问密钥存储为环境变量是访问密钥的先决条件the section called “适用于 Java 4.x 的身份验证插件” (p. 32). 客户端使用默认的凭证提供者链搜索证书，存储为环境变量的访问密钥优先于所有其他位置，例如配置文件。有关更多信息，请参阅 。配置设置和优先顺序.

下面的示例介绍您如何可以为默认用户配置环境变量。

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>
```

设置环境变量会更改使用的值，直到 Shell 会话结束或直到您将该变量设置为其他值。通过在 shell 的启动脚本中设置变量，可使变量在未来的会话中继续有效。

Windows Command Prompt

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

使用 `set` 设置环境变量会更改使用的值，直到当前命令提示符会话结束，或者直到您将该变量设置为其他值。使用 `setx` 设置环境变量会更改当前命令提示符会话和运行该命令后创建的所有命令提示符会话中使用的值。它不 影响在运行该命令时已经运行的其他命令 shell。

PowerShell

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

如果你在 PowerShell 提示符如前面的示例所示，它仅保存当前会话持续时间的值。使环境变量设置在所有环境中保持不变 PowerShell 和命令提示符会话，使用系统中的应用程序控制面板. 或者，您可以为将 future 全部设置该变量 PowerShell 通过将其添加至您的会话 PowerShell 配置文件。请参阅PowerShell 文档了解有关存储环境变量或跨会话保存它们的更多信息。

使用 IAM 角色和 SigV4 插件的 Amazon Keyspaces。

为了增强安全性，您可以使用临时凭证使用 Sigv4 插件进行身份验证。在许多情况下，您并不需要永不过期的长期访问密钥（如 IAM 用户访问密钥）。相反，您可以创建一个 IAM 角色并生成临时安全凭证。临时安全证书包括访问密钥 ID 和秘密访问密钥，以及一个指示证书何时到期的安全令牌。要了解有关如何使用 IAM 角色而不是长期访问密钥的更多信息，请参阅切换到 IAM 角色 (AmazonAPI).

要开始使用临时凭证，您首先需要创建一个 IAM 角色。

创建一个 IAM 角色，该角色授予对 Amazon Keyspaces 的只读访问权限

1. 登录Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色，那么创建角色.
3. 在创建角色页面，下方选择受信任实体的类型，选择Amazon服务. 下面选择用例，选择Amazon EC2，然后选择下一页.
4. 在添加权限页面，下方权限策略，选择亚马逊Keyspaces 只读访问权限从策略列表中，然后选择下一页.
5. 在命名、审阅和创建页面上，为角色输入一个名称，然后查看选择可信实体和添加权限部分。您还可以在此页面上为角色添加可选标签。操作完成后，选择创建角色. 请记住此名称，因为您启动 Amazon EC2 实例时将需要它。

要在代码中使用临时安全证书，您可以通过编程方式调用Amazon Security Token ServiceAPI`AssumeRole`并从上一步中创建的 IAM 角色中提取生成的凭证和会话令牌。然后，您可以使用这些值作为对 Amazon 的后续调用的凭证。以下示例显示了如何使用临时安全证书的伪代码：

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

有关使用 Python 驱动程序实现临时证书以访问 Amazon Keyspaces 的示例，请参阅??? (p. 36).

有关如何调用 `AssumeRole`、`GetFederationToken` 和其他 API 操作的详细信息，请参阅 Amazon Security Token Service API 参考。有关从结果中获取临时安全凭证和会话令牌的信息，请参阅所用开发工具包的文档。可以在主 Amazon 文档页上的 SDKs and Toolkits（开发工具包和工具箱）部分中找到所有 Amazon SDK 的文档。

Amazon Keyspaces

主题

中国区域终端节点

以下 Amazon Keyspaces 终端节点在Amazon中国区域。

区域名称	区域	端点	协议
中国（北京）	cn-north-1	cassandra.cn-north-1.amazonaws.com.cn	TLS
中国（宁夏）	cn-northwest-1	cassandra.cn-northwest-1.amazonaws.com.cn	TLS

端口

下表显示了要用于该端口的身份验证机制。亚马逊Keyspaces 使用 TLS 对 Starfield CA 对服务器进行身份验证。

端口	身份验证机制
9142	TLS

使用cqlsh连接到亚马逊 Keyspaces

以下部分介绍如何将cqlsh连接到 Amazon Keyspaces（针对 Apache Cassandra）。

有关的信息cqlsh，请参阅cqlsh: CQL 外壳.

主题

- 安装和使用cqlsh连接到 Amazon Keyspaces（针对 Apache Cassandra） (p. 22)

安装和使用cqlsh连接到 Amazon Keyspaces（针对 Apache Cassandra）

要安装和使用cqlsh，您必须执行以下操作：

1. 安装 Python 2.7 (p. 22)。
2. 安装cqlsh客户 (p. 23).
3. Encryptcqlsh使用 TLS 连接 (p. 23).

> **Note**
>
> 为了使cqlsh连接到 Amazon Keyspaces 以进行功能测试、轻操作和迁移，您可以使用预配置的 Docker 容器，其中包括针对 Amazon Keyspaces 优化的所有先决条件和配置设置。该工具包可在GitHub网站.

安装 Python 2.7

要确定计算机上是否安装了 Python 以及安装了哪个版本，请运行以下操作。

```
python --version
```

如果您安装了 Python 2.7，则输出类似于以下内容。

```
Python 2.7.16
```

如果您需要安装 Python 2.7，请按照中的说明进行操作。Python 下载.

安装和配置 CQL 客户端

cqlsh与 Apache Cassandra 捆绑在一起。要得到它，请按照中的说明安装 Apache Cassandra。下载和安装 Apache Cassandra. 亚马逊 Keyspaces 支持与 Apache Cassandra 3.11.2 兼容的驱动程序和客户端。当前推荐的版本cqlsh可以从下载阿帕奇.

安装 Cassandra 之后，请验证cqlsh通过运行以下命令安装。

```
cqlsh --version
```

输出应该类似于以下内容。

```
cqlsh 5.0.1
```

如果你使用的是 Windows，请替换cqlsh和cqlsh.bat. 例如，要在 Windows 中检查 cqlsh 版本，请运行以下命令。

```
cqlsh.bat --version
```

下载配置文件的模板cqlshrc针对 Amazon Keyspaces 进行优化Github. 保存下载的cqlshrc_template文件为cqlshrc到卡桑德拉目录。

```
${HOME}/.cassandra/cqlshrc
```

加密cqlsh使用 TLS 连接

Amazon Keyspaces 仅接受使用传输层安全性 (TLS) 的安全连接。

在使用 SSL/TLS 连接之前，您必须执行以下操作：

1. 使用以下命令下载 Starfield 数字证书并保存sf-class2-root.crt本地或在您的主目录中。

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

   > **Note**
   >
   > 您还可以使用亚马逊数字证书连接到亚马逊 Keyspaces，如果您的客户成功连接到亚马逊 Keyspaces，则可以继续这样做。Starfield 证书为使用旧证书颁发机构的客户提供了额外的向后兼容性。

2. 使用以下命令 Connect 到 Amazon Keyspaces。

   > **Important**
   >
   > ServiceUserName 和 ServicePassword 应与您按照生成服务特定凭证 (p. 16)中的步骤生成服务特定凭证时获得的用户名和密码相匹配。
   > 您还可以通过以下方式管理亚马逊 Keyspaces cqlsh 访问Amazon通过使用 IAM 用户和角色Amazoncqlsh 的身份验证插件扩展。要了解更多信息，请参阅Github 上的 Amazon Keyspaces（针对 Apache Cassandra）开发者工具包.

```
cqlsh host 9142 -u ServiceUserName -p ServicePassword --ssl
```

请注意，9142 是安全端口。

以下是示例。

```
cqlsh cassandra.us-east-2.amazonaws.com 9142 -u "alice-at-111122223333" -
p "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY" --ssl
```

连接后，输出将类似于以下内容。目前支持的 Apache Cassandra 版本为 3.11.2。

```
Connected to Amazon Keyspaces at cassandra.us-east-2.amazonaws.com:9142.
[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
alice-at-111122223333@cqlsh>
```

更新现有的配置文件cqlsh连接

例如，如果要编辑现有配置文件以支持 TLS 连接，请在 Cassandra 主目录中打开配置文件，例如${HOME}/.cassandra/cqlshrc添加以下行。

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

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

您可以配置 cqlshCOPY设置以确保cqlsh保持在亚马逊 Keyspaces 内the section called “CQL 查询吞吐量调整” (p. 7)准则。

修改默认值COPY FROM配置文件中的选项${HOME}/.cassandra/cqlshrc添加以下行。

```
[copy-from]
CHUNKSIZE=50
```

CHUNKSIZE 的此设置可以很好地开始使用新创建的表，并且应该进行更改以支持更大的工作负载。有关如何优化的更多信息cqlsh COPY亚马逊 Keyspaces 的配置设置，请参阅the section called “第 4 步：配置cqlsh COPY FROM设置” (p. 62)在数据迁移教程中。

使用 Amazon CLI

您可以使用 Amazon Command Line Interface (Amazon CLI) 从命令行管理多个 Amazon 服务并通过脚本自动执行这些服务。使用亚马逊 Keyspaces，您可以使用Amazon CLI用于数据定义语言 (DDDDDDL) 操作，如创建表。此外，您可以将基础设施用作代码 (iAC) 服务和工具，例如Amazon CloudFormation和 Terraform。

在使用Amazon CLI使用 Amazon Keyspaces，您必须获取访问密钥 ID 和秘密访问密钥。有关更多信息，请参阅 the section called “IAM 凭证Amazon身份验证” (p. 17)。

有关对 Amazon Keyspaces 可用的所有命令的完整列表，请参阅Amazon CLI，请参阅Amazon CLI命令参考.

主题

下载和配置 Amazon CLI

这些区域有：Amazon CLI在以下位置可用http://www.amazonaws.cn/cli. 它在 Windows、macOS 或 Linux 上运行。下载完Amazon CLI，可执行以下步骤安装和配置：

1. 转至Amazon Command Line Interface用户指南
2. 按照说明进行操作安装Amazon CLI和配置Amazon CLI

使用Amazon CLI使用 Amazon Keyspaces

命令行格式包含 Amazon Keyspaces 操作名称，后跟该操作的参数。Amazon CLI 支持参数值的速记语法以及 JSON。以下亚马逊 Keyspaces 示例使用Amazon CLI速记语法。有关更多信息，请参阅 。将速记语法与结合使用AmazonCLI.

以下命令创建名称的键空间：目录.

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

该命令将返回输出中的资源 Amazon 资源名称 (ARN)。

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

确认密钥空间目录存在，您可使用以下命令。

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

该命令的输出将返回以下值。

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

以下命令创建名称的表：book_奖项/奖项. 表的分区键包含以下列。`year`和`award`群集密钥由列组成`category`和`rank`，两个聚类列都使用升序排序顺序。（为便于阅读，本部分中的长命令分行显示。）

```
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}]'
```

此命令将生成以下输出。

```
{
```

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

要确认表的元数据和属性，可以使用以下命令。

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

此命令将返回以下输出。

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

```
                "orderBy": "ASC"
            }
        ],
        "staticColumns": []
    },
    "capacitySpecification": {
        "throughputMode": "PAY_PER_REQUEST",
        "lastUpdateToPayPerRequestTimestamp": 1645564368.628
    },
    "encryptionSpecification": {
        "type": "AWS_OWNED_KMS_KEY"
    },
    "pointInTimeRecovery": {
        "status": "DISABLED"
    },
    "ttl": {
        "status": "ENABLED"
    },
    "defaultTimeToLive": 0,
    "comment": {
        "message": ""
    }
}
```

创建具有复杂模式的表时，从 JSON 文件加载表的架构定义可能会有所帮助。下面是一个示例配置文件。从下载模式定义示例 JSON 文件schema_definition.zip然后提取schema_definition.json，记下文件的路径。在此示例中，架构定义 JSON 文件位于当前目录中。有关不同文件路径选项，请参阅如何从文件加载参数.

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

以下几个示例介绍如何创建名称的简单表。myTable还有其他选项。请注意，这些命令分为单独的行以提高可读性。此命令介绍如何创建表并执行以下操作：

- 设置表的容量模式
- 为表启用时间点恢复
- 将表的默认生存时间 (TTL) 值设置为 1 年
- 为表格添加两个标签

```
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'
```

此示例说明如何创建一个新表，该表使用客户管理的密钥进行加密，并启用 TTL 以允许您为列和行设置过期日期。要运行此示例，您必须为客户管理的资源 ARN 替换资源 ARNAmazon KMS使用您自己的密钥密钥密钥，并确保亚马逊 Keyspaces 有权访问它。

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

```
            --ttl 'status=ENABLED'
```

使用 API

您可以使用Amazon开发工具包和Amazon Command Line Interface(Amazon CLI) 以交互方式与亚马逊 Keyspaces 协作。您可以使用 API 进行数据语言定义 (DDL) 操作，例如创建密钥空间或表。此外，您可以将基础设施用作代码 (iAC) 服务和工具，例如Amazon CloudFormation和 Terraform。

您必须先满足Amazon CLI对于 Amazon Keyspaces，您必须先获取访问密钥 ID 和秘密访问密钥。有关更多信息，请参阅 the section called “IAM 凭证Amazon身份验证” (p. 17)。

有关 API 中对 Amazon Keyspaces 可用的所有操作的完整列表，请参阅Amazon Keyspaces API 参考.

使用 Cassandra 客户端驱动程序以编程方式访问亚马逊Keyspaces

您可以使用许多第三方开源 Cassandra 驱动程序连接到亚马逊 Keyspaces。亚马逊 Keyspaces 与支持 Apache Cassandra 版本 3.11.2 的 Cassandra 驱动程序兼容。有关 Cassandra 驱动程序的更多信息，请参阅 Apache Cassandra 客户端驱动程序。

> Note
>
> 为了帮助您入门，您可以查看并下载 end-to-end 使用常用驱动程序与 Amazon Keyspaces 建立连接的代码示例。请参阅Amazon Keyspaces sp上 GitHub.

本章中的教程包括一个简单的 CQL 查询，用于确认已成功建立与 Amazon Keyspaces 的连接。要了解在连接到 Amazon Keyspaces 终端节点后如何使用密钥空间和表，请参阅CQL 语言参考 (p. 203).

主题

使用 Cassandra Java 客户端驱动程序以编程方式访问亚马逊 Keyspaces

本节说明了如何使用 Java 客户端驱动程序连接到 Amazon Keyspaces。要向用户和应用程序提供对 Amazon Keyspaces 资源进行编程访问的凭证，您可以执行以下任一操作：

- 创建与特定服务关联的服务特定凭证Amazon Identity and Access Management(IAM) 用户。
- 为了增强安全性，我们建议为所有用户或角色创建 IAM 访问密钥Amazon服务。适用于 Cassandra 客户端驱动程序的 Amazon Keyspaces Sigv4 身份验证插件使您能够使用 IAM 访问密钥而不是用户名和密码对对 Amazon Keyspaces 的调用进行身份验证。有关更多信息，请参阅 the section called “IAM 凭证Amazon身份验证” (p. 17)。

> Note
>
> 有关如何在 Spring Boot 中使用亚马逊Keyspaces 的示例，请参阅https://github.com/aws-samples/amazon-keyspaces-spring-app-example.

主题

开始前的准备工作

要连接到 Amazon Keyspaces，您需要完成以下任务。

1. Amazon Keyspaces 需要使用传输层安全性 (TLS) 来帮助保护与客户端的连接。

   a. 使用以下命令下载 Starfield 数字证书并保存`sf-class2-root.crt`本地或您的主目录中。

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

   > **Note**
   >
   > 您也可以使用亚马逊数字证书连接到亚马逊Keyspaces，如果您的客户端成功连接到亚马逊Keyspaces，则可以继续这样做。Starfield 证书为使用旧证书颁发机构的客户提供了额外的向后兼容性。

   b. 将 Starfield 数字证书转换为 TrustStore 文件。

   ```
   openssl x509 -outform der -in sf-class2-root.crt -out temp_file.der
   keytool -import -alias cassandra -keystore cassandra_truststore.jks -file
    temp_file.der
   ```

   在此步骤中，您需要为密钥库创建密码并信任此证书。交互式命令看起来像这样。

   ```
   Enter keystore password:
   Re-enter new password:
   Owner: OU=Starfield Class 2 Certification Authority, O="Starfield Technologies,
    Inc.", C=US
   Issuer: OU=Starfield Class 2 Certification Authority, O="Starfield Technologies,
    Inc.", C=US
   Serial number: 0
   Valid from: Tue Jun 29 17:39:16 UTC 2004 until: Thu Jun 29 17:39:16 UTC 2034
   Certificate fingerprints:
     MD5:  32:4A:4B:BB:C8:63:69:9B:BE:74:9A:C6:DD:1D:46:24
     SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
     SHA256:
    14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB:5F:B6:5
   Signature algorithm name: SHA1withRSA
   Subject Public Key Algorithm: 2048-bit RSA key
   Version: 3
   Extensions:
   #1: ObjectId: 2.5.29.35 Criticality=false
   AuthorityKeyIdentifier [
   KeyIdentifier [
   0000: BF 5F B7 D1 CE DD 1F 86   F4 5B 55 AC DC D7 10 C2  ._.......[U.....
   0010: 0E A9 88 E7                                        ....
   ]
   [OU=Starfield Class 2 Certification Authority, O="Starfield Technologies, Inc.",
    C=US]
   SerialNumber: [    00]
   ```

```
]
#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
  CA:true
  PathLen:2147483647
]
#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: BF 5F B7 D1 CE DD 1F 86   F4 5B 55 AC DC D7 10 C2  ._.......[U.....
0010: 0E A9 88 E7                                        ....
]
]
Trust this certificate? [no]:  y
```

2. 将 trustStore 文件附加到 JVM 参数中：

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

Step-by-step 使用连接到 Amazon Keyspaces DataStax 使用特定服务凭据的 Apache Cassandra 的 Java 驱动程序

以下 step-by-step 教程将指导您使用服务特定证书使用适用于 Cassandra 的 Java 驱动程序连接到 Amazon Keyspaces。具体来说，你将使用 4.0 版本的 DataStax Apache Cassandra 的 Java 驱动程序。

主题

第 1 步：先决条件

要完成本教程，您需要生成服务特定凭证并添加 DataStax 你的 Java 项目的 Apache Cassandra 的 Java 驱动程序。

- 完成中的步骤，为您的 Amazon Keyspaces IAM 用户生成服务专用证书the section called “服务特定凭证” (p. 16). 如果您更喜欢使用 IAM 访问密钥进行身份验证，请参阅the section called “适用于 Java 4.x 的身份验证插件” (p. 32).
- 添加 DataStax 你的 Java 项目的 Apache Cassandra 的 Java 驱动程序。确保你使用的是支持 Apache Cassandra 3.11.2 的驱动程序版本。有关更多信息，请参阅 。DataStax Apache Cassandra 文档的 Java 驱动程序.

第 2 步：配置驱动程序

您可以指定设置 DataStax Java Cassandra 驱动程序通过为您的应用程序创建配置文件来实现。此配置文件会覆盖默认设置，并告知驱动程序使用端口 9142 连接到 Amazon Keyspaces 服务终端节点。有关可用服务终端节点的列表the section called “服务端点” (p. 21).

创建配置文件并将该文件保存在应用程序的资源文件夹中，例如，`src/main/resources/application.conf`. Open（打开）`application.conf`并添加以下配置设置。

1. 身份验证提供商— 使用以下命令创建身份验证提供商`PlainTextAuthProvider`类。*`ServiceUserName`*和*`ServicePassword`*应与您在按照中的步骤生成特定服务凭证时获得的用户名和密码相匹配生成服务特定凭证 (p. 16).

Note

您可以使用身份验证插件来使用短期证书 DataStax 适用于 Apache Cassandra 的 Java 驱动程序，而不是驱动程序配置文件中的硬编码凭据。要了解更多信息，the section called “适用于 Java 4.x 的身份验证插件” (p. 32).

2. 本地数据中心— 设置值`local-datacenter`到你要连接的区域。例如，如果应用程序正在连接到`cassandra.us-east-2.amazonaws.com`，然后将本地数据中心设置为`us-east-2`. 全部可用 Amazon Web Services 区域，请参阅??? (p. 21).
3. SSLSLSL— 初始化 SSLEngineFactory 通过在配置文件中添加一个部分，用一行指定类`class = DefaultSslEngineFactory`. 提供您之前创建的 TrustStore 文件的路径和密码。

```
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"
    }

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

Note

您也可以直接在应用程序代码中添加 TrustStore 路径，也可以将 TrustStore 的路径添加到 JVM 参数中，而不是在配置文件中添加 TrustStore 的路径。

第 3 步：运行示例应用程序

此代码示例显示了一个简单的命令行应用程序，该应用程序使用我们之前创建的配置文件创建了与 Amazon Keyspaces 的连接池。它通过运行一个简单的查询来确认连接已建立。

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

public class App
{

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

            ResultSet rs = session.execute("select * from system_schema.keyspaces");
```

```
            Row row = rs.one();
            System.out.println(row.getString("keyspace_name"));
        }
    }
}
```

Note

使用try阻塞以建立连接，确保连接始终处于关闭状态。如果你不使用try阻止，记得关闭连接以避免资源泄漏。

Step-by-step 使用 4.x 连接到 Amazon Keyspaces DataStax 适用于 Apache Cassandra 的 Java 驱动程序和 SigV4 身份验证插件

以下部分介绍如何使用开源 4.x 的 SigV4 身份验证插件 DataStax for Apache Cassandra 访问亚马逊 Keyspaces spache Cassandra。该插件可从以下网址获得GitHub知识库.

SigV4 身份验证插件允许您在连接到 Amazon Keyspaces 时使用用户或角色的 IAM 证书。该插件不需要用户名和密码，而是使用访问密钥签署 API 请求。有关更多信息，请参阅 the section called “IAM 凭证Amazon身份验证” (p. 17)。

第 1 步：先决条件

要完成以下任务。

- 如果您尚未执行此操作，请按照以下步骤为您的 IAM 用户或角色创建凭证the section called “IAM 凭证Amazon身份验证” (p. 17). 本教程假设访问密钥存储为环境变量。有关更多信息，请参阅 the section called “如何管理访问密钥” (p. 20)。
- 添加 DataStax 你的 Java 项目的 Apache Cassandra 的 Java 驱动程序。确保你使用的是支持 Apache Cassandra 3.11.2 的驱动程序版本。有关更多信息，请参阅 。DataStax Apache Cassandra 的 Java 驱动程序文档.
- 将身份验证插件添加到您的应用程序中。身份验证插件支持 4.x 版 DataStax Apache Cassandra 的 Java 驱动程序。如果您使用的是 Apache Maven 或可以使用 Maven 依赖关系的构建系统，请将以下依赖关系添加到您的 `pom.xml` 文件中。

  Important

  用最新版本替换插件的版本，如下所示GitHub 知识库.

```
<dependency>
        <groupId>software.aws.mcs</groupId>
        <artifactId>aws-sigv4-auth-cassandra-java-driver-plugin</artifactId>
        <version>4.0.5</version>
</dependency>
```

第 2 步：配置驱动程序

您可以指定设置 DataStax Java Cassandra 驱动程序通过为您的应用程序创建配置文件来实现。此配置文件会覆盖默认设置，并告知驱动程序使用端口 9142 连接到 Amazon Keyspaces 服务终端节点。有关可用服务终端节点的列表the section called “服务端点” (p. 21).

创建配置文件并将该文件保存在应用程序的资源文件夹中，例如，`src/main/resources/application.conf`. Open（打开）`application.conf`并添加以下配置设置。

1. 身份验证提供商— 设置`advanced.auth-provider.class`到一个新的实例`software.aws.mcs.auth.SigV4AuthProvider`. Sigv4AuthProvider 是插件提供的用于执行 Sigv4 身份验证的身份验证处理程序。

2. 本地数据中心— 设置值`local-datacenter`到你要连接的区域。例如，如果应用程序正在连接到`cassandra.us-east-2.amazonaws.com`，然后将本地数据中心设置为`us-east-2`. 全部可用 Amazon Web Services 区域，请参阅??? (p. 21).
3. SSLSLSL— 初始化 SSLEngineFactory 通过在配置文件中添加一个部分，用一行指定类`class = DefaultSslEngineFactory`. 提供您之前创建的 TrustStore 文件的路径和密码。

```
datastax-java-driver {
        basic.contact-points = ["cassandra.us-east-2.amazonaws.com:9142"]
        basic.load-balancing-policy {
            class = DefaultLoadBalancingPolicy
            local-datacenter = us-east-2
        }
        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
            }
        }
    }
```

Note

您也可以直接在应用程序代码中添加 TrustStore 路径，也可以将 TrustStore 的路径添加到 JVM 参数中，而不是在配置文件中添加 TrustStore 的路径。

第 3 步：运行应用程序

此代码示例显示了一个简单的命令行应用程序，该应用程序使用我们之前创建的配置文件创建了与 Amazon Keyspaces 的连接池。它通过运行一个简单的查询来确认连接已建立。

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

public class App
{

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

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

Note

使用try阻塞以建立连接，确保连接始终处于关闭状态。如果你不使用try阻止，记得关闭连接以避免资源泄漏。

使用 3.x Connect 亚马逊Keyspaces DataStax适用于 Apache Cassandra 的 Java 驱动程序和 SigV4 身份验证插件

以下部分介绍如何使用适用于 3.x 开源的 SigV4 身份验证插件 DataStax Apache Cassandra 访问亚马逊 Keyspaces 的 Java 驱动程序。该插件可从以下网址获得GitHub 知识库.

SigV4 身份验证插件允许您在连接到 Amazon Keyspaces 时使用用户和角色的 IAM 证书。该插件不需要用户名和密码，而是使用访问密钥签署 API 请求。有关更多信息，请参阅 the section called “IAM 凭证Amazon身份验证” (p. 17)。

第 1 步：先决条件

要运行此代码示例，您首先需要完成以下任务。

- 按照中的步骤为您的 IAM 用户或角色创建证书the section called “IAM 凭证Amazon身份验证” (p. 17). 本教程假设访问密钥存储为环境变量。有关更多信息，请参阅 the section called “如何管理访问密钥” (p. 20)。
- 按中的步骤操作the section called “开始前的准备工作” (p. 29)要下载 Starfield 数字证书，请将其转换为 TrustStore 文件，然后将 JVM 参数中的 TrustStore 文件附加到您的应用程序中。
- 添加 DataStax 你的 Java 项目的 Apache Cassandra 的 Java 驱动程序。确保你使用的是支持 Apache Cassandra 3.11.2 的驱动程序版本。有关更多信息，请参阅 。DataStax Apache Cassandra 的 Java 驱动程序文档.
- 将身份验证插件添加到您的应用程序中。身份验证插件支持 3.x 版的 DataStax Apache Cassandra 的 Java 驱动程序。如果您使用的是 Apache Maven 或可以使用 Maven 依赖关系的构建系统，请将以下依赖关系添加到您的 `pom.xml` 文件中。用最新版本替换插件的版本，如下所示GitHub 知识库.

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

第 2 步：运行应用程序

此代码示例显示了一个简单的命令行应用程序，该应用程序创建了与 Amazon Keyspaces 的连接池。它通过运行一个简单的查询来确认连接已建立。

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

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

public class App
{

    public static void main( String[] args )
    {
        String endPoint = "cassandra.us-east-2.amazonaws.com";
        int portNumber = 9142;
```

```
        Session session = Cluster.builder()
                                 .addContactPoint(endPoint)
                                 .withPort(portNumber)
                                 .withAuthProvider(new SigV4AuthProvider("us-east-2"))
                                 .withSSL()
                                 .build()
                                 .connect();

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

使用说明：

有关可用终端节点的列表，请参阅the section called “服务端点” (p. 21)。

使用 Cassandra Python 客户端驱动程序以编程方式访问亚马逊 Keyspaces

在本节中，我们将向您介绍如何使用 Python 客户端驱动程序连接到 Amazon Keyspaces。要向用户和应用程序提供对 Amazon Keyspaces 资源进行编程访问的凭证，您可以执行以下任一操作：

- 创建与特定服务关联的服务特定凭证Amazon Identity and Access Management(IAM) 用户。
- 为了增强安全性，我们建议为所有用户或角色创建 IAM 访问密钥Amazon服务。适用于 Cassandra 客户端驱动程序的 Amazon Keyspaces Sigv4 身份验证插件使您能够使用 IAM 访问密钥而不是用户名和密码对对 Amazon Keyspaces 的调用进行身份验证。有关更多信息，请参阅 the section called “IAM 凭证Amazon身份验证” (p. 17)。

主题

开始前的准备工作

您需要完成以下任务才能开始执行以下任务。

Amazon Keyspaces 需要使用传输层安全性 (TLS) 来帮助保护与客户端的连接。要使用 TLS 连接到亚马逊 Keyspaces，您需要下载亚马逊数字证书并将 Python 驱动程序配置为使用 TLS。

使用以下命令下载 Starfield 数字证书并保存`sf-class2-root.crt`本地或您的主目录中。

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

> **Note**
>
> 您也可以使用亚马逊数字证书连接到亚马逊Keyspaces，如果您的客户端成功连接到亚马逊 Keyspaces，则可以继续这样做。Starfield 证书为使用旧证书颁发机构的客户提供了额外的向后兼容性。

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

使用 Apache Cassandra 的 Python 驱动程序和服务特定证书Connect 亚马逊 Keyspaces

以下代码示例向您展示如何使用 Python 客户端驱动程序和服务特定证书连接到 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/sf-class2-root.crt')
ssl_context.verify_mode = CERT_REQUIRED
auth_provider = PlainTextAuthProvider(username='ServiceUserName',
 password='ServicePassword')
cluster = Cluster(['cassandra.us-east-2.amazonaws.com'], ssl_context=ssl_context,
 auth_provider=auth_provider, port=9142)
session = cluster.connect()
r = session.execute('select * from system_schema.keyspaces')
print(r.current_rows)
```

使用说明：

1. Replace（替换）"*path_to_file*/sf-class2-root.crt"在第一步中保存了证书的路径。
2. 确保*ServiceUserName*和*ServicePassword*按照以下步骤与您在生成服务特定凭证时获得的用户名和密码相匹配生成服务特定凭证 (p. 16).
3. 有关可用终端节点的列表，请参阅the section called “服务端点” (p. 21)。

使用以下命令Connect 亚马逊Keyspaces DataStax适用于 Apache Cassandra 的 Python 驱动程序和 Sigv4 身份验证插件

以下部分介绍如何使用开源的 SigV4 身份验证插件 DataStax for Apache Cassandra 访问亚马逊 Keyspaces spache Cassandra。

如果您尚未执行此操作，请按照步骤the section called “IAM 凭证Amazon身份验证” (p. 17). 本教程使用临时证书，这需要 IAM 角色。有关临时凭证的临时凭证the section called “使用临时证书连接到 Amazon Keyspaces” (p. 21).

然后，将 Python Sigv4 身份验证插件从中添加到您的环境中GitHub 知识库.

```
pip install cassandra-sigv4
```

以下代码说明如何使用开源连接到 Amazon Keyspaces DataStax 适用于 Cassandra 的 Python 驱动程序和 SigV4 身份验证插件。该插件取决于AmazonSDK for Python (Boto3)。它使用`boto3.session`获取临时凭证。

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

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

# use this if you want to use Boto to set the session parameters.
boto_session = boto3.Session(aws_access_key_id="AKIAIOSFODNN7EXAMPLE",
```

```
                                aws_secret_access_key="wJalrXUtnFEMI/K7MDENG/
bPxRfiCYEXAMPLEKEY",
                                aws_session_token="AQoDYXdzEJr...<remainder of token>",
                                region_name="us-east-2")
auth_provider = SigV4AuthProvider(boto_session)

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

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

使用说明：

1. Replace（替换）"*path_to_file*/sf-class2-root.crt"在第一步中保存了证书的路径。
2. 确保*aws_access_key_id*,*aws_secret_access_key*，还有*aws_session_token*匹配Access Key,Secret Access Key，以及Session Token你使用以下方法获得的boto3.session. 有关更多信息，请参阅 。凭证在Amazon SDK for Python (Boto3).
3. 有关可用终端节点的列表，请参阅the section called “服务端点” (p. 21)。

使用 Cassandra Node.js 客户端驱动程序以编程方式访问亚马逊 Keyspaces

本节说明了如何使用 Node.js 客户端驱动程序连接到 Amazon Keyspaces。要向用户和应用程序提供对 Amazon Keyspaces 资源进行编程访问的凭证，您可以执行以下任一操作：

- 创建与特定服务关联的服务特定凭证Amazon Identity and Access Management(IAM) 用户。
- 为了增强安全性，我们建议为所有用户或角色创建 IAM 访问密钥Amazon服务。适用于 Cassandra 客户端驱动程序的 Amazon Keyspaces Sigv4 身份验证插件使您能够使用 IAM 访问密钥而不是用户名和密码对对 Amazon Keyspaces 的调用进行身份验证。有关更多信息，请参阅 the section called “IAM 凭证Amazon身份验证” (p. 17)。

主题

开始前的准备工作

您需要完成以下任务才能开始执行以下任务。

Amazon Keyspaces 需要使用传输层安全性 (TLS) 来帮助保护与客户端的连接。要使用 TLS 连接到亚马逊 Keyspaces，您需要下载亚马逊数字证书并将 Python 驱动程序配置为使用 TLS。

使用以下命令下载 Starfield 数字证书并保存sf-class2-root.crt本地或您的主目录中。

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

Note

您也可以使用亚马逊数字证书连接到亚马逊Keyspaces，如果您的客户端成功连接到亚马逊Keyspaces，则可以继续这样做。Starfield 证书为使用旧证书颁发机构的客户提供了额外的向后兼容性。

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

使用 Node.js Connect 亚马逊Keyspaces DataStax Apache Cassandra 的驱动程序和特定服务凭证

将您的驱动程序配置为使用 Starfield 数字证书进行 TLS，并使用特定于服务的凭据进行身份验证。例如：

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

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

使用说明：

1. Replace（替换）"*path_to_file*/sf-class2-root.crt"在第一步中保存了证书的路径。
2. 确保*ServiceUserName*和*ServicePassword*按照以下步骤与您在生成服务特定凭证时获得的用户名和密码相匹配生成服务特定凭证 (p. 16).
3. 有关可用终端节点的列表，请参阅the section called “服务端点” (p. 21)。

使用以下命令Connect 亚马逊Keyspaces DataStax 适用于 Apache Cassandra 和 Sigv4 身份验证插件的 Node.js 驱动程序

以下部分介绍如何使用开源的 SigV4 身份验证插件 DataStax Node.js for Apache Cassandra 访问亚马逊Keyspaces spache Cassandra。

如果您尚未执行此操作，请按照以下步骤为您的 IAM 用户或角色创建凭证the section called “IAM 凭证Amazon身份验证” (p. 17).

将 Node.js Sigv4 身份验证插件从中添加到您的应用程序中GitHub 知识库. 该插件支持 4.x 版DataStax适用于 Cassandra 的 Node.js 驱动程序取决于Amazon适用于 Node.js 的开发工具包。它使用`AWSCredentialsProvider`获取证书。

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

此代码说明如何设置区域特定实例SigV4AuthProvider作为身份验证提供商。

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

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

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


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


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

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

使用说明：

1. Replace（替换）"*path_to_file*/sf-class2-root.crt"在第一步中保存了证书的路径。
2. 确保*accessKeyId*和*secretAccessKey*匹配您使用获得的访问密钥和私有访问密钥AWSCredentialsProvider. 有关更多信息，请参阅 。在 Node.js 中设置证书在Amazon适用于 GOR JavaScript 在 Node.js 中.
3. 要在代码之外存储访问密钥，请参阅以下最佳实践the section called “如何管理访问密钥” (p. 20).
4. 有关可用终端节点的列表，请参阅the section called “服务端点” (p. 21)。

使用 Cassandra .NET Core 客户端驱动程序以编程方式访问亚马逊 Keyspaces

本节说明了如何使用.NET Core 客户端驱动程序连接到 Amazon Keyspaces。设置步骤将因您的环境和操作系统而异，您可能需要相应地对其进行修改。Amazon Keyspaces 需要使用传输层安全性 (TLS) 来帮助保护与客户端的连接。要使用 TLS 连接到 Amazon Keyspaces，您需要下载 Starfield 数字证书并将驱动程序配置为使用 TLS。

1. 下载 Starfield 证书并将其保存到本地目录中，记下路径。以下是使用的示例 PowerShell.

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

2. 安装 CassDracSharpDriver 通过 nuget，使用 nuget 控制台。

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

3. 以下示例使用 .NET Core C# 控制台项目连接到 &MCS; 并运行查询。

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

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

        static void Main(string[] args)
        {
            X509Certificate2Collection certCollection = new X509Certificate2Collection();
            X509Certificate2 amazoncert = new X509Certificate2(@"path_to_file\sf-class2-
root.crt");
            var userName = "ServiceUserName";
            var pwd = "ServicePassword";
            certCollection.Add(amazoncert);

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

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

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

使用说明：

a. Replace（替换）"*path_to_file*/sf-class2-root.crt"在第一步中保存了证书的路径。

b. 确保*ServiceUserName*和*ServicePassword*按照以下步骤与您在生成服务特定凭证时获得的用户名和密码相匹配生成服务特定凭证 (p. 16).

c. 有关可用终端节点的列表，请参阅the section called “服务端点” (p. 21)。

使用 Cassandra Go 客户端驱动程序以编程方式访问亚马逊 Keyspaces

本节说明了如何使用 Go 客户端驱动程序连接到 Amazon Keyspaces。要向用户和应用程序提供对 Amazon Keyspaces 资源进行编程访问的凭证，您可以执行以下任一操作：

- 创建与特定服务关联的服务特定凭证Amazon Identity and Access Management(IAM) 用户。
- 为了增强安全性，我们建议为所有用户和角色创建 IAM 访问密钥Amazon服务。适用于 Cassandra 客户端驱动程序的 Amazon Keyspaces Sigv4 身份验证插件使您能够使用 IAM 访问密钥而不是用户名和密码对对 Amazon Keyspaces 的调用进行身份验证。有关更多信息，请参阅 the section called “IAM 凭证Amazon身份验证” (p. 17)。

主题

开始前的准备工作

您需要完成以下任务才能开始执行以下任务。

Amazon Keyspaces 需要使用传输层安全性 (TLS) 来帮助保护与客户端的连接。要使用 TLS 连接到亚马逊 Keyspaces，您需要下载亚马逊数字证书并将 Python 驱动程序配置为使用 TLS。

使用以下命令下载 Starfield 数字证书并保存`sf-class2-root.crt`本地或您的主目录中。

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

> **Note**
>
> 您也可以使用亚马逊数字证书连接到亚马逊Keyspaces，如果您的客户端成功连接到亚马逊 Keyspaces，则可以继续这样做。Starfield 证书为使用旧证书颁发机构的客户提供了额外的向后兼容性。

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

使用 Apache Cassandra 的 Gocql 驱动程序和服务特定证书Connect 亚马逊 Keyspaces

1. 为您的应用程序创建一个目录。

```
mkdir ./gocqlexample
```

2. 导航到新目录。

```
cd gocqlexample
```

3. 为您的应用程序创建一个文件。

```
touch cqlapp.go
```

4. 下载 Go 驱动程序。

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

5. 将以下示例代码添加到 cqlapp.go 文件中。

```
package main

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

func main() {

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

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

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

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

使用说明：

a. Replace（替换）"*path_to_file*/sf-class2-root.crt"在第一步中保存了证书的路径。

b. 确保*ServiceUserName*和*ServicePassword*按照以下步骤与您在生成服务特定凭证时获得的用户名和密码相匹配生成服务特定凭证 (p. 16).

c. 有关可用终端节点的列表，请参阅the section called “服务端点” (p. 21)。

6. 构建程序。

```
go build cqlapp.go
```

7. 运行程序。

```
./cqlapp
```

使用 Apache Cassandra 的 Go 驱动程序和 SigV4 身份验证插件Connect 亚马逊 Keyspaces

以下代码示例显示了如何使用开源 Go 驱动程序的 SigV4 身份验证插件来访问 Amazon Keyspaces（适用于 Apache Cassandra）。

如果您尚未执行此操作，请按照以下步骤为您的 IAM 用户或角色创建凭证the section called “IAM 凭证 Amazon身份验证” (p. 17).

从以下位置将 Go SigV4 身份验证插件添加到您的应用程序中GitHub 知识库. 该插件支持 Cassandra 的开源 Go 驱动程序 4.x 版，依赖于Amazon适用于 SDK for Go。

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

在此代码示例中，Amazon Keyspaces 终端节点由`Cluster`类。它使用`AwsAuthenticator`以便集群的身份验证器属性获取证书。

```
package main

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

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

    cluster.Authenticator = auth

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

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

    // doing the query
    var text string
    iter := session.Query("SELECT keyspace_name FROM system_schema.tables;").Iter()
    for iter.Scan(&text) {
     fmt.Println("keyspace_name:", text)
    }
```

```
    if err := iter.Close(); err != nil {
     log.Fatal(err)
    }
}
```

使用说明：

1. Replace（替换）"*path_to_file*/sf-class2-root.crt"在第一步中保存了证书的路径。
2. 确保*AccessKeyId*和*SecretAccessKey*匹配您使用使用获得的访问密钥和秘密访问密钥AwsAuthenticator. 有关更多信息，请参阅 。配置Amazon适用于 Go 的开发工具包在Amazon SDK for Go.
3. 要在代码之外存储访问密钥，请参阅以下最佳实践the section called “如何管理访问密钥” (p. 20).
4. 有关可用终端节点的列表，请参阅the section called “服务端点” (p. 21)。

使用 Cassandra Perl 客户端驱动程序以编程方式访问亚马逊 Keyspaces

本节说明了如何使用 Perl 客户端驱动程序连接到 Amazon Keyspaces。在此代码示例中，我们使用了 Perl 5。Amazon Keyspaces 需要使用传输层安全性 (TLS) 来帮助保护与客户端的连接。

> **Important**
>
> 为了创建安全连接，我们的代码示例在建立 TLS 连接之前使用 Starfield 数字证书对服务器进行身份验证。Perl 驱动程序不验证服务器的 Amazon SSL 证书，这意味着您无法确认您正在连接亚马逊 Keyspaces。第二步，仍然需要将驱动程序配置为在连接到 Amazon Keyspaces 时使用 TLS，并确保在客户端和服务器之间传输的数据经过加密。

1. 从中下载 Cassandra DBI 驱动程序https://metacpan.org/pod/DBD::Cassandra并将驱动程序安装到您的 Perl 环境中。确切的步骤。以下是一个常见的例子。

```
cpanm DBD::Cassandra
```

2. 为您的应用程序创建一个文件。

```
touch cqlapp.pl
```

3. 将以下示例代码添加到 cqlapp.pl 文件中。

```
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**
>
> 确保*ServiceUserName*和*ServicePassword*按照以下步骤与您在生成服务特定凭证时获得的用户名和密码相匹配生成服务特定凭证 (p. 16).

Note

有关可用终端节点的列表，请参阅the section called “服务端点” (p. 21)。

4. 运行应用程序。

```
perl cqlapp.pl
```

开始使用 Amazon Keyspaces（针对 Apache Cassandra）

如果您是 Apache Cassandra 和 Amazon Keyspaces（Apache Cassandra）新手，那么本节适合您。在本节中，您需要安装所有必要的程序和驱动程序，才能成功使用 Amazon Keyspaces。

本节将介绍以下主题：

主题

教程的先决条件条件

在开始本教程之前，请按照Amazon中的设置说明访问 Amazon Keyspaces（针对 Apache Cassandra）(p. 14). 这些步骤包括注册Amazon然后创建Amazon Identity and Access Management有权访问亚马逊 Keyspaces 的 (IAM) 用户。

此外，如果您使用 `cqlsh` 或 Apache 2.0 许可的 Cassandra 客户端驱动程序完成本教程，请完成使用`cqlsh`连接到亚马逊 Keyspaces (p. 22)中的设置说明。

完成先决条件步骤后，继续执行步骤 1 教程：在亚马逊密钥空间中创建 Keyspaces 间和表 (p. 47).

步骤 1 教程：在亚马逊密钥空间中创建 Keyspaces 间和表

在本节中，您将使用控制台创建键空间，并向其中添加表。

Note

开始之前，请确保您拥有所有教程的前提.

主题

创建键空间

键空间 对与一个或多个应用程序相关的表进行分组。键空间包含一个或多个表，并为其包含的所有表定义复制策略。有关键空间的更多信息，请参阅以下主题：

- 使用密钥空间：the section called “CREATE 键空间” (p. 102)
- 数据定义语言 (DDL) 语句：Keyspaces (p. 207)
- Amazon Keyspaces（针对 Apache Cassandra）的配额 (p. 219)

创建键空间时，必须指定键空间名称。

Note

密钥空间的复制策略必须是`SingleRegionStrategy`.`SingleRegionStrategy`跨其中的三个可用区复制数据Amazon Web Services 区域.

使用控制台

使用控制台创建键空间

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 在导航窗格中，选择 Keyspaces (键空间)。
3. 选择 Create keyspace (创建键空间)。
4. 在 Keyspace name (键空间名称) 框中，输入 **`myGSGKeyspace`** 作为键空间的名称。

   名称约束：

   - 不能为空。
   - 允许的字符：字母数字字符和下划线 (_)。
   - 最大长度为 48 个字符。
5. 要创建键空间，请选择 Create keyspace (创建键空间)。
6. 通过执行以下操作，验证键空间 `myGSGKeyspace` 是否已创建：

   a. 在导航窗格中，选择 Keyspaces (键空间)。

   b. 在键空间列表中，查找键空间 `myGSGKeyspace`。

使用 CQL

以下过程使用 CQL 创建键空间。

使用 CQL 创建键空间

1. 打开命令 shell，输入以下内容：

   ```
   cqlsh
   ```

2. 使用以下 CQL 命令创建键空间。

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

   `SingleRegionStrategy`使用复制因子 3，跨三个复制因子复制数据Amazon其所在区域内的可用区。

   Note

   默认 Keyspaces 有输入为小写，除非括在引号中。在此情况下，请标注 `"myGSGKeyspace"`。

3. 验证键空间是否已创建。

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

   应当列出您的键空间。

创建表

表是组织和存储数据的位置。表的主键决定如何在表中对数据进行分区。主键由一个必需的分区键和一个或多个可选的聚类列组成。组成主键的组合值在表的所有数据中必须是唯一的。有关表的更多信息，请参阅以下主题：

- 使用表：the section called “创建表” (p. 103)
- DDDL 语句：表 (p. 209)
- 表资源管理：无服务器资源管理 (p. 88)
- 监视表资源利用率：the section called “使用 CloudWatch 进行监控” (p. 180)
- Amazon Keyspaces（针对 Apache Cassandra）的配额 (p. 219)

创建表时，应指定以下内容：

- 表的名称。
- 表中每一列的名称和数据类型。
- 表的主键。
  - 分区键— 必需
  - 聚类列— 可选

可使用以下过程创建具有指定列、数据类型、分区键和聚类列的表。

使用控制台

以下过程创建包含如下列和数据类型的表 `employees_tbl`。

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

使用控制台创建表

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 在导航窗格中，选择 Keyspaces (键空间)。
3. 选择 `myGSGKeyspace` 作为要在其中创建此表的键空间。
4. 选择 Create Table（创建表）。
5. 在 Table name (表名称) 框中，输入 **`employees_tbl`** 作为表的名称。

   名称约束：

   - 不能为空。
   - 允许的字符：字母数字字符和下划线 (_)。
   - 最大长度为 48 个字符。
6. 在 Columns (列) 部分中，为要添加到此表的每一列重复以下步骤。

   添加以下列和数据类型。

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

   a. 名称— 输入列的名称。

      名称约束：

      - 不能为空。
      - 允许的字符：字母数字字符和下划线 (_)。
      - 最大长度为 48 个字符。

   b. 类型— 在数据类型列表中，为此列选择数据类型。

   c. 如果要添加另一列，请选择 Add column (添加列)。
7. 选择`id`作为下面的分区键分区键. 每个表都需要一个分区键。分区键可以由一列或多列构成。
8. 添加 `division` 作为聚类列。聚类列是可选的，决定着每个分区内的排序顺序。

   a. 要添加聚类列，请选择 Add clustering column (添加聚类列)。

   b. 在 Column (列) 列表中，选择 division (分类)。在 Order (顺序) 列表中，选择 ASC (升序) 按照升序对此列中的值进行排序。（选择 DESC (降序) 可按降序排列。）
9. 在表设置部分，选择默认设置.

10. 选择 Create Table（创建表）。
11. 验证表是否已创建。
    a. 在导航窗格中，选择表。
    b. 确认您的表位于表列表中。
    c. 选择表的名称。
    d. 确认所有列和数据类型都正确无误。

       Note

       列的顺序可能与添加到表中的顺序不同。
    e. 在聚类列中，确认 division (分类) 标识为 true。其他所有表列应当为 false。

使用 CQL

以下过程使用 CQL 创建包含如下列和数据类型的表。`id` 列将作为分区键。

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

使用 CQL 创建表

1. 打开命令 shell，输入以下内容：

   `cqlsh`
2. 在 `cqlsh` 提示符 (`cqlsh>`) 处，指定要在其中创建表的键空间。

```
USE "myGSGKeyspace" ;
```

3. 在键空间提示符 (`cqlsh:keyspace_name>`) 处，通过在命令窗口中输入以下代码来创建表。

```
CREATE TABLE IF NOT EXISTS "myGSGKeyspace".employees_tbl (
   id text,
   name text,
   region text,
   division text,
   project text,
   role text,
   pay_scale int,
   vacation_hrs float,
   manager_id text,
   PRIMARY KEY (id,division))
   WITH CLUSTERING ORDER BY (division ASC) ;
```

   Note

   `ASC` 是默认的聚类顺序。也可以指定 `DESC` 按降序排列。

   请注意，`id` 列将作为分区键。然后，`division` 是按升序 (`ASC`) 排列的聚类列。

4. 验证表是否已创建。

```
SELECT * from system_schema.tables WHERE keyspace_name='myGSGKeyspace' ;
```

   应当列出您的表。

5. 验证表的结构。

```
SELECT * FROM system_schema.columns WHERE keyspace_name = 'myGSGKeyspace' AND
 table_name = 'employees_tbl' ;
```

   确认所有列和数据类型均符合预期。列的顺序可能与 CREATE 语句中的顺序不同。

要对表中的数据执行 CRUD（创建、读取、更新和删除）操作，请继续执行the section called “第 2 步：CRUD 操作” (p. 51)。

步骤 2 教程：创建、读取、更新和删除数据 (CRUD)

在本节中，您可以使用控制台中的 CQL 编辑器，对表中的数据执行 CRUD（创建、读取、更新和删除）操作。您也可以使用 cqlsh 运行命令。

主题

教程：将数据插入并加载到 Amazon Keyspaces 表中

要在 employees_tbl 表中创建数据，请使用 INSERT 语句添加单行。

1. 在使用 cqlsh 将数据写入 Amazon Keyspaces 表之前，必须将当前 cqlsh 会话的写入一致性设置为LOCAL_QUORUM. 有关支持的一致性级别的更多信息，请参the section called “写入一致性级别” (p. 12). 请注意，如果您使用的是 CQL 编辑器，则不需要此步骤。Amazon Web Services Management Console.

```
CONSISTENCY LOCAL_QUORUM;
```

2. 要插入单个记录，请在 CQL 编辑器中运行以下命令。

```
INSERT INTO "myGSGKeyspace".employees_tbl (id, name, project, region, division, role,
 pay_scale, vacation_hrs, manager_id)
VALUES ('012-34-5678','Russ','NightFlight','US','Engineering','IC',3,12.5,
 '234-56-7890') ;
```

3. 通过运行以下命令，验证数据是否已正确添加到表中。

```
SELECT * FROM "myGSGKeyspace".employees_tbl ;
```

使用 cqlsh 从文件插入多个记录

1. 下载以下压缩文件 sampledata.zip 中包含的样本数据文件 (employees.csv)。此 CSV（逗号分隔值）文件包含以下数据。记住文件的保存路径。

ID	name	project	region	division	role	pay_scale	vacation_hrs	manager_id
123-45-6789	Bob	NightFlight	US	Engineering	Intern	1	0	234-56-7890
234-56-7890	Bob	NightFlight	US	Engineering	Manager	6	72	789-01-2345
345-67-8901	Sarah	Storm	US	Engineering	IC	4	108	234-56-7890
456-78-9012	Beth	NightFlight	US	Engineering	IC	7	100.5	234-56-7890
567-89-0123	Ahmed	NightFlight	US	Marketing	IC	4	88	678-90-1234
678-90-1234	Alan	Storm	US	Marketing	Manager	3	18.4	789-01-2345
789-01-2345	Roberta	All	US	Executive	CEO	15	184	None

2. 打开命令 shell，输入以下内容：

   ```
   cqlsh
   ```

3. 在 cqlsh 提示符 (cqlsh>) 处，指定键空间。

   ```
   USE "myGSGKeyspace" ;
   ```

4. 将写入一致性设置为LOCAL_QUORUM. 有关支持的一致性级别的更多信息，请参the section called “写入一致性级别” (p. 12).

   ```
   CONSISTENCY LOCAL_QUORUM;
   ```

5. 在键空间提示符 (cqlsh:keyspace_name>) 处，运行以下查询。

   ```
   COPY employees_tbl
    (id,name,project,region,division,role,pay_scale,vacation_hrs,manager_id)
   FROM 'path-to-the-csv-file/employees.csv' WITH delimiter=',' AND header=TRUE ;
   ```

6. 通过运行以下查询，验证数据是否已正确添加到表中。

   ```
   SELECT * FROM employees_tbl ;
   ```

教程：从亚马逊 Keyspaces 表中阅读

在教程：将数据插入并加载到 Amazon Keyspaces 表中 (p. 51)一节中，您使用 SELECT 语句验证了已成功将数据添加到表中。在本节中，您可以细化使用 SELECT，以只显示特定列以及满足特定条件的行。

SELECT 语句的一般形式如下所示。

```
SELECT column_list FROM table_name [WHERE condition [ALLOW FILTERING]] ;
```

主题

- 选择表中的所有数据 (p. 52)
- 选择列的子集 (p. 53)
- 选择行的子集 (p. 53)

选择表中的所有数据

SELECT 语句最简单的形式是返回表中的所有数据。

Important

在生产环境中，运行此命令通常不是最佳做法，该命令将返回表中的所有数据。

选择表的所有数据

- 运行以下 查询。

```
SELECT * FROM "myGSGKeyspace".employees_tbl ;
```

将通配符 (*) 用于 column_list 可选择所有列。

选择列的子集

查询列的子集

- 要仅检索 id、name 和 manager_id 列，请运行以下查询。

```
SELECT name, id, manager_id FROM "myGSGKeyspace".employees_tbl ;
```

输出将按照 SELECT 语句中列出的顺序，仅包含指定的列。

选择行的子集

查询大型数据集时，您可能只需要满足特定条件的记录。为此，您可以在 SELECT 语句末尾追加一个 WHERE 子句。

查询行的子集

- 要仅检索 ID 为 '234-56-7890' 的员工的记录，请运行以下查询。

```
SELECT * FROM "myGSGKeyspace".employees_tbl WHERE id='234-56-7890' ;
```

前面的 SELECT 语句仅返回 id 为 234-56-7890 的行。

了解WHERE条款

WHERE 子句用于筛选数据，并仅返回满足指定条件的数据。指定的条件可以是简单条件或复合条件。

如何在 **WHERE** 子句中使用条件

- 一个简单的条件 — 单列。

```
WHERE column_name=value
```

只要满足以下任何条件，便可以在 WHERE 子句中使用简单条件：

- 该列是表分区键中唯一的列。
- 在 WHERE 子句中的条件后添加 ALLOW FILTERING。

请注意，使用 ALLOW FILTERING 可能会导致性能不稳定，特别是对于多分区的大型表。

- 复合条件 — 通过连接的多个简单条件AND.

```
WHERE column_name1=value1 AND column_name2=value2 AND column_name3=value3...
```

只要满足以下任何条件，便可以在 WHERE 子句中使用复合条件：

- WHERE 子句中的列与表分区键中的列完全匹配，不多也不少。
- 在 WHERE 子句中的复合条件后添加 ALLOW FILTERING，如下例所示。

```
SELECT * FROM my_table WHERE col1=5 AND col2='Bob' ALLOW FILTERING ;
```

请注意，使用 ALLOW FILTERING 可能会导致性能不稳定，特别是对于多分区的大型表。

试试看

创建您自己的 CQL 查询，以从 employees_tbl 表中查找以下内容：

- 查找所有员工的 name、project 和 id。
- 查找实习生 Bob 正在从事的项目（在输出中至少包括他的姓名、项目和角色）。
- 高级：创建应用程序，以查找与经理相同的所有员工Bob实习生。提示：这可能需要多个查询。
- 高级：创建应用程序，以查找从事项目的所有员工的选定列NightFlight. 提示：解决此问题可能需要多个语句。

教程：更新亚马逊 Keyspaces 表中的数据

要更新 employees_tbl 表中的数据，请使用 UPDATE 语句。

UPDATE 语句的一般形式如下所示。

```
UPDATE table_name SET column_name=new_value WHERE primary_key=value ;
```

Tip

- 您可以使用逗号分隔的 column_names 和值列表来更新多列，如下例所示。

```
UPDATE my_table SET col1='new_value_1', col2='new_value2' WHERE id='12345' ;
```

- 如果主键由多列构成，则必须将所有主键列及其值包含在 WHERE 子句中。
- 不能更新主键中的任何列，因为这会更改记录的主键。

更新单个单元格

使用 employees_tbl 表，给 ID 为 567-89-0123 的员工加薪。

```
UPDATE "myGSGKeyspace".employees_tbl SET pay_scale=5 WHERE id='567-89-0123' AND
 division='Marketing' ;
```

验证该员工的薪酬等级现在是否为 5。

```
SELECT * FROM "myGSGKeyspace".employees_tbl WHERE id='567-89-0123' ;
```

试试看

高级：你的公司雇用了 Bob 的实习生。更改其记录，使其角色为 'IC'，薪酬等级为 2。

教程：删除亚马逊 Keyspaces 表中的数据

要删除 employees_tbl 表中的数据，请使用 DELETE 语句。

您可以从行或分区中删除数据。删除数据时请务必小心，因为删除操作是不可逆的。

从表中删除一行或所有行不会删除该表。因此，您可以用数据重新填充该表。删除表将删除表及其中的所有数据。要再次使用表，必须重新创建表并向其中添加数据。删除键空间会删除键空间及其中的所有表。要使用键空间和表，必须重新创建，然后用数据填充。

删除单元格

从行中删除列会从指定单元格中删除数据。使用 SELECT 语句显示该列时，数据将显示为 *null*，尽管该位置并未存储 null 值。

删除一个或多个特定列的一般语法如下所示。

```
DELETE column_name1[, column_name2...] FROM table_name WHERE condition ;
```

在 employees_tbl 表中，可以看到 CEO 具有 "None" 位经理。首先，删除该单元格，以便其中不会有任何数据。

删除特定单元格

1. 运行以下 DELETE 查询。

   ```
   DELETE manager_id FROM "myGSGKeyspace".employees_tbl WHERE id='789-01-2345' AND
    division='Executive';
   ```

2. 验证删除操作是否如预期。

   ```
   SELECT * FROM "myGSGKeyspace".employees_tbl WHERE id='789-01-2345' AND
    division='Executive';
   ```

删除行

有时候可能需要删除整个行，例如当员工退休时。删除行的一般语法如下所示。

```
DELETE FROM table_name WHERE condition ;
```

删除行

1. 运行以下 DELETE 查询。

   ```
   DELETE FROM "myGSGKeyspace".employees_tbl WHERE id='456-78-9012' AND
    division='Engineering';
   ```

2. 验证删除操作是否如预期。

   ```
   SELECT * FROM "myGSGKeyspace".employees_tbl WHERE id='456-78-9012' AND
    division='Engineering';
   ```

教程步骤 3：删除亚马逊密钥空间中的表和 Keyspaces 间

为避免为不需要的表和数据支付费用，请删除所有不使用的表和键空间。表删除后，该表及其数据将被删除，其费用累计将停止。但是，键空间仍然保留。键空间删除后，键空间及其所有表将被删除，其费用累计将停止。

删除表

您可以使用控制台或 CQL 删除表。表删除后，该表及其所有数据都将被删除。

使用控制台

以下过程使用 Amazon Web Services Management Console删除表及其所有数据。

使用控制台删除表

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 在导航窗格中，选择表。
3. 选中要删除的每个表名称左侧的框。
4. 请选择 Delete（删除）。
5. 在 Delete table (删除表) 屏幕上，在框中输入 **`Delete`**。然后，选择 Delete table (删除表)。
6. 要验证表是否已删除，请在导航窗格中选择 Tables (表)，然后确认该 `employees_tbl` 表已不再列出。

使用 CQL

以下过程使用 CQL 删除表及其所有数据。

使用 CQL 删除表

1. 打开命令 shell，输入以下内容：

   `cqlsh`
2. 通过在键空间提示符 (`cqlsh:`*`keyspace_name`*`>`) 处输入以下命令来删除表。

   ```
   DROP TABLE IF EXISTS "myGSGKeyspace".employees_tbl ;
   ```
3. 验证表是否已删除。

   ```
   SELECT * FROM system_schema.tables WHERE keyspace_name = 'myGSGKeyspace' ;
   ```

   您的表不应再列出。

删除键空间

您可以使用 Amazon Web Services Management Console或 CQL 删除键空间。键空间删除后，键空间及其所有表和数据都将被删除。

使用 Amazon Web Services Management Console

以下过程使用 Amazon Web Services Management Console删除键空间及其所有表和数据。

使用控制台删除键空间

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 在导航窗格中，选择 Keyspaces (键空间)。
3. 选中要删除的每个键空间名称左侧的框。
4. 请选择 Delete（删除）。
5. 在 Delete keyspace (删除键空间) 屏幕上，在框中输入 **`Delete`**。然后，选择 Delete keyspace (删除键空间)。
6. 要验证键空间 `myGSGKeyspace` 是否已删除，请在导航窗格中选择 Keyspaces (键空间)，然后确认该键空间已不再列出。因为你删除了它的密钥空间，`employees_tbl`下表表也不应再列出。

使用 CQL

以下过程使用 CQL 删除键空间及其所有表和数据。

使用 CQL 删除键空间

1. 打开命令 shell，输入以下内容：

   `cqlsh`
2. 通过在键空间提示符 (`cqlsh:`*`keyspace_name`*`>`) 处输入以下命令来删除键空间。

```
DROP KEYSPACE IF EXISTS "myGSGKeyspace" ;
```

3. 验证键空间是否已删除。

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

   您的键空间不应再列出。

迁移到 Amazon Keyspaces

Amazon Keyspaces（用于 Apache Cassandra）是一种可扩展、高可用、托管的 Apache Cassandra 兼容数据库服务。您可以使用本部分中的步骤将数据从在本地或 Amazon Elastic Compute Cloud (Amazon EC2) 上运行的 Cassandra 数据库迁移到 Amazon Keyspaces。

我们建议您遵循以下最佳实践确保迁移成功：

- 将迁移细分为较小的组件。

  考虑以下迁移单位及其在原始数据大小方面的潜在足迹。在一个或多个阶段迁移少量数据可能有助于简化迁移。

  按集群— 一次迁移所有 Cassandra 数据。这种方法对于较小的集群可能没问题。

  按键空间或表格— 将迁移分解为一组密钥空间或表格。这种方法可以帮助您根据每个工作负载的要求分阶段迁移数据。

  按数据— 考虑为特定用户或产品组迁移数据，以便进一步降低数据的大小。
- 基于简单性，优先迁移哪些数据。

  考虑一下是否有可以更容易地首先迁移的数据，例如，在特定时间内不会更改的数据、来自夜间批处理作业的数据、离线时段未使用的数据或来自内部应用程序的数据。
- 使用特定的工具。
  - 使用 cqlsh 快速开始将数据加载到亚马逊 Keyspaces`COPY FROM`命令。cqlsh 包含在 Apache Cassandra 中，最适合加载小型数据集或测试数据。适用于 step-by-step 请参阅the section called “使用 cqlsh 加载数据” (p. 58).
  - 对于具有大型数据集的生产工作负载，您可以使用 DataStax 批量加载器适用于 Apache Cassandra Keyspaces 用`dsbulk`命令。DSBulk 提供了更强大的导入功能，可从GitHub 存储库. 适用于 step-by-step 请参阅the section called “使用 DSBulk 加载数据” (p. 65).
  - 要了解如何使用 Apache Cassandra Spark 连接器将数据写入 Amazon Keyspaces，请参阅与 Apache Spark 集成 (p. 112).
  - 对于复杂迁移，请考虑使用提取、转换和加载 (ETL) 工具。您可以使用 Amazon EMR 快速有效地执行数据转换工作负载。有关更多信息，请参阅 Amazon EMR 管理指南。

主题

教程：使用 cqlsh 将数据加载到亚马逊Keyspaces

该 step-by-step 教程将指导Keyspaces 使用`cqlsh COPY`命令。在本教程中，您将执行以下操作：

主题

- 问题排查 (p. 64)

先决条件

在开始本教程之前，您必须完成以下任务。

1. 如果您尚未完成此操作，请注册Amazon Web Services 账户按照以下步骤操作the section called “注册 Amazon” (p. 14).
2. 按照中的步骤创建特定于服务的证书the section called “使用控制台生成服务特定凭证” (p. 16).
3. 设置 Cassandra 查询语言 shell (cqlsh) 连接，并按照以下步骤确认您可以连接到 Amazon Keyspacesthe section called “使用 `cqlsh`” (p. 22).

第 1 步：创建源 CSV 文件和目标表

在本教程中，我们使用逗号分隔值 (CSV) 文件`keyspaces_sample_table.csv`作为数据迁移的源文件。提供的示例文件包含名为的表的几行数据`book_awards`.

1. 创建源文件。您可以选择以下选项之一：
   - 下载 CSV 文件示例 (`keyspaces_sample_table.csv`) 包含在以下存档文件中samplemigration.zip. 解压缩存档并记下到`keyspaces_sample_table.csv`.
   - 要使用存储在 Apache Cassandra 数据库中的自己的数据填充 CSV 文件，您可以使用`cqlsh COPY TO`语句，如以下示例所示。

     ```
     cqlsh localhost 9042 -u "username" -p "password" --execute "COPY mykeyspace.mytable
      TO 'keyspaces_sample_table.csv' WITH HEADER=true"
     ```

     确保您创建的 CSV 文件满足以下要求：

     - 第一行包含列名。
     - 源 CSV 文件中的列名称与目标表中的列名称相匹配。
     - 数据用逗号分隔。
     - 所有数据值均为有效的亚马逊Keyspaces 数据类型。请参阅 the section called “数据类型” (p. 204)。
2. 在 Amazon Keyspaces 中创建目标密钥空间和表。
   a. 使用Connect 亚马逊Keyspaces`cqlsh`，将以下示例中的服务终端节点、用户名和密码替换为您自己的值。

      ```
      cqlsh cassandra.us-east-2.amazonaws.com 9142 -u "111122223333" -p "wJalrXUtnFEMI/
      K7MDENG/bPxRfiCYEXAMPLEKEY" --ssl
      ```

   b. 使用名称创建一个新的密钥空间`catalog`如以下示例所示。

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

   c. 当新的密钥空间可用时，使用以下代码创建目标表`book_awards`.

      ```
      CREATE TABLE "catalog.book_awards" (
         year int,
         award text,
         rank int,
         category text,
         book_title text,
      ```

```
    author text,
    publisher text,
    PRIMARY KEY ((year, award), category, rank)
    );
```

如果 Apache Cassandra 是您的原始数据源，那么创建具有匹配标头的 Amazon Keyspaces 目标表的简单方法是生成CREATE TABLE语句，如以下语句所示。

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

然后在 Amazon Keyspaces 中使用与 Cassandra 源表中的描述相匹配的列名和数据类型创建目标表。

第 2 步：准备数据

准备源数据以进行高效传输是一个两步过程。首先，对数据进行随机化。在第二步中，您将分析数据以确定适当的cqlsh参数值和必需的表设置。

随机化数据

这些区域有：cqlsh COPY FROM命令读取和写入数据的顺序与数据在 CSV 文件中显示的顺序相同。如果您将cqlsh COPY TO命令来创建源文件，则在 CSV 中按键排序顺序写入数据。在内部，亚马逊Keyspaces 使用分区键对数据进行分区。尽管 Amazon Keyspaces 具有内置逻辑来帮助对同一个分区键的请求进行负载均衡，但如果您随机排列顺序，加载数据会更快、更高效。这是因为您可以利用 Amazon Keyspaces 写入不同分区时发生的内置负载均衡功能。

要在各分区之间均匀分布写入，必须随机化源文件中的数据。您可以编写应用程序来执行此操作，也可以使用开源工具，例如Shuf. Shuf 可以在 Linux 发行版和 macOS 上免费获得（通过在Homebrew）和 Windows（通过使用适用于 Windows 的 Windows System m system m system m system m 在此步骤中，还需要一个额外的步骤来防止带有列名的标题行被洗牌。

要在保留标头的同时随机化源文件，请输入以下代码。

```
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 将数据重写为一个名为的新 CSV 文件keyspace.table.csv. 您现在可以删除keyspaces_sample_table.csv文件-您不再需要它。

分析数据

通过分析数据确定平均行大小和最大行大小。

您这样做的原因如下：

- 平均行大小有助于估计要传输的数据总量。
- 您需要平均行大小来预置数据上传所需的写入容量。
- 您可以确保每行的大小小于 1 MB，这是 Amazon Keyspaces 中的最大行大小。

> **Note**
>
> 此配额是指行大小，而不是分区大小。与 Apache Cassandra 分区不同，Amazon Keyspaces 分区的大小实际上可以解除绑定。分区键和聚类列需要额外的元数据存储空间，您必须将其添加到行的原始大小中。有关更多信息，请参阅 the section called “计算行大小” (p. 106)。

以下代码使用AWK来分析 CSV 文件并打印平均行大小和最大行大小。

```
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
```

运行此代码将生成以下输出。

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

在本教程的下一步中，您将使用平均行大小来预置表的写入容量。

第 3 步：设置表的吞吐容量

本教程介绍如何调整 cqlsh 以在设定的时间范围内加载数据。由于您事先知道执行了多少读取和写入操作，因此请使用预配置容量模式。完成数据传输后，应将表的容量模式设置为与应用程序的流量模式相匹配。要了解有关容量管理的更多信息，请参阅无服务器资源管理 (p. 88).

在预配置容量模式下，您可以提前指定要为表预配置的读取和写入容量。写入容量按小时计费，并以写入容量单位 (WCU) 计量。每个 WCU 的写入容量足以支持每秒写入 1 KB 的数据。加载数据时，写入速率必须低于最大 WCU（参数：`write_capacity_units`) 在目标表上设置。

默认情况下，您最多可以为一个表配置 40,000 个 WCU，在账户中的所有表中配置最多 80,000 个 WCU。如果您需要额外的容量，您可以在Service Quotas控制台。有关配额的更多信息，请参阅配额 (p. 219)。

计算刀片所需的平均 WCU 数量

每秒插入 1 KB 的数据需要 1 个 WCU。如果您的 CSV 文件有 360,000 行，并且您想在 1 小时内加载所有数据，则必须每秒写入 100 行（360,000 行/60 分钟/60 秒 = 每秒 100 行）。如果每行最多有 1 KB 的数据，要每秒插入 100 行，则必须为表预配 100 个 WCU。如果每行有 1.5 KB 的数据，则需要两个 WCU 才能每秒插入一行。因此，要每秒插入 100 行，必须预置 200 个 WCU。

要确定每秒插入一行需要多少个 WCU，请将平均行大小（以字节为单位）除以 1024，然后向上舍入到最接近的整数。

例如，如果平均行大小为 3000 字节，则需要三个 WCU 才能每秒插入一行。

```
ROUNDUP(3000 / 1024) = ROUNDUP(2.93) = 3 WCUs
```

计算数据加载时间和容量

现在您已经知道了 CSV 文件中的平均大小和行数，您可以计算在给定时间内需要加载数据的 WCU 数量，以及使用不同的 WCU 设置加载 CSV 文件中的所有数据所需的大概时间。

例如，如果文件中的每一行都是 1 KB，而 CSV 文件中有 1,000,000 行，则要在 1 小时内加载数据，则需要在该小时内为表预置至少 278 个 WCU。

```
1,000,000 rows * 1 KBs = 1,000,000 KBs
1,000,000 KBs / 3600 seconds =277.8 KBs / second = 278 WCUs
```

配置预置容量设置

您可以在创建表时设置表的写入容量设置，也可以使用`ALTER TABLE`CQL 命令。以下是使用变更表的预配置容量设置的语法`ALTER TABLE`CQL 语句。

```
ALTER TABLE mykeyspace.mytable WITH custom_properties={'capacity_mode':{'throughput_mode':
 'PROVISIONED', 'read_capacity_units': 100, 'write_capacity_units': 278}} ;
```

有关完整的语言参考，请参阅the section called “ALTER TABLE” (p. 211).

第 4 步：配置cqlsh COPY FROM设置

本部分概述如何确定参数值cqlsh COPY FROM. 这些区域有：cqlsh COPY FROM命令会读取您之前准备的 CSV 文件，然后使用 CQL 将数据插入到 Amazon Keyspaces 中。该命令将行分开并分发INSERT一组工作人员之间的操作。每个工作人员都与 Amazon Keyspaces 建立连接并发送INSERT沿此频道的请求。

这些区域有：cqlsh COPY命令没有内部逻辑来在其工作线程之间均匀分配工作。但是，您可以手动配置它，以确保工作均匀分布。首先查看以下关键 cqlsh 参数：

- 分隔符— 如果使用逗号以外的分隔符，则可以设置此参数，默认为逗号。
- 吸收— 目标行数cqlsh COPY FROM每秒尝试处理的次数。如果未设置，则默认为 100,000。
- 数字进程— clsh 为其创建的子工作进程的数量COPY FROM任务。此设置的最大值为 16，默认值为num_cores - 1，其中num_cores是运行 cqlsh 的主机上的处理内核数。
- 最大批量大小— 批处理大小将确定单个批次中插入到目标表的最大行数。如果未设置，cqlsh 将使用一批 20 个插入的行。
- CHUNKSIZE— 传递给童工的工作单位的大小。默认情况下，将它设置为 5,000。
- MAX ATT— 重试失败的工作程序区块的最大次数。达到最大尝试次数后，失败的记录将写入新的 CSV 文件，您可以在调查故障后再次运行该文件。

SetINGESTRATE基于您预配到目标目标表的 WCU 的数量。这些区域有：INGESTRATE的cqlsh COPY FROM命令不是限制，而是目标平均值。这意味着它可以（而且经常）突破你设定的数字。要允许爆发并确保有足够的容量来处理数据加载请求，请将INGESTRATE到表写入容量的 90%。

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

接下来，将NUMPROCESSES参数等于比系统上的内核数少一。要了解系统的内核数量，可以运行以下代码。

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

在本教程中，我们使用以下值。

```
NUMPROCESSES = 4
```

每个进程都会创建一个工作进程，每个工作进程都建立一个与 Amazon Keyspaces 的连接。Amazon Keyspaces 在每个连接上每秒最多可支持 3,000 个 CQL 请求。这意味着您必须确保每个 worker 每秒处理的请求少于 3,000 个。

和INGESTRATE，workers 通常会突然超过你设置的数字，不受时钟秒数的限制。因此，要考虑突发情况，请将 cqlsh 参数设置为每秒处理 2,500 个请求。要计算分配给工作人员的工作量，请使用以下指南。

- DivideINGESTRATE通过NUMPROCESSES.
- 如果INGESTRATE/NUMPROCESSES > 2,500，降低INGESTRATE使这个公式成真。

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

在配置设置以优化样本数据的上传之前，让我们先查看cqlsh默认设置，并了解使用它们如何影响数据上传流程。由于cqlsh COPY FROM使用CHUNKSIZE来创建大块的作品（INSERT语句）分配给工作人员，则工作不会自动均匀分配。有些工作人员可能会闲置，这取决于INGESTRATE设置。

要在工作人员之间均匀分配工作并使每个工作程序保持在每秒 2,500 个请求的最佳速率，您必须设置CHUNKSIZE、MAXBATCHSIZE, 和INGESTRATE通过更改输入参数。要在数据加载期间优化网络流量利

用率，请为MAXBATCHSIZE这接近最大值 30。通过更改CHUNKSIZE到 100 和MAXBATCHSIZE到 25 时，这 10,000 行在四个工作进程中均匀分布（10,000/2500 = 4）。

以下代码示例对此进行了说明。

```
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
```

要进行总结，请在设置时使用以下公式cqlsh COPY FROM参数：

- 吸收= write_capacity_units * .90
- 数字进程= num_cores -1（默认）
- 吸收/NUMPROCESS= 2,500（这必须是真实的陈述。）
- 最大批量大小= 30（默认为 20。亚马逊Keyspaces 最多可接受 30 个批次。）
- CHUNKSIZE= (INGESTRATE/NUMPROCESS)/MAXBATCHSIZE

现在你已经计算好了NUMPROCESSES、INGESTRATE, 和CHUNKSIZE，您已准备好加载数据。

第 5 步：运行cqlsh COPY FROM命令

运行cqlsh COPY FROM命令，请完成以下步骤。

1. 使用 cqlsh Connect 亚马逊Keyspaces。
2. 使用以下代码选择您的密钥空间。

```
use mykeyspace;
```

3. 将写入一致性设置为LOCAL_QUORUM. 为确保数据持久性，Amazon Keyspaces 不允许其他写入一致性设置。请看下面的代码。

```
CONSISTENCY LOCAL_QUORUM;
```

4. 准备您的cqlsh COPY FROM语法使用以下代码示例。

```
COPY mytable FROM './keyspace.table.csv' WITH HEADER=true
AND INGESTRATE=calculated ingestrate
AND NUMPROCESSES=calculated numprocess
AND MAXBATCHSIZE=20
AND CHUNKSIZE=calculated chunksize;
```

5. 运行在上一步中准备的语句。cqlsh 会回显您配置的所有设置。
   a. 确保设置与您的输入相匹配。请参阅以下 示例。

```
Reading options from the command line: {'chunksize': '120', 'header': 'true',
 'ingestrate': '36000', 'numprocesses': '15', 'maxbatchsize': '20'}
Using 15 child processes
```

   b. 查看传输的行数和当前平均速率，如以下示例所示。

```
Processed: 57834 rows; Rate: 6561 rows/s; Avg. rate: 31751 rows/s
```

c. 当 cqlsh 完成数据上传后，查看数据加载统计信息的摘要（读取的文件数、运行时间和跳过的行数），如下例所示。

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

在本教程的最后一步中，您已将数据上传到 Amazon Keyspaces。

Important

现在，您已经传输了数据，请调整目标表的容量模式设置，以匹配应用程序的常规流量模式。在更改预配置容量之前，您需要按小时费率支付费用。

问题排查

数据上传完成后，检查是否跳过了行。为此，请导航到源 CSV 文件的源目录，然后搜索具有以下名称的文件。

```
import_yourcsvfilename.err.timestamp.csv
```

cqlsh 会将任何跳过的数据行写入具有该名称的文件中。如果文件存在于您的源目录中并且其中包含数据，则这些行不会上传到 Amazon Keyspaces。要重试这些行，请先检查上载期间是否遇到任何错误，然后相应地调整数据。要重试这些行，可以重新运行该流程。

常见错误

未加载行的最常见原因是容量错误和解析错误。

向亚马逊Keyspaces 上传数据时出现无效请求错误

在以下示例中，源表包含一个计数器列，这会导致记录来自 cqlsh 的批处理调用`COPY`命令。亚马逊 Keyspaces 不支持记录的批量调用。

```
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
```

要解决此错误，请使用 DSBulk 迁移数据。有关更多信息，请参阅 the section called “使用 DSBulk 加载数据” (p. 65)。

将数据上传到亚马逊Keyspaces 时出现解析器错误

以下示例显示了由于以下原因而跳过的行`ParseError`.

```
Failed to import 1 rows: ParseError - Invalid ... -
```

要解决此错误，您需要确保要导入的数据与 Amazon Keyspaces 中的表架构匹配。检查导入文件是否存在解析错误。你可以尝试使用单行数据`INSERT`语句来隔离错误。

将数据上传到亚马逊Keyspaces 时出现容量错误

```
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 Keysaces 使用`ReadTimeout`和`WriteTimeout`异常，用于指示写入请求何时因吞吐容量不足而失败。为了帮助诊断容量不足异常，Amazon Keyspaces 发布了`WriteThrottleEvents`和`ReadThrottledEvents`Amazon 中的指标 CloudWatch. 有关更多信息，请参阅 the section called “使用 CloudWatch 进行监控” (p. 180)。

将数据上传到亚马逊Keyspaces 时出现 cqlsh 错误

要帮助排除 cqlsh 错误，请使用`--debug`标志。

使用 cqlsh 版本时，您会看到以下错误。

```
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
```

运行以下命令确认是否已安装 cqlsh 版本。

```
cqlsh --version
```

输出应该类似于以下内容。

```
cqlsh 5.0.1
```

如果您使用的是 Windows，请替换所有实例`cqlsh`和`cqlsh.bat`. 例如，要在 Windows 中检查 cqlsh 版本，请运行以下命令。

```
cqlsh.bat --version
```

cqlsh 客户端从服务器接收到连续三次任何类型的错误后，与 Amazon Keyspaces 的连接将失败。cqlsh 客户端失败并显示以下消息。

```
Failed to import 1 rows: NoHostAvailable - , will retry later, attempt 3 of 100
```

要解决此错误，您需要确保要导入的数据与 Amazon Keyspaces 中的表架构匹配。检查导入文件是否存在解析错误。您可以尝试使用 INSERT 语句来隔离错误，从而使用单行数据。

客户端会自动尝试重新建立连接。

教程：使用 DSBulk 将数据加载到亚马逊Keyspaces

该 step-by-step 教程将指导Keyspaces 使用 DataStax 批量加载器 (DSBulk) 在GitHub. 在本教程中，您完成以下步骤：

主题

先决条件

在开始本教程之前，您必须完成以下任务。

1. 如果您尚未完成此操作，请注册一个Amazon账户，请按照以下步骤操作the section called “注册 Amazon” (p. 14).
2. 按照中的步骤创建凭证the section called “创建凭证” (p. 15).

   Note

   DSBulk 当前仅支持服务特定凭证。如果您更喜欢通过使用来管理对 Amazon Keyspaces 的访问权限Amazon Identity and Access Management用户和角色，则在使用 DSBulk 完成数据上传后，您可以使用 SiGv4 身份验证插件并禁用特定于服务的凭据。
3. 创建 JKS 信任存储库文件。

   a. 使用以下命令下载 Starfield 数字证书并保存`sf-class2-root.crt`本地或您的主目录中。

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

   Note

   您还可以使用亚马逊数字证书连接到 Amazon Keyspaces，如果您的客户端成功连接到 Amazon Keyspaces，则可以继续这样做。Starfield 证书为使用旧证书颁发机构的客户端提供了额外的向后兼容性。

   b. 将星空数字证书转换为 TrustStore 文件。

   ```
   openssl x509 -outform der -in sf-class2-root.crt -out temp_file.der
   keytool -import -alias cassandra -keystore cassandra_truststore.jks -file
    temp_file.der
   ```

   在此步骤中，您需要为密钥库创建密码并信任此证书。交互式命令如下所示。

   ```
   Enter keystore password:
   Re-enter new password:
   Owner: OU=Starfield Class 2 Certification Authority, O="Starfield Technologies,
    Inc.", C=US
   Issuer: OU=Starfield Class 2 Certification Authority, O="Starfield Technologies,
    Inc.", C=US
   Serial number: 0
   Valid from: Tue Jun 29 17:39:16 UTC 2004 until: Thu Jun 29 17:39:16 UTC 2034
   Certificate fingerprints:
     MD5:  32:4A:4B:BB:C8:63:69:9B:BE:74:9A:C6:DD:1D:46:24
     SHA1: AD:7E:1C:28:B0:64:EF:8F:60:03:40:20:14:C3:D0:E3:37:0E:B5:8A
     SHA256:
    14:65:FA:20:53:97:B8:76:FA:A6:F0:A9:95:8E:55:90:E4:0F:CC:7F:AA:4F:B7:C2:C8:67:75:21:FB:5F:B6:5
   Signature algorithm name: SHA1withRSA
   Subject Public Key Algorithm: 2048-bit RSA key
   Version: 3
   Extensions:
   #1: ObjectId: 2.5.29.35 Criticality=false
   AuthorityKeyIdentifier [
   KeyIdentifier [
   0000: BF 5F B7 D1 CE DD 1F 86   F4 5B 55 AC DC D7 10 C2  ._.......[U.....
   0010: 0E A9 88 E7                                        ....
   ]
   [OU=Starfield Class 2 Certification Authority, O="Starfield Technologies, Inc.",
    C=US]
   SerialNumber: [    00]
   ]
   ```

```
#2: ObjectId: 2.5.29.19 Criticality=false
BasicConstraints:[
  CA:true
  PathLen:2147483647
]
#3: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: BF 5F B7 D1 CE DD 1F 86   F4 5B 55 AC DC D7 10 C2  ._.......[U.....
0010: 0E A9 88 E7                                        ....
]
]
Trust this certificate? [no]:  y
```

4. 设置 Cassandra 查询语言 shell (cqlsh) 连接，并按照以下步骤确认您可以连接到 Amazon Keyspacesthe section called “使用 cqlsh” (p. 22).
5. 下载并安装 DSBulk。

   a. 要下载 DSBulk，您可以使用以下代码。

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

   b. 然后解压 tar 文件并将 dsBulk 添加到您的PATH如以下示例所示。

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

   c. 创建application.conf文件来存储 DSBulk 要使用的设置。您可以将以下示例另存为./dsbulk_keyspaces.conf. Replacelocalhost如果您不在本地节点上，请与本地 Cassandra 群集的联系人联系，例如 DNS 名称或 IP 地址。Replaceusername和password使用您的服务器凭据。记下文件名和路径，因为稍后需要在dsbulk load命令。

   ```
   datastax-java-driver {
     basic.contact-points = [ "localhost"]
     advanced.auth-provider {
           class = PlainTextAuthProvider
           username = "username"
           password = "password"
     }
   }
   ```

第 1 步：创建源 CSV 文件和目标表

在本教程中，我们使用逗号分隔值 (CSV) 文件keyspaces_sample_table.csv作为数据迁移的源文件。提供的示例文件包含名为的表的几行数据book_awards.

1. 创建源文件。您可以选择以下选项之一：
   - 下载 CSV 文件样例 (keyspaces_sample_table.csv) 包含在以下存档文件中samplemigration.zip. 解压缩存档并记下到keyspaces_sample_table.csv.
   - 要使用存储在 Apache Cassandra 数据库中的您自己的数据填充 CSV 文件，您可以使用填充源 CSV 文件dsbulk unload如以下示例所示。

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

     确保您创建的 CSV 文件满足以下要求：

- 第一行包含列名。
- 源 CSV 文件中的列名称与目标表中的列名称相匹配。
- 数据用逗号分隔。
- 所有数据值均为有效的亚马逊Keyspaces 数据类型。请参阅 the section called “数据类型” (p. 204)。

2. 在 Amazon Keyspaces 中创建目标密钥空间和表。

   a. 使用Connect 亚马逊Keyspacescqlsh，将以下示例中的服务终端节点、用户名和密码替换为您自己的值。

   ```
   cqlsh cassandra.us-east-2.amazonaws.com 9142 -u "111122223333" -p "wJalrXUtnFEMI/
   K7MDENG/bPxRfiCYEXAMPLEKEY" --ssl
   ```

   b. 使用名称创建一个新的密钥空间catalog如以下示例所示。

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

   c. 在新密钥空间的状态为 available 后，使用以下代码创建目标表book_awards. 要详细了解异步资源创建以及如何检查资源是否可用，请参阅the section called “CREATE 键空间” (p. 102).

   ```
   CREATE TABLE catalog.book_awards (
      year int,
      award text,
      rank int,
      category text,
      book_title text,
      author text,
      publisher text,
      PRIMARY KEY ((year, award), category, rank)
      );
   ```

   如果 Apache Cassandra 是您的原始数据源，那么创建具有匹配标头的 Amazon Keyspaces 目标表的简单方法是生成CREATE TABLE语句，如以下语句所示。

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

   然后在 Amazon Keyspaces 中使用与 Cassandra 源表中的描述相匹配的列名和数据类型创建目标表。

第 2 步：准备数据

为高效传输准备源数据是一个两步过程。首先，对数据进行随机化。在第二步中，您将分析数据以确定适当的dsbulk参数值和必需的表设置。

随机化数据

这些区域有：dsbulk命令读取和写入数据的顺序与数据在 CSV 文件中显示的顺序相同。如果您将dsbulk命令来创建源文件，则在 CSV 中按键排序顺序写入数据。在内部，亚马逊Keyspaces 使用分区键对数据进行分区。尽管 Amazon Keyspaces 具有内置逻辑来帮助对同一个分区键的请求进行负载均衡，但如果您随机排列顺序，加载数据会更快、更高效。这是因为您可以利用 Amazon Keyspaces 写入不同分区时发生的内置负载均衡功能。

要在各分区之间均匀分布写入，必须随机化源文件中的数据。您可以编写应用程序来执行此操作，也可以使用开源工具，例如Shuf. Shuf 可以在 Linux 发行版和 macOS 上免费获得（通过在Homebrew) 和 Windows

(通过使用适用于 Windows 子系统 (WSL))。在此步骤中，还需要一个额外的步骤来防止带有列名的标题行被洗牌。

要在保留标头的同时随机化源文件，请输入以下代码。

```
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 将数据重写为一个名为的新 CSV 文件`keyspace.table.csv`. 您现在可以删除`keyspaces_sample_table.csv`文件-您不再需要它。

分析数据

通过分析数据确定平均行大小和最大行大小。

您这样做的原因如下：

- 平均行大小有助于估计要传输的数据总量。
- 您需要平均行大小来预置数据上传所需的写入容量。
- 您可以确保每行的大小小于 1 MB，这是 Amazon Keyspaces 中的最大行大小。

> **Note**
>
> 此配额是指行大小，而不是分区大小。与 Apache Cassandra 分区不同，Amazon Keyspaces 分区的大小实际上可以解除绑定。分区键和聚类列需要额外的元数据存储空间，您必须将其添加到行的原始大小中。有关更多信息，请参阅 the section called “计算行大小” (p. 106)。

以下代码使用AWK来分析 CSV 文件并打印平均行大小和最大行大小。

```
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
```

运行此代码将生成以下输出。

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

确保最大行大小不超过 1 MB。如果是这样，则必须拆分行或压缩数据以使行大小低于 1 MB。在本教程的下一步中，您将使用平均行大小来预置表的写入容量。

第 3 步：设置表的吞吐容量

本教程向您演示如何调整 DSBulk 以在设定的时间范围内加载数据。由于您事先知道执行了多少读取和写入操作，因此请使用预配置容量模式。完成数据传输后，应将表的容量模式设置为与应用程序的流量模式相匹配。要了解有关容量管理的更多信息，请参阅无服务器资源管理 (p. 88).

在预配置容量模式下，您可以提前指定要为表预配置的读取和写入容量。写入容量按小时计费，并以写入容量单位 (WCU) 计量。每个 WCU 的写入容量足以支持每秒写入 1 KB 的数据。加载数据时，写入速率必须低于最大 WCU（参数：`write_capacity_units`) 在目标表上设置的。

默认情况下，您最多可以为一个表配置 40,000 个 WCU，在账户中的所有表中配置最多 80,000 个 WCU。如果您需要额外容量，您可以在Service Quotas控制台。有关配额的更多信息，请参阅配额 (p. 219)。

计算刀片所需的平均 WCU 数量

每秒插入 1 KB 的数据需要 1 个 WCU。如果您的 CSV 文件有 360,000 行，并且您想在 1 小时内加载所有数据，则必须每秒写入 100 行（360,000 行/60 分钟/60 秒 = 每秒 100 行）。如果每行最多有 1 KB 的数据，要每秒插入 100 行，则必须为表预配 100 个 WCU。如果每行有 1.5 KB 的数据，则需要两个 WCU 才能每秒插入一行。因此，要每秒插入 100 行，必须预置 200 个 WCU。

要确定每秒插入一行需要多少个 WCU，请将平均行大小（以字节为单位）除以 1024，然后向上舍入到最接近的整数。

例如，如果平均行大小为 3000 字节，则需要三个 WCU 才能每秒插入一行。

```
ROUNDUP(3000 / 1024) = ROUNDUP(2.93) = 3 WCUs
```

计算数据加载时间和容量

现在您已经知道了 CSV 文件中的平均大小和行数，您可以计算在给定时间内需要加载数据的 WCU 数量，以及使用不同的 WCU 设置加载 CSV 文件中的所有数据所需的大概时间。

例如，如果文件中的每一行都是 1 KB，而 CSV 文件中有 1,000,000 行，则要在 1 小时内加载数据，则需要在该小时内为表预置至少 278 个 WCU。

```
1,000,000 rows * 1 KBs = 1,000,000 KBs
1,000,000 KBs / 3600 seconds =277.8 KBs / second = 278 WCUs
```

配置预置容量设置

您可以在创建表时设置表的写入容量设置，也可以使用ALTER TABLE命令。以下是使用变更表的预配置容量设置的语法ALTER TABLE命令。

```
ALTER TABLE catalog.book_awards WITH custom_properties={'capacity_mode':{'throughput_mode':
 'PROVISIONED', 'read_capacity_units': 100, 'write_capacity_units': 278}} ;
```

有关完整的语言参考，请参阅the section called “CREATE TABLE” (p. 209)和the section called “ALTER TABLE” (p. 211).

第 4 步：配置DSBulk设置

本节概述了配置 DSBulk 以将数据上传到 Amazon Keyspaces 所需的步骤。您可以使用配置文件配置 DSBulk。您可以直接从命令行指定配置文件。

1. 为迁移到 Amazon Keyspaces 创建一个 DSBulk 配置文件，在此示例中，我们使用文件名dsbulk_keyspaces.conf. 在 DSBulk 配置文件中指定以下设置。

   a. *PlainTextAuthProvider*— 使用创建身份验证提供程序PlainTextAuthProvider类。ServiceUserName和ServicePassword应与您按照中的步骤生成服务特定凭证时获得的用户名和密码相匹配the section called “创建凭证” (p. 15).

   b. *local-datacenter*— 设置的值local-datacenter转到Amazon Web Services 区域你正在连接的。例如，如果应用程序正在连接到cassandra.us-east-2.amazonaws.com，然后将本地数据中心设置为us-east-2. 对于所有可用Amazon Web Services 区域，请参阅the section called “服务端点” (p. 21).

   c. *SSLEngineFactory*— 要配置 SSL/TLS，请初始化SSLEngineFactory通过在配置文件中添加一个部分，用一行指定类class = DefaultSslEngineFactory. 提供路径cassandra_truststore.jks和您之前已创建的密码。

   d. *consistency*— 将一致性级别设置为LOCAL QUORUM并关闭token_metadata设置。不支持其他写入一致性级别，有关更多信息，请参阅the section called “支持的 Cassandra 一致性级别” (p. 12).

   以下是完整的示例配置文件。

```
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"
}

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.metadata {
    schema {
      token-map.enabled = false
    }
}
}
```

2. 查看 DSBulk 的参数load命令。

   a. *executor.maxPerSecond*— load 命令每秒尝试并发处理的最大行数。如果未设置，则该设置将以 -1 禁用。

      Setexecutor.maxPerSecond基于您预配到目标目标表的 WCU 的数量。这些区域有：executor.maxPerSecond的load命令不是限制，而是目标平均值。这意味着它可以（而且经常）突破你设定的数字。要允许爆发并确保有足够的容量来处理数据加载请求，请将executor.maxPerSecond到表写入容量的 90%。

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

      在本教程中，我们设置executor.maxPerSecond到 5.

      > Note
      >
      > 如果您使用 DSBulk 1.6.0 或更高版本，您可以使用dsbulk.engine.maxConcurrentQueries相反。

   b. 为 DSBulk 配置这些附加参数load命令。

      - *batch-mode*— 此参数告诉系统按分区键对操作进行分组。由于这可能会干扰其他设置，因此我们建议禁用批处理模式。
      - *driver.advanced.retry-policy-max-retries*— 确定要重试失败查询的次数。如果未设置，则默认值为 10。您可以按需调整该值。
      - *driver.basic.request.timeout*— 系统等待查询返回的时间（以分钟为单位）。如果未设置，则默认值为 “5 分钟”。您可以按需调整该值。

第 5 步：运行 dsBulkload命令

在本教程的最后一步中，您将数据上传到 Amazon Keyspaces。

运行 DSBulkload命令后，完成以下步骤。

1. 运行以下代码，将数据从 csv 文件上传到亚马逊Keyspaces 表。确保更新之前创建的应用程序配置文件的路径。

```
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. 输出包括日志文件的位置，该日志文件详细说明了成功和不成功的操作。该文件存储在以下目录中。

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

3. 日志文件条目将包括指标，如下例所示。检查以确保行数与 csv 文件中的行数一致。

```
total | failed | rows/s | p50ms | p99ms | p999ms
   200 |      0 |    200 | 21.63 | 21.89 |  21.89
```

Important

现在，您已经传输了数据，请调整目标表的容量模式设置，以匹配应用程序的常规流量模式。在更改预配置容量之前，您需要按小时费率支付费用。有关更多信息，请参阅the section called “读/写容量模式” (p. 88)。

对 Amazon Keyspaces 进行故障排除（针对 Apache Cassandra）

以下部分提供了有关如何排查您在使用 Amazon Keyspaces 时可能遇到的常见配置问题（适用于 Apache Cassandra）的信息。

有关 IAM 访问特定的故障排除指导，请参阅the section called “问题排查” (p. 175).

有关安全性最佳实践的更多信息，请参阅the section called “安全最佳实践” (p. 200).

主题

亚马逊 Keyspaces 中的连接问题

连接时遇到问题？ 以下内容介绍一些常见问题以及如何解决这些问题

连接到 Amazon Keyspaces 终端节点时出错

连接失败和连接错误可能会导致不同的错误消息。以下部分介绍了最常见的场景。

主题

我无法使用 cqlsh 连接到 Amazon Keyspaces

您正在尝试使用 cqlsh 连接到 Amazon Keyspaces 终端节点，但连接失败并显示**`Connection error`**.

如果您尝试连接到 Amazon Keyspaces 表并且 cqlsh 尚未正确配置，则连接将失败。以下部分提供了在您尝试使用 cqlsh 建立连接时导致连接错误的最常见配置问题的示例。

你正在尝试使用 cqlsh 连接到亚马逊 Keyspaces，但是你得到了连接**`timed out`**错误消息。

如果你没有提供正确的端口，可能会出现这种情况，这会导致以下错误。

```
#  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")})
```

要解决此问题，请验证您使用的是端口 9142 进行连接。

你正在尝试使用 cqlsh 连接到亚马逊 Keyspaces，但是你会得到一个**Name or service not known**错误消息。

如果您使用的终端节点拼写错误或不存在，可能会出现这种情况。在以下示例中，终端节点的名称拼写错误。

```
#  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
```

要在使用公共终端节点进行连接时解决此问题，请从the section called “服务端点” (p. 21)，并验证终端节点的名称没有任何错误。如果您使用 VPC 终端节点进行连接，请验证 VPC 终端节点信息在 cqlsh 配置中是否正确。

你正在尝试使用 cqlsh 连接到亚马逊 Keyspaces，但是你会收到一个**OperationTimedOut**错误消息。

Amazon Keyspaces 要求为连接启用 SSL 以确保强大的安全性。如果您收到以下错误，则可能会丢失 SSL 参数。

```
# 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',)})
#
```

若要解决此问题，请将以下标志添加到 cqlsh 连接命令中。

```
--ssl
```

你正在尝试使用 cqlsh 连接到亚马逊 Keyspaces，你会收到一个**SSL transport factory requires a valid certfile to be specified**错误消息。

在这种情况下，SSL/TLS 证书的路径丢失，导致以下错误。

```
# 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.
#
```

若要解决此问题，请添加计算机上的 Certfile 的路径。

```
certfile =  path_to_file/sf-class2-root.crt
```

你正在尝试使用 cqlsh 连接到亚马逊 Keyspaces，但是你会收到一个**No such file or directory**错误消息。

如果计算机上证书文件的路径错误，可能会出现这种情况，从而导致以下错误。

```
# cat .cassandra/cqlshrc
[connection]
port = 9142
factory = cqlshlib.ssl.ssl_transport_factory

[ssl]
validate = true
certfile = /root/wrong_path/sf-class2-root.crt
#



# cqlsh cassandra.us-east-1.amazonaws.com -u "USERNAME" -p "PASSWORD" --ssl
Connection error: ('Unable to connect to any servers', {'3.234.248.192': IOError(2, 'No
 such file or directory')})
#
```

要解决此问题，请验证计算机上证书文件的路径是否正确。

你正在尝试使用 cqlsh 连接到亚马逊 Keyspaces，但是你会收到一个**[X509] PEM lib**错误消息。

如果 SSL/TLS 证书文件，则可能会被覆盖`sf-class2-root.crt`无效，将导致以下错误。

```
# 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)")})
#
```

要解决此问题，请使用以下命令下载 Starfield 数字证书。Save（保存）`sf-class2-root.crt`在本地或在您的主目录中。

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

你正在尝试使用 cqlsh 连接到亚马逊 Keyspaces，但是你会收到一个**unknown**SSL 错误。

如果 SSL/TLS 证书文件，则可能会被覆盖`sf-class2-root.crt`为空，将导致以下错误。

```
# 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)")})
#
```

要解决此问题，请使用以下命令下载 Starfield 数字证书。Save（保存）`sf-class2-root.crt`在本地或在您的主目录中。

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

你正在尝试使用 cqlsh 连接到亚马逊 Keyspaces，但是你会收到一个**SSL: CERTIFICATE_VERIFY_FAILED**错误消息。

如果 SSL/TLS 证书文件无法验证，则可能会出现这种情况，从而导致以下错误。

```
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)")})
```

要解决此问题，请使用以下命令再次下载该证书文件。Save（保存）`sf-class2-root.crt`在本地或在您的主目录中。

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

你正在尝试使用 cqlsh 连接到亚马逊 Keyspaces，但是你收到了**`Last error: timed out`**错误消息。

如果您没有为 Amazon EC2 安全组中的 Amazon Keyspaces 配置出站规则，则可能会出现这种情况，这将导致以下错误。

```
# 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")})
#
```

要解决此问题，请检查您的 Amazon EC2 实例是否具有阻止与 Amazon Keyspaces 连接的出站规则。如果是这种情况，您需要查看 EC2 实例的安全组，并添加允许向 Amazon Keyspaces 资源出站流量的规则。要更新出站规则以允许流量进入 Amazon Keyspaces，请输入端口`9142`(对于)端口范围和`0.0.0.0/0`作为目的地.

有关如何查看和编辑 EC2 出站规则的更多信息，请参阅。在适用于 Linux 实例的 Amazon EC2 用户指南中向安全组添加规则。.

你正在尝试使用 cqlsh 连接到亚马逊 Keyspaces，但是你会收到一个**`Unauthorized`**错误消息。

如果您在 IAM 用户策略中缺少 Amazon Keyspaces 权限，可能会出现这种情况，这会导致以下错误。

```
# 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."',)})
#
```

要解决此问题，请确保 IAM 用户`testuser-at-12345678910`有权访问 Amazon Keyspaces。有关授予对 Amazon Keyspaces 访问权限的 IAM 策略示例，请参阅。the section called “基于身份的策略示例” (p. 165).

有关特定于 IAM 访问的故障排除指导，请参阅the section called “问题排查” (p. 175).

你正在尝试使用 cqlsh 连接到亚马逊 Keyspaces，但是你会收到一个**`Bad credentials`**错误消息。

如果用户名或密码错误，可能会出现这种情况，导致以下错误。

```
# 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"',)})
#
```

要解决此问题，请验证###和##代码中的用户名和密码匹配生成时获取的用户名和密码。服务特定凭证 (p. 16).

Important

如果尝试连接 cqlsh 时仍然看到错误，请使用`--debug`选项并在联系时包括详细输出Amazon Web Services Support.

我无法使用 Cassandra 客户端驱动程序连接到 Amazon Keyspaces

以下各节显示了连接 Cassandra 客户端驱动程序时最常见的错误。

您正在尝试使用驱动程序和 sigv4 插件连接到 Amazon Keyspaces 表，但是您会收到一个**`AttributeError`**错误消息。

如果证书配置不正确，则会导致以下错误。

```
cassandra.cluster.NoHostAvailable: ('Unable to connect to any servers',
 {'44.234.22.154:9142': AttributeError("'NoneType' object has no attribute 'access_key'")})
```

要解决此问题，请确认您在使用 Sigv4 插件时传递了与 IAM 用户或角色关联的凭证。sigv4 插件需要以下凭据。

- `AWS_ACCESS_KEY_ID`— 指定Amazon与 IAM 用户或角色关联的访问密钥。
- `AWS_SECRET_ACCESS_KEY`— 指定与访问密钥关联的私有密钥。这基本上是访问密钥的“密码”。

要了解有关访问密钥和 Sigv4 插件的更多信息，请参阅the section called “IAM 凭证Amazon身份验证” (p. 17).

您正在尝试使用驱动程序连接到 Amazon Keyspaces 表，但是您会收到**`PartialCredentialsError`**错误消息。

如果`AWS_SECRET_ACCESS_KEY`丢失，则可能会导致以下错误。

```
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')})
```

若要解决此问题，请验证您是否同时通过了`AWS_ACCESS_KEY_ID`和`AWS_SECRET_ACCESS_KEY`在使用 sigv4 插件时。要了解有关访问密钥和 Sigv4 插件的更多信息，请参阅the section called “IAM 凭证Amazon身份验证” (p. 17).

您正在尝试使用驱动程序连接到 Amazon Keyspaces 表，但是您会收到**`Invalid signature`**错误消息。

如果您使用了错误的凭据，可能会出现这种情况，导致以下错误。

```
cassandra.cluster.NoHostAvailable: ('Unable to connect to any servers',
 {'44.234.22.134:9142':
 AuthenticationFailed('Failed to authenticate to 44.234.22.134:9142: Error from server:
 code=0100
 [Bad credentials] message="Authentication failure: Invalid signature"')})
```

要解决此问题，请验证您传递的证书是否与您配置为访问 Amazon Keyspaces 的 IAM 用户或角色关联。要了解有关访问密钥和 Sigv4 插件的更多信息，请参阅the section called “IAM 凭证Amazon身份验证” (p. 17).

我无法使用连接cassandra-stress

您正在尝试 Keyspaces 用**cassandra-stress**命令，但是你正在收到**SSL context**错误消息。

如果您尝试连接到亚马逊 Keyspaces，但是您没有正确的 TrustStore 设置，就会发生这种情况。Amazon Keyspaces 要求使用传输层安全性 (TLS) 来帮助保护与客户端的连接。

在此情况下，您会看到以下错误。

```
Error creating the initializing the SSL Context
```

要解决此问题，请按照本主题中所示的说明设置 TrustStorethe section called “开始前的准备工作” (p. 29).

TrustStore 设置完毕后，您应该可以使用以下命令进行连接。

```
./cassandra-stress user profile=./profile.yaml n=100 "ops(insert=1,select=1)"
 cl=LOCAL_QUORUM -node "cassandra.eu-north-1.amazonaws.com" -port native=9142 -transport
 ssl-alg="PKIX" truststore="./cassandra_truststore.jks" truststore-password="trustStore_pw"
 -mode native cql3 user="user_name" password="password"
```

我无法使用 IAM 身份进行连接

您正在尝试使用 IAM 身份连接到 Amazon Keyspaces 表，但是您收到了**Unauthorized**错误消息。

如果您尝试使用 IAM 身份（例如，IAM 用户）连接到 Amazon Keyspaces 表而不实施策略并首先向用户授予所需权限，则会发生这种情况。

在此情况下，您会看到以下错误。

```
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."',)})
```

若要解决此问题，请验证 IAM 用户的权限。要使用标准驱动程序进行连接，用户必须至少具有SELECT要访问系统表，因为大多数驱动程序在建立连接时读取系统键空间/表。

例如，授予对 Amazon Keyspaces 系统和用户表的访问权限的 IAM 策略，请参阅the section called “Amazon Keyspaces” (p. 167).

要查看 IAM 特定的故障排除部分，请参阅the section called “问题排查” (p. 175).

我正在尝试使用 cqlsh 导入数据，并且与我的亚马逊 Keyspaces 表的连接丢失了

您正在尝试使用 cqlsh 将数据上传到亚马逊 Keyspaces，但是收到连接错误。

在 cqlsh 客户端收到来自服务器的连续三个任何类型的错误后，与 Amazon Keyspaces 的连接失败。cqlsh 客户端失败并显示以下消息。

```
Failed to import 1 rows: NoHostAvailable - , will retry later, attempt 3 of 100
```

要解决此错误，您需要确保要导入的数据与 Amazon Keyspaces 中的表架构匹配。查看导入文件以了解错误。你可以尝试使用一行数据，方法是使用 INSERT 语句隔离错误。

客户端会自动尝试重新建立连接。

亚马逊 Keyspaces 中的容量管理故障

无服务器容量遇到问题？ 以下内容介绍一些常见问题以及如何解决这些问题

无服务器容量错误

本节概述了如何识别与无服务器容量管理相关的错误以及如何解决这些错误。例如，当您的应用程序超出了预置吞吐容量时，您可能会发现容量不足事件。

由于 Apache Cassandra 是基于群集的软件，旨在在一群节点上运行，因此它没有与无服务器功能（例如吞吐量容量）相关的异常消息。大多数驱动程序只了解 Apache Cassandra 中可用的错误代码，因此 Amazon Keyspaces 使用同一组错误代码来维护兼容性。

要将 Cassandra 错误映射到底层容量事件，您可以使用亚马逊 CloudWatch 以监控相关的亚马逊 Keyspaces 指标。可以根据导致该事件的资源将导致客户端错误的容量不足事件分为以下三组：

- 表— 如果您选择已预置表的容量模式，并且您的应用程序超出了预置吞吐量，则可能会发现容量不足错误。有关更多信息，请参阅 the section called “读/写容量模式” (p. 88)。
- 分区— 如果给定分区的流量超过 3000 RCU 或 1000 WCU，则可能会遇到容量不足的事件。作为最佳实践，我们建议您在分区间统一分配流量。有关更多信息，请参阅 数据建模 (p. 110)。
- Connection— 如果超过每个连接每秒每秒最大操作数的配额，则可能会遇到吞吐量不足的情况。要提高吞吐量，您可以在使用驱动程序配置连接时增加默认连接数。有关更多信息，请参阅 the section called “CQL 查询吞吐量调整” (p. 7) 和 the section called “负载均衡” (p. 8)。

要确定哪些资源导致容量不足事件返回客户端错误，您可以在 Amazon Keyspaces 控制台中查看控制面板。默认情况下，控制台提供最常见容量和流量相关的聚合视图 CloudWatch 中的指标容量和相关指标部分容量表格的选项卡。

要使用 Amazon CloudWatch 创建自己的控制面板，请查看以下亚马逊 Keyspaces 指标。

- `PerConnectionRequestRateExceeded`— 对亚马逊 Keyspaces 的请求超过每个连接请求费率的配额。与 Amazon Keyspaces 的每个客户端连接每秒最多可支持 3000 个 CQL 请求。通过创建多个连接，每秒可以执行 3000 多个请求。
- `ReadThrottleEvents`— 对亚马逊 Keyspaces 的请求超过表的读取容量。
- `StoragePartitionThroughputCapacityExceeded`— 对超过分区吞吐量的 Amazon Keyspaces 存储分区的请求。Amazon Keyspaces 存储分区最多可以支持每秒 1000 个 WCU/WRU 或每秒 3000 个 RCU/RRU，或者两者的线性组合。为了减少这些例外情况，我们建议您查看数据模型，以便在更多分区之间分配读/写流量。
- `WriteThrottleEvents`— 对超出表写入容量的 Amazon Keyspaces 的请求。

要了解有关 CloudWatch 的更多信息，请the section called “使用 CloudWatch 进行监控” (p. 180). 要获得所有可用的列表 CloudWatch Amazon Keyspaces 指标，请参阅the section called “指标与维度” (p. 181).

> Note
>
> 要开始使用显示 Amazon Keyspaces 的所有常见指标的自定义仪表板，您可以使用预构建的 CloudWatch 模板可用于 GitHub 中的Amazon样本存储库。

主题

我正在收到NoHostAvailable客户端驱动程序的容量不足错误

你看**Read_Timeout**要么**Write_Timeout**表格的例外情况。

反复尝试对容量不足的 Amazon Keyspaces 表进行写入或读取操作可能会导致特定于驱动程序的客户端错误。

使用 CloudWatch 以监视您的预置和实际吞吐量指标以及表的容量不足事件。例如，没有足够的吞吐容量的读取请求会失败，并出现Read_Timeout例外并发布到ReadThrottleEvents指标。没有足够的吞吐容量的写入请求会失败，并出现Write_Timeout例外并发布到WriteThrottleEvents指标。有关这些指标的更多信息，请参阅 the section called “指标与维度” (p. 181)。

若要解决这些问题，请考虑以下选项之一。

- 增加预配置吞吐对于表，这是应用程序可以消耗的最大吞吐容量。有关更多信息，请参阅 the section called “读取容量单位和写入容量单位” (p. 90)。
- 让服务通过自动扩展代表您管理吞吐量容量。有关更多信息，请参阅 the section called “将 Application Auto Scaling 管理吞吐容量” (p. 92)。
- 选择按需表容量模式。有关更多信息，请参阅 the section called “按需容量模式” (p. 89)。

如果您需要增加账户的默认容量配额，请参阅配额 (p. 219).

你看到与超出分区容量有关的错误。

当分区容量暂时超出时（可能由自适应容量或按需容量自动处理）时，可能会发生分区限制。此错误也可能表明您的数据模型存在问题。Amazon Keyspaces 存储分区最多可以支持每秒 1000 个 WCU/WRU 或每秒 3000 个 RCU/RRU，或者两者的线性组合。要了解有关如何改进数据模型以在更多分区之间分配读/写流量的详细信息，请参阅数据建模 (p. 110).

Write_Timeout例外也可能是由于同一逻辑分区中包含静态和非静态数据的并发写入操作的速率提高造成的。如果预计流量将运行多个并发写入操作，这些操作包括同一逻辑分区中的静态和非静态数据，我们建议分别编写静态和非静态数据。单独编写数据也有助于优化吞吐量成本。

您看到与超出连接请求率有关的错误。

连接限制可能是由以下某个原因所致。

- 您可能没有为每个会话配置足够的连接。
- 由于您没有正确配置 VPC 终端节点权限，您获得的连接可能比可用的对等体少。有关 VPC 终端节点策略的更多信息，请参阅the section called “对 Amazon Keyspaces 使用接口 VPC 终端节点” (p. 196).
- 如果您使用的是 4.x 驱动程序，请检查是否启用了主机名验证。默认情况下驱动程序启用 TLS 主机名验证此配置导致 Amazon Keyspaces 在驱动程序中显示为单节点群集。我们建议您关闭主机名验证。

我们建议您遵循以下最佳实践，以确保您的连接和吞吐量得到优化：

- 配置 CQL 查询吞吐量调整。

  Amazon Keyspaces 支持每秒每个 TCP 连接 3000 次 CQL 查询，但驱动程序可建立的连接数没有限制。

  大多数开源的 Cassandra 驱动程序都建立了一个与 Cassandra 的连接池，并通过该连接池进行负载平衡查询。Amazon Keyspaces 向驱动程序公开 9 个对等 IP 地址。大多数驱动程序的默认行为是建立到每个对等 IP 地址的单个连接。因此，使用默认设置的驱动程序的最大 CQL 查询吞吐量将是每秒 27,000 次 CQL 查询。

  要增大此数字，我们建议您增加驱动程序在其连接池中维护的每个 IP 地址的连接数。例如，如果将每个 IP 地址的最大连接数设置为 2，则将使驱动程序的最大吞吐量增加一倍，达到每秒 54,000 次 CQL 查询。

- 优化单节点连接。

  默认情况下，大多数开源的 Cassandra 驱动程序与在`system.peers`在建立会话时表。但是，某些配置可能会导致驱动程序连接到单个 Amazon Keyspaces IP 地址。如果驱动程序正在尝试对对等节点进行 SSL 主机名验证，可能会发生这种情况（例如， DataStax Java 驱动程序），或者当它通过 VPC 终端节点进行连接时。

  要获得与连接到多个 IP 地址的驱动程序相同的可用性和性能，我们建议您执行以下操作：
  - 根据所需的客户端吞吐量，将每个 IP 的连接数增加到 9 个或更高。
  - 创建自定义重试策略，以确保对同一节点运行重试。
  - 如果您使用 VPC 终端节点，请授予用于连接到 Amazon Keyspaces 的 IAM 实体访问权限，以便查询 VPC 以获取终端节点和网络接口信息。这改善了负载平衡并提高了读/写吞吐量。有关更多信息，请参阅 ??? (p. 197)。

我在数据导入过程中收到写入超时错误

使用上传数据时，您收到超时错误**`cqlsh COPY`**命令。

```
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`表示写入请求何时由于吞吐容量不足而失败的例外情况。为了帮助诊断容量不足的例外情况，Amazon Keyspaces 在 Amazon CloudWatch 中发布了以下指标。

- `WriteThrottleEvents`
- `ReadThrottledEvents`
- `StoragePartitionThroughputCapacityExceeded`

要解决数据加载过程中容量不足的错误，请降低每个工作器的写入速率或总摄入速率，然后重试上传行。有关更多信息，请参阅 the section called “第 4 步：配置`cqlsh COPY FROM`设置” (p. 62)。要获得更强大的数据上传选项，请考虑使用 DSBulk，它可从GitHub 存储库. 如需分步指导，请参阅 the section called “使用 DSBulk 加载数据” (p. 65)。

我看不到密钥空间或表的实际存储大小

你看不到密钥空间或表的实际存储大小。

您可以通过开始计算表中的行大小来估计存储大小。有关计算行大小的详细说明可在the section called “计算行大小” (p. 106).

Amazon Keyspaces 中的数据定义语言疑难解答

在创建资源时遇到困难？ 以下内容介绍一些常见问题以及如何解决这些问题

数据定义语言错误

Amazon Keyspaces 异步执行数据定义语言 (DDL) 操作，例如，创建和删除密钥空间和表。如果应用程序在资源准备就绪之前尝试使用该资源，则操作将失败。

您可以在中监控新密钥空间和表的创建状态Amazon Web Services Management Console，表示键空间或表处于挂起状态或活动状态的时间。您还可以通过查询系统架构表以编程方式监视新密钥空间或表的创建状态。键空间或表在可供使用时在系统架构中变得可见。

Note

使用以下方法优化密钥空间的创建Amazon CloudFormation，您可以使用此实用程序将 CQL 脚本转换为 CloudFormation 模板。可从中获得此工具GitHub 存储库.

主题

我创建了一个新的密钥空间，但我无法查看或访问它

您收到来自试图访问新密钥空间的应用程序的错误。

如果您尝试访问仍在异步创建的新创建的 Amazon KKeyspaces 密钥空间，您将收到错误消息。以下错误就是一个示例。

```
InvalidRequest: Error from server: code=2200 [Invalid query] message="unconfigured keyspace
 mykeyspace"
```

检查新密钥空间何时可供使用的推荐设计模式是轮询 Amazon Keyspaces 系统架构表 (system_schema_mcs.*)。

有关更多信息，请参阅 the section called “CREATE 键空间” (p. 102)。

我创建了一张新表，但我无法查看或访问它

您收到来自试图访问新表的应用程序的错误。

如果您尝试访问仍在异步创建的新创建的 Amazon Keyspaces 表，您将收到错误消息。例如，尝试查询尚不可用的表失败并显示`unconfigured table`错误消息。

```
InvalidRequest: Error from server: code=2200 [Invalid query] message="unconfigured table
 mykeyspace.mytable"
```

尝试使用查看表`sync_table()`无法使用`KeyError`.

```
KeyError: 'mytable'
```

检查新表何时可供使用的推荐设计模式是轮询 Amazon Keyspaces 系统架构表 (system_schema_mcs.*)。

这是正在创建的表的示例输出。

```
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)
```

这是处于活动状态的表的示例输出。

```
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)
```

有关更多信息，请参阅 the section called “创建表” (p. 103)。

我正在尝试使用亚马逊 Keyspaces 还原表 point-in-time 恢复 (PITR)，但恢复失败

如果您尝试使用还原 Amazon Keyspaces 表 point-in-time 恢复 (PITR)，您会看到恢复过程已开始但未成功完成，因此您可能尚未配置恢复过程为此特定表所需的所有必需权限。

除了用户权限之外，Amazon Keyspaces 可能需要许可才能在恢复过程中代表您的委托人执行操作。如果表格使用客户托管密钥加密，或者您使用的是限制传入流量的 IAM 策略，则属于这种情况。

例如，如果您在 IAM 策略中使用条件密钥将源流量限制到特定终端节点或 IP 范围，则恢复操作将失败。要允许 Amazon Keyspaces 代表您的委托人执行表还原操作，您必须添加`aws:ViaAWSService`IAM 策略中的全局条件密钥。

有关恢复表的权限的更多信息，请参阅。the section called “还原权限” (p. 123).

我正在尝试使用 INSERT/UPDATE 来编辑自定义生存时间 (TTL) 设置，但操作失败

如果您尝试插入或更新自定义 TTL 值，则操作可能会失败并显示以下错误。

```
TTL is not yet supported.
```

使用以下方法为行或列指定自定义 TTL 值`INSERT`要么`UPDATE`操作，您必须先为表启用 TTL。您可以使用`ttl`自定义属性。

有关启用表自定义 TTL 设置的更多信息，请参阅。the section called “如何使用自定义属性在现有表上启用生存时间 (TTL)” (p. 137).

我正在尝试将数据上传到我的 Amazon Keyspaces 表中，出现超过列数的错误

您正在上传数据，已超过可同时更新的列数。

当您的表架构超过 350 KB 的最大大小时，会发生此错误。有关更多信息，请参阅 配额 (p. 219)。

我正在尝试删除我的 Amazon Keyspaces 表中的数据，但删除该范围内失败

你试图通过分区键删除数据并收到范围删除错误。

当您尝试在一次删除操作中删除 1,000 多行时，会出现此错误。

```
Range delete requests are limited by the amount of items that can be deleted in a single
 range.
```

有关更多信息，请参阅 the section called “范围删除” (p. 8)。

要删除单个分区中的 1,000 多行，请考虑以下选项。

- 按分区删除 — 如果大多数分区位于 1,000 行以下，则可以尝试按分区删除数据。如果分区包含 1,000 行以上，请尝试通过群集列删除。
- 按聚类删除列 — 如果模型包含多个聚类列，则可以使用列层次结构删除多行。聚类列是一种嵌套结构，您可以通过对顶级列进行操作来删除许多行。
- 按单个行删除 — 您可以遍历这些行，然后按其完整的主键（分区列和聚类列）删除每行。
- 作为最佳做法，请考虑在分区之间拆分行-在 Amazon Keyspaces 中，我们建议您在表分区之间分配吞吐量。这样可以在物理资源之间均匀分配数据和访问，从而提供最佳吞吐量。有关更多信息，请参阅 数据建模 (p. 110)。

在为繁重的工作负载规划删除操作时，还可以考虑以下建议。

- 使用 Amazon Keyspaces，分区可以包含几乎无限制的行数。这使您可以将分区扩展到比传统的 100MB 卡桑德拉指南 “更宽”。随着时间的推移，时间序列或账本增长超过一千兆字节的数据并不罕见。
- 使用 Amazon Keyspaces，当您必须对繁重的工作负载执行删除操作时，不需要考虑压缩策略或墓碑。您可以根据需要删除尽可能多的数据，而不会影响读取性能。

Amazon Keyspaces（针对 Apache Cassandra）代码示例和工具

本节提供有关 Amazon Keyspaces（针对 Apache Cassandra）代码示例和工具的信息。

主题

图书馆和示例

您可以在 GitHub 上找到 Amazon Keyspaces 开源库和开发人员工具Amazon和Amazon样本存储库。

Amazon Keyspaces (for Apache Cassandra) 开发者工具包

此存储库提供了一个 docker 图像，其中包含 Amazon Keyspaces 的有用开发人员工具 例如，它包括一个包含最佳实践的 CQLSHRC 文件，一个可选的Amazon使用 cqlsh 的身份验证扩展，以及执行常见任务的帮助工具。该工具包针对亚马逊 Keyspaces 进行了优化，但也适用于 Apache Cassandra 集群。

https://github.com/aws-samples/amazon-keyspaces-toolkit.

Amazon Keyspaces（针对 Apache Cassandra）示例

此回购是我们的亚马逊 Keyspaces 示例代码的官方列表。回购按语言细分为几个部分（请参阅示例)。每种语言都有自己的例子部分。这些示例演示了您在构建应用程序时可以使用的常见 Amazon Keyspaces 服务实施和模式。

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

Amazon签名版本 4 (SigV4) 身份验证插件

这些插件使您可以通过使用以下方式管理对 Amazon Keyspaces 的访问Amazon Identity and Access Management(IAM) 用户和角色。

Amazon 开发工具包: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.

突出显示的示例和开发者工具库

下面是针对 Amazon Keyspaces（针对 Apache Cassandra）的有用社区工具。

Amazon CloudFormation针对 Amazon Keyspaces（针对 Apache Cassandra）指标创建 Amazon CloudWatch 控制面板的模板

此存储库提供Amazon CloudFormation用于快速为亚马逊 Keyspaces 设置 CloudWatch 指标的模板。通过使用此模板，您可以通过提供具有常用指标的可部署预构建的 CloudWatch 控制面板，更轻松地开始使用。

https://github.com/aws-samples/amazon-keyspaces-cloudwatch-cloudformation-templates.

将 Amazon Keyspaces（针对 Apache Cassandra）结合使用Amazon Lambda

存储库包含演示如何从 Lambda 连接到 Amazon Keyspaces 的示例。下面是一些示例。

C#/.NET：https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/dotnet/datastax-v3/connection-lambda.

Amazon 开发工具包:https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/java/datastax-v4/connection-lambda.

另一个展示如何从 Python Lambda 部署和使用 Amazon Keyspaces 的 Lambda 示例可从以下仓库获得。

https://github.com/aws-samples/aws-keyspaces-lambda-python

将 Amazon Keyspaces（针对 Apache Cassandra）与 Spring 结合使用

下面是一个示例，向您展示如何将 Amazon Keyspaces 与 Spring Boot 结合使用。

https://github.com/aws-samples/amazon-keyspaces-spring-app-example

将 Amazon Keyspaces（针对 Apache Cassandra）与 Scala 结合使用

这是一个示例，展示了如何使用 Scala 的 Sigv4 身份验证插件连接到 Amazon Keyspaces。

https://github.com/aws-samples/amazon-keyspaces-examples/tree/main/scala/datastax-v4/connection-sigv4

Amazon Keyspaces (for Apache Cassandra) 查询语言 (CQL)Amazon CloudFormation转换器

该软件包实现了一个命令行工具，用于将 Apache Cassandra 查询语言 (CQL) 脚本转换为Amazon CloudFormation(CloudFormation) 模板，它允许在 CloudFormation 堆栈中轻松管理 Amazon Keyspaces 模式。

https://github.com/aws/amazon-keyspaces-cql-to-cfn-converter.

针对 Java 的 Apache Cassandra 驱动程序的 Amazon Keyspaces（针对 Apache Cassandra）的帮助

此存储库包含将 DataSax Java 驱动程序与 Amazon Keyspaces 结合使用时的驱动程序策略、示例和最佳实践（适用于 Apache Cassandra）。

https://github.com/aws-samples/amazon-keyspaces-java-driver-helpers.

Amazon Keyspaces（针对 Apache Cassandra）

此存储库演示了如何压缩、存储和读取/写入大型对象，以提高性能并降低吞吐量和存储成本。

https://github.com/aws-samples/amazon-keyspaces-compression-example.

Amazon Keyspaces（针对 Apache Cassandra）和 Amazon S3 编解码器演示

自定义 Amazon S3 编解码器支持透明、用户可配置的指向 Amazon S3 对象的 UUID 指针映射。

https://github.com/aws-samples/amazon-keyspaces-large-object-s3-demo.

Amazon Keyspaces 中的无服务器资源管理（针对 Apache Cassandra）

Amazon Keyspaces（针对 Apache Cassandra）是无服务器的。Amazon Keyspaces 将存储和读/写吞吐量资源直接分配给表，而不是通过集群中的节点为您的工作负载部署、管理和维护存储和计算资源。

本章提供有关 Amazon Keyspaces 中的无服务器资源管理的详细信息。要了解如何使用 Amazon CloudWatch 监控无服务器资源，请参阅the section called “使用 CloudWatch 进行监控” (p. 180).

主题

Amazon Keyspaces 中的存储

Amazon Keyspaces（针对 Apache Cassandra）会根据表中存储的实际数据自动预置表的存储。您无需预先为表预置存储。Amazon Keyspaces 会在应用程序写入、更新和删除数据时自动扩展表存储空间。与传统的 Apache Cassandra 群集不同，Amazon Keyspaces 不需要额外的存储来支持低级系统操作（如压缩）。您只需为使用的存储付费。

默认情况下，Amazon KKeyspaces 配置密钥空间的复制因子为三。您无法修改复因子。Amazon Keyspaces 在多次中自动复制表数据三次Amazon高可用性的可用区。Amazon Keyspaces 存储的每 GB 价格已包括复制。请参阅Amazon Keyspaces（针对 Apache Cassandra）定价要了解更多信息。

Amazon Keyspaces 会持续监控表的大小，以确定存储费用。有关 Amazon Keyspaces 如何计算数据的可计费大小的更多信息，请参阅the section called “计算行大小” (p. 106).

Amazon Keyspaces 中的读取/写入容量模式

Amazon Keyspaces 具有两个读/写容量模式来处理表的读写：

- 按需（默认）
- 已预置

您选择的读/写容量模式控制对读写吞吐量收费的方式以及管理表吞吐容量的方式。

主题

按需容量模式

Amazon Keyspaces（针对 Apache Cassandra）按需容量模式是一个灵活的计费选项，可以每秒处理数千个请求而不需要进行容量规划。此选项提供 pay-per-request 为读写请求定价，以便您只需为实际使用的资源付费。

当您选择按需模式时，Amazon Keyspaces 可以将表的吞吐容量立即扩展到以前达到的任何流量级别，然后在应用程序流量减少时调整回来。如果工作负载的流量级别达到新的峰值，则该服务会迅速调整以便为您的表增加吞吐容量。您可以为新表和现有表启用按需容量模式。

如果满足以下任意条件，则按需模式是很好的选项：

- 您创建工作负载未知的新表。
- 您具有不可预测的应用程序流量。
- 您更喜欢只为您使用的容量付费。

要开始使用按需模式，您可以使用控制台或使用几行 Cassandra 查询语言 (CQL) 代码创建新表或更新现有表以使用按需容量模式。有关更多信息，请参阅 the section called “表” (p. 209)。

主题

读取请求单位和写入请求单元

对于按需容量模式表，您无需预先指定预期应用程序的读取和写入吞吐量。Amazon Keyspaces 会按照读取请求单位 (RU) 和写入请求单位 (WRU) 对在表上执行的读取和写入操作收费。

- 第一RRU代表一个`LOCAL_QUORUM`阅读请求，或两个 `LOCAL_ONE`读取请求，对大小不超过 4 KB 的行。如果您需要读取大于 4 KB 的行，则读取操作将使用额外的 RU。所需的 RRU 总数取决于行大小，以及要使用 `LOCAL_QUORUM` 还是 `LOCAL_ONE` 读取一致性。例如，使用 `LOCAL_QUORUM` 读取一致性读取一个 8 KB 行需要 2 个 RRU，如果选择 `LOCAL_ONE` 读取一致性，则需要 1 RRU。
- 第一WRU表示对大小不超过 1 KB 的行执行一次写入。所有写入操作都使用 `LOCAL_QUORUM` 一致性，使用轻量级事务 (LWT) 不收取额外费用。如果您需要写入大于 1 KB 的行，则写入操作将使用额外的 WRU。所需 WRU 的总数取决于行大小。例如，如果行大小为 2 KB，则需要 2 个 WRU 才能执行一个写入请求。

有关支持的一致性级别的信息，请参阅the section called “支持的 Cassandra 一致性级别” (p. 12)。

峰值流量和扩展属性

使用按需容量模式的 Amazon Keyspaces 表会自动适应应用程序的流量。按需容量模式会即时在表中承受之前双倍的峰值流量。例如，您的应用程序流量模式可能在每秒 5,000 到 10,000 次 `LOCAL_QUORUM` 读取之间变化，其中每秒 10,000 次读取是以前的流量峰值。

使用这种模式，按需容量模式可即时容纳最高每秒 20,000 次读取的持续流量。如果应用程序承受每秒 20,000 次读取的流量，则该峰值将成为新的之前峰值，从而使后续流量高达每秒 40,000 次读取。

如果您对于表需要的峰值高于之前峰值的两倍，Amazon Keyspaces 会自动分配更多容量作为流量增量。这有助于确保您的表具有足够的吞吐容量来处理额外的请求。但是，如果您在 30 分钟内超过前一个峰值的两倍，则可能会出现吞吐容量不足错误。

例如，假设您的应用程序流量模式在每秒 5,000 到 10,000 个强一致性读取之间变化，而上一次达到的流量峰值为每秒 20,000 次读取。在这种情况下，服务建议您在将流量推动到每秒 40,000 次读取之前，至少将流量增长的时间间隔 30 分钟。

要了解有关账户的默认配额以及如何增加此配额的更多信息，请参阅 配额 (p. 219)。

按需容量模式的初始吞吐量

如果您在启用按需容量模式的情况下创建了新表，或者首次将现有表切换为按需容量模式，则该表将具有以下之前峰值设置，即使该表之前尚未使用按需容量模式提供流量也是如此。

- 具有按需容量模式的新建表：前一个峰值是 2,000 个 WRU 或 6,000 个 RU。您可以立即将以前的峰值翻倍。这样，新创建的按需表可以提供最高 4,000 个 WRU 或 12,000 个RRU，或者两者的任意线性组合。
- 现有表切换为按需容量模式：先前峰值是为该表预置的 WCU 和 RCU 的一半，或者是按需容量模式的新建表的设置，以较高者为准。

预置吞吐容量模式

如果您选择预置的吞吐容量 模式，则指定您的应用程序需要的每秒读取和写入次数。这有助于您管理 Amazon Keyspaces 的使用，以保持在或低于定义的请求速率，以优化价格并保持可预测性。要详细了解预置吞吐量的自动扩展，请参阅the section called “将 Application Auto Scaling 管理吞吐容量” (p. 92)。

如果满足以下任意条件，则预置的吞吐容量模式是很好的选项：

- 您具有可预测的应用程序流量。
- 您运行流量比较稳定或逐渐增加的应用程序。
- 您可以预测容量要求以优化价格。

读取容量单位和写入容量单位

对于预置的吞吐容量模式表，您可以按读取容量单位 (RCU) 和写入容量单位 (WCU) 指定吞吐容量：

- 对于大小不超过 4 KB 的行，一个 RCU 表示每秒一个 `LOCAL_QUORUM` 读取操作或每秒两个 `LOCAL_ONE` 读取操作。如果您需要读取大于 4 KB 的行，则读取操作将使用额外的 RCU。

  所需的 RCU 总数取决于行大小，以及要使用 `LOCAL_QUORUM` 还是 `LOCAL_ONE` 读取。例如，如果您的行大小为 8 KB，则需要 2 个 RCU 来维持每秒一个 `LOCAL_QUORUM` 读取；如果选择 `LOCAL_ONE` 读取，则需要 1 个 RCU。
- 第一WCU表示对大小不超过 1 KB 的行每秒执行一次写入。所有写入操作都使用 `LOCAL_QUORUM` 一致性，使用轻量级事务 (LWT) 不收取额外费用。如果需要写入大于 1 KB 的行，则写入操作将使用额外的 WCU。

  所需 WCU 的总数取决于行大小。例如，如果您的行大小为 2 KB，则您需要 2 个 WCU 来维持每秒一个写入请求。

如果您的应用程序读取或写入较大的行（最大为 1 MB 的 Amazon Keyspaces 行大小上限），它将消耗更多的容量单位。要了解有关如何估计行大小的更多信息，请参阅the section called “计算行大小” (p. 106). 例如，假设您创建了具有 6 个 RCU 和 6 个 WCU 的预置表。使用这些设置，您的应用程序可以执行以下操作：

- 执行`LOCAL_QUORUM`每秒读取高达 24KB（4 KB × 6 个 RCU）。
- 执行最高每秒 48 KB 的 `LOCAL_ONE` 读取（读取吞吐量的两倍）。
- 每秒写入高达 6KB（1 KB × 6 个 WCU）。

预置的吞吐量 是应用程序可以从表消耗的最大吞吐容量。如果您的应用程序超出了预置的吞吐容量，则可能会发现容量不足错误。

例如，没有足够的吞吐容量的读取请求会失败，并出现`Read_Timeout`例外并发布到`ReadThrottleEvents`指标。没有足够的吞吐容量的写入请求失败，并出现`Write_Timeout`例外并发布到`WriteThrottleEvents`指标。

你可以使用亚马逊 CloudWatch 监控您的预配置和实际吞吐量指标以及容量不足事件。有关这些指标的更多信息，请参阅 the section called “指标与维度” (p. 181)。

Note

由于容量不足而重复出现的错误可能会导致客户端驱动程序特定的异常，例如 DataStax Java 驱动程序失败并显示`NoHostAvailableException`.

要更改表的吞吐量容量设置，可以使用Amazon Web Services Management Console或者`ALTER TABLE`语句使用 CQL，有关详细信息，请参阅the section called “ALTER TABLE” (p. 211).

要了解有关账户的默认配额以及如何增加此配额的更多信息，请参阅 配额 (p. 219)。

管理和查看容量模式

您可以在 Amazon Keyspaces 系统密钥空间中查询系统表，以查看有关表的容量模式信息。您还可以查看表使用的是按需吞吐容量模式还是预置吞吐容量模式。如果表配置为预置吞吐量模式，您会看到为表预置的吞吐容量。

示例

```
SELECT * from system_schema_mcs.tables where keyspace_name = 'mykeyspace' and table_name =
 'mytable';
```

使用按需容量模式配置的表返回以下内容。

```
                    {
                        'capacity_mode': {
                            'last_update_to_pay_per_request_timestamp': '1579551547603',
                            'throughput_mode': 'PAY_PER_REQUEST'
                         }
                    }
```

使用预置吞吐容量模式配置的表将返回以下内容。

```
                    {
                        '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` 值以毫秒为单位。

要更改表的预置吞吐容量，请使用 the section called “ALTER TABLE” (p. 211)。

更改容量模式的注意事项

当您将表从预置容量模式切换到按需容量模式时，Amazon Keyspaces 会对表和分区的结构进行若干更改。此过程可能耗时数分钟。在切换期间，您的表将提供与先前预置的 WCU 和 RCU 数量相一致的吞吐量。

当您从按需容量模式切换回预置的容量模式时，表将提供与表设置为按需容量模式时达到的先前峰值一致的吞吐量。

Note

您可以每 24 小时在读/写容量模式之间切换一次。

使用 Application Auto Scaling 管理 Amazon Keyspaces 吞吐容量

许多数据库工作负载本质上是周期性的，或者难以提前进行预测。例如，考虑一个大多数用户在白天处于活跃状态的社交网络应用程序。数据库必须能够处理白天活动，但夜间不需要相同级别的吞吐量。

另一个示例是面临快速采用的新移动游戏应用程序。如果此游戏变得极受欢迎，它可能会超出可用的数据库资源，从而导致性能降低并使客户感到不满。这些类型的工作负载通常需要手动干预来扩展或缩减数据库资源，以便响应不断变化的使用量级别。

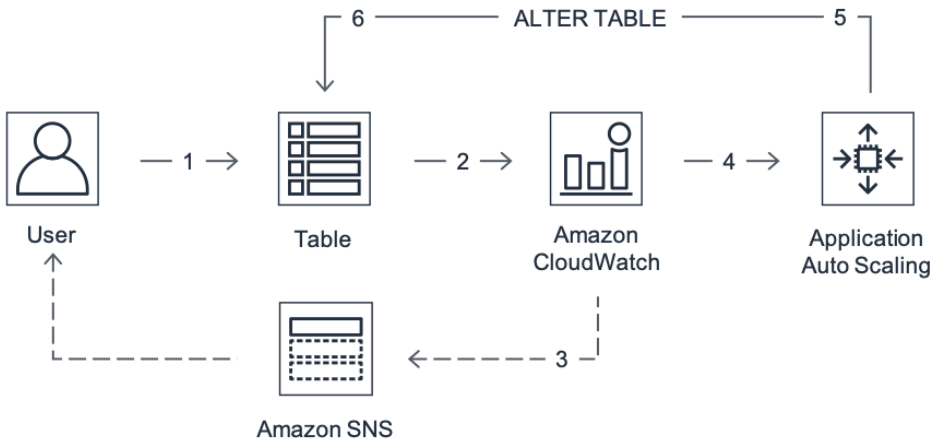
Amazon Keyspaces（Apache Cassandra）将自动调整吞吐容量以响应实际应用程序流量，帮助您高效预置可变工作负载的吞吐容量。Amazon Keyspaces 代表您使用 Application Auto Scaling 服务增加和减少表的读写容量。有关 Application Auto Scaling 的更多信息，请参阅Application Auto Scaling 用户指南.

Note

要快速开始使用 Amazon Keyspaces 自动扩展，请参阅the section called “使用控制台” (p. 93). 您无法使用 Cassandra 查询语言 (CQL) 管理 Amazon KeySpaces 扩展策略。要了解如何通过编程方式管理 Amazon KeySpaces 扩展策略，请参阅。the section called “通过编程方式管理” (p. 96).

Amazon KKeyspaces 自动扩展的工作原理

下图高度概述了 Amazon KeySpaces 自动扩展管理表的吞吐容量的方式。



要为表启用自动扩展，请创建扩展策略。扩展策略指定是要扩展读取容量还是写入容量（或二者），并为表指定最小的和最大的预置容量单位设置。

扩展策略还定义了 目标利用率。目标利用率是在某个时间点上使用的容量单位与预置容量单位的比率（以百分比表示）。自动扩展使用目标跟踪 算法向上或向下调整表的预置吞吐量以响应实际工作负载。这样做的目的是使实际容量利用率保持在目标利用率或接近目标利用率。

您可以为读取和写入容量设置介于 20% 和 90% 之间的自动扩展目标利用率值。默认的目标利用率为 70%。如果您的流量快速变化，并且您希望容量能够尽快开始扩展，则可以将目标利用率设置为较低的百分比。如果您的应用程序流量变化较慢，并且您希望降低吞吐量成本，则也可以将目标利用率设置为较高的百分比。有关扩展策略的更多信息，请参阅Application Auto Scaling 的目标跟踪扩缩策略.

创建扩展策略时，Application Auto Scaling 将创建两对亚马逊。CloudWatch代表您发出警报。每对警报均表示预置和使用的吞吐量设置的上限和下限。当表的实际使用率在一段持续时间内偏离目标使用率时，将触发这些 CloudWatch 警报。要了解亚马逊的更多信息CloudWatch，请参阅亚马逊CloudWatch用户指南.

当其中一个CloudWatch警报已触发，亚马逊 Simple Notification Service (Amazon SNS) 将向您发送通知（如果您已启用通知）。这些区域有：CloudWatch警报随后会调用 Application Auto Scaling 来评估扩展策略。这将向 Amazon Keyspaces 发出 Alter Table 请求，以便视情况向上或向下调整表的预配置容量。要了解有关 Amazon SNS 通知的更多信息，请参阅设置 Amazon SNS 通知.

Amazon Keyspaces 处理 Alter Table 请求，方式是动态增加（或减少）表的预置吞吐容量，使它接近目标利用率。

> Note
>
> 仅当实际工作负载在几分钟的持续时段内保持提高（或降低）时，Amazon Keyspaces 自动扩展才会修改预置吞吐量设置。Application Auto Scaling 目标跟踪算法寻求使目标使用率长期达到或接近选定值。表的内置突增容量将容纳活动的短时间突增峰值。

使用说明

在开始使用 Amazon Keyspaces 自动扩展之前，您应了解以下内容：

- Amazon Keyspaces 自动扩展会根据您的扩展策略在需要时不限次数地增加读取容量或写入容量。所有亚马逊 Keyspaces 配额都将保持有效，如中所述。配额 (p. 219).
- Amazon Keyspaces 自动扩展不会阻止您手动修改预置吞吐量设置。这些手动调整不会影响附加到扩展策略的任何现有 CloudWatch 警报。
- 如果您使用控制台创建预置了吞吐容量的表，则默认情况下将启用 Amazon KeySpaces 自动扩展。您可以随时修改自动扩展设置。有关更多信息，请参阅 the section called “使用控制台” (p. 93)。
- 如果您使用 Amazon CloudFormation 创建扩展策略，则应管理 Amazon CloudFormation 中的扩展策略以使堆栈与堆栈模板同步。如果您从 Amazon KeyScaling 或 Application Auto Scaling 更改扩展策略，则这些策略将被Amazon CloudFormation堆栈重置时的堆栈模板。
- 如果您使用CloudTrail要监控 Amazon Keyspaces 自动扩展，您可能会在配置验证过程中看到应用程 Application Auto Scaling 发出的呼叫的警报。您可以使用`invokedBy`字段，其中包含`application-autoscaling.amazonaws.com`对于这些验证检查。

使用控制台管理 Amazon Keyspaces 自动扩展策略

您可以使用控制台为新表和现有表启用 Amazon Keyspaces 自动扩展。您还可以使用该控制台修改自动扩展设置或禁用自动扩展。

> Note
>
> 有关设置缩减和向外扩展冷却时间等更高级的功能，请使用Amazon Command Line Interface(Amazon CLI) 可通过编程方式管理 Amazon Keyspaces 扩展策略。有关更多信息，请参阅 以编程方式管理 Amazon Keyspaces 扩展策略 (p. 96)。

主题

开始前的准备工作：授予用户对 Amazon Keyspaces 自动扩展的权限

要开始使用，请确认用户具有创建和管理自动扩展设置所需的适当权限。InAmazon Identity and Access Management(IAM)，Amazon管理的策略`AmazonKeyspacesFullAccess`是管理亚马逊 Keyspaces 扩展策略所必需的。

> Important
>
> 需要 `application-autoscaling:*` 权限才能对表禁用自动扩展。在删除表之前，必须使用 the section called “禁用现有表的自动缩放功能：注销可扩展目标” (p. 101) 禁用自动扩展。

要针对 Amazon Keyspace 控制台访问和 Amazon KKeyspaces 自动扩展设置 IAM 用户，请添加以下策略。

要附加**`AmazonKeyspacesFullAccess`**政策

1. 登录 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台控制面板上，选择 Users (用户)，然后从列表中选择您的 IAM 用户。
3. 在 Summary (摘要) 页上，选择 Add permissions (添加权限)。
4. 选择直接附加现有策略。
5. 从策略列表中，选择AmazonKeyspacesFullAccess，然后选择后续：审核。
6. 选择 Add permissions（添加权限）。

创建启用了 Amazon KKeyspaces 自动扩展的新表

> Note
>
> 自动扩展 Amazon Keyspaces 需要存在服务相关角色 (`AWSServiceRoleForApplicationAutoScaling_CassandraTable`) 它代表您执行自动扩展操作。将自动为您创建此角色。有关更多信息，请参阅 the section called “使用服务相关角色” (p. 177)。

创建启用了自动扩展的新表

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 在导航窗格中，选择 Tables (表)，然后选择 Create table (创建表)。
3. 在存储库的创建表中的页面表详细信息部分，选择键空间并提供新表的名称。
4. 在架构部分中，为表创建架构。
5. 在表设置选择部分，选择自定义设置.
6. 继续读取/写入容量设置。
7. 对于 Capacity mode (容量模式)，选择 Provisioned (预置)。
8. 在 Read capacity (读取容量) 部分中，确认已选择 Scale automatically(自动扩展)。

在此步骤中，您将选择表的最小和最大读取容量单位以及目标利用率。

- 最小容量单位— 输入表应始终能够支持的最小吞吐量级别的值。该值必须介于 1 和账户的每秒最大吞吐量配额（默认为 40000）之间。
- 最大容量单位— 输入要为表预置的最大吞吐量。该值必须介于 1 和账户的每秒最大吞吐量配额（默认为 40000）之间。
- 目标利用率— 输入介于 20% 和 90% 之间的目标利用率。当流量超过定义的目标利用率时，容量将自动扩展。当流量低于定义的目标时，容量将自动重新缩减。

Note

要了解有关账户的默认配额以及如何增加此配额的更多信息，请参阅 配额 (p. 219)。

9. 在 Write capacity (写入容量) 部分中，选择上一步中为读取容量定义的相同设置，或手动输入写入容量值。
10. 选择 Create Table。使用指定的自动扩展参数创建表。

Note

控制台中显示的 Cassandra 查询语言 (CQL) 命令不包括启用 Amazon Keyspaces 自动扩展。此操作通过一个单独的 API 调用完成，该调用将表作为可扩展目标注册到应用程序 Auto Scaling。

在现有表上启用 Amazon Keyspaces 自动扩展

Note

自动扩展 Amazon Keyspaces 需要存在服务相关角色 (`AWSServiceRoleForApplicationAutoScaling_CassandraTable`) 它代表您执行自动扩展操作。将自动为您创建此角色。有关更多信息，请参阅 the section called “使用服务相关角色” (p. 177)。

为现有表启用 Amazon Keyspaces 自动扩展

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 选择要处理的表，然后选择更改容量设置。
3. 选择自动扩展，然后按照创建启用了 Amazon KKeyspaces 自动扩展的新表 (p. 94)中的步骤 6 操作来编辑读取和写入容量。
4. 在定义自动扩展设置时，选择 Save (保存)。

Note

控制台中显示的 CQL 命令不包括启用 Amazon Keyspaces 自动扩展。此操作通过一个单独的 API 调用完成，该调用将表作为可扩展目标注册到应用程序 Auto Scaling。

修改或禁用 Amazon Keyspaces 自动扩展设置

您可以使用Amazon Web Services Management Console修改您的 Amazon Keyspaces 自动扩展设置。为此，请转至要编辑的表，然后选择 Change capacity settings (更改容量设置)。可以修改 Read capacity (读取容量) 或 Write capacity (写入容量) 部分中的设置。有关这些设置的更多信息，请参阅 创建启用了 Amazon KKeyspaces 自动扩展的新表 (p. 94)。

要禁用 Amazon Keyspaces 自动扩展，请取消选择。自动扩展. 禁用自动扩展会取消将表注册为可扩展目标。要删除应用程序 Application Auto Scaling 来访问表的服务相关角色，请按照中的步骤操作。??? (p. 178).

Note

要删除 Application Auto Scaling 所使用的服务相关角色，您必须针对所有账户中的所有表禁用自动扩展。Amazon Web Services 区域.

在控制台上查看 Amazon Keyspaces 自动扩展活动

您可以使用亚马逊监控 Amazon KKeyspaces 自动扩展使用资源的方式。CloudWatch它将生成有关您的使用情况和性能的指标。按照中的步骤操作。Application Auto Scaling用户指南要创建CloudWatch控制面板。

以编程方式管理 Amazon Keyspaces 扩展策略

要以编程方式更新和管理 Amazon Keyspaces 自动扩展设置，可以使用Amazon Command Line Interface(Amazon CLI) 或者AmazonAPI。无法使用 Cassandra 查询语言 (CQL) 管理 Amazon Keyspaces 扩展策略。本主题概述了您可以使用 Amazon CLI 通过编程方式管理的自动扩展任务。

有关 Application Auto Scaling 的更多信息Amazon CLI本主题中介绍的命令，请参阅Application Auto Scaling中的Amazon CLI命令参考. 有关使用的更多信息AmazonAPI，请参阅Application Auto Scaling API 参考.

主题

开始前的准备工作

在开始之前，您需要完成以下任务。

配置权限

如果未完成这些任务，您必须为用户配置相应的权限，以创建和管理自动扩展设置。InAmazon Identity and Access Management(IAM)，Amazon管理的策略`AmazonKeyspacesFullAccess`是管理亚马逊 Keyspaces 扩展策略所必需的。有关详细步骤，请参阅the section called “开始前的准备工作：授予用户对 Amazon Keyspaces 自动扩展的权限” (p. 94)。

安装 Amazon CLI

如果您尚未安装和配置 Amazon CLI，则必须先执行此操作。要执行此操作，请转至Amazon Command Line Interface用户指南并按照以下说明操作：

- 安装 Amazon CLI
- 配置 Amazon CLI

对现有表启用自动扩展功能：注册可扩展目标

对于现有的 Amazon Keyspaces 表，您可以将表的写入或读取容量注册为使 Application Auto Scaling 的可扩展目标。这样可允许 Application Auto Scaling 调整您指定的表的预置写入或读取容量。在以下示例中，我们注册mytable作为具有写入容量的可扩展目标，容量范围为 5—10 个容量单位。

Note

Amazon Keyspaces 的自动扩展需要存在服务相关角色 (`AWSServiceRoleForApplicationAutoScaling_CassandraTable`) 它代表您执行自动扩展操作。将自动为您创建此角色。有关更多信息，请参阅 the section called “使用服务相关角色” (p. 177)。

输入以下命令，将表注册为可扩展目标。

```
aws application-autoscaling register-scalable-target \
    --service-namespace cassandra \
    --resource-id "keyspace/mykeyspace/table/mytable" \
    --scalable-dimension "cassandra:table:WriteCapacityUnits" \
    --min-capacity 5 \
    --max-capacity 10
```

查看向 Application Auto Scaling 注册的可扩展

要查看注册详细信息，请使用以下命令。

```
aws application-autoscaling describe-scalable-targets
    --service-namespace cassandra
    --resource-ids "keyspace/mykeyspace/table/mytable"
```

该命令的输出如下所示。

```
{
    "ScalableTargets": [
        {
            "ServiceNamespace": "cassandra",
            "ResourceId": "keyspace/mykeyspace/table/mytable",
            "ScalableDimension": "cassandra:table:WriteCapacityUnits",
            "MinCapacity": 5,
            "MaxCapacity": 10,
            "RoleARN": "arn:aws:iam::012345678910:role/aws-
service-role/cassandra.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_CassandraTable",
            "CreationTime": 1587495787.412,
            "SuspendedState": {
                "DynamicScalingInSuspended": false,
                "DynamicScalingOutSuspended": false,
                "ScheduledScalingSuspended": false
            }
        }
    ]
}
```

查看向 Application Auto Scaling 注册的可扩展

要查看注册详细信息，请使用以下命令。

```
aws application-autoscaling describe-scalable-targets \
    --service-namespace cassandra \
    --resource-id "keyspace\mytable"
```

创建扩展策略

要通过编程方式管理自动扩展设置，请为表创建扩展策略。该策略定义指示 Application Auto Scaling 调整表的预置吞吐量的条件，以及它在执行此操作时需要采取的措施。在此示例中，您将此策略与您以前定义的可扩展目标（mytable 表的写入容量单位）相关联。

该策略包含以下元素：

- `PredefinedMetricSpecification`— 允许 Application Auto Scaling 调整的指标。对于亚马逊 Keyspaces，以下值是的有效值`PredefinedMetricType`：

  - `CassandraReadCapacityUtilization`
  - `CassandraWriteCapacityUtilization`
- `ScaleOutCooldown`— 横向扩展活动会增加您的表的预置吞吐量。要为横向扩展活动增加冷却时间，请为 `ScaleOutCooldown` 指定一个值（以秒为单位）。默认值是 0。有关更多信息，请参阅 。Application Auto Scaling 用户指南中的目标跟踪扩展策略.
- `ScaleInCooldown`— 扩展活动会减小您的表的预置吞吐量。要为缩减活动增加冷却时间，请为 `ScaleInCooldown` 指定一个值（以秒为单位）。默认值是 0。有关更多信息，请参阅 。Application Auto Scaling 用户指南中的目标跟踪扩展策略.
- `TargetValue`— Application Auto Scaling 可确保消耗的容量与预置容量的比例保持在该值或接近该值。您将 `TargetValue` 定义为百分比。

  Note

  为了进一步了解 `TargetValue` 的工作原理，假设您的表的预配置吞吐量设置为 200 个写入容量单位。您决定为此表创建扩展策略，并使用 `TargetValue` 的 70%。
  现在假设您开始将写入流量驱动到表，以便实际写入吞吐量为 150 个容量单位。这些区域有：consumed-to-provisioned现在的比例为 (150/ 200)，即 75%。此比率超出了您的目标值，因此 Application Auto Scaling 会将预置写入容量增加到 215，使该比率为 (150/215) 或 69.77% — 尽可能接近 `TargetValue`，但不超过。

适用于mytable，你设置`TargetValue`至百分之 50。Application Auto Scaling 将在 5—10 个容量单位的范围内调整表的预置吞吐量（请参阅）对现有表启用自动扩展功能：注册可扩展目标 (p. 96)) 以便 consumed-to-provisioned比率保持在或接近 50%。您可以将 `ScaleOutCooldown` 和 `ScaleInCooldown` 值设置为 60 秒。

1. 创建一个文件，在其中包含要应用于表的策略，如以下示例所示。然后，在本示例中，使用名称 `scaling-policy.json` 保存文件。

```
{
    "PredefinedMetricSpecification": {
        "PredefinedMetricType": "CassandraWriteCapacityUtilization"
    },
    "ScaleOutCooldown": 60,
    "ScaleInCooldown": 60,
    "TargetValue": 50.0
}
```

2. 使用以下 Amazon CLI 命令创建策略：

```
aws application-autoscaling put-scaling-policy \
     --service-namespace cassandra \
     --resource-id "keyspace/mykeyspace/table/mytable" \
     --scalable-dimension "cassandra:table:WriteCapacityUnits" \
     --policy-name "MyScalingPolicy" \
     --policy-type "TargetTrackingScaling" \
     --target-tracking-scaling-policy-configuration file://scaling-policy.json
```

在此命令的输出中，您可以看到CloudWatchApplication Auto Scaling 创建的警报，分别针对消耗的上限和下限，分别针对扩展目标范围的上限和下限。

```
{
    "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:8d606c33-2078-4f37-8305-36e89c56a779:resource/cassandra/
keyspace/mykeyspace/table/mytable:policyName/MyScalingPolicy",
    "Alarms": [
```

```
        {
            "AlarmName": "TargetTracking-keyspace/mykeyspace/table/mytable-AlarmHigh-
d421fec6-fa82-44b4-aab6-6a9bfb6f0ced",
            "AlarmARN": "arn:aws:cloudwatch:us-west-2:012345678910:alarm:TargetTracking-
keyspace/mykeyspace/table/mytable-AlarmHigh-d421fec6-fa82-44b4-aab6-6a9bfb6f0ced"
        },
        {
            "AlarmName": "TargetTracking-keyspace/mykeyspace/table/mytable-
AlarmLow-04479372-e50b-4652-a06d-b3055744ae23",
            "AlarmARN": "arn:aws:cloudwatch:us-west-2:012345678910:alarm:TargetTracking-
keyspace/mykeyspace/table/mytable-AlarmLow-04479372-e50b-4652-a06d-b3055744ae23"
        },
        {
            "AlarmName": "TargetTracking-keyspace/mykeyspace/table/mytable-
MCSProvisionedCapacityHigh-c6a26783-837e-4f70-919e-1fbe4362b6ab",
            "AlarmARN": "arn:aws:cloudwatch:us-west-2:012345678910:alarm:TargetTracking-
keyspace/mykeyspace/table/mytable-MCSProvisionedCapacityHigh-
c6a26783-837e-4f70-919e-1fbe4362b6ab"
        },
        {
            "AlarmName": "TargetTracking-keyspace/mykeyspace/table/mytable-
MCSProvisionedCapacityLow-e1ce121a-48ea-4148-ace2-25c5f854c215",
            "AlarmARN": "arn:aws:cloudwatch:us-west-2:012345678910:alarm:TargetTracking-
keyspace/mykeyspace/table/mytable-MCSProvisionedCapacityLow-e1ce121a-48ea-4148-
ace2-25c5f854c215"
        }
    ]
}
```

查看扩展策略

您可以使用以下 Amazon CLI 命令，查看现有扩展策略的详细信息：

```
aws application-autoscaling describe-scaling-policies \
    --service-namespace cassandra \
    --resource-ids "keyspace/mykeyspace/table/mytable" \
    --policy-name "MyScalingPolicy"
```

在此命令的输出中，您可以看到扩展策略的详细信息以及CloudWatchApplication Auto Scaling 创建的警报，分别针对消耗的上限和下限，分别针对扩展目标范围的上限和下限。

```
{
    "ScalingPolicies": [
        {
            "PolicyARN": "arn:aws:autoscaling:us-
west-2:012345678910:scalingPolicy:8d606c33-2078-4f37-8305-36e89c56a779:resource/cassandra/
keyspace/mykeyspace/table/mytable:policyName/MyScalingPolicy",
            "PolicyName": "MyScalingPolicy",
            "ServiceNamespace": "cassandra",
            "ResourceId": "keyspace/mykeyspace/table/mytable",
            "ScalableDimension": "cassandra:table:WriteCapacityUnits",
            "PolicyType": "TargetTrackingScaling",
            "TargetTrackingScalingPolicyConfiguration": {
                "TargetValue": 50.0,
                "PredefinedMetricSpecification": {
                    "PredefinedMetricType": "CassandraWriteCapacityUtilization"
                },
                "ScaleOutCooldown": 60,
                "ScaleInCooldown": 60
            },
            "Alarms": [
                {
```

```
                    "AlarmName": "TargetTracking-keyspace/mykeyspace/table/mytable-
AlarmHigh-d421fec6-fa82-44b4-aab6-6a9bfb6f0ced",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-keyspace/mykeyspace/table/mytable-AlarmHigh-
d421fec6-fa82-44b4-aab6-6a9bfb6f0ced"
                },
                {
                    "AlarmName": "TargetTracking-keyspace/mykeyspace/table/mytable-
AlarmLow-04479372-e50b-4652-a06d-b3055744ae23",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-keyspace/mykeyspace/table/mytable-
AlarmLow-04479372-e50b-4652-a06d-b3055744ae23"
                },
                {
                    "AlarmName": "TargetTracking-keyspace/mykeyspace/table/mytable-
MCSProvisionedCapacityHigh-c6a26783-837e-4f70-919e-1fbe4362b6ab",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-keyspace/mykeyspace/table/mytable-
MCSProvisionedCapacityHigh-c6a26783-837e-4f70-919e-1fbe4362b6ab"
                },
                {
                    "AlarmName": "TargetTracking-keyspace/mykeyspace/table/mytable-
MCSProvisionedCapacityLow-e1ce121a-48ea-4148-ace2-25c5f854c215",
                    "AlarmARN": "arn:aws:cloudwatch:us-
west-2:012345678910:alarm:TargetTracking-keyspace/mykeyspace/table/mytable-
MCSProvisionedCapacityLow-e1ce121a-48ea-4148-ace2-25c5f854c215"
                }
            ],
            "CreationTime": 1587497009.591
        }
    ]
}
```

查看 Application Auto Scaling 操作

您可以通过以下方式查看代表您启动的 Application Auto Scaling 操作：Amazon CLI命令。

```
aws application-autoscaling describe-scaling-activities \
    --service-namespace cassandra
```

如果您对某个表运行此命令，而该表上的自动扩展功能正在更改写入容量，您会看到类似下面这样的输出：

```
...

{
    "ScalableDimension": "cassandra:table:WriteCapacityUnits",
    "Description": "Setting write capacity units to 10.",
    "ResourceId": "keyspace/mykeyspace/table/mytable",
    "ActivityId": "0cc6fb03-2a7c-4b51-b67f-217224c6b656",
    "StartTime": 1489088210.175,
    "ServiceNamespace": "cassandra",
    "EndTime": 1489088246.85,
    "Cause": "monitor alarm AutoScaling-keyspace/mykeyspace/table/mytable-
AlarmHigh-1bb3c8db-1b97-4353-baf1-4def76f4e1b9 in state ALARM triggered policy
 MyScalingPolicy",
    "StatusMessage": "Successfully set write capacity units to 10. Change successfully
 fulfilled by cassandra.",
    "StatusCode": "Successful"
},
...
```

此示例输出表明 Application Auto Scaling 已向发布`Alter Table`请求亚马逊 Keyspaces 更改写入容量。

删除扩展策略

删除 mytable 的扩展策略。如果不再需要扩展表的写入容量，则应考虑删除您的扩展策略，以便 Amazon Keyspaces 不会继续修改您的表的写入容量设置。您可以通过注销可扩展目标来删除扩展基础设施，也可以仅删除您的扩展策略并保持可扩展目标的注册状态，以便日后使用。

使用以下命令可删除指定的目标跟踪扩展策略。它还会删除CloudWatchApplication Auto Scaling 代表您创建的警报。

```
aws application-autoscaling delete-scaling-policy \
    --service-namespace cassandra \
    --resource-id "keyspace/mykeyspace/table/mytable" \
    --scalable-dimension "cassandra:table:WriteCapacityUnits" \
    --policy-name "MyScalingPolicy"
```

禁用现有表的自动缩放功能：注销可扩展目标

使用以下 Amazon CLI 命令来注销可扩展目标。这还将删除附加到此目标的扩展策略。

```
aws application-autoscaling deregister-scalable-target \
    --service-namespace cassandra \
    --resource-id "keyspace/mykeyspace/table/mytable" \
    --scalable-dimension "cassandra:table:WriteCapacityUnits"
```

高效使用亚马逊 Keyspaces 突增容量

Amazon Keyspaces 为每个分区吞吐量调配提供一定的灵活性容量爆增. 当您未完全使用分区的吞吐量时，Amazon Keyspaces 将为稍后保留一部分未使用的容量以供稍后使用。爆发处理使用率高峰的吞吐量。

当前，亚马逊 Keyspaces 保留最多五分钟（300 秒）未使用的读取和写入容量。在读取或写入活动偶尔突增期间，这些额外的容量单元会被快速消耗-甚至比您为表定义的每秒预置的吞吐容量还快。

Amazon Keyspaces 还可能在不事先通知的情况下，将暴增容量用于后台维护和其他任务。

请注意，这些暴增容量详细信息未来可能发生变化。

使用 Amazon Keyspaces（针对 Apache Cassandra）中的键空间、表和行

本章提供了有关在 Amazon Keyspaces（针对 Apache Cassandra）中使用键空间、表、行等的详细信息。要了解如何使用 Amazon CloudWatch 监控键空间和表，请参阅。the section called “使用 CloudWatch 进行监控” (p. 180).

主题

在亚马逊密 Keyspaces 中使用密钥空间

本节提供有关在 Amazon Keyspaces（针对 Apache Cassandra）中使用键空间的详细信息。

主题

在亚马逊 Keyspaces 中创建键空间

异步执行 Keyspaces 据定义语言 (DDL) 操作，例如创建和删除键空间。您可以在Amazon Web Services Management Console，它将指示键空间处于挂起状态或活动状态的时间。还可以使用系统架构表以编程方式监控新密钥空间的创建状态。一旦键空间可供使用，它就会在系统架构中变得可见。检查新密钥空间何时可供使用的推荐设计模式是轮询 Amazon Keyspaces 系统架构表 (system_schema_mcs.*)。有关键空间的 DDL 语句列表，请参阅the section called “Keyspaces” (p. 207)CQL 语言参考中的部分。

以下查询显示键空间是否已成功创建。

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

对于已成功创建的键空间，查询的输出如下所示。

```
keyspace_name | durable_writes  | replication
--------------+-----------------+--------------
   mykeyspace | true            |{...} 1 item
```

在亚马逊 Keyspaces 中使用表

本节提供有关使用 Amazon Keyspaces (Apache Cassandra) 中表的详细信息。

主题

在亚马逊 Keyspaces 中创建表

Amazon Keyspaces 以异步方式执行数据定义语言 (DDL) 操作，如创建和删除表。您可以在Amazon Web Services Management Console，它指示表处于挂起状态或活动状态的时间。还可以使用系统架构表以编程方式监控新表的创建状态。

当系统架构可供使用时，表将显示为活动状态。检查新表何时可供使用的推荐设计模式是轮询 Amazon Keyspaces 系统架构表 (`system_schema_mcs.*`)。有关表的 DDL 语句列表，请参阅the section called “表” (p. 209)CQL 语言参考中的部分。

以下查询显示表的状态。

```
SELECT keyspace_name, table_name, status FROM system_schema_mcs.tables WHERE keyspace_name
 = 'mykeyspace' AND table_name = 'mytable';
```

对于仍在创建且处于挂起状态的表，查询的输出如下所示。

```
keyspace_name | table_name | status
--------------+------------+--------
   mykeyspace |    mytable | CREATING
```

对于已成功创建并处于活动状态的表，查询的输出如下所示。

```
keyspace_name | table_name | status
--------------+------------+--------
   mykeyspace |    mytable | ACTIVE
```

亚马逊 Keyspaces 中的静态列

当您将 Amazon Keyspaces 表中的某列声明为静态列时，存储在此列中的值将在逻辑分区中的所有行之间共享。当您更新此列的值时，Amazon Keyspaces 会自动将更改应用于分区中的所有行。

本节介绍如何在写入静态列时计算编码的数据大小。此过程与将数据写入行中的非静态列的过程分开处理。除了静态数据的大小配额外，静态列的读取和写入操作还独立影响表的计量和吞吐量容量。

计算 Amazon Keyspaces 中每个逻辑分区的静态列大小

本节提供有关如何估计 Amazon Keyspaces 中静态列的编码大小的详细信息。当您计算账单和配额使用时，将使用编码的大小。在计算表的预配置吞吐量容量需求时，还应使用编码的大小。要计算 Amazon Keyspaces 中静态列的编码大小，您可以使用以下准则。

- 分区键可包含最多 2048 字节的数据。分区键中的每个键列最多需要 3 个字节的元数据。这些元数据字节计入每个分区 1 MB 的静态数据大小配额。计算静态数据的大小时，应假设每个分区键列都使用完整的 3 个字节的元数据。
- 根据数据类型使用静态列数据值的原始大小。有关数据类型的更多信息，请参阅 the section called “数据类型” (p. 204)。
- 为元数据的静态数据大小添加 104 个字节。
- 聚类列和常规的非主键列不计入静态数据的大小。要了解如何估计行内非静态数据的大小，请参阅the section called “计算行大小” (p. 106).

静态列的总编码大小根据以下公式计算：

```
partition key columns + static columns + metadata = total encoded size of static data
```

考虑以下表示例，其中所有列均为整型。该表有两个分区键列、两个聚类列、一个常规列和一个静态列。

```
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));
```

在此示例中，我们计算以下语句的静态数据的大小：

```
INSERT INTO mykeyspace.mytable (pk_col1, pk_col2, static_col1) values(1,2,6);
```

要估计此写入操作所需的总字节数，可以使用以下步骤。

1. 通过添加存储在列中的数据类型的字节和元数据字节来计算分区键列的大小。对所有分区键列重复此操作。

   a. 计算分区键 (pk_col1) 第一列的大小：

   ```
   4 bytes for the integer data type + 3 bytes for partition key metadata = 7 bytes
   ```

   b. 计算分区键 (pk_col2) 第二列的大小：

   ```
   4 bytes for the integer data type + 3 bytes for partition key metadata = 7 bytes
   ```

   c. 添加两列以获取分区键列的总估计大小：

   ```
   7 bytes + 7 bytes = 14 bytes for the partition key columns
   ```

2. 添加静态列的大小。在本例中，我们只有一个存储整数的静态列（需要 4 个字节）。
3. 最后，要获得静态列数据的总编码大小，请将主键列和静态列的字节加起来，然后为元数据添加额外的 104 个字节：

   ```
   14 bytes for the partition key columns + 4 bytes for the static column + 104 bytes for
    metadata = 122 bytes.
   ```

您还可以使用相同的语句更新静态和非静态数据。要估计写操作的总大小，必须首先计算非静态数据更新的大小。然后计算行更新的大小，如示例所示the section called “计算行大小” (p. 106)，然后添加结果。

在这种情况下，您总共可以写入 2 MB—1 MB 表示最大行大小配额，1 MB 是每个逻辑分区的最大静态数据大小的配额。

要计算同一语句中静态和非静态数据更新的总大小，可以使用以下公式：

```
(partition key columns + static columns + metadata = total encoded size of static data)
 + (partition key columns + clustering columns + regular columns + row metadata = total
 encoded size of row)
= total encoded size of data written
```

考虑以下表示例，其中所有列均为整型。该表有两个分区键列、两个聚类列、一个常规列和一个静态列。

```
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));
```

在此示例中，我们在向表中写入一行时计算数据的大小，如以下语句所示：

```
INSERT INTO mykeyspace.mytable (pk_col1, pk_col2, ck_col1, ck_col2, reg_col1, static_col1)
 values(2,3,4,5,6,7);
```

要估计此写入操作所需的总字节数，可以使用以下步骤。

1. 如前所示，计算静态数据的编码总大小。在本例中，它是 122 个字节。
2. 按照中的步骤，根据非静态数据的更新添加行的编码总大小的大小the section called “计算行大小” (p. 106). 在此示例中，行更新的总大小为 134 个字节。

   ```
   122 bytes for static data + 134 bytes for nonstatic data = 256 bytes.
   ```

计量亚马逊Keyspaces 中静态数据的读/写操作

静态数据与 Cassandra 中的逻辑分区相关联，而不是单个行。通过跨越多个物理存储分区，Amazon Keyspaces 中的逻辑分区可以实际上解除大小的绑定。因此，Amazon Keyspaces 分别计量静态和非静态数据的写入操作。此外，包含静态和非静态数据的写入需要额外的底层操作才能提供数据一致性。

如果对静态数据和非静态数据执行混合写入操作，则会导致两个单独的写入操作 — 一个用于非静态数据，另一个用于静态数据。这适用于按需和预配置的读/写容量模式。

以下示例提供有关如何在计算具有静态列的 Amazon Keyspaces 表的预置吞吐容量要求时，如何估计所需的读取容量单位 (RCU) 和写入容量单位 (WCU) 的详细信息。您可以使用以下公式估计表处理包含静态和非静态数据的写入操作所需的容量：

```
2 x WCUs required for nonstatic data + 2 x WCUs required for static data
```

例如，如果应用程序每秒写入 27 kB 的数据，并且每次写入包括 25.5 kB 的非静态数据和 1.5 kB 的静态数据，那么您的表需要 56 个 WCU（2 x 26 wCU + 2 x 2 wCU）。

Amazon Keyspaces 将静态和非静态数据的读取量与对多行的读取量相同。因此，在同一操作中读取静态和非静态数据的价格取决于为执行读取而处理的数据的总大小。

要了解如何使用 Amazon CloudWatch 监控无服务器资源，请参阅the section called “使用 CloudWatch 进行监控” (p. 180).

使用亚马逊 Keyspaces 中的行

本节提供有关处理 Amazon Keyspaces（适用于 Apache Cassandra）中行的详细信息。表是 Amazon Keyspaces 中的主要数据结构，表中的数据按列和行组织。

主题

计算亚马逊 Keyspaces 中的行大小

Amazon Keyspaces（适用于 Apache Cassandra）提供完全托管的存储，提供了一位数毫秒的读写性能，并且可以跨多个持久存储数据Amazon可用区。Amazon Keyspaces 将元数据附加到所有行和主键列，以支持高效的数据访问和高可用性。

本节提供有关如何估计 Amazon Keyspaces 中行的编码大小的详细信息。计算账单和配额使用时使用编码的行大小。在计算表的预配置吞吐量容量要求时，还应使用编码的行大小。要计算 Amazon Keyspaces 中的编码行大小，您可以使用以下准则。

- 分区键可包含最多 2048 字节的数据。分区键中的每个键列最多需要 3 个字节的元数据。这些元数据字节计入您的 1 MB 行大小配额中。计算行的大小时，应假设每个分区键列都使用完整的 3 个字节的元数据。
- 每行最多可以有 850 字节的聚类列数据，每个聚类列最多需要 4 个字节的元数据。这些元数据字节计入您的 1 MB 行大小配额中。计算行的大小时，应假设每个聚类列都使用完整的 4 个字节的元数据。
- 对于常规、非静态、非主键列，请根据数据类型使用单元格数据的原始大小。有关数据类型的更多信息，请参阅 the section called “数据类型” (p. 204)。
- 静态列数据不计入 1 MB 的最大行大小。要计算静态列的数据大小，请参阅the section called “计算每个逻辑分区的静态列大小” (p. 103).
- 为行元数据的每行大小添加 100 个字节。

编码数据行的总大小根据以下公式计算：

```
partition key columns + clustering columns + regular columns + row metadata = total encoded
 size of row
```

考虑以下表示例，其中所有列均为整型。该表有两个分区键列、两个聚类列和一个常规列。

```
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));
```

在此示例中，我们计算向表中写入一行时的数据大小，如以下语句所示：

```
INSERT INTO mykeyspace.mytable (pk_col1, pk_col2, ck_col1, ck_col2, reg_col1)
 values(1,2,3,4,5);
```

要估计此写入操作所需的总字节数，可以使用以下步骤。

1. 通过添加存储在列中的数据类型的字节和元数据字节来计算分区键列的大小。对所有分区键列重复此操作。

   a. 计算分区键的第一列 (pk_col1) 的大小：

   ```
   4 bytes for the integer data type + 3 bytes for partition key metadata = 7 bytes
   ```

   b. 计算分区键 (pk_col2) 的第二列的大小：

   ```
   4 bytes for the integer data type + 3 bytes for partition key metadata = 7 bytes
   ```

   c. 添加两列以获取分区键列的总估计大小：

```
7 bytes + 7 bytes = 14 bytes for the partition key columns
```

2. 通过添加存储在列中的数据类型的字节和元数据字节来计算聚类列的大小。对所有聚类列重复此操作。

   a. 计算聚类列的第一列 (ck_col1) 的大小：

   ```
   4 bytes for the integer data type + 4 bytes for clustering column metadata = 8
    bytes
   ```

   b. 计算聚类列的第二列 (ck_col2) 的大小：

   ```
   4 bytes for the integer data type + 4 bytes for clustering column metadata = 8
    bytes
   ```

   c. 添加两列以获取聚类列的总估计大小：

   ```
   8 bytes + 8 bytes = 16 bytes for the clustering columns
   ```

3. 添加常规列的大小。在这个例子中，我们只有一个存储整数的列，它需要 4 个字节。
4. 最后，要获得编码行的总大小，请将所有列的字节加起来，然后为行元数据添加额外的 100 个字节：

```
14 bytes for the partition key columns + 16 bytes for clustering columns + 4 bytes for
 the regular column + 100 bytes for row metadata = 134 bytes.
```

要了解如何使用 Amazon CloudWatch 监控无服务器资源，请参阅the section called “使用 CloudWatch 进行监控” (p. 180).

在亚马逊 Keyspaces 中处理查询

本节介绍使用 Amazon Keyspaces（Apache Cassandra）中的查询。可用于查询、转换和管理数据的 CQL 语句包括SELECT、INSERT、UPDATE, 和DELETE. 以下主题概述了处理查询时可用的一些更复杂的选项。有关带示例的完整语言语法，请参阅the section called “DML 语句” (p. 214).

主题

在亚马逊 Keyspaces 中订购结果

这些区域有：ORDER BY子句用于指定在SELECT网页。语句接受列名列表作为参数，并且可以为每列指定数据的排序顺序。您只能在排序子句中指定聚类列，不允许使用非聚类列。

返回结果的两个可用的排序顺序选项是ASC用于升序和DESC用于降序排序顺序。如果您没有在查询语句中指定排序顺序，则使用聚类列的默认顺序。

您可以在排序子句中使用的可能排序顺序取决于在创建表时分配给每个聚类列的排序顺序。查询结果只能按照创建表时为所有聚类列定义的顺序或与定义的排序顺序相反的顺序对查询结果进行排序。不允许使用其他可能的组合。

例如，如果表的聚类顺序为（col1 ASC、col2 DESC、col3 ASC），则 ORDER BY 的有效参数为（col1 ASC、col2 DESC、col3 ASC）或（col1 DESC、col2 ASC、col3 DESC）。有关集群顺序的更多信息，请参阅table_options下the section called “CREATE TABLE” (p. 209).

在亚马逊 Keyspaces 中对结果进行分页

自动使用 Amazon Keyspaces分页来自的结果SELECT读取数据以处理SELECT语句超过 1 MB。使用分页，SELECT语句结果将分成若干“页”大小为 1 MB（或更小）的数据。应用程序可以先处理第一页结果，然后处理第二页结果，依此类推。客户在执行时应始终检查分页令牌SELECT返回多个行的查询。

如果客户端提供的页面大小需要读取超过 1 MB 的数据，Amazon Keyspaces 会根据 1 MB 的数据读取增量自动将结果划分为多个页面。

例如，如果行的平均大小为 100 KB，并且您指定页面大小为 20，则 Amazon Keyspaces 在读取 10 行（读取 1000 KB 的数据）后自动分页数据。

由于 Amazon Keyspaces 根据为处理请求而读取的行数而不是结果集中返回的行数对结果进行分页，因此如果您运行过滤的查询，某些页面可能不包含任何行。

例如，如果您将 PAGE SIZE 设置为 10 并且 Keyspaces 会评估 30 行以处理SELECT查询，亚马逊 Keyspaces 将返回三页。如果只有一部分行与您的查询匹配，则某些页面的行可能少于 10 行。

在亚马逊 Keyspaces 中与分区员合作

在 Apache Cassandra 中，分区员控制群集中存储哪些节点数据。分区人使用分区键的哈希值创建数字令牌。Cassandra 使用此令牌跨节点分配数据。客户也可以在SELECT运算和WHERE子句来优化读取和写入操作。例如，通过在每个 parallel 行作业中指定要查询的不同令牌范围，客户端可以高效地对大型表执 parallel 查询。

使用亚马逊 Keyspaces，您可以选择将账户的分区器设置为 CassandraRandomPartitioner以便与 Apache Spark Cassandra 连接器等开发人员工具兼容。亚马逊 Keyspaces 还提供了DefaultPartitioner，它返回相同TOKEN函数结果为RandomPartitioner.

您可以随时安全地更改账户级别的分区员。更改分区程序设置时，您无需重新加载 Amazon Keyspaces 数据。客户端下次连接时将自动使用新的分区程序设置。

您可以使用更改分区器。Amazon Web Services Management Console或 Cassandra 查询语言 (CQL)。

Amazon Web Services Management Console

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 在导航窗格中，选择 Configuration（配置）。
3. 在存储库的配置页面，转到编辑分区器.
4. Selectorg.apache.cassandra.dht.dht.随机分区器.

   Note

   您必须断开连接并重新连接到 Amazon Keyspaces 才能请求使用新的分区程序。

Cassandra Query Language (CQL)

1. 要查看为该帐户配置的分区程序，您可以使用以下查询。

   ```
   SELECT partitioner from system.local;
   ```

   如果分区程序尚未更改，则查询具有以下输出。

   ```
   partitioner
   ```

```
-------------------------------------------
com.amazonaws.cassandra.DefaultPartitioner
```

2. 将分区程序更新为RandomPartitioner，您可以使用以下查询。

```
UPDATE system.local set partitioner='org.apache.cassandra.dht.RandomPartitioner'
 where key='local';
```

3. 要确认分区程序已设置，您可以运行SELECT再次查询。请注意，由于读取的最终一致性，响应反映的可能不是刚刚刚完成的分区程序更改的结果。如果你重复SELECT在短时间后再次运行，响应将返回最新的数据。

```
SELECT partitioner from system.local;
```

Note

您必须断开连接并重新连接到 Amazon Keyspaces，以便请求使用新的分区程序。

Amazon Keyspaces (针对 Apache Cassandra) 中的数据建模

本主题介绍 Amazon Keyspaces (针对 Apache Cassandra) 中的数据建模概念。使用本节可以查找有关设计符合应用程序数据访问模式的数据模型的建议。使用 Amazon Keyspaces 时，实施数据建模最佳实践可提高性能并最大限度地降低吞吐量成本。

主题

如何在亚马逊密钥空间中有效地使用分 Keyspaces

唯一标识 Amazon Keyspaces 表中每一行的主键可以包括一个或多个分区键列，这些列决定了数据存储在哪些分区，以及一个或多个可选的群集列，它们定义了数据在分区中的聚类和排序方式。

由于分区键确定了数据存储在的分区数以及数据在这些分区之间的分布方式，因此您选择分区密钥的方式可能会对查询的性能产生重大影响。一般而言，您应针对跨磁盘上所有分区的统一活动来设计应用程序。

在所有分区之间均匀分配应用程序的读写活动有助于最大限度地降低吞吐量成本，这适用于按需模式和预配置的读/写容量模式。例如，如果您使用预配置容量模式，您可以确定应用程序所需的访问模式，并估计每个表所需的读取容量单位 (RCU) 和写入容量单位 (WCU) 总计。Amazon Keyspaces 支持使用预置吞吐量的访问模式，前提是针对给定分区的流量不超过 3000 RCU 或 1000 WCU。

Amazon Keyspaces 提供容量爆增，为每个分区吞吐量调配提供额外的灵活性。the section called “容量爆增” (p. 101).

主题

使用写入分片在 Amazon Keyspaces 中均匀分配工作负载

扩大空间是一种在 Amazon Keyspaces 中跨分区更好地分发写入的一种方式。可以通过多种不同方式来进行。您可以添加一个额外的分区键列，您可以在其中写入随机数字以在分区之间分配行。或者，可以使用基于查询内容计算的数字。

使用复合分区键和随机值进行分片

在分区之间更均匀分配负载的一种策略是添加一个额外的分区键列，将随机数写入。然后在更大空间内随机写入。

例如，请考虑下表，其中有一个分区键表示日期。

```
CREATE TABLE IF NOT EXISTS tracker.blogs (
   publish_date date,
   title text,
   description int,
```

```
   PRIMARY KEY (publish_date));
```

为了在分区之间更均匀地分布此表，您可以添加一个附加的分区键列shard存储随机数。例如：

```
CREATE TABLE IF NOT EXISTS tracker.blogs (
   publish_date date,
   shard int,
   title text,
   description int,
   PRIMARY KEY ((publish_date, shard)));
```

插入数据时，你可以选择一个随机数1和200(对于)shardcolumn. 这会产生复合分区键值，如(2020-07-09, 1)、(2020-07-09, 2)，以此类推，直到(2020-07-09, 200). 由于随机化分区键，每天表的写入将均匀分布在多个分区。这样可以提高并行度和总体吞吐量。

但是，要读取指定日期的所有行，必须查询所有分片的行，然后合并结果。例如，您应先发布SELECT分区键值的语句(2020-07-09, 1). 然后发布另一个SELECT对于语句(2020-07-09, 2)，以此类推，直到(2020-07-09, 200). 最后，应用程序必须合并所有这些结果。SELECT语句。

使用复合分区键和计算值进行分片

随机化策略可以显著改善写入吞吐量，但难以读取特定行，因为不知道写入的是哪个值。shard写入该行的时候的列。要使各个行的读取变得简单，可使用其他策略。不是使用随机数在分区之间分配行，而是使用可根据要查询的内容计算得出的数字。

考虑上一个示例，表在分区键中使用当天日期。现在假设每一行都有一个可访问的title列，并且除日期外，最经常需要按标题查找行。在应用程序将行写入表之前，它可根据标题计算得出一个哈希值并使用此值填充shardcolumn. 计算可能产生一个介于 1 和 200 之间非常均匀分布的数字，类似于随机策略所生成的数字。

简单的计算可能已足够，例如标题中字符的 UTF-8 码点值的积，模 200，+ 1。然后，复合分区键值将是日期和计算结果的组合。

利用此策略，写入均匀分布在分区键值之间，从而均匀分布在物理分区之间。您可以轻松执行SELECT特殊行和日期的语句，因为可以为特定行计算出分区键值title值。

要读取给定日期的所有行，你仍然必须SELECT每个(2020-07-09, N)钥匙（在哪里N是 1—200），应用程序之后必须合并所有结果。好处是避免出现单个“热门”分区键值，承担所有工作负载。

将亚马逊 Keyspaces 与 Apache Spark 集成

Apache Spark 是用于大规模数据分析的开源引擎。Apache Spark 使您能够更高效地对存储在 Amazon Keyspaces 中的数据执行分析。您还可以使用 Amazon Keyspaces 为应用程序提供一致、 single-digit-millisecond 从 Spark 读取对分析数据的访问权限。开源的 Spark Cassandra 连接器简化了 Amazon Keyspaces 和 Spark 之间的数据的读写。

Amazon KKeyspaces 对 Spark Cassandra Connector 的支持通过使用完全托管的无服务器数据库服务，简化了在基于 Spark 的分析管道中运行 Cassandra 工作负载的流程。使用 Amazon Keyspaces，您将无需担心 Spark 竞争与表相同的基础设施资源。Amazon Keyspaces 表将根据您的应用程序流量自动扩展和缩减。

以下教程将向您介绍使用 Spark Cassandra 连接器向 Amazon Keyspaces 读取和写入数据所需的步骤和最佳实践。本教程演示了如何通过使用 Spark Cassandra 连接器从文件中加载数据并将数据写入 Amazon Keyspace 表，将数据迁移到 Amazon Keyspace。然后，本教程展示了如何使用 Spark Cassandra 连接器从亚马逊 Keyspaces 读回数据。您可以这样做是为了在基于 Spark 的分析管道中运行 Cassandra 工作负载。

主题

使用 Spark Cassandra 连接器建立与亚马逊 Keyspaces 的连接的先决条件

在使用 Spark Cassandra 连接器连接到亚马逊 Keyspaces 之前，您需要确保已安装以下内容。亚马逊 Keyspaces 与 Spark Cassandra 连接器的兼容性已通过以下推荐版本进行了测试：

- Java 版本 8
- Scala 2.11、2.12
- Spark 2.4
- Cassandra 2.5 及更高版本
- 卡桑德拉驱动程序 4.12

1. 要安装 Scala，请按照https://www.scala-lang.org/download/scala2.html.
2. 要安装 Spark 3.1.2，请按照此示例进行操作。

```
curl -o spark-3.1.2-bin-hadoop3.tgz -k https://dlcdn.apache.org/spark/spark-3.1.2/
spark-3.1.2-bin-hadoop3.2.tgz

# now to untar
tar -zxvf spark-3.1.2-bin-hadoop3.tgz
```

```
# set this variable.
export SPARK_HOME=$PWD/spark-3.1.2-bin-hadoop3.2
```

第 1 步：配置亚马逊 Keyspaces 以与 Apache Cassandra Spark 连接器集成

在此步骤中，您确认账户的分区程序与 Apache Spark Connector 兼容，并设置所需的 IAM 权限。以下最佳做法可帮助您为表配置足够的读/写容量。

1. 为您的帐户选择一个与 Spark Cassandra 连接器兼容的分区员。有关分区员以及如何更改分区的更多信息，请参阅the section called “与分区员合作” (p. 108).
2. 使用接口 VPC 终端节点和 Apache Spark 设置 Amazon Keyspaces 的 IAM 权限。
   - 分配对用户表的读/写访问权限和对系统表的读取访问权限，如以下所示的 IAM 策略示例所示。
   - 对于使用 Spark 通过访问 Amazon Keyspaces 的客户端，需要使用可用的接口 VPC 终端节点填充 system.peer 表VPC 终端节点.

```
{
   "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. 考虑以下最佳实践，为 Amazon Keyspaces 表配置足够的读/写吞吐量，以支持来自 Spark Cassandra 连接器的流量。
   - 开始使用按需容量来帮助您测试场景。
   - 要优化生产环境的表吞吐量成本，请对来自连接器的流量使用速率限制器，并将表配置为使用自动扩展的预配置容量。有关更多信息，请参阅 the section called “将 Application Auto Scaling 管理吞吐容量” (p. 92)。
   - 你可以使用卡桑德拉驱动程序附带的固定费率限制器。有一些为亚马逊 Keyspaces 量身定制的速率限制器中的Amazon样本存储库。
   - 有关容量管理的更多信息，请参阅the section called “读/写容量模式” (p. 88).

第 2 步：配置 Apache Cassandra Spark 连接器

Apache Spark 是一个通用的计算平台，你可以用不同的方式进行配置。要配置 Spark 和 Spark Cassandra 连接器以与 Amazon Keyspaces 集成，我们建议您从下一节中描述的最低配置设置开始，然后根据您的工作负载在稍后增加这些设置。

- 创建小于 8 MB 的 Spark 分区大小。

  在 Spark 中，分区表示可以 parallel 行运行的原子数据块。当您使用 Spark Cassandra 连接器将数据写入 Amazon Keyspaces 时，Spark 分区越小，任务将写入的记录量就越小。如果 Spark 任务遇到多个错误，则在指定的重试次数用完之后，它将失败。为避免重播大型任务和重新处理大量数据，请保持小 Spark 分区的大小。
- 每个执行者使用较少的并发写入次数，并进行大量重试。

  Amazon Keyspaces 将容量不足错误返回给 Cassandra 驱动程序，因为操作超时。您无法通过更改配置的超时持续时间来解决容量不足导致的超时问题，因为 Spark Cassandra Connector 尝试使用`MultipleRetryPolicy`. 为了确保重试次数不会超过驱动程序的连接池，请使用较少的并发写入次数，并进行大量重试次数。以下代码段是一个示例。

```
spark.cassandra.query.retry.count = 500
spark.cassandra.output.concurrent.writes = 3
```

- 细分总吞吐量并将其分配到多个 Cassandra 会话之间。
  - Cassandra Spark 连接器为每个 Spark 执行者创建一个会话。将此会话视为确定所需吞吐量和所需连接数的规模单位。
  - 在定义每个执行者的内核数量和每个任务的内核数时，请从低开始并根据需要增加。
  - 将 Spark 任务失败设置为允许在出现暂时错误时进行处理。熟悉应用程序的流量特征和要求之后，我们建议设置`spark.task.maxFailures`转换为有限的值。
  - 例如，以下配置可以在每个会话中处理每个执行者两个并发任务：

```
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
```

- 关闭批处理。
  - 我们建议您关闭批处理以改善随机访问模式。以下代码段是一个示例。

```
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)
```

- Set**`SPARK_LOCAL_DIRS`**转换为具有足够空间的快速本地磁盘。
  - 默认情况下，Spark 将地图输出文件和弹性分布式数据集 (RDD) 保存到`/tmp` folder。根据您的 Spark 主机的配置，这可能会导致设备上没有剩余空间样式错误。
  - 要设置`SPARK_LOCAL_DIRS`环境变量到名为的目录`/example/spark-dir`，您可以使用以下命令。

```
export SPARK_LOCAL_DIRS=/example/spark-dir
```

第 3 步：创建应用程序配置文件

要将开源 Spark Cassandra 连接器与 Amazon Keyspaces 结合使用，您需要提供一个应用程序配置文件，其中包含连接 DataStax Java 驱动程序。您可以使用特定于服务的凭据或 Sigv4 插件进行连接。

如果您尚未执行此操作，则需要将 Starfield 数字证书转换为 TrustStore 文件。你可以按照详细的步骤操作the section called “开始前的准备工作” (p. 29)来自 Java 驱动程序连接教程。请注意 TrustStore 文件路径和密码，因为您在创建应用程序配置文件时需要此信息。

Connect Sigv4 身份验证

本节向您介绍一个示例`application.conf`连接时可以使用的文件Amazon凭据和 sigv4 插件。如果您尚未生成 IAM 访问密钥（访问密钥 ID 和秘密访问密钥），然后将其保存在您的Amazon配置文件或作为环境变量。有关详细说明，请参阅the section called “的必需证书Amazon身份验证” (p. 17)。

在以下示例中，替换 TrustStore 文件的文件路径，然后替换密码。

```
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 = 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
            }
        metadata = {
        schema {
                token-map.enabled = true
               }
        }
   }
}
```

更新此配置文件并将其另存为`/home/user1/application.conf`. 以下示例使用此路径。

Connect 服务特定凭证联系

本节向您介绍一个示例`application.conf`使用特定于服务的凭据进行连接时可以使用的文件。如果您尚未执行此操作，则需要为 Amazon Keyspaces 生成服务特定凭证。有关详细说明，请参阅the section called “服务特定凭证” (p. 16)。

在以下示例中：`username`和`password`使用您自己的凭证。此外，请替换 TrustStore 文件的文件路径，然后替换密码。

```
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
                }
            }
        }
}
```

更新此配置文件并将其另存为/home/user1/application.conf与代码示例一起使用。

使用固定费率 Connect

要强制每个 Spark 执行器固定费率，您可以定义请求限制器。此请求限制器限制了每秒请求的速率。Spark Cassandra 连接器为每位执行者部署一个 Cassandra 会话。使用以下公式可以帮助您实现对表的一致吞吐量。

```
max-request-per-second * numberOfExecutors = total throughput against a table
```

您可以将此示例添加到之前创建的应用程序配置文件中。

```
datastax-java-driver {
  advanced.throttler {
    class = RateLimitingRequestThrottler

    max-requests-per-second = 3000
    max-queue-size = 30000
    drain-interval = 1 millisecond
  }
}
```

第 4 步：在 Amazon Keyspaces 中准备源数据和目标表

在此步骤中，您将创建一个包含示例数据和 Amazon Keyspaces 表的源文件。

1. 创建源文件。您可以选择以下选项之一：
   - 在本教程中，您将使用带有逗号分隔值 (CSV) 文件的名称keyspaces_sample_table.csv作为数据迁移的源文件。提供的示例文件包含名为名称的表的几行数据book_awards.
     - 下载示例 CSV 文件 (keyspaces_sample_table.csv) 包含在以下存档文件中samplemigration.zip. 将存档解压缩并记下到的路径keyspaces_sample_table.csv.

- 如果您想跟随自己的 CSV 文件将数据写入 Amazon Keyspaces，请确保数据是随机化的。直接从数据库读取或导出到平面文件的数据通常按分区和主键排序。将有序的数据导入 Amazon Keyspaces 可能会导致将其写入 Amazon Keyspaces 分区的较小部分，从而导致流量分布不均衡。这可能会导致性能降低和更高的错误率。

  相比之下，随机化数据有助于利用 Amazon Keyspaces 的内置负载平衡功能，方法是更均匀地跨分区分配流量。您可以使用各种工具来随机化数据。对于使用开源工具的示例舒夫，请参阅the section called “第 2 步：准备数据” (p. 68)在数据迁移教程中。下面是一个示例，展示了如何将数据作为DataFrame.

```
import org.apache.spark.sql.functions.randval
shuffledDF = dataframe.orderBy(rand())
```

2. 在亚马逊密钥空间中创建目标 Keyspaces 间和表格。

   a. 使用 Connect 到亚马逊 Keyspacescqlsh，然后将以下示例中的服务终端节点、用户名和密码替换为您自己的值。

```
cqlsh cassandra.us-east-2.amazonaws.com 9142 -u "111122223333" -p "wJalrXUtnFEMI/
K7MDENG/bPxRfiCYEXAMPLEKEY" --ssl
```

   b. 使用名称创建新的密钥空间catalog如以下示例所示。

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

   c. 在新密钥空间状态为可用后，请使用以下代码创建目标表book_awards. 要了解有关异步资源创建以及如何检查资源是否可用的详细信息，请参阅the section called “CREATE 键空间” (p. 102).

```
CREATE TABLE catalog.book_awards (
   year int,
   award text,
   rank int,
   category text,
   book_title text,
   author text,
   publisher text,
   PRIMARY KEY ((year, award), category, rank)
   );
```

第 5 步：使用 Apache Cassandra Spark 连接器写入和读取亚马逊 Keyspaces 数据

在此步骤中，您首先将示例文件中的数据加载到DataFrame使用火花卡桑德拉连接器。接下来，你从DataFrame进入您的亚马逊 Keyspaces 表中。例如，您还可以单独使用此部分将数据迁移到 Amazon Keyspaces 表中。最后，您将表格中的数据读入DataFrame使用 Spark Cassandra 连接器。例如，您还可以独立使用此部分，从 Amazon Keyspaces 表中读取数据，以便使用 Apache Spark 执行数据分析。

1. 启动 Spark Shell，如以下示例所示。请注意，此示例使用的是 Sigv4 身份验证。

```
./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. 使用以下代码导入 Spark Cassandra 连接器。

```
import org.apache.spark.sql.cassandra._
```

3. 从 CSV 文件中读取数据并将其存储在DataFrame，您可以使用以下代码示例。

```
var df =
 spark.read.option("header","true").option("inferSchema","true").csv("keyspaces_sample_table.csv")
```

您可以使用以下命令显示结果。

```
scala> df.show();
```

输出应该类似于此输出。

```
+----------------+----+-----------+----+------------------+--------------------
+-------------+
|           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|
+----------------+----+-----------+----+------------------+--------------------
+-------------+
```

您可以在DataFrame如以下示例所示。

```
scala> df.printSchema
```

输出应如下所示。

```
root
|-- award: string (nullable = true)
|-- year: integer (nullable = true)
|-- category: string (nullable = true)
|-- rank: integer (nullable = true)
|-- author: string (nullable = true)
|-- book_title: string (nullable = true)
|-- publisher: string (nullable = true)
```

4. 使用以下命令将数据写入到DataFrame转到亚马逊 Keyspaces 表格。

```
df.write.cassandraFormat("book_awards", "catalog").mode("APPEND").save()
```

5. 要确认数据已保存，您可以将其读回数据框，如以下示例所示。

```
var newDf = spark.read.cassandraFormat("book_awards", "catalog").load()
```

然后你可以显示现在包含在数据框中的数据。

```
scala> newDf.show()
```

该命令的输出应如下所示。

```
+--------------------+-----------------+---------------+-----------+-------------
+----+----+
|          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|
+--------------------+-----------------+---------------+-----------+-------------
+----+----+
```

解决将 Spark Cassandra 连接器与亚马逊 Keyspaces 结合使用时的常见错

你可以使用亚马逊 CloudWatch 帮助您排查与 Amazon Keyspaces 中 Spark Cassandra 连接器配置相关的问题的指标。要了解有关将 Amazon Keyspaces 与 CloudWatch 结合使用的更多信息，请the section called “使用 CloudWatch 进行监控” (p. 180).

本节介绍了在使用 Spark Cassandra 连接器时需要观察的最有用的指标。

已超过每个连接请求的速率

亚马逊 Keyspaces 的配额为每个连接每秒 3,000 个请求。每个 Spark 执行者都建立了与亚马逊 Keyspaces 的连接。运行多次重试可能会耗尽每个连接请求的费率配额。如果您超过此配额，亚马逊 Keyspaces 会发出`PerConnectionRequestRateExceeded`CloudWatch 中的指标。

如果你明白 PerConnectionRequestRateExceeded 与其他系统或用户错误一起存在的事件，Spark 可能会在每个连接分配的请求数之外运行多次重试。

如果你明白`PerConnectionRequestRateExceeded`没有其他错误的事件，那么您可能需要增加驱动程序设置中的连接数以允许更多吞吐量，或者您可能需要增加 Spark 作业中的执行者数量。

超出存储分区吞吐量

Amazon Keyspaces 的配额为每秒 1,000 个 WCU 或 WRU /每秒 3,000 个 RCU 或 RU，每个分区每秒 3,000 个 RU 或 RU。如果您看到`StoragePartitionThroughputCapacityExceeded` CloudWatch 事件，它可能表明数据在加载时没有随机化。有关如何随机播放数据的示例，请参阅the section called “第 4 步：准备源数据和目标表” (p. 116).

常见错误和警告

如果您使用的是 Amazon Virtual Private Cloud 并连接到 Amazon Keyspaces，则 Cassandra 驱动程序可能会在`system.peers`表。有关更多信息，请参阅 the section called “常见错误和警告” (p. 199)。您可以放心地忽略此警告。

Amazon Keyspaces (a) 时间点恢复（针对 Apache Cassandra）

通过提供表数据的持续备份，时间点恢复 (PITR) 有助于保护 Amazon Keyspaces 表免遭意外写入或删除操作。

例如，假设测试脚本意外写入生产 Amazon Keyspaces 表中。与 point-in-time 恢复，您可以将该表中的数据还原到最近 35 天内启用 PITR 以来此表中的任何时间点。如果您删除表 point-in-time 启用恢复，您可以查询已删除的表中的数据 35 天（无需额外费用），并将其还原到删除点之前此表所处的状态。

您可以使用控制台将 Amazon Keyspaces 表还原到某个时间点。Amazon开发工具包和Amazon Command Line Interface(Amazon CLI) 或 Cassandra 查询语言 (CQL)。有关更多信息，请参阅 将 Amazon Keyspaces 表还原到某个时间点 (p. 127)。

时间点操作对基表没有性能或可用性影响，恢复表不会占用额外的吞吐量。

有关 PITR 配额的信息，请参阅配额 (p. 219).

有关定价的信息，请参阅亚马逊 Keyspaces（针对 Apache Cassandra）定价.

主题

如何 point-in-time 恢复在亚马逊 Keyspaces 中有效

本部分提供 Amazon Keyspaces 的概述。 point-in-time 恢复 (PITR) 有效。有关定价的更多信息，请参阅Amazon Keyspaces（针对 Apache Cassandra）定价.

主题

启用 point-in-time 恢复（PITR）

您可以使用控制台启用 PITR，也可以以编程方式启用它。

使用控制台启用 PITR

新表的 PITR 设置可以在自定义设置选项。默认情况下，在通过控制台创建的新表上启用 PITR。

要为现有表启用 PITR，请完成以下步骤。

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 在导航窗格中，选择和。表然后选择要编辑的表。
3. 在存储库的备份选项卡上，选择编辑.
4. 在编辑 point-in-time 恢复设置设部分，选择启用时间点恢复.

您可以随时通过以下步骤禁用表上的 PITR。

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 在导航窗格中，选择和。表然后选择要编辑的表。
3. 在存储库的备份选项卡上，选择编辑.
4. 在编辑 point-in-time 恢复设置设部分中，清除启用时间点恢复”复选框。

Important

禁用 PITR 会立即删除备份历史记录，即使您在 35 天内在表上重新启用 PITR 也是如此。

要了解如何使用控制台还原表，请参阅the section called “将表还原到某个时间点 (控制台)” (p. 127).

使用启用 PITRAmazon CLI

您可以使用`UpdateTable`API。

使用创建新表时使用Amazon CLI在创建新表时，您必须显式启用 PITR。

要在创建新表时启用 PITR，可以使用以下命令：Amazon CLI以命令为例。为了提高可读性，该命令已分成单独的行。

```
aws keyspaces create-table --keyspace-name 'myKeyspace' --table-name 'myTable'
            --schema-definition 'allColumns=[{name=id,type=int},{name=name,type=text},
{name=date,type=timestamp}],partitionKeys=[{name=id}]'
            --point-in-time-recovery 'status=ENABLED'
```

Note

如果没有 point-in-time 已指定恢复值，默认情况下将禁用时间点恢复。

要确认 point-in-time 表的恢复设置，您可以使用以下内容：Amazon CLI命令。

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

要使用为现有表启用 PITRAmazon CLI中，运行以下命令。

```
aws keyspaces update-table --keyspace-name 'myKeyspace' --table-name 'myTable' --point-in-
time-recovery 'status=ENABLED'
```

要在现有表上禁用 PITR，请运行以下命令：Amazon CLI命令。

```
aws keyspaces update-table --keyspace-name 'myKeyspace' --table-name 'myTable' --point-in-
time-recovery 'status=DISABLED'
```

Important

禁用 PITR 会立即删除备份历史记录，即使您在 35 天内在表上重新启用 PITR 也是如此。

使用 CQL 启用 PITR

您可以使用`point_in_time_recovery`自定义属性。

使用 CQL 创建新表时，必须在创建新表时显式启用 PITR。

要在创建新表时启用 PITR，可以使用以下 CQL 命令作为示例。

```
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

如果没有 point-in-time 指定了恢复自定义属性，默认情况下将禁用时间点恢复。

要使用 CQL 为现有表启用 PITR，请运行以下 CQL 命令。

```
ALTER TABLE mykeyspace.mytable
WITH custom_properties = {'point_in_time_recovery': {'status': 'enabled'}}
```

要在现有表中禁用 PITR，请运行以下 CQL 命令。

```
ALTER TABLE mykeyspace.mytable
WITH custom_properties = {'point_in_time_recovery': {'status': 'disabled'}}
```

Important

禁用 PITR 会立即删除备份历史记录，即使您在 35 天内在表上重新启用 PITR 也是如此。

有关 CQL 语言参考中的更多信息，请参阅the section called “CREATE TABLE” (p. 209)和the section called “ALTER TABLE” (p. 211). 要了解如何使用 CQL 还原表，请参阅the section called “使用 CQL 将表还原到某个时间点” (p. 129).

还原表所需的权限

要成功还原表，IAM 用户或角色需要以下最低权限：

- `cassandra:Restore`— 还原目标表需要执行还原操作。
- `cassandra:Select`— 从源表中读取需要选择操作。
- `cassandra:TagResource`— 标签操作是可选的，只有在还原操作添加标签时才需要执行此操作。

下面是一个策略示例，该策略向用户授予还原密钥空间中表所需的最低权限。`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*"
         ]
      }
   ]
}
```

根据其他选定的功能，可能需要额外的权限才能恢复表。例如，如果源表使用客户托管密钥进行静态加密，则 Amazon Keyspaces 必须具有访问源表的客户托管密钥的权限才能成功还原表。有关更多信息，请参阅the section called “PITR 和加密表” (p. 126)。

如果您使用 IAM 策略条件键要将传入流量限制到特定来源，您必须确保 Amazon Keyspaces 有权代表您的委托人执行还原操作。您必须添加`aws:ViaAWSService`如果您的策略将传入流量限制为以下任何一项，则为 IAM 策略的条件密钥：

- 带有 VPC 终端节点`aws:SourceVpce`
- IP 范围与`aws:SourceIp`
- 带有`aws:SourceVpc`

这些区域有：`aws:ViaAWSService`条件密钥允许在任何时候访问Amazon服务使用委托人的凭证发出请求。有关更多信息，请参阅 。IAM JSON 策略元素：条件键中的IAM 用户指南.

以下是一个策略示例，该策略将源流量限制为特定 IP 地址，并允许 Amazon Keyspaces 代表委托人还原表。

```
{
   "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"
            }
```

```
        }
      }
   ]
}
```

有关使用的示例策略`aws:ViaAWSService`全局条件键，请参阅the section called “VPC 终端节点和 Amazon Keyspaces 时间点恢复 (PITR)” (p. 199).

PITR 连续备份的时间窗口

Amazon Keyspaces PITR 使用两个时间戳来维护表可恢复备份的时间范围。

- 最早可恢复时间 — 标记最早可恢复备份的时间。最早的可恢复备份最长可以备份 35 天，或者在启用 PITR 时（以较新的时间为准）。无法修改最长 35 天的备份窗口。
- 当前时间 — 最新可还原备份的时间戳为当前时间。如果在还原期间没有提供时间戳，则使用当前时间。

启用 PITR 后，您可以还原到之间的任何时间点`EarliestRestorableDateTime`和`CurrentTime`. 您只能将表数据还原到启用 PITR 的时间。

如果禁用 PITR 然后再次重新启用它，则将第一个可用备份的开始时间重置为重新启用 PITR 的时间。这意味着禁用 PITR 会删除备份历史记录。

> **Note**
>
> 表的数据定义语言 (DDL) 操作（如架构更改）均以异步方式执行。您只能在还原的表数据中看到已完成的操作，但是如果在还原时正在进行，则可能会看到对源表执行的其他操作。有关 DDL 语句列表，请参阅the section called “DDL 语句” (p. 207).

还原表不必处于活动状态。如果已删除的表启用了 PITR，并且删除是在备份窗口内（或过去 35 天内）发生的，也可以恢复已删除的表。

> **Note**
>
> 如果创建的新表与之前删除的表具有相同的限定名称（例如 mykeyspace.mytable），则删除的表将不再可恢复。如果尝试从控制台执行此操作，将显示警告。

PITR 还原设置

当您使用 PITR 还原表时，Amazon Keyspaces 会根据所选时间戳将源表的架构和数据恢复到状态（`day:hour:minute:second`）到新表上。PITR 不会覆盖现有表。

除了表的架构和数据外，PITR 还原`custom_properties`从源表中。与表的数据（基于最早还原时间和当前时间之间的选定时间戳还原）不同，自定义属性始终会根据表的设置在当前时间内恢复。

还原表的设置将源表的设置与启动恢复的时间戳相匹配。如果要在还原过程中覆盖这些设置，可以使用`WITH custom_properties`. 自定义属性包括以下设置。

- 读/写容量模式
- 配置吞吐量容量设置
- PITR 设置设置

执行完整表还原时，还原表的所有表设置都来自还原时源表的当前设置。

例如，假设一个表的预配置的吞吐量最近下降到 50 个读取容量单位和 50 个写入容量单位。然后，你将表的状态恢复到三周前。目前，其预配置吞吐量为 100 个读取容量单位和 100 个写入容量单位。在此情况下，Amazon Keyspaces 将表数据还原到该时间点，但使用当前预配置吞吐量设置（50 个读取容量单位和 50 个写入容量单位）。

以下设置不会恢复，您必须为新表手动配置它们。

- 自动扩展策略（适用于使用预配置容量模式的表）
- Amazon Identity and Access Management (IAM) 策略
- 亚马逊 CloudWatch 指标和警报
- 可添加到 CQL 标签（可添加到 CQL）`RESTORE`语句使用`WITH TAGS`）

PITR 恢复加密表

当您使用 PITR 还原表时，Amazon Keyspaces 会恢复源表的加密设置。如果表格是用Amazon 拥有的密钥（默认值），表将自动使用相同的设置恢复。如果要恢复的表是使用客户托管密钥加密的，则 Amazon Keyspace 需要可以访问相同的客户托管密钥才能恢复表数据。

您可以在恢复时更改表的加密设置。从更改Amazon 拥有的密钥对于客户管理的密钥，您需要在恢复时提供有效且可访问的客户托管密钥。

如果你想从客户管理的密钥更改为Amazon 拥有的密钥中，确认 Amazon Keyspaces 有权访问源表的客户管理密钥以使用Amazon 拥有的密钥. 有关表静态加密设置的更多信息，请参阅the section called “工作方式” (p. 146).

> Note
>
> 如果由于 Amazon Keyspaces 无法访问您的客户托管密钥而删除表，则在尝试恢复表之前，您需要确保 Amazon Keyspaces 可以访问客户托管密钥。如果 Amazon Keyspace 无法访问该密钥，则无法恢复使用客户托管密钥加密的表。有关更多信息，请参阅 。密钥访问故障排除中的Amazon Key Management Service开发人员指南 的第一个版本。

使用 PITR 恢复表时间

还原表所用的时间取决于多个因素，并不总是直接与表的大小相关。

以下是还原时的一些注意事项。

- 将备份还原到新表。执行用于创建新表和启动还原流程的所有操作可能最多需要 20 分钟（即使表是空的）。
- 具有分布良好的数据模型的大型表的恢复时间可能是几个小时或更长时间。
- 如果您的源表中包含的数据严重倾斜，则还原用时可能会增加。例如，如果表的主键使用一年中的月份作为分区键，而您的所有数据均来自 12 月，那么就出现了倾斜数据。

规划灾难恢复的最佳做法是定期记录平均还原完成时间，并确定这些时间对整个恢复时间目标的影响。

亚马逊 Keyspaces PITR 和与Amazon服务

使用记录以下 PITR 操作Amazon CloudTrail以实现持续监控和审计。

- 在启用或禁用 PITR 的情况下创建新表。
- 在现有表上启用或禁用 PITR。
- 还原活动表或已删除的表。

有关更多信息，请参阅 使用 记录 Amazon Keyspaces API 调用Amazon CloudTrail (p. 190)。

可以使用以下方法执行以下 PITR 操作：Amazon CloudFormation.

- 在启用或禁用 PITR 的情况下创建新表。

- 在现有表上启用或禁用 PITR。

有关更多信息，请参阅 。Cassandra 资源类型参考中的Amazon CloudFormation用户指南.

将 Amazon Keyspaces 表还原到某个时间点

Amazon Keyspaces（针对 Apache Cassandra） point-in-time 使用恢复 (PITR)，您可以将 Amazon Keyspaces 表还原到最近 35 天中的任何时间点。本教程的第一部分将向您展示如何使用 Amazon Keyspaces 控制台将表还原到某个时间点。Amazon Command Line Interface(Amazon CLI) 和 Cassandra 查询语言 (CQL)。第二部分向您展示了如何使用Amazon CLI和 CQL。

主题

开始前的准备工作

如果未完成这些任务，您必须为用户配置相应的权限，以还原 Amazon Keyspaces 表。InAmazon Identity and Access Management(IAM)，Amazon管理的策略`AmazonKeyspacesFullAccess`包括恢复 Amazon Keyspaces 表的权限。有关实施具有最低所需权限的策略的详细步骤，请参阅the section called “还原权限” (p. 123).

将表还原到某个时间点 (控制台)

以下示例演示如何使用 Amazon Keyspaces 控制台还原名为的现有表。`mytable`到某个时间点。

> Note
>
> 此过程假设您已启用 point-in-time 恢复。启用 PITR`mytable`表中的步骤，请按照中的步骤the section called “使用控制台” (p. 121).

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 在控制台左侧的导航窗格中，选择 Tables (表)。
3. 在表的列表中，选择 `mytable` 表。
4. 在存储库的备份的选项卡`mytable`表格，在时间点恢复部分，选择还原.
5. 对于新表名称，输入 **`mytable_restored`**。
6. 要定义还原操作的时间点，您可以选择两个选项：
   - 选择预配置最早时间。
   - Select指定日期和时间然后输入要将新表还原到的日期和时间。

     > Note
     >
     > 您可以还原到内部的任何时间点最早时间和当前时间。Amazon Keyspaces 将您的表数据还原到基于选定的日期和时间的状态（日:time: time: min: min: second)。

7. 选择还原以启动还原过程。

   正在还原的表显示状态为 Restoring (正在还原)。还原过程完成后，mytable_restored 表的状态更改为 Active (活动)。

   Important

   正在还原时，请勿修改或删除授予 IAM 实体（例如，用户、组或角色）执行还原的权限的 Amazon Identity and Access Management (IAM) 策略。否则，可能会出现意外行为。例如，假设您在还原表时删除了对该表的写入权限。在这种情况下，底层 RestoreTableToPointInTime 操作将无法向表中写入任何还原的数据。您只能在还原操作完成之后修改或删除权限。

使用将表还原到某个时间点Amazon CLI

以下过程演示如何使用 Amazon CLI 将名为 myTable 的现有表还原到某个时间点。

1. 在第一步中，您将创建一个名为的简单表。myTable已启用 PITR。为了便于阅读，该命令分成单独的行。

```
aws keyspaces create-table --keyspace-name 'myKeyspace' --table-name 'myTable'
            --schema-definition 'allColumns=[{name=id,type=int},{name=name,type=text},
{name=date,type=timestamp}],partitionKeys=[{name=id}]'
            --point-in-time-recovery 'status=ENABLED'
```

2. 确认新表的属性并查看earliestRestorableTimestamp对于 PITR。

```
aws keyspaces get-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

   此命令的输出将返回以下内容。

```
{
    "keyspaceName": "myKeyspace",
    "tableName": "myTable",
    "resourceArn": "arn:aws:cassandra:us-east-1:111222333444:/keyspace/myKeyspace/
table/myTable",
    "creationTimestamp": "2022-06-20T14:34:57.049000-07:00",
    "status": "ACTIVE",
    "schemaDefinition": {
        "allColumns": [
            {
                "name": "id",
                "type": "int"
            },
            {
                "name": "date",
                "type": "timestamp"
            },
            {
                "name": "name",
                "type": "text"
            }
        ],
        "partitionKeys": [
            {
                "name": "id"
            }
        ],
        "clusteringKeys": [],
```

```
        "staticColumns": []
    },
    "capacitySpecification": {
        "throughputMode": "PAY_PER_REQUEST",
        "lastUpdateToPayPerRequestTimestamp": "2022-06-20T14:34:57.049000-07:00"
    },
    "encryptionSpecification": {
        "type": "AWS_OWNED_KMS_KEY"
    },
    "pointInTimeRecovery": {
        "status": "ENABLED",
        "earliestRestorableTimestamp": "2022-06-20T14:35:13.693000-07:00"
    },
    "defaultTimeToLive": 0,
    "comment": {
        "message": ""
    }
}
```

您可以将活动表还原到任何 point-in-time 介于`earliestRestorableTimestamp`以及每秒钟内的当前时间。当前时间是默认值。

3. 要将表还原到某个时间点，请指定`restore_timestamp`采用 ISO 8601 格式。您可以在每秒间隔内选择最近 35 天中的任何时间点。例如，以下命令使表还原到 `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"
```

此命令的输出将返回还原表的 ARN。

```
{
    "restoredTableARN": "arn:aws:cassandra:us-east-1:111222333444:/keyspace/myKeyspace/
table/myTable_restored"
}
```

要将表恢复到当前时间，可以省略`restore-timestamp`.

```
aws keyspaces restore-table --source-keyspace-name 'myKeyspace' --source-table-name
 'myTable' --target-keyspace-name 'myKeyspace' --target-table-name 'myTable_restored1'"
```

Important

正在还原时，请勿修改或删除授予 IAM 实体（例如，用户、组或角色）执行还原的权限的 Amazon Identity and Access Management (IAM) 策略。否则，可能会出现意外行为。例如，假设您在还原表时删除了对该表的写入权限。在这种情况下，底层 `RestoreTableToPointInTime` 操作将无法向表中写入任何还原的数据。
您只能在还原操作完成之后修改或删除权限。

使用 CQL 将表还原到某个时间点

以下过程演示如何使用 CQL 还原名为的现有表。`mytable`到某个时间点。

Note

此过程假设您已启用 point-in-time 恢复。要在桌面上启用 PITR，请执行中的步骤the section called “CQL” (p. 123).

1. 您可以将活动表还原到 point-in-time 之间`earliest_restorable_timestamp`以及当前时间。当前时间是默认值。

   为了确认 point-in-time 已启用恢复`mytable`表中，查询`system_schema_mcs.tables`如下所示。

   ```
   SELECT custom_properties
   FROM system_schema_mcs.tables
   WHERE keyspace_name = 'mykeyspace' AND table_name = 'mytable';
   ```

   如下面的示例输出所示，启用了时间点恢复。

   ```
   custom_properties
   -----------------
   {
     ...,
       "point_in_time_recovery": {
       "earliest_restorable_timestamp":"2020-06-30T19:19:21.175Z"
       "status":"enabled"
     }
   }
   ```

2. 将表还原到某个时间点，由`restore_timestamp`采用 ISO 8601 格式。在这种情况下，`mytable`表将恢复到当前时间。您可以省略`WITH restore_timestamp = ...`子句。如果没有该子句，则使用当前时间戳。

   ```
   RESTORE TABLE mykeyspace.mytable_restored
   FROM TABLE mykeyspace.mytable;
   ```

   您还可以还原到特定时间点。您可以指定最近 35 天内的任何时间点。例如，以下命令使表还原到`EarliestRestorableDateTime`。

   ```
   RESTORE TABLE mykeyspace.mytable_restored
   FROM TABLE mykeyspace.mytable
   WITH restore_timestamp = '2020-06-30T19:19:21.175Z';
   ```

   有关完整的语法说明，请参阅the section called “还原表” (p. 213)在语言参考中。

要验证表的还原是否成功，请查询`system_schema_mcs.tables`以确认表格的状态。

```
SELECT status
FROM system_schema_mcs.tables
WHERE keyspace_name = 'mykeyspace' AND table_name = 'mytable_restored'
```

此查询将显示以下输出。

```
status
------
RESTORING
```

正在还原的表显示状态为 Restoring (正在还原)。还原过程完成后，`mytable_restored` 表的状态更改为 Active (活动)。

Important

正在还原时，请勿修改或删除授予 IAM 实体（例如，用户、组或角色）执行还原的权限的 Amazon Identity and Access Management (IAM) 策略。否则，可能会出现意外行为。例如，假设您在还原

表时删除了对该表的写入权限。在这种情况下，底层 RestoreTableToPointInTime 操作将无法向表中写入任何还原的数据。
您只能在还原操作完成之后修改或删除权限。

使用恢复已删除的表Amazon CLI

以下过程演示如何使用Amazon CLI要还原名为的已删除表myTable直到删除时间。

Note

此过程假设已在删除的表上启用了 PITR。

1. 删除您在上一教程中创建的表。

```
aws keyspaces delete-table --keyspace-name 'myKeyspace' --table-name 'myTable'
```

2. 使用以下命令将已删除的表恢复到删除时。

```
aws keyspaces restore-table --source-keyspace-name 'myKeyspace' --source-table-name
 'myTable' --target-keyspace-name 'myKeyspace' --target-table-name 'myTable_restored2'
```

此命令的输出将返回还原表的 ARN。

```
{
    "restoredTableARN": "arn:aws:cassandra:us-east-1:111222333444:/keyspace/myKeyspace/
table/myTable_restored2"
}
```

使用 CQL 恢复已删除的表

以下过程演示如何使用 CQL 还原名为的已删除表。mytable直到删除时间。

Note

此过程假设已在删除的表上启用了 PITR。

1. 为了确认 point-in-time 已删除表启用恢复，请查询系统表。只有带有 point-in-time 此时将显示启用恢复。

```
SELECT custom_properties
FROM system_schema_mcs.tables_history
WHERE keyspace_name = 'mykeyspace' AND table_name = 'my_table';
```

此查询将显示以下输出。

```
custom_properties
------------------
{
    ...,
   "point_in_time_recovery":{
      "restorable_until_time":"2020-08-04T00:48:58.381Z",
      "status":"enabled"
   }
}
```

2. 使用以下示例语句将表恢复到删除时间。

```
RESTORE TABLE mykeyspace.mytable_restored
FROM TABLE mykeyspace.mytable;
```

使用 Amazon Keyspaces 生存时间 (TTL) 使数据过期

Amazon Keyspaces（适用于 Apache Cassandra）生存时间 (TTL) 通过自动将表中的数据过期，帮助您简化应用程序逻辑并优化存储价格。根据您设置的生存时间值，您不再需要的数据将自动从表中删除。这使得根据业务、行业或法规要求（定义数据需要保留多长时间或指定必须删除数据的时间）更轻松地遵守数据保留策略。

例如，您可以在 AdTech 应用程序中使用 TTL 来安排特定广告的数据何时过期且客户不再可见。您还可以使用 TTL 自动淘汰较旧的数据并节省存储成本。您可以为整个表设置默认 TTL 值，然后为单个行和列覆盖该值。TTL 操作不会影响应用程序的性能。此外，标记为使用 TTL 过期的行数和列数不会影响表的可用性。

Amazon Keyspaces 会自动过滤掉过期的数据，以便过期的数据不会在查询结果中返回，也不会在数据操作语言 (DML) 语句中使用。Amazon Keyspaces 通常会在过期日期后 10 天内从存储中删除过期的数据。在极少数情况下，如果底层存储分区有持续活动以保护可用性，Amazon Keyspaces 可能无法在 10 天内删除数据。在这些情况下，一旦分区上的流量减少，Amazon Keyspaces 将继续尝试删除过期的数据。从存储中永久删除数据后，您将停止支付存储费。有关更多信息，请参阅the section called “工作原理” (p. 133)。

您可以使用控制台或 Cassandra 查询语言 (CQL) 为新表和现有表设置、修改或禁用默认 TTL 设置。在配置了默认 TTL 的表中，您可以使用 Cassandra 查询语言 (CQL) 覆盖默认 TTL 设置并将自定义 TTL 值应用于行和列。有关更多信息，请参阅the section called “如何使用生存时间” (p. 135)。

TTL 定价基于使用上线时间删除或更新的行的大小。TTL 运营按以下单位计量`TTL deletes`. 删除或更新的每行每 KB 数据消耗一个 TTL 删除。例如，要更新存储 2.5 KB 数据的行并同时删除该行中的一个或多个列，需要删除三个 TTL。或者，要删除包含 3.5 KB 数据的整行，需要删除四次 TTL。每行每 KB 删除的数据消耗一个 TTL 删除操作。有关定价的更多信息，请参阅Amazon Keyspaces（针对 Apache Cassandra）定价.

主题

工作原理：Amazon Keyspaces 生存时间 (TL)

Amazon Keyspaces 生存时间 (TL) 已完全托管。您不必管理低级系统设置，例如压缩策略。数据在您指定的时间到期，Amazon Keyspaces 会自动删除过期的数据（通常在 10 天内），而不会影响您的应用程序性能或可用性。

过期数据已标记为删除，无法用于数据操作语言 (DML) 语句。当您继续对包含过期数据的行执行读写操作时，过期数据将继续计入读取容量单位 (RCU) 和写入容量单位 (WCU)，直到将其从存储中删除。

主题

为表设置默认 TTL 值

在 Amazon Keyspaces 中，您可以在创建表时为表中的所有行设置默认 TTL 值。您还可以编辑现有表以设置或更改插入到表中的新行的默认 TTL 值。更改表的默认 TTL 值不会修改表中任何现有数据的 TTL 值。表的默认 TTL 值为零，这意味着数据不会自动过期。如果表的默认 TTL 值大于零，则会向每行添加一个过期时间戳。

每次更新数据时，Amazon Keyspaces 都会计算一个新的 TTL 时间戳。TTL 值以秒为单位设置，最大可配置值为 630,720,000 秒，相当于 20 年。有关如何设置、修改和禁用表的默认 TL 值的详细信息，请使用Amazon Web Services Management Console或 CQL，请参阅the section called “如何使用生存时间” (p. 135).

为行和列设置自定义 TTL 值

> Note
>
> 在为行和列设置自定义 TTL 值之前，必须先在表上启用 TTL。有关更多信息，请参阅 the section called “如何使用自定义属性在现有表上启用生存时间 (TTL)” (p. 137)。

要覆盖表的默认 TTL 值或为各个行设置过期日期，可以使用以下 CQL 数据操作语言 (DML) 语句：

- `INSERT`— 用于插入设置了 TTL 值的新数据行。
- `UPDATE`— 用于使用新的 TTL 值修改现有数据行。

为行设置 TTL 值优先于表的默认 TTL 设置。

对于 CQL 语法和示例，请参阅the section called “使用`INSERT`使用 CQL 编辑自定义生存时间 (TTL) 设置” (p. 137).

要覆盖或设置单个列的 TTL 值，可以使用以下 CQL DML 语句更新现有行中列子集的 TTL 设置：

- `UPDATE`— 用于更新一列数据。

为列设置 TTL 值优先于表的默认 TTL 设置和该行的任何自定义 TTL 设置。对于 CQL 语法和示例，请参阅the section called “使用`UPDATE`使用 CQL 编辑自定义生存时间 (TTL) 设置” (p. 137).

在表格上启用 TTL

当您指定 TTL 时，会自动为表格启用 TTL`default_time_to_live`任一值均大于 0`CREATE TABLE`要么`ALTER TABLE`语句。如果你没有指定`default_time_to_live`用于表，但您想使用为行或列指定自定义 TTL 值`INSERT`要么`UPDATE`操作，您必须首先为表启用 TL。您可以使用以下命令为表启用 TTL`ttl`自定义属性。

当您在表上启用 TTL 时，Amazon Keyspaces 开始为每行存储额外的 TTL 相关元数据。此外，TTL 使用到期时间戳来跟踪行或列何时过期。时间戳作为行元数据存储，会产生该行的存储成本。

启用 TTL 还要求 Amazon Keyspaces 先读取该行，然后再对该行执行写入操作，以确保数据的准确性。例如，要写入 3 KB 的行，除了 3 个 WCU/WRU 之外，还需要 1 个 RCU/RRU。启用 TL 功能后，您无法为表禁用它。设置桌子的`default_time_to_live`设置为 0 会禁用新数据的默认过期时间，但它不会停用 TTL 功能，也不会将表恢复到原来的 Amazon Keyspaces 存储元数据或写入行为。

Amazon Keyspaces 上线时间和集成Amazon服务

以下 TTL 指标在亚马逊上可用 CloudWatch 启用持续监控。

- `TTLDeletes`— 使用生存时间 (TTL) 删除或更新连续数据所消耗的单位。

有关如何监控的更多信息 CloudWatch 指标，请参阅the section called “使用 CloudWatch 进行监控” (p. 180).

当您使用以下应用程序时：Amazon CloudFormation，您可以在创建 Amazon Keyspaces 表时打开 TTL。有关更多信息，请参阅 Amazon CloudFormation 用户指南。

如何使用生存时间 (TTL)

您可以使用 Amazon Keyspaces (针对 Apache Cassandra) 控制台或 CQL 启用、更新和禁用生存时间设置。

主题

在启用默认生存时间 (TTL) 设置的情况下创建新表 (控制台)

请按照以下步骤创建一个使用 Amazon Keyspaces 控制台启用 “上线时间” 设置的新表。

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 在导航窗格中，选择 Tables (表)，然后选择 Create table (创建表)。
3. 在存储库的创建表在表详细信息部分中，选择键空间并为新表提供名称。
4. 在架构部分中，为表创建架构。
5. 在表设置部分，选择自定义设置.
6. 继续到数据过期.

   在此步骤中，您为表选择默认 TTL 设置。

   对于默认 TTL中，输入到期时间并选择您输入的时间单位，例如秒、天或年。亚马逊 Keyspaces 将在几秒钟内存储值。
7. 选择 Create Table。您的表是使用指定的默认 TTL 值创建的。

   Note

   您可以使用 CQL 编辑器中的数据操作语言 (DML) 覆盖特定行或列的默认 TTL 设置。

更新现有表 (控制台) 上默认生存时间 (TTL) 设置

请按照以下步骤使用 Amazon Keyspaces 控制台更新现有表的上线时间设置。

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 选择要更新的表，然后选择。编辑 TTL.
3. 对于默认 TTL中，输入到期时间并选择您输入的时间单位，例如秒、天或年。亚马逊 Keyspaces 将在几秒钟内存储值。这不会更改现有行的 TTL 值。
4. 在定义 TTL 设置时，选择保存更改。.

禁用现有表 (控制台) 上的默认生存时间 (TTL) 设置

按照以下步骤使用 Amazon Keyspaces 禁用现有表的上线时间设置Amazon Web Services Management Console.

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 选择要更新的表，然后选择。编辑 TTL.
3. Select默认 TTL然后将该值设置为零。默认情况下，对于将来的数据，这将禁用表的 TTL。它不会更改现有行的 TTL 值。
4. 在定义 TTL 设置时，选择保存更改。.

使用 CQL 在启用默认生存时间 (TTL) 设置的情况下创建新表

在创建默认 TTL 值设置为 3,024,000 秒（代表 35 天）的新表时，启用 TTL。

```
CREATE TABLE my_table (
                userid uuid,
                time timeuuid,
                subject text,
                body text,
                user inet,
                PRIMARY KEY (userid, time)
                ) WITH default_time_to_live = 3024000;
```

要确认新表的 TTL 设置，请使用cqlsh `describe`语句，如以下示例所示。输出将表的默认 TTL 设置显示为`default_time_to_live`.

```
describe my_table;
```

使用ALTER TABLE使用 CQL 编辑默认生存时间 (TTL) 设置

将现有表的 TTL 设置更新为 2,592,000 秒，即 30 天。

```
ALTER TABLE my_table WITH default_time_to_live = 2592000;
```

要确认更新表的 TTL 设置，请使用cqlsh `describe`语句，如以下示例所示。输出将表的默认 TTL 设置显示为`default_time_to_live`.

```
describe my_table;
```

如何使用自定义属性在新表上启用生存时间 (TTL)

要启用可以在不为整个表启用 TTL 默认设置的情况下应用于行和列的 “生效时间” 自定义设置，可以使用以下 CQL 语句。

```
CREATE TABLE my_keyspace.my_table (id int primary key) WITH CUSTOM_PROPERTIES={'ttl':
{'status': 'enabled'}};
```

晚于ttl启用了，则无法为表禁用它。

如何使用自定义属性在现有表上启用生存时间 (TTL)

要启用可以在不为整个表启用 TTL 默认设置的情况下应用于行和列的 “生效时间” 自定义设置，可以使用以下 CQL 语句。

```
ALTER TABLE my_table WITH CUSTOM_PROPERTIES={'ttl':{'status': 'enabled'}};
```

晚于ttl启用了，则无法为表禁用它。

使用INSERT使用 CQL 编辑自定义生存时间 (TTL) 设置

以下 CQL 语句将一行数据插入到表中，并将默认 TTL 设置更改为 259,200 秒（相当于 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;
```

要确认插入行的 TTL 设置，请使用以下语句。

```
SELECT TTL (subject) from my_table;
```

使用UPDATE使用 CQL 编辑自定义生存时间 (TTL) 设置

要将之前插入的 “主题” 列的 TTL 设置从 259,200 秒（3 天）更改为 86,400 秒（一天），请使用以下语句。

```
UPDATE my_table USING TTL 86400 set subject = 'Updated Message' WHERE userid =
 B79CB3BA-745E-5D9A-8903-4A02327A7E09 and time = 96a29100-5e25-11ec-90d7-b5d91eceda0a;
```

您可以运行一个简单的 select 查询，在到期时间之前查看更新的记录。

```
SELECT * from my_table;
```

查询将显示以下输出。

```
userid                                | time                                 | body  |
 subject         | user
--------------------------------------+--------------------------------------+-------
+-----------------+-----------------
b79cb3ba-745e-5d9a-8903-4a02327a7e09  | 96a29100-5e25-11ec-90d7-b5d91eceda0a | Hello |
 Updated Message | 205.212.123.123
```

```
50554d6e-29bb-11e5-b345-feff819cdc9f  | cf03fb21-59b5-11ec-b371-dff626ab9620 | Hello |
    Message | 205.212.123.123
```

要确认过期成功，请在配置的过期时间后再次运行同一查询。

```
SELECT * from my_table;
```

查询显示 “主题” 列过期后的以下输出。

```
userid                               | time                                 | body  |
 subject | user
--------------------------------------+--------------------------------------+-------
+---------+-----------------
b79cb3ba-745e-5d9a-8903-4a02327a7e09  | 96a29100-5e25-11ec-90d7-b5d91eceda0a | Hello |
 null | 205.212.123.123
50554d6e-29bb-11e5-b345-feff819cdc9f  | cf03fb21-59b5-11ec-b371-dff626ab9620 | Hello |
 Message | 205.212.123.123
```

使用创建亚马逊 Keyspaces 资源 Amazon CloudFormation

Amazon Keyspaces（针对 Apache Cassandra）与Amazon CloudFormation，是一项服务，可帮助您建模和设置Amazon这样您只需花较少的时间来创建和管理资源与基础设施。您创建一个模板来描述所有Amazon你想要的资源（例如密钥空间和表格），以及Amazon CloudFormation负责为您预配置和配置这些资源。

当您使用Amazon CloudFormation，您可重复使用您的模板来不断地重复设置您的 Amazon Keyspaces 资源。只需描述您的资源一次，然后在多个中反复配置相同的资源。Amazon Web Services 账户和地区。

Amazon Keyspaces 和Amazon CloudFormation模板

要为 Amazon Keyspaces 设置和配置资源，您必须了解Amazon CloudFormation模板. 模板是 JSON 或 YAML 格式的文本文件。这些模板描述要在 Amazon CloudFormation 堆栈中调配的资源。如果您不熟悉 JSON 或 YAML，可以在 Amazon CloudFormation Designer 的帮助下开始使用 Amazon CloudFormation 模板。有关更多信息，请参阅 。是什么Amazon CloudFormation设计器？中的Amazon CloudFormation用户指南.

亚马逊 Keyspaces 支持在中创建密钥空间和表Amazon CloudFormation. 对于您使用创建的表Amazon CloudFormation模板中，可指定架构、读/写模式和预置的吞吐容量设置。有关更多信息（包括键空间和表的 JSON 和 YAML 模板示例），请参阅Cassandra 资源类型参考中的Amazon CloudFormation用户指南.

了解有关 Amazon CloudFormation 的更多信息

要了解有关 Amazon CloudFormation 的更多信息，请参阅以下资源：

- Amazon CloudFormation
- Amazon CloudFormation 用户指南
- Amazon CloudFormation命令行界面

向亚马逊 Keyspaces 资源添加标签和标签

您可使用使用标签为 Amazon Keyspaces (Apache Cassandra) 资源进行标记。标签. 标签可让您按各种标准对资源进行分类，例如，按用途、所有者、环境或其他标准进行分类。标签可帮助您：

- 根据您分配到资源的标签来快速识别资源。
- 按标签查看 Amazon 账单细分。
- 支持基于标签控制对 Amazon Keyspaces 资源的访问。有关使用标签的 IAM 策略示例，请参阅the section called “基于 Amazon Keyspaces 标签的” (p. 165).

支持标签Amazon服务，例如 Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Simple Storage Service (Amazon S3)、Amazon Keyspaces 等。有效的标签让您可对具有特定标签的服务创建报告，从而提供成本分析。

要开始使用标签，请执行以下操作：

1. 了解 Amazon Keyspaces 的标签限制 (p. 140)。
2. 使用 亚马逊 Keyspaces 标记操作 (p. 140) 创建标签。
3. 使用 亚马逊 Keyspaces 的成本分配报告 (p. 144) 跟踪各个活动标签的 Amazon 成本。

最后，最佳实践是遵循最佳标签策略。想要了解有关信息，请参阅Amazon标记策略.

Amazon Keyspaces 的标签限制

每个标签都由键 和值组成，这两个参数都由您定义。以下限制适用：

- 每个 Amazon Keyspaces 键空间或表的同一个键只能有一个标签。如果您尝试添加现有标签（相同键），现有标签值会更新为新值。
- 应用于密钥空间的标签不会自动应用于该密钥空间中的表。要将相同的标签应用于密钥空间及其所有表，必须单独标记每个资源。
- 值充当标签类别（键）中的描述符。在 Amazon Keyspaces 中，值不能为空或为 null。
- 标签键和值区分大小写。
- 最大键长度为 128 个 Unicode 字符。
- 最大值长度为 256 个 Unicode 字符。
- 允许的字符包括字母、空格和数字，以及以下特殊字符：`+ - = . _ : /`
- 每个资源的最大标签数是 50。
- Amazon分配的标签名称和值将自动被分配`aws:`前缀，这是您无法分配的。Amazon-分配的标签名称不会计入 50 个标签限制之列。用户分配的标签名称在成本分配报告中具有 `user:` 前缀。
- 您不能回溯标签的应用日期。

亚马逊 Keyspaces 标记操作

您可以使用 Amazon Keyspaces (对于 Apache Cassandra) 控制台添加、列出、编辑或删除键空间和表的标签。AmazonCLI 或 Cassandra 查询语言 (CQL)。然后，您可以激活这些用户定义的标签，以便在 Amazon

Billing and Cost Management 控制台上显示这些标签以进行成本分配跟踪。有关更多信息，请参阅 亚马逊 Keyspaces 的成本分配报告 (p. 144)。

对于批量编辑，您还可以使用控制台上的标签编辑器。有关更多信息，请参阅 Amazon Resource Groups 用户指南中的使用标签编辑器。

主题

使用控制台将标签添加到新的或现有的键空间和表。

您可以在创建新键空间和表时使用 Amazon Keyspaces 控制台将标签添加到新的键空间和表。您还可以添加、编辑或删除现有表的标签。

在创建键空间表时对键空间进行标记（控制台）

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 在导航窗格中，选择 Keyspaces (键空间)，然后选择 Create keyspace (创建键空间)。
3. 在 Create keyspace (创建键空间) 页面上，提供键空间的名称。输入标签的键和值，然后选择 Add new tag (添加新标签)。
4. 选择 Create keyspace (创建键空间)。

在创建表时对表进行标记（控制台）

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 在导航窗格中，选择 Tables (表)，然后选择 Create table (创建表)。
3. 在存储库的创建表中的页面表详细信息部分中，选择键空间并提供表的名称。
4. 在架构部分中，为表创建架构。
5. 在表设置部分，选择自定义设置.
6. 继续浏览表格标签 —可选的部分，然后选择添加新标签来创建新标签。
7. 选择 Create Table。

标记现有资源（控制台）

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 在导航窗格中，选择 Keyspaces (键空间) 或 Tables (表)。
3. 在列表中选择键空间或表。然后选择 Manage tags (管理标签) 以添加、编辑或删除标签。

有关标签结构的信息，请参阅 Amazon Keyspaces 的标签限制 (p. 140)。

将标签添加到新的或现有的键空间和表使用AmazonCLI

此部分中的示例演示如何使用AmazonCLI 将在创建键空间和表时指定标签、如何从现有资源中添加或删除标签以及如何列出标签。

以下示例演示如何创建带有标签的新表。该命令创建一个表myTable在已经存在的密钥空间中myKeySpace. 请注意，为了提高可读性，该命令已分解为不同的行。

```
aws keyspaces create-table --keyspace-name 'myKeyspace' --table-name 'myTable'
            --schema-definition 'allColumns=[{name=id,type=int},{name=name,type=text},
{name=date,type=timestamp}],partitionKeys=[{name=id}]'
            --tags 'key=key1,value=val1' 'key=key2,value=val2'
```

以下示例演示如何将新标签添加到现有的表。

```
aws keyspaces tag-resource --resource-arn 'arn:aws:cassandra:us-east-1:111222333444:/
keyspace/myKeyspace/table/myTable' --tags 'key=key3,value=val3' 'key=key4,value=val4'
```

下一个示例说明如何列出指定资源的标签。

```
aws keyspaces list-tags-for-resource --resource-arn 'arn:aws:cassandra:us-
east-1:111222333444:/keyspace/myKeyspace/table/myTable'
```

最后一个命令的输出如下所示。

```
{
    "tags": [
        {
            "key": "key1",
            "value": "val1"
        },
        {
            "key": "key2",
            "value": "val2"
        },
        {
            "key": "key3",
            "value": "val3"
        },
        {
            "key": "key4",
            "value": "val4"
        }
    ]
}
```

使用 CQL 将标签添加到新的或现有的键空间和表。

以下示例演示了在创建键空间和表时如何使用 CQL 指定标签、如何标记现有资源以及如何读取标签。

以下示例创建一个带有标签的新键空间：

```
CREATE KEYSPACE mykeyspace WITH TAGS = {'key1':'val1', 'key2':'val2'} ;
```

以下示例创建带有标签的新表。

```
CREATE TABLE mytable(...) WITH TAGS = {'key1':'val1', 'key2':'val2'};
```

在语句中使用其他命令标记资源。

```
CREATE KEYSPACE mykeyspace WITH REPLICATION = {'class': 'Simple Strategy'} AND TAGS
 = {'key1':'val1', 'key2':'val2'};
```

以下示例说明如何添加或删除现有键空间和表上的标签。

```
ALTER KEYSPACE mykeyspace ADD TAGS {'key1':'val1', 'key2':'val2'};
```

```
ALTER TABLE mytable DROP TAGS {'key1':'val1', 'key2':'val2'};
```

要读取附加到资源的标签，请使用以下 CQL 语句。

```
SELECT * FROM system_schema_mcs.tags WHERE valid_where_clause;
```

WHERE 子句必需，且必须为以下格式之一：

- keyspace_name = **'mykeyspace'** AND resource_type = 'keyspace'
- keyspace_name = **'mykeyspace'** AND resource_name = **'mytable'**
- resource_id = *arn*

示例：

以下查询显示键空间是否具有标签。

```
SELECT * FROM system_schema_mcs.tags WHERE keyspace_name = 'mykeyspace' AND resource_type =
 'keyspace';
```

查询的输出如下所示。

```
resource_id                                                              | keyspace_name |
 resource_name | resource_type | tags
--------------------------------------------------------------------------+---------------
+---------------+---------------+------
arn:aws:cassandra:us-east-1:123456789:/keyspace/mykeyspace/      | mykeyspace    |
 mykeyspace    | keyspace      | {'key1': 'val1', 'key2': 'val2'}
```

以下查询显示表的标签。

```
SELECT * FROM system_schema_mcs.tags WHERE keyspace_name = 'mykeyspace' AND resource_name
 = 'mytable';
```

该查询的输出如下所示。

```
resource_id                                                                    | keyspace_name
 | resource_name | resource_type | tags
--------------------------------------------------------------------------------
+---------------+---------------+---------------+------
arn:aws:cassandra:us-east-1:123456789:/keyspace/mykeyspace/table/mytable    | mykeyspace
 | mytable       | table         | {'key1': 'val1', 'key2': 'val2'}
```

亚马逊 Keyspaces 的成本分配报告

Amazon 使用标签组织成本分配报告上的资源成本。Amazon 提供了两种类型的成本分配标签：

- Amazon 生成的标签 Amazon 为您定义、创建和应用此标签。
- 用户定义的标签。您定义、创建和应用这些标签。

您必须先分别激活两种类型的标签，然后这些标签才能显示在 Cost Explorer 中或成本分配报告上。

要激活 Amazon 生成的标签：

1. 登录 Amazon Web Services Management Console，打开 Billing and Cost Management 控制台：https://console.aws.amazon.com/billing/home#/。
2. 在导航窗格中，选择 Cost Allocation Tags (成本分配标签)。
3. 在 Amazon 生成的成本分配标签下，选择激活。

激活用户定义的标签：

1. 登录 Amazon Web Services Management Console，打开 Billing and Cost Management 控制台：https://console.aws.amazon.com/billing/home#/。
2. 在导航窗格中，选择 Cost Allocation Tags (成本分配标签)。
3. 在 User-Defined Cost Allocation Tags (用户生成的成本分配标签) 下，选择 Activate (激活)。

创建并激活标签后，Amazon 生成成本分配报告，其中按活动标签分组了使用情况和成本。成本分配报告包括您每个账单周期的所有 Amazon 成本。该报告包括标记资源和未标记资源，因此您可以清晰地排列资源费用。

Note

目前，从 Amazon Keyspaces 传出的所有数据不会在成本分配报告上按标签细分。

有关更多信息，请参阅 。使用成本分配标签.

Amazon Keyspaces（针对 Apache Cassandra）的安全性

Amazon 的云安全性的优先级最高。作为 Amazon 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。

安全性是Amazon和您的共同责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云的安全性 – Amazon负责保护在 Amazon Web Services 云 中运行 Amazon 服务的基础设施。Amazon 还向您提供可安全使用的服务。作为 Amazon 合规性计划的一部分，我们的安全措施的有效性定期由第三方审计员进行测试和验证。要了解适用于 Amazon Keyspaces 的合规性计划，请参阅Amazon按合规性计划划分的服务范围.
- 云中的安全性 - 您的责任由您使用的 Amazon 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您组织的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 Amazon Keyspaces 时应用责任共担模型。以下主题说明如何配置 Amazon Keyspaces 以实现您的安全性和合规性目标。您还将了解如何使用其他Amazon可以帮助您监控和保护 Amazon Keyspaces 资源的服务。

主题

Amazon Keyspaces 中的数据保护

这些区域有：Amazon 责任共担模式适用于 Amazon Keyspaces (in Apache Cassandra) 中的数据保护。如该模式中所述，Amazon 负责保护运行所有 Amazon Web Services 云 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。此内容包括您所使用的 Amazon Web Services 的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。

出于数据保护目的，我们建议您保护 Amazon Web Services 账户 凭证并使用 Amazon Identity and Access Management (IAM) 设置单独的用户账户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用 Multi-Factor Authentication (MFA)。
- 使用 SSL/TLS 与 Amazon 资源进行通信。建议使用 TLS 1.2 或更高版本。
- 使用 Amazon CloudTrail 设置 API 和用户活动日志记录。
- 使用 Amazon 加密解决方案以及 Amazon 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Simple Storage Service（Amazon S3）中的个人数据。
- 如果在通过命令行界面或 API 访问 Amazon 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准 (FIPS) 第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（例如您客户的电子邮件地址）放入标签或自由格式字段（例如名称字段）。这包括使用 Amazon Keyspaces 或其他操作时Amazon使用控制台、API、Amazon CLI，或者Amazon开发工具包。您在用于名称的标签或自由格式字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，我们强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

主题

亚马逊密 Keyspaces 中的静态加密

Amazon Keyspaces（针对 Apache Cassandra）静态加密通过使用存储在Amazon Key Management Service(Amazon KMS）. 此功能减少保护敏感数据时涉及的操作负担和复杂性。通过静态加密，您可以构建符合数据保护方面的严格合规性和法规要求的安全敏感型应用程序。

Amazon Keyspaces 静态加密使用 256 位高级加密标准 (AES-256) 来对您的数据加密。这有助于保护您的数据，防止对底层存储进行未经授权的访问。

Amazon Keyspaces 会以透明方式加密和解密表数据。Amazon Keyspaces 使用信封加密和密钥层次结构来保护数据加密密钥。它与集成Amazon KMS用于存储和管理根加密密钥。有关加密密钥层次结构的更多信息，请参阅the section called “工作方式” (p. 146). 有关 的更多信息Amazon KMS信封加密等概念，请参阅Amazon KMS管理服务概念中的Amazon Key Management Service开发人员指南.

创建新表时，可以选择以下选项之一：Amazon KMS密钥（KMS 密钥）：

- Amazon 拥有的密钥— 这是默认加密类型。此密钥归亚马逊 Keyspaces 拥有（不另外收费）。
- 客户管理的密钥 — 此密钥存储在账户中，由您创建、拥有和管理。您对客户管理的密钥拥有完全控制权（Amazon KMS收取费用）。

您可以在Amazon 拥有的密钥客户在任何指定时间管理的密钥。您可以在创建新表时指定客户托管密钥，或使用控制台或以编程方式使用 CQL 语句来更改现有表的 KMS 密钥。要了解如何操作，请参阅静态加密：如何使用客户管理的密钥加密 Amazon Keyspaces 中的表 (p. 149)。

使用默认选项进行静态加密Amazon 拥有的密钥不另外收取 费用。但是，Amazon KMS客户托管密钥需支付费用。有关定价的更多信息，请参阅 Amazon KMS 定价。

所有亚马逊静态密 Keyspaces 加密Amazon Web Services 区域，包括Amazon中国（北京）和Amazon中国（宁夏）区域。有关更多信息，请参阅静态加密：它在亚马逊 Keyspaces 中的运作方 (p. 146)。

主题

静态加密：它在亚马逊 Keyspaces 中的运作方

Amazon Keyspaces（针对 Apache Cassandra）静态加密使用 256 位高级加密标准 (AES-256) 来加密数据。这有助于保护您的数据，防止对底层存储进行未经授权的访问。默认情况下，Amazon Keyspaces 表中的所有客户数据都是静态加密的，服务器端加密是透明的，这意味着不需要对应用程序进行更改。

静态加密集成 Amazon Key Management Service (Amazon KMS)，管理用于加密表的加密密钥。创建新表或更新现有表时，可以选择以下之一：Amazon KMS密钥选项

- Amazon 拥有的密钥— 这是默认加密类型。此密钥归 Amazon Keyspaces 拥有（不另外收费）。
- 客户托管密钥 — 此密钥存储在账户中，由您创建、拥有和管理。您对客户托管密钥拥有完全控制权 (Amazon KMS将收取费用）。

Amazon KMS密钥（KMS 密钥）

静态加 Keyspaces 用Amazon KMS键。默认情况下，亚马逊 Keyspaces 使用Amazon 拥有的密钥，在 Amazon Keyspace 服务账户中创建和管理的多租户加密密钥。

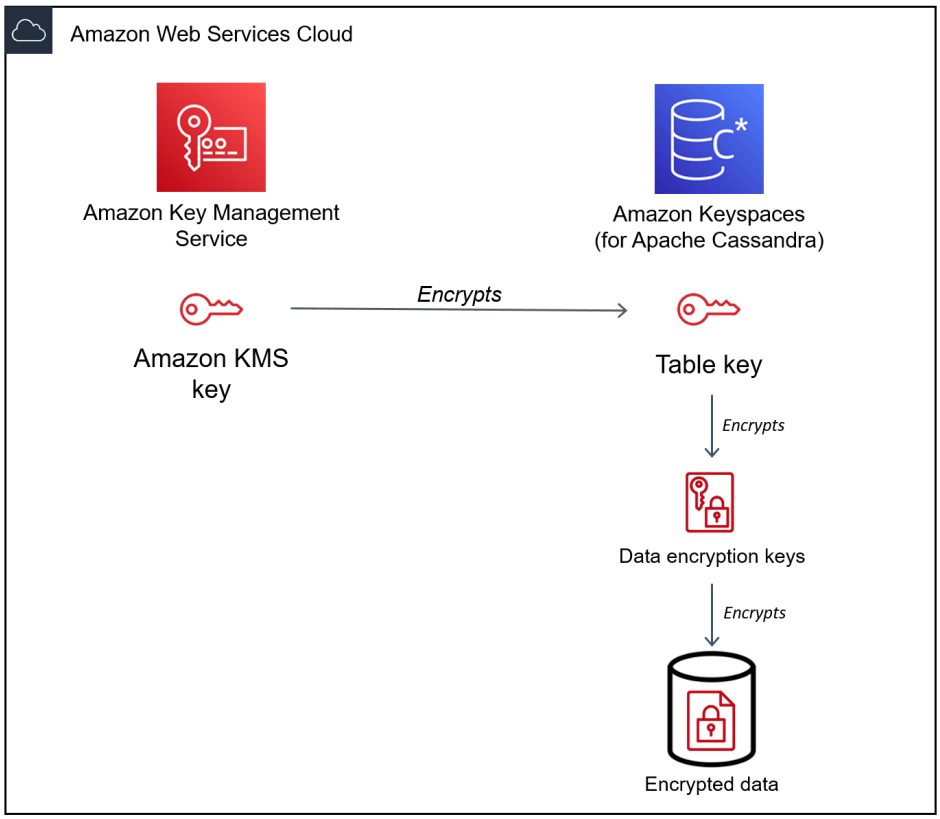
但是，您可以使用客户托管密钥在您的Amazon Web Services 账户. 您可以为密钥空间中的每个表选择不同的 KMS 密钥。您为表选择的 KMS 密钥也可用于加密表的所有元数据和可还原备份。

您可以在创建或更新表时为表选择 KMS 密钥。可通过以下方式随时更改表的 KMS 密钥：在 Amazon KKeyspaces 控制台中或使用更改表 (p. 208)网页。切换 KMS 密钥的过程是无缝的，不需要停机时间或导致服务降级。

键层次结构

Amazon Keyspaces 使用密钥层次结构来加密数据。在此密钥层次结构中，KMS 密钥是根密钥。它用于加密和解密 Amazon Keyspaces 表加密密钥。表加密密钥用于加密 Amazon Keyspaces 内部使用的加密密钥，以便在执行读取和写入操作时加密和解密数据。

使用加密密钥层次结构，您可以更改 KMS 密钥，而无需重新加密数据或影响应用程序和正在进行的数据操作。



表密钥

Amazon Keyspaces 表密钥用作密钥。Amazon Keyspaces 使用表密钥来保护用于加密表、日志文件和可还原备份中的数据的内部数据加密密钥。Amazon Keyspaces 会为表中的每个底层结构生成唯一的数据加密密钥。但是，多个表行可能受相同的数据加密密钥保护。

当您第一次将 KMS 密钥设置为客户托管密钥时，Amazon KMS生成数据密钥. 这些区域有：Amazon KMS数据密钥是指亚马逊密钥空间中的表格键。

当您访问加密表时，Amazon Keyspaces 会向Amazon KMS使用 KMS 密钥来解密表密钥。然后，它会使用明文表密钥解密 Amazon Keyspaces 数据加密密钥，它会使用明文数据加密密钥解密表数据。

Amazon Keyspaces 在外部使用和存储表密钥和数据加密密钥。Amazon KMS. 它会借助高级加密标准 (AES) 加密和 256 位加密密钥保护所有密钥。然后，它会将加密密钥与加密数据存储在一起，以便它们可根据需要用于解密表数据。

表密钥缓存

为了避免打电话Amazon KMS针对每个 Amazon Keyspaces 操作，Amazon Keyspaces 会针对每个连接将明文表密钥缓存在内存中。如果 Amazon Keyspaces 在处于不活动状态 5 分钟后获取缓存表密钥的请求，它会向Amazon KMS解密表密钥。此调用捕获对中的 KMS 密钥的访问策略所做的任何更改Amazon KMS要么Amazon Identity and Access Management自上次请求解密表密钥以来 (IAM)。

信封加密

如果更改表的客户托管密钥，Amazon Keyspaces 会生成新的表密钥。然后，它将使用新的表密钥来重新加密数据加密密钥。它还使用新的表密钥来加密之前用于保护可恢复备份的表密钥。这个过程称为信封加密。这可确保即使轮换客户托管密钥，也可以访问可恢复的备份。有关信封加密的更多信息，请参阅信封加密中的Amazon Key Management Service开发人员指南.

主题

Amazon拥有的钥匙

Amazon 拥有的密钥没有存储在你的Amazon Web Services 账户. 它们是 KMS 密钥集合的一部分Amazon拥有和管理以便在多个中使用Amazon Web Services 账户.Amazon服务可以使用Amazon 拥有的密钥以保护您的数据。

你无法查看、管理或使用Amazon 拥有的密钥，或者审计它们的使用情况。但是，您无需执行任何工作或更改任何程序即可保护用于加密您的数据的密钥。

您无需支付月费或使用费Amazon 拥有的密钥，并且它们不计入Amazon KMS您的账户的配额。

客户托管密钥

客户管理的密钥是您的密钥Amazon Web Services 账户您创建、拥有和管理的。您对这些 KMS 密钥拥有完全控制权。

使用客户托管密钥可获得以下功能：

- 您创建和管理客户管理的密钥，包括设置和维护密钥策略、IAM 策略, 和GRANT以控制对客户托管密钥的访问。您可以启用和禁用客户管理的密钥，启用和禁用自动密钥轮换, 和安排客户托管密钥当它不再使用时将其删除。您可以为您管理的客户托管密钥创建标签和别名。
- 您可以使用具有导入的密钥材料的客户托管密钥，或者您拥有和管理的自定义密钥存储中的客户托管密钥。
- 您可以使用Amazon CloudTrail和 AmazonCloudWatch用于跟踪 Amazon Keyspaces 发送到的请求的日志Amazon KMS代表您。有关更多信息，请参阅 the section called “第 6 步：使用配置监控Amazon CloudTrail” (p. 154)。

客户托管密钥产生费用对于每次 API 调用，以及Amazon KMS配额适用于这些 KMS 密钥。有关更多信息，请参阅 。Amazon KMS资源或请求配额.

如果指定客户托管密钥作为表的根加密密钥，可还原备份将使用创建备份时为表指定的同一加密密钥加密。如果轮换表的 KMS 密钥，则密钥包络可确保最新的 KMS 密钥可以访问所有可恢复的备份。

Amazon Keyspaces 必须有权访问您的客户托管密钥，才能为您提供对表数据的访问权限。如果加密密钥的状态设置为已禁用或计划删除，则 Amazon Keyspaces 将无法加密或解密数据。因此，您无法对表执行读取和写入操作。如果服务检测到您的加密密钥无法访问，Amazon Keyspaces 会立即向您发送电子邮件通知以提醒您。

您必须在七天内恢复对加密密钥的访问权限，否则 Amazon Keyspaces 会自动删除表。作为预防措施，Amazon Keyspaces 在删除表之前创建表数据的可恢复备份。亚马逊 Keyspaces 将可恢复的备份维护 35 天。35 天后，您无法再恢复表数据。您无需为可恢复的备份付费，但需支付的费用将收取恢复费用.

您可以使用此可恢复的备份将数据还原到新表。要还原，必须启用用于表的最后一个客户托管密钥，Amazon Keyspaces 必须具有访问权限。

> Note
>
> 当您创建使用无法访问或计划在创建过程完成之前删除的客户托管密钥加密的表时，会出现错误。创建表操作失败，并向您发送电子邮件通知。

静态加密

在 Amazon Keyspaces 中使用静态加密时，请注意以下事项。

- 所有 Amazon Keyspaces 表上都启用了服务器端静态加密，无法禁用。整个表都是静态加密的，你不能选择特定的列或行进行加密。
- 默认情况下，Amazon Keyspaces 使用单一服务默认密钥（Amazon 拥有的密钥）来加密您的所有表。如果此密钥不存在，将创建该密钥。无法禁用服务默认密钥。
- 静态加密仅加密持久存储介质上的静态数据。如果对于正在传输的数据或正在使用的数据而言，数据安全性很重要，则必须执行额外措施：
  - 传输中的数据：Amazon Keyspaces 中的所有数据都在传输过程中加密。默认情况下，与 Amazon Keyspaces 的通信将通过安全套接字层 (SSL)/传输层安全性 (TLS) 加密来进行保护。
  - 正在使用的数据：在将数据发送到 Amazon Keyspaces 之前，使用客户端加密保护您的数据。
  - 客户托管密钥：表中的静态数据始终使用客户管理的密钥进行加密。但是，对多行执行原子更新的操作临时加密数据Amazon 拥有的密钥在处理过程中。这包括范围删除操作和同时访问静态和非静态数据的操作。
- 单个客户托管密钥最多可拥有 50,000GRANT. 与客户托管密钥关联的每个 Amazon Keyspace 表都消耗 2 项授权。删除表时释放一笔赠款。第二项授权用于创建表的自动快照，以防止 Amazon Keyspace 无意中无意中失去对客户管理密钥的访问权限时，防止数据丢失。该补助金将在删除表格后 42 天发布。

静态加密：如何使用客户管理的密钥加密 Amazon Keyspaces 中的表

您可以使用控制台或 CQL 语句来指定Amazon KMS key用于新表并更新 Amazon Keyspaces 中现有表的加密密钥。以下主题概述了如何为新表和现有表实施客户托管密钥。

主题

- 第 6 步：使用配置监控Amazon CloudTrail (p. 154)

先决条件：使用创建客户托管密钥Amazon KMS并向 Amazon Keyspaces 授予权限

在您可以使用保护亚马逊 Keyspaces 表之前客户托管密钥 (p. 148)，首先必须在中创建密钥Amazon Key Management Service(Amazon KMS) 然后授权亚马逊密 Keyspaces 使用该密钥。

第 1 步：使用创建客户托管密钥Amazon KMS

要创建用于保护 Amazon Keyspace 表的客户托管密钥，您可以按照中的步骤操作创建对称加密 KMS 密钥使用控制台或AmazonAPI。

第 2 步：授权使用您的客户管理密钥

在你可以选择客户托管密钥 (p. 148)要保护 Amazon Keyspaces 表，该客户托管密钥的策略必须赋予 Amazon Keyspaces 代表您使用该 Keyspace 的权限。您可以完全控制客户托管密钥的策略和授权。您可以在密钥策略、IAM 策略或授权中提供这些权限。

亚马逊 Keyspaces 不需要额外的授权即可使用默认值Amazon 拥有的密钥 (p. 148)保护您的亚马逊 Keyspaces 表Amazonaccount.

以下主题展示了如何使用允许 Amazon Keyspace 表使用客户托管密钥的 IAM 策略和授权配置所需的权限。

主题

- 客户托管密钥的主要策略 (p. 150)
- 示例密钥策略 (p. 150)
- 使用授予来向 Amazon Keyspaces 授权 (p. 151)

客户托管密钥的主要策略

当你选择客户托管密钥 (p. 148)为保护 Amazon Keyspaces 表，Amazon Keyspaces 将获得代表做出选择的委托人使用客户托管密钥的权限。该委托人（用户或角色）必须具有 Amazon Keyspace 所需的客户托管密钥权限。

Amazon Keyspaces 对客户托管密钥至少需要具备以下权限：

- kms:Encrypt
- kms:Decrypt
- kms:ReEncrypt* (对于公里：ReEncrypt来自和公里：ReEncrypt至)
- kms:GenerateData键 * (用于kms:GenerateData密钥和kms:GenerateDataKeyWithout明文)
- kms:DescribeKey
- kms:CreateGrant

示例密钥策略

例如，以下示例密钥策略仅提供所需的权限。该策略具有以下效果：

- 允许 Amazon Keyspaces 在加密操作中使用客户托管密钥并创建授权，但仅当它代表账户中具备 Amazon Keyspaces 使用权限的委托人行事时才可如此。如果策略语句中指定的委托人无权使用 Amazon Keyspaces，调用将失败，即使调用来自 Amazon Keyspaces 服务也是如此。
- 这些区域有：kms:ViaService条件密钥仅当 Amazon Keyspaces 代表策略语句中所列委托人发出请求时，才允许权限。这些委托人不能直接调用这些操作。请注意，`kms:ViaService` 值 `cassandra.*.amazonaws.com`在“区域”位置中有一个星号 (*)。亚马逊 Keyspaces 需要独立于任何特定的权限Amazon Web Services 区域.

- 为客户管理的密钥管理员（可以假设db-teamrole) 对客户托管密钥的只读访问权限以及撤销授权的权限，包括亚马逊 Keyspaces 所需的授予 (p. 151)来保护桌子。
- 赋予 Amazon Keyspaces 对客户托管密钥的只读访问权限。在此情况下，Amazon Keyspaces 可以直接调用这些操作。它不必代表账户委托人行事。

在使用示例密钥策略之前，请将示例委托人替换为 Amazon Web Services 账户 中的实际委托人。

```
{
  "Id": "key-policy-cassandra",
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid" : "Allow access through Amazon Keyspaces for all principals in the account that
 are authorized to use Amazon Keyspaces",
      "Effect": "Allow",
      "Principal": {"AWS": "arn:aws:iam::111122223333:user/db-lead"},
      "Action": [
        "kms:Encrypt",
        "kms:Decrypt",
        "kms:ReEncrypt*",
        "kms:GenerateDataKey*",
        "kms:DescribeKey",
        "kms:CreateGrant"
      ],
      "Resource": "*",
      "Condition": {
         "StringLike": {
           "kms:ViaService" : "cassandra.*.amazonaws.com"
         }
      }
    },
    {
      "Sid":  "Allow administrators to view the customer managed key and revoke grants",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/db-team"
       },
      "Action": [
        "kms:Describe*",
        "kms:Get*",
        "kms:List*",
        "kms:RevokeGrant"
      ],
      "Resource": "*"
    }
  ]
}
```

使用授予来向 Amazon Keyspaces 授权

除密钥策略以外，Amazon Keyspaces 还使用授权来对客户托管密钥设置权限。要查看有关账户中客户托管密钥的授权，请使用ListGrantsoperation. Amazon Keyspaces 不需要授予或任何其他权限即可使用Amazon 拥有的密钥 (p. 148)保护桌。

Amazon Keyspaces 在执行后台系统维护和连续数据保护任务时使用授予权限。它还使用授权来生成表密钥。

每个授权特定于一个表。如果账户中包含使用同一客户托管密钥加密的多个表，则每个表都有一种授权。补助受到Amazon Keyspaces 加密上下文，其中包括表名和Amazon Web Services 账户ID. 该授权包括以下权限：停用授予如果不再需要。

要创建授权，Amazon Keyspaces 必须具备调用的权限。CreateGrant代表创建了加密表的用户。

该密钥策略还可以允许账户撤销授予在客户托管密钥上。但是，如果您对某个活动加密表撤销授权，Amazon Keyspaces 将无法保护和维护该表。

第 3 步：为新表指定客户托管密钥

请按照以下步骤，使用 Amazon Keyspaces 控制台或 CQL 在新表上指定客户托管密钥。

使用客户托管密钥（控制台）创建加密表

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.
2. 在导航窗格中，选择 Tables (表)，然后选择 Create table (创建表)。
3. 在存储库的创建表在中的页面表详细信息部分中，选择键空间并提供新表的名称。
4. 在架构部分中，为表创建架构。
5. 在表设置部分，选择。自定义设置.
6. 继续到加密设置.

   在此步骤中，您将选择表的加密设置。

   在静态加密位置在部分选择Amazon KMS key中，选择选项选择不同的 KMS 密钥（高级），然后在搜索字段中，选择Amazon KMS key也可输入 Amazon Resource Name（ARN）。

   Note

   如果您选择的密钥无法访问或缺少所需的权限，请参阅密钥访问故障排除中的Amazon Key Management Service开发人员指南 的第一个版本。
7. 选择 Create (创建) 以创建加密表。

使用客户托管密钥创建新表以进行静态加密 (CQL)

要创建使用客户托管密钥进行静态加密的新表，可以使用`CREATE TABLE`语句，如以下示例所示。确保将密钥 ARN 替换为 ARN，以获得授予 Amazon Keyspace 的权限的有效密钥。

```
CREATE TABLE my_keyspace.my_table(id bigint, name text, place text STATIC, PRIMARY KEY(id,
 name)) WITH CUSTOM_PROPERTIES = {
        'encryption_specification':{
                'encryption_type': 'CUSTOMER_MANAGED_KMS_KEY',
                'kms_key_identifier':'arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111'
            }
    };
```

如果你收到`Invalid Request Exception`，您需要确认客户托管密钥有效且 Amazon Keyspaces 具有所需的权限。要确认密钥已正确配置，请参阅密钥访问故障排除中的Amazon Key Management Service开发人员指南 的第一个版本。

第 4 步：更新现有表的加密密钥

还可以使用 Amazon Keyspaces 控制台或 CQL 来更改现有表的加密密钥。Amazon 拥有的密钥而且客户随时托管 KMS 密钥。

使用新的客户管理密钥（控制台）更新现有表

1. 登录到Amazon Web Services Management Console，然后打开亚马逊 Keyspaces 控制台https://console.aws.amazon.com/keyspaces/home.

2. 在导航窗格中，选择表。
3. 选择要更新的表，然后选择其他设置选项卡。
4. 在静态加密部分，选择。管理加密以编辑表的加密设置。

   位置在选择Amazon KMS key中，选择选项选择不同的 KMS 密钥（高级）在搜索字段中，选择一个 Amazon KMS key也可输入 Amazon Resource Name（ARN）。

   Note

   如果您选择的密钥无效，请参阅密钥访问故障排除中的Amazon Key Management Service开发人员指南 的第一个版本。

   或者，您也可以选择Amazon 拥有的密钥对于使用客户托管密钥加密的表。
5. 选择保存更改以保存对表所做的更改。

更新用于现有表的加密密钥

要更改现有表的加密密钥，请使用`ALTER TABLE`指定适用于静态加密的客户托管密钥。确保将密钥 ARN 替换为 ARN，以获得授予 Amazon Keyspace 的权限的有效密钥。

```
ALTER TABLE my_keyspace.my_table WITH CUSTOM_PROPERTIES = {
              'encryption_specification':{
                     'encryption_type': 'CUSTOMER_MANAGED_KMS_KEY',
                     'kms_key_identifier':'arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111'
                   }
         };
```

如果你收到`Invalid Request Exception`，您需要确认客户托管密钥有效且 Amazon Keyspaces 具有所需的权限。要确认密钥已正确配置，请参阅密钥访问故障排除中的Amazon Key Management Service开发人员指南 的第一个版本。

要将加密密钥更改回默认的静态加密选项，请使用Amazon 拥有的密钥，您可以使用`ALTER TABLE`语句，如以下示例所示。

```
ALTER TABLE my_keyspace.my_table WITH CUSTOM_PROPERTIES = {
                'encryption_specification':{
                     'encryption_type' : 'AWS_OWNED_KMS_KEY'
                   }
         };
```

第 5 步：在日志中使用 Amazon Keyspaces 加密上下文

加密上下文 是一组包含任意非机密数据的键值对。在请求中包含加密上下文以加密数据时，Amazon KMS 以加密方式将加密上下文绑定到加密的数据。要解密数据，您必须传入相同的加密上下文。

Amazon Keyspaces 在所有时候使用相同的加密上下文。Amazon KMS加密操作。如果您将客户托管密钥 (p. 148)要保护 Amazon Keyspaces 表，您可以使用加密上下文在审核记录和日志中标识客户托管密钥的使用。它还以明文形式显示在日志中，例如，在日志中显示。Amazon CloudTrail和亚马逊CloudWatch日志.

在它的请求中Amazon KMS，Amazon Keyspaces 使用具有三个密钥-值对的加密上下文。

```
"encryptionContextSubset": {
    "aws:cassandra:keyspaceName": "my_keyspace",
    "aws:cassandra:tableName": "mytable"
    "aws:cassandra:subscriberId": "111122223333"
```

```
}
```

- 键空间— 第一个密钥-值对用于标识包含 Amazon Keyspace 正在加密的表的密 Keyspaces 间。密钥是 `aws:cassandra:keyspaceName`。值是密钥空间的名称。

```
"aws:cassandra:keyspaceName": "<keyspace-name>"
```

例如：

```
"aws:cassandra:keyspaceName": "my_keyspace"
```

- 表Account — 第二个密钥-值对用于标识 Amazon Keyspaces 正在加密的表。密钥是 `aws:cassandra:tableName`。值为表的名称。

```
"aws:cassandra:tableName": "<table-name>"
```

例如：

```
"aws:cassandra:tableName": "my_table"
```

- 账户— 第三个密钥-值对用于标识Amazon Web Services 账户. 密钥是 `aws:cassandra:subscriberId`。该值为账户 ID。

```
"aws:cassandra:subscriberId": "<account-id>"
```

例如：

```
"aws:cassandra:subscriberId": "111122223333"
```

第 6 步：使用配置监控Amazon CloudTrail

如果您将客户托管密钥 (p. 148)为了保护您的亚马逊 Keyspaces 表，您可以使用Amazon CloudTrail用于跟踪 Amazon Keyspaces 发送到的请求的日志Amazon KMS代表您。

这些区域有：`GenerateDataKey`、`DescribeKey`、`Decrypt`, 和`CreateGrant`本部分将讨论请求。此外，亚马逊 Keyspaces 使用RetireGrant操作来在您删除表时删除授权。

GenerateData密钥

Amazon Keyspaces 会创建唯一的表密钥来加密静态数据。它会发送GenerateData密钥请求Amazon KMS它指定表 KMS 密钥。

记录 `GenerateDataKey` 操作的事件与以下示例事件类似。该用户是 Amazon Keyspaces 服务账户。参数包括客户托管密钥的 Amazon 资源名称 (ARN)、需要 256 位密钥的密钥说明符以及加密上下文 (p. 153)标识密钥空间、表格和Amazon Web Services 账户.

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2021-04-16T04:56:05Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "GenerateDataKey",
```

```
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keySpec": "AES_256",
        "encryptionContext": {
            "aws:cassandra:keyspaceName": "my_keyspace",
            "aws:cassandra:tableName": "my_table",
            "aws:cassandra:subscriberId": "123SAMPLE012"
        },
        "keyId": "arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111"
    },
    "responseElements": null,
    "requestID": "5e8e9cb5-9194-4334-aacc-9dd7d50fe246",
    "eventID": "49fccab9-2448-4b97-a89d-7d5c39318d6f",
    "readOnly": true,
    "resources": [
        {
            "accountId": "123SAMPLE012",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123SAMPLE012",
    "sharedEventID": "84fbaaf0-9641-4e32-9147-57d2cb08792e"
}
```

DescribeKey

Amazon Keyspaces 使用DescribeKey操作来确定所选 KMS 密钥是否存在于账户和区域中。

记录 `DescribeKey` 操作的事件与以下示例事件类似。该用户是 Amazon Keyspaces 服务账户。参数包括客户托管密钥的 ARN 和需要 256 位密钥的密钥说明符。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAZ3FNIIVIZZ6H7CFQG",
        "arn": "arn:aws:iam::123SAMPLE012:user/admin",
        "accountId": "123SAMPLE012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "admin",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-04-16T04:55:42Z"
            }
        },
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2021-04-16T04:55:58Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "DescribeKey",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
```

```
    "requestParameters": {
        "keyId": "arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111"
    },
    "responseElements": null,
    "requestID": "c25a8105-050b-4f52-8358-6e872fb03a6c",
    "eventID": "0d96420e-707e-41b9-9118-56585a669658",
    "readOnly": true,
    "resources": [
        {
            "accountId": "123SAMPLE012",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123SAMPLE012"
}
```

Decrypt

当您访问 Amazon Keyspaces 表时，Amazon Keyspaces 需要解密表密钥，以便它可以解密层次结构中位于其下方的密钥。然后，解密表中的数据。要解密表密钥，Amazon Keyspaces 将Decrypt请求Amazon KMS它指定表 KMS 密钥。

记录 `Decrypt` 操作的事件与以下示例事件类似。用户是您的 Amazon Web Services 账户 中正在访问表的委托人。参数包括加密表密钥（作为密文 blob）和加密上下文 (p. 153)标识表格和Amazon Web Services 账户.Amazon KMS从密文中派生客户托管密钥的 ID。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AWSService",
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2021-04-16T05:29:44Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "Decrypt",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "encryptionContext": {
            "aws:cassandra:keyspaceName": "my_keyspace",
            "aws:cassandra:tableName": "my_table",
            "aws:cassandra:subscriberId": "123SAMPLE012"
        },
        "encryptionAlgorithm": "SYMMETRIC_DEFAULT"
    },
    "responseElements": null,
    "requestID": "50e80373-83c9-4034-8226-5439e1c9b259",
    "eventID": "8db9788f-04a5-4ae2-90c9-15c79c411b6b",
    "readOnly": true,
    "resources": [
        {
            "accountId": "123SAMPLE012",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111"
        }
    ],
```

```
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123SAMPLE012",
    "sharedEventID": "7ed99e2d-910a-4708-a4e3-0180d8dbb68e"
}
```

CreateGrant

当您使用客户托管密钥 (p. 148)为了保护您的亚马逊 Keyspaces 表，亚马逊 Keyspaces 使用授予 (p. 151)允许该服务执行连续数据保护和维护和持久性任务。这些补助金不是必需的Amazon 拥有的密钥 (p. 148).

Amazon Keyspaces 创建的授权特定于表。中的委托人CreateGrant请求是创建表的用户。

记录 `CreateGrant` 操作的事件与以下示例事件类似。参数包括表的客户托管密钥的 ARN、被授权委托人和停用委托人（Amazon Keyspaces 服务）以及授权涵盖的操作。它还包括要求所有加密操作均使用指定的加密上下文 (p. 153).

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAZ3FNIIVIZZ6H7CFQG",
        "arn": "arn:aws:iam::arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111:user/admin",
        "accountId": "arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111",
        "accessKeyId": "AKIAI44QH8DHBEXAMPLE",
        "userName": "admin",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2021-04-16T04:55:42Z"
            }
        },
        "invokedBy": "AWS Internal"
    },
    "eventTime": "2021-04-16T05:11:10Z",
    "eventSource": "kms.amazonaws.com",
    "eventName": "CreateGrant",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "AWS Internal",
    "requestParameters": {
        "keyId": "a7d328af-215e-4661-9a69-88c858909f20",
        "operations": [
            "DescribeKey",
            "GenerateDataKey",
            "Decrypt",
            "Encrypt",
            "ReEncryptFrom",
            "ReEncryptTo",
            "RetireGrant"
        ],
        "constraints": {
            "encryptionContextSubset": {
                "aws:cassandra:keyspaceName": "my_keyspace",
                "aws:cassandra:tableName": "my_table",
                "aws:cassandra:subscriberId": "123SAMPLE012"
            }
        },
```

```
        "retiringPrincipal": "cassandratest.us-east-1.amazonaws.com",
        "granteePrincipal": "cassandratest.us-east-1.amazonaws.com"
    },
    "responseElements": {
        "grantId": "18e4235f1b07f289762a31a1886cb5efd225f069280d4f76cd83b9b9b5501013"
    },
    "requestID": "b379a767-1f9b-48c3-b731-fb23e865e7f7",
    "eventID": "29ee1fd4-28f2-416f-a419-551910d20291",
    "readOnly": false,
    "resources": [
        {
            "accountId": "123SAMPLE012",
            "type": "AWS::KMS::Key",
            "ARN": "arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "eventCategory": "Management",
    "recipientAccountId": "123SAMPLE012"
}
```

Amazon Keyspaces 中的传输中加密

Amazon Keyspaces 仅接受使用传输层安全性 (TLS) 协议的安全连接。传输中加密通过对传输中的数据加密，为 Amazon Keyspaces 提供额外一层数据保护。组织政策、行业或政府法规以及合规性需求通常要求使用传输中加密增强数据安全性。

学习如何加密`cqlsh`使用 TLS 连接到亚马逊 Keyspaces，请参阅the section called “加密`cqlsh`使用 TLS 连接” (p. 23). 要了解如何通过客户端驱动程序使用 TLS 加密，请参阅the section called “使用 Cassandra 客户端驱动程序” (p. 28).

Amazon Keyspaces 中的互联网络流量保密性

本主题介绍了 Amazon Keyspaces（适用于 Apache Cassandra）如何保护从本地应用程序到亚马逊密钥空间以及亚马逊密 Keyspaces 间与其他密 Keyspaces 间之间的连接Amazon相同的资源Amazon Web Services 区域.

服务与本地客户端和应用之间的流量

私有网络和 Amazon 之间有两种连接方式：

- Amazon Site-to-Site VPN 连接。有关更多信息，请参阅 Amazon Site-to-Site VPN 用户指南的 什么是 Amazon Site-to-Site VPN？。
- Amazon Direct Connect 连接。有关更多信息，请参阅 Amazon Direct Connect 用户指南的 什么是 Amazon Direct Connect？。

通过网络访问 Amazon Keyspaces 是通过的Amazon发布的 API。客户端必须支持传输层安全性 (TLS) 1.0。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Diffie-Hellman Ephemeral (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

Amazon Keyspaces 支持两种验证客户端请求的方法。第一种方法使用特定于服务的凭据，这些凭据是为特定特定生成的基于密码Amazon Identity and Access Management(IAM) 用户。您可以使用 IAM 控制台来创建和管理密码。Amazon CLI，或者AmazonAPI。有关更多信息，请参阅 。将 IAM 与 Amazon Keyspaces 结合使用.

第二种方法使用开源代码的身份验证插件。DataStax卡桑德拉的 Java 驱动程序。启用此插件IAM 用户、角色和联合身份验证使用向 Amazon Keyspaces（针对 Apache Cassandra）API 请求添加身份验证信息Amazon签名版本 4 流程 (SigV4). 有关更多信息，请参阅 ??? (p. 15)。

同一区域中 Amazon 资源之间的流量

接口 VPC 终端节点可在 Amazon VPC 中运行的 Virtual Private Cloud (VPC) 与 Amazon Keyspaces 之间实现私有通信。接口 VPC 终端节点由 Amazon PrivateLink 支持，这是一种实现 VPC 与 Amazon 服务之间私有通信的 Amazon 服务。Amazon PrivateLink 通过对 VPC 中的私有 IP 使用弹性网络接口启用此功能，以便网络流量不会离开 Amazon 网络。接口 VPC 终端节点不需要 Internet 网关、NAT 设备、VPN 连接或 Amazon Direct Connect 连接。有关更多信息，请参阅 。Amazon Virtual Private Cloud和接口 VPC 终端节点 (Amazon PrivateLink) . 有关示例策略，请参阅 the section called “对 Amazon Keyspaces 使用接口 VPC 终端节点” (p. 196)。

Amazon Identity and Access Management针对 Amazon Keyspaces

Amazon Identity and Access Management (IAM) 是一项 Amazon Web Service，可以帮助管理员安全地控制对 Amazon 资源的访问。IAM 管理员控制谁可以成为身份验证（已登录）和授权（有权限）使用亚马逊 Keyspaces 资源。IAM 是一项无需额外费用即可使用的 Amazon Web Service。

主题

Audience

如何使用Amazon Identity and Access Management(IAM) 因您可以在 Amazon Keyspaces 中执行的操作而异。

服务用户— 如果您使用 Amazon Keyspaces 服务来完成作业，则您的管理员会为您提供所需的凭证和权限。随着您使用更多 Amazon Keyspaces 功能来完成工作，您可能需要额外权限。了解如何管理访问权限可帮助您向管理员请求适合的权限。如果您无法访问 Amazon Keyspaces 中的一项功能，请参阅Amazon Keyspaces 身份和访问 (p. 175).

服务管理员— 如果您在公司负责管理 Amazon Keyspaces 资源，则您可能具有 Amazon Keyspaces 的完全访问权限。您有责任确定您的服务用户应访问哪些 Amazon Keyspaces 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Amazon Keyspaces 搭配使用的更多信息，请参阅Amazon Keyspaces 与 IAM (p. 162).

IAM 管理员— 如果您是 IAM 管理员，您可能希望了解如何编写策略以管理 Amazon Keyspaces 的访问权限的详细信息。要查看您可在 IAM 中使用的 Amazon Keyspaces 基于身份的策略示例，请参阅Amazon Keyspaces 基于身份的策略示例 (p. 165).

使用身份进行身份验证

身份验证是您使用身份凭证登录 Amazon 的方法。有关使用Amazon Web Services Management Console登录的更多信息，请参阅 IAM 用户指南中的以 IAM 用户或根用户身份登录Amazon Web Services Management Console。

您必须作为Amazon Web Services 账户根用户、IAM 用户或代入 IAM 角色以进行身份验证（登录到 Amazon）。您还可以使用公司的单一登录身份验证方法，甚至使用 Google 或 Facebook 登录。在这些情况下，您的管理员以前使用 IAM 角色设置了联合身份验证。在您使用来自其它公司的凭证访问 Amazon 时，您间接地代入了角色。

要直接登录到Amazon Web Services Management Console，请将密码与根用户电子邮件地址或 IAM 用户名一起使用。您可以使用根用户或 IAM 用户访问密钥以编程方式访问 Amazon。Amazon 提供了 SDK 和命令行工具，可使用您的凭证对您的请求进行加密签名。如果您不使用 Amazon 工具，则必须自行对请求签名。使用 Signature Version 4（用于对入站 API 请求进行验证的协议）完成此操作。有关验证请求的更多信息，请参阅《Amazon 一般参考》中的 Signature Version 4 签名流程。

无论使用何种身份验证方法，您可能还需要提供其它安全信息。例如，Amazon 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《IAM 用户指南》中的在 Amazon 中使用多重身份验证 (MFA)。

Amazon Web Services 账户根用户

当您创建 Amazon Web Services 账户时，最初使用的是一个对账户中所有 Amazon Web Services和资源拥有完全访问权限的登录身份。此身份称为 Amazon Web Services 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《Amazon 一般参考》中的需要根用户凭证的任务。

IAM 用户和组

IAM 用户是Amazon Web Services 账户内对某个人员或应用程序具有特定权限的一个身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有密码和访问密钥等长期凭证的 IAM 用户。但是，如果您有特定的使用场景需要带有 IAM 用户的长期凭证，我们建议您轮换访问密钥。有关更多信息，请参阅 。对于需要长期凭证的使用场景，请定期轮换访问密钥在里面IAM 用户指南.

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色是Amazon Web Services 账户中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 Amazon Web Services Management Console 中暂时代入 IAM 角色。您可以调用 Amazon CLI 或 Amazon API 操作或使用自定义 URL 以代入角色。有关使用角色的方法的更多信息，请参阅 IAM 用户指南 中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合身份用户访问— 要向联合身份分配权限，您需要创建角色并为角色定义权限。当联合身份进行身份验证时，该身份会与角色关联，被授予角色定义的权限。有关联合角色的信息，请参见针对第三方身份提供商创建角色在里面IAM 用户指南.

如果您使用 IAM Identity Center，则需要配置权限集。为了控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集关联到 IAM 中的角色。有关权限集的信息，请参见权限集在里面Amazon IAM Identity Center (successor to Amazon Single Sign-On)用户指南.

- IAM 用户的临时权限— IAM 用户或角色可以代入 IAM 角色，以暂时获得不同的权限以执行特定的任务。
- 跨账户访问 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些 Amazon Web Services，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅 IAM 用户指南中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 – 某些 Amazon Web Services使用其它 Amazon Web Services中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Simple Storage Service（Amazon S3）中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。
  - 主体权限 – 当您使用 IAM 用户或角色在 Amazon 中执行操作时，您将被视为主体。策略向主体授予权限。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中触发另一个操作。在这种情况下，您必须具有执行这两个操作的权限。要查看某个操作是否需要策略中的其他相关操作，请参阅Amazon Keyspaces（针对 Apache Cassandra）的操作、资源和条件键在里面服务授权参考.
  - 服务角色 – 服务角色是服务代表您在您的账户中执行操作而担任的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅 IAM 用户指南中的创建向 Amazon Web Service 委派权限的角色。
  - 服务相关角色 – 服务相关角色是与 Amazon Web Service关联的一种服务角色。服务可以代入代表您执行操作的角色。服务相关角色显示在您的 IAM 账户中，并归该服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 Amazon CLI 或 Amazon API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 Amazon 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 IAM 用户指南中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅 IAM 用户指南中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您将创建策略并将其附加到 Amazon 身份或资源，以控制 Amazon 中的访问。策略是 Amazon 中的对象；在与身份或资源相关联时，策略定义它们的权限。在主体（用户、根用户或角色会话）发出请求时，Amazon 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 Amazon 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南 中的 JSON 策略概览。

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

每个 IAM 实体（用户或角色）最初没有任何权限。原定设置情况下，用户什么都不能做，甚至不能更改他们自己的密码。要为用户授予执行某些操作的权限，管理员必须将权限策略附加到用户。或者，管理员可以将用户添加到具有预期权限的组中。当管理员为某个组授予访问权限时，该组内的全部用户都会获得这些访问权限。

IAM policy 定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。具有该策略的用户可以从 Amazon Web Services Management Console、Amazon CLI 或 Amazon API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM policy。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管式策略是可以附加到Amazon Web Services 账户中的多个用户、组和角色的独立策略。托管式策略包括 Amazon 托管式策略和客户托管式策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅 IAM 用户指南 中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Simple Storage Service（Amazon S3）存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。主体可以包括账户、用户、角色、联合身份用户或 Amazon Web Services。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用来自 IAM 的 Amazon 托管式策略。

访问控制列表 (ACL)

访问控制列表 (ACL) 控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Simple Storage Service（Amazon S3）、Amazon WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅 Amazon Simple Storage Service 开发人员指南 中的访问控制列表 (ACL) 概览。

其它策略类型

Amazon 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界 – 权限边界是一个高级功能，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体的基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅 IAM 用户指南中的 IAM 实体的权限边界。
- 服务控制策略 (SCP) – SCP 是 JSON 策略，指定了组织或组织单位 (OU) 在 Amazon Organizations 中的最大权限。Amazon Organizations 服务可以分组和集中管理您的企业拥有的多个 Amazon Web Services 账户 账户。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体的权限，包括每个 Amazon Web Services 账户 根用户。有关 Organizations 和 SCP 的更多信息，请参阅 Amazon Organizations 用户指南中的 SCP 的工作原理。
- 会话策略 – 会话策略是当您以编程方式为角色或联合身份用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解 Amazon 如何确定在涉及多种策略类型时是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

Amazon Keyspaces 与 IAM

在使用 IAM 管理对 Amazon Keyspaces 的访问权限之前，您应该了解哪些 IAM 功能可用于 Amazon Keyspaces。大致了解Amazon Keyspaces 和其他方法Amazon服务与 IAM 一起使用，请参阅Amazon使用 IAM 的服务在里面IAM 用户指南.

主题

Amazon Keyspaces 基于身份的策略

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。Amazon Keyspaces 支持特定的操作和资源以及条件键。要了解您在 JSON 策略中使用的所有元素，请参阅 IAM 用户指南中的 IAM JSON 策略元素参考。

要查看 Amazon Keyspaces 的服务特定资源和操作，以及可用于 IAM 权限策略的条件上下文密钥，请参阅Amazon Keyspaces（针对 Apache Cassandra）的操作、资源和条件键在里面服务授权参考.

操作

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么 条件 下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 Amazon API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行相关操作的权限。

Amazon Keyspaces 中的策略操作在操作前使用以下前缀：`cassandra:`. 例如，授予某人使用亚马逊 Keyspaces 创建亚马逊Keyspaces 密钥空间的权限`CREATE`CQL 语句，你要包括`cassandra:Create`在他们的政策中采取行动。策略语句必须包含 `Action` 或 `NotAction` 元素。Amazon Keyspaces 定义了一组自己的操作，以描述您可以使用该服务执行的任务。

要在单个语句中指定多项操作，请使用逗号将它们隔开，如下所示：

```
"Action": [
      "cassandra:CREATE",
      "cassandra:MODIFY"
          ]
```

要查看 Amazon Keyspaces 操作的列表，请参阅Amazon Keyspaces（针对 Apache Cassra）定义的操作在里面服务授权参考.

资源

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源 执行操作，以及在什么条件 下执行。

`Resource` JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 `Resource` 或 `NotResource` 元素。作为最佳实践，请使用其 Amazon Resource Name (ARN) 指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

在 Amazon Keyspaces 中，密钥空间和表可以用于`Resource`IAM 权限的元素。

Amazon Keyspaces 键空间资源具有以下 ARN：

```
arn:${Partition}:cassandra:${Region}:${Account}:/keyspace/${KeyspaceName}/
```

Amazon Keyspaces 表资源具有以下 ARN：

```
arn:${Partition}:cassandra:${Region}:${Account}:/keyspace/${KeyspaceName}/table/
${tableName}
```

有关 ARN 格式的更多信息，请参阅 Amazon Resource Name (ARN) 和 Amazon 服务命名空间。

例如，要在语句中指定 `mykeyspace` 键空间，请使用以下 ARN：

```
"Resource": "arn:aws:cassandra:us-east-1:123456789012:/keyspace/mykeyspace/"
```

要指定属于特定账户的所有键空间，请使用通配符 (*)：

```
"Resource": "arn:aws:cassandra:us-east-1:123456789012:/keyspace/*"
```

无法对特定资源执行某些 Amazon Keyspaces 操作，例如，用于创建资源的操作。在这些情况下，您必须使用通配符 (*)。

```
"Resource": "*"
```

要使用标准驱动程序以编程方式连接到 Amazon Keyspaces，用户必须具有系统表的 SELECT 访问权限，因为大多数驱动程序在连接时都会读取系统密钥空间/表。例如，授予`SELECT`向用户授予的权限`mytable`在`mykeyspace`，IAM 用户必须具有读取两者的权限，`mytable`还有`system keyspace`. 要在单个语句中指定多个资源，请使用逗号分隔 ARN。

```
"Resource": "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/table/mytable",
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
```

要查看Amazon Keyspaces 资源类型及其 ARN 的列表，请参阅Amazon Keyspaces（针对 Apache Cassra）定义的资源在里面服务授权参考. 要了解您可以使用哪些操作指定每个资源的 ARN，请参阅Amazon Keyspaces（针对 Apache Cassra）定义的操作.

条件键

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 `Condition` 元素，或在单个 `Condition` 元素中指定多个键，则 Amazon 使用逻辑 `AND` 运算评估它们。如果您为单个条件键指定多个值，则 Amazon 使用逻辑 `OR` 运算来评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅 IAM 用户指南 中的 IAM policy 元素：变量和标签。

Amazon支持全局条件键和特定于服务的条件键。要查看所有 Amazon 全局条件键，请参阅《IAM 用户指南》中的 Amazon 全局条件上下文键。

Amazon Keyspaces 定义了自己的一组条件键，还支持使用一些全局条件键。要查看所有 Amazon 全局条件键，请参阅《IAM 用户指南》中的 Amazon 全局条件上下文键。

所有 Amazon Keyspaces 操作都支持`aws:RequestTag/${TagKey}`，`aws:ResourceTag/${TagKey}`，还有`aws:TagKeys`条件键。有关更多信息，请参阅 the section called “ 基于标签的亚马逊 Keyspaces 资源访问权限” (p. 170)。

要查看 Amazon Keyspaces 条件键的列表，请参阅Amazon Keyspaces（针对 Apache Cassra）的条件键在里面服务授权参考. 要了解您可以对哪些操作和资源使用条件键，请参阅Amazon Keyspaces（针对 Apache Cassra）定义的操作.

示例

要查看 Amazon Keyspaces 基于身份的策略的示例，请参阅Amazon Keyspaces 基于身份的策略示例 (p. 165).

Amazon Keyspaces 基于资源的

Amazon Keyspaces 不支持基于资源的策略。要查看详细的基于资源的策略页面示例，请参阅 https://docs.amazonaws.cn/lambda/latest/dg/access-control-resource-based.html。

基于 Amazon Keyspaces 标签的

您可以使用标签管理对您的 Amazon Keyspaces 资源的访问。要基于标签管理资源访问，您需要在条件元素使用的策略`cassandra:ResourceTag/`*`key-name`*,`aws:RequestTag/`*`key-name`*，或`aws:TagKeys`条件键。有关标记 Amazon Keyspaces 资源的更多信息，请参阅为 资源添加标签 (p. 140).

要查看基于身份的策略（用于根据资源上的标签来限制对该资源的访问）的示例，请参阅 基于标签的亚马逊 Keyspaces 资源访问权限 (p. 170)。

Amazon Keyspaces

IAM 角色是 Amazon Web Services 账户中具有特定权限的实体。

将临时凭证用于 Amazon Keyspaces

您可以使用临时凭证进行联合身份登录，担任 IAM 角色或担任跨账户角色。您可以通过调用获得临时安全凭证Amazon STSAPI 操作，如AssumeRole要么GetFederationToken.

Amazon Keyspaces 支持将临时凭证用于Amazon Keyspaces (p. 32). 要查看如何使用身份验证插件以编程方式访问表的示例，请参阅the section called “使用身份验证插件访问亚马逊Keyspaces” (p. 168)。

服务相关角色

服务相关角色允许 Amazon 服务访问其它服务中的资源以代表您完成操作。服务相关角色显示在您的 IAM 账户中，并归该服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理 Amazon Keyspaces 服务相关角色的详细信息，请参阅the section called “使用服务相关角色” (p. 177).

服务角色

亚马逊Keyspaces 不支持服务角色。

Amazon Keyspaces 基于身份的策略示例

默认情况下，IAM 用户和角色没有创建或修改 Amazon Keyspaces 资源的权限。它们还无法使用控制台CQLSH 执行任务Amazon CLI，或AmazonAPI。IAM 管理员必须创建 IAM policy，以便为用户和角色授予权限以对所需的指定资源执行特定的 API 操作。然后，管理员必须将这些策略附加到需要这些权限的 IAM 用户或组。

要了解如何使用这些示例 JSON 策略文档创建 IAM 基于身份的策略，请参阅 IAM 用户指南中的在 JSON 选项卡上创建策略。

主题

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 Amazon Keyspaces 资源。这些操作可能会使 Amazon Web Services 账户 产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- Amazon 托管策略及转向最低权限许可入门 - 要开始向用户和工作负载授予权限，请使用 Amazon 托管策略来为许多常见使用场景授予权限。您可以在 Amazon Web Services 账户 中找到这些策略。我们建议通过定义特定于您的使用场景的 Amazon 客户管理型策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 Amazon 托管策略或工作职能的 Amazon 托管策略。
- 应用最低权限 – 在使用 IAM policy 设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM policy 中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果通过特定 Amazon Web Service（例如 Amazon CloudFormation）使用服务操作，您还可以使用条件来授予对服务操作的访问权限。有关更多信息，请参阅 。IAM JSON 策略元素：Condition在里面IAM 用户指南.
- 使用 IAM Access Analyzer 验证您的 IAM policy，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM policy语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Acess Analyzer 策略验证。
- 需要多重身份验证 (MFA) – 如果您的账户需要 IAM 用户或根用户，请启用 MFA 来提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实践的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 Amazon Keyspaces 控制

Amazon Keyspaces 不需要特定的权限即可访问到 Amazon Keyspaces（针对 Apache Cassandra）控制台。您至少需要只读权限才能列出和查看有关您的 Amazon Keyspaces 资源的详细信息Amazon Web Services 账户. 如果您创建的基于身份的策略比所需的最低权限更严格，则无法为具有该策略的实体（IAM 用户或角色）正常运行控制台。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上完成此操作或者以编程方式使用 Amazon CLI 或 Amazon API 所需的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon Keyspaces

以下是授予对系统表的只读 (SELECT) 访问权限的示例策略。对于所有示例，请将 Amazon 资源名称 (ARN) 中的区域和账户 ID 替换为您服务中的相应内容。

> **Note**
>
> 要使用标准驱动程序进行连接，用户必须至少具有对系统表的 SELECT 访问权限，因为大多数驱动程序在连接时读取系统键空间/表。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "cassandra:Select"
         ],
         "Resource":[
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
         ]
      }
   ]
}
```

以下示例策略添加对用户表的只读访问权限。

```
{
   "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*"
         ]
      }
   ]
}
```

以下示例策略分配对用户表的读/写访问权限和对系统表的读取访问权限。

Note

系统表为只读。

```
{
   "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*"
         ]
      }
   ]
}
```

以下示例策略允许用户在键空间 `mykeyspace` 中创建表。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":[
            "cassandra:Create",
            "cassandra:Select"
         ],
         "Resource":[
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/mykeyspace/*",
            "arn:aws:cassandra:us-east-1:111122223333:/keyspace/system*"
         ]
      }
   ]
}
```

使用身份验证插件访问亚马逊Keyspaces

在以下示例中，您运行的应用程序使用Amazon Keyspaces (p. 32)从与 IAM 角色关联的 EC2 实例访问亚马逊Keyspaces，以查询您的 Amazon Keyspaces 表。应用程序从名为 `orders` 的表中返回给定客户 ID 的所有订单。

此示例包括以下步骤：

1. 创建 Amazon Keyspaces 密钥空间和表，然后插入记录。
2. 启动 EC2 实例并将其与 IAM 角色关联。

定义向您的 Amazon Keyspaces 表授予 EC2 实例权限的 IAM 策略。

3. 在 EC2 实例上部署并运行示例代码以查询您的 Amazon Keyspaces 表。

创建示例表

要开始，请创建键空间和表，然后插入一些记录。创建`orders`表，你可以使用 Amazon Keyspaces 控制台上的 CQL 编辑器，也可以使用 cqlsh。有关更多信息，请参阅 the section called “安装和使用`cqlsh`” (p. 22)。

首先，使用以下表架构创建键空间和表。

```
create keyspace acme with replication = {'class': 'SimpleStrategy', 'replication_factor' :
 1 };
create table acme.orders (
       customer_id text,
       order_timestamp timestamp,
       order_id uuid,
       primary key (customer_id, order_timestamp)) with clustering order by
 (order_timestamp desc);
```

接下来，在表中插入一些订单记录。

```
insert into acme.orders (customer_id, order_timestamp, order_id)
          values ('1234', toTimestamp(now()), uuid());
          insert into acme.orders (customer_id, order_timestamp, order_id)
          values ('1234', toTimestamp(now()), uuid());
          insert into acme.orders (customer_id, order_timestamp, order_id)
          values ('1234', toTimestamp(now()), uuid());
          insert into acme.orders (customer_id, order_timestamp, order_id)
          values ('1234', toTimestamp(now()), uuid());
```

启动与 IAM 角色关联的 EC2 实例

创建将与运行应用程序的 EC2 实例关联的 IAM 角色。

将以下文档保存到名为 `iam-keyspaces-ec2-role.json` 的文件中。

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

要授予 EC2 实例对 `orders` 表的只读访问权限，请创建以下 IAM 策略。将亚马逊资源名称 (ARN) 中的区域和账户 ID 替换为 Amazon Keyspaces 账户中的 ID。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Effect":"Allow",
         "Action":"cassandra:Select",
         "Resource":[
```

```
            "arn:aws:cassandra:us-east-2:111122223333:/keyspace/acme/table/orders",
            "arn:aws:cassandra:us-east-2:111122223333:/keyspace/system*"
        ]
    }
  ]
}
```

将以下文档保存到名为 `iam-keyspaces-ec2-policy.json` 的文件中。

Note

您必须在策略中包含 `system` 键空间的权限，因为驱动程序必须访问此键空间以检索集群元数据。

要启动 EC2 实例并将创建的 IAM 角色与该实例关联，请按照中的步骤操作启动带有 IAM 角色的实例. 要将角色与现有 EC2 实例关联，请参阅附加 IAM 角色.

Note

为此示例选择预安装了 Java 的 Amazon Linux AMI。如果您使用的是现有 Amazon Linux 实例，请确认您已安装 Java，或使用命令 `sudo yum install java-11-amazon-corretto-headless` 完成此步骤。

部署和运行代码

要部署和运行代码来查询您的 Amazon Keyspaces 表，请使用以下步骤：

1. 从下载示例代码 GitHub.
2. 使用 Apache Maven 编译示例代码。
3. 将 JAR 文件复制到 EC2 实例，然后运行应用程序。

本示例使用了一个示例应用程序，该应用程序实现了Amazon Keyspaces (p. 32)使用 IAM 角色访问该表。在您自己的代码中，您必须按照 the section called “适用于 Java 4.x 的身份验证插件” (p. 32) 中的步骤集成插件。

从下载示例后GitHub 知识库，使用以下命令使用 Apache Maven 3.6.3 或更高版本编译代码。

```
mvn package
```

这将创建以下 JAR 文件，其所有依赖项包含在 `target` 目录中。

```
aws-sigv4-auth-cassandra-java-driver-examples-1.0.0.jar
```

将此 JAR 文件以及位于示例代码目录中的 `cassandra_truststore.jks` 复制到 EC2 实例。使用以下命令运行应用程序。更换 AWS 区域`us-east-2`使用您正在使用的区域。

```
java -Djavax.net.ssl.trustStore=./cassandra_truststore.jks \
     -Djavax.net.ssl.trustStorePassword=amazon -jar \
      aws-sigv4-auth-cassandra-java-driver-examples-1.0.0.jar \
      us-east-2 cassandra.us-east-2.amazonaws.com 1234
```

此命令会生成您之前插入到表中的记录。应用程序使用分配给 IAM 角色会话的临时凭证访问表，并具有在 IAM 策略中定义的只读权限。

基于标签的亚马逊Keyspaces 资源访问权限

您可以在基于身份的策略中使用条件，以便基于标签控制对 Amazon Keyspaces 资源的访问。这些策略控制账户中密钥空间和表的可见性。请注意，使用系统表发出请求时，基于标签的系统表权限的行为会有所不同 AmazonSDK 与通过 Cassandra 驱动程序和开发者工具调用卡桑德拉查询语言 (CQL) API 进行了比较。

- 制作List和Get资源请求AmazonSDK 在使用基于标签的访问时，调用者需要具有对系统表的读取权限。例如，Select需要操作权限才能通过系统表读取数据GetTable操作。如果调用者对特定表只有基于标签的访问权限，则需要额外访问系统表的操作将失败。
- 为了与既定的 Cassandra 驱动程序行为兼容，在通过 Cassandra 驱动程序和开发者工具使用 Cassandra 查询语言 (CQL) API 调用对系统表执行操作时，不强制执行基于标签的授权策略。

以下示例说明如何创建一个策略，该策略授予用户查看表的权限（如果表的 Owner 包含该用户的用户名的值）。在此示例中，您还提供了对系统表的读取权限。

```
{
   "Version":"2012-10-17",
   "Statement":[
      {
         "Sid":"ReadOnlyAccessTaggedTables",
         "Effect":"Allow",
         "Action":"cassandra:Select",
         "Resource":[
            "arn:aws:cassandra:us-east-2:111122223333:/keyspace/myKeyspace/table/*",
            "arn:aws:cassandra:us-east-2:111122223333:/keyspace/myKeyspace/system*"
         ],
         "Condition":{
            "StringEquals":{
               "aws:ResourceTag/Owner":"${aws:username}"
            }
         }
      }
   ]
}
```

您可以将该策略附加到您账户中的 IAM 用户。如果用户名为richard-roe尝试查看 Amazon Keyspaces 表，必须对该表进行标记Owner=richard-roe要么owner=richard-roe. 否则，他将被拒绝访问。条件标签键 Owner 匹配 Owner 和 owner，因为条件键名称不区分大小写。有关更多信息，请参阅 。IAM JSON 策略元素：Condition在里面IAM 用户指南.

以下策略授予用户创建带有标签的表的权限（如果表的 Owner 包含该用户的用户名的值）。

```
{
      "Version": "2012-10-17",
      "Statement": [
         {
            "Sid": "CreateTagTableUser",
            "Effect": "Allow",
            "Action": ["cassandra:Create", "cassandra:TagResource"],
            "Resource": "arn:aws:cassandra:us-east-2:111122223333:/keyspace/mykeyspace/
table/*",
            "Condition": {
                  "ForAnyValue:StringEquals" : {"aws:RequestTag/Owner": "${aws:username}"}
         }
      ]
}
```

AmazonAmazon Keyspaces（适用于

要向用户、组和角色添加权限，与自己编写策略相比，使用 Amazon 托管策略更简单。创建仅为团队提供所需权限的 IAM 客户托管策略需要时间和专业知识。要快速入门，您可以使用我们的 Amazon 托管式策略。这些策略涵盖常见使用案例，可在您的 Amazon Web Services 账户 中使用。有关 Amazon 托管式策略的更多信息，请参阅 IAM 用户指南中的Amazon 托管式策略。

Amazon Web Services 负责维护和更新 Amazon 托管式策略。您无法更改 Amazon 托管式策略中的权限。服务偶尔会向 Amazon 托管式策略添加额外权限以支持新功能。此类更新会影响附加策略的所有身份（用户、组和角色）。当启动新功能或新操作可用时，服务最有可能会更新 Amazon 托管式策略。服务不会从 Amazon 托管式策略中删除权限，因此策略更新不会破坏您的现有权限。

此外，Amazon 还支持跨多种服务的工作职能的托管式策略。例如，`ViewOnlyAccess` Amazon 托管式策略提供对许多 Amazon Web Services 服务和资源的只读访问权限。当服务启动新功能时，Amazon 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅 IAM 用户指南中的适用于工作职能的 Amazon 托管策略。

Amazon托管策略： AmazonKeyspacesReadOnlyAccess

您可以将 `AmazonKeyspacesReadOnlyAccess` 策略附加得到 IAM 身份。

此策略授予Amazon Keyspaces 的只读访问权限

权限详细信息

此策略包含以下权限。

- `Amazon Keyspaces`— 提供对 Amazon Keyspaces 的只读访问。
- `Application Auto Scaling`— 允许委托人查看来自 Application Auto Scaling 的配置。这是必需的，这样用户才能查看附加到表的自动扩展策略。
- `CloudWatch`— 允许委托人查看在中配置的指标数据和警报 CloudWatch. 这是必需的，以便用户可以查看计费表的大小和 CloudWatch 为表配置的警报。
- `Amazon KMS`— 允许委托人查看中配置的密钥Amazon KMS. 这是必需的，以便用户可以查看Amazon KMS他们在自己的账户中创建和管理的密钥，以确认分配给 Amazon Keyspaces 的密钥是已启用的对称加密密钥。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cassandra:Select"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:DescribeScalableTargets",
                "application-autoscaling:DescribeScalingActivities",
                "application-autoscaling:DescribeScalingPolicies",
                "application-autoscaling:DescribeScheduledActions",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:GetMetricData",
```

```
                "kms:DescribeKey",
                "kms:ListAliases"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon托管策略： AmazonKeyspacesFullAccess

您可以将 AmazonKeyspacesFullAccess 策略附加得到 IAM 身份。

此策略授予管理权限，允许您的管理员不受限制地访问 Amazon Keyspaces。

权限详细信息

此策略包含以下权限。

- Amazon Keyspaces— 允许原则访问任何 Amazon Keyspaces 资源并执行所有操作。
- Application Auto Scaling— 允许委托人创建、查看和删除 Amazon Keyspaces 表的自动扩展策略。这是必需的，这样管理员才能管理 Amazon Keyspaces 表的自动扩展策略。
- CloudWatch— 允许委托人查看可计费表的大小以及创建、查看和删除 CloudWatch Amazon Keyspaces 自动扩展策略的警报。这是必需的，以便管理员可以查看可计费表的大小并创建 CloudWatch 控制面板。
- IAM— 当管理员为表启用应用程序自动扩展时，允许 Amazon Keyspaces 通过 IAM 自动创建服务相关角色。Amazon Keyspaces 可以代表您执行自动扩展操作，这是必需的。
- Amazon KMS— 允许委托人查看中配置的密钥Amazon KMS. 这是必需的，以便用户可以查看Amazon KMS他们在自己的账户中创建和管理的密钥，以确认分配给 Amazon Keyspaces 的密钥是已启用的对称加密密钥。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "cassandra:*"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "application-autoscaling:DeleteScalingPolicy",
    "application-autoscaling:DeleteScheduledAction",
    "application-autoscaling:DeregisterScalableTarget",
    "application-autoscaling:DescribeScalableTargets",
    "application-autoscaling:DescribeScalingActivities",
    "application-autoscaling:DescribeScalingPolicies",
    "application-autoscaling:DescribeScheduledActions",
    "application-autoscaling:PutScheduledAction",
    "application-autoscaling:PutScalingPolicy",
    "application-autoscaling:RegisterScalableTarget",
    "kms:DescribeKey",
```

```
    "kms:ListAliases"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "cloudwatch:DeleteAlarms",
    "cloudwatch:DescribeAlarms",
    "cloudwatch:GetMetricData",
    "cloudwatch:PutMetricAlarm"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": "iam:CreateServiceLinkedRole",
   "Resource": "arn:aws:iam::*:role/aws-service-role/cassandra.application-
autoscaling.amazonaws.com/AWSServiceRoleForApplicationAutoScaling_CassandraTable",
   "Condition": {
    "StringLike": {
     "iam:AWSServiceName": "cassandra.application-autoscaling.amazonaws.com"
    }
   }
  }
 ]
}
```

Amazon KeyspacesAmazon托管策略

查看有关更新的详细信息AmazonAmazon Keyspaces（自从其开始跟踪这些更改以来）的托管策略。要获得有关此页面更改的自动提示，请订阅 文档历史记录 (p. 222) 页面上的 RSS 源。

更改	说明	日期
AmazonKeyspacesReadOnlyAccess (p. 172) – 对现有策略的更新	Amazon Keyspaces 添加了新的权限，允许用户使用查看表的计费大小 CloudWatch. 亚马逊 Keyspaces 与亚马逊集成 CloudWatch 以允许您监控计费表的大小。有关更多信息，请参阅 the section called “Amazon Keyspaces 指标和维度” (p. 181)。	2022 年 7 月 7 日
AmazonKeyspacesFullAccess (p. 173) – 对现有策略的更新	Amazon Keyspaces 添加了新的权限，允许用户使用查看表的计费大小 CloudWatch. 亚马逊 Keyspaces 与亚马逊集成 CloudWatch 以允许您监控计费表的大小。有关更多信息，请参阅 the section called “Amazon Keyspaces 指标和维度” (p. 181)。	2022 年 7 月 7 日

更改	说明	日期
AmazonKeyspacesReadOnlyAccess (p. 170) – 对现有策略的更新	Amazon Keyspaces 添加了新的权限，以允许用户查看Amazon KMS为静态的 Amazon Keyspaces 加密配置的密钥。 Amazon Keyspaces e 静态加密与Amazon KMS用于保护和管理用于加密静态数据的加密密钥。查看Amazon KMS已为 Amazon Keyspaces 配置密钥，已添加只读权限。	2021 年 6 月 1 日
AmazonKeyspacesFullAccess (p. 170) – 对现有策略的更新	Amazon Keyspaces 添加了新的权限，以允许用户查看Amazon KMS为静态的 Amazon Keyspaces 加密配置的密钥。 Amazon Keyspaces e 静态加密与Amazon KMS用于保护和管理用于加密静态数据的加密密钥。查看Amazon KMS已为 Amazon Keyspaces 配置密钥，已添加只读权限。	2021 年 6 月 1 日
Amazon Keyspaces 开始跟踪	Amazon Keyspaces 开始跟踪其Amazon托管策略。	2021 年 6 月 1 日

Amazon Keyspaces 身份和访问

可以使用以下信息，以帮助您诊断和修复在使用 Amazon Keyspaces 和 IAM 时可能遇到的常见问题。

主题

我无权在 Amazon Keyspaces 中执行操作

如果 Amazon Web Services Management Console 告诉您，您无权执行某个操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。

以下示例错误发生在`mateojackson`IAM 用户尝试使用控制台查看有关*##*但没有`cassandra:`*`Select`*表的权限。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 cassandra:Select on resource: mytable
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `cassandra:Select` 操作访问 *mytable* 资源。

我修改了 IAM 用户或角色，但更改没有立即生效

对于已建立与 Amazon Keyspaces 的现有连接的应用程序，IAM 策略更改最多可能需要 10 分钟才能生效。IAM 策略更改在应用程序建立新连接时立即生效。如果您对现有 IAM 用户或角色进行了修改，但修改尚未立即生效，请等待 10 分钟，或者断开连接并重新连接到 Amazon Keyspaces。

我无法使用亚马逊Keyspaces 恢复表 point-in-time 恢复 (PITR)

如果您正在尝试使用以下方法恢复 Amazon Keyspaces 表 point-in-time 恢复 (PITR)，您看到还原过程开始但未成功完成，您可能没有配置还原过程所需的所有必要权限。您必须联系您的管理员以寻求帮助，请求该人员更新您的策略，以允许您在 Amazon Keyspaces 中恢复表。

除了用户权限外，Amazon Keyspaces 可能还需要权限才能在恢复过程中代表您的委托人执行操作。如果表使用客户管理的密钥加密，或者您使用限制传入流量的 IAM 策略，则会出现这种情况。例如，如果您在 IAM 策略中使用条件键将源流量限制到特定的终端节点或 IP 范围，则还原操作将失败。要允许 Amazon Keyspaces 代表您的委托人执行表恢复操作，您必须添加一个`aws:ViaAWSService`IAM 策略中的全局条件密钥。

有关恢复表的权限的更多信息，请参阅the section called “还原权限” (p. 123).

我无权执行 iam：PassRole

如果您收到错误消息，提示您无权执行`iam:PassRole`操作，您必须更新您的策略，以允许您将角色传递给到 Amazon Keyspaces。

有些 Amazon Web Services允许您将现有角色传递到该服务，而不是创建新服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为的 IAM 用户时，会发生以下示例错误`marymajor`尝试使用控制台在 Amazon Keyspaces 中执行操作。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform: iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 Amazon 管理员。管理员是向您提供登录凭证的人。

我想要查看我的访问密钥

在创建 IAM 用户访问密钥后，您可以随时查看您的访问密钥 ID。但是，您无法再查看您的秘密访问密钥。如果您丢失了私有密钥，则必须创建一个新的访问密钥对。

访问密钥包含两部分：访问密钥 ID（例如 `AKIAIOSFODNN7EXAMPLE`）和秘密访问密钥（例如 `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`）。与用户名和密码一样，您必须同时使用访问密钥 ID 和秘密访问密钥对请求执行身份验证。像对用户名和密码一样，安全地管理访问密钥。

> **Important**
>
> 请不要向第三方提供访问密钥，即便是为了帮助找到您的规范用户 ID 也不行。如果您这样做，可能会向某人提供对您的账户的永久访问权限。

当您创建访问密钥对时，系统会提示您将访问密钥 ID 和秘密访问密钥保存在一个安全位置。秘密访问密钥仅在您创建它时可用。如果丢失了您的秘密访问密钥，您必须为 IAM 用户添加新的访问密钥。您最多可拥有两个访问密钥。如果您已有两个密钥，则必须删除一个密钥对，然后再创建新的密钥。要查看说明，请参阅 IAM 用户指南中的管理访问密钥。

我是管理员并希望允许其他人访问 Amazon Keyspaces

要允许其他人访问 Amazon Keyspaces，您必须为需要访问权限的人员或应用程序创建一个 IAM 实体（用户或角色）。它们将使用该实体的凭证访问 Amazon。然后，您必须将策略附加到实体，以便在 Amazon Keyspaces 中向其授予正确的权限。

要立即开始使用，请参阅 IAM 用户指南中的创建您的第一个 IAM 委派用户和组。

我想要允许我的之外的用户Amazon Web Services 账户访问我的亚马逊 Keyspaces 资源

您可以创建一个角色，以便其它账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACL) 的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon Keyspaces 是否支持这些功能，请参阅Amazon Keyspaces 与 IAM (p. 162).
- 要了解如何为您拥有的 Amazon Web Services 账户 中的资源提供访问权限，请参阅 IAM 用户指南中的为您拥有的另一个 Amazon Web Services 账户 中的 IAM 用户提供访问权限。
- 要了解如何为第三方Amazon Web Services 账户提供您的资源的访问权限，请参阅 IAM 用户指南中的为第三方拥有的Amazon Web Services 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南中的为经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅 IAM 用户指南中的 IAM 角色与基于资源的策略有何不同。

对 Amazon Keyspaces 使用服务相关角色

Amazon Keyspaces（针对 Apache Cassandra）使用Amazon Identity and Access Management(IAM)服务相关角色. 服务相关角色是一种独特类型的 IAM 角色，它与 Amazon Keyspaces 直接相关。服务相关角色由 Amazon Keyspaces 预定义，并包含该服务调用其他服务所需的所有权限。Amazon代表您提供服务。

服务相关角色可让您更轻松地设置 Amazon Keyspaces 自动扩展，因为您不必为 Application Auto Scaling 手动添加必要的权限。Amazon Keyspaces 定义其服务相关角色的权限。定义的权限包括信任策略和权限策略，以及不能附加到任何其它 IAM 实体的权限策略。

有关支持服务相关角色的其它服务的信息，请参阅使用 IAM 的 Amazon 服务并查找 Service-Linked Role（服务相关角色）列中显示为 Yes（是）的服务。请选择 Yes 与查看该服务的服务相关角色文档的链接。

Amazon Keyspaces 的服务相关角色权限

Amazon Keyspaces 自动扩展使用名为的服务相关角色AWSServiceRoleForApplicationAutoScaling_CassandraTable以允许 Application Auto Scaling 调用 Amazon Keyspaces 和 AmazonCloudWatch代表您。

AWSServiceRoleForApplicationAutoScaling_CassandraTable 服务相关角色仅信任以下服务来担任该角色：

- `cassandra.application-autoscaling.amazonaws.com`

角色权限策略允许 Application Auto Scaling 对指定的 Amazon Keyspaces 资源完成以下操作：

- 操作：资源 `arn:*:cassandra:*:*:/keyspace/system/table/*` 上的 `cassandra:Select`

- 操作：资源 `arn:*:cassandra:*:*:/keyspace/system_schema/table/*` 上的 `cassandra:Select`
- 操作：资源 `arn:*:cassandra:*:*:/keyspace/system_schema_mcs/table/*` 上的 `cassandra:Select`
- 操作：资源 `arn:*:cassandra:*:*:"*"` 上的 `cassandra:Alter`

为 Amazon Keyspaces 创建服务相关角色

您无需手动为 Amazon Keyspaces 自动扩展创建服务相关角色。当您使用 Amazon Keyspaces 对表启用 Amazon Keyspaces 时，Amazon Web Services Management Console，Amazon CLI，或者 AmazonAPI、Application Auto Scaling 为您创建服务相关角色。

如果删除此服务相关角色，然后需要再次创建，可以使用相同流程在账户中重新创建此角色。当您为表启用 Amazon Keyspaces 自动扩展时，Application Auto Scaling 会再次为您创建服务相关角色。

为 Amazon Keyspaces 编辑服务相关角色

Amazon Keyspaces 不允许您编辑AWSServiceRoleForApplicationAutoScaling_CassandraTable服务相关角色。创建服务相关角色后，将无法更改角色名称，因为可能有多个实体引用该角色。但是可以使用 IAM 编辑角色说明。有关更多信息，请参见《IAM 用户指南》中的编辑服务相关角色。

删除适用于 Amazon Keyspaces 的服务相关角色

如果您不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。但是，您必须首先跨所有区域对账户中的所有表禁用自动扩展，然后才能手动删除服务相关角色。要禁用 Amazon Keyspaces 表的自动扩展，请参阅??? (p. 95).

> Note
>
> 如果在您试图修改资源时 Amazon Keyspaces 自动扩展使用该角色，则取消注册可能会失败。如果发生这种情况，请等待几分钟后重试。

使用 IAM 手动删除服务相关角色

使用 IAM 控制台，即 Amazon CLI 或 Amazon API 来删除 AWSServiceRoleForApplicationAutoScaling_CassandraTable 服务相关角色。有关更多信息，请参见 IAM 用户指南中的删除服务相关角色。

> Note
>
> 要删除 Amazon Keyspaces 自动扩展所使用的服务相关角色，您必须首先对账户中的所有表禁用自动扩展。

支持Amazon Web Services 区域对于 Amazon Keyspaces 服务相关角色

提供 Amazon Keyspaces 自动扩展和 Application Auto Scaling 服务相关角色Amazon Web Services 区域提供 Amazon Keyspaces 的区域。有关更多信息，请参阅 。Amazon Keyspaces 的服务终端节点.

Amazon Keyspaces 中的记录和监控（针对 Apache Cassandra）

监控是保持 Amazon Keyspaces 和您的可靠性、可用性和性能的重要方面。Amazon解决方案。您应从 Amazon 解决方案的所有部分收集监控数据，以便更轻松地调试出现的多点故障。不过，在开始监控 Amazon Keyspaces 之前，您应制定一个监控计划并在计划中回答下列问题：

- 监控目的是什么？
- 您将监控哪些资源？
- 监控这些资源的频率如何？
- 您将使用哪些监控工具？
- 谁负责执行监控任务？
- 出现错误时应通知谁？

为了提供概述，Amazon Web Services Management Consolefor Amazon Keyspaces 提供了一个预配置的仪表板，显示账户中所有表中聚合的延迟和错误。下一步是设置 CloudWatch 针对单个 Amazon Keyspaces 资源的仪表板。首先，通过在不同时间和不同负载条件下测量性能，在您的环境中建立正常 Amazon Keyspaces 性能的基准。在监控 Amazon Keyspaces 时，存储历史监控数据，以便将此数据与当前性能数据进行比较，确定正常性能模式和性能异常，并设计解决问题的方法。

要建立基准，您至少应监控以下各项：

- 系统错误，以便您可以确定是否有任何请求导致了错误。

主题

Amazon Keyspaces 监控工具

Amazon提供各种可以用来监控 Amazon Keyspaces 的工具。您可以配置其中的一些工具来为您执行监控任务，但有些工具需要手动干预。建议您尽可能实现监控任务自动化。

主题

Amazon Keyspaces 自动监控工具

您可以使用以下自动化监控工具来监控 Amazon Keyspaces，并在出现错误时进行报告：

- 亚马逊 CloudWatch 警报— 按您指定的时间段观察单个指标，并根据相对于给定阈值的指标值在若干时间段内执行一项或多项操作。操作是一个发送到 Amazon Simple Notification Service (Amazon SNS) 主题或 Amazon EC2 Auto Scaling 策略的通知。

  CloudWatch 警报将不会调用操作，因为这些操作处于特定状态。该状态必须已改变并在指定的若干个时间段内保持不变。有关更多信息，请参阅 使用 Amazon CloudWatch 监控 Amazon Keyspaces (p. 180)。

Amazon Keyspaces 手动监控工具

监控 Amazon Keyspaces 时的另一个重要环节是手动监控 CloudWatch 警报不包括在内。亚马逊 Keyspaces、CloudWatchTrusted Advisor和其他Amazon Web Services Management Console仪表板提供了 at-a-glance 查看的状态Amazon环境。

- 这些区域有： CloudWatch 主页显示以下内容：

- 当前告警和状态
- 告警和资源图表
- 服务运行状况

此外，您还可以使用 CloudWatch 执行以下操作：

- 创建自定义控制面板以监控您关心的服务
- 绘制指标数据图，以排除问题并弄清楚趋势
- 搜索并浏览您所有的 Amazon 资源指标
- 创建和编辑警报以接收有关问题的通知

使用 Amazon CloudWatch 监控 Amazon Keyspaces

您可以使用 CloudWatch 监控 Amazon Keyspaces，CloudWatch 会收集原始数据，并将数据处理为易读且近乎实时的指标。这些统计数据会保存 15 个月，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。

此外，可以设置用于监测特定阈值的警报，并在达到相应阈值时发送通知或执行操作。有关更多信息，请参阅 。亚马逊 CloudWatch 用户指南.

> Note
>
> 快速开始使用预配置 CloudWatch 显示亚马逊 Keyspaces 的常见指标的仪表板，您可以使用 Amazon CloudFormation从中获得模板https://github.com/aws-samples/amazon-keyspaces-cloudwatch-cloudformation-templates.

主题

如何使用 Amazon Keyspaces 指标？

Amazon Keyspaces 报告的指标您提供可通过不同方式分析的信息。下面的列表显示这些指标的一些常见用途。这些是入门建议，并不全面。有关指标和保留率的更多信息，请参阅指标.

我如何？	相关指标
如何确定是否发生了任何系统错误？	您可以监控 `SystemErrors`，以确定是否有任何请求导致了服务器错误代码。通常，此指标应等于零。如果不是，您可能需要调查。
如何比较平均预置读取容量与使用的读取容量？	监控平均预置读取容量和使用的读取容量 1. 设置Period为了`ConsumedReadCapacityUnits`和`ProvisionedReadCapacityUnits`到希望监控的时间间隔。 2. 将更改为Statistic为了`ConsumedReadCapacityUnits`从`Average`到`Sum`. 3. 新建一个空的 Math expression (数学表达式)。 4. 在详细信息在新数学表达式的部分中，输入Id的`ConsumedReadCapacityUnits`然后将指标除以 CloudWatch 期间指标的函数 (metric_id/ (期间(metric_id))。

我如何？	相关指标
	5. 取消选择 `ConsumedReadCapacityUnits`。 现在，您可以将使用的平均读取容量与预置容量进行比较。有关基本算术函数和如何创建时间序的更多信息，请参阅。使用指标数学.
如何比较平均预置写入容量与使用的写入容量？	监控平均预置写入容量和使用的写入容量 1. 设置Period为了`ConsumedWriteCapacityUnits`和`ProvisionedWriteCapacityUnits`希望监控的时间间隔。 2. 将更改为Statistic为了`ConsumedWriteCapacityUnits`从`Average`到`Sum`. 3. 新建一个空的 Math expression (数学表达式)。 4. 在详细信息在新数学表达式的部分中，输入Id的`ConsumedWriteCapacityUnits`然后将指标除以 CloudWatch 期间指标的函数 (metric_id/ (期间(metric_id))。 5. 取消选择 `ConsumedWriteCapacityUnits`。 现在，您可以将使用的平均写入容量与预置容量进行比较。有关基本算术函数和如何创建时间序的更多信息，请参阅。使用指标数学.

Amazon Keyspaces 指标和维度

与 Amazon Keyspaces 交互时，会向 Amazon CloudWatch 发送以下指标和维度。所有指标每分钟汇总和报告一次。您可以使用以下过程查看 Amazon Keyspaces 的指标。

使用 查看指标 CloudWatch 控制台

指标的分组首先依据服务命名空间，然后依据每个命名空间内的各种维度组合。

1. 打开 CloudWatch 控制台https://console.aws.amazon.com/cloudwatch/.
2. 如果需要，可以更改 区域。在导航栏上，选择您的区域Amazon资源驻留。有关更多信息，请参阅Amazon服务终端节点。
3. 在导航窗格中，选择 Metrics（指标）。
4. 在 All metrics (全部指标) 选项卡下，选择 `AWS/Cassandra.`。

使用Amazon CLI 查看指标

- 在命令提示符处，使用以下命令。

```
aws cloudwatch list-metrics --namespace "AWS/Cassandra"
```

Amazon Keyspaces 指标和维度

Amazon Keyspaces 发送到 Amazon 的指标和维度 CloudWatch 在这里列出。

Amazon Keyspaces 指标

亚马逊 CloudWatch 每隔一分钟汇总一次 Amazon Keyspaces 指标。

并非所有的统计数据，如 Average 或 Sum，都适用于每个指标。不过，所有值均可通过 Amazon Keyspaces 控制台获得，也可通过使用 CloudWatch 控制台，Amazon CLI，或者Amazon所有指标的 SDK。在下表中，每个度量都有一个适用于该度量的有效统计信息列表。

指标	描述
AccountMaxTableLevelReads	账户的表可以使用的最大读取容量单位数。对于按需表，此值将限制表可以使用的最大读取请求单位。 单位：Count 有效统计数据： • Maximum— 账户的表可以使用的最大读取容量单位数。
AccountMaxTableLevelWrites	账户的表可以使用的最大写入容量单位数。对于按需表，此值将限制表可以使用的最大写入请求单位。 单位：Count 有效统计数据： • Maximum— 账户的表可以使用的最大写入容量单位数。
AccountProvisionedReadCapacityUtilization	账户使用的预置读取容量单位百分比。 单位：Percent 有效统计数据： • Maximum – 账户使用的最大预置读取容量单位百分比。 • Minimum – 账户使用的最小预置读取容量单位百分比。 • Average – 账户使用的平均预置读取容量单位百分比。该指标每五分钟发布一次。因此，如果快速调整预置读取容量单位，则此统计数据可能不会反映实际平均值。
AccountProvisionedWriteCapacityUtilization	账户使用的预置写入容量单位百分比。 单位：Percent 有效统计数据： • Maximum – 账户使用的最大预置写入容量单位百分比。 • Minimum – 账户使用的最小预置写入容量单位百分比。 • Average – 账户使用的平均预置写入容量单位百分比。该指标每五分钟发布一次。因此，如果快速调整预置写入容量单位，则此统计数据可能不会反映实际平均值。
BillableTableSizeInBytes	表的可计费大小（以字节为单位）。它是表中所有行的编码大小的总和。此指标可帮助您跟踪随时间的推移表存储成本。 单位：Bytes 维度：Keyspace, TableName 有效统计数据： • Maximum— 表的最大存储大小。 • Minimum— 表的最小存储大小。

指标	描述
	• Average— 表的平均存储大小。此指标的计算间隔为 4-6 小时。
ConditionalCheckFailedRequests	失败的轻量级事务 (LWT) 写入请求数。INSERT、UPDATE 和 DELETE 操作允许提供一个逻辑条件，该条件计算结果必须为 true，才能继续操作。如果此条件计算结果为 false，ConditionalCheckFailedRequests增加 1。 单位：Count 维度：Keyspace, TableName 有效统计数据： • Minimum • Maximum • Average • SampleCount • Sum

指标	描述
ConsumedReadCapacityUnits	在指定时间段内使用的读取容量单位数。有关更多信息，请参阅 。读/写容量模式. Note 要了解每秒平均吞吐量利用率，请使用 Sum 统计数据计算一分钟内消耗的吞吐量。然后，将和值除以每分钟秒数 (60) 来计算每秒的平均 ConsumedReadCapacityUnits（确认此平均值不会突出这一分钟内出现的任何大而短暂的读取活动峰值）。有关将使用的平均读取容量与预置读取容量进行比较的详细信息，the section called “使用指标” (p. 180) 单位：Count 维度：Keyspace, TableName 有效统计数据： • Minimum— 对表的任何单个请求所使用的最小读取容量单位数。 • Maximum— 对表的任何单个请求所使用的最大读取容量单位数。 • Average – 每个请求占用的平均读取容量。 Note Average 值受不活动时间的影响，不活动时的采样值将为零。 • Sum – 占用的总读取容量单位。这是对 ConsumedReadCapacityUnits 指标最有用的统计数据。 • SampleCount— 对 Amazon Keyspaces 的请求数（即使未使用读取容量）。 Note SampleCount 值受不活动时间的影响，不活动时的采样值将为零。

指标	描述
ConsumedWriteCapacityUnits	在指定时间段内使用的写入容量单位数。您可以检索表使用的总写入容量。有关更多信息，请参阅 。读/写容量模式. Note 要了解每秒平均吞吐量利用率，请使用 Sum 统计数据计算一分钟内消耗的吞吐量。然后，将和值除以每分钟秒数 (60) 来计算每秒的平均 ConsumedWriteCapacityUnits（确认此平均值不会突出这一分钟内出现的任何大而短暂的写入活动峰值）。有关将使用的平均写入容量与预置写入容量进行比较的详细信息，the section called “使用指标” (p. 180) 单位：Count 维度：Keyspace, TableName 有效统计数据： • Minimum— 对表的任何单个请求所使用的最小写入容量单位数。 • Maximum— 对表的任何单个请求所使用的最大写入容量单位数。 • Average – 每个请求占用的平均写入容量。 Note Average 值受不活动时间的影响，不活动时的采样值将为零。 • Sum – 占用的总写入容量单位。这是对 ConsumedWriteCapacityUnits 指标最有用的统计数据。 • SampleCount— 对 Amazon Keyspaces 的请求数（即使未使用写入容量）。 Note SampleCount 值受不活动时间的影响，不活动时的采样值将为零。
MaxProvisionedTableReadCapacityUtilization	账户的最高预置读取表所使用的预置读取容量单位的百分比。 单位：Percent 有效统计数据： • Maximum –：账户的最高预置读取表使用的最大预置读取容量单位百分比。 • Minimum – 账户的最高预置读取表使用的最小预置读取容量单位百分比。 • Average – 账户的最高预置读取表使用的平均预置读取容量单位百分比。该指标每五分钟发布一次。因此，如果快速调整预置读取容量单位，则此统计数据可能不会反映实际平均值。

指标	描述
MaxProvisionedTableWriteCapacityUtilization	账户的最高预置写入表所使用的预置写入容量的百分比。 单位：Percent 有效统计数据： • Maximum— 账户的最高预置写入表所使用的预置写入容量单位的最大百分比。 • Minimum— 账户的最高预置写入表所使用的预置写入容量单位的最小百分比。 • Average— 账户的最高预置写入表所使用的预置写入容量单位的平均百分比。该指标每五分钟发布一次。因此，如果快速调整预置写入容量单位，则此统计数据可能不会反映实际平均值。
PerConnectionRequestRateExceeded	对 Amazon Keyspaces 的请求超过每个连接请求费率限额的请求。与 Amazon Keyspaces 的每个客户端连接每秒最多可支持 3000 个 CQL 请求。客户端可以创建多个连接来提高吞吐量。 单位：Count 维度：Keyspace, TableName, Operation 有效统计数据： • SampleCount • Sum
ProvisionedReadCapacityUnits	表的预置读取容量单位数。 TableName 维度返回表的 ProvisionedReadCapacityUnits。 单位：Count 维度：Keyspace, TableName 有效统计数据： • Minimum – 预置读取容量的最低设置。如果使用 ALTER TABLE 增加读取容量，则此指标显示此时间段的预置 ReadCapacityUnits 最低值。 • Maximum – 预置读取容量的最高设置。如果使用 ALTER TABLE 减少读取容量，则此指标显示此时间段的预置 ReadCapacityUnits 最高值。 • Average – 平均预置读取容量。ProvisionedReadCapacityUnits 指标每五分钟发布一次。因此，如果快速调整预置读取容量单位，则此统计数据可能不会反映实际平均值。

指标	描述
ProvisionedWriteCapacityUnits	表的预置写入容量单位数。 TableName 维度返回表的 ProvisionedWriteCapacityUnits。 单位：Count 维度：Keyspace, TableName 有效统计数据： • Minimum – 预置写入容量的最低设置。如果使用 ALTER TABLE 增加写入容量，则此指标显示此时间段的预置 WriteCapacityUnits 最低值。 • Maximum – 预置写入容量的最高设置。如果使用 ALTER TABLE 减少写入容量，则此指标显示此时间段的预置 WriteCapacityUnits 最高值。 • Average – 平均预置写入容量。ProvisionedWriteCapacityUnits 指标每五分钟发布一次。因此，如果快速调整预置写入容量单位，则此统计数据可能不会反映实际平均值。
ReadThrottleEvents	对 Amazon Keyspaces 的请求超出表的预置读取容量。 单位：Count 维度：Keyspace, TableName, Operation 有效统计数据： • SampleCount • Sum
ReturnedItemCount	多行返回的行数SELECT指定时间段内的查询。多行SELECT查询是不包含完全限定主键的查询，例如完整表扫描和范围查询。 返回的 行数并不一定与已计算的行数相同。例如，假设您请求对一个具有 100 行的表执行具有 ALLOW FILTERING 的 SELECT *，但指定 WHERE 子句来缩小结果范围，以便仅返回 15 行。在此情况下，来自 SELECT 的响应包含的 ScanCount 为 100，Count 为返回的 15 行。 单位：Count 维度：Keyspace, TableName, Operation 有效统计数据： • Minimum • Maximum • Average • SampleCount • Sum

指标	描述
StoragePartitionThroughputCapacityExceeded	对超过分区吞吐量的 Amazon Keyspaces 存储分区的请求。Amazon Keyspaces 存储分区最多可以支持每秒 1000 个 WCU/WRU 或每秒 3000 个 RCU/RRU，或者两者的线性组合。我们建议您查看数据模型，以便在更多分区之间分配读/写流量，以减轻这些例外情况。 Note 逻辑 Amazon Keyspaces 分区可以跨越多个存储分区，而且大小几乎没有限制。 单位：Count 维度：Keyspace, TableName, Operation 有效统计数据： • SampleCount • Sum
SuccessfulRequestCount	指定时间段内处理的成功请求数。 单位：Count 维度：Keyspace, TableName, Operation 有效统计数据： • SampleCount
SuccessfulRequestLatency	指定时间段内收到的 Amazon Keyspaces 成功请求。SuccessfulRequestLatency可以提供两种不同类型的信息： • 成功请求的所用时间（Minimum、Maximum、Sum 或 Average）。 • 成功的请求数 (SampleCount)。 SuccessfulRequestLatency仅反映 Amazon Keyspaces 中的活动，不考虑网络延迟或客户端活动。 单位：Milliseconds 维度：Keyspace, TableName, Operation 有效统计数据： • Minimum • Maximum • Average • SampleCount

指标	描述
SystemErrors	对亚马逊 Keyspaces 的请求生成ServerError在指定时间段内。ServerError 通常指示内部服务错误。 单位：Count 维度：Keyspace, TableName, Operation 有效统计数据： • Sum • SampleCount
TTLDeletes	使用生存时间 (TTL) 连续删除或更新数据所消耗的单位。每个TTLDelete提供足够的容量来删除或更新每行最多 1KB 的数据。例如，要更新存储 2.5 KB 数据的行并同时删除该行中的一个或多个列，需要删除 3 个 TTL。或者，要删除包含 3.5 KB 数据的整行，需要删除 4 个 TTL。 单位：Count 维度：Keyspace, TableName 有效统计数据： • Sum— 一个时间段内消耗的 TTLDelete 总数。
UserErrors	对亚马逊 Keyspaces 的请求生成InvalidRequest指定时间段内出现错误。InvalidRequest 通常指示客户端错误，例如，参数组合无效、正在尝试更新不存在的表或请求签名错误。 UserErrors表示当前的无效请求汇总Amazon Web Services区域和最新的Amazon Web Services 账户. 单位：Count 维度：Keyspace, TableName, Operation 有效统计数据： • Sum • SampleCount
WriteThrottleEvents	对 Amazon Keyspaces 的请求超出表的预置写入容量。 单位：Count 维度：Keyspace, TableName, Operation 有效统计数据： • SampleCount • Sum

Amazon Keyspaces 指标的维度

Amazon Keyspaces 的指标由账户、表名或操作进行限定。您可以使用 CloudWatch 控制台，按下表中的任意维度检索 Amazon Keyspaces 数据。

维度	描述
`Keyspace`	该维度将数据限定为特定键空间。该值可以是当前区域和当前区域中的任意键空间。Amazon Web Services 账户.
`Operation`	此维度将数据限定为 Amazon Keyspaces CQL 操作之一，如`INSERT`要么`SELECT`运算。
`TableName`	此维度将数据限制为特定表。该值可以是当前区域和当前区域中的任意表名称。Amazon Web Services 账户. 如果表名称在账户中不是唯一的，则还必须指定 `Keyspace`。

创建 CloudWatch 监控 Amazon Keyspaces 的警报

您可以创建 Amazon CloudWatch Amazon Keyspaces 的告警，在告警改变状态时发送 Amazon Simple Notification Service (Amazon SNS) 消息。告警会监控您指定的时间段内的某个指标。它在多个时间段内根据相对于给定阈值的指标值，执行一项或多项操作。操作是一个发送到 Amazon SNS 主题或 Auto Scaling 策略的通知。

警报只会调用操作进行持续的状态变更。 CloudWatch 警报将不会调用操作，因为这些操作处于特定状态。该状态必须已改变并在指定的若干个时间段内保持不变。

有关创建的更多信息 CloudWatch 警报，请参阅使用 Amazon CloudWatch 警报中的亚马逊 CloudWatch 用户指南.

使用 记录 Amazon Keyspaces API 调用Amazon CloudTrail

亚马逊 Keyspaces 与Amazon CloudTrail，提供用户、角色或用户所执行操作的记录的服务。Amazon亚马逊 Keyspaces 中的服务。CloudTrail将 Amazon Keyspaces 的数据定义语言 (DDL) API 调用捕获为事件。捕获的调用包括来自 Amazon Keyspaces 控制台的调用和对 Amazon Keyspaces API 操作的代码调用。

如果您创建跟踪，则可以使CloudTrail发生到 Amazon Simple Storage Service (Amazon S3) 存储桶的事件，包括 Amazon Keyspaces 的事件。

如果您不配置跟踪，则仍可在 CloudTrail 控制台的 Event history (事件历史记录) 中查看最新事件。使用收集的信息CloudTrail中，您可以确定向 Amazon Keyspaces 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅 Amazon CloudTrail 用户指南。

中的 Amazon Keyspaces 信息CloudTrail

CloudTrail在上启用了Amazon Web Services 账户在您创建账户时。当 Amazon Keyspaces 中发生 DDL 活动时，该活动将记录在CloudTrail活动以及其他Amazon中的服务事件事件史记录. 下表显示了为亚马逊 Keyspaces 记录的 DDL 语句。请注意，数据操作语言 (DML) 语句未登录CloudTrail.

CloudTrail **`eventName`**	Statement	CQL 操作	AmazonSDK 操作
CreateTable	DDL	`CreateTable`	`CreateTable`

CloudTrail **eventName**	Statement	CQL 操作	AmazonSDK 操作
CreateKeyspace	DDL	`CreateKeyspace`	`CreateKeyspace`
DropKeyspace	DDL	`DropKeyspace`	`DeleteKeyspace`
DropTable	DDL	`DropTable`	`DeleteTable`
AlterTable	DDL	`AlterTable`	`UpdateTable`, `TagResource`, `UntagResource`
不CloudTrail事件	DML	`Select`	`GetKeyspace`, `GetTable`, `ListKeyspaces`, `ListTables` `ListTagsForResource`

已登录的每个操作CloudTrail包括始终以 CQL 查询语言表示的请求参数。

您可以在中查看、搜索和下载最新事件Amazon Web Services 账户. 有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录中的事件Amazon Web Services 账户，包括 Amazon Keyspaces 的事件，请创建跟踪。一个踪迹启用CloudTrail将日志文件传送到 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 Amazon 区域。此跟踪记录在 Amazon 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Simple Storage Service（Amazon S3）存储桶。此外，您可以配置其他 Amazon 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。

有关更多信息，请参阅 Amazon CloudTrail 用户指南 中的以下主题：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域中接收 CloudTrail 日志文件
- 从多个账户中接收 CloudTrail 日志文件

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 Amazon Identity and Access Management (IAM) 用户凭证发出的
- 请求是使用角色还是联合身份用户的临时安全凭证发出的
- 请求是否由其它 Amazon 服务发出

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 Amazon Keyspaces 日志文件条目

跟踪记录是一种配置，可用于将事件作为日志文件传送到您指定的 Simple Storage Service（Amazon S3）存储桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公用 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

以下示例显示了一个 CloudTrail 日志条目，该条目演示了 `CreateKeyspace`、`DropKeyspace`、`CreateTable` 和 `DropTable` 操作。

```
{

  "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-2",
      "sourceIPAddress": "52.95.24.72",
      "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-2:111122223333:/keyspace/mykeyspace/"
        }
      ],
      "eventType": "AwsApiCall",
      "apiVersion": "3.4.4",
      "recipientAccountId": "111122223333"
    },
    {
      "eventVersion": "1.05",
      "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AKIAIOSFODNN7EXAMPLE:alice",
        "arn": "arn:aws:sts::111122223333:assumed-role/users/alice",
        "accountId": "111122223333",
        "sessionContext": {
          "sessionIssuer": {
            "type": "Role",
            "principalId": "AKIAIOSFODNN7EXAMPLE",
            "arn": "arn:aws:iam::111122223333:role/Admin",
            "accountId": "111122223333",
            "userName": "Admin"
```

```
          },
          "webIdFederationData": {},
          "attributes": {
            "mfaAuthenticated": "false",
            "creationDate": "2020-01-15T18:47:56Z"
          }
        }
      },
      "eventTime": "2020-01-15T19:28:39Z",
      "eventSource": "cassandra.amazonaws.com",
      "eventName": "DropKeyspace",
      "awsRegion": "us-east-2",
      "sourceIPAddress": "52.95.24.73",
      "userAgent": "Cassandra Client/ProtocolV4",
      "requestParameters": {
        "rawQuery": "DROP KEYSPACE \"mykeyspace\"",
        "keyspaceName": "mykeyspace"
      },
      "responseElements": null,
      "requestID": "66f3d86a-56ae-4c29-b46f-abcd489ed86b",
      "eventID": "e5aebeac-e1dd-41e3-a515-84fe6aaabd7b",
      "readOnly": false,
      "resources": [
        {
          "accountId": "111122223333",
          "type": "AWS::Cassandra::Keyspace",
          "ARN": "arn:aws:cassandra:us-east-2:111122223333:/keyspace/mykeyspace/"
        }
      ],
      "eventType": "AwsApiCall",
      "apiVersion": "3.4.4",
      "recipientAccountId": "111122223333"
    },
    {
      "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-2",
      "sourceIPAddress": "52.95.24.72",
      "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-2:111122223333:/keyspace/mykeyspace/table/
mytable"
        }
      ],
      "eventType": "AwsApiCall",
      "apiVersion": "3.4.4",
      "recipientAccountId": "111122223333"
    },
    {
      "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-2",
      "sourceIPAddress": "52.95.24.66",
      "userAgent": "Cassandra Client/ProtocolV4",
      "requestParameters": {
        "rawQuery": "DROP TABLE \"mykeyspace\".\"mytable\"",
        "keyspaceName": "mykeyspace",
        "tableName": "mytable"
      },
      "responseElements": null,
      "requestID": "025501b0-3582-437e-9d18-8939e9ef262f",
      "eventID": "1a5cbedc-4e38-4889-8475-3eab98de0ffd",
      "readOnly": false,
      "resources": [
        {
          "accountId": "111122223333",
          "type": "AWS::Cassandra::Table",
          "ARN": "arn:aws:cassandra:us-east-2:111122223333:/keyspace/mykeyspace/table/
mytable"
        }
      ],
      "eventType": "AwsApiCall",
      "apiVersion": "3.4.4",
      "recipientAccountId": "111122223333"
```

```
    }
  ]

}
```

Amazon Keyspaces（针对 Apache Cassandra）合规

作为多项计划的一部分，第三方审计员将评估 Amazon Keyspaces（针对 Apache Cassandra）的安全性和合规性Amazon合规性计划 其中包括：

- ISO/IEC 27001:2013、27017:2015、27018:2019 和 ISO/IEC 9001:2015
- 系统和组织控制 (SOC)
- 支付卡行业 (PCI)
- 联邦风险与授权管理项目 (FedRAMP) High
- 健康保险流通与责任法案 (HIPAA)

要了解是亚马逊Keyspaces 还是其他Amazon Web Services在特定合规计划的范围内，请参阅Amazon Web Services合规性计划范围内的服务. 有关常规信息，请参阅Amazon Web Services合规性计划。

您可以使用 Amazon Artifact 下载第三方审计报告。有关更多信息，请参阅、在 Amazon Artifact 中下载报告。

您使用 Amazon Web Services 的合规性责任取决于您数据的敏感度、贵公司的合规性目标以及适用的法律法规。Amazon 提供以下资源来帮助满足合规性：

- 安全性与合规性快速入门指南安全性与合规性快速入门指南 – 这些部署指南讨论了架构注意事项，并提供了在Amazon上部署注重安全性和合规性的基准环境的步骤。
- Amazon Web Services 上的 HIPAA 安全性和合规性架构设计– 该白皮书介绍了公司如何使用 Amazon Web Services 创建符合 HIPAA 标准的应用程序。

  Note

  并非所有 Amazon Web Services都符合 HIPAA 要求。有关更多信息，请参阅符合 HIPAA 要求的服务参考。
- Amazon合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- Amazon Config 开发人员指南中的使用规则评估资源 – 此 Amazon Config 服务评估您的资源配置对内部实践、行业指南和法规的遵循情况。
- Amazon Security Hub：此 Amazon Web Service 提供了 Amazon 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践规范。

亚马逊 Keyspaces 中的弹性和灾难恢复

Amazon全球基础设施围绕Amazon Web Services 区域和可用区构建。Amazon Web Services 区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

Amazon Keyspaces 将数据自动复制三次多次Amazon在相同区域内可用区Amazon Web Services 区域以实现持久性和高可用性。

有关 的更多信息Amazon Web Services 区域和可用区，请参阅Amazon全球基础设.

除了Amazon全球基础设施，Amazon Keyspaces 提供时间点恢复 (PITR)，以帮助支持您的数据恢复能力和备份需求。

通过提供表数据的持续备份，PITR 有助于保护 Amazon Keyspaces 表免遭意外写入或删除操作。有关更多信息，请参阅 。亚马逊 Keyspaces 间的时间点恢复.

Amazon Keyspaces 中的基础设施安全性

作为托管服务，Amazon Keyspaces（针对 Apache Cassandra）Amazon中描述的全局网络安全程序Amazon Web Services：安全流程概述白皮书。

你用Amazon发布的 API 调用通过网络访问 Amazon Keyspaces。客户端必须支持传输层安全性 (TLS) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

Amazon Keyspaces 支持两种验证客户端请求的方法。第一种方法使用特定于服务的凭据，这些凭据是为特定特定生成的基于密码Amazon Identity and Access Management(IAM) 用户。您可以使用 IAM 控制台、Amazon CLI，或者AmazonAPI。有关更多信息，请参阅 。将 IAM 与 Amazon Keyspaces 结合使用.

第二种方法将身份验证插件用于开源。 DataStax 卡桑德拉的 Java 驱动程序。此插件启用IAM 用户、角色和联合身份将身份验证信息添加到 Amazon Keyspaces（针对 Apache Cassandra）API 请求Amazon签名版本 4 流程 (SigV4). 有关更多信息，请参阅 ??? (p. 15)。

您可以使用接口 VPC 终端节点，以防止 Amazon VPC 和 Amazon Keyspaces 之间的流量离开 Amazon 网络。接口 VPC 终端节点由提供支持Amazon PrivateLink，一个Amazon实现之间私人通信的技术AmazonAmazon VPC 中的私有 IP 结合使用 elastic network interface 的服务。有关更多信息，请参阅the section called “使用 接口 VPC 终端节点” (p. 196)。

对接口 VPC 终端节点一起使用

接口 VPC 终端节点可在 Amazon VPC 中运行的虚拟私有云 (VPC) 与 Amazon Keyspaces 之间实现私有通信。接口 VPC 终端节点由Amazon PrivateLink，这是Amazon支持 VPC 和之间的私人通信的服务Amazon服务。

Amazon PrivateLink通过对 VPC 中的私有 IP 使用 elastic network interface 启用此功能，以便网络流量不会离开 Amazon 网络。接口 VPC 终端节点不需要 Internet 网关、NAT 设备、VPN 连接或 Amazon Direct Connect 连接。有关更多信息，请参阅 。Amazon Virtual Private Cloud和接口 VPC 终端节点 (Amazon PrivateLink) .

主题

对 Amazon Keyspaces 使用接口 VPC 终端节点

您可以创建接口 VPC 终端节点，以便 Amazon Keyspaces 与 Amazon VPC 资源之间的流量开始流过接口 VPC 终端节点。要开始使用，请按照以下步骤创建接口终端节点。接下来，编辑与您在上一步中创建的终端节点关联的安全组，并为端口 9142 配置入站规则。有关更多信息，请参阅 。添加、删除和更新规则.

填充system.peers带接口 VPC 终端节点信息的表条目

Apache Cassandra 司机使用system.peers表以查询有关集群的节点信息。Cassandra 驱动程序使用节点信息对连接进行负载平衡和重试操作。亚马逊 Keyspaces 填充了system.peers表自动用于通过公共终端节点连接的客户端。

为了向通过接口 VPC 终端节点进行连接的客户端提供类似的功能，Amazon Keyspaces 填充system.peers账户中的表，其中包含可用 VPC 终端节点的每个可用区的条目。VPC 在system.peers表中，Amazon Keyspaces 要求您授予用于连接到 Amazon Keyspaces 的 IAM 实体访问权限，以便查询 VPC 以获取终端节点和网络接口信息。

> **Important**
>
> 填充system.peers带有可用接口 VPC 终端节点的表可改进负载平衡并提高读/写吞吐量。建议所有使用接口 VPC 终端节点访问 Amazon Keyspaces 的客户端使用，Apache Spark 是必需的。

要授予用于连接到 Amazon Keyspaces 的 IAM 实体查找必要的接口 VPC 终端节点信息的权限，您可以更新现有的 IAM 角色或用户策略，或者创建新的 IAM 策略，如以下示例所示。

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

要确认策略已正确设置，请查询system.peers表来查看网络信息。如果system.peers表为空，它可能表示策略未成功配置，或者您已超过了DescribeNetworkInterfaces和DescribeVPCEndpointsAPI 操作。DescribeVPCEndpoints落入Describe*类别，被认为是非变异动作.DescribeNetworkInterfaces属于的子集未过滤和未分页的非变异操作，并适用不同的配额。有关更多信息，请参阅 。请求令牌桶大小和补充费率Amazon EC2 API 参考中。

如果您确实看到一张空表，请在几分钟后重试以排除请求费率配额问题。如果查询从表中返回结果，则表明您的策略已正确配置。

控制对 Amazon Keyspaces 的接口 VPC 终端节点的访问

通过 VPC 终端节点，您可以通过两种方式控制对资源的访问：

- IAM 策略— 您可以控制允许通过特定的 VPC 终端节点访问 Amazon Keyspaces 的请求、用户或组。您可以通过在附加到 IAM 用户、组或角色的策略中使用条件键来完成此操作。
- VPC 策略— 您可以通过向 VPC Keyspace 资源附加策略来控制哪些终端节点可以访问您的 Amazon Keyspaces 资源。要限制仅允许通过特定 VPC 终端节点的流量来访问特定键空间或表，请编辑限制资源访问的现有 IAM 策略并添加该 VPC 终端节点。

下面是访问 Amazon Keyspaces 资源的终端节点示例。

- IAM 策略示例：除非流量来自指定的 VPC 终端节点，否则限制对特定 Amazon Keyspaces 表的所有访问— 此示例策略可以附加到 IAM 用户、角色或组。它限制对指定 Amazon Keyspaces 表的访问，除非传入流量来自指定的 VPC 终端节点。

```
{
   "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

要限制对特定表的访问，还必须包括对系统表的访问权限。系统表为只读。

- VPC 策略示例：只读访问权限— 此示例策略可以附加到 VPC 终端节点。（有关更多信息，请参阅控制对 Amazon VPC 资源的访问)。它限制对 Amazon Keyspaces 资源的只读访问，以便通过所附加到的 VPC 终端节点访问 Amazon Keyspace 资源。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "ReadOnly",
      "Principal": "*",
      "Action": [
        "cassandra:Select"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

- VPC 策略示例：限制对特定 Amazon Keyspaces 表的访问— 此示例策略可以附加到 VPC 终端节点。它限制通过所附加到的 VPC 终端节点访问特定表。

```
{
   "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

要限制对特定表的访问，还必须包括对系统表的访问权限。系统表为只读。

可用性

Amazon Keyspaces 支持在所有Amazon Web Services 区域提供服务的位置。有关更多信息，请参阅 ??? (p. 21)。

VPC 终端节点和 Amazon Keyspaces 时间点恢复 (PITR)

如果您使用 IAM 策略条件键为了限制传入流量，表还原操作可能会失败。例如，如果您使用将源流量限制到特定 VPC 终端节点`aws:SourceVpce`条件键，表还原操作失败。要允许 Amazon Keyspaces 代表您的委托人执行还原操作，您必须添加`aws:ViaAWSService`IAM 策略的条件密钥。这些区域有：`aws:ViaAWSService`条件密钥允许在任何时候访问Amazon服务使用委托人的凭证发出请求。有关更多信息，请参阅 。IAM JSON 策略：条件键中的IAM 用户指南. 下面的策略是此示例。

```
{
   "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"
            }
         }
      }
   ]
}
```

常见错误和警告

如果您使用的是 Amazon Virtual Private Cloud 并连接到亚马逊 Keyspaces，则可能会看到以下警告。

```
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.
```

出现此警告的原因是`system.peers`表中包含 Amazon Keyspaces 有权查看的所有 Amazon VPC 终端节点的条目，包括您通过连接的 Amazon VPC 终端节点。您可以放心地忽略此警告。

Amazon Keyspaces 的配置和漏洞分析

Amazon 负责处理基本安全任务，如来宾操作系统 (OS) 和数据库补丁、防火墙配置和灾难恢复等。这些流程已通过相应第三方审核和认证。有关更多详细信息，请参阅以下 资源：

- 责任共担模式
- Amazon Web Services：安全流程概述（白皮书）

Amazon Keyspaces 的安全最佳实践

Amazon Keyspaces（适用于 Apache Cassandra）提供了在您开发和实施自己的安全策略时需要考虑的大量安全功能。以下最佳实践是一般指导原则，并不代表完整安全解决方案。这些最佳实践可能不适合您的环境或不满足您的环境要求，请将其视为有用的考虑因素而不是惯例。

主题

Amazon Keyspaces 的预防性安全最佳实践

以下安全最佳实践被认为是预防性的，因为这些实践可以帮助您预测和预防 Amazon Keyspaces 中的安全事件。

静态使用加密

Amazon Keyspaces 使用存储在中的加密密钥对存储在表中的所有用户数据进行静态加密。Amazon Key Management Service(Amazon KMS)。保护您的数据免受未经授权访问，为基础存储提供额外一层数据保护。

默认情况下，亚马逊 Keyspaces 使用Amazon 拥有的密钥用于加密所有表格。如果此密钥不存在，将创建该密钥。无法禁用服务默认密钥。

或者，您也可以使用客户托管密钥进行静态加密。有关更多信息，请参阅 。静态 Amazon Keyspaces 加密.

使用 IAM 角色对 Amazon Keyspaces 的访问进行身份验证

对于用户、应用程序和其他Amazon要访问亚马逊 Keyspaces 的服务，它们必须包含有效Amazon他们的凭证AmazonAPI 请求。不应直接在应用程序或 EC2 实例中存储 Amazon 凭证。这些长期凭证不会自动轮换，如果外泄，可能造成重大业务影响。利用 IAM 角色，可以获得临时访问密钥，用于访问 Amazon 服务和资源。

有关更多信息，请参阅 IAM 角色。

使用 IAM 策略进行 Amazon Keyspaces 基本授权

在授予权限时，您要决定谁获得权限，获得对哪些 Amazon Keyspaces API 的权限，以及您允许对这些资源执行的具体操作。实施最低权限对于减小安全风险以及错误或恶意意图造成的影响至关重要。

将权限策略附加到 IAM 身份（即用户、组和角色），从而授予对 Amazon Keyspaces 资源执行操作的权限。

可以通过以下方法实现：

- Amazon托管（预定义）策略
- 客户托管策略

使用 IAM 策略条件进行精细访问控制

在 Amazon Keyspaces 中授予权限时，可以指定确定权限策略如何生效的条件。实施最低权限对于减小安全风险以及错误或恶意意图造成的影响至关重要。

使用 IAM 策略授予权限时，可以指定条件。例如，可以：

- 授予权限以允许用户对特定密钥空间或表进行只读访问。
- 根据用户身份授予权限，以允许用户对特定表进行写访问。

有关更多信息，请参阅 。基于身份的策略示例.

考虑客户端加密

如果您将敏感或机密数据存储在 Amazon Keyspaces 中，您可能希望将该数据加密到尽可能靠近其源的位置，以便在该数据的整个生命周期内对其进行保护。加密传输中和静态敏感数据有助于确保明文数据不会提供给任何第三方。

Amazon Keyspaces 的 Detective 性安全最佳实践

以下安全最佳实践被认为是检测性的，因为这些实践可以帮助发现潜在安全弱点和事件。

使用Amazon CloudTrail监控Amazon Key Management Service(Amazon KMS）Amazon KMS密钥用法

如果您使用的是客户托管Amazon KMS密钥对于静态加密，将此密钥的使用登录到中。Amazon CloudTrail. CloudTrail 按照帐户上执行的记录操作，显示用户活动。CloudTrail 记录有关每个操作的重要信息，包括谁发出请求、所使用的服务、执行的操作、操作的参数以及 Amazon 服务返回的响应元素。这些信息有助于跟踪 Amazon 资源更改，解决运营问题。利用 CloudTrail，可以更轻松确保符合内部策略和法规标准。

可以使用 CloudTrail 审计密钥使用情况。CloudTrail 创建日志文件，其中包含账户的 Amazon API 调用和相关事件历史记录。这些日志文件包括所有Amazon KMS使用控制台发出的 API 请求，Amazon以及通过集成的 SDK 和命令行工具，以及通过集成的方式生产的工具Amazon服务。您可以使用这些日志文件来获取有关何时间的信息。Amazon KMS使用密钥、请求的操作、请求者的身份、发出请求的 IP 地址等。有关更多信息，请参见用 Amazon CloudTrail 记录 Amazon Key Management Service API 调用和 Amazon CloudTrail 用户指南。

使用 CloudTrail 监控 Amazon Keyspaces 数据定义语言 (DDL) 操作

CloudTrail 按照帐户上执行的记录操作，显示用户活动。CloudTrail 记录有关每个操作的重要信息，包括谁发出请求、所使用的服务、执行的操作、操作的参数以及 Amazon 服务返回的响应元素。此信息可帮助您跟踪对您的更改。Amazon资源并解决操作问题。利用 CloudTrail，可以更轻松确保符合内部策略和法规标准。

所有 Amazon KeyspacesDL 操作 (p. 207)将自动登录 CloudTrail。使用 DDL 操作，您可以创建和管理 Amazon Keyspaces 密钥空间和表格。

当 Amazon Keyspaces 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他活动一起保存在 CAmazon事件历史记录中的服务事件。有关更多信息，请参阅 。使用以下方式记录 Amazon Keyspaces 操作Amazon CloudTrail. 您可以在中查看、搜索和下载最新事件。Amazon Web Services 账户. 有关更多信息，请参阅 Amazon CloudTrail 用户指南中的使用 CloudTrail 事件历史记录查看事件。

要持续记录在中的事件Amazon Web Services 账户，包括亚马逊 Keyspaces 的活动，创建踪迹. 跟踪记录让 CloudTrail 可以将日志文件发送至 Amazon Simple Storage Service (Amazon S3) 存储桶。默认情

况下，在控制台中创建跟踪时，此跟踪应用于所有Amazon Web Services 区域. 跟踪记录 Amazon 分区所有区域的事件，将日志文件传送至指定的 S3 存储桶。此外，可以配置其他 Amazon 服务，进一步分析和应对 CloudTrail 日志中收集的事件数据。

标记 Amazon Keyspaces 资源以进行标识和自动化

可以将自己的元数据以标签形式分配给 Amazon 资源。每个标签都是一个标注，包含一个客户定义的密钥和一个可选值，让您能够更轻松地管理、搜索和筛选资源。

标签可实现分组控制。尽管没有固有类型的标签，但利用标签，可以根据用途、所有者、环境或其他条件分类资源。下面是一些示例：

- 访问权限 — 用于基于标签控制对 Amazon Keyspaces 资源的访问。有关更多信息，请参阅the section called “基于 Amazon Keyspaces 标签的” (p. 165)。
- 安全性 — 用于确定数据保护设置等要求。
- 机密性 — 资源支持的特定数据机密级别的标识符。
- 环境 – 用于区分开发、测试和生产基础设施。

有关更多信息，请参阅 。Amazon标记策略和向资源添加标签和标签.

Amazon Keyspaces（针对 Apache Cassandra）

连接到亚马逊Keyspaces（适用于 Apache Cassandra）终端节点后，您可以使用 Cassandra 查询语言 (CQL) 来处理您的数据库。CQL 在许多方面与结构化查询语言 (SQL) 相似。

主题

亚马逊Keyspaces 中的 Cassandra 查询语言 (CQL) 元素

了解亚马逊Keyspaces 支持的卡桑德拉查询语言 (CQL) 元素，包括标识符、常量、术语和数据类型。

主题

标识符

标识符（或名称）用于标识表、列和其他对象。标识符可以带引号，也可以不带引号。以下情况将适用。

```
identifier          ::=  unquoted_identifier | quoted_identifier
unquoted_identifier ::=  re('[a-zA-Z][a-zA-Z0-9_]*')
quoted_identifier   ::=  '"' (any character where " can appear if doubled)+ '"'
```

常量

定义以下常量。

```
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+
```

条款

术语表示受支持的值的类型。术语由以下内容定义。

```
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
```

数据类型

Amazon Keyspaces 支持以下数据类型。

字符串类型

数据类型	描述
ascii	表示 ASCII 字符串。
text	表示 UTF-8 编码的字符串。
varchar	表示 UTF-8 编码的字符串（varchar 是 text 的别名）。

数字类型

数据类型	描述
bigint	表示 64 位有符号长整型。
counter	表示 64 位有符号整数计数器。有关更多信息，请参阅 the section called “计数器” (p. 204)。
decimal	表示可变精度小数。
double	表示 64 位 IEEE 754 浮点。
float	表示 32 位 IEEE 754 浮点。
int	表示 32 位有符号整数。
varint	表示任意精度的整数。

计数器

counter 列包含 64 位有符号整数。计数器值通过 the section called “更新” (p. 216) 语句进行递增或递减，并且无法直接设置。这使得 counter 列对于跟踪计数很有用。例如，您可以使用计数器来跟踪日志文件中的条目数或某个社交网络上的文章被查看的次数。以下限制适用于 counter 列：

- 类型 counter 的列不能是表的 primary key 的一部分。
- 在包含一个或多个类型 counter 的列的表中，所有列的类型都必须为 counter。

如果计数器更新失败（例如，由于超时或与 Amazon Keyspaces 的连接中断），客户端不知道计数器值是否已更新。如果重试更新，则可能会再次应用对计数器值的更新。

Blob 类型

数据类型	描述
`blob`	表示任意字节。

布尔值类型

数据类型	描述
`boolean`	表示 `true` 或 `false`。

与时间相关的类型

数据类型	描述
`timestamp`	表示时间戳。
`timeuuid`	表示版本 1 UUID。

集合类型

数据类型	描述
`list`	表示文本元素的有序集合。
`map`	表示键/值对的无序集合。
`set`	表示一个或多个文本元素的无序集合。
`tuple`	表示文本元素的有界组。

其他类型

数据类型	描述
`inet`	一个表示 IPv4 或 IPv6 格式的 IP 地址的字符串。

亚马逊Keyspaces 数据类型的 JSON 编码

亚马逊 Keyspaces 提供与 Apache Cassandra 相同的 JSON 数据类型映射。下表介绍Amazon Keyspaces 接受的数据类型`INSERT JSON`语句以及 Amazon Keyspaces 在返回数据时使用的数据类型`SELECT JSON`语句。

对于单字段数据类型，例如float,int,UUID，以及date，你也可以将数据作为string. 用于复合数据类型和集合，例如tuple,map，以及list，你也可以以 JSON 或编码形式插入数据JSON string.

JSON 数据类型	在以下版本中受理的数据类型：**INSERT JSON**声明	返回的数据类型**SELECT JSON**声明	注意
ascii	string	string	使用 JSON 字符转义\u.
bigint	integer, string	integer	字符串必须是有效的 64 位整数。
blob	string	string	字符串应以开头0x后面是偶数的十六进制数字。
boolean	boolean, string	boolean	字符串必须是其中之一true要么false.
date	string	string	格式化日期YYYY-MM-DD，时区世界标准时间。
decimal	integer, float, string	float	在客户端解码器中可以超过 32 位或 64 位 IEEE-754 浮点精度。
double	integer, float, string	float	字符串必须是有效的整数或浮点数。
float	integer, float, string	float	字符串必须是有效的整数或浮点数。
inet	string	string	IPv4 或 IPv6 地址。
int	integer, string	integer	字符串必须是有效的 32 位整数。
list	list, string	list	使用原生 JSON 列表表示形式。
map	map, string	map	使用原生 JSON 地图表示法。
smallint	integer, string	integer	字符串必须是有效的 16 位整数。
set	list, string	list	使用原生 JSON 列表表示形式。
text	string	string	使用 JSON 字符转义\u.
time	string	string	Time of day dayHH-MM-SS[.fffffffff].
timestamp	integer, string	string	时间戳。字符串常量允许您将时间戳存储为日期。带格式的日期戳YYYY-MM-DD HH:MM:SS.SSS返回。

JSON 数据类型	在以下版本中受理的数据类型：**INSERT JSON**声明	返回的数据类型**SELECT JSON**声明	注意
timeuuid	string	string	键入 1 UUID。请参阅constants (p. 203)用于 UUID 格式。
tinyint	integer, string	integer	字符串必须是有效的 8 位整数。
tuple	list, string	list	使用原生 JSON 列表表示形式。
uuid	string	string	请参阅constants (p. 203)用于 UUID 格式。
varchar	string	string	使用 JSON 字符转义\u.
varint	integer, string	integer	可变长度；可能会溢出客户端解码器中的 32 位或 64 位整数。

亚马逊Keyspaces 中的 DDL 语句（数据定义语言）

数据定义语言(DDL) 是一组卡桑德拉查询语言 (CQL) 语句，用于管理亚马逊Keyspaces（适用于 Apache Cassandra）中的数据结构，例如密钥空间和表。可以使用 DDL 创建这些数据结构，在创建数据结构后对其进行修改，以及在不再使用数据结构时将其删除。亚马逊Keyspaces 异步执行 DDL 操作。请参阅the section called “异步创建和删除键空间和表” (p. 7).

支持以下 DDL 语句：

- CREATE KEYSPACE (p. 208)
- ALTER KEYSPACE (p. 208)
- DROP KEYSPACE (p. 209)
- 创建表 (p. 209)
- ALTER TABLE (p. 211)
- 还原表 (p. 213)
- DROP 子句 (p. 213)

主题

- Keyspaces (p. 207)
- 表 (p. 209)

Keyspaces

键空间 对与一个或多个应用程序相关的表进行分组。就关系数据库管理系统 (RDBMS) 而言，键空间与数据库、表空间或类似的结构大致相似。

Note

在 Apache Cassandra 中，键空间将决定如何在多个存储节点之间复制数据。但是，亚马逊 Keyspaces 是一项完全托管的服务：其存储层的详细信息由您管理。因此，Amazon Keyspaces 中的密钥空间仅是逻辑结构，与底层物理存储无关。

有关 Amazon Keyspaces 密钥空间的配额限制和限制的信息，请参阅配额 (p. 219).

以下是键空间唯一可用的复制策略：

`SingleRegionStrategy`— 在其区域的三个可用区之间复制数据。

CREATE KEYSPACE

使用 `CREATE KEYSPACE` 语句可创建新的键空间。

语法

```
create_keyspace_statement ::=
    CREATE KEYSPACE [ IF NOT EXISTS ] keyspace_name
    WITH options
```

其中：

- *keyspace_name* 是要创建的键空间的名称。
- 选项 为下列一个或多个项：
  - `REPLICATION`— 表示密钥空间复制策略的映射 (`SingleRegionStrategy`)，必要时添加其他值。（必需）
  - `DURABLE_WRITES`— 写入 Amazon Keyspaces 始终是持久的，因此不需要使用此选项。但是，如果指定此选项，则值必须为 `true`。
  - `TAGS`— 创建时要附加到资源的键值对标签列表。

示例

创建键空间，如下所示。

```
CREATE KEYSPACE "myGSGKeyspace"
    WITH REPLICATION = {'class': 'SingleRegionStrategy'} and TAGS ={'key1':'val1',
 'key2':'val2'} ;
```

ALTER KEYSPACE

使用 `ALTER KEYSPACE` 可在键空间中添加或删除标签。

语法

```
alter_keyspace_statement ::=
    ALTER KEYSPACE keyspace_name
    [[ADD | DROP] TAGS
```

其中：

- *keyspace_name* 是要更改的键空间的名称。
- `TAGS`— 要在键盘空间添加或删除的键值对标签列表。

示例

更改键空间，如下所示。

```
ALTER KEYSPACE "myGSGKeyspace" ADD TAGS {'key1':'val1', 'key2':'val2'};
```

DROP KEYSPACE

使用DROP KEYSPACE语句用于删除键空间，包括其所有内容，例如表。

语法

```
drop_keyspace_statement ::=
    DROP KEYSPACE [ IF EXISTS ] keyspace_name
```

其中：

- keyspace_name 是要删除的键空间的名称。

示例

```
DROP KEYSPACE "myGSGKeyspace";
```

表

表是亚马逊Keyspaces 中的主要数据结构。表中的数据按行和列进行组织。这些列的子集用于通过指定分区键来确定分区（以及最终的数据放置）。

可以将另一组列定义为聚类列，这意味着它们可以作为谓词参与查询执行。

默认情况下，将创建具有按需 吞吐容量的新表。您可以更改新表和现有表的容量模式。有关读/写容量吞吐量模式的更多信息，请参阅 the section called “读/写容量模式” (p. 88)。

有关 Amazon Keyspaces 表的配额限制和限制的信息，请参阅配额 (p. 219).

CREATE TABLE

使用 CREATE TABLE 语句可创建新表。

语法

```
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 [ STATIC ][ PRIMARY KEY]

primary_key            ::=  partition_key [ ',' clustering_columns ]

partition_key          ::=  column_name
                              | '(' column_name ( ',' column_name )* ')'

clustering_columns     ::=  column_name ( ',' column_name )*
```

```
table_options          ::=  [table_options]
                              | CLUSTERING ORDER BY '(' clustering_order
 ')' [ AND table_options ]
                              | options
                              | CUSTOM_PROPERTIES
                              | WITH TAGS
                              | WITH default_time_to_live

clustering_order       ::=  column_name (ASC | DESC) ( ',' column_name (ASC | DESC) )*
```

其中：

- *table_name* 是要创建的表的名称。
- *column_definition* 包含以下各项：
  - *column_name*— 列的名称。
  - *cql_type*— Amazon Keyspaces 数据类型（参见数据类型 (p. 204)）。
  - *STATIC*— 将此列指定为静态列。静态列存储由同一分区中的所有行共享的值。
  - *PRIMARY KEY*— 将此列指定为表的主键。
- *primary_key* 包含以下各项：
  - partition_key
  - clustering_columns
- *partition_key*:
  - 分区键可以是单个列，也可以是由两列或多个列组成的复合值。主键的分区键部分是必需的，它决定了 Amazon Keyspaces 存储数据的方式。
- *clustering_columns*:
  - 主键的可选聚类列部分决定如何在每个分区中聚类和排序数据。
- *table_options*由以下各项组成：
  - *CLUSTERING ORDER BY*— 表上的默认集群顺序由你在表中的集群密钥组成ASC（升序）排序方向。指定它可覆盖默认排序行为。
  - *CUSTOM_PROPERTIES*— 特定于亚马逊Keyspaces 的设置地图。
    - capacity_mode：指定表的读/写吞吐容量模式。选项为 throughput_mode:PAY_PER_REQUEST 和 throughput_mode:PROVISIONED。预置容量模式要求将 read_capacity_units 和 write_capacity_units 作为输入。默认为 throughput_mode:PAY_PER_REQUEST。
    - encryption_specification：为静态加密指定加密选项。如果未指定，则默认值为encryption_type:AWS_OWNED_KMS_KEY. 加密选项客户管理的密钥需要Amazon KMS以Amazon 资源名称 (ARN) 格式键入作为输入。kms_key_identifier:ARN：kms_key_identifier:ARN.
    - point_in_time_recovery：指定是否 point-in-time 对表启用或禁用了还原。选项为 status:enabled 和 status:disabled。如果未指定，则默认值为status:disabled.
    - ttl: 为表启用生存时间自定义设置。要启用，请使用status:enabled. 默认为 status:disabled。晚于ttl您无法为表禁用它。
  - TAGS— 创建资源时要附加到资源的键值对标签列表。
  - default_time_to_live— 表格的默认生存时间设置（以秒为单位）。
- *clustering_order* 包含以下各项：
  - *column_name*— 列的名称。
  - *ASC | DESC*— 设置上升点 (ASC) 或后代 (DESC) 订单修改器。如果未指定，默认顺序则默认值为 ASC。

示例

```
CREATE TABLE IF NOT EXISTS "my_keyspace".my_table (
```

```
                                            id text,
                                            name text,
                                            region text,
                                            division text,
                                            project text,
                                            role text,
                                            pay_scale int,
                                            vacation_hrs float,
                                            manager_id text,
                                            PRIMARY KEY (id,division))
                                            WITH CUSTOM_PROPERTIES={
                                                'capacity_mode':{
                                                        'throughput_mode': 'PROVISIONED',
 'read_capacity_units': 10, 'write_capacity_units': 20
                                                        },
                                                'point_in_time_recovery':{'status':
 'enabled'},
                                                'encryption_specification':{
                                                        'encryption_type':
 'CUSTOMER_MANAGED_KMS_KEY',

 'kms_key_identifier':'arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111'
                                                        }
                                            }
                                            AND CLUSTERING ORDER BY (division ASC)
                                            AND TAGS={'key1':'val1', 'key2':'val2'}
                                            AND default_time_to_live = 3024000;
```

在使用聚类列的表中，可以在表定义中将非聚类列声明为静态列。请参阅the section called “静态列” (p. 103).

示例

```
CREATE TABLE "my_keyspace".my_table (
                                            id int,
                                            name text,
                                            region text,
                                            division text,
                                            project text STATIC,
                                            PRIMARY KEY (id,division));
```

ALTER TABLE

使用ALTER TABLE语句用于添加新列、添加标签或更改表的自定义属性。

语法

```
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* 是要更改的表的名称。

- *column_definition* 是要添加的列和数据类型的名称。
- *column_definition_list* 是放在括号内的列的逗号分隔列表。
- *TAGS* 是要附加到资源的键/值对标签的列表。
- default_time_to_live：表格的默认生存时间设置（以秒为单位）。
- *table_options*由以下各项组成：
  - *CUSTOM_PROPERTIES*— 亚马逊Keyspaces 特定的设置地图。
    - capacity_mode：指定表的读/写吞吐容量模式。选项为 throughput_mode:PAY_PER_REQUEST 和 throughput_mode:PROVISIONED。预置容量模式要求将 read_capacity_units 和 write_capacity_units 作为输入。默认为 throughput_mode:PAY_PER_REQUEST。
    - encryption_specification：为静态加密指定加密选项。选项为 encryption_type:AWS_OWNED_KMS_KEY 和 encryption_type:CUSTOMER_MANAGED_KMS_KEY。加密选项客户管理的密钥需要Amazon KMS以Amazon 资源名称 (ARN) 格式键入作为输入。kms_key_identifier:ARN.
    - point_in_time_recovery：指定是否 point-in-time 启用或禁用了表还原功能。选项为 status:enabled 和 status:disabled。默认为 status:disabled。
    - ttl: 为表启用生存时间自定义设置。要启用，请使用status:enabled. 默认为 status:disabled。晚于ttl您无法为表禁用它。

> **Note**
>
> 使用 ALTER TABLE，您只能更改单个自定义属性。不能在同一个语句中组合多个 ALTER TABLE 命令。

示例

以下语句说明如何将列添加到现有表中。

```
ALTER TABLE mykeyspace.mytable ADD (ID int);
```

要更改表的容量模式并指定读取和写入容量单位，可以使用以下语句。

```
ALTER TABLE mykeyspace.mytable WITH CUSTOM_PROPERTIES={'capacity_mode':{'throughput_mode':
 'PROVISIONED', 'read_capacity_units': 10, 'write_capacity_units': 20}};
```

以下语句为表指定了客户托管的 KMS 密钥。

```
ALTER TABLE mykeyspace.mytable WITH CUSTOM_PROPERTIES={
              'encryption_specification':{
                      'encryption_type': 'CUSTOMER_MANAGED_KMS_KEY',
                      'kms_key_identifier':'arn:aws:kms:eu-
west-1:5555555555555:key/11111111-1111-111-1111-111111111111'
                  }
         };
```

启用 point-in-time 为表还原可以使用以下语句。

```
ALTER TABLE mykeyspace.mytable WITH CUSTOM_PROPERTIES={'point_in_time_recovery': {'status':
 'enabled'}};
```

要为表设置默认的生存时间值（以秒为单位），可以使用以下语句。

```
ALTER TABLE my_table WITH default_time_to_live = 2592000;
```

此语句启用表格的自定义 “生存时间” 设置。

```
ALTER TABLE mytable WITH CUSTOM_PROPERTIES={'ttl':{'status': 'enabled'}};
```

还原表

使用RESTORE TABLE语句。本语句要求 point-in-time 要在表上启用恢复。有关更多信息，请参阅 时间点恢复 (p. 121)。

语法

```
restore_table_statement ::=
    RESTORE TABLE table_name FROM TABLE table_name
                    [ WITH table_options [ , ... ] ];
```

其中：

- *restored_table_name*是已还原的表的名称。
- *source_table_name*是源表的名称。
- *table_options* 包含以下各项：
  - *restore_timestamp*是 ISO 8601 格式的还原点时间。如果未指定，则使用当前时间戳。
  - *CUSTOM_PROPERTIES*— 亚马逊Keyspaces 特定的设置地图。
    - capacity_mode：指定表的读/写吞吐容量模式。选项为 throughput_mode:PAY_PER_REQUEST 和 throughput_mode:PROVISIONED。预置容量模式要求将 read_capacity_units 和 write_capacity_units 作为输入。默认为源表中的当前设置。
    - encryption_specification：为静态加密指定加密选项。选项为 encryption_type:AWS_OWNED_KMS_KEY 和 encryption_type:CUSTOMER_MANAGED_KMS_KEY。加密选项客户管理的密钥需要Amazon KMS以Amazon 资源名称 (ARN) 格式键入作为输入。kms_key_identifier:ARN. 将使用客户托管密钥加密的表还原到使用Amazon 拥有的密钥，亚马逊Keyspaces 需要访问Amazon KMS源表的密钥。
    - point_in_time_recovery：指定是否 point-in-time 启用或禁用了表还原功能。选项为 status:enabled 和 status:disabled。与创建新表时不同，还原表的默认状态是status:enabled因为设置继承自源表。要对恢复的表禁用 PITR，必须设置status:disabled明确地。
  - *TAGS* 是要附加到资源的键/值对标签的列表。

> **Note**
>
> 已删除的表只能恢复到删除时。

示例

```
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 语句可从键空间中删除表。

语法

```
drop_table_statement ::=
    DROP TABLE [ IF EXISTS ] table_name
```

其中：

- `IF EXISTS` 防止 `DROP TABLE` 在表不存在的情况下失败。(可选)
- *`table_name`* 是要删除的表的名称。

示例

```
DROP TABLE "myGSGKeyspace".employees_tbl;
```

亚马逊Keyspaces 中的 DML 语句（数据操作语言）

数据操作语言(DML) 是一组用于管理Amazon Keyspaces（针对 Apache Cassandra）表中的数据的 Cassandra 查询语言 (CQL) 表中的数据的 Cassandra 语句。可以使用 DML 语句在表中添加、修改或删除数据。

还可以使用 DML 语句查询表中的数据。（请注意，CQL 不支持联接或子查询。）

主题

SELECT

使用 SELECT 语句可查询数据。

语法

```
select_statement ::=  SELECT  [ JSON ] ( select_clause | '*' )
                      FROM table_name
                      [ WHERE where_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         ::=  '=' | '<' | '>' | '<=' | '>=' | CONTAINS | CONTAINS KEY
ordering_clause  ::=  column_name [ ASC | DESC ] ( ',' column_name [ ASC | DESC ] )*
```

示例

```
SELECT name, id, manager_id FROM "myGSGKeyspace".employees_tbl ;
```

```
SELECT JSON name, id, manager_id FROM "myGSGKeyspace".employees_tbl ;
```

有关将 JSON 编码的数据类型映射到 Amazon Keyspaces 数据类型的表，请参阅the section called “亚马逊 Keyspaces 数据类型的 JSON 编码” (p. 205).

代币

您可以应用TOKEN函数，将转到PARTITION KEY列中受益SELECT和WHERE子句。使用TOKEN函数，Amazon Keyspaces 根据映射的令牌值返回行PARTITION_KEY而不是以价值为依据PARTITION KEY.

示例

```
SELECT TOKEN(id) from myTable;

SELECT TOKEN(id) from myTable where TOKEN(id) > 100 and TOKEN(id) < 10000;
```

对结果进行排序

这些区域有：ORDER BY子句指定返回结果的排序顺序排序顺序。它将列名列表以及每列的排序顺序作为参数。您只能在排序子句中指定聚类列。不允许使用非群集列。排序顺序选项是ASC用于升序和DESC用于降序排序顺序。如果省略排序顺序，则使用聚类列的默认顺序。有关可能的排序顺序，请参阅the section called “排序结果” (p. 107).

示例

```
SELECT name, id, division, manager_id FROM "myGSGKeyspace".employees_tbl WHERE id =
 '012-34-5678' ORDER BY division;
```

> **Note**
>
> 为了与既定的 Cassandra 驱动程序行为兼容，当您通过 Cassandra 驱动程序和开发者工具使用 Cassandra 查询语言 (CQL) API 调用对系统表执行操作时，不强制执行基于标签的授权策略。有关更多信息，请参阅 the section called “ 基于标签的亚马逊Keyspaces 资源访问权限” (p. 170)。

INSERT

使用 INSERT 语句可向表添加行。

语法

```
insert_statement ::=  INSERT INTO table_name ( names_values | json_clause )
                      [ IF NOT EXISTS ]
names_values     ::=  names VALUES tuple_literal
json_clause      ::=  JSON string [ DEFAULT ( NULL | UNSET ) ]
names            ::=  '(' column_name ( ',' column_name )* ')'
```

示例

```
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') ;
```

JSON 支持

有关将 JSON 编码的数据类型映射到 Amazon Keyspaces 数据类型的表，请参阅the section called “亚马逊 Keyspaces 数据类型的 JSON 编码” (p. 205).

您可以使用JSON用于插入 a 的关键字JSON-编码的地图为单行。对于表中存在但在 JSON 插入语句中省略的列，请使用DEFAULT UNSET以保留现有值。使用DEFAULT NULL在每行省略的列中写入一个 NULL 值并覆盖现有值（收取标准写入费用）。DEFAULT NULL是默认选项。

示例

```
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 数据包含重复的密钥，则 Amazon Keyspaces 会存储密钥的最后一个值（类似于 Apache Cassandra）。在以下示例中，重复密钥是id，值234-56-7890使用。

示例

```
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"}';
```

更新

使用 UPDATE 语句可修改表中的行。

语法

```
uupdate_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
```

示例

```
UPDATE "myGSGKeyspace".employees_tbl SET pay_scale = 5 WHERE id = '567-89-0123' AND
 division = 'Marketing' ;
```

要递增 counter，请使用以下语法。有关更多信息，请参阅 the section called “计数器” (p. 204)。

```
UPDATE ActiveUsers SET counter = counter + 1  WHERE user = A70FE1C0-5408-4AE3-
BE34-8733E5K09F14 AND action = 'click';
```

删除

使用 DELETE 语句可从表中删除行。

语法

```
delete_statement ::=  DELETE [ simple_selection ( ',' simple_selection ) ]
                      FROM table_name

                      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* 是包含要删除的行的表。

示例

```
DELETE manager_id FROM "myGSGKeyspace".employees_tbl WHERE id='789-01-2345' AND
 division='Executive' ;
```

亚马逊Keyspaces 中的内置函数

Amazon Keyspaces（适用于 Apache Cassandra）支持各种内置函数，您可以在卡桑德拉查询语言 (CQL) 语句中使用这些函数。

主题

- 标量函数 (p. 217)

标量函数

一个标量函数对单个值执行计算，并将结果作为单个值返回。亚马逊Keyspaces 支持以下标量函数。

函数	描述
blobAsType	返回指定数据类型的值。
cast	将一种本机数据类型转换为另一种本机数据类型。
currentDate	将当前日期/时间作为日期返回。
currentTime	将当前日期/时间作为时间返回。

函数	描述
currentTimestamp	将当前日期/时间作为时间戳返回。
currentTimeUUID	返回 timeuuid 形式的当前日期/时间。
fromJson	将 JSON 字符串转换为选定列的数据类型。
maxTimeuuid	返回尽可能大的值timeuuid用于时间戳或日期字符串。
minTimeuuid	返回可能的最小值timeuuid用于时间戳或日期字符串。
now	返回新的唯一 timeuuid。支持：INSERT,UPDATE，以及DELETE语句，并作为的一部分WHERE子句SELECT语句。
toDate	将 timeuuid 或时间戳转换为日期类型。
toJson	以 JSON 格式返回所选列的列值。
token	返回分区键的哈希值。
toTimestamp	将 timeuuid 或日期转换为时间戳。
typeAsBlob	将指定的数据类型转换为blob.
toUnixTimestamp	将 timeuuid 或时间戳转换为 bigInt。
uuid	返回一个随机版本 4 的 UUID。支持：INSERT,UPDATE，以及DELETE语句，并作为的一部分WHERE子句SELECT语句。
dateOf	（已弃用）提取 timeuuid 的时间戳，并将该值作为日期返回。
unixTimestampOf	（已弃用）提取 timeuuid 的时间戳，并将值作为原始 64 位整数时间戳返回。

Amazon Keyspaces（针对 Apache Cassandra）的配额

本节介绍 Amazon Keyspaces（针对 Apache Cassandra）的当前配额和默认值。

主题

Amazon Keyspaces 服务配额

下表包含 Amazon Keyspaces（针对 Apache Cassandra）配额和默认值。有关配额的更多信息，请联系 Amazon Web Services Support.

配额	说明	默认 Amazon Keyspaces
每个最大键空间数Amazon Web Services 区域	每个区域此订阅者的最大键空间数。此默认值可在AmazonService Quotas控制台。	256
每个最大表数Amazon Web Services 区域	每个区域此订阅者在所有键空间中的最大表数。此默认值可在AmazonService Quotas控制台。	256
最大表架构大小	表架构的最大大小	350 KB
最大并发 DDL 操作数	每个区域此订阅者允许的最大并发 DL 操作数。	50
每个连接的最大查询数	每秒单个客户端 TCP 连接可处理的最大 CQL 查询数。	3000
最大行大小	行的最大大小，不包括静态列数据。有关详细信息，请参阅 the section called “计算行大小” (p. 106)。	1MB
每个逻辑分区的最大静态数据量	逻辑分区中静态数据的最大聚合大小。有关详细信息，请参阅 the section called “计算每个逻辑分区的静态列大小” (p. 103)。	1MB
每秒最大读取吞吐量	每秒最大读取吞吐量（读取请求单位 (RU) 或读取容量单位 (RU) 或读取容量单位 (RCU) —— 可以分配给每个区域的表。此默认值可在AmazonService Quotas控制台。	40000

配额	说明	默认 Amazon Keyspaces
每秒最大写入吞吐量	每秒最大写入吞吐量（写请求单元 (WRU) 或写入容量单位 (WCU)——可以分配给每个区域的表。此默认值可在AmazonService Quotas控制台。	40000
账户级读取吞吐量（预置）	为每个区域的账户分配的最大聚合读取容量单位 (RCU) 数，仅适用于处于预置读/写容量模式的表。此默认值可在AmazonService Quotas控制台。	80,000
账户级写入吞吐量（预置）	为每个区域的账户分配的最大聚合写入容量单位 (WCU) 数，仅适用于处于预置读/写容量模式的表。此默认值可在AmazonService Quotas控制台。	80,000
最大分区键大小	复合分区键的最大大小。为元数据分区键中包含的每一列的原始大小添加最多 3 字节的附加存储空间。	2048 字节
最大聚类键大小	所有聚类列的最大组合大小。为元数据每一聚类列的原始大小添加最多 4 字节的附加存储空间。	850 字节
使用时间点恢复（PITR）的最大并发表恢复数	每个订阅者使用 PITR 进行的并发表还原的最大数量为 4 个。此默认值可在AmazonService Quotas控制台。	4
使用恢复的最大数据量 point-in-time 恢复（PITR）	24 小时内使用 PITR 可恢复的最大数据大小。此默认值可在AmazonService Quotas控制台。	5 TB

增加或减小吞吐量（对于预配置表）

增加预置吞吐量

你可以增加`ReadCapacityUnits`要么`WriteCapacityUnits`根据需要使用控制台或`ALTER TABLE`网页。新设置在 `ALTER TABLE` 操作完成后才会生效。

添加预置容量时不能超过每个账户的配额，且可以根据需要为表增加预置容量。有关每个账户的配额的更多信息，请参阅上一部分 the section called “Amazon Keyspaces 服务配额” (p. 219)。

减少预置吞吐量

对于每张桌子`ALTER TABLE`声明，你可以减少`ReadCapacityUnits`要么`WriteCapacityUnits`（或两者兼而有）。新设置在 `ALTER TABLE` 操作完成后才会生效。每天的任何时间最多可执行 4 次减小操作。天依

据通用协调时间 (UTC) 定义。此外，如果过去 1 小时内未执行减小操作，则可以执行额外的减小操作。这实际上将每日的减小操作的最大次数设置为 27 次（在第一个小时内为 4 次减小操作，对于一天内的每个后续 1 小时时段，为 1 次减小操作）。

静态 Amazon Keyspaces 加密

您可以在Amazon拥有的Amazon KMSKey 和客户托管Amazon KMS从创建表开始，每 24 小时时段内最多四次。此外，如果过去 6 小时内未执行任何更改，则可以执行额外的更改。这实际上将每日的更改操作的最大次数设置为 8 次（在前 6 个小时内为 4 次更改操作，对于一天内的每个后续 6 小时时段，为 1 次更改操作）。

您可以更改加密选项以使用Amazon拥有的Amazon KMS根据需要，即使先前的配额已用尽。

相关配额如下，但您也可以请求提高配额。要请求提高服务配额，请参阅 https://aws.amazon.com/support。

Amazon Keyspaces（针对 Apache Cassandra）的文档

下表列出了自 Amazon Keyspaces (for Apache Cassandra) 以来对文档所做的重要更改。如需对此文档更新的通知，您可以订阅 RSS 源。

- 文档最新更新时间：2022 年 8 月 8 日

update-history-change	update-history-description	update-history-date
Keyspaces 中可用Amazon GovCloud (US)	Amazon Keyspaces 目前在中可用Amazon GovCloud (US) Region并且在 Fedramp-High 合规的范围内。有关可用 Apaces 的信息，请参阅Amazon GovCloud (US) RegionFIPS 终端节点.	2022 年 8 月 4 日
使用亚马逊监控亚马逊 Keyspaces 表存储成本 CloudWatch	Amazon Keyspaces 现在可以帮助您监控和跟踪一段时间内的表存储成本`BillableTableSizeInBytes` CloudWatch 指标。	2022 年 6 月 14 日
亚马逊Keyspaces 现在支持 Terraform	现在，您可以使用 Terraform 在 Amazon Keyspaces 中执行数据定义语言 (DDL) 操作。	2022 年 6 月 9 日
Amazon Keyspaces`token`函数支持	Amazon Keyspaces 现在可以帮助您优化应用程序查询，方法是使用`token`函数。	2022 年 4 月 19 日
亚马逊Keyspaces 与 Apache Spark 集成	Amazon Keyspaces 现在可以帮助你使用开源 Apache Spark 更轻松地读取和写入数据Spark 卡桑德拉连接器.	2022 年 4 月 19 日
Amazon Keyspaces	Amazon Keyspaces 支持使用控制平面操作来管理密钥空间和表Amazon开发工具和Amazon CLI. API 参考指南详细描述了支持的控制平面操作。	2022 年 3 月 2 日
如何解决使用 Amazon Keyspaces 时的常见配置问题。	详细了解如何解决您在使用 Amazon Keyspaces 时可能遇到的常见配置问题。	2021 年 11 月 22 日
Amazon Keyspaces 支持生存时间 (TTTL)。	Amazon Keyspaces 生存时间 (TTL) 可通过自动使表中的数据过期，帮助您简化应用程序逻辑并优化存储价格。	2021 年 10 月
使用 DBlk 将数据迁移到Amazon Keyspaces	Step-by-step 使用将数据从 Apache Cassandra 迁移到亚马逊	2021 年 8 月 9 日

	Keyspaces 的教程 DataStax 批量加载器 (dsBulk)。	
Amazon Keyspaces 支持 VPC 终端节点条目在`system.peers`table。	亚马逊Keyspaces 允许您填充`system.peers`包含可用接口 VPC 终端节点信息的表，用于改善负载均衡和提高读/写吞吐量。	2021 年 7 月 29 日
更新 IAM 托管策略以支持客户管理Amazon KMS钥匙。	Amazon Keyspaces 的 IAM 托管策略现在包括列出和查看可用客户管理的权限Amazon KMS密钥存储在Amazon KMS.	2021 年 6 月 1 日
Amazon KeyspacesAmazon KMS钥匙。	Amazon Keyspaces 允许您控制客户托管的Amazon KMS密钥存储在Amazon KMS用于静加加密。	2021 年 6 月 1 日
亚马逊Keyspaces 支持 JSON 语法	Amazon Keyspaces 支持插入和选择操作的 JSON 语法，可帮助您更轻松地读写 JSON 文档。	2021 年 1 月 21 日
Amazon Keyspaces 支持静态列	Amazon Keyspaces 现在可通过使用静态列帮助您高效地更新和存储多行之间的常用数据。	2020 年 11 月 9 日
GA 发布了对亚马逊Keyspaces 的 NoSQL Workbench 支持	NoSQL Workbench 是一款客户端应用程序，可帮助您更轻松地为 Amazon Keyspaces 设计和可视化非关系数据模型。NoSQL WorkbmacOS 和 Linux 上受支持。	2020 年 10 月 28 日
预览版 NoSQL Workbench 对亚马逊Keyspaces 的支持	NoSQL Workbench 是一款客户端应用程序，可帮助您更轻松地为 Amazon Keyspaces 设计和可视化非关系数据模型。NoSQL WorkbmacOS 和 Linux 上受支持。	2020 年 10 月 5 日
编程访问亚马逊Keyspaces 的新代码示例	我们将继续添加用于以编程方式访问 Amazon Keyspaces 的代码示例。支持 Apache Cassandra 版本 3.11.2 的 Java、Python、Go、C# 和 Perl Cassandra 驱动程序的示例现已推出。	2020 年 7 月 17 日
Amazon Keyspaces point-in-time 恢复 (PITR)	Amazon Keyspaces point-in-time 恢复 (PITR) 通过为您提供表数据的持续备份，帮助保护您的表免受意外写入或删除操作的影响。	2020 年 7 月 9 日

Amazon Keyspaces	借助 Amazon Keyspaces（在预览版中以前称为亚马逊托管 Apache Cassandra Service (MCS)），您可以使用卡桑德拉查询语言 (CQL) 代码、Apache 2.0（获得许可的 Cassandra 驱动程序）和你现在已经使用的开发者工具。	2020 年 4 月 23 日
Amazon Keyspaces	Amazon Keyspaces（适用于 Apache Cassandra）与 Application Auto Scaling 集成，通过自动调整吞吐容量，帮助您高效地为可变工作负载预置吞吐容量，从而响应实际的应用程序流量。	2020 年 4 月 23 日
Amazon Keyspaces	Amazon Keyspaces 提供服务和 VPC 之间的私密通信，因此网络流量不会脱离 Amazon 网络。	2020 年 4 月 16 日
基于标签的访问策略	现在，您可以使用 IAM 策略中的资源标签来管理对 Amazon Keyspaces 的访问权限。	2020 年 4 月 8 日
计数器数据类型	Amazon Keyspaces 现在可通过使用计数器来帮助您协调列值的增量和递减。	2020 年 4 月 7 日
为资源添加标签	Amazon Keyspaces 现在允许您使用标签对资源进行标记和分类。	2020 年 3 月 31 日
Amazon CloudFormation 支持	Amazon Keyspaces 目前通过使用可帮助您自动创建和管理资源 Amazon CloudFormation.	2020 年 3 月 25 日
Support IAM 角色和策略以及 Sigv4 身份验证	添加了有关如何使用的信息Amazon Identity and Access Management(IAM)管理 Amazon Keyspaces 的访问权限以及如何使用 Amazon Keyspaces 的身份验证插件 DataStax 适用于 Cassandra 的 Java Driver 使用 IAM 角色和联合身份以编程方式访问亚马逊Keyspaces。	2020 年 3 月 17 日
读/写容量模式	Amazon Keyspaces 现在支持两种读/写吞吐量容量模式。读取/写入容量模式控制如何对读取和写入吞吐量收费以及如何管理表吞吐量容量。	2020 年 2 月 20 日
首次发布	本文档涵盖Amazon Keyspaces（针对 Apache Cassandra）的初次发布。	2019 年 12 月 3 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。