aws

开发人员指南

Amazon MemoryDB

Amazon MemoryDB: 开发人员指南

Table of Contents

MemoryDB 是什么 1
MemoryDB 的特征 1
MemoryDB 核心组件 2
集群 2
Nodes 4
分片 4
参数组 4
子网组 4
访问控制列表 5
Users 5
相关服务 5
选择区域和可用区 5
找到您的节点 7
支持的区域和端点 8
访问 MemoryDB 11
MemoryDB 安全性 11
开始使用 MemoryDB 13
步骤 1：设置 13
注册获取 AWS 账户 13
创建具有管理访问权限的用户 14
授权以编程方式访问 15
设置权限（仅限新的 MemoryDB 用户） 17
下载和配置 AWS CLI 17
步骤 2：创建集群 18
创建 MemoryDB 集群 18
设置身份验证 27
步骤 3：授予对集群的访问权限 28
步骤 4：连接到集群 30
查找您的集群端点 30
连接到 MemoryDB 集群（Linux） 30
步骤 5：删除集群 32
后续步骤 34
管理节点 35
MemoryDB 节点和分片 35

受支持的节点类型 36
预留节点 38
预留节点概述 38
产品类型 39
大小灵活的预留节点 39
将节点从 Redis OSS 升级到 Valkey 41
删除预留节点 41
使用预留节点 42
替换节点 49
管理集群 51
数据分层 52
最佳实践 52
数据分层限制 53
数据分层定价 53
数据分层监控 53
数据分层功能的使用 54
将数据从快照还原到集群 55
准备集群 57
确定要求 57
创建集群 60
查看集群的详细信息 61
修改集群 65
如何触发从 Redis OSS 到 Valkey 的跨引擎升级 67
从集群中添加/移除节点 69
访问您的集群 71
授予对集群的访问权限 71
从外部 AWS 访问 MemoryDB 73
查找连接端点 79
分片 82
查找分片的名称 82
管理您的 MemoryDB 实施 87
引擎版本 87
MemoryDB 7.3 87
Valkey 7.2.6 88
Redis OSS 7.0（加强版） 89
Redis OSS 7.0（加强版） 89

Redis OSS 6.2（加强版） 90
升级引擎版本 91
开始使用 JSON 93
JSON 数据类型概述 93
支持的 命令 104
标记 MemoryDB 资源 145
使用标签监控成本 150
使用AWS CLI管理标签 151
使用 MemoryDB API 管理标签 154
管理维护 156
最佳实践 158
恢复能力 159
最佳实践：发布/订阅和增强型 I/O 多路复用 161
最佳实践：在线调整集群大小 161
了解 MemoryDB 复制 162
一致性 162
在集群中的复制 162
利用多可用区最大限度减少停机时间 164
更改副本数量 171
快照和还原 181
约束 182
成本 182
计划自动快照 183
手动创建快照 184
生成最终快照 187
描述快照 189
复制快照 192
导出快照 195
从快照还原 204
使用快照为集群做种 209
标记快照 215
删除快照 216
扩展 217
扩展 MemoryDB 集群 218
使用参数组配置引擎参数 238
参数管理 239

参数组层 240
创建参数组 240
按名称列出参数组 245
列出参数组的值 250
修改参数组 251
删除参数组 253
引擎特定参数 255
受限命令 270
教程：配置 Lambda 函数，以便在 Amazon VPC 中访问 MemoryDB 270
步骤 1：创建集群 271
第 2 步：创建 Lambda 函数 274
步骤 3：测试 Lambda 函数 277
步骤 4：清除（可选） 278
向量搜索 280
向量搜索概述 280
索引和键空间 281
索引字段类型 281
向量索引算法 282
向量搜索查询表达式 282
INFO 命令 285
向量搜索安全 287
使用案例 287
检索增强生成（RAG） 287
持久语义缓存 288
欺诈侦测 289
其他使用案例 289
向量搜索功能和限制 290
向量搜索可用性 290
参数限制 290
扩展限制 291
操作限制 291
快照 import/export 和实时迁移 291
内存消耗 291
在回填期间内存不足 294
事务 294
创建已启用向量搜索的集群 294

使用 AWS 管理控制台 294
使用 AWS Command Line Interface 295
向量搜索命令 296
FT.CREATE 296
FT.SEARCH 300
FT.AGGREGATE 302
FT.DROPINDEX 303
FT.INFO 304
FT._LIST 306
FT.ALIASADD 306
FT.ALIASDEL 306
FT.ALIASUPDATE 307
FT._ALIASLIST 307
FT.PROFILE 307
FT.EXPLAIN 308
FT.EXPLAINCLI 308
MemoryDB 多区域 309
先决条件和限制 309
工作原理 311
一致性和冲突解决 312
CRDT 及示例 313
通过控制台使用 MemoryDB 多区域 316
在 MemoryDB 多区域中创建新集群 316
将快照还原到多区域集群内的新集群或现有集群 318
修改 MemoryDB 多区域中的集群 321
删除 MemoryDB 多区域中的集群 324
通过 CLI 使用 MemoryDB 多区域 327
创建带有内存DBMulti 区域的集群 327
更新多区域集群 328
扩展 MemoryDB 集群 328
正在删除 MemoryDB 多区域中的集群 328
监控 MemoryDB 多区域 329
MemoryDB 多区域扩缩 330
支持与不支持的命令 331
安全性 335
数据保护 335

MemoryDB 中的数据安全性 336
静态加密 338
传输中加密（TLS） 340
使用对用户进行身份验证 ACLs 341
使用 IAM 进行身份验证 354
Identity and access management 361
受众 362
使用身份进行身份验证 362
使用策略管理访问 363
MemoryDB 如何与 IAM 协同工作 364
基于身份的策略示例 372
问题排查 374
访问控制 376
有关管理访问的概述 377
日志记录和监控 404
使用 CloudWatch 进行监控 404
监控事件 422
使用 AWS CloudTrail 记录 MemoryDB API 调用 433
合规性验证 439
基础结构安全性 440
互联网络流量隐私 440
MemoryDB 和 Amazon VPC 440
子网和子网组 451
MemoryDB API 和接口 VPC 端点（AWS PrivateLink） 464
修复安全漏洞 467
服务更新 468
管理服务更新 468
应用服务更新 472
使用 AWS CLI 474
参考 475
使用 MemoryDB API 476
使用查询 API 476
可用的库 479
对应用程序进行问题排查 479
限额 481
文档历史记录 483

..... cdlxxxv

MemoryDB 是什么

Amazon MemoryDB 是一项耐用的内存数据库服务，可提供超快性能。它专为采用微服务架构的现代应用程序而构建。

Amazon MemoryDB 与热门的开源数据存储 Valkey 和 Redis OSS 兼容，使您能够使用他们已使用的同样灵活友好的数据结构、API 和命令来快速构建应用程序。借助 MemoryDB，您的所有数据都存储在内存中，从而使您能够实现微秒级读取和个位数毫秒级写入延迟和高吞吐量。MemoryDB 还使用多可用区事务日志跨多个可用区（AZ）持久存储数据，以实现快速失效转移、数据库恢复和节点重新启动。

MemoryDB 兼具内存中性能和多可用区持久性，可用作微服务应用程序的高性能主数据库，无需单独管理缓存和耐用数据库。

主题

- MemoryDB 的特征
- MemoryDB 核心组件
- 相关服务
- 选择区域和可用区
- 访问 MemoryDB
- MemoryDB 安全性

MemoryDB 的特征

Amazon MemoryDB 是一项耐用的内存数据库服务，可提供超快性能。MemoryDB 的特征包括：

- 主节点强一致性以及保证副本节点最终一致性。有关更多信息，请参阅 一致性。
- 微秒级读取和个位数毫秒级写入延迟，每个集群高达 1.6 亿 TPS。
- 灵活友好的 Valkey 和 Redis OSS 数据结构和 API。几乎不进行任何修改就可以轻松构建新应用程序或迁移现有的基于 Valkey 和基于 Redis OSS 的应用程序。
- 使用多可用区事务日志实现数据持久性，提供快速的数据库恢复和重启。
- 多可用区可用性，具有自动失效转移和自动检测和恢复节点故障的功能。
- 通过添加和移除节点轻松进行横向扩展，或者通过移动到较大或较小的节点类型轻松进行纵向扩展。您可以通过添加分片扩展写入吞吐量，通过添加副本扩展读取吞吐量。

- 主节点的先写后读一致性以及保证副本节点的最终一致性。
- MemoryDB 支持传输中的加密、静态加密以及通过 使用访问控制列表对用户进行身份验证 () ACLs 对用户进行身份验证。
- 在 Amazon S3 中的自动快照的保留期可长达 35 天。
- 每个集群最多支持 500 个节点和超过 100 TB 的存储空间（每个分片 1 个副本）。
- 使用 TLS 进行传输中加密，使用 AWS KMS 密钥进行静态加密。
- 使用 Valkey 和 Redis OSS 使用访问控制列表对用户进行身份验证 () ACLs 进行用户身份验证和授权。
- 支持 AWS Graviton2 实例类型。
- 出于监控、安全性和通知而与其他 AWS 服务集成，例如 CloudWatch、Amazon VPC、CloudTrail 和 Amazon SNS。
- 完全托管的软件修补和升级。
- AWS 身份和访问管理（IAM）集成和针对管理 API 的基于标签的访问控制。

MemoryDB 核心组件

下面，您可以找到 MemoryDB 部署的主要组件概述。

主题

- 集群
- Nodes
- 分片
- 参数组
- 子网组
- 访问控制列表
- Users

集群

集群是服务单个数据集的一个或多个节点的集合。MemoryDB 数据集分为多个分片，每个分片有一个主节点和最多 5 个可选副本节点。主节点处理读取和写入请求，而副本节点仅处理读取请求。主节点可失效转移至副本节点，副本节点随之提升为该分片的新主节点。MemoryDB 将 Valkey 或 Redis

OSS 作为数据库引擎运行，并在创建集群时为集群指定引擎版本。您可以使用 AWS CLI、MemoryDB API 或创建和修改集群。 AWS 管理控制台

每个 MemoryDB 集群都会运行一个 Valkey 或 Redis OSS 引擎版本。每个引擎版本都有其自身支持的特征。此外，每个引擎版本在参数组中均有一组参数，用于控制其管理的集群的行为。

集群的计算和内存容量由节点类型决定。您可以选择最能满足您需求的节点类型。如果一段时间后您的需求出现了变化，可以更改节点类型。有关信息，请参阅 受支持的节点类型。

> **Note**
>
> 有关 MemoryDB 节点类型的定价信息，请参阅 MemoryDB 定价。

您可以使用 Amazon Virtual Private Cloud（Amazon VPC）服务，在虚拟私有云（VPC）上运行集群。使用 VPC 时，您的虚拟联网环境完全由您控制。您可以选择自己的 IP 地址范围、创建子网以及配置路由和访问控制列表。MemoryDB 可以管理快照、软件修补、自动故障检测和恢复。在 VPC 中运行集群不会产生额外费用。有关将 Amazon VPC 与 MemoryDB 结合使用的更多信息，请参阅 MemoryDB 和 Amazon VPC。

许多 MemoryDB 操作面向集群：

- 创建集群
- 修改集群
- 拍摄集群快照
- 删除集群
- 查看集群中的元素
- 在集群中添加和删除成本分配标签

有关更多详细信息，请参阅以下相关主题：

- 管理集群 和 管理节点

  有关集群、节点和相关操作的信息。
- MemoryDB 中的故障恢复能力

  有关增强集群的容错能力的信息。

Nodes

节点是 MemoryDB 部署中的最小构建基块，使用 Amazon EC2 实例运行。每个节点都运行在您创建集群时选择的引擎版本。节点属于集群所含的分片。

每个节点都运行在您创建集群时选择版本的引擎实例。如果需要，您可以将集群中的节点纵向扩展或缩减到不同类型。有关更多信息，请参阅 扩展。

一个集群中的所有节点都具有相同的节点类型。支持多种节点类型，每种可有不同的内存量。有关受支持的节点类型的列表，请参阅 受支持的节点类型。

有关节点的更多信息，请参阅管理节点。

分片

分片是 1 到 6 个节点组成的分组，一个主写入节点和 5 个只读副本节点。一个 MemoryDB 集群始终有至少一个分片。

MemoryDB 集群最多可以拥有 500 个分区，并且跨分区对您的数据进行分区。例如，您可以选择配置一个 500 节点的集群，范围介于 83 个分片（一个主分片和 5 个副本分片）和 500 个分片（一个主分片，无副本分片）之间。确保可提供足够的 IP 地址来满足增长需求。常见的陷阱包括子网组中的子网 CIDR 范围太小，或者子网被其他集群共享和大量使用。

多节点分区通过指定一个读/写主节点和 1 到 5 个副本节点来实现复制。有关更多信息，请参阅 了解 MemoryDB 复制。

有关分片的更多信息，请参阅使用分片。

参数组

参数组是管理集群上的引擎运行时设置的简单方法。参数用于控制内存使用率、项目大小等。MemoryDB 参数组是可应用于集群的特定于引擎的参数的命名集合，该集群中的所有节点都以完全相同的方式进行配置。

有关 MemoryDB 参数组的更多详细信息，请参见 使用参数组配置引擎参数。

子网组

子网组是您可为在 Amazon Virtual Private Cloud（VPC）环境中运行的集群指定的子网（通常为私有子网）集合。

在 Amazon VPC 中创建集群时，请指定一个子网组或使用提供的默认子网组。MemoryDB 使用该子网组选择与节点关联的子网和子网中的 IP 地址。

有关 MemoryDB 子网组的更多详细信息，请参阅 子网和子网组。

访问控制列表

访问控制列表是一个或多个用户的集合。访问字符串根据 ACL 规则授权用户访问 Valkey 或 Redis OSS 命令和数据。

有关 MemoryDB 访问控制列表的更多详细信息，请参阅 使用访问控制列表对用户进行身份验证 () ACLs。

Users

用户都有用户名和密码，可用于在 MemoryDB 集群上访问数据和发出命令。用户是访问控制列表（ACL）的成员，ACL 可用于确定用户在 MemoryDB 集群上的权限。有关更多信息，请参阅 使用访问控制列表对用户进行身份验证 () ACLs。

相关服务

ElastiCache

在决定是使用 MemoryDB 还是使用 ElastiCache 时，请考虑以下比较结果：

- MemoryDB 是一个耐用的内存数据库，适用于需要超快速主数据库的工作负载。如果您的工作负载需要可提供超快性能（微秒级读取和个位数毫秒级写入延迟）的耐用数据库，则应考虑使用 MemoryDB。如果您想构建使用 Valkey 或 Redis OSS 数据结构和 API 访问耐用主数据库的应用程序，MemoryDB 也可能非常适合您的使用场景。最后，您应该考虑使用 MemoryDB 来简化应用程序架构并降低成本，从使用数据库替换为使用缓存以提高耐用性和性能。
- ElastiCache 是一项服务，通常用于缓存来自其他使用 Valkey 和 Redis OSS 的数据库和数据存储的数据。您应该考虑将 ElastiCache 用于缓存需要加速与现有主数据库或数据存储之间数据访问的工作负载（微秒读写性能）。对于想要使用 Valkey 或 Redis OSS 数据结构和 API 访问存储在主数据库或数据存储中的数据的使用场景，您也应该考虑使用 ElastiCache。

选择区域和可用区

AWS 云计算资源存储在具有高度可用性的数据中心设施中。为了提供额外的扩展性和可靠性，这些数据中心设施位于不同的物理位置。这些位置按照区域和可用区进行分类。

AWS 区域是指大型、分布范围广泛的单独地理位置。可用区是 AWS 区域中的不同位置，旨在隔离其他可用区中的故障。它们提供与同一 AWS 区域中不同可用区之间的低成本、低延迟网络连接。

> **Important**
>
> 每一个区域都是完全独立的。您启动的任何 MemoryDB 活动（例如，创建集群）仅可在您当前的默认区域中运行。

若要在特定地区创建或使用集群，请使用相应的区域服务端点。有关服务端点，请参阅MemoryDB 多区域。

使用 MemoryDB 多区域，您可以提高可用性和弹性，同时受益于多区域应用程序的低延迟本地读取和写入。有关使用 MemoryDB 多区域的信息，请参阅 支持的区域和端点。

找到您的节点

任何至少有一个副本的集群都必须分布在多个可用区中。在单个可用区内找到所有内容的唯一方法是使用单节点分片组成的集群。

通过在不同的可用区内放置节点，MemoryDB 可排除某个可用区内的故障（如停电）导致可用性丢失的可能性。

- 创建 MemoryDB 集群
- 修改 MemoryDB 集群

支持的区域和端点

MemoryDB 在多个 AWS 区域中提供。这意味着，您可在满足您要求的位置启动 MemoryDB 集群。例如，您可以在最靠近您客户的 AWS 区域或者满足某些法律要求的特定 AWS 区域中启动。此外，随着 MemoryDB 将可用性扩展到新的 AWS 区域，MemoryDB 支持将当时最新的两个 `MAJOR.MINOR` 版本用于此新区域。有关 MemoryDB 版本的更多信息，请参阅 引擎版本。

默认情况下，AWS 开发工具包、AWS CLI、MemoryDB API 和 MemoryDB 控制台参考美国东部（弗吉尼亚州北部）区域。随着 MemoryDB 不断向新区域扩展，这些区域的新端点同样可以在您的 HTTP 请求、AWS 开发工具包、AWS CLI 和控制台中使用。

从设计而言，每个区域都与其他区域完全隔离。每个区域中有多个可用区（AZ）。通过在不同的可用区内启动节点，您可以实现可能的最大容错。有关区域和可用区域的更多信息，请参阅本主题开端的选择区域和可用区。

支持 MemoryDB 的区域

区域名称/区域	终端节点	协议	
美国东部（俄亥俄州）区域 `us-east-2`	`memory-db.us-east-2.amazonaws.com`	HTTPS	
美国东部（弗吉尼亚州北部）区域 `us-east-1`	`memory-db.us-east-1.amazonaws.com`	HTTPS	
美国西部（北加利福尼亚）区域 `us-west-1`	`memory-db.us-west-1.amazonaws.com`	HTTPS	
美国西部（俄勒冈州）区域 `us-west-2`	`memory-db.us-west-2.amazonaws.com`	HTTPS	

区域名称/区域	终端节点	协议	
加拿大（中部）区域 ca-central-1	memory-db.ca-central-1.amazonaws.com	HTTPS	
亚太地区（香港）区域 ap-east-1	memory-db.ap-eastl-1.amazonaws.com	HTTPS	
亚太地区（孟买）区域 ap-south-1	memory-db.ap-south-1.amazonaws.com	HTTPS	
亚太地区（东京）区域 ap-northeast-1	memory-db.ap-northeast-1.amazonaws.com	HTTPS	
亚太地区（首尔）区域 ap-northeast-2	memory-db.ap-northeast-2.amazonaws.com	HTTPS	
亚太地区（新加坡）区域 ap-southeast-1	memory-db.ap-southeast-1.amazonaws.com	HTTPS	
亚太地区（悉尼）区域 ap-southeast-2	memory-db.ap-southeast-2.amazonaws.com	HTTPS	
欧洲地区（法兰克福）区域 eu-central-1	memory-db.eu-central-1.amazonaws.com	HTTPS	

区域名称/区域	终端节点	协议	
欧洲地区（爱尔兰）区域 eu-west-1	memory-db.eu-west-1.amazonaws.com	HTTPS	
欧洲地区（伦敦）区域 eu-west-2	memory-db.eu-west-2.amazonaws.com	HTTPS	
欧洲地区（巴黎）区域 eu-west-3	memory-db.eu-west-3.amazonaws.com	HTTPS	
欧洲地区（斯德哥尔摩）区域 eu-north-1	memory-db.eu-north-1.amazonaws.com	HTTPS	
欧洲地区（米兰）区域 eu-south-1	memory-db.eu-south-1.amazonaws.com	HTTPS	
欧洲地区（西班牙）区域 eu-south-2	memory-db.eu-south-2.amazonaws.com	HTTPS	
南美洲（圣保罗）区域 sa-east-1	memory-db.sa-east-1.amazonaws.com	HTTPS	
中国（北京）区域 cn-north-1	memory-db.cn-north-1.amazonaws.com.cn	HTTPS	

区域名称/区域	终端节点	协议	
中国（宁夏）区域 cn-northwest-1	memory-db.cn-northwest-1.amazonaws.com.cn	HTTPS	

有关按区域划分的 AWS 产品和服务表，请参阅按区域划分的产品和服务。

有关区域内支持的可用区表，请参阅 子网和子网组。

访问 MemoryDB

每个 MemoryDB 集群端点都包含一个地址和一个端口。此集群端点支持 Valkey 和 Redis OSS 集群协议，使得客户端可以发现集群中每个节点的特定角色、IP 地址和插槽。当主节点出现故障并且副本取而代之时，您可以使用 Valkey 或 Redis OSS 集群协议连接到集群端点以发现新的主节点。

您需要使用 cluster nodes 或 cluster slots 命令连接到集群端点才能发现节点端点。发现正确的密钥节点后，您可以直接连接到该节点以处理读取/写入请求。Valkey 或 Redis OSS 客户端可以使用集群端点自动连接到正确的节点。

要对集群中的特定节点进行问题排除，您还可以使用特定于节点的端点，但这些并非正常使用所必需。

要找到集群端点，请参阅：

- 查找 MemoryDB 集群的端点（AWS CLI）
- 查找 MemoryDB 集群的端点（MemoryDB API）

有关连接到节点或集群的详细信息，请参阅 使用 redis-cli 连接到 MemoryDB 节点。

MemoryDB 安全性

MemoryDB 的安全性在三个级别上进行管理：

- 要控制可对 MemoryDB 集群和节点执行托管操作的人员，请使用AWS Identity and Access Management（IAM）。使用 IAM 策略连接到 AWS 时，您的 AWS 账户必须具有授予执行操作所需的权限的 IAM 策略。有关更多信息，请参阅 MemoryDB 中的身份和访问管理。

- 要控制对集群的访问权限级别，您需要创建具有指定权限的用户并将其分配到访问控制列表（ACL）。ACL 随即相应地与一个或多个集群关联。有关更多信息，请参阅 使用访问控制列表对用户进行身份验证 () ACLs。
- 必须基于 Amazon VPC 服务在虚拟私有云（VPC）中创建 MemoryDB 集群。要控制哪些设备和 Amazon EC2 实例能够建立与 VPC 中 MemoryDB 集群的节点的端点和端口的连接，请使用 VPC 安全组。您可以使用传输层安全性（TLS）/安全套接字层（SSL）建立这些终端节点和端口连接。此外，公司的防火墙规则也可以控制公司中运行的哪些设备可以建立与 MemoryDB 集群的连接。有关 VPC 的更多信息，请参阅 MemoryDB 和 Amazon VPC。

有关配置安全性的信息，请参阅MemoryDB 中的安全性。

开始使用 MemoryDB

此练习将指导您完成使用 MemoryDB 管理控制台创建、授予访问权限、连接并最终删除 MemoryDB 集群的步骤。

> **Note**
>
> 在此次练习中，我们建议您在创建集群时使用轻松创建选项，并在进一步探索 MemoryDB 的功能后再尝试其他两个选项。

主题

- 步骤 1：设置
- 步骤 2：创建集群
- 步骤 3：授予对集群的访问权限
- 步骤 4：连接到集群
- 步骤 5：删除集群
- 后续步骤

步骤 1：设置

下面，您可以找到描述开始使用 MemoryDB 时必须采取的一次性操作的主题。

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

报名参加 AWS 账户

1. 打开https://portal.aws.amazon.com/billing/注册。
2. 按照屏幕上的说明操作。

   在注册时，将接到电话或收到短信，要求使用电话键盘输入一个验证码。

   当您注册时 AWS 账户，就会创建AWS 账户根用户一个。根用户有权访问该账户中的所有 AWS 服务 和资源。作为最佳安全实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

AWS 注册过程完成后会向您发送一封确认电子邮件。您可以随时前往 https://aws.amazon.com/并选择“我的账户”，查看您当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

注册后，请保护您的安全 AWS 账户 AWS 账户根用户 AWS IAM Identity Center，启用并创建管理用户，这样您就不会使用 root 用户执行日常任务。

保护你的 AWS 账户根用户

1. 选择 Root 用户并输入您的 AWS 账户 电子邮件地址，以账户所有者的身份登录。AWS 管理控制台在下一页上，输入您的密码。

   要获取使用根用户登录方面的帮助，请参阅《AWS 登录 用户指南》中的 Signing in as the root user。
2. 为您的根用户启用多重身份验证（MFA）。

   有关说明，请参阅 I A M 用户指南中的为 AWS 账户 根用户启用虚拟 MFA 设备（控制台）。

创建具有管理访问权限的用户

1. 启用 IAM Identity Center。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的启用 AWS IAM Identity Center。
2. 在 IAM Identity Center 中，为用户授予管理访问权限。

   有关使用 IAM Identity Center 目录 作为身份源的教程，请参阅《用户指南》 IAM Identity Center 目录中的使用默认设置配置AWS IAM Identity Center 用户访问权限。

以具有管理访问权限的用户身份登录

- 要使用您的 IAM Identity Center 用户身份登录，请使用您在创建 IAM Identity Center 用户时发送到您的电子邮件地址的登录 URL。

  有关使用 IAM Identity Center 用户登录的帮助，请参阅AWS 登录 用户指南中的登录 AWS 访问门户。

将访问权限分配给其他用户

1. 在 IAM Identity Center 中，创建一个权限集，该权限集遵循应用最低权限的最佳做法。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的 Create a permission set。

2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的 Add groups。

授权以编程方式访问

如果用户想在 AWS 外部进行交互，则需要编程访问权限 AWS 管理控制台。授予编程访问权限的方式取决于正在访问的用户类型 AWS。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
IAM	（推荐）使用控制台凭证作为临时凭证，签署对 AWS CLI AWS SDKs、或的编程请求 AWS APIs。	按照您希望使用的界面的说明进行操作。 • 有关的 AWS CLI，请参阅《AWS Command Line Interface 用户指南》中的“登录 AWS 本地开发”。 • 有关信息 AWS SDKs，请参阅《工具参考指南》AWS SDKs 和《工具参考指南》中的 “登录进行 AWS 本地开发”。
人力身份 （在 IAM Identity Center 中管理的用户）	使用临时证书签署向 AWS CLI AWS SDKs、或发出的编程请求 AWS APIs。	按照您希望使用的界面的说明进行操作。 • 有关的 AWS CLI，请参阅《AWS Command Line Interface 用户指南》 AWS

哪个用户需要编程式访问权限？	目的	方式
		IAM Identity Center中的 “配置 AWS CLI 要使用”。 • 有关工具和 AWS SDKs AWS APIs，请参阅《工具参考指南》中的 IAM 身份中心身份验证AWSSDKs 和工具参考指南。
IAM	使用临时证书签署向 AWS CLI AWS SDKs、或发出的编程请求 AWS APIs。	按照 IAM 用户指南中的将临时证书与 AWS 资源配合使用中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署向 AWS CLI AWS SDKs、或发出的编程请求 AWS APIs。	按照您希望使用的界面的说明进行操作。 • 有关信息 AWS CLI，请参阅用户指南中的使用 IAM 用户证书进行身份验证。AWS Command Line Interface • 有关 AWS SDKs 和工具，请参阅《工具参考指南》AWS SDKs 和《工具参考指南》中的使用长期凭证进行身份验证。 • 有关信息 AWS APIs，请参阅 IAM 用户指南中的管理 IAM 用户的访问密钥。

相关主题:

- IAM 用户指南中的什么是 IAM？
- AWS AWS 一般参考中的@@ 安全证书。

设置权限（仅限新的 MemoryDB 用户）

要提供访问权限，请为您的用户、组或角色添加权限：

- 中的用户和群组 AWS IAM Identity Center：

  创建权限集合。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。
- 通过身份提供商在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中针对第三方身份提供商创建角色（联合身份验证）的说明进行操作。
- IAM 用户：
  - 创建您的用户可以担任的角色。按照《IAM 用户指南》中为 IAM 用户创建角色的说明进行操作。
  - （不推荐使用）将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中向用户添加权限（控制台）中的说明进行操作。

MemoryDB 会代表您创建并使用服务相关角色以预配置资源并访问其他 AWS 资源和服务。要让 MemoryDB 为您创建服务相关角色，请使用名为的托 AWS管策略。`AmazonMemoryDBFullAccess`此角色预配置了该服务您代表您创建服务相关角色所需的权限。

您可能决定不使用默认策略，而是使用自定义托管策略。在这种情况下，请确保您具有调用 `iam:createServiceLinkedRole` 的权限或创建了 MemoryDB 服务相关角色。

有关更多信息，请参阅下列内容：

- 创建新策略（IAM）
- 适用于 MemoryDB 的AWS托管（预定义）策略
- 使用 MemoryDB 的服务相关角色

下载和配置 AWS CLI

可在以下 AWS CLI 网址获得：http://aws.amazon.com/cli。它在 Windows、MacOS 和 Linux 上运行。下载后 AWS CLI，请按照以下步骤对其进行安装和配置：

1. 转到 AWS Command Line Interface 用户指南。
2. 按照安装 AWS CLI 和配置 CL AWS I 的说明进行操作。

步骤 2：创建集群

在创建用于生产使用的集群之前，您显然需要考虑如何配置集群以满足您的业务需求。这些问题在 准备集群 部分中解决。就本入门练习而言，您可以在其适用时接受默认配置值。

您所创建的集群将是活动的，不会在沙盒中运行。您需要为实例支付标准的 MemoryDB 使用费，直到您删除该实例。如果您一鼓作气完成此处描述的练习并在使用完毕后删除集群，则产生的全部费用将非常少（通常不到一美元）。有关 MemoryDB 使用费率的更多信息，请参阅 MemoryDB。

在虚拟私有云（VPC）中基于 Amazon VPC 服务启动集群。

创建 MemoryDB 集群

以下示例展示了如何使用 AWS 管理控制台、 AWS CLI 和 MemoryDB API 创建集群。

创建集群（控制台）

使用 MemoryDB 控制台创建集群

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在左侧导航窗格中，选择集群，然后选择创建。

Easy create

1. 填写 Configuration (配置) 部分。这将配置集群的节点类型和默认配置。从以下选项中选择所需的适当内存大小和网络性能：
   - 生产
   - 开发/测试
   - 演示
2. 完成集群信息部分。
   a. 在名称中，输入集群的名称。

      集群命名约束如下：
      - 必须包含 1 – 40 个字母数字字符或连字符。
      - 必须以字母开头。

- 不能包含两个连续连字符。
- 不能以连字符结束。

b. 在描述 框中，输入此集群的描述。

3. 完成子网组部分：

- 对于子网组，创建新的子网组，或从可用列表中选择要应用于此集群的现有子网组。如果要创建一个新的：
  - 输入名称
  - 输入描述
  - 如果启用了多可用区，则子网组必须至少包含两个位于不同可用区中的子网。有关更多信息，请参阅 子网和子网组。
  - 如果要创建新的子网组但不具有现有 VPC，则系统会要求您创建 VPC。有关更多信息，请参阅《Amazon VPC 用户指南》中的什么是 Amazon VPC？。

4. 对于向量搜索，您可以启用向量搜索功能来存储向量嵌入和执行向量搜索。请注意，这将修复引擎版本兼容性、参数组和分片的值。有关更多信息，请参阅 向量搜索。

5. 查看默认设置：

   使用轻松创建时，其余集群设置均为默认设置。请注意，其中一些设置可以在创建后更改，如创建后可编辑所示。

6. 对于标签，您可以选择应用标签来搜索和筛选集群或跟踪 AWS 成本。

7. 查看您的所有输入和选择，然后进行任意所需的更正。准备就绪后，请选择创建启动集群或选择取消取消操作。

当您的集群状态为 available 时，您可向其授予 EC2 访问权限，连接到集群并开始使用它。有关更多信息，请参阅 步骤 3：授予对集群的访问权限。

> **Important**
>
> 一旦您的集群变为可用状态，您便需要为集群处于活动状态的每个小时或分钟支付费用（即使您并未主动使用集群）。要停止此集群产生的费用，您必须将其删除。请参阅步骤 5：删除集群。

Create new cluster

1. 完成集群信息部分。

   a. 在名称中，输入集群的名称。

      集群命名约束如下：

      - 必须包含 1 – 40 个字母数字字符或连字符。
      - 必须以字母开头。
      - 不能包含两个连续连字符。
      - 不能以连字符结束。

   b. 在描述 框中，输入此集群的描述。

2. 完成子网组部分：

   - 对于子网组，创建新的子网组，或从可用列表中选择要应用于此集群的现有子网组。如果要创建一个新的：

     - 输入名称
     - 输入描述
     - 如果启用了多可用区，则子网组必须至少包含两个位于不同可用区中的子网。有关更多信息，请参阅 子网和子网组。
     - 如果要创建新的子网组但不具有现有 VPC，则系统会要求您创建 VPC。有关更多信息，请参阅《Amazon VPC 用户指南》中的什么是 Amazon VPC？。

3. 完成集群设置部分：

   a. 对于启用向量搜索功能，您可以启用此功能来存储向量嵌入和执行向量搜索。请注意，这将修复引擎版本兼容性、参数组和分片的值。有关更多信息，请参阅 向量搜索。

   b. 对于引擎版本兼容性，请接受默认值。例如，在 Valkey 中，默认值为 7.2.6，在 Redis OSS 中，默认值为 6.2。

   c. 对于端口，请接受默认端口 6379，或者，如果您出于某个原因需要使用其他端口，请输入相应的端口号。

   d. 对于参数组，如果您启用了向量搜索，请使用 `default.memorydb-valkey7.search`。否则，对于 Valkey，请接受 `default.memorydb-valkey7` 参数组。

      参数组控制集群的运行时参数。有关参数组的更多信息，请参阅 引擎特定参数。

e. 对于节点类型，请为所需节点类型（及其关联的内存大小）选择一个值。

如果您选择 r6gd 系列的节点类型，系统会自动启用数据分层，从而在内存和 SSD 之间拆分数据存储。有关更多信息，请参阅 数据分层。

f. 对于分片数，选择要用于此集群的分片数。为实现更高的集群可用性，我们建议您至少添加 2 个分片。

您可以动态更改集群中的分片数量。有关更多信息，请参阅 扩展 MemoryDB 集群。

g. 对于每个分片的副本数量，请选择每个分片中需要的只读副本节点数。

存在以下限制：

- 如果启用了多可用区，请确保每个分片至少有一个副本。
- 使用控制台创建集群时，每个分片的副本数相同。

h. 选择下一步。

i. 完成高级设置部分：

i. 对于安全组，选择要用于该集群的安全组。安全组 充当防火墙来控制对集群的网络访问。您可以使用 VPC 的默认安全组或创建一个新的安全组。

有关安全组的更多信息，请参阅 Amazon VPC 用户指南中的您的 VPC 的安全组。

ii. 要加密您的数据，您可以选择以下选项：

- Encryption at rest（静态加密）– 对磁盘上存储的数据启用加密。有关更多信息，请参阅静态加密。

> **Note**
>
> 您可以选择提供默认加密密钥以外的加密密钥，方法是选择客户管理的 KM AWS S 密钥并选择密钥。

- Encryption in-transit（传输中加密）– 对传输中数据启用加密。如果您选择不加密，则系统将使用默认用户创建一个名为“开放访问”的开放访问控制列表。有关更多信息，请参阅 使用访问控制列表对用户进行身份验证 () ACLs。

iii. 对于快照，请选择性地指定快照保留期和快照时段。默认情况下，启用自动快照处于预先选中状态。

iv. 对于维护时段，请选择性地指定维护时段。维护时段是每周中 MemoryDB 为您的集群计划系统维护的时间，通常以小时为时间长度。您可以允许 MemoryDB 选择维护时段的日期和时间（无首选项），或者自行选择日期、时间和持续时间（指定维护时段）。如果您从列表中选择指定维护时段，请选择维护时段的起始日、起始时间和持续时间（以小时为单位）。所有时间均为 UCT 时间。

有关更多信息，请参阅 管理维护。

v. 对于 通知，选择现有 Amazon Simple Notification Service (Amazon SNS) 主题，或选择手动 ARN 输入，然后输入主题的 Amazon 资源名称（ARN）。Amazon SNS 允许您向联网的智能设备推送通知。默认设置为禁用通知。有关更多信息，请参阅 https://aws.amazon.com/sns/。

vi. 对于标签，您可以选择应用标签来搜索和筛选集群或跟踪 AWS 成本。

j. 查看您的所有输入和选择，然后进行任意所需的更正。准备就绪后，请选择创建启动集群或选择取消取消操作。

当您的集群状态为 available 时，您可向其授予 EC2 访问权限，连接到集群并开始使用它。有关更多信息，请参阅 步骤 3：授予对集群的访问权限。

> **Important**
>
> 一旦您的集群变为可用状态，您便需要为集群处于活动状态的每个小时或分钟支付费用（即使您并未主动使用集群）。要停止此集群产生的费用，您必须将其删除。请参阅步骤 5：删除集群。

Restore from snapshots

在快照源下，选择要从中迁移数据的源快照。有关更多信息，请参阅 快照和还原。

> **Note**
>
> 如果您希望新集群启用向量搜索，则源快照还必须启用向量搜索。

目标集群默认使用源集群的设置。（可选）您可以更改目标集群上的以下设置：

1. 集群信息

a. 在名称中，输入集群的名称。

集群命名约束如下：

- 必须包含 1 – 40 个字母数字字符或连字符。
- 必须以字母开头。
- 不能包含两个连续连字符。
- 不能以连字符结束。

b. 在描述 框中，输入此集群的描述。

2. 子网组

- 对于子网组，创建新的子网组，或从可用列表中选择要应用于此集群的现有子网组。如果要创建一个新的：
  - 输入名称
  - 输入描述
  - 如果启用了多可用区，则子网组必须至少包含两个位于不同可用区中的子网。有关更多信息，请参阅 子网和子网组。
  - 如果要创建新的子网组但不具有现有 VPC，则系统会要求您创建 VPC。有关更多信息，请参阅《Amazon VPC 用户指南》中的什么是 Amazon VPC？。

3. 集群设置

a. 对于启用向量搜索功能，您可以启用此功能来存储向量嵌入和执行向量搜索。请注意，这将修复引擎版本兼容性、参数组和分片的值。有关更多信息，请参阅 向量搜索。

b. 对于引擎版本兼容性，请接受默认值 6.2。

c. 对于端口，请接受默认端口 6379，或者，如果您出于某个原因需要使用其他端口，请输入相应的端口号。

d. 对于参数组，如果您启用了向量搜索，请使用 `default.memorydb-redis7.search.preview`。否则，请接受 `default.memorydb-redis7` 参数组。

参数组控制集群的运行时参数。有关参数组的更多信息，请参阅 引擎特定参数。

e. 对于节点类型，请为所需节点类型（及其关联的内存大小）选择一个值。

如果您选择 r6gd 系列的节点类型，系统会自动启用数据分层，从而在内存和 SSD 之间拆分数据存储。有关更多信息，请参阅 数据分层。

f. 对于分片数，选择要用于此集群的分片数。为实现更高的集群可用性，我们建议您至少添加 2 个分片。

您可以动态更改集群中的分片数量。有关更多信息，请参阅 扩展 MemoryDB 集群。

g. 对于每个分片的副本数量，请选择每个分片中需要的只读副本节点数。

存在以下限制：

- 如果启用了多可用区，请确保每个分片至少有一个副本。
- 使用控制台创建集群时，每个分片的副本数相同。

h. 选择下一步。

i. 高级设置

i. 对于安全组，选择要用于该集群的安全组。安全组 充当防火墙来控制对集群的网络访问。您可以使用 VPC 的默认安全组或创建一个新的安全组。

有关安全组的更多信息，请参阅 Amazon VPC 用户指南中的您的 VPC 的安全组。

ii. 要加密您的数据，您可以选择以下选项：

- Encryption at rest（静态加密）– 对磁盘上存储的数据启用加密。有关更多信息，请参阅静态加密。

> **Note**
>
> 您可以选择提供默认加密密钥以外的加密密钥，方法是选择客户管理的 KM AWS S 密钥并选择密钥。

- Encryption in-transit（传输中加密）– 对传输中数据启用加密。如果您选择不加密，则系统将使用默认用户创建一个名为“开放访问”的开放访问控制列表。有关更多信息，请参阅 使用访问控制列表对用户进行身份验证 () ACLs。

iii. 对于快照，请选择性地指定快照保留期和快照时段。默认情况下，启用自动快照处于预先选中状态。

iv. 对于维护时段，请选择性地指定维护时段。维护时段是每周中 MemoryDB 为您的集群计划系统维护的时间，通常以小时为时间长度。您可以允许 MemoryDB 选择维护时段的日期和时间（无首选项），或者自行选择日期、时间和持续时间（指定维护时段）。如果您从列表中选择指定维护时段，请选择维护时段的起始日、起始时间和持续时间（以小时为单位）。所有时间均为 UCT 时间。

有关更多信息，请参阅 管理维护。

v. 对于 通知，选择现有 Amazon Simple Notification Service (Amazon SNS) 主题，或选择手动 ARN 输入，然后输入主题的 Amazon 资源名称（ARN）。Amazon SNS 允许您向联网的智能设备推送通知。默认设置为禁用通知。有关更多信息，请参阅 https://aws.amazon.com/sns/。

vi. 对于标签，您可以选择应用标签来搜索和筛选集群或跟踪 AWS 成本。

j. 查看您的所有输入和选择，然后进行任意所需的更正。准备就绪后，请选择创建启动集群或选择取消取消操作。

当您的集群状态为 available 时，您可向其授予 EC2 访问权限，连接到集群并开始使用它。有关更多信息，请参阅 步骤 3：授予对集群的访问权限。

> **Important**
>
> 一旦您的集群变为可用状态，您便需要为集群处于活动状态的每个小时或分钟支付费用（即使您并未主动使用集群）。要停止此集群产生的费用，您必须将其删除。请参阅步骤 5：删除集群。

创建集群 (AWS CLI)

要使用创建集群 AWS CLI，请参阅create-cluster。以下是示例：

对于 Linux、macOS 或 Unix：

```
aws memorydb create-cluster \
    --cluster-name my-cluster \
    --node-type db.r6g.large \
    --acl-name my-acl \
    --engine valkey \
    --subnet-group my-sg
```

对于 Windows：

```
aws memorydb create-cluster ^
   --cluster-name my-cluster ^
   --node-type db.r6g.large ^
   --acl-name my-acl ^
   --engine valkey
   --subnet-group my-sg
```

您应得到以下 JSON 响应：

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "creating",
        "NumberOfShards": 1,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "7.2",
        "EnginePatchVersion": "7.2.6",
        "ParameterGroupName": "default.memorydb-valkey7",
        "Engine": "valkey"
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxxxx:cluster/my-cluster",
```

```
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "ACLName": "my-acl",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

一旦集群的状态更改为 `available`，您即可开始使用该集群。

> **Important**
>
> 一旦您的集群变为可用状态，您便需要为集群处于活动状态的每个小时或分钟支付费用（即使您并未主动使用集群）。要停止此集群产生的费用，您必须将其删除。请参阅步骤 5：删除集群。

创建集群（MemoryDB API）

要使用 MemoryDB API 创建集群，请使用操作。CreateCluster

> **Important**
>
> 一旦您的集群变为可用状态，您便需要为集群处于活动状态的每个小时或分钟支付费用（即使您并未使用集群）。要停止此集群产生的费用，您必须将其删除。请参阅步骤 5：删除集群。

设置身份验证

有关为集群设置身份验证的信息，请参阅 使用 IAM 进行身份验证 和 使用访问控制列表对用户进行身份验证 () ACLs。

步骤 3：授予对集群的访问权限

此部分假设您熟悉 Amazon EC2 实例的启动和连接。有关更多信息，请参阅 Amazon EC2 入门指南。

MemoryDB 集群旨在通过 Amazon EC2 实例进行访问。它们也可以通过在 Amazon 弹性容器服务或 AWS Lambda中运行的容器化的应用程序或无服务器应用程序进行访问。最常见的情况是从同一 Amazon Virtual Private Cloud（Amazon VPC）中的 Amazon EC2 实例访问 MemoryDB 集群，这将是本练习的情况。

必须先授权 EC2 实例访问集群，然后您才能从 EC2 实例连接到集群。

最常见的使用场景是，当 EC2 实例上部署的应用程序需要连接到同一 VPC 中的集群时。要管理同一 VPC 中 EC2 实例与集群之间的访问，最简单方法如下所示：

1. 为集群创建 VPC 安全组。此安全组可用于限制对集群的访问权限。例如，可为此安全组创建自定义规则，允许使用您创建集群时分配给该集群的端口以及将用来访问集群的 IP 地址进行 TCP 访问。

   MemoryDB 集群的默认端口为 6379。
2. 为 EC2 实例（Web 和应用程序服务器）创建 VPC 安全组。如果需要，此安全组可允许通过 VPC 的路由表从 Internet 访问 EC2 实例。例如，您可设置此安全组的规则以允许通过端口 22 对 EC2 实例进行 TCP 访问。
3. 在集群的安全组中创建自定义规则，允许从为 EC2 实例创建的安全组连接。这将允许安全组的任何成员均可访问集群。

在 VPC 安全组中创建允许从另一安全组连接的规则

1. 登录 AWS 管理控制台并在 https://console.aws.amazon.com/vpc 上打开 Amazon VPC 控制台。
2. 在左侧导航窗格中，选择安全组。
3. 选择或创建一个要用于集群的安全组。在入站规则下，选择编辑入站规则，然后选择添加规则。此安全组将允许访问其他安全组的成员。
4. 从 Type 中选择 Custom TCP Rule。

   a. 对于 Port Range，指定在创建集群时使用的端口。

      MemoryDB 集群的默认端口为 6379。

   b. 在 Source 框中，开始键入安全组的 ID。从列表中选择要用于 Amazon EC2 实例的安全组。
5. 完成后选择 Save。

启用访问后，您现在就可以连接到集群，如下一部分中所述。

有关从不同的 Amazon VPC、不同的 AWS 区域甚至您的公司网络访问您的 MemoryDB 集群的信息，请参阅以下内容：

- 用于访问 Amazon VPC 中 MemoryDB 集群的访问模式
- 从外部 AWS 访问 MemoryDB 资源

步骤 4：连接到集群

在继续之前，请完成步骤 3：授予对集群的访问权限。

此部分假设您已创建了 Amazon EC2 实例并可以连接到该实例。有关如何执行此操作的说明，请参阅 Amazon EC2 入门指南。

仅当您进行授权后，Amazon EC2 实例才能连接到集群。

查找您的集群端点

在您的集群处于可用状态且您已授予对该集群的访问权限时，您可以登录 Amazon EC2 实例并连接到该集群。为此，您必须先确定端点。

为进一步了解如何查找您的端点，请参阅以下内容：

- 查找 MemoryDB 集群（AWS 管理控制台）的端点
- 查找 MemoryDB 集群的端点（AWS CLI）
- 查找 MemoryDB 集群的端点（MemoryDB API）

连接到 MemoryDB 集群（Linux）

现在您有了所需的端点，便可以登录 EC2 实例并连接到集群。在以下示例中，您通过 Ubuntu 22 使用 cli 实用工具连接到集群。最新版本的 cli 还支持 SSL/TLS 连接 encryption/authentication 已启用的集群。

使用 redis-cli 连接到 MemoryDB 节点

要从 MemoryDB 节点中访问数据，您可以使用与安全套接字层（SSL）一起工作的客户端。你也可以在亚马逊 Linux 和 TLS/SSL 亚马逊 Linux 2 上使用 redis-cli。

使用 redis-cli 连接到 Amazon Linux 2 或 Amazon Linux 上的 MemoryDB 集群

1. 下载并编译 redis-cli 实用工具。此实用工具包含在 Redis OSS 软件发布版中。
2. 在 EC2 实例的命令提示符处，键入适合您所用 Linux 版本的相应命令。

   Amazon Linux 2023

如果使用的是 Amazon Linux 2023，请输入以下命令：

```
sudo yum install redis6 -y
```

然后键入以下命令，并使用您集群的端点和端口来替换此示例中显示的相应内容。

```
redis-cli -h Primary or Configuration Endpoint --tls -p 6379
```

有关查找端点的更多信息，请参阅查找您的节点端点。

Amazon Linux 2

如果使用的是 Amazon Linux 2，请输入以下命令：

```
sudo yum -y install openssl-devel gcc
wget https://download.redis.io/releases/redis-7.2.5.tar.gz
tar xvzf redis-7.2.5.tar.gz
cd redis-7.2.5
make distclean
make redis-cli BUILD_TLS=yes
sudo install -m 755 src/redis-cli /usr/local/bin/
```

Amazon Linux

如果使用的是 Amazon Linux，请输入以下命令：

```
sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel clang wget
wget https://download.redis.io/releases/redis-7.2.5.tar.gz
tar xvzf redis-7.2.5.tar.gz
cd redis-7.2.5
make redis-cli CC=clang BUILD_TLS=yes
sudo install -m 755 src/redis-cli /usr/local/bin/
```

在 Amazon Linux 上，您可能还需要执行以下额外步骤：

```
sudo yum install clang
CC=clang make
sudo make install
```

3. 下载并安装 redis-cli 实用程序后，建议运行可选的 `make-test` 命令。

4. 要连接到已启用加密和身份验证的集群，请输入以下命令：

```
redis-cli -h Primary or Configuration Endpoint --tls -a 'your-password' -p 6379
```

> **Note**
>
> 如果您在 Amazon Linux 2023 上安装 redis6，那么现在可以使用命令 `redis6-cli`，而不是使用 `redis-cli`：
>
> ```
> redis6-cli -h Primary or Configuration Endpoint --tls -p 6379
> ```

步骤 5：删除集群

只要集群处于可用 状态，您就绪需为它付费，无论您是否主动使用它。要停止产生费用，请删除此集群。

> **Warning**
>
> - 当您删除 MemoryDB 集群时，您的手动快照将保留。您也可以在删除集群之前创建最终快照。自动快照不会保留。有关更多信息，请参阅 快照和还原。
> - 创建最终快照需要 `CreateSnapshot` 权限。如果没有此权限，API 调用将失败，并出现 `Access Denied` 异常。

使用 AWS 管理控制台

以下过程从您的部署中删除单个集群。要删除多个集群，请对要删除的每个集群重复此过程。在开始删除一个集群的过程之前，您无需等待删除另一个集群完成。

删除集群

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 要选择要删除的集群，请从集群列表中选择该集群的名称旁边的单选按钮。在这种情况下，是您在步骤 2：创建集群 创建的集群的名称。
3. 对于操作，选择删除。

4. 首先选择是否在删除集群之前创建集群的快照，然后在确认框中输入 delete 并选择删除删除集群，或者选择取消保留集群。

   如果选择了 Delete，集群的状态将变为正在删除。

只要您的集群不再在集群列表中列出，您就无需为该集群付费。

使用 AWS CLI

以下代码删除集群 my-cluster。在这种情况下，将 my-cluster 替换为您在 步骤 2：创建集群 中创建的集群的名称。

```
aws memorydb delete-cluster --cluster-name my-cluster
```

delete-cluster CLI 操作仅删除一个集群。要删除多个集群，请对要删除的每个集群调用 delete-cluster。在删除一个集群之前，您无需等待删除另一个集群的完成。

对于 Linux、macOS 或 Unix：

```
aws memorydb delete-cluster \
    --cluster-name my-cluster \
    --region us-east-1
```

对于 Windows：

```
aws memorydb delete-cluster ^
    --cluster-name my-cluster ^
    --region us-east-1
```

有关更多信息，请参阅 delete-cluster。

使用 MemoryDB API

以下代码删除集群 my-cluster。在这种情况下，将 my-cluster 替换为您在 步骤 2：创建集群 中创建的集群的名称。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DeleteCluster
    &ClusterName=my-cluster
    &Region=us-east-1
```

```
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T220302Z
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210802T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210802T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

DeleteCluster API 操作仅删除一个集群。要删除多个集群，请对要删除的每个集群调用 DeleteCluster。在删除一个集群之前，您无需等待删除另一个集群的完成。

有关更多信息，请参阅 DeleteCluster。

后续步骤

至此，您已尝试入门练习，接下来可以探索以下部分以了解有关 MemoryDB 和可用工具的更多信息：

- 入门 AWS
- 用于 Amazon Web Services 的工具
- AWS 命令行界面
- MemoryDB API 参考。

管理节点

节点是 MemoryDB 部署中的最小构建模块。节点属于集群所含的分片。每个节点都运行创建集群或最后一次修改集群时选择的引擎版本。每个节点都有自己的域名服务（DNS）名称和端口。支持多种类型的 MemoryDB 节点，每种类型的节点具有不同的关联内存量和计算能力。

主题

- MemoryDB 节点和分片
- 受支持的节点类型
- MemoryDB 预留节点
- 替换节点

涉及节点的重要操作包括：

- 从集群中添加/移除节点
- 扩展
- 查找连接端点

MemoryDB 节点和分片

分片是节点层次结构（每个都包含在一个集群中）。分片支持复制。在一个分片内，一个节点充当读/写主节点。分片中的所有其他节点充当主节点的只读副本。MemoryDB 支持集群中的多个分片。此支持允许在 MemoryDB 集群中对数据进行分区。

MemoryDB 支持通过分片进行复制。API 操作 DescribeClusters 列出了成员节点、节点名称、端点等分片信息。

在创建 MemoryDB 集群后，可以对其进行修改（扩展或缩减）。有关更多信息，请参阅扩展和替换节点。

创建新集群时，可以使用旧集群中的数据为其设定种子，以免从头开始创建。如果您需要更改节点类型、引擎版本或从 Amazon ElastiCache（Redis OSS）迁移，这样做会很有用。有关更多信息，请参阅手动创建快照和从快照还原。

受支持的节点类型

MemoryDB 支持以下节点类型。

内存优化型： | | | |

实例类型	基准带宽 (Gbps)	突增带宽 (Gbps)	增强型 I/O 多路复用（Valkey 7.2 和 Redis OSS 7.0.4+）	最低引擎版本
db.r7g.large	0.937	12.5	否	6.2
db.r7g.xlarge	1.876	12.5	否	6.2
db.r7g.2xlarge	3.75	15	是	6.2
db.r7g.4xlarge	7.5	15	是	6.2
db.r7g.8xlarge	15	不适用	是	6.2
db.r7g.12xlarge	22.5	不适用	是	6.2
db.r7g.16xlarge	30	不适用	是	6.2
db.r6g.large	0.75	10.0	否	6.2
db.r6g.xlarge	1.25	10.0	否	6.2
db.r6g.2xlarge	2.5	10.0	是	6.2
db.r6g.4xlarge	5.0	10.0	是	6.2
db.r6g.8xlarge	12	不适用	是	6.2
db.r6g.12xlarge	20	不适用	是	6.2
db.r6g.16xlarge	25	不适用	是	6.2

利用数据分层功能优化内存

实例类型	基准带宽 (Gbps)	突增带宽 (Gbps)	增强型 I/O 多路复用（Valkey 7.2 和 Redis OSS 7.0.4+）	最低引擎版本
db.r6gd.xlarge	1.25	10	否	6.2
db.r6gd.2xlarge	2.5	10	否	6.2
db.r6gd.4xlarge	5.0	10	否	6.2
db.r6gd.8xlarge	12	不适用	否	6.2

通用型节点

实例类型	基准带宽 (Gbps)	突增带宽 (Gbps)	增强型 I/O 多路复用（Valkey 7.2 和 Redis OSS 7.0.4+）	最低引擎版本
db.t4g.small	0.128	5.0	否	6.2
db.t4g.medium	0.256	5.0	否	6.2

有关 AWS 区域可用性，请参阅 MemoryDB 定价

所有节点类型都在虚拟私有云（VPC）中创建。

MemoryDB 预留节点

相比按需节点定价，预留节点可以提供大幅折扣。预留节点不是物理节点，而是对账户中使用的按需型节点所应用的账单折扣。预留节点的折扣与节点类型和 AWS 区域相关联。

> **Note**
>
> 当前所有 MemoryDB 预留节点的定价均基于运行 Redis OSS 引擎的节点，并为这些节点提供支持或相关服务。这些保留节点可以应用于 Valkey 引擎，如 大小灵活的预留节点 中所述，但 Valkey 特定的保留节点不可用。

使用预留节点的一般过程如下：

- 查看有关可用预留节点产品的信息
- 使用 AWS 管理控制台、 AWS Command Line Interface 或 SDK 购买预留节点产品
- 查看有关您的现有预留节点的信息

主题

- 预留节点概述
- 产品类型
- 大小灵活的预留节点
- 将节点从 Redis OSS 升级到 Valkey
- 删除预留节点
- 使用预留节点

预留节点概述

如果购买了 MemoryDB 预留节点，将承诺在预留节点的持续时间内为您提供特定节点类型的折扣费率。要使用 MemoryDB 预留节点，应创建新节点，就像您为按需节点创建节点一样。创建的新节点必须与预留节点的规格完全匹配。如果新节点的规格与您的账户的现有预留节点匹配，则会按照为预留节点提供的折扣费率向您收费。否则，将以按需费率对节点进行收费。您可以使用 AWS 管理控制台 AWS CLI、或 MemoryDB API 列出和购买可用的预留节点产品。

MemoryDB 为内存优化型 R7g、R6g 和 R6gd（带数据分层）节点提供预留节点。有关定价信息，请参阅 MemoryDB 定价。

产品类型

预留节点有三种类型（无预付费用、预付部分费用和预付全部费用），使您可以基于预期使用情况优化 MemoryDB 成本。

无预付费用 – 该选项无需预付款即可访问预留节点。无论使用情况如何，您的“无费用预付”预留节点都将按照期限内的小时数，采用打折小时费率进行计费，无需任何预付款。

预付部分费用 – 该选项需要预付部分预留节点费用。无论使用量如何，期限内的剩余时数均按折扣小时费率计费。

预付全部费用 – 所有款项于期限开始时支付，无论使用了多少小时数，剩余期限不会再产生其他任何费用。

所有三种产品类型都以一年和三年的期限提供。

大小灵活的预留节点

购买预留节点时，您需要指定节点类型，例如 db.r6g.xlarge。有关节点类型的更多信息，请参阅 MemoryDB 定价。

如果您具有节点，则需要将其扩展为更大的容量，预留节点将自动应用于扩展的节点。即，同一节点系列中任何大小的使用率将自动应用预留节点。大小灵活的预留节点可用于具有相同 AWS 区域的节点。大小灵活的预留节点只能在其节点系列中横向缩减。例如，db.r6g.xlarge 的预留节点可以应用于 db.r6g.2xlarge，但不能应用于 db.r6gd.large，因为 db.r6g 和 db.r6gd 属于不同类型的节点系列。

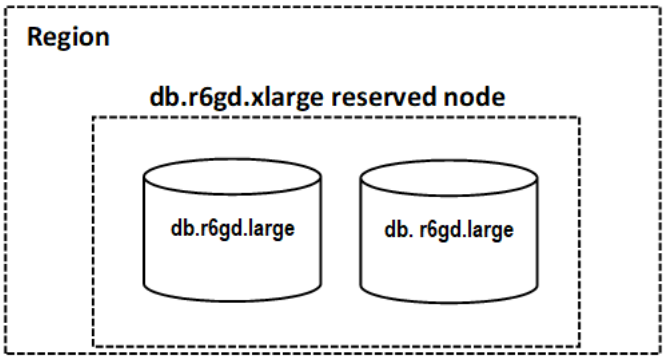
大小灵活性意味着您可以在相同节点系列的配置之间自由移动。例如，您可以从同一区域的 r6g.xlarge 预留节点（8 个标准化单位）移动到同一区域中的两个 r6g.large 保留节点（8 个标准化单位）（2*4 = 8 个标准化单位），无需支付额外费用。 AWS

您可以使用标准化单位比较不同预留节点大小的使用情况。例如，在两个 db.r6g.4xlarge 节点上使用一小时相当于在一个 db.r6g.large 节点上使用 16 个小时。下表显示了每个节点大小的标准化单位数：

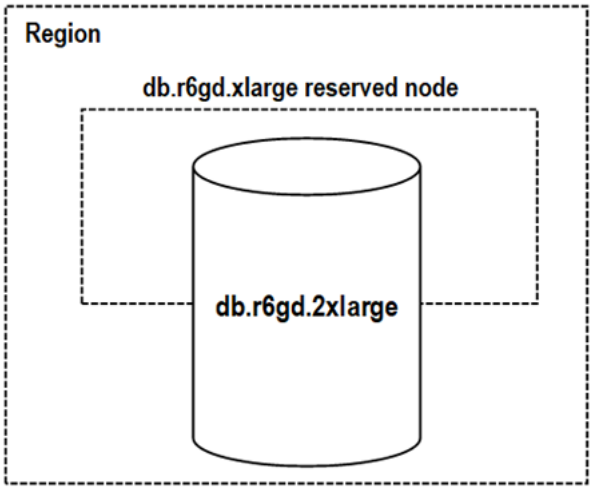
节点大小	标准化单位 (Redis OSS)	标准化单位（Valkey）
small	1	7.
medium	2	1.4
large	4	2.8

节点大小	标准化单位 (Redis OSS)	标准化单位（Valkey）
xlarge	8	5.6
2xlarge	16	11.2
4xlarge	32	22.4
6xlarge	48	33.6
8xlarge	64	44.8
10xlarge	80	56
12xlarge	96	67.2
16xlarge	128	89.6
24xlarge	192	134.4

例如，您购买了一个 db.r6gd.xlarge 预留节点，并且您的账户中有两个正在运行的 db.r6gd.large 预留节点位于同一区域。 AWS 在这种情况下，账单优惠将完全应用于两个节点。



或者，如果您在同一 AWS 地区的账户中运行一个 db.r6gd.2xlarge 实例，则账单优惠适用于预留节点使用量的 50%。



将节点从 Redis OSS 升级到 Valkey

随着 Valkey 在 MemoryDB 中的推出，您现在可以将 Redis OSS 预留节点折扣应用于 Valkey 引擎。您可以从 Redis OSS 升级到 Valkey，同时仍可从现有合同和预留中受益。除了能够在节点系列和引擎内应用您的权益外，您甚至可以获得更多的增量价值。Valkey 相对于 Redis OSS 有 30% 的价格折扣，并且凭借预留节点的灵活性，您可以使用 Redis OSS 预留节点来覆盖更多运行的 Valkey 节点。

为了计算折扣率，每个 MemoryDB 节点和引擎组合都有一个标准化因子，以单位衡量。预留节点单位可以应用于给定引擎的预留节点实例系列内的任何运行节点。此外，Redis OSS 预留节点还可以跨引擎应用，以覆盖正在运行的 Valkey 节点。由于 Valkey 相对于 Redis OSS 有折扣，其对于给定实例类型的单位较低，这使得 Redis OSS 预留节点可以覆盖更多 Valkey 节点。

例如，假设您为 Redis OSS 引擎购买了一个 db.r7g.4xlarge 的预留节点（32 单位），并且正在运行一个 db.r7g.4xlarge Redis OSS 节点（32 单位）。如果您将节点升级到 Valkey，运行节点的标准化因子降至 22.4 单位，而您现有的预留节点为您提供了额外的 9.6 单位，可用于该区域内 db.r7g 系列中的任何其他运行 Valkey 或 Redis OSS 节点。您可以使用这个来覆盖账户中另一个 db.r7g.4xlarge Valkey 节点的 42%（22.4 单位），或者 100% 覆盖一个 db.r7g.xlarge Valkey 节点（5.6 单位）和 100% 覆盖一个 db.r7g.large Valkey 节点（2.8 单位）。

删除预留节点

预留节点具有一年或三年的使用期限。您无法取消预留节点。不过，您可以删除预留节点折扣涵盖的节点。删除预留节点折扣涵盖的节点的过程与删除任何其他节点相同。

如果删除了预留节点折扣涵盖的节点，您可以启动另一个具有兼容规格的节点。在这种情况下，您可以在预留期限（一年或三年）内继续享受折扣费率。

使用预留节点

您可以使用 AWS 管理控制台 AWS Command Line Interface、和 MemoryDB API 来处理预留节点。

控制台

获取有关可用预留节点产品的定价和信息

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在导航窗格中，选择预留节点。
3. 选择购买预留节点。
4. 对于节点类型，请选择要部署的节点类型。
5. 对于数量，请选择要部署的节点数量。
6. 对于期限，选择希望预留数据库节点的时间长度。
7. 对于产品类型，请选择产品类型。

做出这些选择后，您可以在预留摘要中看到定价信息。

> **Important**
>
> 选择取消可避免购买这些预留节点和产生任何费用。

在获得有关可用预留节点产品的信息后，您可以使用该信息来购买以下过程中所示的产品：

要购买预留节点

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在导航窗格中，选择预留节点。
3. 选择购买预留节点。
4. 对于节点类型，请选择要部署的节点类型。
5. 对于数量，请选择要部署的节点数量。
6. 对于期限，选择希望预留数据库节点的时间长度。

7. 对于产品类型，请选择产品类型。
8. （可选）您可以将自己的标识符分配给购买的预留节点，以帮助您跟踪这些节点。对于预留 ID，请为您的预留节点键入一个标识符。

   做出这些选择后，您可以在预留摘要中看到定价信息。
9. 选择购买预留节点。
10. 您的预留节点已购买，然后显示在预留节点列表中。

获取有关您 AWS 账户的预留节点的信息

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在导航窗格中，选择预留节点。
3. 您的账户的预留节点出现。要查看有关特定预留节点的详细信息，请在列表中选择该节点。之后，您可以在详细信息中查看有关该节点的详细信息。

AWS Command Line Interface

以下 `describe-reserved-nodes-offerings` 示例将返回预留节点产品的详细信息。

```
aws memorydb describe-reserved-nodes-offerings
```

此操作将生成类似于下述信息的输出：

```
{
    "ReservedNodesOfferings": [
        {
            "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
            "NodeType": "db.xxx.large",
            "Duration": 94608000,
            "FixedPrice": $xxx.xx,
            "OfferingType": "Partial Upfront",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": $xx.xx,
                    "RecurringChargeFrequency": "Hourly"
```

```
                }
            ]
        }
    ]
}
```

您还可以传递以下参数以限制返回内容的范围：

- `--reserved-nodes-offering-id` – 您要购买的产品的 ID。
- `--node-type` – 节点类型筛选值。使用此参数仅显示与指定节点类型匹配的预留。
- `--duration` – 以年或秒为单位指定的持续时间筛选值。使用此参数仅显示此时段的预留。
- `--offering-type` – 使用此参数仅显示与指定产品类型匹配的可用产品。

在获得有关可用预留节点产品的信息后，您可以使用该信息来购买产品。

以下 `purchase-reserved-nodes-offering` 示例将购买新预留节点

对于 Linux、macOS 或 Unix：

```
aws memorydb purchase-reserved-nodes-offering \

    --reserved-nodes-offering-id 0193cc9d-7037-4d49-b332-d5e984f1d8ca \
    --reservation-id reservation \
    --node-count 2
```

对于 Windows：

```
aws memorydb purchase-reserved-nodes-offering ^
    --reserved-nodes-offering-id 0193cc9d-7037-4d49-b332-d5e984f1d8ca ^
    --reservation-id MyReservation
```

- `--reserved-nodes-offering-id` 表示要购买的预留节点产品的名称。
- `--reservation-id` 是一个用来跟踪此预留的客户指定的标识符。

> **Note**
>
> 预留 ID 是用来跟踪此预留的客户指定的唯一标识符。如果此参数未指定，则 MemoryDB 将自动生成此预留的标识符。

- --node-count 是要预留的节点数量。其默认值为 1。

此操作将生成类似于下述信息的输出：

```
{
    "ReservedNode": {
        "ReservationId": "reservation",
        "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
        "NodeType": "db.xxx.large",
        "StartTime": 1671173133.982,
        "Duration": 94608000,
        "FixedPrice": $xxx.xx,
        "NodeCount": 2,
        "OfferingType": "Partial Upfront",
        "State": "payment-pending",
        "RecurringCharges": [
            {
                "RecurringChargeAmount": $xx.xx,
                "RecurringChargeFrequency": "Hourly"
            }
        ],
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxx:reservednode/reservation"
    }
}
```

在购买了预留节点后，您可以获取有关预留节点的信息。

以下 describe-reserved-nodes 示例将返回有关该账户的预留节点的信息。

```
aws memorydb describe-reserved-nodes
```

此操作将生成类似于下述信息的输出：

```
{
    "ReservedNodes": [
        {
            "ReservationId": "ri-2022-12-16-00-28-40-600",
            "ReservedNodesOfferingId": "0193cc9d-7037-4d49-b332-xxxxxxxxxxxx",
            "NodeType": "db.xxx.large",
```

```
            "StartTime": 1671150737.969,
            "Duration": 94608000,
            "FixedPrice": $xxx.xx,
            "NodeCount": 1,
            "OfferingType": "Partial Upfront",
            "State": "active",
            "RecurringCharges": [
                {
                    "RecurringChargeAmount": $xx.xx,
                    "RecurringChargeFrequency": "Hourly"
                }
            ],
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxx:reservednode/
ri-2022-12-16-00-28-40-600"
        }
    ]
}
```

您还可以传递以下参数以限制返回内容的范围：

- `--reservation-id` – 您可以将自己的标识符分配给购买的预留节点，以帮助您跟踪这些实例。
- `--reserved-nodes-offering-id` – 产品标识符筛选值。使用此参数仅显示与指定产品标识符相匹配的已购买的预留。
- `--node-type` – 节点类型筛选值。使用此参数仅显示与指定节点类型匹配的预留。
- `--duration` – 以年或秒为单位指定的持续时间筛选值。使用此参数仅显示此时段的预留。
- `--offering-type` – 使用此参数仅显示与指定产品类型匹配的可用产品。

MemoryDB API

以下示例演示如何将 MemoryDB 查询 API 用于预留节点：

DescribeReservedNodesOfferings

返回预留节点产品的详细信息。

```
https://memorydb.us-west-2.amazonaws.com/
    ?Action=DescribeReservedNodesOfferings
    &ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f
 &"Duration": 94608000,
    &NodeType="db.r6g.large"
```

```
    &OfferingType="Partial Upfront"
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

以下参数限制了返回内容的范围：

- ReservedNodesOfferingId 表示要购买的预留节点产品的名称。
- Duration – 以年或秒为单位指定的持续时间筛选值。使用此参数仅显示此时段的预留。
- NodeType – 节点类型筛选值。使用此参数仅显示与指定节点类型匹配的产品。
- OfferingType – 使用此参数仅显示与指定产品类型匹配的可用产品。

在获得有关可用预留节点产品的信息后，您可以使用该信息来购买产品。

PurchaseReservedNodesOffering

允许您购买预留节点产品。

```
https://memorydb.us-west-2.amazonaws.com/
    ?Action=PurchasedReservedNodesOffering
    &ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f
    &ReservationID=myreservationID
    &NodeCount=1
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

- ReservedNodesOfferingId 表示要购买的预留节点产品的名称。

- ReservationID 是一个用来跟踪此预留的客户指定的标识符。

> **Note**
>
> 预留 ID 是用来跟踪此预留的客户指定的唯一标识符。如果此参数未指定，则 MemoryDB 将自动生成此预留的标识符。

- NodeCount 是要预留的节点数量。其默认值为 1。

在购买了预留节点后，您可以获取有关预留节点的信息。

DescribeReservedNodes

返回有关该账户的预留节点的信息。

```
https://memorydb.us-west-2.amazonaws.com/
 ?Action=DescribeReservedNodes
 &ReservedNodesOfferingId=649fd0c8-xxxx-xxxx-xxxx-06xxxx75e95f
 &ReservationID=myreservationID
 &NodeType="db.r6g.large"
 &Duration=94608000
 &OfferingType="Partial Upfront"
 &Version=2021-01-01
 &SignatureVersion=4
 &SignatureMethod=HmacSHA256
 &Timestamp=20141201T220302Z
 &X-Amz-Algorithm
 &X-Amz-SignedHeaders=Host
 &X-Amz-Expires=20141201T220302Z
 &X-Amz-Credential=<credential>
 &X-Amz-Signature=<signature>
```

以下参数限制了返回内容的范围：

- ReservedNodesOfferingId 表示预留节点的名称。
- ReservationID – 您可以将自己的标识符分配给购买的预留节点，以帮助您跟踪这些实例。
- NodeType – 节点类型筛选值。使用此参数仅显示与指定节点类型匹配的预留。
- Duration – 以年或秒为单位指定的持续时间筛选值。使用此参数仅显示此时段的预留。
- OfferingType – 使用此参数仅显示与指定产品类型匹配的可用产品。

查看预留节点的账单

您可以在 AWS 管理控制台中账单控制面板上查看预留节点的账单。

要查看预留节点账单

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 从控制台顶部的搜索按钮中，选择账单。
3. 从控制面板的左侧选择账单。
4. 在AWS 服务费用下，展开 MemoryDB。
5. 扩展您的预留节点所在的 AWS 区域，例如美国东部（弗吉尼亚北部）。

您的预留节点及其当月的小时费用显示在 Amazon MemoryDB CreateCluster 预留实例下。

Amazon MemoryDB CreateCluster Reserved Instances		
AmazonMemoryDB, db.r6g.large reserved instance applied	81.000 Hrs	
AmazonMemoryDB, db.r6g.4xlarge reserved instance applied	324.000 Hrs	
AmazonMemoryDB, db.r6g.4xlarge reserved instance applied	162.000 Hrs	
USD (hourly fee per AmazonMemoryDB, db.r6g.large instance	1,488.000 Hrs	
USD (hourly fee per AmazonMemoryDB, db.r6gd.2xlarge instance	744.000 Hrs	
USD 0 hourly fee per AmazonMemoryDB, db.r6g.4xlarge instance	744.000 Hrs	
USD (hourly fee per AmazonMemoryDB, db.r6gd.xlarge instance	744.000 Hrs	
USD 1 hourly fee per AmazonMemoryDB, db.r6gd.4xlarge instance	2,976.000 Hrs	

替换节点

MemoryDB 频繁（通常无缝地）升级其实例集。但是，我们需要经常重启您的 MemoryDB 节点，以将必需的操作系统更新应用于底层主机。我们需要进行升级来增强安全性、可靠性和操作性能，而应用这些升级就需要进行替换。

您还可以选择在计划节点替换时段之前的任意时间自己管理这些替换。当您自己管理替换时，您的实例将在重启节点时收到操作系统更新，并且您的计划节点替换将被取消。您可能会继续接收指示节点替换将发生的提醒。如果您已手动缓解对于维护的需求，则可以忽略这些提醒。

> **Note**
>
> 由 MemoryDB 自动生成的替换节点可能具有不同的 IP 地址。您负责查看应用程序配置，以确保节点与适当的 IP 地址关联。

以下列表标识了在 MemoryDB 计划替换节点时可执行的操作：

MemoryDB 节点替换选项

- 不执行任何操作 – 如果您不执行任何操作，则 MemoryDB 将按计划替换节点。

  如果节点是已启用多可用区的集群的成员，则 MemoryDB 可在修补、更新和其他与维护相关的节点替换期间提供更高的可用性。

  在集群处理传入的写请求时，完成替换。
- 更改维护时段 – 对于计划的维护事件，您将收到来自 MemoryDB 的电子邮件或通知事件。在这些情况下，如果在计划替换时间之前更改维护时段，则现在将在新时间替换您的节点。有关更多信息，请参阅 修改 MemoryDB 集群。

  > **Note**
  >
  > 仅当 MemoryDB 通知包括维护时段时，您可以通过移动维护时段的方式更改替换窗口。如果该通知不包括维护时段，您则无法更改替换窗口。

  例如，假设现在是 11 月 9 日星期四 15:00，下一个维护时段是 11 月 10 日星期五 17:00。下面是 3 种情况及其结果：
  - 您将维护时段更改为星期五 16:00，这在当前日期和时间之后且在下一个计划维护时段之前。将在 11 月 10 日星期五 16:00 替换节点。
  - 您将维护时段更改为星期六 16:00，这在当前日期和时间之后且在下一个计划维护时段之后。将在 11 月 11 日星期六 16:00 替换节点。
  - 您将维护时段更改为星期三 16:00，这在当前日期和时间之前。将在 11 月 15 日下一个星期三 16:00 替换节点。

  有关说明，请参阅管理维护。

管理集群

大多数 MemoryDB 操作在集群级别上执行。可以使用特定数量的节点和一个控制各个节点属性的参数组来设置集群。一个集群中的所有节点都应该是相同的节点类型，具有相同的参数和安全组设置。

每个集群必须有一个集群标识符。集群标识符是用户为集群提供的名称。此标识符指定了一个在与 MemoryDB API 和 AWS CLI 命令互动时的特殊集群。该客户在一个 AWS 区域中的集群标识符必须是唯一的。

MemoryDB 集群旨在通过 Amazon EC2 实例进行访问。您只能根据 Amazon VPC 服务在虚拟私有云（VPC）中启动 MemoryDB 集群，但是您可以从 AWS外部进行访问。有关更多信息，请参阅 从外部 AWS 访问 MemoryDB 资源。

数据分层

使用 r6gd 系列节点类型的集群将在内存和本地 SSD（固态硬盘）存储之间进行数据分层。借助数据分层功能，除可在内存中存储数据外，还可以在每个集群节点中使用成本更低的固态硬盘（SSD），从而为 Valkey 和 Redis OSS 工作负载提供新的高性价比选择。类似于其他节点类型，写入 r6gd 节点的数据持久存储在多可用区事务日志中。数据分层非常适合经常访问的数据不超过总体数据集的 20% 的工作负载，以及能够容忍访问 SSD 中数据时所出现的额外延迟的应用程序。

对于启用了数据分层功能的集群，MemoryDB 会监控集群所存储每个项目的最近访问时间。当可用内存（DRAM）耗尽时，MemoryDB 将使用最近最少使用（LRU）算法，自动将不频繁访问的项目从内存移动到 SSD 中。随后访问 SSD 上的数据时，MemoryDB 会在处理请求之前自动异步将其移回内存中。如果您的工作负载只会经常访问部分数据，则数据分层将是经济高效地扩缩容量的极佳方法。

请注意，使用数据分层时，键本身始终保留在内存中，而 LRU 将控制值在内存和磁盘上的位置。通常，在使用数据分层时，我们建议您的键大小小于值。

数据分层旨在将对应用程序工作负载的性能影响降至最低。例如，假设 500 字节的字符串值，与读取请求存储在内存中的数据相比，读取请求存储在 SSD 上的数据通常预计增加 450 微秒的延迟。

如果使用最大型号的数据分层节点（db.r6gd.8xlarge），您可以在单个 500 节点集群中存储最高 ~500TB 的数据（使用 1 个只读副本时 250TB）。对于数据分层，请 MemoryDB 为每个节点预留 19% 的（DRAM）内存用于非数据目的。数据分层功能兼容 MemoryDB 中支持的所有 Valkey 和 Redis OSS 命令和数据结构。使用此功能无需任何客户端更改。

主题

- 最佳实践
- 数据分层限制
- 数据分层定价
- 数据分层监控
- 数据分层功能的使用
- 将数据从快照还原到集群

最佳实践

我们建议您遵循以下最佳实践：

- 数据分层非常适合经常访问的数据不超过总体数据集的 20% 的工作负载，以及能够容忍访问 SSD 中数据时所出现的额外延迟的应用程序。

- 在数据分层节点上使用可用的 SSD 容量时，我们建议值大小大于键。值大小不能大于 128MB；否则无法将其移动到磁盘。在 DRAM 和 SSD 之间移动项目时，键将始终保留在内存中，并且只有值会移动到 SSD 层。

数据分层限制

数据分层功能存在以下限制：

- 您使用的节点类型必须属于 r6gd 系列，目前可在以下区域使用：us-east-2、us-east-1、us-west-2、us-west-1、eu-west-1、eu-west-3、eu-central-1、ap-northeast-1、ap-southeast-1、ap-southeast-2、ap-south-1、ca-central-1 和 sa-east-1。
- 除非两个集群都为 r6gd 集群，否则不能将 r6gd 集群的快照还原到其他集群。
- 不能将使用数据分层功能的集群快照导出到 Amazon S3。
- 不支持无分支保存。
- 不支持将使用数据分层功能的集群（例如，使用 r6gd 节点类型的集群）扩缩至不使用数据分层功能的集群（例如，使用 r6g 节点类型的集群）。
- 数据分层仅支持 volatile-lru、allkeys-lru 和 noeviction maxmemory 策略。
- 大于 128MiB 的项目不会移动到 SSD。

数据分层定价

与 R6g 节点（仅内存）相比，R6gd 节点的总存储容量（内存 + SSD）提高了 5 倍，以最大利用率运行时可帮助实现超过 60% 的存储成本节省。有关更多信息，请参阅 MemoryDB 定价。

数据分层监控

MemoryDB 提供了若干专用于监控使用数据分层功能的高性能集群的指标。要监控 DRAM 中项目与 SSD 中项目的比率，您可以在 MemoryDB 的指标 使用 CurrItems 指标。您可以按以下方式计算百分比：(CurrItems with Dimension: Tier = Memory * 100) / (CurrItems with no dimension filter)。

如果配置的驱逐策略允许，则当内存中项目的百分比降至 5% 以下时，MemoryDB 将开始驱逐项目。在配置了 noeviction 策略的节点上，写入操作将收到内存不足错误。

当内存中项目的百分比降至 5% 以下时，仍建议您考虑 扩展 MemoryDB 集群。有关更多信息，请参阅 MemoryDB 的指标 中的适用于使用数据分层功能的 MemoryDB 集群的指标。

数据分层功能的使用

通过 AWS 管理控制台使用数据分层功能

您可在创建集群时选择 r6gd 系列的节点类型（例如 db.r6gd.xlarge），从而使用数据分层功能。选择该节点类型将会自动启用数据分层功能。

有关创建集群的更多信息，请参阅步骤 2：创建集群。

通过 AWS CLI 启用数据分层

您可在使用 AWS CLI 创建集群时选择 r6gd 系列的节点类型（例如 db.r6gd.xlarge）并设置 `--data-tiering` 参数，从而使用数据分层功能。

选择 r6gd 系列的节点类型时，您将不能选择停止使用数据分层功能。如果您设置 `--no-data-tiering` 参数，操作将会失败。

对于 Linux、macOS 或 Unix：

```
aws memorydb create-cluster \
   --cluster-name my-cluster \
   --node-type db.r6gd.xlarge \
   --engine valkey  \
   --acl-name my-acl \
   --subnet-group my-sg \
   --data-tiering
```

对于 Windows：

```
aws memorydb create-cluster ^
   --cluster-name my-cluster ^
   --node-type db.r6gd.xlarge ^
   --engine valkey ^
   --acl-name my-acl ^
   --subnet-group my-sg
   --data-tiering
```

运行此操作后，您将会看到一条与以下类似的响应：

```
{
    "Cluster": {
```

```
        "Name": "my-cluster",
        "Status": "creating",
        "NumberOfShards": 1,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Port": 6379
        },
        "NodeType": "db.r6gd.xlarge",
        "EngineVersion": "7.2",
        "EnginePatchVersion": "7.2.6",
        "Engine": "valkey"
        "ParameterGroupName": "default.memorydb-valkey7",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxxxx:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "ACLName": "my-acl",
        "DataTiering":"true",
        "AutoMinorVersionUpgrade": true
    }
}
```

将数据从快照还原到集群

您可以通过（控制台）、（AWS CLI）或（MemoryDB API）将数据从快照还原到启用数据分层的新集群。当您使用 r6gd 系列的节点类型创建集群时，系统会启用数据分层。

将数据从快照还原到启用数据分层的集群（控制台）

要将快照还原到启用数据分层的新集群（控制台），请按照 从快照还原（控制台） 中的步骤操作

请注意，如要启用数据分层，需要选择 r6gd 系列的节点类型。

将数据从快照还原到启用数据分层的集群（AWS CLI）

使用 AWS CLI 创建集群时，选择 r6gd 系列的节点类型（例如 db.r6gd.xlarge）并设置 `--data-tiering` 参数后，系统会默认启用数据分层。

选择 r6gd 系列的节点类型时，您将不能选择停止使用数据分层功能。如果您设置 `--no-data-tiering` 参数，操作将会失败。

对于 Linux、macOS 或 Unix：

```
aws memorydb create-cluster \
   --cluster-name my-cluster \
   --node-type db.r6gd.xlarge \
   --engine valkey
   --acl-name my-acl \
   --subnet-group my-sg \
   --data-tiering \
   --snapshot-name my-snapshot
```

对于 Windows：

```
aws memorydb create-cluster ^
   --cluster-name my-cluster ^
   --node-type db.r6gd.xlarge ^
   --engine valkey ^
   --acl-name my-acl ^
   --subnet-group my-sg ^
   --data-tiering ^
   --snapshot-name my-snapshot
```

运行此操作后，您将会看到一条与以下类似的响应：

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "creating",
        "NumberOfShards": 1,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Port": 6379
        },
        "NodeType": "db.r6gd.xlarge",
        "EngineVersion": "7.2",
        "EnginePatchVersion": "7.2.6",
        "Engine": "valkey"
        "ParameterGroupName": "default.memorydb-valkey7",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxxxxxxxxx:cluster/my-cluster",
```

```
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "ACLName": "my-acl",
        "DataTiering": "true"
}
```

准备集群

接下来，可找到有关使用 MemoryDB 控制台、 AWS CLI或 MemoryDB API 创建集群的说明。

每当创建集群时，最好做一些准备工作，这样就无需立即升级或进行更改。

主题

- 确定要求

确定要求

准备

了解以下问题的答案有助于使集群的创建更加流畅：

- 开始创建集群之前，请确保同一 VPC 中创建一个子网组。或者使用提供的默认子网组。有关更多信息，请参阅 子网和子网组。

  MemoryDB 专为 AWS 使用 Amazon EC2 从内部访问而设计。但是，如果根据 Amazon VPC 在 VPC 中启动，则可以提供从 AWS外部进行访问的权限。有关更多信息，请参阅 从外部 AWS 访问 MemoryDB 资源。
- 您是否需要自定义任何参数值？

  如果这样做，请创建自定义参数组。有关更多信息，请参阅 创建参数组。
- 您是否需要创建 VPC 安全组？

  有关更多信息，请参阅您的 VPC 的安全性。
- 您想如何实现容错？

  有关更多信息，请参阅 缓解故障。

主题

- 内存和处理器要求
- MemoryDB 集群配置
- 增强的 I/O 多路复用
- 扩展要求
- 访问要求
- 区域和可用区

内存和处理器要求

MemoryDB 的基本构建模块是节点。节点在分片中配置以形成集群。在确定用于集群的节点类型时，请考虑集群的节点配置以及必须存储的数据量。

MemoryDB 集群配置

MemoryDB 集群由 1 到 500 个分片组成。MemoryDB 集群中的数据在集群的分片间分区。您的应用程序使用称为端点的网络地址与 MemoryDB 集群连接。除了节点端点外，MemoryDB 集群本身还具有一个称为集群端点的端点。您的应用程序可以使用此端点来读取或写入集群，从而由 MemoryDB 决定要读取或写入的节点。

增强的 I/O 多路复用

如果您运行的是 Valkey 或 Redis OSS 7.0 或更高版本，则通过增强的 I/O 多路复用将获得额外的加速，其中每个专用的网络 IO 利用高效地批量处理命令的能力，将来自多个客户端的命令线程化到引擎中。有关更多信息，请参阅超快性能和 the section called “受支持的节点类型”。

扩展要求

所有集群都可以纵向扩展为更大的节点类型。纵向扩展 MemoryDB 集群时，您可以联机操作，以确保集群保持可用，也可以从快照为新集群做种，以免新集群启动时为空。

有关更多信息，请参阅本指南中的扩展。

访问要求

根据设计，MemoryDB 集群可通过 Amazon EC2 实例进行访问。对 MemoryDB 集群的网络访问限制为创建该集群的账户。因此，必须先授权对集群的输入，然后您才能从 Amazon EC2 实例访问集群。有关详细说明，请参阅本指南中的 步骤 3：授予对集群的访问权限。

区域和可用区

通过将 MemoryDB 集群放置在靠近应用程序的 AWS 区域中，可以减少延迟。如果集群有多个节点，将节点放置在不同的可用区可减少故障对集群的影响。

有关更多信息，请参阅下列内容：

- 选择区域和可用区
- 缓解故障

创建集群

MemoryDB 提供了三种创建集群的方法。有关更多信息，请参阅 步骤 2：创建集群。

查看集群的详细信息

您可以使用 MemoryDB 控制台或 MemoryDB API 查看有关一个或多个集群的详细信息。 AWS CLI

查看 MemoryDB 集群的详细信息（控制台）

以下过程详细说明了如何使用 MemoryDB 控制台查看 MemoryDB 集群的详细信息。

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 要查看集群的详细信息，请选择集群名称左侧的单选按钮，然后选择查看详细信息。您还可以直接单击集群查看集群详细信息页面。

   集群详细信息页面显示集群的详细信息，包括集群端点。您可以使用集群详细信息页面中的多个选项卡查看更多详细信息。
3. 要查看集群分片的列表及每个分片的节点数量，请选择分片和节点选项卡。
4. 要查看有关节点的特定信息，请展开下表中的分片。或者使用搜索框搜索分片。

   这样做会显示有关每个节点的信息，包括其可用 slots/keyspaces 区和状态。
5. 选择指标选项卡监控各指标的进程，例如 CPU 利用率和引擎 CPU 利用率。有关更多信息，请参阅 MemoryDB 的指标。
6. 选择网络和安全性选项卡查看子网组和安全组的详细信息。

   a. 在子网组中，可以查看子网组名称、子网所属 VPC 的链接以及子网组的 Amazon 资源名称（ARN）。

   b. 在安全组中，可以查看安全组 ID、名称和描述。
7. 选择维护和快照选项卡查看快照设置的详细信息。

   a. 在快照中，可以查看是否启用了自动快照、快照保留期和快照时段。

   b. 在快照中，可以查看该集群的所有快照列表，包括快照名称、大小、分片数量和状态。

   有关更多信息，请参阅 快照和还原。
8. 选择维护和快照选项卡查看维护时段的详细信息以及所有待处理的 ACL、重新分片或服务更新。有关更多信息，请参阅 管理维护。
9. 选择服务更新选项卡查看适用于此集群的任何服务更新的详细信息。有关更多信息，请参阅 MemoryDB 中的服务更新。

10. 选择标签选项卡查看与此集群关联的任何资源或成本分配标签的详细信息。有关更多信息，请参阅标记快照。

查看集群的详细信息 (AWS CLI)

您可以使用 AWS CLI `describe-clusters`命令查看集群的详细信息。如果省略 `--cluster-name` 参数，则会返回多个集群（最多 `--max-results` 个）的详细信息。如果包含 `--cluster-name` 参数，则将返回指定的集群的详细信息。您可以使用 `--max-results` 参数限制返回的记录数。

以下代码列出了 `my-cluster` 的详细信息。

```
aws memorydb describe-clusters --cluster-name my-cluster
```

以下代码列出了最多 25 个集群的详细信息。

```
aws memorydb describe-clusters --max-results 25
```

Example

对于 Linux、macOS 或 Unix：

```
aws memorydb describe-clusters \
    --cluster-name my-cluster \
    --show-shard-details
```

对于 Windows：

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster ^
    --show-shard-details
```

以下 JSON 输出显示响应：

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Description": "my cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
```

```
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": 1629230643.961,
                            "Endpoint": {
                                "Address": "my-cluster-0001-001.my-
cluster.abcdef.memorydb.us-east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "CreateTime": 1629230644.025,
                            "Endpoint": {
                                "Address": "my-cluster-0001-002.my-
cluster.abcdef.memorydb.us-east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.abcdef.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "default",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:000000000:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
```

```
            "MaintenanceWindow": "sat:06:30-sat:07:30",
            "SnapshotWindow": "04:00-05:00",
            "ACLName": "open-access",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true,
        }
```

有关更多信息，请参阅 for MemoryDB 主题。 AWS CLI describe-clusters

查看集群的详细信息（MemoryDB API）

您可以使用 MemoryDB API DescribeClusters 操作查看集群的详细信息。如果包含 ClusterName 参数，则将返回指定的集群的详细信息。如果省略 ClusterName 参数，则会返回最多 MaxResults 个（默认 100 个）集群的详细信息。MaxResults 的值不能小于 20 或大于 100。

以下代码列出了 my-cluster 的详细信息。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeClusters
   &ClusterName=my-cluster
   &Version=2021-01-01
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

以下代码列出了最多 25 个集群的详细信息。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeClusters
   &MaxResults=25
   &Version=2021-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 MemoryDB API 参考主题 DescribeClusters。

修改 MemoryDB 集群

除了对集群添加或移除节点外，有时您可能还需要对现有集群做出其他更改，如添加安全组、更改维护时段或参数组。

我们建议您将维护时段设置在使用率最低的时间内。因此，维护时段需要不时进行修改。

当您更改集群的参数时，更改会立即应用到集群。无论您是更改集群的参数组本身，还是更改集群参数组内的参数值，都是如此。

您还可以更新集群的引擎版本。例如，您可以选择新的引擎次要版本，MemoryDB 将立即开始更新集群。

使用 AWS 管理控制台

修改集群

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 从右上角的列表中，选择要修改的集群所在的 AWS 区域。
3. 在左侧导航栏中，转到集群。在集群的详细信息中，使用单选按钮选择集群，并转到操作，然后选择修改。
4. 此时会显示修改页面。
5. 在修改窗口中，根据需要做出修改。选项包括：

   - 说明
   - 子网组
   - VPC 安全组
   - 节点类型

   > **Note**
   >
   > 如果集群使用 r6gd 系列的节点类型，则只能选择该系列中的不同节点大小。如果您选择 r6gd 系列的节点类型，则系统会自动启用数据分层。有关更多信息，请参阅 数据分层。

   - Valkey 或 Redis OSS 版本兼容性
   - 启用自动快照

- 快照保留期
- 快照窗口
- 维护窗口
- SNS 通知的主题

6. 选择保存更改。

您还可以转到集群详细信息页面，然后单击修改对集群进行修改。如果要修改集群的特定章节，请转到集群详细信息页面中的相应选项卡，然后单击修改。

使用 AWS CLI

您可以使用 AWS CLI `update-cluster`操作修改现有集群。要修改集群的配置值，请指定集群的 ID、要更改的参数和此参数的新值。以下示例更改名为 `my-cluster` 的集群的维护时段，并立即应用此更改。

对于 Linux、macOS 或 Unix：

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --preferred-maintenance-window sun:23:00-mon:02:00
```

对于 Windows：

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --preferred-maintenance-window sun:23:00-mon:02:00
```

有关更多信息，请参阅《命令参考》中的 update-cluster。 AWS CLI

使用 MemoryDB API

您可以使用 MemoryDB API UpdateCluster操作修改现有集群。要修改集群的配置值，请指定集群的 ID、要更改的参数和此参数的新值。以下示例更改名为 `my-cluster` 的集群的维护时段，并立即应用此更改。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateCluster
    &ClusterName=my-cluster
```

```
    &PreferredMaintenanceWindow=sun:23:00-mon:02:00
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210802T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

如何触发从 Redis OSS 到 Valkey 的跨引擎升级

您可以使用控制台、API 或 CLI 将现有 Redis OSS 集群升级到 Valkey 引擎。

如果您有一个使用默认参数组的现有 Redis OSS 集群，则可以通过使用 update-cluster API 指定新的引擎和引擎版本来升级到 Valkey。

对于 Linux、macOS 或 Unix：

```
aws memorydb update-cluster \
   --cluster-name myCluster \
   --engine valkey \
   --engine-version 7.2
```

对于 Windows：

```
aws memorydb update-cluster ^
   --cluster-name myCluster ^
   --engine valkey ^
   --engine-version 7.2
```

如果您对要升级的现有 Redis OSS 集群应用了自定义参数组，则还需要在请求中传递自定义 Valkey 参数组。输入 Valkey 自定义参数组必须具有与现有 Redis OSS 自定义参数组相同的 Redis OSS 静态参数值。

对于 Linux、macOS 或 Unix：

```
aws memorydb update-cluster \
   --cluster-name myCluster \
   --engine valkey \
```

```
   --engine-version 7.2 \
   --parameter-group-name myParamGroup
```

对于 Windows：

```
aws memorydb update-cluster ^
   --cluster-name myCluster ^
   --engine valkey ^
   --engine-version 7.2 ^
   --parameter-group-name myParamGroup
```

从集群中添加/移除节点

您可以使用 AWS 管理控制台、或 MemoryDB API 在集群中 AWS CLI添加或移除节点。

使用 AWS 管理控制台

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 从集群列表中，选择要从中添加或删除节点的集群的名称。
3. 在分片和节点选项卡下，选择添加/删除节点
4. 在节点数中，输入所需的节点数。
5. 选择确认。

> **Important**
> 如果您将节点数量设为 1，则无法再启动多可用区。您还可以选择启用自动失效转移。

使用 AWS CLI

1. 确定要删除的节点的名称。有关更多信息，请参阅 查看集群的详细信息。
2. 将 `update-cluster` CLI 操作与要删除的节点列表一起使用，如下例所示。

   要使用命令行界面从集群中移除节点，请结合以下参数使用命令 `update-cluster`：

   - `--cluster-name` 要从其中删除节点的集群的 ID。
   - `--replica-configuration` – 允许您设置副本的数量：
     - `ReplicaCount` – 设置此属性指定想要的副本节点数量。
   - `--region`指定要从中移除节点的集群 AWS 区域。

   对于 Linux、macOS 或 Unix：

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --replica-configuration \
        ReplicaCount=1 \
```

```
    --region us-east-1
```

对于 Windows：

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --replica-configuration ^
        ReplicaCount=1 ^
    --region us-east-1
```

有关更多信息，请参阅 AWS CLI 主题update-cluster。

使用 MemoryDB API

要使用 MemoryDB API 删除节点，请使用集群名称和要删除节点的列表调用 UpdateCluster API 操作，如下所示：

- ClusterName 要从其中删除节点的集群的 ID。
- ReplicaConfiguration – 允许您设置副本的数量：
  - ReplicaCount – 设置此属性指定想要的副本节点数量。
- Region指定要从中移除节点的集群 AWS 区域。

有关更多信息，请参阅 UpdateCluster。

访问您的集群

MemoryDB 实例设计为通过 Amazon EC2 实例对其进行访问。

您可以从同一 Amazon VPC 中的 Amazon EC2 实例访问 MemoryDB 节点。或者，通过使用 VPC 对等连接，您可以从不同 Amazon VPC 中的 Amazon EC2 访问您的 MemoryDB 节点。

主题

- 授予对集群的访问权限
- 从外部 AWS 访问 MemoryDB 资源

授予对集群的访问权限

您只能从正在同一 Amazon VPC 中运行的 Amazon EC2 实例连接到您的 MemoryDB 集群。在此情况下，您需要向集群授予网络进入。

授予从 Amazon VPC 安全组到集群的网络入口

1. 登录到 AWS 管理控制台 并打开 Amazon EC2 控制台（https://console.aws.amazon.com/ec2/）。
2. 在左侧导航窗格中的网络和安全下，选择安全组。
3. 从安全组列表中，为 Amazon VPC 选择安全组。除非创建安全组供 MemoryDB 使用，否则此安全组将命名为默认。
4. 选择 Inbound 选项卡，然后执行以下操作：

    a. 选择 Edit（编辑）。

    b. 选择添加规则。

    c. 在 Type 列中，选择 Custom TCP rule。

    d. 在 Port range 框中，为您的集群节点键入端口号。此端口号必须与启动集群时指定的端口号相同。Valkey 和 Redis OSS 的默认端口均为 **6379**。

    e. 在源框中，选择端口范围为（0.0.0.0/0）的任何位置，以便从 Amazon VPC 中启动的任何 Amazon EC2 实例都可以连接到您的 MemoryDB 节点。

    > **Important**
    >
    > 向 0.0.0.0/0 公开 MemoryDB 集群时，不会在互联网上公开集群，因为它没有公有 IP 地址，因此无法从 VPC 外部访问。但是，默认安全组可以应用到客户账户中的其他

Amazon EC2 实例，这些实例可能具有公有 IP 地址。如果这些实例碰巧在默认端口上运行某些内容，则该服务可能会意外暴露。因此，我们建议创建将由 MemoryDB 独占使用的 VPC 安全组。有关更多信息，请参阅自定义安全组。

f. 选择保存。

当您将 Amazon EC2 实例启动到您的 Amazon VPC 中时，该实例将能够连接到您的 MemoryDB 集群。

从外部 AWS 访问 MemoryDB 资源

MemoryDB 是一项设计为在 VPC 内部使用的服务。由于 Internet 流量的延迟以及安全问题，不鼓励外部访问。但是，如果出于测试或开发目的需要对 MemoryDB 进行外部访问，则可以通过 VPN 完成。

使用 AWS 客户端 VPN，您允许对 MemoryDB 节点进行外部访问且具有以下优势：

- 限制访问获得批准的用户或身份验证密钥；
- VPN 客户端与 AWS VPN 端点之间的加密流量；
- 对特定子网或节点的限制访问；
- 轻松撤消对用户或身份验证密钥的访问；
- 审核连接；

以下过程演示如何：

主题

- 创建证书颁发机构
- 配置 AWS 客户端 VPN 组件
- 配置 VPN 客户端

创建证书颁发机构

可以使用不同的技术或工具创建证书颁发机构（CA）。我们建议使用 OpenVPN 项目提供的 easy-rsa 实用程序。无论您选择哪种选项，请确保密钥安全。以下过程下载 easy-rsa 脚本，创建证书颁发机构和用于验证第一个 VPN 客户端的密钥：

- 要创建初始证书，请打开终端并执行以下操作：
  - `git clone` https://github.com/OpenVPN/easy-rsa
  - `cd easy-rsa`
  - `./easyrsa3/easyrsa init-pki`
  - `./easyrsa3/easyrsa build-ca nopass`
  - `./easyrsa3/easyrsa build-server-full server nopass`
  - `./easyrsa3/easyrsa build-client-full client1.domain.tld nopass`

  pki 子目录包含将在 easy-rsa 下创建的证书。

- 将服务器证书提交给 AWS Certificate Manager（ACM）：
  - 在 ACM 控制台上，选择 Certificate Manager。
  - 选择导入证书。
  - 将 `easy-rsa/pki/issued/server.crt` 文件中提供的公有密钥证书输入到证书文本字段中。
  - 在 Certificate private key（证书私有密钥）中 `easy-rsa/pki/private/server.key` 的粘贴可用私有密钥。确保选择 `BEGIN AND END PRIVATE KEY` 之间的所有行（包括 `BEGIN` 和 `END` 行）。
  - 将 `easy-rsa/pki/ca.crt` 文件中提供的 CA 公用密钥粘贴到证书链字段中。
  - 选择查看并导入。
  - 选择导入。

  要使用 AWS CLI 将服务器的证书提交给 ACM，请运行以下命令：`aws acm import-certificate --certificate fileb://easy-rsa/pki/issued/server.crt --private-key file://easy-rsa/pki/private/server.key --certificate-chain file://easy-rsa/pki/ca.crt --region region`

  请记下证书 ARN 以供将来使用。

配置 AWS 客户端 VPN 组件

使用 AWS 控制台

在 AWS 控制台上，选择 Services（服务），然后选择 VPC。

在虚拟专用网下，选择客户端 VPN 终端节点并执行以下操作：

配置 AWS Client VPN 组件

- 选择创建客户端 VPN 端点。
- 指定以下选项：
  - 客户端 IPv4 CIDR：使用具有至少 /22 范围的网络掩码的专用网络。确保所选子网与 VPC 网络的地址不冲突。示例：10.0.0.0/22。
  - 在服务器证书 ARN 中，选择之前导入的证书的 ARN。
  - 选择使用双向身份验证。
  - 在客户端证书 ARN 中，选择之前导入的证书的 ARN。
  - 选择创建客户端 VPN 端点。

使用 AWS CLI

运行以下命令：

```
aws ec2 create-client-vpn-endpoint --client-cidr-block
"10.0.0.0/22" --server-certificate-arn arn:aws:acm:us-
east-1:012345678912:certificate/0123abcd-ab12-01a0-123a-123456abcdef --
authentication-options Type=certificate-
authentication,,MutualAuthentication={ClientRootCertificateChainArn=arn:aws:acm:
east-1:012345678912:certificate/123abcd-ab12-01a0-123a-123456abcdef} --
connection-log-options Enabled=false
```

输出示例：

```
"ClientVpnEndpointId": "cvpn-endpoint-0123456789abcdefg",
"Status": { "Code": "pending-associate" }, "DnsName": "cvpn-
endpoint-0123456789abcdefg.prod.clientvpn.us-east-1.amazonaws.com" }
```

将目标网络关联到 VPN 终端节点

- 选择新的 VPN 终端节点，然后选择关联选项卡。
- 选择关联并指定以下选项。
  - VPC：选择 MemoryDB 集群的 VPC。
  - 选择其中一个 MemoryDB 集群的网络。如果有疑问，请在 MemoryDB 控制面板上查看子网组 中的网络。
  - 选择关联。如有必要，请为其余网络重复执行这些步骤。

使用 AWS CLI

运行以下命令：

```
aws ec2 associate-client-vpn-target-network --client-vpn-endpoint-id cvpn-
endpoint-0123456789abcdefg --subnet-id subnet-0123456789abdcdef
```

输出示例：

```
"Status": { "Code": "associating" }, "AssociationId": "cvpn-
assoc-0123456789abdcdef" }
```

查看 VPN 安全组

VPN 终端节点将自动采用 VPC 的默认安全组。检查入站和出站规则，并确认安全组是否允许从 VPN 网络（在 VPN 端点设置中定义）到服务端口上的 MemoryDB 网络的流量（默认情况下，对于 Redis 为 6379）。

如果您需要更改分配给 VPN 终端节点的安全组，请按以下步骤操作：

- 选择当前安全组。
- 选择应用安全组。
- 选择新的安全组。

使用 AWS CLI

运行以下命令：

```
aws ec2 apply-security-groups-to-client-vpn-target-network --
client-vpn-endpoint-id cvpn-endpoint-0123456789abcdefga  --vpc-id
vpc-0123456789abdcdef --security-group-ids sg-0123456789abdcdef
```

输出示例：

```
"SecurityGroupIds": [ "sg-0123456789abdcdef" ] }
```

> **Note**
>
> MemoryDB 安全组还需要允许来自 VPN 客户端的流量。根据 VPC 网络，客户端的地址将被 VPN 终端节点地址掩盖。因此，在 MemoryDB 安全组上创建入站规则时，请考虑 VPC 网络（而不是 VPN 客户端的网络）。

授权 VPN 访问目标网络

在授权选项卡上，选择授权入口并指定以下内容：

- 启用访问的目标网络：使用 0.0.0.0/0 以允许访问任何网络（包括互联网），或限制 MemoryDB 网络/主机。
- 在授予访问权限：下，选择允许访问所有用户。
- 选择添加授权规则。

使用 AWS CLI

运行以下命令：

```
aws ec2 authorize-client-vpn-ingress --client-vpn-endpoint-id cvpn-
endpoint-0123456789abcdefg --target-network-cidr 0.0.0.0/0 --authorize-all-
groups
```

输出示例：

```
{ "Status": { "Code": "authorizing" } }
```

允许从 VPN 客户端访问 Internet

如果您需要通过 VPN 浏览 Internet，则需要创建一个额外的路由。选择路由表选项卡，然后单击创建路由。

- 路由目的地：0.0.0.0/0
- 目标 VPC 子网 ID：选择可访问 Internet 的关联子网之一。
- 选择创建路由。

使用 AWS CLI

运行以下命令：

```
aws ec2 create-client-vpn-route --client-vpn-endpoint-id cvpn-
endpoint-0123456789abcdefg --destination-cidr-block 0.0.0.0/0 --target-vpc-
subnet-id subnet-0123456789abdcdef
```

输出示例：

```
{ "Status": { "Code": "creating" } }
```

配置 VPN 客户端

在 AWS Client VPN 控制面板上，选择最近创建的 VPN 端点，然后选择 Download Client Configuration（下载客户端配置）。复制配置文件，以及文件 `easy-rsa/pki/issued/client1.domain.tld.crt` 和 `easy-rsa/pki/private/client1.domain.tld.key`。编辑配置文件并更改或添加以下参数：

- cert：添加一个新行，其参数证书指向 `client1.domain.tld.crt` 文件。使用到文件的完整路径。示例：`cert /home/user/.cert/client1.domain.tld.crt`

- cert: key：添加一个新行，其参数键指向 client1.domain.tld.key 文件。使用到文件的完整路径。示例：key /home/user/.cert/client1.domain.tld.key

使用以下命令建立 VPN 连接：sudo openvpn --config downloaded-client-config.ovpn

撤消访问权限

如果您需要使来自特定客户端密钥的访问失效，则需要在 CA 中撤消该密钥。然后，将吊销列表提交到 AWS Client VPN。

使用 easy-rsa 撤消密钥：

- cd easy-rsa
- ./easyrsa3/easyrsa revoke client1.domain.tld
- 输入“是” 以继续，或输入任何其他输入以中止。

  Continue with revocation: `yes` ... * `./easyrsa3/easyrsa gen-crl
- 已创建更新的 CRL。CRL 文件：/home/user/easy-rsa/pki/crl.pem

将吊销列表导入到 AWS Client VPN：

- 在 AWS 管理控制台 上，选择服务，然后选择 VPC。
- 选择客户端 VPN 端点。
- 选择客户端 VPN 端点，然后选择操作 -> 导入客户端证书 CRL。
- 查看 crl.pem 文件的内容。

使用 AWS CLI

运行以下命令：

```
aws ec2 import-client-vpn-client-certificate-revocation-list --certificate-
revocation-list file://./easy-rsa/pki/crl.pem --client-vpn-endpoint-id
cvpn-endpoint-0123456789abcdefg
```

输出示例：

```
Example output: { "Return": true }
```

查找连接端点

您的应用程序使用端点连接到集群。端点是集群的唯一的地址。使用该集群的集群端点进行所有操作。

以下部分将引导您发现所需的端点。

查找 MemoryDB 集群（AWS 管理控制台）的端点

查找 MemoryDB 集群的端点

1. 登录到 AWS 管理控制台 并打开 MemoryDB 控制台，网址：https://console.aws.amazon.com/memorydb/。
2. 从导航窗格中，选择集群。

   集群屏幕随即出现，其中显示集群的列表。选择您想要连接到的集群。
3. 要查找集群的端点，请选择集群的名称（不是单选按钮）。
4. 集群详细信息下将显示集群端点。要复制它，请选择位于端点左侧的 copy（复制）图标。

查找 MemoryDB 集群的端点（AWS CLI）

您可以使用 `describe-clusters` API 命令来搜索集群的端点。该命令将返回集群的端点。

以下操作检索集群 `mycluster` 的端点，本例以*##*形式表示。

返回以下 JSON 响应：

```
aws memorydb describe-clusters \
  --cluster-name mycluster
```

对于 Windows：

```
aws memorydb describe-clusters ^
   --cluster-name mycluster
```

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
```

```
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.4",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:zzzexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

有关更多信息，请参阅 describe-clusters。

查找 MemoryDB 集群的端点（MemoryDB API）

您可以使用 MemoryDB API 来搜索集群的端点。

查找 MemoryDB 集群的端点（MemoryDB API）

您可以使用 MemoryDB API，通过 `DescribeClusters` 操作查找集群的端点。该操作将返回集群的端点。

以下操作检索集群 `mycluster` 的集群端点。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeClusters
    &ClusterName=mycluster
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T192317Z
    &Version=2021-01-01
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 DescribeClusters。

使用分片

分片是 1 到 6 个节点的集合。您可以创建具有更多分片和更少副本的集群，每个集群最多可包含 500 个节点。此集群配置的范围可以从 500 个分片和 0 个副本到 100 个分片和 4 个副本，这是允许的最大副本数。集群的数据分配到该集群的各个分片上。如果分片包含多个节点，则该分片实现了一个节点作为读取/写入主节点且其他节点为只读副本节点的复制。

使用 AWS 管理控制台 创建 MemoryDB 集群时，您可以指定集群中的分片数和分片中的节点数。有关更多信息，请参阅 创建 MemoryDB 集群。

分片中每个节点的计算、存储和内存规格均相同。通过 MemoryDB API 可以控制集群范围的属性，如节点数、安全设置和系统维护时段。

有关更多信息，请参阅MemoryDB 的离线重新分片和MemoryDB 的在线重新分片。

查找分片的名称

您可以使用 AWS 管理控制台、AWS CLI 或 MemoryDB API 查找分片名称。

使用 AWS 管理控制台

以下过程使用 AWS 管理控制台 来查找 MemoryDB 集群的分片名称。

1. 登录到 AWS 管理控制台 并打开 MemoryDB 控制台，网址：https://console.aws.amazon.com/memorydb/。
2. 在左侧导航窗格中，选择集群。
3. 在名称下，选择要查找其分片名称的集群。
4. 在分片和节点选项卡下，在名称下查看分片列表。您还可以展开每项以查看这些节点的详细信息。

使用 AWS CLI

要查找 MemoryDB 集群的分片（分片）名称，请使用带有以下可选参数的 AWS CLI 操作 `describe-clusters`。

- **`--cluster-name`** ——用来将输出限制为指定集群的详细信息的可选参数。如果忽略此参数，将返回最多 100 个集群的详细信息。
- **`--show-shard-details`**——返回分片详细信息，包括分片名称。

此命令将返回 `my-cluster` 的详细信息。

对于 Linux、macOS 或 Unix：

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

对于 Windows：

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

返回以下 JSON 响应：

添加换行符以便于阅读。

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
```

```
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

使用 MemoryDB API

要查找 MemoryDB 集群的分片 ID，请使用带有以下可选参数的 API 操作 `DescribeClusters`。

- **`ClusterName`** ——用来将输出限制为指定集群的详细信息的可选参数。如果忽略此参数，将返回最多 100 个集群的详细信息。
- **`ShowShardDetails`**——返回分片详细信息，包括分片名称。

Example

此命令将返回 `my-cluster` 的详细信息。

对于 Linux、macOS 或 Unix：

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeClusters
   &ClusterName=sample-cluster
   &ShowShardDetails=true
   &Version=2021-01-01
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
```

```
    &X-Amz-Credential=<credential>
```

管理您的 MemoryDB 实施

在此部分中，您可以找到有关如何管理 MemoryDB 实施的各种组件的详细信息。

主题

- 引擎版本
- 开始使用 JSON
- 标记 MemoryDB 资源
- 管理维护
- 最佳实践
- 了解 MemoryDB 复制
- 快照和还原
- 扩展
- 使用参数组配置引擎参数
- 受限命令
- 教程：配置 Lambda 函数，以便在 Amazon VPC 中访问 MemoryDB

引擎版本

本部分介绍支持的 Valkey 和 Redis OSS 引擎版本。

主题

- MemoryDB 版本 7.3
- MemoryDB 版本 7.2.6
- MemoryDB 版本 7.1（加强版）
- MemoryDB 版本 7.0（加强版）
- MemoryDB 和 Redis OSS 版本 6.2（加强版）
- 升级引擎版本

MemoryDB 版本 7.3

2024 年 12 月 1 日，MemoryDB 7.3 发布。MemoryDB 版本 7.3 支持多区域集群，使您能够构建具有高达 99.999% 可用性和极低延迟的多区域应用程序。MemoryDB 多区域当前在以下 AWS 区域受支

持：美国东部（弗吉尼亚州北部和俄亥俄州）、美国西部（俄勒冈州、加利福尼亚州北部）、欧洲地区（爱尔兰、法兰克福和伦敦）以及亚太地区（东京、悉尼、孟买、首尔和新加坡）。有关更多信息，请参阅 MemoryDB 多区域。

MemoryDB 版本 7.2.6

2024 年 10 月 8 日，Valkey 7.2.6 发布。Valkey 7.2.6 与之前版本的 Redis OSS 7.2.5 有相似的兼容性差异。以下是 Valkey 和 Redis OSS 7.0 和 7.1 之间的主要差异：

- ZRANK 和 ZREVRANK 命令有新的 WITHSCORE 选项
- CLIENT NO-TOUCH 让客户端可以在不影响键的 LRU/LFU 的情况下运行命令。
- 新命令 CLUSTER MYSHARDID 返回节点的分片 ID，以便根据复制在集群模式下对节点进行逻辑分组。
- 针对各种数据类型的性能和内存优化。

以下是 Valkey 7.2 和 Redis OSS 7.1（或 7.0）之间可能发生的重大行为变化：

- 使用同样订阅了同一频道的 RESP3 客户端调用 PUBLISH 时，顺序会更改，并在发布消息之前发送回复。
- 脚本的客户端跟踪现在可以跟踪脚本读取的键值，而不是由 EVAL/FCALL 调用方声明的键。
- 冻结时间采样在命令执行期间和在脚本中进行。
- 解除阻止的命令后，会重新评估 ACL、OOM 等检查。
- ACL 故障错误消息文本和错误代码是统一的。
- 当对应的键不再存在时，阻塞的流命令在解除阻塞时会返回不同的错误代码（返回 -NOGROUP 或 -WRONGTYPE，而不是返回 -UNBLOCKED）。
- 仅当实际执行命令时才会更新阻塞命令的命令统计信息。
- ACL 用户的内部存储不再删除多余的命令和类别规则。这可能会改变这些规则在 ACL SAVE、ACL GETUSER 和 ACL LIST 中的显示方式。
- 如果可能，为基于 TLS 的复制创建的任何客户端连接都使用 SNI。
- XINFO STREAM：seen-time 响应字段现在表示上次尝试的交互，而不是最后一次成功的交互。现在，新的 active-time 响应字段表示最后一次成功的交互。
- 无论是否能够执行某些读取/声明，XREADGROUP 和 X[AUTO]CLAIM 都会创建使用者。
- ACL 默认新创建的用户在 ACL LIST/GETUSER 中设置了 sanitize-payload 标志。

- 除非成功执行，否则 HELLO 命令不会影响客户端状态。
- NAN 回复被标准化为单个 nan 类型，类似于 inf 的当前行为。

有关 Valkey 的更多信息，请参阅 Valkey

有关 Valkey 7.2 版本的更多信息，请参阅 GitHub 上有关 Valkey 的 Redis OSS 7.2.4 发布说明（Valkey 7.2 包括从 Redis OSS 到 7.2.4 版本的所有更改）和 Valkey 7.2 发布说明。

MemoryDB 版本 7.1（加强版）

MemoryDB 版本 7.1 增加了对所有区域的向量搜索功能的支持，还增加了关键错误修复和性能增强。

- 向量搜索特征：向量搜索可以与现有的 MemoryDB 功能结合使用。不使用向量搜索的应用程序不会受到其存在的影响。在 MemoryDB 7.1 及更高版本中，所有地区均提供向量搜索功能。如需了解更多信息，请参阅此处的文档。

> **Note**
>
> MemoryDB 7.1 与 Redis OSS 7.0 兼容。有关 Redis OSS 7.0 的更多信息，请参阅 GitHub 上有关 Redis OSS 的 Redis OSS 7.0 发布说明。

MemoryDB 版本 7.0（加强版）

MemoryDB 7.0 增加了多项改进和对新功能的支持：

- Functions：MemoryDB 7 增加了对 Functions 的支持，并提供了托管体验，使开发人员能够使用存储在 MemoryDB 集群上的应用程序逻辑执行 LUA 脚本，而无需客户端在每次连接时都将脚本重新发送到服务器。
- ACL 改进：MemoryDB 7 增加了对下一版本的访问控制列表（ACL）的支持。借助 MemoryDB OSS Valkey 7 或 Redis OSS 7，客户端现在可为特定键或键空间指定多组权限。
- 分片发布/订阅：MemoryDB 7 增加了对在启用集群模式（CME）下运行 MemoryDB 时以分片方式运行发布/订阅功能的支持。发布/订阅功能使发布者能够向频道中任意数量的订阅用户发布消息。借助 Amazon MemoryDB Valkey 7 和 Redis OSS 7，频道可绑定到 MemoryDB 集群中的分片，无需在分片之间传播频道信息。这会提高可扩展性。
- 增强型 I/O 多路复用：MemoryDB Valkey 7 和 Redis OSS 版本 7 引入了增强型 I/O 多路复用，此功能为与 MemoryDB 集群有着许多并发客户端连接的高吞吐量工作负载提供了更高的吞吐量和更短的

延迟。例如，与 MemoryDB 版本 6 相比，当使用由 r6g.4xlarge 节点组成的集群并运行 5200 个并发客户端时，吞吐量（每秒读写操作数）可以提高多达 46%，P99 延迟可减少多达 21%。

有关 Valkey 的更多信息，请参阅 Valkey

有关 Valkey 7.2 版本的更多信息，请参阅 GitHub 上有关 Valkey 的 Redis OSS 7.2.4 发布说明（Valkey 7.2 包括从 Redis OSS 到 7.2.4 版本的所有更改）和 Valkey 7.2 发布说明。

MemoryDB 和 Redis OSS 版本 6.2（加强版）

MemoryDB 推出了新版本的 Redis OSS 引擎，其中包括 使用访问控制列表对用户进行身份验证 () ACLs、自动版本升级支持、客户端缓存和重要的操作改进。

Redis 引擎版本 6.2.6 还引入了对原生 JavaScript 对象表示法（JSON）格式的支持，这是在 Redis OSS 集群中对复杂数据集进行编码的一种简单的无 Schema 方法。借助 JSON 支持，您可以帮助基于 JSON 运行的应用程序利用性能和 Redis OSS API。有关更多信息，请参阅 开始使用 JSON。还包括与 JSON 相关的指标 `JsonBasedCmds`，它被合并到 CloudWatch 中以监控此数据类型的使用情况。有关更多信息，请参阅 MemoryDB 的指标。

从 Redis OSS 6 开始，MemoryDB 将为每个 Redis OSS 次要版本提供单一版本，而不提供多个补丁版本。其旨在最大限度减少不得不从多个次要版本中选择时所产生的混淆和歧义。MemoryDB 还将自动管理处于运行状态下集群的次要版本和补丁版本，确保提高性能和增强安全性。这将通过服务更新活动借助标准客户通知渠道进行处理。有关更多信息，请参阅 MemoryDB 中的服务更新。

如果您在创建过程中未指定引擎版本，则 MemoryDB 将自动为您选择首选 Redis OSS 版本。另一方面，如果您使用 `6.2` 指定引擎版本，则 MemoryDB 将自动调用可用的 Redis OSS 6.2 首选补丁版本。

例如，当您创建集群时，可以将 `--engine-version` 参数设置为 `6.2`。则在创建时，系统将会使用当前可用的首选补丁版本启动集群。任何包含全文引擎版本值的请求都将被拒绝，同时引发异常且进程会失败。

当调用 `DescribeEngineVersions` API 时，`EngineVersion` 参数值将设置为 6.2，实际全文引擎版本会返回在 `EnginePatchVersion` 字段中。

有关 Redis OSS 6.2 版本的更多信息，请参阅 GitHub 上有关 Redis OSS 的 Redis 6.2 发布说明。

升级引擎版本

默认情况下，MemoryDB 自动管理正在运行状态中的集群的补丁版本。此外，如果您将集群的 AutoMinorVersionUpgrade 属性设为 false，则可以选择退出自动次要版本升级。但是，您不能选择退出自动补丁版本更新。

您可以控制为集群提供支持的符合协议标准的软件是否及何时升级到自动升级启动之前的 MemoryDB 所支持的新版本。此级别的控制使您能够与特定版本保持兼容、在生产中部署进行之前使用应用程序测试新版本以及根据自己的条件和时间表执行版本升级。

您也可以从带有 Redis OSS 引擎的现有 MemoryDB 升级到 Valkey 引擎。

您可以通过以下方式对您的集群启动引擎版本升级：

- 通过更新并指定新的引擎版本。有关更多信息，请参阅 修改 MemoryDB 集群。
- 为相应的引擎版本应用服务更新。有关更多信息，请参阅 MemoryDB 中的服务更新。

请注意以下几点：

- 您可以升级到较新的引擎版本，但不能降级到较早的引擎版本。要使用较早的引擎版本，必须删除现有的集群，并使用较早的引擎版本重新创建。
- 我们建议定期升级到最新的主要版本，因为大多数主要改进都不会向后移植到旧版本。随着 MemoryDB 将可用性扩展到新的 AWS 区域，MemoryDB 支持将当时最新的两个 MAJOR.MINOR 版本用于此新区域。例如，如果一个新 AWS 区域推出，并且最新的 MAJOR.MINOR MemoryDB 版本为 7.0 和 6.2，则 MemoryDB 将在新的 AWS 区域中支持版本 7.0 和 6.2。随着 MemoryDB 更新的 MAJOR.MINOR 版本发布，MemoryDB 将继续加大对新发布的 MemoryDB 版本的支持。要详细了解如何为 MemoryDB 选择区域，请参阅 支持的区域和端点。
- 引擎版本管理的设计使您可以尽可能多地控制修补的发生方式。但是，如果发生系统或软件中存在严重安全漏洞这种不太可能发生的情况，MemoryDB 保留代表您修补集群的权利。
- MemoryDB 将为每个 Valkey 或 Redis OSS 次要版本提供单一版本，而不提供多个补丁版本。其支持旨在最大限度减少不得不从多个版本中选择时所产生的混淆和歧义。MemoryDB 还将自动管理处于运行状态下集群的次要版本和补丁版本，确保提高性能和增强安全性。这将通过服务更新活动借助标准客户通知渠道进行处理。有关更多信息，请参阅 MemoryDB 中的服务更新。
- 您可以在最短的停机时间内升级集群版本。集群在整个升级过程中可供读取，并在大部分升级持续时间内可供写入，但在只持续几秒钟的故障转移操作期间则例外。
- 我们建议您在传入的写流量较低期间安排引擎升级。

将处理和修补带多个分片的集群，如下所示：

- 在任何时候，仅在每个分片上执行一次升级操作。
- 在每个分片中，在处理主副本之前，会先处理所有其他副本。如果一个分片中的副本较少，则可能会在处理完其他分片中的副本之前处理该分片中的主副本。
- 在所有分片中，主节点都是按顺序处理的。一次只升级一个主节点。

主题

- 如何升级引擎版本
- 解决被阻止的 Redis OSS 引擎升级

如何升级引擎版本

通过使用 MemoryDB 控制台、AWS CLI 或 MemoryDB API 修改集群并指定较新的引擎版本，启动集群的版本升级。有关更多信息，请参阅以下主题。

- 使用 AWS 管理控制台
- 使用 AWS CLI
- 使用 MemoryDB API

解决被阻止的 Redis OSS 引擎升级

如下表所示，如果您有待处理的纵向扩展操作，则会阻止 Redis OSS 引擎升级操作。

待处理的操作	阻止的操作
纵向扩展	立即引擎升级
引擎升级	立即纵向扩展
纵向扩展和引擎升级	立即纵向扩展
	立即引擎升级

开始使用 JSON

MemoryDB 支持原生 JavaScript 对象表示法 (JSON) 格式，这是一种在 Valkey 或 Redis OSS 集群中对复杂数据集进行编码的简单、无架构的方式。您可以在集群内使用 JavaScript 对象表示法 (JSON) 格式在本地存储和访问数据，并更新存储在这些集群中的 JSON 数据，而无需管理自定义代码来对其进行序列化和反序列化。

除了将 Valkey 或 Redis OSS APIs 用于通过 JSON 运行的应用程序外，您现在还可以高效地检索和更新 JSON 文档的特定部分，而无需操作整个对象，这可以提高性能并降低成本。您还可以使用 Goessner 样式的 `JSONPath` 查询来搜索您的 JSON 文档内容。

使用受支持的引擎版本创建集群后，JSON 数据类型和关联的命令将自动可用。这与版本 2 的 RedisJSON 模块的 API 和 RDB 都兼容，因此您可以轻松地将现有的基于 JSON 的 Valkey 或 Redis OSS 应用程序迁移到 MemoryDB。有关受支持的命令的更多信息，请参阅 支持的 命令。

与 JSON 相关的指标`JsonBasedCmds`已纳入其中 CloudWatch ，以监控此数据类型的使用情况。有关更多信息，请参阅 MemoryDB 的指标。

> **Note**
>
> 要使用 JSON，您必须运行 Valkey 7.2 或更高版本，或运行 Redis OSS 引擎版本 6.2.6 或更高版本。

主题

- JSON 数据类型概述
- 支持的 命令

JSON 数据类型概述

MemoryDB 支持许多用于处理 JSON 数据类型的 Valkey 和 Redis OSS 命令。以下是 JSON 数据类型的概述和支持的命令的详细列表。

术语

租期	说明
JSON 文档	指 JSON 键的值

租期	说明
JSON 值	指 JSON 文档的子集，包括代表整个文档的根。值可以是容器或容器内的条目
JSON 元素	相当于 JSON 值

支持的 JSON 标准

JSON 格式符合 RFC 7159 和 ECMA-404 JSON 数据交换标准。支持 JSON 文本中的 UTF-8 Unicode。

根元素

根元素可以是任何 JSON 数据类型。请注意，在早期的 RFC 4627 中，只允许将对象或数组作为根值。自 RFC 7159 更新以来，JSON 文档的根目录可以是任何 JSON 数据类型。

文档大小限制

JSON 文档以针对快速访问和修改而优化的格式在内部存储。此格式通常会导致比同一文档的等效序列化表示使用稍多的内存。单个 JSON 文档的内存使用限制为 64 MB，这是内存中数据结构的大小，而不是 JSON 字符串的大小。可使用 `JSON.DEBUG MEMORY` 命令检查 JSON 文档所使用的内存量。

json ACLs

- JSON 数据类型完全集成到 Valkey 和 Redis OSS 访问控制列表（ACL）功能中。与现有的每数据类型类别（@string、@hash 等）类似，添加了一个新的类别 @json，以简化对 JSON 命令和数据的访问管理。没有其他现有的 Valkey 或 Redis OSS 命令属于 @json 类别。所有 JSON 命令均强制执行任何键空间或命令限制和权限。
- 有五个现有的 ACL 类别已更新为包含新 JSON 命令：@read、@write、@fast、@slow 和 @admin。下表指示 JSON 命令到相应类别的映射。

ACL

JSON 命令	@read	@write	@fast	@slow	@admin
JSON.ARRAPPEND		y	y		

JSON 命令	@read	@write	@fast	@slow	@admin
JSON.ARRINDEX	y		y		
JSON.ARRINSERT		y	y		
JSON.ARRLEN	y		y		
JSON.ARRPOP		y	y		
JSON.ARRTRIM		y	y		
JSON.CLEAR		y	y		
JSON.DEBUG	y			y	y
JSON.DEL		y	y		
JSON.FORGET		y	y		
JSON.GET	y		y		
JSON.MGET	y		y		
JSON.NUMINCRBY		y	y		
JSON.NUMMULTBY		y	y		
JSON.OBJKEYS	y		y		

JSON 命令	@read	@write	@fast	@slow	@admin
JSON.OBJLEN	y		y		
JSON.RESP	y		y		
JSON.SET		y		y	
JSON.STRAPPEND		y	y		
JSON.STRLEN	y		y		
JSON.STRLEN	y		y		
JSON.TOGGLE		y	y		
JSON.TYPE	y		y		
JSON.NUMINCRBY		y	y		

嵌套深度限制

当 JSON 对象或数组有一个元素本身就是其他 JSON 对象或数组时，该内部对象或数组被称为“嵌套”在外部对象或数组中。最大嵌套深度上限为 128。任何创建包含嵌套深度大于 128 的文档的尝试都将被拒绝，并出现错误。

命令语法

大多数命令均要求将 Valkey 或 Redis OSS 键名称作为第一个参数。某些命令还带有一个路径参数。如果该路径参数是可选的且未提供，则默认为根目录。

表示法：

- 必需的参数括在尖括号内，例如：<key>

- 可选的参数括在方括号内，例如 [path]
- 其他可选参数由…表示，例如 [json…]

路径语法

Valkey 和 Redis OSS 的 JSON 支持两种路径语法：

- 增强语法-遵循 Goessner 描述的 JSONPath 语法，如下表所示。我们对表中的描述进行了重新排序和修改使其更加清楚。
- 受限的语法 – 查询功能有限。

> **Note**
>
> 某些命令的结果对使用哪种类型的路径语法很敏感。

如果查询路径以“$”开头，则使用的是增强的语法。否则使用的是受限的语法。

增强的语法

符号/表达式	说明
$	根元素
. 或 []	子运算符
..	递归下降
*	通配符。对象或数组中的所有元素。
[]	数组下标运算符。索引从 0 开始。
[,]	联合运算符
[start:end:step]	数组 Slice 运算符
?()	将筛选（脚本）表达式应用于当前的数组或对象
()	筛选表达式

符号/表达式	说明
@	用于引用当前正在处理的节点的筛选表达式
==	等于，用于筛选表达式。
!=	不等于，用于筛选表达式。
>	大于，用于筛选表达式。
>=	大于或等于，用于筛选表达式。
<	小于，用于筛选表达式。
<=	小于或等于，用于筛选表达式。
&&	逻辑 AND，用于组合多个筛选表达式。
||	逻辑 OR，用于组合多个筛选表达式。

示例

以下示例基于 Goessner 的示例 XML 数据而构建，我们已通过添加其他字段对数据进行了修改。

```
{ "store": {
    "book": [
      { "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95,
        "in-stock": true,
        "sold": true
      },
      { "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99,
        "in-stock": false,
        "sold": true
      },
      { "category": "fiction",
        "author": "Herman Melville",
```

```
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
        "price": 8.99,
        "in-stock": true,
        "sold": false
      },
      { "category": "fiction",
        "author": "J. R. R. Tolkien",
        "title": "The Lord of the Rings",
        "isbn": "0-395-19395-8",
        "price": 22.99,
        "in-stock": false,
        "sold": false
      }
    ],
    "bicycle": {
      "color": "red",
      "price": 19.95,
      "in-stock": true,
      "sold": false
    }
  }
}
```

路径	说明
$.store.book[*].author	商店中所有书籍的作者
$..author	所有作者
$.store.*	商店的所有会员
$["store"].*	商店的所有会员
$.store..price	商店中所有商品的价格
$..*	JSON 结构的所有递归成员
$..book[*]	所有书籍
$..book[0]	第一本书籍

路径	说明
$..book[-1]	最后一本书籍
$..book[0:2]	前两本书籍
$..book[0,1]	前两本书籍
$..book[0:4]	从索引 0 到 3 的书籍（不包括结尾索引）
$..book[0:4:2]	索引为 0、2 的书籍
$..book[?(@.isbn)]	所有带 isbn 编号的书籍
$..book[?(@.price<10)]	所有价格低于 10 美元的书籍
'$..book[?(@.price < 10)]'	所有价格低于 10 美元的书籍。（如果路径包含空格，则必须为其加引号）
'$..book[?(@["price"] < 10)]'	所有价格低于 10 美元的书籍
'$..book[?(@.["price"] < 10)]'	所有价格低于 10 美元的书籍
$..book[?(@.price>=10&&@.price<=100)]	所有价格在 10 美元到 100 美元之间（含 10 美元和 100 美元）的书籍
'$..book[?(@.price>=10 && @.price<=100)]'	所有价格在 10 美元到 100 美元之间（含 10 美元和 100 美元）的书籍。（如果路径包含空格，则必须为其加引号）
$..book[?(@.sold==true||@.in-stock==false)]	所有已售出或缺货的书籍
'$..book[?(@.sold == true || @.in-stock == false)]'	所有已售出或缺货的书籍。（如果路径包含空格，则必须为其加引号）
'$.store.book[?(@.["category"] == "fiction")]'	所有小说类书籍
'$.store.book[?(@.["category"] != "fiction")]'	所有非小说类书籍

更多筛选表达式示例：

```
127.0.0.1:6379> JSON.SET k1 . '{"books": [{"price":5,"sold":true,"in-
stock":true,"title":"foo"}, {"price":15,"sold":false,"title":"abc"}]}'
OK
127.0.0.1:6379> JSON.GET k1 $.books[?(@.price>1&&@.price<20&&@.in-stock)]
"[{\"price\":5,\"sold\":true,\"in-stock\":true,\"title\":\"foo\"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?(@.price>1 && @.price<20 && @.in-stock)]'
"[{\"price\":5,\"sold\":true,\"in-stock\":true,\"title\":\"foo\"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?((@.price>1 && @.price<20) && (@.sold==false))]'
"[{\"price\":15,\"sold\":false,\"title\":\"abc\"}]"
127.0.0.1:6379> JSON.GET k1 '$.books[?(@.title == "abc")]'
[{"price":15,"sold":false,"title":"abc"}]

127.0.0.1:6379> JSON.SET k2 . '[1,2,3,4,5]'
127.0.0.1:6379> JSON.GET k2 $.*.[?(@>2)]
"[3,4,5]"
127.0.0.1:6379> JSON.GET k2 '$.*.[?(@ > 2)]'
"[3,4,5]"

127.0.0.1:6379> JSON.SET k3 . '[true,false,true,false,null,1,2,3,4]'
OK
127.0.0.1:6379> JSON.GET k3 $.*.[?(@==true)]
"[true,true]"
127.0.0.1:6379> JSON.GET k3 '$.*.[?(@ == true)]'
"[true,true]"
127.0.0.1:6379> JSON.GET k3 $.*.[?(@>1)]
"[2,3,4]"
127.0.0.1:6379> JSON.GET k3 '$.*.[?(@ > 1)]'
"[2,3,4]"
```

受限的语法

符号/表达式	说明
. 或 []	子运算符
[]	数组下标运算符。索引从 0 开始。

示例

路径	说明
.store.book[0].author	第一本书籍的作者
.store.book[-1].author	最后一本书籍的作者
.address.city	城市名称
["store"]["book"][0]["title"]	第一本书籍的书名
["store"]["book"][-1]["title"]	最后一本书籍的书名

> **Note**
>
> 本文档中引用的所有 Goessner 内容均受知识共享许可证的约束。

常见错误前缀

每条错误消息均有一个前缀。以下是常见错误前缀的列表：

Prefix	说明
ERR	一般性错误
LIMIT	超出大小限制错误。例如，超出文档大小限制或嵌套深度限制
NONEXISTENT	键或路径不存在
OUTOFBOUNDARIES	数组索引超出界限
SYNTAXERR	语法错误
WRONGTYPE	错误的值类型

JSON 相关指标

提供了以下 JSON 信息指标：

信息	说明
json_total_memory_bytes	分配给 JSON 对象的总内存
json_num_documents	Valkey 或 Redis OSS 引擎中的文档总数

要查询核心指标，请运行命令：

```
info json_core_metrics
```

MemoryDB 如何与 JSON 交互

以下内容说明了 MemoryDB 如何与 JSON 数据类型交互。

运算符优先顺序

当评估条件表达式以进行筛选时，&& 优先评估，然后评估 ||，这一点在大多数语言中很常见。首先执行括号内的操作。

最大路径嵌套限制行为

MemoryDB 的最大路径嵌套限制为 128。因此，像 `$.a.b.c.d...` 这样的值只能达到 128 个级别。

处理数字值

JSON 没有针对整数和浮点数的单独数据类型。它们都被称为数字。

当接收 JSON 数字时，以两种格式之一存储该数字。如果该数字适合 64 位有符号整数，则将其转换为该格式；否则，将其存储为字符串。尽量保持两个 JSON 数字（例如 JSON.NUMINCRBY 和 JSON.NUMMULTBY）的算术运算精度。如果两个操作数和结果值适合一个 64 位有符号整数，则执行整数运算。否则，输入操作数将转换为 64 位 IEEE 双精度浮点数，执行算术运算，并将结果转换回字符串。

算术命令 `NUMINCRBY` 和 `NUMMULTBY`：

- 如果两个数字都是整数并且结果超出 int64 的范围，则它会自动变成一个双精度浮点数。

- 如果至少有一个数字是浮点，则结果将是双精度浮点数。
- 如果结果超出双精度值的范围，则该命令将返回 OVERFLOW 错误。

> **Note**
>
> 在 Redis OSS 引擎版本 6.2.6.R2 之前，如果在输入时接收 JSON 数字，它会转换为两种内部二进制表示之一：64 位有符号整数或 64 位 IEEE 双精度浮点。不保留原始字符串及其所有格式。因此，当数字作为 JSON 响应的一部分输出时，它会从内部二进制表示转换为使用通用格式规则的可打印字符串。这些规则可能会导致生成的字符串与收到的字符串不同。
>
> - 如果两个数字都是整数并且结果超出 int64 的范围，则它会自动变成一个 64 位 IEEE 双精度浮点数。
> - 如果至少有一个数字是浮点，则结果是 64 位 IEEE 双精度浮点数。
> - 如果结果超出 64 位 IEEE 双精度值的范围，则该命令将返回 OVERFLOW 错误。

有关可用命令的详细列表，请参阅 支持的 命令。

严格语法评估

MemoryDB 不允许使用无效语法的 JSON 路径，即使路径的子集包含有效路径也是如此。这是为了维护我们客户的正确行为。

支持的 命令

支持以下 JSON 命令：

主题

- JSON.ARRAPPEND
- JSON.ARRINDEX
- JSON.ARRINSERT
- JSON.ARRLEN
- JSON.ARRPOP
- JSON.ARRTRIM
- JSON.CLEAR
- JSON.DEBUG

- JSON.DEL
- JSON.FORGET
- JSON.GET
- JSON.MGET
- JSON.NUMINCRBY
- JSON.NUMMULTBY
- JSON.OBJLEN
- JSON.OBJKEYS
- JSON.RESP
- JSON.SET
- JSON.STRAPPEND
- JSON.STRLEN
- JSON.TOGGLE
- JSON.TYPE

JSON.ARRAPPEND

将一个或多个值附加到路径上的数组值。

语法

```
JSON.ARRAPPEND <key> <path> <json> [json ...]
```

- key（必需）– JSON 文档类型的键
- path（必需）– 一个 JSON 路径
- json（必需）– 要附加到数组的 JSON 值

Return

如果路径是增强的语法：

- 表示每个路径处数组的新长度的整数数组。
- 如果值不是数组，则其对应的返回值为 Null。

- 如果输入 json 参数之一不是有效的 JSON 字符串，则为 SYNTAXERR 错误。
- 如果路径不存在，则为 NONEXISTENT 错误。

如果路径是受限的语法：

- 整数，该数组的新长度。
- 如果选择了多个数组值，该命令将返回上次更新数组的新长度。
- 如果路径中的值不是数组，则为 WRONGTYPE 错误。
- 如果输入 json 参数之一不是有效的 JSON 字符串，则为 SYNTAXERR 错误。
- 如果路径不存在，则为 NONEXISTENT 错误。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRAPPEND  k1 $[*] '"c"'
1) (integer) 1
2) (integer) 2
3) (integer) 3
127.0.0.1:6379> JSON.GET k1
"[[\"c\"],[\"a\",\"c\"],[\"a\",\"b\",\"c\"]]"
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRAPPEND  k1 [-1] '"c"'
(integer) 3
127.0.0.1:6379> JSON.GET k1
"[[],[\"a\"],[\"a\",\"b\",\"c\"]]"
```

JSON.ARRINDEX

搜索标量 JSON 值在路径的数组中的首次出现。

- 超出范围错误的处理方法是将索引舍入到数组的开头和结尾。
- 如果 start > end，则返回 -1（未找到）。

语法

```
JSON.ARRINDEX <key> <path> <json-scalar> [start [end]]
```

- key（必需）– JSON 文档类型的键
- path（必需）– 一个 JSON 路径
- json-scalar（必需）– 要搜索的标量值；JSON 标量是指不是对象或数组的值。即字符串、数字、布尔值和 null 都是标量值。
- start（可选）– 起始索引（含）。如果未提供，则默认为 0。
- end（可选）– 结束索引（不含）。如果未提供，则默认为 0，这意味着包含最后一个元素。0 或 -1 意味着包含最后一个元素。

Return

如果路径是增强的语法：

- 整数数组。每个值都是路径内数组中匹配元素的索引。如果未找到，则值为 -1。
- 如果值不是数组，则其对应的返回值为 Null。

如果路径是受限的语法：

- 整数、匹配元素的索引，如果未找到，则为 -1。
- 如果路径中的值不是数组，则为 WRONGTYPE 错误。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRINDEX k1 $[*] '"b"'
1) (integer) -1
```

```
2) (integer) -1
3) (integer) 1
4) (integer) 1
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.ARRINDEX k1 .children '"Tom"'
(integer) 2
```

JSON.ARRINSERT

将一个或多个值插入到索引之前路径处的数组值中。

语法

```
JSON.ARRINSERT <key> <path> <index> <json> [json ...]
```

- key（必需）– JSON 文档类型的键
- path（必需）– 一个 JSON 路径
- index（必需）– 在其前面插入值的数组索引。
- json（必需）– 要附加到数组的 JSON 值

Return

如果路径是增强的语法：

- 表示每个路径处数组的新长度的整数数组。
- 如果值为空数组，则其对应的返回值为 Null。
- 如果值不是数组，则其对应的返回值为 Null。
- 如果索引参数超出界限，则为 OUTOFBOUNDARIES 错误。

如果路径是受限的语法：

- 整数，该数组的新长度。
- 如果路径中的值不是数组，则为 WRONGTYPE 错误。
- 如果索引参数超出界限，则为 OUTOFBOUNDARIES 错误。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRINSERT k1 $[*] 0 '"c"'
1) (integer) 1
2) (integer) 2
3) (integer) 3
127.0.0.1:6379> JSON.GET k1
"[[\"c\"],[\"c\",\"a\"],[\"c\",\"a\",\"b\"]]"
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRINSERT k1 . 0 '"c"'
(integer) 4
127.0.0.1:6379> JSON.GET k1
"[\"c\",[],[\"a\"],[\"a\",\"b\"]]"
```

JSON.ARRLEN

获取路径中数组值的长度。

语法

```
JSON.ARRLEN <key> [path]
```

- key（必需）– JSON 文档类型的键
- path（可选）– 一个 JSON 路径。如果未提供，则默认为根目录

Return

如果路径是增强的语法：

- 表示每个路径处数组长度的整数数组。
- 如果值不是数组，则其对应的返回值为 Null。
- 如果文档键不存在，则为 Null。

如果路径是受限的语法：

- 批量字符串数组。每个元素都是对象中的键名称。
- 整数，数组长度。
- 如果选择了多个对象，该命令将返回第一个数组的长度。
- 如果路径中的值不是数组，则为 WRONGTYPE 错误。
- 如果路径不存在，则为 WRONGTYPE 错误。
- 如果文档键不存在，则为 Null。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '[[], [\"a\"], [\"a\", \"b\"], [\"a\", \"b\", \"c\"]]'
(error) SYNTAXERR Failed to parse JSON string due to syntax error
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRLEN k1 $[*]
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 3

127.0.0.1:6379> JSON.SET k2 . '[[], "a", ["a", "b"], ["a", "b", "c"], 4]'
OK
127.0.0.1:6379> JSON.ARRLEN k2 $[*]
1) (integer) 0
2) (nil)
3) (integer) 2
4) (integer) 3
5) (nil)
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRLEN k1 [*]
(integer) 0
127.0.0.1:6379> JSON.ARRLEN k1 $[3]
1) (integer) 3

127.0.0.1:6379> JSON.SET k2 . '[[], "a", ["a", "b"], ["a", "b", "c"], 4]'
OK
127.0.0.1:6379> JSON.ARRLEN k2 [*]
(integer) 0
127.0.0.1:6379> JSON.ARRLEN k2 $[1]
1) (nil)
127.0.0.1:6379> JSON.ARRLEN k2 $[2]
1) (integer) 2
```

JSON.ARRPOP

从数组中删除并返回索引处的元素。弹出空数组会返回 Null。

语法

```
JSON.ARRPOP <key> [path [index]]
```

- key（必需）– JSON 文档类型的键
- path（可选）– 一个 JSON 路径。如果未提供，则默认为根目录
- index（可选）– 数组中要开始弹出的位置。
  - 如果未提供，则默认为 -1，这表示最后一个元素。
  - 负值表示距离最后一个元素的位置。
  - 超出边界的索引会舍入到其各自的数组边界。

Return

如果路径是增强的语法：

- 表示每个路径上弹出值的批量字符串数组。
- 如果值为空数组，则其对应的返回值为 Null。
- 如果值不是数组，则其对应的返回值为 Null。

如果路径是受限的语法：

- 批量字符串，表示弹出的 JSON 值
- 如果数组为空，则为 Null。
- 如果路径中的值不是数组，则为 WRONGTYPE 错误。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRPOP k1 $[*]
1) (nil)
2) "\"a\""
3) "\"b\""
127.0.0.1:6379> JSON.GET k1
"[[],[],[\"a\"]]"
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRPOP k1
"[\"a\",\"b\"]"
127.0.0.1:6379> JSON.GET k1
"[[],[\"a\"]]"

127.0.0.1:6379> JSON.SET k2 . '[[], ["a"], ["a", "b"]]'
OK
127.0.0.1:6379> JSON.ARRPOP k2 . 0
"[]"
127.0.0.1:6379> JSON.GET k2
"[[\"a\"],[\"a\",\"b\"]]"
```

JSON.ARRTRIM

修剪路径处的数组，以便其成为子数组 [start、end]（两者都包括在内）。

- 如果该数组为空，则不执行任何操作，返回 0。
- 如果 start <0，则将其视为 0。
- 如果 end >= size（数组的大小），则将其视为 size-1。
- 如果 start >= size 或 start > end，则清空数组并返回 0。

语法

```
JSON.ARRINSERT <key> <path> <start> <end>
```

- key（必需）– JSON 文档类型的键
- path（必需）– 一个 JSON 路径
- start（必需）– 起始索引（含）。
- end（必需）– 结束索引（含）。

Return

如果路径是增强的语法：

- 表示每个路径处数组的新长度的整数数组。
- 如果值为空数组，则其对应的返回值为 Null。
- 如果值不是数组，则其对应的返回值为 Null。
- 如果有索引参数超出界限，则为 OUTOFBOUNDARIES 错误。

如果路径是受限的语法：

- 整数，该数组的新长度。
- 如果数组为空，则为 Null。
- 如果路径中的值不是数组，则为 WRONGTYPE 错误。
- 如果有索引参数超出界限，则为 OUTOFBOUNDARIES 错误。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '[[], ["a"], ["a", "b"], ["a", "b", "c"]]'
OK
127.0.0.1:6379> JSON.ARRTRIM k1 $[*] 0 1
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 2
   127.0.0.1:6379> JSON.GET k1
   "[[],[\"a\"],[\"a\",\"b\"],[\"a\",\"b\"]]"
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.ARRTRIM k1 .children 0 1
(integer) 2
127.0.0.1:6379> JSON.GET k1 .children
"[\"John\",\"Jack\"]"
```

JSON.CLEAR

清除路径中的数组或对象。

语法

```
JSON.CLEAR <key> [path]
```

- key（必需）– JSON 文档类型的键
- path（可选）– 一个 JSON 路径。如果未提供，则默认为根目录

Return

- 整数，已清除的容器数量。
- 清除空数组或对象会导致清除 0 个容器。

> **Note**
>
> 在 Redis OSS 版本 6.2.6.R2 之前，清除空数组或对象会导致清除 1 个容器。

- 清除非容器值会返回 0。
- 如果路径中未找到任何数组或对象值，则该命令将返回 0。

示例

```
127.0.0.1:6379> JSON.SET k1 . '[[], [0], [0,1], [0,1,2], 1, true, null, "d"]'
OK
127.0.0.1:6379>  JSON.CLEAR k1  $[*]
(integer) 6
127.0.0.1:6379> JSON.CLEAR k1  $[*]
(integer) 0
127.0.0.1:6379> JSON.SET k2 . '{"children": ["John", "Jack", "Tom", "Bob", "Mike"]}'
OK
127.0.0.1:6379> JSON.CLEAR k2 .children
(integer) 1
127.0.0.1:6379> JSON.GET k2 .children
"[]"
```

JSON.DEBUG

报告信息。支持的子命令有：

- MEMORY <key> [path] – 报告 JSON 值的内存使用量（以字节为单位）。如果未提供，则路径默认为根目录。
- DEPTH <key> [path] – 报告 JSON 文档的最大路径深度。

> **Note**
>
> 此子命令仅适用于使用 Valkey 7.2 或更高版本，或使用 Redis OSS 引擎版本 6.2.6.R2 或更高版本。

- FIELDS <key> [path] – 报告指定文档路径中的字段数。如果未提供，则路径默认为根目录。每个非容器 JSON 值均计为一个字段。对象和数组以递归方式为其包含的每个 JSON 值计数一个字段。除根容器外，每个容器值均计为一个额外字段。

- HELP – 打印命令的帮助消息。

语法

```
JSON.DEBUG <subcommand & arguments>
```

取决于子命令：

MEMORY

- 如果路径是增强的语法：
  - 返回一个整数数组，该数组表示每个路径处 JSON 值的内存大小（以字节为单位）。
  - 如果键不存在，则返回空数组。
- 如果路径是受限的语法：
  - 返回整数、内存大小和 JSON 值（以字节为单位）。
  - 如果键不存在，则返回 Null。

DEPTH

- 返回一个表示 JSON 文档最大路径深度的整数。
- 如果键不存在，则返回 Null。

FIELDS

- 如果路径是增强的语法：
  - 返回一个整数数组，该数组表示每个路径中 JSON 值的字段数。
  - 如果键不存在，则返回空数组。
- 如果路径是受限的语法：
  - 返回一个整数，即 JSON 值的字段数。
  - 如果键不存在，则返回 Null。

HELP – 返回一个帮助消息数组。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '[1, 2.3, "foo", true, null, {}, [], {"a":1, "b":2},
 [1,2,3]]'
OK
127.0.0.1:6379> JSON.DEBUG MEMORY k1 $[*]
1) (integer) 16
2) (integer) 16
3) (integer) 19
4) (integer) 16
5) (integer) 16
6) (integer) 16
7) (integer) 16
8) (integer) 50
9) (integer) 64
127.0.0.1:6379> JSON.DEBUG FIELDS k1 $[*]
1) (integer) 1
2) (integer) 1
3) (integer) 1
4) (integer) 1
5) (integer) 1
6) (integer) 0
7) (integer) 0
8) (integer) 2
9) (integer) 3
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.DEBUG MEMORY k1
(integer) 632
127.0.0.1:6379> JSON.DEBUG MEMORY k1 .phoneNumbers
(integer) 166

127.0.0.1:6379> JSON.DEBUG FIELDS k1
(integer) 19
127.0.0.1:6379> JSON.DEBUG FIELDS k1 .address
(integer) 4
```

```
127.0.0.1:6379> JSON.DEBUG HELP
1) JSON.DEBUG MEMORY <key> [path] - report memory size (bytes) of the JSON element.
 Path defaults to root if not provided.
2) JSON.DEBUG FIELDS <key> [path] - report number of fields in the JSON element. Path
 defaults to root if not provided.
3) JSON.DEBUG HELP - print help message.
```

JSON.DEL

删除文档键中路径处的 JSON 值。如果路径是根目录，则相当于从 Valkey 或 Redis OSS 中删除键。

语法

```
JSON.DEL <key> [path]
```

- key（必需）– JSON 文档类型的键
- path（可选）– 一个 JSON 路径。如果未提供，则默认为根目录

Return

- 已删除元素的数量。
- 如果键不存在，则为 0。
- 如果 JSON 路径无效或不存在，则为 0。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}, "e": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.DEL k1 $.d.*
(integer) 3
127.0.0.1:6379> JSOn.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[1,2,3,4,5]}"
127.0.0.1:6379> JSON.DEL k1 $.e[*]
```

```
(integer) 5
127.0.0.1:6379> JSOn.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[]}"
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}, "e": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.DEL k1 .d.*
(integer) 3
127.0.0.1:6379> JSON.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[1,2,3,4,5]}"
127.0.0.1:6379> JSON.DEL k1 .e[*]
(integer) 5
127.0.0.1:6379> JSON.GET k1
"{\"a\":{},\"b\":{\"a\":1},\"c\":{\"a\":1,\"b\":2},\"d\":{},\"e\":[]}"
```

JSON.FORGET

JSON.DEL 的别名

JSON.GET

返回一个或多个路径的序列化 JSON。

语法

```
JSON.GET <key>
[INDENT indentation-string]
[NEWLINE newline-string]
[SPACE space-string]
[NOESCAPE]
[path ...]
```

- key（必需）– JSON 文档类型的键
- INDENT/NEWLINE/SPACE（可选）— 控制返回的 JSON 字符串的格式，即 “漂亮的打印”。每个字符串的默认值都是空字符串。他们在任意组合中都可被覆盖。可以按任意顺序指定这些字符串。
- NOESCAPE – 可选，允许存在以实现与旧版的兼容性，且没有其他影响。

- path（可选）– 零个或多个 JSON 路径，如果没有给出，则默认为根目录。路径参数必须放在末尾。

Return

增强的路径语法：

如果给出了一条路径：

- 返回值数组的序列化字符串。
- 如果未选择值，则此命令会返回一个空数组。

如果给出了多条路径：

- 返回一个字符串化的 JSON 对象，其中的每个路径都是一个键。
- 如果混合了增强的路径语法和受限的路径语法，则结果符合增强的语法。
- 如果路径不存在，则其相应的值为空数组。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.GET k1 $.address.*
"[\"21 2nd Street\",\"New York\",\"NY\",\"10021-3100\"]"
127.0.0.1:6379> JSON.GET k1 indent "\t" space " " NEWLINE "\n" $.address.*
"[\n\t\"21 2nd Street\",\n\t\"New York\",\n\t\"NY\",\n\t\"10021-3100\"\n]"
127.0.0.1:6379> JSON.GET k1 $.firstName $.lastName $.age
"{\"$.firstName\":[\"John\"],\"$.lastName\":[\"Smith\"],\"$.age\":[27]}"
127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}}'
OK
127.0.0.1:6379> json.get k2 $..*
"[{},{\"a\":1},{\"a\":1,\"b\":2},1,1,2]"
```

受限的路径语法：

```
 127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.GET k1 .address
"{\"street\":\"21 2nd Street\",\"city\":\"New York\",\"state\":\"NY\",\"zipcode\":
\"10021-3100\"}"
127.0.0.1:6379> JSON.GET k1 indent "\t" space " " NEWLINE "\n" .address
"{\n\t\"street\": \"21 2nd Street\",\n\t\"city\": \"New York\",\n\t\"state\": \"NY\",\n
\t\"zipcode\": \"10021-3100\"\n}"
127.0.0.1:6379> JSON.GET k1 .firstName .lastName .age
"{\".firstName\":\"John\",\".lastName\":\"Smith\",\".age\":27}"
```

JSON.MGET

从多个文档密钥 JSONs 在路径上进行序列化。对于不存在的键或 JSON 路径，将返回 null。

语法

```
JSON.MGET <key> [key ...] <path>
```

- key（必需）– 一个或多个文档类型的键。
- path（必需）– 一个 JSON 路径

Return

- 批量字符串数组。数组的大小等于命令中的键数。数组中的每个元素都填充有（a）由路径定位的序列化 JSON 或（b）如果键不存在、路径在文档中不存在或路径无效（语法错误），则填充 Null。
- 如果存在任何指定的键且不是 JSON 键，该命令返回 WRONGTYPE 错误。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"address":{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021"}}'
OK
127.0.0.1:6379> JSON.SET k2 . '{"address":{"street":"5 main
 Street","city":"Boston","state":"MA","zipcode":"02101"}}'
OK
127.0.0.1:6379> JSON.SET k3 . '{"address":{"street":"100 Park
 Ave","city":"Seattle","state":"WA","zipcode":"98102"}}'
OK
127.0.0.1:6379> JSON.MGET k1 k2 k3 $.address.city
1) "[\"New York\"]"
2) "[\"Boston\"]"
3) "[\"Seattle\"]"
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"address":{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021"}}'
OK
127.0.0.1:6379> JSON.SET k2 . '{"address":{"street":"5 main
 Street","city":"Boston","state":"MA","zipcode":"02101"}}'
OK
127.0.0.1:6379> JSON.SET k3 . '{"address":{"street":"100 Park
 Ave","city":"Seattle","state":"WA","zipcode":"98102"}}'
OK

127.0.0.1:6379> JSON.MGET k1 k2 k3 .address.city
1) "\"New York\""
2) "\"Seattle\""
3) "\"Seattle\""
```

JSON.NUMINCRBY

将路径上的数字值增加给定的数字。

语法

```
JSON.NUMINCRBY <key> <path> <number>
```

- key（必需）– JSON 文档类型的键

- path（必需）– 一个 JSON 路径
- number（必填）– 一个数字

Return

如果路径是增强的语法：

- 表示每个路径的结果值的批量字符串数组。
- 如果值不是数字，其对应的返回值为 Null。
- 如果该数字无法解析，则为 WRONGTYPE 错误。
- 如果结果超出 64 位 IEEE 双精度范围，则为 OVERFLOW 错误。
- 如果文档键不存在，则为 NONEXISTENT。

如果路径是受限的语法：

- 表示结果值的批量字符串。
- 如果选择了多个值，该命令将返回上次更新值的结果。
- 如果路径中的值不是数字，则为 WRONGTYPE 错误。
- 如果该数字无法解析，则为 WRONGTYPE 错误。
- 如果结果超出 64 位 IEEE 双精度范围，则为 OVERFLOW 错误。
- 如果文档键不存在，则为 NONEXISTENT。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 $.d[*] 10
"[11,12,13]"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[11,12,13]}"

127.0.0.1:6379> JSON.SET k1 $ '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 $.a[*] 1
"[]"
```

```
127.0.0.1:6379> JSON.NUMINCRBY k1 $.b[*] 1
"[2]"
127.0.0.1:6379> JSON.NUMINCRBY k1 $.c[*] 1
"[2,3]"
127.0.0.1:6379> JSON.NUMINCRBY k1 $.d[*] 1
"[2,3,4]"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[2,3,4]}"

127.0.0.1:6379> JSON.SET k2 $ '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k2 $.a.* 1
"[]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.b.* 1
"[2]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.c.* 1
"[2,3]"
127.0.0.1:6379> JSON.NUMINCRBY k2 $.d.* 1
"[2,3,4]"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":3},\"d\":{\"a\":2,\"b\":3,\"c\":4}}"

127.0.0.1:6379> JSON.SET k3 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k3 $.a.* 1
"[null]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.b.* 1
"[null,2]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.c.* 1
"[null,null]"
127.0.0.1:6379> JSON.NUMINCRBY k3 $.d.* 1
"[2,null,4]"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\",\"b\":2},\"c\":{\"a\":\"a\",\"b\":\"b\"},\"d
\":{\"a\":2,\"b\":\"b\",\"c\":4}}"
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
```

```
127.0.0.1:6379> JSON.NUMINCRBY k1 .d[1] 10
"12"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[1,12,3]}"

127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k1 .a[*] 1
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMINCRBY k1 .b[*] 1
"2"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[1,2],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMINCRBY k1 .c[*] 1
"3"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMINCRBY k1 .d[*] 1
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,3],\"d\":[2,3,4]}"

127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k2 .a.* 1
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMINCRBY k2 .b.* 1
"2"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":1,\"b\":2},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMINCRBY k2 .c.* 1
"3"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":3},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMINCRBY k2 .d.* 1
"4"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":3},\"d\":{\"a\":2,\"b\":3,\"c\":4}}"

127.0.0.1:6379> JSON.SET k3 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k3 .a.* 1
```

```
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMINCRBY k3 .b.* 1
"2"
127.0.0.1:6379> JSON.NUMINCRBY k3 .c.* 1
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMINCRBY k3 .d.* 1
"4"
```

JSON.NUMMULTBY

将路径上的数值乘以给定的数字。

语法

```
JSON.NUMMULTBY <key> <path> <number>
```

- key（必需）– JSON 文档类型的键
- path（必需）– 一个 JSON 路径
- number（必填）– 一个数字

Return

如果路径是增强的语法：

- 表示每个路径的结果值的批量字符串数组。
- 如果值不是数字，其对应的返回值为 Null。
- 如果该数字无法解析，则为 WRONGTYPE 错误。
- 如果结果超出 64 位 IEEE 双精度范围，则为 OVERFLOW 错误。
- 如果文档键不存在，则为 NONEXISTENT。

如果路径是受限的语法：

- 表示结果值的批量字符串。
- 如果选择了多个值，该命令将返回上次更新值的结果。
- 如果路径中的值不是数字，则为 WRONGTYPE 错误。

- 如果该数字无法解析，则为 WRONGTYPE 错误。
- 如果结果超出 64 位 IEEE 双精度范围，则为 OVERFLOW 错误。
- 如果文档键不存在，则为 NONEXISTENT。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 $.d[*] 2
"[2,4,6]"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[2,4,6]}"

127.0.0.1:6379> JSON.SET k1 $ '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 $.a[*] 2
"[]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.b[*] 2
"[2]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.c[*] 2
"[2,4]"
127.0.0.1:6379> JSON.NUMMULTBY k1 $.d[*] 2
"[2,4,6]"

127.0.0.1:6379> JSON.SET k2 $ '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k2 $.a.* 2
"[]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.b.* 2
"[2]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.c.* 2
"[2,4]"
127.0.0.1:6379> JSON.NUMMULTBY k2 $.d.* 2
"[2,4,6]"

127.0.0.1:6379> JSON.SET k3 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k3 $.a.* 2
```

```
"[null]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.b.* 2
"[null,2]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.c.* 2
"[null,null]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.d.* 2
"[2,null,6]"
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 .d[1] 2
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[1],\"c\":[1,2],\"d\":[1,4,3]}"

127.0.0.1:6379> JSON.SET k1 . '{"a":[], "b":[1], "c":[1,2], "d":[1,2,3]}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k1 .a[*] 2
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMMULTBY k1 .b[*] 2
"2"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[1,2],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMMULTBY k1 .c[*] 2
"4"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,4],\"d\":[1,2,3]}"
127.0.0.1:6379> JSON.NUMMULTBY k1 .d[*] 2
"6"
127.0.0.1:6379> JSON.GET k1
"{\"a\":[],\"b\":[2],\"c\":[2,4],\"d\":[2,4,6]}"

127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k2 .a.* 2
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMMULTBY k2 .b.* 2
"2"
127.0.0.1:6379> JSON.GET k2
```

```
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":1,\"b\":2},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k2 .c.* 2
"4"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":4},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k2 .d.* 2
"6"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":4},\"d\":{\"a\":2,\"b\":4,\"c\":6}}"

127.0.0.1:6379> JSON.SET k3 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k3 .a.* 2
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMMULTBY k3 .b.* 2
"2"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\",\"b\":2},\"c\":{\"a\":\"a\",\"b\":\"b\"},\"d
\":{\"a\":1,\"b\":\"b\",\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k3 .c.* 2
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMMULTBY k3 .d.* 2
"6"
127.0.0.1:6379> JSON.GET k3
"{\"a\":{\"a\":\"a\"},\"b\":{\"a\":\"a\",\"b\":2},\"c\":{\"a\":\"a\",\"b\":\"b\"},\"d
\":{\"a\":2,\"b\":\"b\",\"c\":6}}"
```

JSON.OBJLEN

获取路径对象值中的键数。

语法

```
JSON.OBJLEN <key> [path]
```

- key（必需）– JSON 文档类型的键
- path（可选）– 一个 JSON 路径。如果未提供，则默认为根目录

Return

如果路径是增强的语法：

- 表示每个路径的对象长度的整数数组。
- 如果值不是对象，其对应的返回值为 Null。
- 如果文档键不存在，则为 Null。

如果路径是受限的语法：

- 整数，对象中的键数。
- 如果选择了多个对象，该命令将返回第一个对象的长度。
- 如果路径中的值不是对象，则为 WRONGTYPE 错误。
- 如果路径不存在，则为 WRONGTYPE 错误。
- 如果文档键不存在，则为 Null。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJLEN k1 $.a
1) (integer) 0
127.0.0.1:6379> JSON.OBJLEN k1 $.a.*
(empty array)
127.0.0.1:6379> JSON.OBJLEN k1 $.b
1) (integer) 1
127.0.0.1:6379> JSON.OBJLEN k1 $.b.*
1) (nil)
127.0.0.1:6379> JSON.OBJLEN k1 $.c
1) (integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 $.c.*
1) (nil)
2) (nil)
127.0.0.1:6379> JSON.OBJLEN k1 $.d
1) (integer) 3
127.0.0.1:6379> JSON.OBJLEN k1 $.d.*
1) (nil)
2) (nil)
```

```
3) (integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 $.*
1) (integer) 0
2) (integer) 1
3) (integer) 2
4) (integer) 3
5) (nil)
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJLEN k1 .a
(integer) 0
127.0.0.1:6379> JSON.OBJLEN k1 .a.*
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.OBJLEN k1 .b
(integer) 1
127.0.0.1:6379> JSON.OBJLEN k1 .b.*
(error) WRONGTYPE JSON element is not an object
127.0.0.1:6379> JSON.OBJLEN k1 .c
(integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 .c.*
(error) WRONGTYPE JSON element is not an object
127.0.0.1:6379> JSON.OBJLEN k1 .d
(integer) 3
127.0.0.1:6379> JSON.OBJLEN k1 .d.*
(integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 .*
(integer) 0
```

JSON.OBJKEYS

获取路径对象值中的键名。

语法

```
JSON.OBJKEYS <key> [path]
```

- key（必需）– JSON 文档类型的键
- path（可选）– 一个 JSON 路径。如果未提供，则默认为根目录

Return

如果路径是增强的语法：

- 批量字符串数组的数组。每个元素都是匹配对象中的键数组。
- 如果值不是对象，其对应的返回值为空值。
- 如果文档键不存在，则为 Null。

如果路径是受限的语法：

- 批量字符串数组。每个元素都是对象中的键名称。
- 如果选择了多个对象，该命令将返回第一个对象的键。
- 如果路径中的值不是对象，则为 WRONGTYPE 错误。
- 如果路径不存在，则为 WRONGTYPE 错误。
- 如果文档键不存在，则为 Null。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJKEYS k1 $.*
1) (empty array)
2) 1) "a"
3) 1) "a"
   2) "b"
4) 1) "a"
   2) "b"
   3) "c"
5) (empty array)
127.0.0.1:6379> JSON.OBJKEYS k1 $.d
1) 1) "a"
```

```
   2) "b"
   3) "c"
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{}, "b":{"a":"a"}, "c":{"a":"a", "b":"bb"}, "d":
{"a":1, "b":"b", "c":{"a":3,"b":4}}, "e":1}'
OK
127.0.0.1:6379> JSON.OBJKEYS k1 .*
1) "a"
127.0.0.1:6379> JSON.OBJKEYS k1 .d
1) "a"
2) "b"
3) "c"
```

JSON.RESP

返回 Valkey 或 Redis OSS 序列化协议（RESP）中给定路径的 JSON 值。如果值为容器，则响应为 RESP 数组或嵌套数组。

- JSON 空值映射到 RESP Null 批量字符串。
- JSON 布尔值映射到相应的 RESP 简单字符串。
- 整数被映射到 RESP 整数。
- 64 位 IEEE 双浮点数映射到 RESP 批量字符串。
- JSON 字符串被映射到 RESP 批量字符串。
- JSON 数组表示为 RESP 数组，其中第一个元素是简单字符串 [，然后是数组的元素。
- JSON 对象表示为 RESP 数组，其中第一个元素是简单字符串 {，然后是键值对，每个键值对都是 RESP 批量字符串。

语法

```
JSON.RESP <key> [path]
```

- key（必需）– JSON 文档类型的键
- path（可选）– 一个 JSON 路径。如果未提供，则默认为根目录

Return

如果路径是增强的语法：

- 数组的数组。每个数组元素都代表一个路径上值的 RESP 形式。
- 如果文档键不存在，则为空数组。

如果路径是受限的语法：

- 表示路径中值的 RESP 形式的数组。
- 如果文档键不存在，则为 Null。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK

127.0.0.1:6379> JSON.RESP k1 $.address
1) 1) {
   2) 1) "street"
      2) "21 2nd Street"
   3) 1) "city"
      2) "New York"
   4) 1) "state"
      2) "NY"
   5) 1) "zipcode"
      2) "10021-3100"

127.0.0.1:6379> JSON.RESP k1 $.address.*
1) "21 2nd Street"
2) "New York"
3) "NY"
4) "10021-3100"
```

```
127.0.0.1:6379> JSON.RESP k1 $.phoneNumbers
1) 1) [
   2) 1) {
      2) 1) "type"
         2) "home"
      3) 1) "number"
         2) "555 555-1234"
   3) 1) {
      2) 1) "type"
         2) "office"
      3) 1) "number"
         2) "555 555-4567"

127.0.0.1:6379> JSON.RESP k1 $.phoneNumbers[*]
1) 1) {
   2) 1) "type"
      2) "home"
   3) 1) "number"
      2) "212 555-1234"
2) 1) {
   2) 1) "type"
      2) "office"
   3) 1) "number"
      2) "555 555-4567"
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK

127.0.0.1:6379> JSON.RESP k1 .address
1) {
2) 1) "street"
   2) "21 2nd Street"
3) 1) "city"
   2) "New York"
4) 1) "state"
```

```
   2) "NY"
5) 1) "zipcode"
   2) "10021-3100"

127.0.0.1:6379> JSON.RESP k1
 1) {
 2) 1) "firstName"
    2) "John"
 3) 1) "lastName"
    2) "Smith"
 4) 1) "age"
    2) (integer) 27
 5) 1) "weight"
    2) "135.25"
 6) 1) "isAlive"
    2) true
 7) 1) "address"
    2) 1) {
       2) 1) "street"
          2) "21 2nd Street"
       3) 1) "city"
          2) "New York"
       4) 1) "state"
          2) "NY"
       5) 1) "zipcode"
          2) "10021-3100"
 8) 1) "phoneNumbers"
    2) 1) [
       2) 1) {
          2) 1) "type"
             2) "home"
          3) 1) "number"
             2) "212 555-1234"
       3) 1) {
          2) 1) "type"
             2) "office"
          3) 1) "number"
             2) "555 555-4567"
 9) 1) "children"
    2) 1) [
10) 1) "spouse"
    2) (nil)
```

JSON.SET

在路径中设置 JSON 值。

如果路径调用对象成员：

- 如果父元素不存在，该命令将返回 NONEXISTENT 错误。
- 如果父元素存在但不是对象，该命令将返回 ERROR。
- 如果父元素存在并且是对象：
  - 如果成员不存在，当且仅当父对象是路径中的最后一个子对象时，才会将新成员附加到父对象。否则，该命令将返回 NONEXISTENT 错误。
  - 如果成员存在，则其值将替换为 JSON 值。

如果路径调用数组索引：

- 如果父元素不存在，该命令将返回 NONEXISTENT 错误。
- 如果父元素存在但不是数组，该命令将返回 ERROR。
- 如果父元素存在但索引超出界限，该命令返回 OUTOFBOUNDARIES 错误。
- 如果父元素存在且索引有效，该元素将被新的 JSON 值替换。

如果路径调用对象或数组，该值（对象或数组）将被新的 JSON 值替换。

语法

```
JSON.SET <key> <path> <json> [NX | XX]
```

[NX | XX]，其中您可以有 0 或 1 个 [NX | XX] 标识符

- key（必需）– JSON 文档类型的键
- path（必需）– JSON 路径。对于新的键，JSON 路径必须是根目录“.”。
- NX（可选）– 如果路径是根目录，仅在键不存在时设置该值，即插入新文档。如果路径不是根目录，仅在路径不存在时设置该值，即在文档中插入一个值。
- XX（可选）– 如果路径是根目录，仅在键存在时设置该值，即替换现有文档。如果路径不是根目录，则仅在路径存在时设置该值，即更新现有值。

Return

- 成功时使用简单字符串“OK”。
- 如果未满足 NX 或 XX 条件，则为 Null。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{"a":1, "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.SET k1 $.a.* '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"a\":{\"a\":0,\"b\":0,\"c\":0}}"

127.0.0.1:6379> JSON.SET k2 . '{"a": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.SET k2 $.a[*] '0'
OK
127.0.0.1:6379> JSON.GET k2
"{\"a\":[0,0,0,0,0]}"
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"c":{"a":1, "b":2}, "e": [1,2,3,4,5]}'
OK
127.0.0.1:6379> JSON.SET k1 .c.a '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"c\":{\"a\":0,\"b\":2},\"e\":[1,2,3,4,5]}"
127.0.0.1:6379> JSON.SET k1 .e[-1] '0'
OK
127.0.0.1:6379> JSON.GET k1
"{\"c\":{\"a\":0,\"b\":2},\"e\":[1,2,3,4,0]}"
127.0.0.1:6379> JSON.SET k1 .e[5] '0'
(error) OUTOFBOUNDARIES Array index is out of bounds
```

JSON.STRAPPEND

将字符串附加到路径的 JSON 字符串。

语法

```
JSON.STRAPPEND <key> [path] <json_string>
```

- key（必需）– JSON 文档类型的键
- path（可选）– 一个 JSON 路径。如果未提供，则默认为根目录
- json_string（必需）– 字符串的 JSON 表示。请注意，JSON 字符串必须用引号括起来，即'"foo"'。

Return

如果路径是增强的语法：

- 表示每个路径字符串的新长度的整数数组。
- 如果路径上的值不是字符串，其对应的返回值为 Null。
- 如果输入 json 参数不是有效的 JSON 字符串，则为 SYNTAXERR 错误。
- 如果路径不存在，则为 NONEXISTENT 错误。

如果路径是受限的语法：

- 整数，该字符串的新长度。
- 如果选择了多个字符串值，该命令将返回上次更新字符串的新长度。
- 如果路径中的值不是字符串，则为 WRONGTYPE 错误。
- 如果输入 json 参数不是有效的 JSON 字符串，则为 WRONGTYPE 错误。
- 如果路径不存在，则为 NONEXISTENT 错误。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRAPPEND k1 $.a.a '"a"'
1) (integer) 2
127.0.0.1:6379> JSON.STRAPPEND k1 $.a.* '"a"'
1) (integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 $.b.* '"a"'
```

```
1) (integer) 2
2) (nil)
127.0.0.1:6379> JSON.STRAPPEND k1 $.c.* '"a"'
1) (integer) 2
2) (integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 $.c.b '"a"'
1) (integer) 4
127.0.0.1:6379> JSON.STRAPPEND k1 $.d.* '"a"'
1) (nil)
2) (integer) 2
3) (nil)
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRAPPEND k1 .a.a '"a"'
(integer) 2
127.0.0.1:6379> JSON.STRAPPEND k1 .a.* '"a"'
(integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 .b.* '"a"'
(integer) 2
127.0.0.1:6379> JSON.STRAPPEND k1 .c.* '"a"'
(integer) 3
127.0.0.1:6379> JSON.STRAPPEND k1 .c.b '"a"'
(integer) 4
127.0.0.1:6379> JSON.STRAPPEND k1 .d.* '"a"'
(integer) 2
```

JSON.STRLEN

获取路径中 JSON 字符串值的长度。

语法

```
JSON.STRLEN <key> [path]
```

- key（必需）– JSON 文档类型的键
- path（可选）– 一个 JSON 路径。如果未提供，则默认为根目录

Return

如果路径是增强的语法：

- 表示每个路径的字符串值长度的整数数组。
- 如果值不是字符串，其对应的返回值为 Null。
- 如果文档键不存在，则为 Null。

如果路径是受限的语法：

- 整数，该字符串的长度。
- 如果选择了多个字符串值，该命令将返回第一个字符串的长度。
- 如果路径中的值不是字符串，则为 WRONGTYPE 错误。
- 如果路径不存在，则为 NONEXISTENT 错误。
- 如果文档键不存在，则为 Null。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRLEN k1 $.a.a
1) (integer) 1
127.0.0.1:6379> JSON.STRLEN k1 $.a.*
1) (integer) 1
127.0.0.1:6379> JSON.STRLEN k1 $.c.*
1) (integer) 1
2) (integer) 2
127.0.0.1:6379> JSON.STRLEN k1 $.c.b
1) (integer) 2
127.0.0.1:6379> JSON.STRLEN k1 $.d.*
1) (nil)
2) (integer) 1
3) (nil)
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"bb"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.STRLEN k1 .a.a
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .a.*
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .c.*
(integer) 1
127.0.0.1:6379> JSON.STRLEN k1 .c.b
(integer) 2
127.0.0.1:6379> JSON.STRLEN k1 .d.*
(integer) 1
```

JSON.TOGGLE

在路径的 true 和 false 之间切换布尔值。

语法

```
JSON.TOGGLE <key> [path]
```

- key（必需）– JSON 文档类型的键
- path（可选）– 一个 JSON 路径。如果未提供，则默认为根目录

Return

如果路径是增强的语法：

- 表示每个路径的结果布尔值的整数数组（0 – false, 1 – true）。
- 如果值不是布尔值，其对应的返回值为 Null。
- 如果文档键不存在，则为 NONEXISTENT。

如果路径是受限的语法：

- 表示结果布尔值的字符串（"true"/"false")。
- 如果文档键不存在，则为 NONEXISTENT。

- 如果路径中的值不是布尔值，则为 WRONGTYPE 错误。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '{"a":true, "b":false, "c":1, "d":null, "e":"foo", "f":
[], "g":{}}'
OK
127.0.0.1:6379> JSON.TOGGLE k1 $.*
1) (integer) 0
2) (integer) 1
3) (nil)
4) (nil)
5) (nil)
6) (nil)
7) (nil)
127.0.0.1:6379> JSON.TOGGLE k1 $.*
1) (integer) 1
2) (integer) 0
3) (nil)
4) (nil)
5) (nil)
6) (nil)
7) (nil)
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . true
OK
127.0.0.1:6379> JSON.TOGGLE k1
"false"
127.0.0.1:6379> JSON.TOGGLE k1
"true"

127.0.0.1:6379> JSON.SET k2 . '{"isAvailable": false}'
OK
127.0.0.1:6379> JSON.TOGGLE k2 .isAvailable
"true"
127.0.0.1:6379> JSON.TOGGLE k2 .isAvailable
"false"
```

JSON.TYPE

报告给定路径的值类型。

语法

```
JSON.TYPE <key> [path]
```

- key（必需）– JSON 文档类型的键
- path（可选）– 一个 JSON 路径。如果未提供，则默认为根目录

Return

如果路径是增强的语法：

- 表示每个路径的值类型的字符串数组。类型为 {"null"、"boolean"、"string"、"number"、"integer"、"object" 和 "array"} 之一。
- 如果路径不存在，则其相应的返回值为 Null。
- 如果文档键不存在，则为空数组。

如果路径是受限的语法：

- 字符串，值的类型
- 如果文档键不存在，则为 Null。
- 如果 JSON 路径无效或不存在，则为 Null。

示例

增强的路径语法：

```
127.0.0.1:6379> JSON.SET k1 . '[1, 2.3, "foo", true, null, {}, []]'
OK
127.0.0.1:6379> JSON.TYPE k1 $[*]
1) integer
2) number
3) string
```

```
4) boolean
5) null
6) object
7) array
```

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.TYPE k1
object
127.0.0.1:6379> JSON.TYPE k1 .children
array
127.0.0.1:6379> JSON.TYPE k1 .firstName
string
127.0.0.1:6379> JSON.TYPE k1 .age
integer
127.0.0.1:6379> JSON.TYPE k1 .weight
number
127.0.0.1:6379> JSON.TYPE k1 .isAlive
boolean
127.0.0.1:6379> JSON.TYPE k1 .spouse
null
```

标记 MemoryDB 资源

为了帮助您管理集群和其他 MemoryDB 资源，您可以标签的形式为每个资源分配您自己的元数据。标签可让您按各种标准（例如用途、所有者或环境）对 AWS 资源进行分类。这在您具有相同类型的很多资源时会很有用 – 您可以根据分配给特定资源的标签快速识别该资源。本主题介绍标签并说明如何创建标签。

> **Warning**
>
> 作为最佳实践，我们建议您不要在标签中包含敏感数据。

标签基本知识

标签是为AWS资源分配的标记。每个标签都包含定义的一个键 和一个可选值。标签可让您按各种标准（例如用途或拥有者）对 AWS 资源进行分类。例如，您可以为账户中的 MemoryDB 集群定义一组标签，以帮助跟踪每个集群的拥有者和用户组。

我们建议您针对每类资源设计一组标签，以满足您的需要。使用一组连续的标签键，管理资源时会更加轻松。您可以根据添加的标签搜索和筛选资源。有关如何实施有效的资源标记策略的更多信息，请参阅 AWS 白皮书标记最佳实践。

标签对 MemoryDB 没有任何语义意义，应严格按字符串进行解析。同时，标签不会自动分配至您的资源。您可以修改标签的密钥和值，还可以随时删除资源的标签。您可以将标签的值设置为 `null`。如果您添加的标签的值与该实例上现有标签的值相同，新的值就会覆盖旧值。如果删除资源，资源的所有标签也会被删除。

可以使用 AWS 管理控制台、AWS CLI 和 MemoryDB API 处理标签。

如果您使用的是 IAM，则可以控制 AWS 账户中的哪些用户拥有创建、编辑或删除标签的权限。有关更多信息，请参阅 资源级权限。

您可以为之添加标签的资源

您可以标记您的账户中已存在的大多数 MemoryDB 资源。下表列出了支持标记的资源。如果您使用的是 AWS 管理控制台，则可以使用标签编辑器向资源应用标签。在您创建资源时，某些资源屏幕支持为资源指定标签；例如，包含 Name 键和您指定的值的标签。在大多数情况下，控制台会在资源创建后（而不是在资源创建期间）立即应用标签。控制台可能根据名称标签对资源进行组织，但此标签对 MemoryDB 服务没有任何语义意义。

此外，某些资源创建操作让您可以在创建资源时为其指定标签。如果无法在资源创建期间应用标签，系统会回滚资源创建过程。这样可确保要么创建带有标签的资源，要么根本不创建资源，即任何时候都不会创建出未标记的资源。通过在创建时标记资源，您不需要在资源创建后运行自定义标记脚本。

如果您使用的是 Amazon MemoryDB API，AWS CLI 或 AWS 开发工具包，则可以使用相关 MemoryDB API 操作上的 `Tags` 参数来应用标签。它们是：

- `CreateCluster`
- `CopySnapshot`
- `CreateParameterGroup`
- `CreateSubnetGroup`
- `CreateSnapshot`

- `CreateACL`
- `CreateUser`
- `CreateMultiRegionCluster`

下表描述了可以标记的 AWS MemoryDB 资源以及可在创建时使用 MemoryDB API、 CLI 或 AWS 软件开发工具包标记的资源。

MemoryDB 资源标记支持

支持标签	支持在创建时标记
支持	是
支持	是
支持	是
支持	是
是	是
支持	是
支持	是

对于支持在创建时进行标记的 MemoryDB API 操作，您可以在 IAM policies 中应用基于标签的资源级权限，以对可在创建时标记资源的用户和组实施精细控制。资源从创建开始就会受到适当的保护 – 标签会立即应用于资源。因此，控制资源使用的任何基于标签的资源级权限都会立即生效。可以更准确地对您的资源进行跟踪和报告。您可以强制对新资源使用标记，可以控制对资源设置哪些标签键和值。

有关更多信息，请参阅 标记资源示例。

有关标记资源以便于计费的更多信息，请参阅 使用成本分配标签监控成本。

为集群和快照以及多区域集群添加标签

以下规则适用于请求操作中的标记：

- CreateCluster：
  - 如果提供了 `--cluster-name`：

    如果请求中包含标签，则对集群进行标记。
  - 如果提供了 `--snapshot-name`：

    如果请求中包含标签，则仅使用这些标签对集群进行标记。如果请求中未包含任何标签，则将向集群添加快照标签。
- CreateSnapshot：
  - 如果提供了 `--cluster-name`：

    如果请求中包含标签，则仅将请求标签添加到快照。如果请求中未包含任何标签，则集群标签将添加到快照。
  - 自动快照：

    标签将传播自集群标签。
- CopySnapshot：

  如果请求中包含标签，则仅将请求标签添加到快照。如果请求中未包含任何标签，则源快照标签将添加到复制的快照。
- TagResource 和 UntagResource：

  向资源添加/删除标签。

为多区域集群添加标签

MemoryDB 多区域集群是全局资源。因此，可以在任何支持 MemoryDB 多区域的给定区域中，通过调用相关 API 来指定、修改或列出多区域集群上的标签。有关区域支持的更多信息，请参阅 先决条件和限制。

多区域集群上的标签与区域集群上的标签是独立的。您可以在多区域集群及其包含的区域集群上指定不同的标签集。这些标签之间没有层次连接关系，也不会通过这些资源类型之间的层次结构进行复制。

当您通过 TagResource 和 UntagResource API 添加或删除标签时，由于标签对于多区域集群最终是一致的，您可能不会立即在 ListTags API 响应中看到最新的有效标签。

标签限制

下面是适用于标签的基本限制：

- 每个资源的标签数上限 – 50
- 对于每个资源，每个标签键都必须是唯一的，每个标签键只能有一个值。
- 最大键长度 – 128 个 Unicode 字符（采用 UTF-8 格式)。
- 最大值长度 – 256 个 Unicode 字符（采用 UTF-8 格式)。
- 虽然 MemoryDB 允许在其标签中使用任何字符，但其他服务对此具有严格限制。允许在不同的服务中使用的字符包括：可以使用 UTF-8 表示的字母、数字和空格以及以下字符：+ - = . _ : / @
- 标签键和值区分大小写。
- aws: 前缀专门预留供 AWS 使用。如果某个标签具有带有此标签键，则您无法编辑该标签的键或值。具有 aws: 前缀的标签不计入每个资源的标签数限制。

您不能仅依据标签终止或删除资源，而必须指定资源的标识符。例如，要删除您使用名为 DeleteMe 的标签键标记的快照，您必须将 DeleteSnapshot 操作与快照的资源标识符（如 snap-1234567890abcdef0）结合使用。

有关可以标记的 MemoryDB 资源的详细信息，请参阅 您可以为之添加标签的资源。

标记资源示例

- 向集群添加标签。

```
aws memorydb tag-resource \
--resource-arn arn:aws:memorydb:us-east-1:111111222233:cluster/my-cluster \
--tags Key="project",Value="XYZ" Key="memorydb",Value="Service"
```

- 使用标签创建集群。

```
aws memorydb create-cluster \
--cluster-name testing-tags \
```

```
--description cluster-test \
--subnet-group-name test \
--node-type db.r6g.large \
--acl-name open-access \
--tags Key="project",Value="XYZ" Key="memorydb",Value="Service"
```

- 创建具有标签的快照。

  在此情况下，如果您根据请求添加标签，即使集群包含标签，快照也将仅接收请求标签。

```
aws memorydb create-snapshot \
--cluster-name testing-tags \
--snapshot-name bkp-testing-tags-mycluster \
--tags Key="work",Value="foo"
```

使用成本分配标签监控成本

在 MemoryDB 中向资源添加成本分配标签时，可以根据资源标签值对发票上的费用进行分组，从而跟踪您的成本。

MemoryDB 成本分配标签是您定义的键-值对，并与 MemoryDB 资源关联。键和值区分大小写。您可以使用标签键定义类别，而标签值作为该类别中的项目。例如，通过定义标签键 `CostCenter` 和标签值 `10010`，可以表示将资源分配给 10010 成本中心。再如，通过为标签使用 `Environment` 键和 `test` 或 `production` 值，可以将资源指定为测试或生产用途。我们建议您使用一组一致的标签键，从而方便跟踪与资源相关联的成本。

使用成本分配标签整理 AWS 账单，以反映您自己的成本结构。要执行此操作，请注册以获取包含标签键值的 AWS 账户账单。然后，如需查看组合资源的成本，请按有同样标签键值的资源组织您的账单信息。例如，您可以将特定的应用程序名称用作几个资源的标签，然后组织账单信息，以查看在数个服务中的使用该应用程序的总成本。

您也可以合并标签以采用更高详细信息级别跟踪成本。例如，要按区域跟踪服务成本，可以使用标签键 `Service` 和 `Region`。这样，一个资源的值可以有 `MemoryDB` 和 `Asia Pacific (Singapore)` 值，另一个资源可以有 `MemoryDB` 和 `Europe (Frankfurt)` 值。之后，您可以按区域查看 MemoryDB 的总体成本细分。有关更多信息，请参阅《AWS Billing 用户指南》中的使用成本分配标签。

可以向 MemoryDB 集群添加 MemoryDB 成本分配标签。在您添加、列出、修改、复制或删除标签时，操作仅应用到指定的集群。

MemoryDB 成本分配标签的特性

- 成本分配标签应用到在 CLI 和 API 操作中指定为 ARN 的 MemoryDB 资源。资源类型将是“cluster”。

  ARN 格式：`arn:aws:memorydb:<region>:<customer-id>:<resource-type>/<resource-name>`

  示例 ARN：`arn:aws:memorydb:us-east-1:1234567890:cluster/my-cluster`
- 标签键是标签的名称，属于必填内容。键的字符串值的长度可以在 1 到 128 个 Unicode 字符之间，并且不能带有前缀 `aws:`。字符串只能包含一组 Unicode 字母、数字、空格、下划线 (_)、句点 (.)、冒号 (:)、斜杠 (\ )、等号 (=)、加号 (+)、连字符 (-) 或 @ 符号。
- 标签值是标签的可选值。值的字符串值的长度可以在 1 到 256 个 Unicode 字符之间，并且不能带有前缀 `aws:`。字符串只能包含一组 Unicode 字母、数字、空格、下划线 (_)、句点 (.)、冒号 (:)、斜杠 (\ )、等号 (=)、加号 (+)、连字符 (-) 或 @ 符号。
- 一个 MemoryDB 资源最多可以有 50 个标签。
- 在标签集中，值不必具有唯一性。例如，在您的标签集内，键 `Service` 和 `Application` 可同时具有值 `MemoryDB`。

AWS 不会对您的标签应用任何语义意义。标签会严格地作为字符串进行解析。AWS 不会自动在任何 MemoryDB 资源上设置任何标签。

使用 AWS CLI 管理成本分配标签

您可以使用 AWS CLI 添加、修改或删除成本分配标签。

示例 arn：`arn:aws:memorydb:us-east-1:1234567890:cluster/my-cluster`

主题

- 使用 AWS CLI 列出标签
- 使用 AWS CLI 添加标签
- 使用 AWS CLI 修改标签
- 使用 AWS CLI 删除标签

使用 AWS CLI 列出标签

可以使用 AWS CLI，通过 list-tags 操作列出现有 MemoryDB 资源上的标签。

以下代码使用 AWS CLI 列出 us-east-1 区域中的 MemoryDB 集群 `my-cluster` 上的标签。

对于 Linux、macOS 或 Unix：

```
aws memorydb list-tags \
  --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
```

对于 Windows：

```
aws memorydb list-tags ^
  --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
```

此操作的输出类似于下文，即列出资源上的所有标签。

```
{
   "TagList": [
      {
         "Value": "10110",
         "Key": "CostCenter"
      },
      {
         "Value": "EC2",
         "Key": "Service"
      }
   ]
}
```

如果资源上没有任何标签，则输出空标签列表。

```
{
   "TagList": []
}
```

有关更多信息，请参阅适用于 MemoryDB 的 AWS CLI list-tags。

使用 AWS CLI 添加标签

您可以使用 AWS CLI，通过 tag-resource CLI 操作向现有 MemoryDB 资源添加标签。如果资源上不存在标签键，则键和值将添加到资源。如果资源上已存在该键，则与该键关联的值将更新为新值。

下面的代码使用 AWS CLI 向 us-east-1 区域中集群 `my-cluster` 添加键 `Service` 和 `Region`，这两个键的值分别为 `memorydb` 和 `us-east-1`。

对于 Linux、macOS 或 Unix：

```
aws memorydb tag-resource \
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster \
 --tags Key=Service,Value=memorydb \
        Key=Region,Value=us-east-1
```

对于 Windows：

```
aws memorydb tag-resource ^
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster ^
 --tags Key=Service,Value=memorydb ^
        Key=Region,Value=us-east-1
```

此操作的输出将类似于下文，先列出资源上的所有标签，后面跟随操作。

```
{
   "TagList": [
      {
         "Value": "memorydb",
         "Key": "Service"
      },
      {
         "Value": "us-east-1",
         "Key": "Region"
      }
   ]
}
```

有关更多信息，请参阅适用于 MemoryDB 的 AWS CLI tag-resource。

还可以在创建新集群时使用 AWS CLI 向集群添加标签，方法是使用操作 create-cluster。

使用 AWS CLI 修改标签

您可以使用 AWS CLI 修改 MemoryDB 集群上的标签。

修改标签：

- 使用 tag-resource 可添加新标签和值，或更改与现有标签关联的值。
- 使用 untag-resource 移除资源的指定标签。

以上任意操作的输出将是指定集群上标签及其值的列表。

使用 AWS CLI 删除标签

您可以使用 AWS CLI 从现有 MemoryDB 集群中移除标签，方法是使用 untag-resource 操作。

下面的代码使用 AWS CLI 移除了 us-east-1 区域中集群 `my-cluster` 的键 `Service` 和 `Region` 的标签。

对于 Linux、macOS 或 Unix：

```
aws memorydb untag-resource \
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster \
 --tag-keys Region Service
```

对于 Windows：

```
aws memorydb untag-resource ^
 --resource-arn arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster ^
 --tag-keys Region Service
```

此操作的输出将类似于下文，先列出资源上的所有标签，后面跟随操作。

```
{
   "TagList": []
}
```

有关更多信息，请参阅适用于 MemoryDB 的 AWS CLI untag-resource。

使用 MemoryDB API 管理成本分配标签

您可以使用 MemoryDB API 添加、修改或删除成本分配标签。

成本分配标签应用到适用于集群的 MemoryDB。要添加标签的集群是使用 ARN（Amazon 资源名称）指定的。

示例 arn：`arn:aws:memorydb:us-east-1:1234567890:cluster/my-cluster`

主题

- 使用 MemoryDB API 列出标签

- 使用 MemoryDB API 添加标签
- 使用 MemoryDB API 修改标签
- 使用 MemoryDB API 移除标签

使用 MemoryDB API 列出标签

可以使用 MemoryDB API，通过 ListTags 操作列出现有资源上的标签。

以下代码使用 MemoryDB API 列出 us-east-1 区域中的资源 `my-cluster` 上的标签。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=ListTags
   &ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Version=2021-01-01
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

使用 MemoryDB API 添加标签

您可以使用 MemoryDB API，通过 TagResource 操作向现有 MemoryDB 集群添加标签。如果资源上不存在标签键，则键和值将添加到资源。如果资源上已存在该键，则与该键关联的值将更新为新值。

以下代码使用 MemoryDB API 向 us-east-1 区域中的资源 `my-cluster` 添加键 `Service` 和 `Region`，两个键的值分别为 `memorydb` 和 `us-east-1`。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=TagResource
   &ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Tags.member.1.Key=Service
   &Tags.member.1.Value=memorydb
   &Tags.member.2.Key=Region
   &Tags.member.2.Value=us-east-1
   &Version=2021-01-01
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 TagResource。

使用 MemoryDB API 修改标签

您可以使用 MemoryDB API 修改 MemoryDB 集群上的标签。

修改标签的值：

- 使用 TagResource 操作可添加新标签和值，或更改现有标签的值。
- 要从资源中删除标签，请使用 UntagResource。

以上任意操作的输出将是指定资源上标签及其值的列表。

使用 MemoryDB API 移除标签

您可以使用 MemoryDB API 从现有 MemoryDB 集群中移除标签，方法是使用 UntagResource 操作。

下面的代码使用 MemoryDB API 移除了 us-east-1 区域中集群 `my-cluster` 的包含键 `Service` 和 `Region` 的标签。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=UntagResource
   &ResourceArn=arn:aws:memorydb:us-east-1:0123456789:cluster/my-cluster
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &TagKeys.member.1=Service
   &TagKeys.member.2=Region
   &Version=2021-01-01
   &Timestamp=20210802T192317Z
   &X-Amz-Credential=<credential>
```

管理维护

每个集群都有一个每周维护时段，在此期间会应用任何系统更改。如果在创建或修改集群时未指定首选维护时段，则 MemoryDB 随机选择一周中的某一天，在您区域的维护时段内分配 60 分钟的维护时段。

这个 60 分钟维护时段是随机从每个地区的 8 小时时间段中选择出来的。下表列出了每个区域分配默认维护时段的时间段。您可以选择区域的维护时段之外的首选维护时段。

区域代码	区域名称	区域维护时段
ap-northeast-1	亚太（东京）区域	13:00–21:00 UTC
ap-northeast-2	亚太地区（首尔）区域	12:00–20:00 UTC
ap-south-1	亚太地区（孟买）区域	17:30–1:30 UTC
ap-southeast-1	亚太（新加坡）区域	14:00–22:00 UTC
ap-east-1	亚太地区（香港）区域	13:00–21:00 UTC
ap-southeast-2	亚太（悉尼）区域	12:00–20:00 UTC
cn-north-1	中国（北京）区域	14:00–22:00 UTC
cn-northwest-1	中国（宁夏）区域	14:00–22:00 UTC
eu-west-3	欧洲（巴黎）区域	23:59–07:29 UTC
eu-central-1	欧洲地区（法兰克福）区域	23:00–07:00 UTC
eu-west-1	欧洲地区（爱尔兰）区域	22:00–06:00 UTC
eu-west-2	欧洲地区（伦敦）区域	23:00–07:00 UTC
sa-east-1	南美洲（圣保罗）区域	01:00–09:00 UTC
ca-central-1	加拿大（中部）区域	03:00–11:00 UTC
us-east-1	美国东部（弗吉尼亚州北部）区域	03:00–11:00 UTC
us-east-1	美国东部（俄亥俄州）区域	04:00–12:00 UTC
us-west-1	美国西部（北加利福尼亚）区域	06:00–14:00 UTC
us-west-2	美国西部（俄勒冈）区域	06:00–14:00 UTC

更改您的集群的维护时段

维护时段应当选在使用量最小的时段上，因而可能必须不时予以修改。您可以修改您的集群以指定一个持续时间长达 24 小时的时间范围，您已请求的任何维护活动均应在此期间发生。您请求的任何延期或待处理集群修改都将在此期间进行。

更多信息

有关维护时段和节点替换的信息，请参阅：

- 替换节点 – 管理节点替换
- 修改 MemoryDB 集群 – 更改集群的维护时段

最佳实践

您可以在下面找到推荐的 MemoryDB 最佳实践。遵循最佳实践可改进您集群的性能和可靠性。

主题

- MemoryDB 中的故障恢复能力
- 最佳实践：发布/订阅和增强型 I/O 多路复用
- 最佳实践：在线调整集群大小

MemoryDB 中的故障恢复能力

AWS全球基础架构围绕AWS区域和可用区构建。AWS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架构相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅 AWS全球基础设施。

除了 AWS 全球基础设施之外，MemoryDB 还提供多种特征，以帮助支持您的数据故障恢复能力和快照需求。

主题

- 缓解故障

缓解故障

规划 MemoryDB 实施时，您应做好计划以最大程度地减少故障对应用程序和数据产生的影响。本部分中的主题涵盖了可用来防止应用程序和数据出现故障的方法。

缓解故障：MemoryDB 集群

MemoryDB 集群包含一个应用程序可从中读取和写入的主节点和 0 至 5 个只读副本节点。但是，为获得高可用性，我们强烈建议至少使用 1 个副本。在向主节点写入数据时，会在事务日志中保存并在副本节点上异步更新此数据。

在只读副本发生故障的情况下

1. MemoryDB 将检测发生失效副本。
2. MemoryDB 使发生故障的节点脱机。
3. MemoryDB 在同一可用区中启动和预配置替换节点。
4. 新节点与事务日志同步。

在此期间，应用程序可使用其他节点继续读取和写入。

MemoryDB 多可用区

如果在 MemoryDB 集群上激活多可用区，则系统将自动检测并替换失效的主节点。

1. MemoryDB 检测到主节点故障。
2. MemoryDB 失效转移到与失效主节点一致的副本。
3. MemoryDB 在发生故障的主集群的可用区中启动副本。
4. 新节点与事务日志同步。

故障转移到副本节点的速度通常比创建并预置新主节点的速度要快。这意味着，您的应用程序可更快地恢复对主节点的写入。

有关更多信息，请参阅 利用多可用区最大限度地减少 MemoryDB 停机时间。

最佳实践：发布/订阅和增强型 I/O 多路复用

使用 Valkey 或 Redis OSS 版本 7 或更高版本时，我们建议使用分片发布/订阅。您还可以使用增强型 I/O 多路复用来改善吞吐量和延迟，该功能在使用 Valkey 或 Redis OSS 版本 7 或更高版本时自动可用，无需更改客户端。它是发布/订阅工作负载的理想之选，此类工作负载通常受多个客户端连接的吞吐量限制。

最佳实践：在线调整集群大小

重新分片涉及在集群中增加或删除分片或节点以及重新分配密钥空间。因此，多重因素会对重新分片的操作产生影响，如集群的负载、内存使用率和整体数据大小。对于最佳体验，我们建议您遵循整体集群最佳实践进行统一工作负载模式分配。此外，我们建议执行以下步骤。

在启动重新分片前，建议进行以下操作：

- 测试应用程序 – 尽可能在过渡环境中在重新分片期间测试应用程序行为。
- 获取扩展问题的提前通知 – 重新分片是一项需使用大量计算资源的操作。因此，我们建议在重新分片期间，多核心实例的情况下 CPU 保持 80% 以下的利用率，单核心实例的情况下 CPU 保持 50% 以下的利用率。在应用程序开始监测扩展问题前监控 MemoryDB 指标并启动重新分片。跟踪的有用指标为 `CPUUtilization`、`NetworkBytesIn`、`NetworkBytesOut`、`CurrConnections`、`NewConnections` 和 `BytesUsedForMemoryDB`。
- 横向缩减前请确保有足够的空余内存可用 – 如果要进行横向缩减，请确保要保留的分片上的可用空余内存至少是您计划删除的分片上已用内存的 1.5 倍。
- 在非高峰时间启动重新分片 – 此做法有助于减少重新分片操作期间对客户端的延迟和吞吐量的影响。同样有助于更快完成重新分片，因为有更多资源可用于槽重新分配。
- 审核客户端超时行为 – 部分客户端可能会在联机集群调整大小期间出现更高的延迟。为客户端库配置更高的超时会有所帮助，即便服务器处于更高的负载条件下，系统也有时间进行连接。在某些情况下，您可能会打开与服务器的大量连接。在这些情况下，请考虑增加指数回退以便重新连接逻辑。这样做可防止突增的新连接同时连接服务器。

在重新分片期间，建议进行以下操作：

- 避免耗费大量资源的命令 – 避免运行任何计算型和输入/输出密集型操作，例如 `KEYS` 和 `SMEMBERS` 命令。我们推荐此方法是因为这些操作可增加集群上的负载并能对集群的性能产生影响。改用 `SCAN` 和 `SSCAN` 命令。

- 遵循 Lua 最佳实践 – 避免长时间运行 Lua 脚本并始终预先声明在 Lua 脚本中使用的密钥。我们建议使用此方法确定 Lua 脚本未使用跨槽命令。请确保 Lua 脚本中使用的密钥属于同一槽。

重新分片完成后，请注意以下事项：

- 如果目标分片上的内存不足，缩减可能会部分完成。如果发生此结果，必要时请查看可用内存并重新进行操作。
- 带有大型项目的槽不会迁移。特别是带有超过 256 MB 后序列化的槽不会迁移。
- 在重新分片操作期间，Lua 脚本中不支持 FLUSHALL 和 FLUSHDB 命令。

了解 MemoryDB 复制

MemoryDB 通过数据在最多 500 个分片中进行分区来实现复制。

集群中的每个分片都包含一个读/写主节点和最多 5 个只读副本节点。每个主节点每秒可以承受最多 100 Mb。您可以创建具有更多分片和更少副本的集群，每个集群最多可包含 500 个节点。此集群配置的范围可以从 500 个分片和 0 个副本到 100 个分片和 4 个副本，这是允许的最大副本数。

一致性

在 MemoryDB 中，主节点具有强一致性。在返回给客户端之前，成功的写入操作会长期存储在分布式多可用区事务日志中。对主节点执行的读取操作始终返回最新的数据，这些数据反映了来自所有先前成功的写入操作的影响。在主失效转移全程保持强一致性。

在 MemoryDB 中，副本节点具有最终一致性。从副本读取操作（使用 READONLY 命令）并非总能反映来自最近成功的写入操作的影响，其中延迟指标已发布到 CloudWatch 上。但是，从单个副本执行的读取操作的顺序是一致的。成功的写入操作对每个副本节点的生效顺序与在主节点上执行的写入操作的顺序相同。

在集群中的复制

分片中的每个只读副本都保留分片主节点中数据的一份副本。可通过事务日志使用异步复制机制使只读副本与主集群同步。应用程序可以从集群中的任何节点进行读取。应用程序只能对主节点进行写入。只读副本可增强读取可扩展性。由于 MemoryDB 将数据存储在持久事务日志中，因此不存在数据丢失的风险。数据分配在 MemoryDB 集群中的分片上。

应用程序使用 MemoryDB 集群的集群端点来连接该集群中的节点。有关更多信息，请参阅 查找连接端点。

MemoryDB 集群具有区域性，只能包含来自同一个区域的节点。要增强容错能力，您必须在该区域内的多个可用区中预配置主副本和只读副本。

强烈建议所有 MemoryDB 集群都使用提供多可用区的复制。有关更多信息，请参阅 利用多可用区最大限度地减少 MemoryDB 停机时间。

利用多可用区最大限度地减少 MemoryDB 停机时间

有许多情况下，MemoryDB 可能需要替换主节点；这些情况包括特定类型的计划维护以及主节点或可用区出现故障的意外事件。

节点故障的响应取决于哪个节点出现故障。但是，在所有情况下，MemoryDB 都确保在节点更换或失效转移期间不会出现任何数据的丢失。例如，如果副本出现故障，则会替换故障节点，并同步事务日志中的数据。如果主节点出现故障，则会触发失效转移到一致副本，从而确保失效转移期间不会发生任何数据的丢失。现在，写入数据通过新主节点提供。随即替换旧主节点，并从事务日志同步。

如果主节点在单节点分片（没有副本）上失效，则在替换主节点并同步事务日志之前，MemoryDB 将停止接受写入。

节点替换会导致集群出现停机时间，但如果多可用区处于活动状态，则会最大限度缩短停机时间。主节点的角色会自动将故障转移到其中一个副本。MemoryDB 会透明地处理这一点，因此无需创建和预置新的主节点。此故障转移和副本提升可确保您在提升完成后立即继续写入新的主节点。

如果由于维护更新或服务更新而启动了计划的节点替换，请注意，在集群处理传入的写请求时，完成计划的节点替换。

MemoryDB 集群上的多可用区提高容错能力。在集群的主节点出于任何原因变得无法连接或发生故障时，此情况尤其如此。MemoryDB 集群上的多可用区要求每个分片具有多个节点，并且可以自动启用。

主题

- 故障情形及多可用区响应
- 测试自动失效转移

故障情形及多可用区响应

如果多可用区处于活动状态，则失效的主节点将失效转移到可用副本。副本自动与事务日志同步并变为主节点，这比创建并重新预配置新的主节点快得多。提升过程通常只需几秒钟的时间，然后您可以再次对集群进行写入。

在多可用区处于活动状态时，MemoryDB 将持续监控主节点的状态。如果主节点发生故障，则根据故障的类型执行以下操作之一。

主题

- 仅主节点出现故障时的故障情形

- 当主节点和一些副本发生故障时的故障情形
- 整个集群出现故障时的故障情形

仅主节点出现故障时的故障情形

如果只有主节点失效，则副本将自动变为主节点。然后，将在与发生故障的主节点相同的可用区域中创建和预置替换只读副本。

仅当主节点出现故障时，MemoryDB 多可用区才执行以下操作：

1. 发生故障的主节点脱机。
2. up-to-date副本会自动变为主副本。

   一旦失效转移过程完成（通常只需几秒钟的时间），写入操作就会恢复。
3. 启动和预配置替代副本。

   将在可用区（发生故障的主节点的位置）启动替换副本，以便维护节点的分配。
4. 副本与事务日志同步。

有关查找集群的终端节点的信息，请参阅以下主题：

- 查找 MemoryDB 集群的端点（MemoryDB API）

当主节点和一些副本发生故障时的故障情形

如果主群集和至少一个副本出现故障，则 up-to-date副本将升级为主群集。并在与故障节点相同的可用区中创建新副本。

在主节点和一些副本发生故障时，MemoryDB 多可用区执行以下操作：

1. 发生故障的主节点和发生故障的副本脱机。
2. 可用副本将变为主节点。

   一旦失效转移过程完成（通常只需几秒钟的时间），写入操作就会恢复。
3. 创建和预调配替换副本。

   将在可用区（发生故障的节点的位置）创建替换副本，以便维护节点的分配。

4. 所有节点都与事务日志同步。

有关查找集群的终端节点的信息，请参阅以下主题：

- 查找 MemoryDB 集群的端点（AWS CLI）
- 查找 MemoryDB 集群的端点（MemoryDB API）

整个集群出现故障时的故障情形

如果整个集群全部发生故障，则在与原始节点相同的可用区中重新创建所有节点并预配置。

在此场景中，不会发生数据的丢失，原因在于数据保留在事务日志中。

当整个集群发生故障时，MemoryDB 多可用区将执行以下操作：

1. 发生故障的主节点和副本脱机。
2. 已创建并配置替换主节点，并与事务日志同步。
3. 创建和预置替换副本，并与事务日志同步。

   将在可用区（发生故障的节点的位置）创建替换，以便维护节点的分配。

有关查找集群的终端节点的信息，请参阅以下主题：

- 查找 MemoryDB 集群的端点（AWS CLI）
- 查找 MemoryDB 集群的端点（MemoryDB API）

测试自动失效转移

您可以使用 MemoryDB 控制台、 AWS CLI和 MemoryDB API 测试自动失效转移。

在测试时，请注意以下内容：

- 在任意 24 小时期间，此操作最多可以使用五次。
- 如果在不同集群的分片上调用此操作，您可以让调用同时进行。
- 在某些情况下，您可能会在同一 MemoryDB 集群中的不同分片上多次调用此操作。在这种情况下，必须先完成第一个节点替换，然后再进行后续调用。
- 要确定节点替换是否已完成，请使用 MemoryDB 控制台 AWS CLI、或 MemoryDB API 检查事件。查找下列与 `FailoverShard` 相关的事件，此处按事件的可能发生顺序列出：
  1. 集群消息：`FailoverShard API called for shard <shard-id>`
  2. 集群消息：`Failover from primary node <primary-node-id> to replica node <node-id> completed`
  3. 集群消息：`Recovering nodes <node-id>`
  4. 集群消息：`Finished recovery for nodes <node-id>`

  有关更多信息，请参阅下列内容：
  - DescribeEvents在 MemoryDB API 参考中
- 此 API 旨在测试发生 MemoryDB 故障转移时您的应用程序的行为。它不是用于启动故障转移以解决集群问题的操作工具。此外，在某些情况下，例如大规模运营事件， AWS 可能会阻止此 API。

主题

- 使用测试自动故障转移 AWS 管理控制台
- 使用测试自动故障转移 AWS CLI
- 使用 MemoryDB API 测试自动失效转移

使用测试自动故障转移 AWS 管理控制台

使用以下过程测试通过控制台进行自动故障转移。

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 选择要测试的集群左侧的单选按钮。此集群必须至少有一个副本节点。

3. 在 Details 区域中，确认此集群已启用多可用区。如果集群未启用多可用区，则选择其他集群或者修改此集群以启用多可用区。有关更多信息，请参阅 修改 MemoryDB 集群。
4. 选择集群的名称。
5. 在分片和节点页面上，对于要测试失效转移的分片，选择分片的名称。
6. 在节点上，选择失效转移主节点。
7. 选择 Continue 可对主节点进行故障转移，选择 Cancel 可取消操作，不对主节点进行故障转移。

   故障转移过程中，控制台继续将节点状态显示为可用。要跟踪您的故障转移测试进度，请从控制台导航窗格选择 Events。在 Events 选项卡上，观察指示故障转移已开始（`FailoverShard API called`）和已完成（`Recovery completed`）的事件。

使用测试自动故障转移 AWS CLI

您可以使用 failover-shard AWS CLI 操作在任何启用多可用区的集群上测试自动故障转移。

参数

- `--cluster-name` – 必需。要测试的集群。
- `--shard-name` – 必需。要在其上测试自动故障转移的分片的名称。在 24 小时滚动期间您最多可以测试 5 个分片。

以下示例使用调`failover-shard`用 MemoryDB 集群`0001`中的分片。 AWS CLI `my-cluster`

对于 Linux、macOS 或 Unix：

```
aws memorydb failover-shard \
   --cluster-name my-cluster \
   --shard-name 0001
```

对于 Windows：

```
aws memorydb failover-shard ^
   --cluster-name my-cluster ^
   --shard-name 0001
```

要跟踪故障转移的进度，请使用 AWS CLI `describe-events`操作。

返回以下 JSON 响应：

```
{
    "Events": [
        {
            "SourceName": "my-cluster",
            "SourceType": "cluster",
            "Message": "Failover to replica node my-cluster-0001-002 completed",
            "Date": "2021-08-22T12:39:37.568000-07:00"
        },
        {
            "SourceName": "my-cluster",
            "SourceType": "cluster",
            "Message": "Starting failover for shard 0001",
            "Date": "2021-08-22T12:39:10.173000-07:00"
        }
    ]
}
```

有关更多信息，请参阅下列内容：

- 失效转移分片
- describe-events

使用 MemoryDB API 测试自动失效转移

以下示例调用 集群 `memorydb00` 中的分片 `0003` 上的 `FailoverShard`。

Example测试自动失效转移

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=FailoverShard
    &ShardName=0003
    &ClusterName=memorydb00
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T192317Z
    &X-Amz-Credential=<credential>
```

要跟踪失效转移的进度，请使用 MemoryDB `DescribeEvents` API 操作。

有关更多信息，请参阅下列内容：

- FailoverShard
- DescribeEvents

更改副本数量

您可以使用 AWS 管理控制台、AWS CLI 或 MemoryDB API 动态地增加或减少您的 MemoryDB 集群中的只读副本数量。所有分片的副本数量必须相同。

增加集群中的副本数量

您可以将一个 MemoryDB 集群中每个分片的副本数量最多增加到 5 个。可使用 AWS 管理控制台、AWS CLI 或 MemoryDB API 完成此操作。

主题

- 使用 AWS 管理控制台
- 使用 AWS CLI
- 使用 MemoryDB API

使用 AWS 管理控制台

要增加 MemoryDB 集群（控制台）中的副本数量，请参阅 从集群中添加/移除节点。

使用 AWS CLI

要增加 MemoryDB 集群中的副本数量，请使用带有以下参数的 `update-cluster` 命令：

- `--cluster-name` – 必需。确定要在其中增加副本数量的集群。
- `--replica-configuration` – 必需。允许设置副本数量。若要增加副本数量，请将 `ReplicaCount` 属性设置为您希望在此操作结束时在此分片中所具有的副本数量。

Example

以下示例将集群 `my-cluster` 中的副本数量增加到 2 个。

对于 Linux、macOS 或 Unix：

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --replica-configuration \
        ReplicaCount=2
```

对于 Windows：

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --replica-configuration ^
        ReplicaCount=2
```

返回以下 JSON 响应：

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 1,
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

若要在已更新集群的状态从更新变为可用后查看其详细信息，请使用以下命令：

对于 Linux、macOS 或 Unix：

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

对于 Windows：

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

返回以下 JSON 响应：

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-003",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-22T12:59:31.844000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
```

```
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 3
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

有关使用 CLI 增加副本数量的更多信息，请参阅 AWS CLI 命令参考中的 update-cluster。

使用 MemoryDB API

要增加 MemoryDB 分片中的副本数量，请使用带有以下参数的 UpdateCluster 操作：

- ClusterName – 必需。确定要在其中增加副本数量的集群。
- ReplicaConfiguration – 必需。允许设置副本数量。若要增加副本数量，请将 ReplicaCount 属性设置为您希望在此操作结束时在此分片中所具有的副本数量。

Example

以下示例将集群 `sample-cluster` 中的副本数量增加到 3 个。在完成此示例后，每个分片中将有 3 个副本。无论是带单个分片的 MemoryDB 集群还是带多个分片的 MemoryDB 集群，此数字都适用。

```
https://memory-db.us-east-1.amazonaws.com/
      ?Action=UpdateCluster
      &ReplicaConfiguration.ReplicaCount=3
      &ClusterName=sample-cluster
      &Version=2021-01-01
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20210802T192317Z
      &X-Amz-Credential=<credential>
```

有关使用 API 增加副本数量的更多信息，请参阅 UpdateCluster。

减少集群中的副本数量

您可以减少 MemoryDB 集群中的副本数量。您可以将副本数量减少为 0，但如果主节点出现故障，则无法失效转移到副本。

您可以使用 AWS 管理控制台、AWS CLI 或 MemoryDB API 来减少集群中的副本数量。

主题

- 使用 AWS 管理控制台
- 使用 AWS CLI
- 使用 MemoryDB API

使用 AWS 管理控制台

要减少 MemoryDB 集群（控制台）中的副本数量，请参阅 从集群中添加/移除节点。

使用 AWS CLI

要减少 MemoryDB 集群中的副本数量，请使用带有以下参数的 `update-cluster` 命令：

- `--cluster-name` – 必需。确定要在其中减少副本数量的集群。
- `--replica-configuration` – 必需。

  `ReplicaCount` – 设置此属性指定想要的副本节点数量。

Example

以下示例使用 `--replica-configuration` 将集群 `my-cluster` 中的副本数量减少为指定值。

对于 Linux、macOS 或 Unix：

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --replica-configuration \
        ReplicaCount=1
```

对于 Windows：

```
aws memorydb update-cluster ^
```

```
    --cluster-name my-cluster ^
    --replica-configuration ^
        ReplicaCount=1 ^
```

返回以下 JSON 响应：

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 1,
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

若要在已更新集群的状态从更新变为可用后查看其详细信息，请使用以下命令：

对于 Linux、macOS 或 Unix：

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

对于 Windows：

```
aws memorydb describe-clusters ^
```

```
    --cluster-name my-cluster
    --show-shard-details
```

返回以下 JSON 响应：

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 1,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
```

```
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

有关使用 CLI 减少副本数量的更多信息，请参阅 AWS CLI 命令参考中的 update-cluster。

使用 MemoryDB API

要减少 MemoryDB 集群中的副本数量，请使用带有以下参数的 `UpdateCluster` 操作：

- `ClusterName` – 必需。确定要在其中减少副本数量的集群。
- `ReplicaConfiguration` – 必需。允许设置副本数量。

  `ReplicaCount` – 设置此属性指定想要的副本节点数量。

Example

以下示例使用 `ReplicaCount` 将集群 `sample-cluster` 中的副本数量减少为 1 个。在完成此示例后，每个分片中将有一个副本。无论是带单个分片的 MemoryDB 集群还是带多个分片的 MemoryDB 集群，此数字都适用。

```
https://memory-db.us-east-1.amazonaws.com/
```

```
    ?Action=UpdateCluster
    &ReplicaConfiguration.ReplicaCount=1
    &ClusterName=sample-cluster
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T192317Z
    &X-Amz-Credential=<credential>
```

有关使用 API 减少副本数量的更多信息，请参阅 UpdateCluster。

快照和还原

MemoryDB 集群会自动将数据备份到多可用区事务日志，但您可以选择定期或按需创建集群 point-in-time快照。这些快照可用于重新创建之前某个时间点的集群或为全新集群做种。快照包含集群的元数据以及集群中的所有数据。所有快照都会写入 Amazon Simple Storage Service（Amazon S3），该服务提供持久存储。您随时可通过创建新的 MemoryDB 集群并向该集群填充快照中的数据来还原数据。使用 MemoryDB，你可以使用 AWS 管理控制台、 AWS Command Line Interface (AWS CLI) 和 MemoryDB API 来管理快照。

主题

- 快照约束
- 快照费用
- 计划自动快照
- 手动创建快照
- 生成最终快照
- 描述快照
- 复制快照
- 导出快照
- 从快照还原
- 使用外部创建的快照为新集群做种
- 标记快照
- 删除快照

快照约束

在计划或创建快照时考虑以下约束：

- 对于 MemoryDB 集群，为支持的所有节点类型提供快照和还原。
- 在任何连续的 24 小时期间，针对每个集群所创建的手动快照不能超过 20 个。
- MemoryDB 仅支持在集群级别拍摄快照。MemoryDB 不支持在分片或节点级别拍摄快照。
- 在快照过程中，您无法在集群上运行任何其他 API 或 CLI 操作。
- 如果您删除集群并请求最终快照，则 MemoryDB 始终从主节点创建快照。这可确保您在删除集群之前捕获最新的数据。

快照费用

使用 MemoryDB，您可以免费为每个活动 MemoryDB 集群存储一个快照。对于所有 AWS 区域，针对其他快照所用的存储空间按每月 0.085 美元/GB 的费率收费。对于创建快照或者将快照中的数据还原到 MemoryDB 集群，没有数据传输费。

计划自动快照

对于任何 MemoryDB 集群，您可以启用自动快照。当启用自动快照时，MemoryDB 每天为集群创建一个快照。自动备份不会对集群产生任何影响，而且更改是即时发生的。有关更多信息，请参阅 从快照还原。

当您计划自动快照时，应规划以下设置：

- 快照时段 – MemoryDB 每天开始创建快照的时间段。快照时段的最小长度为 60 分钟。您可以将快照时段设置为自己最方便的任何时间，或设置为在一天中使用率特别高的时间段之外执行快照操作。

  如果您未指定快照时段，则 MemoryDB 会自动分配一个。
- 快照保留期限 – 快照在 Amazon S3 中保留的天数。例如，如果您将保留期限设置为 5，则当天进行的快照将保留 5 天。保留期限过期时，会自动删除快照。

  最大快照保留期限为 35 天。如果快照保留期限设置为 0，则会为集群禁用自动快照。MemoryDB 数据仍完全耐用，即使禁用了自动快照。

在创建 MemoryDB 集群时，您可以使用 MemoryDB 控制台、或 MemoryDB API 启用或禁用自动快照。 AWS CLI您可在创建 MemoryDB 集群时启动自动快照，具体方法是选中快照部分中的启动自动快照复选框。有关更多信息，请参阅 创建 MemoryDB 集群。

手动创建快照

除了自动快照以外，您还可以随时创建手动快照。与在指定保留期之后自动删除的自动快照不同，手动快照并没有在超过之后就会自动删除的保留期。您必须手动删除任何手动快照。即使您删除某个集群或节点，该集群或节点的所有手动快照都会保留。如果您不再需要保留某个手动快照，您必须自行显式删除它。

手动快照对于测试和存档非常有用。例如，假设您开发了一组基线数据用于测试。您可以创建这些数据的手动快照并在需要时还原数据。对用于修改数据的应用程序进行测试之后，您可通过创建新集群并从基线快照还原来重置数据。集群准备就绪后，您可以再次针对基线数据测试应用程序并且可根据需要随时重复此过程。

除了直接创建手动快照外，您还可以通过下列方法之一创建手动快照：

- 复制快照 – 源快照是自动还是手动创建并不重要。
- 生成最终快照 – 创建快照，然后立即删除集群。

其他重要性主题

- 快照约束
- 快照费用

您可以使用 AWS 管理控制台、或 MemoryDB API 创建节点的手动快照。 AWS CLI

创建手动快照（控制台）

创建集群的快照（控制台）

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在左侧导航窗格中，选择集群。

   此时会显示 MemoryDB 集群屏幕。
3. 选择要备份的 MemoryDB 集群名称左侧的单选按钮。
4. 选择操作，然后选择拍摄快照。
5. 在快照窗口，在快照名称框中键入快照名称。建议该名称指明所备份的集群以及进行快照的日期和时间。

集群命名约束如下：

- 必须包含 1 – 40 个字母数字字符或连字符。
- 必须以字母开头。
- 不能包含两个连续连字符。
- 不能以连字符结束。

6. 在加密下，选择是使用默认加密密钥还是客户托管密钥。有关更多信息，请参阅 MemoryDB 传输中加密（TLS）。
7. 在 “标签” 下，可以选择添加标签以搜索和筛选快照或跟踪 AWS 成本。
8. 选择拍摄快照。

集群的状态将变为快照。当状态变回可用时，快照已完成。

创建手动快照 (AWS CLI)

要使用创建群集的手动快照 AWS CLI，请使用带有以下参数的`create-snapshot` AWS CLI 操作：

- `--cluster-name` – 用作快照源的 MemoryDB 集群的名称。在备份 MemoryDB 集群时使用此参数。

  集群命名约束如下：

  - 必须包含 1 – 40 个字母数字字符或连字符。
  - 必须以字母开头。
  - 不能包含两个连续连字符。
  - 不能以连字符结束。

- `--snapshot-name` – 要创建的快照的名称。

相关主题

有关更多信息，请参阅 AWS CLI 命令参考 中的 `create-snapshot`。

创建手动快照（MemoryDB API）

要使用 MemoryDB API 创建集群的手动快照，请使用带以下参数的 `CreateSnapshot` MemoryDB API 操作：

- `ClusterName` – 用作快照源的 MemoryDB 集群的名称。在备份 MemoryDB 集群时使用此参数。

  集群命名约束如下：
  - 必须包含 1 – 40 个字母数字字符或连字符。
  - 必须以字母开头。
  - 不能包含两个连续连字符。
  - 不能以连字符结束。
- `SnapshotName` – 要创建的快照的名称。

相关主题

有关更多信息，请参阅 CreateSnapshot。

生成最终快照

您可以使用 MemoryDB 控制台 AWS CLI、或 MemoryDB API 创建最终快照。

创建最终快照（控制台）

使用 MemoryDB 控制台删除 MemoryDB 集群时，您可以创建最终快照。

要在删除 MemoryDB 集群时创建最终快照，请在删除页面上选择是，并在 步骤 5：删除集群 上为快照提供一个名称。

创建最终快照 (AWS CLI)

使用 AWS CLI删除 MemoryDB 集群时，您可以创建最终快照。

删除 MemoryDB 集群时

要在删除集群时创建最终快照，请使用带有以下参数的`delete-cluster` AWS CLI 操作：

- `--cluster-name` – 要删除的集群的名称。
- `--final-snapshot-name` – 最终快照的名称。

以下代码在删除集群 `myCluster` 时拍摄最终快照 `bkup-20210515-final`。

对于 Linux、macOS 或 Unix：

```
aws memorydb delete-cluster \
        --cluster-name myCluster \
        --final-snapshot-name bkup-20210515-final
```

对于 Windows：

```
aws memorydb delete-cluster ^
        --cluster-name myCluster ^
        --final-snapshot-name bkup-20210515-final
```

有关更多信息，请参阅AWS CLI 命令参考中的 delete-cluster。

创建最终快照（MemoryDB API）

使用 MemoryDB API 删除 MemoryDB 集群时，您可以创建最终快照。

删除 MemoryDB 集群时

要创建最终快照，请使用带以下参数的 `DeleteCluster` MemoryDB API 操作。

- `ClusterName` – 要删除的集群的名称。
- `FinalSnapshotName` – 快照的名称。

以下 MemoryDB API 操作在`bkup-20210515-final`删除集群时`myCluster`创建快照。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DeleteCluster
    &ClusterName=myCluster
    &FinalSnapshotName=bkup-20210515-final
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210515T192317Z
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 DeleteCluster。

描述快照

以下过程演示如何显示快照列表。如果需要，您还可以查看特定快照的详细信息。

描述快照（控制台）

要显示快照，请使用 AWS 管理控制台

1. 登录控制台
2. 在左侧导航窗格中，选择快照。
3. 使用搜索筛选手动、自动或所有快照。
4. 要查看特定快照的详细信息，请选中快照名称左侧的单选按钮。选择操作，然后选择查看详细信息。
5. 或者，在查看详细信息页面中执行其他快照操作，例如复制、恢复或删除。您还可以为快照添加标签

描述快照 (AWS CLI)

要显示快照列表及特定快照的可选详情，请使用 `describe-snapshots` CLI 操作。

示例

以下操作使用参数 `--max-results` 列出与您的账户关联的最多 20 个快照。忽略参数 `--max-results` 最多可列出 50 个快照。

```
aws memorydb describe-snapshots --max-results 20
```

以下操作使用参数 `--cluster-name` 仅列出与集群 `my-cluster` 关联的快照。

```
aws memorydb describe-snapshots --cluster-name my-cluster
```

以下操作使用参数 `--snapshot-name` 显示快照 `my-snapshot` 的详细信息。

```
aws memorydb describe-snapshots --snapshot-name my-snapshot
```

有关更多信息，请参阅 describe-snapshots。

描述快照（MemoryDB API）

要显示快照列表，请使用 `DescribeSnapshots` 操作。

示例

以下操作使用参数 MaxResults 列出与您的账户关联的最多 20 个快照。忽略参数 MaxResults 最多可列出 50 个快照。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeSnapshots
    &MaxResults=20
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

以下操作使用参数 ClusterName 列出与集群 MyCluster 关联的所有快照。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeSnapshots
    &ClusterName=MyCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

以下操作使用参数 SnapshotName 显示快照 MyBackup 的详细信息。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeSnapshots
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &SnapshotName=MyBackup
```

```
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

有关更多信息，请参阅 DescribeSnapshots。

复制快照

您可以复制任何快照，无论它是自动还是手动创建的。当复制快照时，除非明确覆盖，否则目标将使用与源相同的 KMS 加密密钥。您还可以导出自己的快照，以便从 MemoryDB 外部访问它。有关导出快照的指南，请参阅 导出快照。

以下过程演示如何复制快照。

复制快照（控制台）

复制快照（控制台）

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 要查看您的快照列表，请从左侧导航窗格中，选择快照。
3. 在快照列表中，选择要复制的快照的名称左侧的单选按钮。
4. 选择操作，然后选择复制。
5. 在复制快照页面中，执行以下操作：
    a. 在新快照名称框中，输入新快照的名称。
    b. 将可选 Target S3 Bucket 框留空。该字段只能用于导出快照，它需要特殊的 S3 权限。有关导出快照的信息，请参阅 导出快照。
    c. 选择是使用默认 AWS KMS 加密密钥还是使用自定义密钥。有关更多信息，请参阅 MemoryDB 传输中加密（TLS）。
    d. 或者，为快照副本添加标签。
    e. 选择复制。

复制快照 (AWS CLI)

要复制快照，请使用 `copy-snapshot` 操作。

参数

- `--source-snapshot-name` – 要复制的快照的名称。
- `--target-snapshot-name` – 快照副本的名称。
- `--target-bucket` – 为导出快照预留。在复制快照时，请勿使用此参数。有关更多信息，请参阅 导出快照。

以下示例复制自动快照。

对于 Linux、macOS 或 Unix：

```
aws memorydb copy-snapshot \
    --source-snapshot-name automatic.my-primary-2021-03-27-03-15 \
    --target-snapshot-name my-snapshot-copy
```

对于 Windows：

```
aws memorydb copy-snapshot ^
    --source-snapshot-name automatic.my-primary-2021-03-27-03-15 ^
    --target-snapshot-name my-snapshot-copy
```

有关更多信息，请参阅 copy-snapshot。

复制快照（MemoryDB API）

要复制快照，请使用带有以下参数的 copy-snapshot 操作：

参数

- SourceSnapshotName – 要复制的快照的名称。
- TargetSnapshotName – 快照副本的名称。
- TargetBucket – 为导出快照预留。在复制快照时，请勿使用此参数。有关更多信息，请参阅 导出快照。

以下示例复制自动快照。

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=CopySnapshot
    &SourceSnapshotName=automatic.my-primary-2021-03-27-03-15
    &TargetSnapshotName=my-snapshot-copy
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
```

```
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

有关更多信息，请参阅 CopySnapshot。

导出快照

MemoryDB 支持将 MemoryDB 快照导出到 Amazon Simple Storage Service（Amazon S3）存储桶，这让您可以从 MemoryDB 之外访问它。导出的 MemoryDB 快照完全符合 Valkey 和开源 Redis OSS，并且可以使用适当的版本或工具进行加载。您可以使用 MemoryDB 控制台 AWS CLI、或 MemoryDB API 导出快照。

如果您需要在其他 AWS 区域启动集群，则导出快照会很有帮助。您可以将数据导出到一个 AWS 区域，将.rdb 文件复制到新 AWS 区域，然后使用该.rdb 文件为新集群做种子，而不必等待新集群通过使用进行填充。有关为新集群做种的信息，请参阅 使用外部创建的快照为新集群做种。您可能希望导出集群数据的另一个原因是将 .rdb 文件用于脱机处理。

Important

- MemoryDB 快照和您要将其复制到的 Amazon S3 存储桶必须位于同一 AWS 区域。

  尽管复制到 Amazon S3 存储桶的快照已加密，但我们强烈建议您不要将要存储快照的 Amazon S3 存储桶的访问权限授予他人。
- 使用数据分层功能的集群不支持将快照导出到 Amazon S3。有关更多信息，请参阅 数据分层。

在将快照导出到 Amazon S3 存储桶之前，您必须将 Amazon S3 存储桶与快照位于同一 AWS 区域。向 MemoryDB 授予对存储桶的访问权限。前两个步骤向您演示了如何执行此操作。

Warning

以下方案会以您可能不希望的方式公开您的数据：

- 其他人具有您将快照导出到其中的 Amazon S3 存储桶的访问权限时。

  要控制对快照的访问权限，请将对 Amazon S3 存储桶的访问权限仅授予您允许访问数据的人员。有关管理对 Amazon S3 存储桶的访问权限的信息，请参阅 Amazon S3 开发人员指南中的管理访问权限。
- 当其他人有权使用 CopySnapshot API 操作时。

  有权限使用 `CopySnapshot` API 操作的用户或组可以创建自己的 Amazon S3 存储桶并将快照复制到其中。要控制对快照的访问权限，请使用 AWS Identity and Access Management

(IAM) 策略来控制谁有权使用 CopySnapshot API。有关使用 IAM 控制 MemoryDB API 操作使用的更多信息，请参阅 MemoryDB 用户指南中的 MemoryDB 中的身份和访问管理。

主题

- 步骤 1：创建 Amazon S3 存储桶
- 步骤 2：授予 MemoryDB 对 Amazon S3 存储桶的访问权限
- 步骤 3：导出 MemoryDB 快照

步骤 1：创建 Amazon S3 存储桶

以下过程使用 Amazon S3 控制台创建您可以在其中导出和存储 MemoryDB 快照的 Amazon S3 存储桶。

创建 Amazon S3 存储桶

1. 登录 AWS 管理控制台 并打开 Amazon S3 控制台，网址为https://console.aws.amazon.com/s3/。
2. 选择创建存储桶。
3. 在 Create a Bucket - Select a Bucket Name and Region 中，执行以下操作：
    a. 在 Bucket Name（存储桶名称）中键入 Amazon S3 存储桶的名称。
    b. 从区域列表中，为您的 Amazon S3 存储桶选择一个 AWS 区域。此 AWS 区域必须与您要导出的 MemoryDB 快照位于同一 AWS 区域。
    c. 选择创建。

有关创建 Amazon S3 存储桶的更多信息，请参阅 Amazon Simple Storage Service 用户指南中的创建存储桶。

步骤 2：授予 MemoryDB 对 Amazon S3 存储桶的访问权限

AWS 2019 年 3 月 20 日之前推出的区域默认处于启用状态。您可以立即开始在这些 AWS 地区工作。2019 年 3 月 20 日之后推出的区域默认情况下处于禁用状态。您必须按照管理 AWS 区域所述，先启用或选择加入这些区域，然后才能使用它们。

授予 MemoryDB 访问某个区域内您的 S3 存储桶的权限 AWS

要在某个 AWS 区域的 Amazon S3 存储桶上创建适当的权限，请执行以下步骤。

向 MemoryDB 授予对 S3 存储桶的访问权限

1. 登录 AWS 管理控制台 并打开 Amazon S3 控制台，网址为https://console.aws.amazon.com/s3/。
2. 选择要将快照复制到其中的 Amazon S3 存储桶的名称。这应该是您在步骤 1：创建 Amazon S3 存储桶中创建的 S3 存储桶。
3. 选择权限选项卡，在权限下面，选择存储桶策略。
4. 更新策略以授予 MemoryDB 执行操作所需的权限：

   - 将 `[ "Service" : "region-full-name.memorydb-snapshot.amazonaws.com" ]` 添加到 `Principal`。
   - 添加将快照导出到 Amazon S3 存储桶所需的以下权限。
     - `"s3:PutObject"`
     - `"s3:GetObject"`
     - `"s3:ListBucket"`
     - `"s3:GetBucketAcl"`
     - `"s3:ListMultipartUploadParts"`
     - `"s3:ListBucketMultipartUploads"`

   以下是更新策略具体形式的示例。

   JSON

```
{
    "Version":"2012-10-17",
    "Id": "Policy15397346",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "aws-region.memorydb-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:PutObject",
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl",
```

```
                "s3:ListMultipartUploadParts",
                "s3:ListBucketMultipartUploads"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

步骤 3：导出 MemoryDB 快照

现在您已经创建了 S3 存储桶并向 MemoryDB 授予了访问它的权限。将 S3 对象所有权更改为ACLs 已启用-首选存储桶所有者。接下来，你可以使用 MemoryDB 控制台、CL AWS I 或 MemoryDB API 将快照导出到控制台。下面假设您拥有以下附加的 S3 特定 IAM 权限。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "s3:GetBucketLocation",
   "s3:ListAllMyBuckets",
   "s3:PutObject",
   "s3:GetObject",
   "s3:DeleteObject",
   "s3:ListBucket"
  ],
  "Resource": "arn:aws:s3:::*"
 }]
}
```

导出 MemoryDB 快照（控制台）

以下步骤使用 MemoryDB 控制台将备份导出到 Amazon S3 存储桶，以便从 MemoryDB 外部访问它。Amazon S3 存储桶必须与 MemoryDB 快照位于同一 AWS 区域。

将 MemoryDB 快照导出到 Amazon S3 桶

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 要查看您的快照列表，请从左侧导航窗格中，选择快照。
3. 在快照列表中，选择要导出的快照名称左侧的单选按钮。
4. 选择复制。
5. 在 Create a Copy of the Backup?（创建备份副本?）中，执行以下操作：

   a. 在新快照名称框中，输入新快照的名称。

      名称必须在 1 到 1000 个字符之间，并能够以 UTF-8 编码。

      MemoryDB 对您在此处输入的值添加分片标识符和 `.rdb`。例如，如果您输入 `my-exported-snapshot`，则 MemoryDB 创建 `my-exported-snapshot-0001.rdb`。

   b. 从目标 S3 位置列表中，选择要将快照复制到其中的 Amazon S3 存储桶（您在 步骤 1：创建 Amazon S3 存储桶 中创建的存储桶）的名称。

      目标 S3 位置必须是快照 AWS 区域中具有以下权限的 Amazon S3 存储桶，导出过程才能成功。

      - 对象访问 – Read（读取）和 Write（写入）。
      - 权限访问 – Read（读取）。

      有关更多信息，请参阅 步骤 2：授予 MemoryDB 对 Amazon S3 存储桶的访问权限。

   c. 选择复制。

> **Note**
>
> 如果您的 S3 存储桶没有供 MemoryDB 将快照导出到其中所需的权限，则您将收到以下某个错误消息。返回到 步骤 2：授予 MemoryDB 对 Amazon S3 存储桶的访问权限，添加指定权限并重试导出快照的操作。
>
> - 未授予 MemoryDB 在 S3 存储桶上的 READ 权限 %s。
>
>   解决方案：在存储桶上添加 Read 权限。
> - 未授予 MemoryDB 在 S3 存储桶上的 WRITE 权限 %s。

解决方案：在存储桶上添加 Write 权限。

- 未授予 MemoryDB 在 S3 存储桶上的 READ_ACP 权限 %s。

  解决方案：为存储桶的权限访问添加 Read。

如果您想将快照复制到其他 AWS 区域，请使用 Amazon S3 将其复制。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的复制对象。

导出 MemoryDB 快照 (CLI AWS)

使用带有以下参数的 `copy-snapshot` CLI 操作将快照导出到 Amazon S3 存储桶：

参数

- `--source-snapshot-name` – 要复制的快照的名称。
- `--target-snapshot-name` – 快照副本的名称。

  名称必须在 1 到 1000 个字符之间，并能够以 UTF-8 编码。

  MemoryDB 对您在此处输入的值添加分片标识符和 `.rdb`。例如，如果您输入 `my-exported-snapshot`，则 MemoryDB 创建 `my-exported-snapshot-0001.rdb`。
- `--target-bucket` – 您要将快照导出到其中的 Amazon S3 存储桶的名称。在指定存储桶中生成快照的副本。

  `--target-bucket`必须是快照 AWS 所在区域中具有以下权限的 Amazon S3 存储桶，导出过程才能成功。
  - 对象访问 – Read（读取）和 Write（写入）。
  - 权限访问 – Read（读取）。

  有关更多信息，请参阅 步骤 2：授予 MemoryDB 对 Amazon S3 存储桶的访问权限。

以下操作将快照复制到 amzn-s3-demo-bucket。

对于 Linux、macOS 或 Unix：

```
aws memorydb copy-snapshot \
    --source-snapshot-name automatic.my-primary-2021-06-27-03-15 \
```

```
    --target-snapshot-name my-exported-snapshot \
    --target-bucket amzn-s3-demo-bucket
```

对于 Windows：

```
aws memorydb copy-snapshot ^
    --source-snapshot-name automatic.my-primary-2021-06-27-03-15 ^
    --target-snapshot-name my-exported-snapshot ^
    --target-bucket amzn-s3-demo-bucket
```

> **Note**
>
> 如果您的 S3 存储桶没有供 MemoryDB 将快照导出到其中所需的权限，则您将收到以下某个错误消息。返回到 步骤 2：授予 MemoryDB 对 Amazon S3 存储桶的访问权限，添加指定权限并重试导出快照的操作。
>
> - 未授予 MemoryDB 在 S3 存储桶上的 READ 权限 %s。
>
>   解决方案：在存储桶上添加 Read 权限。
> - 未授予 MemoryDB 在 S3 存储桶上的 WRITE 权限 %s。
>
>   解决方案：在存储桶上添加 Write 权限。
> - 未授予 MemoryDB 在 S3 存储桶上的 READ_ACP 权限 %s。
>
>   解决方案：为存储桶的权限访问添加 Read。

有关更多信息，请参阅《AWS CLI 命令参考》中的 `copy-snapshot`。

如果您想将快照复制到其他 AWS 区域，请使用 Amazon S3 副本。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的复制对象。

导出 MemoryDB 快照（MemoryDB API）

使用带有以下参数的 `CopySnapshot` API 操作将快照导出到 Amazon S3 存储桶。

参数

- `SourceSnapshotName` – 要复制的快照的名称。
- `TargetSnapshotName` – 快照副本的名称。

名称必须在 1 到 1000 个字符之间，并能够以 UTF-8 编码。

MemoryDB 对您在此处输入的值添加分片标识符和 .rdb。例如，如果您输入 my-exported-snapshot，则将获得 my-exported-snapshot-0001.rdb。

- TargetBucket – 您要将快照导出到其中的 Amazon S3 存储桶的名称。在指定存储桶中生成快照的副本。

  TargetBucket必须是快照 AWS 所在区域中具有以下权限的 Amazon S3 存储桶，导出过程才能成功。

  - 对象访问 – Read（读取）和 Write（写入）。
  - 权限访问 – Read（读取）。

  有关更多信息，请参阅 步骤 2：授予 MemoryDB 对 Amazon S3 存储桶的访问权限。

以下示例演示将自动快照复制到 Amazon S3 存储桶 amzn-s3-demo-bucket。

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=CopySnapshot
    &SourceSnapshotName=automatic.my-primary-2021-06-27-03-15
    &TargetBucket=&example-s3-bucket;
    &TargetSnapshotName=my-snapshot-copy
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

> **Note**
>
> 如果您的 S3 存储桶没有供 MemoryDB 将快照导出到其中所需的权限，则您将收到以下某个错误消息。返回到 步骤 2：授予 MemoryDB 对 Amazon S3 存储桶的访问权限，添加指定权限并重试导出快照的操作。

- 未授予 MemoryDB 在 S3 存储桶上的 READ 权限 %s。

  解决方案：在存储桶上添加 Read 权限。
- 未授予 MemoryDB 在 S3 存储桶上的 WRITE 权限 %s。

  解决方案：在存储桶上添加 Write 权限。
- 未授予 MemoryDB 在 S3 存储桶上的 READ_ACP 权限 %s。

  解决方案：为存储桶的权限访问添加 Read。

有关更多信息，请参阅 CopySnapshot。

如果您想将快照复制到其他 AWS 区域，请使用 Amazon S3 副本将导出的快照复制到另一个 AWS 区域的 Amazon S3 存储桶。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的复制对象。

从快照还原

您可以随时将数据从 MemoryDB 或 ElastiCache (Redis OSS) .rdb 快照文件恢复到新集群。

MemoryDB 还原流程支持以下操作：

- 从您从 ElastiCache (Redis OSS) 创建的一个或多个.rdb 快照文件迁移到 MemoryDB 集群。

  .rdb 文件必须放在 S3 中来执行还原。
- 在新集群中指定多个分片，其数量不同于创建快照文件时所用集群中分片的数量。
- 为新集群指定不同节点类型 – 较大或更小的节点类型。如果要缩减到较小的节点类型，则必须确保新节点类型拥有足量内存以适应您的数据和引擎开销。
- 以不同于创建快照文件时所用集群中的方法，配置新 MemoryDB 集群的槽。

> **Important**
>
> - MemoryDB 集群不支持多个数据库。因此，还原到 MemoryDB 时，如果 .rdb 文件引用多个数据库，还原将会失败。
> - 您不能将使用数据分层功能的集群（例如，r6gd 节点类型的集群）快照还原到不使用数据分层功能的集群（例如，r6g 节点类型的集群）。

从快照还原集群时是否进行任何更改取决于您所做的选择。您可以在还原集群页面中使用 MemoryDB 控制台进行还原。在使用 AWS CLI 或 MemoryDB API 进行还原时，您可以通过设置参数值来做出这些选择。

在还原操作过程中，MemoryDB 会创建新集群，然后使用快照文件中的数据填充。此过程完成后，集群即完成预热，准备好接受请求。

> **Important**
>
> 在继续之前，请确保您已创建要从中进行还原的集群快照。有关更多信息，请参阅 手动创建快照。
> 如果要从外部创建的快照进行还原，请参阅 使用外部创建的快照为新集群做种。

以下过程向您展示了如何使用 MemoryDB 控制台、或 MemoryDB API 将 AWS CLI快照还原到新集群。

从快照还原（控制台）

将快照还原到新集群（控制台）

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在导航窗格中，选择快照。
3. 在快照列表中，选中要还原的快照名称旁边的按钮。
4. 选择操作，然后选择还原
5. 在集群配置下，输入以下内容：
   a. 集群名称– 必填。新集群的名称。
   b. 描述 – 可选。新集群的描述。
6. 完成子网组部分：
   - 对于子网组，创建新的子网组，或从可用列表中选择要应用于此集群的现有子网组。如果要创建一个新的：
     - 输入名称
     - 输入描述
     - 如果启用了多可用区，则子网组必须至少包含两个位于不同可用区中的子网。有关更多信息，请参阅 子网和子网组。
     - 如果要创建新的子网组但不具有现有 VPC，则系统会要求您创建 VPC。有关更多信息，请参阅《Amazon VPC 用户指南》中的什么是 Amazon VPC？。
7. 完成集群设置部分：
   a. 为了实现 Valkey 版本兼容性或 Redis OSS 版本兼容性，请接受默认 `6.0`。
   b. 对于端口，请接受默认端口 6379，或者，如果您出于某个原因需要使用其他端口，请输入相应的端口号。
   c. 对于参数组，请接受 `default.memorydb-redis6` 参数组。

      参数组控制集群的运行时参数。有关参数组的更多信息，请参阅 引擎特定参数。
   d. 对于节点类型，请为所需节点类型（及其关联的内存大小）选择一个值。

如果您选择 r6gd 系列的节点类型，则系统会自动在集群中启用数据分层。有关更多信息，请参阅 数据分层。

e. 对于分片数，选择要用于此集群的分片数。

您可以动态更改集群中的分片数量。有关更多信息，请参阅 扩展 MemoryDB 集群。

f. 对于每个分片的副本数量，请选择每个分片中需要的只读副本节点数。

存在以下限制；。

- 如果启用了多可用区，请确保每个分片至少有一个副本。
- 使用控制台创建集群时，每个分片的副本数相同。

g. 选择下一步。

h. 完成高级设置部分：

i. 对于安全组，选择要用于该集群的安全组。安全组 充当防火墙来控制对集群的网络访问。您可以使用 VPC 的默认安全组或创建一个新的安全组。

有关安全组的更多信息，请参阅 Amazon VPC 用户指南中的您的 VPC 的安全组。

ii. 数据通过以下方式加密：

- Encryption at rest（静态加密）– 对磁盘上存储的数据启用加密。有关更多信息，请参阅静态加密。

> **Note**
>
> 您可以选择提供不同的加密密钥，方法是选择 “客户托管 AWS KMS 密钥” 并选择密钥。

- Encryption in-transit（传输中加密）– 对传输中数据启用加密。默认为启用状态。有关更多信息，请参阅传输中加密。

如果您选择不加密，则系统将使用默认用户创建一个名为“开放访问”的开放访问控制列表。有关更多信息，请参阅 使用访问控制列表对用户进行身份验证 () ACLs。

iii. 对于快照，请选择性地指定快照保留期和快照时段。默认情况下，启用自动快照处于选中状态。

iv. 对于维护时段，请选择性地指定维护时段。维护时段是每周中 MemoryDB 为您的集群计划系统维护的时间，通常以小时为时间长度。您可以允许 MemoryDB 选择维护时段的日期和时间（无首选项），或者自行选择日期、时间和持续时间（指定维护时段）。如果您从列表中选择指定维护时段，请选择维护时段的起始日、起始时间和持续时间（以小时为单位）。所有时间均为 UCT 时间。

有关更多信息，请参阅 管理维护。

v. 对于 通知，选择现有 Amazon Simple Notification Service (Amazon SNS) 主题，或选择手动 ARN 输入，然后输入主题的 Amazon 资源名称（ARN）。Amazon SNS 允许您向联网的智能设备推送通知。默认设置为禁用通知。有关更多信息，请参阅 https://aws.amazon.com/sns/。

i. 对于标签，您可以选择应用标签来搜索和筛选集群或跟踪 AWS 成本。

j. 查看您的所有输入和选择，然后进行任意所需的更正。准备就绪后，请选择创建集群启动集群或选择取消取消操作。

当您的集群状态为 available 时，您可向其授予 EC2 访问权限，连接到集群并开始使用它。有关更多信息，请参阅步骤 3：授予对集群的访问权限和步骤 4：连接到集群。

> **Important**
>
> 一旦您的集群变为可用状态，您便需要为集群处于活动状态的每个小时或分钟支付费用（即使您并未主动使用集群）。要停止此集群产生的费用，您必须将其删除。请参阅步骤 5：删除集群。

从快照恢复 (AWS CLI)

使用 `create-cluster` 操作时，请确保包括参数 `--snapshot-name` 或 `--snapshot-arns`，以使用来自快照的数据为新集群做种。

有关更多信息，请参阅下列内容：

- 创建集群 (AWS CLI) 在 MemoryDB 用户指南中。
- 在《 AWS CLI 命令@@ 参考》中创建集群。

从快照还原（MemoryDB API）

您可以使用 MemoryDB API 操作 `CreateCluster` 还原 MemoryDB 快照。

使用 `CreateCluster` 操作时，请确保包括参数 `SnapshotName` 或 `SnapshotArns`，以使用来自快照的数据为新集群做种。

有关更多信息，请参阅下列内容：

- 创建集群（MemoryDB API） 在 MemoryDB 用户指南中。
- CreateCluster在 MemoryDB API 参考中。

使用外部创建的快照为新集群做种

创建新 MemoryDB 集群时，可以使用 Valkey 或 Redis OSS .rdb 快照文件中的数据来作为种子。

要从 MemoryDB 快照或 ElastiCache (Redis OSS) 快照中播种新的 MemoryDB 集群，请参阅。从快照还原

使用 .rdb 文件为新 MemoryDB 集群做种时，您可以执行以下操作：

- 指定新集群中的分片数量。此数量可以与用于创建快照文件的集群中的分片数量不同。
- 为新集群指定不同的节点类型 – 大于或小于创建快照的集群中使用的节点类型。如果您决定缩减到较小的节点类型，则必须确保新节点类型拥有足量内存以适应您的数据和引擎开销。

> **Important**
>
> - 您必须确保快照数据不超过节点的资源容量。
>
> 如果快照太大，则所生成集群的状态将为 `restore-failed`。如果发生这种情况，您必须删除集群，从头再来。
>
> 有关节点类型和规范的完整列表，请参阅 MemoryDB 节点类型特定的参数。
> - 您只能使用 Amazon S3 服务器端加密（SSE-S3）对 .rdb 文件进行加密。有关更多信息，请参阅使用服务器端加密保护数据。

步骤 1：在外部集群上创建快照

创建快照为您的 MemoryDB 集群做种

1. 连接到现有 Valkey 或 Redis OSS 实例。
2. 运行 `BGSAVE` 或 `SAVE` 操作以创建快照。记录 .rdb 文件的位置。

   `BGSAVE` 是异步的，在处理期间不阻止其他客户端。有关更多信息，请参阅 BGSAVE。

   `SAVE` 同步的，在完成之前会阻止其他进程。有关更多信息，请参阅 SAVE。

有关创建快照的其他信息，请参阅持久化。

步骤 2：创建 Amazon S3 存储桶和文件夹

创建快照文件后，您需要将其上传到 Amazon S3 存储桶中的文件夹。要执行该操作，您必须先拥有 Amazon S3 存储桶以及该存储桶中的文件夹。如果您已有 Amazon S3 存储桶和文件夹并具备相应权限，则可以跳到 步骤 3：将快照上传到 Amazon S3。

创建 Amazon S3 存储桶

1. 登录 AWS 管理控制台 并打开 Amazon S3 控制台，网址为https://console.aws.amazon.com/s3/。
2. 按照 Amazon Simple Storage Service 用户指南中的创建存储桶的说明，创建 Amazon S3 存储桶。

   Amazon S3 存储桶的名称必须符合 DNS 标准。否则，MemoryDB 无法访问您的备份文件。DNS 合规性规则包括：

   - 名称的长度必须为至少 3 个字符，且不能超过 63 个字符。
   - 名称必须是由句点（.）分隔的一个或多个标签组成的系列，其中每个标签：
     - 以小写字母或数字开头。
     - 以小写字母或数字结尾。
     - 仅包含小写字母、数字和短划线。
   - 名称不能采用 IP 地址格式（例如 192.0.2.0）。

   我们强烈建议您在与新 MemoryDB 集群相同的 AWS 区域中创建 Amazon S3 存储桶。此方式可确保当 MemoryDB 从 Amazon S3 读取 .rdb 文件时，数据传输速度达到最高。

   > **Note**
   >
   > 为了使您的数据尽可能安全，请尽可能限制您的 Amazon S3 存储桶的权限。同时，权限仍然需要允许存储桶及其内容用于为新的 MemoryDB 集群设定种子。

向 Amazon S3 存储桶添加文件夹

1. 登录 AWS 管理控制台 并打开 Amazon S3 控制台，网址为https://console.aws.amazon.com/s3/。
2. 选择将 .rdb 文件上传到的存储桶的名称。
3. 请选择 Create folder（创建文件夹）。
4. 输入新文件夹的名称。

5. 选择保存。

   记录存储桶名称和文件夹名称。

步骤 3：将快照上传到 Amazon S3

现在，上传您在步骤 1：在外部集群上创建快照中创建的 .rdb 文件。将其上传到您在 步骤 2：创建 Amazon S3 存储桶和文件夹 中创建的 Amazon S3 存储桶和文件夹。有关该任务的更多信息，请参阅上传对象。在步骤 2 和 3 之间，选择您创建的文件夹的名称。

将 .rdb 文件上传到 Amazon S3 文件夹

1. 登录 AWS 管理控制台 并打开 Amazon S3 控制台，网址为https://console.aws.amazon.com/s3/。
2. 选择您在步骤 2 中创建的 Amazon S3 存储桶的名称。
3. 选择您在步骤 2 中创建的文件夹的名称。
4. 选择上传。
5. 选择添加文件。
6. 浏览查找要上传的一个或多个文件，然后选择文件。要选择多个文件，请在选择每个文件名时按住 Ctrl 键。
7. 选择 Open（打开）。
8. 确认上传页面中列出了正确的文件，然后选择上传。

记下 .rdb 文件的路径。例如，如果存储桶名称为 `amzn-s3-demo-bucket` 并且路径为 `myFolder/redis.rdb`，请输入 `amzn-s3-demo-bucket/myFolder/redis.rdb`。使用此快照中的数据为新集群做种时需要此路径。

有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的存储桶命名规则。

步骤 4：授予 MemoryDB 对 .rdb 文件的读取访问权限

AWS 2019 年 3 月 20 日之前推出的区域默认处于启用状态。您可以立即开始在这些 AWS 地区工作。2019 年 3 月 20 日之后推出的区域默认情况下处于禁用状态。您必须按照管理 AWS 区域所述，先启用或选择加入这些区域，然后才能使用它们。

授予 MemoryDB 对 .rdb 文件的读取访问权限

向 MemoryDB 授予对快照文件的读取访问权限

1. 登录 AWS 管理控制台 并打开 Amazon S3 控制台，网址为https://console.aws.amazon.com/s3/。
2. 选择包含您 .rdb 文件的 S3 存储桶的名称。
3. 选择包含 .rdb 文件的文件夹的名称。
4. 选择 .rdb 快照文件的名称。所选文件的名称将显示在页面顶部的选项卡上方。
5. 选择权限选项卡。
6. 在 Permissions（权限）下，选择 Bucket policy（存储桶策略），然后选择 Edit（编辑）。
7. 更新策略以授予 MemoryDB 执行操作所需的权限：

   - 将 `[ "Service" : "region-full-name.memorydb-snapshot.amazonaws.com" ]` 添加到 `Principal`。
   - 添加将快照导出到 Amazon S3 存储桶所需的以下权限：
     - `"s3:GetObject"`
     - `"s3:ListBucket"`
     - `"s3:GetBucketAcl"`

   以下是更新策略具体形式的示例。

   JSON

```
{
    "Version":"2012-10-17",
    "Id": "Policy15397346",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "us-east-1.memorydb-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/snapshot1.rdb",
                "arn:aws:s3:::amzn-s3-demo-bucket/snapshot2.rdb"
            ]
        }
    ]
}
```

8. 选择保存。

步骤 5：使用 .rdb 文件数据为 MemoryDB 集群做种

现在，您已准备好创建 MemoryDB 集群并使用 .rdb 文件中的数据为其做种。要创建集群，请按照 创建 MemoryDB 集群 中的说明操作。

当告知 MemoryDB 在何处查找已上传到 Amazon S3 的快照时所采用的方法取决于您创建集群时采用的方法：

使用 .rdb 文件数据为 MemoryDB 集群做种

- 使用 MemoryDB 控制台

  选择引擎之后，展开高级设置部分，然后找到将数据导入集群。在 Seed RDB file S3 location（使用 RDB 文件 S3 位置设定种子）框中，键入文件的 Amazon S3 路径。如果您有多个 .rdb 文件，则以逗号分隔的列表形式键入各文件的路径。Amazon S3 路径类似于 *amzn-s3-demo-bucket*/*myFolder*/*myBackupFilename*.rdb。

- 使用 AWS CLI

  如果您使用 `create-cluster` 或 `create-cluster` 操作，请使用参数 `--snapshot-arns` 为各 .rdb 文件指定完全限定的 ARN。例如 `arn:aws:s3:::`*amzn-s3-demo-bucket*/*myFolder*/*myBackupFilename*`.rdb`。ARN 必须解析为您存储在 Amazon S3 中的快照文件。

- 使用 MemoryDB API

  如果您使用 `CreateCluster` 或 `CreateCluster` MemoryDB API 操作，请使用参数 `SnapshotArns` 为各 .rdb 文件指定完全限定的 ARN。例如 `arn:aws:s3:::`*amzn-s3-demo-bucket*/*myFolder*/*myBackupFilename*`.rdb`。ARN 必须解析为您存储在 Amazon S3 中的快照文件。

在创建集群的过程中，快照中的数据将写入集群。您可通过查看 MemoryDB 事件消息来监控进度。为此，请参阅 MemoryDB 控制台，然后选择事件。您也可以使用 M AWS emoryDB 命令行界面或 MemoryDB API 来获取事件消息。

标记快照

您可以标签形式将自己的元数据分配给各个快照。标签可让您按各种标准（例如用途、所有者或环境）对快照进行分类。这在您具有相同类型的很多资源时会很有用 – 您可以根据分配给特定资源的标签快速识别该资源。有关更多信息，请参阅 您可以为之添加标签的资源。

成本分配标签是一种通过标签值对发票上的费用进行分组来跟踪多项 AWS 服务成本的方法。要了解有关成本分配标签的更多信息，请参阅使用成本分配标签。

使用 MemoryDB 控制台 AWS CLI、或 MemoryDB API，您可以在快照上添加、列出、修改、移除或复制成本分配标签。有关更多信息，请参阅 使用成本分配标签监控成本。

删除快照

自动快照会在其保留期限过期时自动删除。如果您删除某个集群，则会删除其所有的自动快照。

MemoryDB 提供了一个删除 API 操作，可用于随时删除快照，无论快照是自动还是手动创建的。由于手动快照没有保留期限，所以手动删除是移除备份的唯一方法。

您可以使用 MemoryDB 控制台 AWS CLI、或 MemoryDB API 删除快照。

删除快照（控制台）

以下过程使用 MemoryDB 控制台删除快照。

删除快照

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在左侧导航窗格中，选择快照。

   此时会显示“快照”屏幕，其中包含您的快照列表。
3. 选择要删除的快照名称左侧的单选按钮。
4. 选择操作，然后选择删除。
5. 如果要删除此快照，请在文本框中输入 `delete`，然后选择删除。要取消删除，请选择取消。状态将变为正在删除。

删除快照 (AWS CLI)

使用带有以下参数的删除快照 AWS CLI 操作来删除快照。

- `--snapshot-name` – 要删除的快照名称。

以下代码删除快照 `myBackup`。

```
aws memorydb delete-snapshot --snapshot-name myBackup
```

有关更多信息，请参阅 AWS CLI 命令参考中的 delete-snapshot。

删除快照（MemoryDB API）

使用带以下参数的 `DeleteSnapshot` API 操作删除快照。

- SnapshotName – 要删除的快照名称。

以下代码删除快照 myBackup。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DeleteSnapshot
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SnapshotName=myBackup
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 DeleteSnapshot。

扩展

您的应用程序需要处理的数据量几乎不会保持不变。它会随着您的业务增长或遇到正常的业务波动时增减。如果您自行管理应用程序，则需要预配置足量硬件来满足您的需求高峰，这会产生很高的费用。通过使用 MemoryDB，您可以扩展以满足当前需求，并且只需为您的使用量付费。

以下信息可帮助您查找有关要执行的扩展操作的正确主题。

扩展 MemoryDB

Action	MemoryDB
横向扩展	MemoryDB 的在线重新分片
更改节点类型	通过修改节点类型来在线纵向扩展
更改分片数量	扩展 MemoryDB 集群

扩展 MemoryDB 集群

由于对您的集群的需求发生变化，您可能决定通过更改 MemoryDB 集群中的分片数量来提高性能或降低成本。我们建议使用在线水平扩展来实现这一目的，因为采用这种方法，您的集群在扩展过程中可以继续为请求提供服务。

您决定重新调节集群的情况包括以下几种：

- 内存压力：

  如果集群中的节点存在内存压力，您可能会决定进行横向扩展，以便获得更多资源来更好地存储数据并为请求提供服务。

  您可以通过监控以下指标来确定您的节点是否承受内存压力：FreeableMemorySwapUsage、和 BytesUsedForMemoryDB。
- CPU 或网络瓶颈：

  如果 latency/throughput 问题困扰着您的集群，则可能需要向外扩展以解决问题。

  您可以通过监控以下指标来监控延迟和吞吐量级
  别：CPUUtilizationNetworkBytesIn、NetworkBytesOut、CurrConnections、和NewConnections。
- 您的集群过度扩展：

  对集群的当前需求是缩减集群不会降低性能，并可以降低成本。

  您可以使用以下指标监控集群的使用情况，以确定是否可以安全地进行扩
  展：FreeableMemorySwapUsageBytesUsedForMemory、、CPUUtilization、NetworkBytesInNetworkByte
  和NewConnections。

扩展的性能影响

当使用离线过程进行扩展时，您的集群在大部分过程中处于离线状态，因此无法为请求提供服务。当使用在线方法进行扩展时，由于扩展是计算密集型操作，因此会导致一定程度的性能下降，但是在整个扩展操作过程中您的集群仍然会继续为请求提供服务。性能的降低程度取决于您的常规 CPU 利用率和数据。

有两种方法可以扩展您的 MemoryDB 集群：横向和纵向扩展。

- 利用横向扩展，可以通过添加或移除分片来更改集群中的分片数量。在线重新分片过程允许在集群继续处理传入请求的 in/out 同时进行扩展。

- 纵向扩展 – 更改节点类型以调整集群大小。在线垂直扩展允许在集群继续处理传入请求的 up/down 同时进行扩展。

如果要通过横向缩减或纵享缩减来减小集群的大小和内存容量，请确保新配置具有足够的内存用于数据和引擎开销。

MemoryDB 的离线重新分片

离线分片重新配置带来的主要优势便是，除了在集群中添加或删除分片以外，您还可以执行更多操作。在进行离线重新分片时，除了更改集群中的分片数量，您还可以执行以下操作：

- 更改集群的节点类型。
- 升级为更新的引擎版本。

> **Note**
>
> 启用了数据分层的集群不支持离线重新分片。有关更多信息，请参阅数据分层。

离线分片重新配置的主要缺点是，从过程的还原部分开始直到更新应用程序中的终端节点，集群一直处于离线状态。您的集群处于离线状态的时间长短因集群中的数据量而异。

在离线状态下重新配置分片 MemoryDB 集群

1. 创建现有 MemoryDB 集群的手动快照。有关更多信息，请参阅 手动创建快照。
2. 通过从快照中还原来创建新集群。有关更多信息，请参阅 从快照还原。
3. 将您的应用程序中的终端节点更新为新集群的终端节点。有关更多信息，请参阅 查找连接端点。

MemoryDB 的在线重新分片

通过 MemoryDB 使用在线重新分片，您可以在无需停机的情况下动态扩展 MemoryDB。此方法意味着，即使在进行扩展或重新平衡的过程中，您的集群也可以继续为请求提供服务。

您可执行以下操作：

- 横向扩展 – 通过向 MemoryDB 集群添加分片来增加读写容量。

如果您向集群添加一个或多个分片，则每个新分片中的节点数量与最小的现有分片中的节点数量相同。

- 横向缩减 – 通过删除 MemoryDB 集群中的分片降低读写容量，从而降低成本。

目前，以下限制适用于 MemoryDB 在线重新分片：

- 槽或键空间和大型项目存在以下限制：

  如果分片中的任何键包含一个大型项，在横向扩展时关键字不会迁移到新分片。此功能会导致分片不平衡。

  如果某个分片中的任何密钥包含大型项目（序列化后大于 256MB 的项目），则在缩减时不会删除该分片。此功能可导致某些分片无法删除。
- 在横向扩展时，任何新分片中的节点数量等于现有分片中的节点数量。

有关更多信息，请参阅 最佳实践：在线调整集群大小。

您可以使用 AWS 管理控制台、 AWS CLI和 MemoryDB API 水平扩展 MemoryDB 集群。

通过在线重新分片功能添加分片

你可以使用 AWS 管理控制台、 AWS CLI或 MemoryDB API 向你的 MemoryDB 集群添加分片。

添加分片（控制台）

您可以使用将一个或多个分片 AWS 管理控制台 添加到您的 MemoryDB 集群。以下步骤描述了这个过程。

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 从集群列表中，选择要从中添加分片的集群的名称。
3. 在分片和节点选项卡下，选择添加/删除分片
4. 在新分片数中，输入所需的分片数。
5. 选择确认保留更改，或选择取消放弃更改。

添加分片（AWS CLI）

以下过程介绍了如何通过使用 AWS CLI添加分片的方法重新配置 MemoryDB 集群中的分片。

在update-cluster中使用以下参数：

参数

- --cluster-name – 必需。指定在哪个集群上执行分片重新配置操作。
- --shard-configuration – 必需。允许设置分片数量。
  - ShardCount – 设置此属性指定所需要的分片数量。

Example

以下示例将集群 my-cluster 中的分片数量修改为 2。

对于 Linux、macOS 或 Unix：

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --shard-configuration \
        ShardCount=2
```

对于 Windows：

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --shard-configuration ^
        ShardCount=2
```

返回以下 JSON 响应：

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 2,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
```

```
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

若要在已更新集群的状态从更新变为可用后查看其详细信息，请使用以下命令：

对于 Linux、macOS 或 Unix：

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

对于 Windows：

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

返回以下 JSON 响应：

```
{
    "Clusters": [
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 2,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
```

```
                    "Slots": "0-8191",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                },
                {
                    "Name": "0002",
                    "Status": "available",
                    "Slots": "8192-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0002-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-22T14:26:18.693000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
```

```
                            "Name": "my-cluster-0002-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-22T14:26:18.765000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

有关更多信息，请参阅《命令参考》中的 update-cluster。 AWS CLI

添加分片（MemoryDB API）

您可以通过 UpdateCluster 操作使用 MemoryDB API 在线重新配置 MemoryDB 集群中的分片。

在UpdateCluster中使用以下参数：

参数

- `ClusterName` – 必需。指定在哪个集群上执行分片重新配置操作。
- `ShardConfiguration` – 必需。允许设置分片数量。
  - `ShardCount` – 设置此属性指定所需要的分片数量。

有关更多信息，请参阅 UpdateCluster。

通过在线重新分片功能删除分片

您可以使用 AWS 管理控制台、 AWS CLI或 MemoryDB API 从 MemoryDB 集群中移除分片。

删除分片（控制台）

以下过程介绍了如何通过使用 AWS 管理控制台删除分片的方法重新配置 MemoryDB 集群中的分片。

> **Important**
>
> 在从集群中删除分片之前，MemoryDB 可确保所有数据将适合其余分片。如果数据适合，将根据要求从集群中删除分片。如果数据不适合剩余的分片，则过程将终止，并且集群的分片配置将保留为与发出请求之前相同。

您可以使用从 MemoryDB 集群中移除一个或多个分片。 AWS 管理控制台 您无法移除集群中的所有分片。而是必须删除集群。有关更多信息，请参阅 步骤 5：删除集群。以下步骤描述了移除一个或多个分片的过程。

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 从集群列表中，选择要从中删除分片的集群的名称。
3. 在分片和节点选项卡下，选择添加/删除分片
4. 在新分片数中，输入所需的分片数（最少为 1）。
5. 选择确认保留更改，或选择取消放弃更改。

删除分片（AWS CLI）

以下过程介绍了如何通过使用 AWS CLI删除分片的方法重新配置 MemoryDB 集群中的分片。

> **Important**
>
> 在从集群中删除分片之前，MemoryDB 可确保所有数据将适合其余分片。如果数据适合，将根据要求从集群中删除分片，并将其密钥空间映射到其余分片。如果数据不适合剩余的分片，则过程将终止，并且集群的分片配置将保留为与发出请求之前相同。

您可以使用从 MemoryDB 集群中移除一个或多个分片。 AWS CLI 您无法移除集群中的所有分片。而是必须删除集群。有关更多信息，请参阅 步骤 5：删除集群。

在update-cluster中使用以下参数：

参数

- --cluster-name – 必需。指定在哪个集群上执行分片重新配置操作。
- --shard-configuration – 必需。允许使用 ShardCount 属性设置分片数量：

  ShardCount – 设置此属性指定所需要的分片数量。

Example

以下示例将集群 my-cluster 中的分片数量修改为 2。

对于 Linux、macOS 或 Unix：

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --shard-configuration \
        ShardCount=2
```

对于 Windows：

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --shard-configuration ^
        ShardCount=2
```

返回以下 JSON 响应：

```
{
    "Cluster": {
```

```
        "Name": "my-cluster",
        "Status": "updating",
        "NumberOfShards": 2,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-east-1.amazonaws.com",
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "6.2",
        "EnginePatchVersion": "6.2.6",
        "ParameterGroupName": "default.memorydb-redis6",
        "ParameterGroupStatus": "in-sync",
        "SubnetGroupName": "my-sg",
        "TLSEnabled": true,
        "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "DataTiering": "false",
        "AutoMinorVersionUpgrade": true
    }
}
```

若要在已更新集群的状态从更新变为可用后查看其详细信息，请使用以下命令：

对于 Linux、macOS 或 Unix：

```
aws memorydb describe-clusters \
    --cluster-name my-cluster
    --show-shard-details
```

对于 Windows：

```
aws memorydb describe-clusters ^
    --cluster-name my-cluster
    --show-shard-details
```

返回以下 JSON 响应：

```
{
    "Clusters": [
```

```
        {
            "Name": "my-cluster",
            "Status": "available",
            "NumberOfShards": 2,
            "Shards": [
                {
                    "Name": "0001",
                    "Status": "available",
                    "Slots": "0-8191",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0001-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0001-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
                            "CreateTime": "2021-08-21T20:22:12.405000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                },
                {
                    "Name": "0002",
                    "Status": "available",
                    "Slots": "8192-16383",
                    "Nodes": [
                        {
                            "Name": "my-cluster-0002-001",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1b",
```

```
                            "CreateTime": "2021-08-22T14:26:18.693000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        },
                        {
                            "Name": "my-cluster-0002-002",
                            "Status": "available",
                            "AvailabilityZone": "us-east-1a",
                            "CreateTime": "2021-08-22T14:26:18.765000-07:00",
                            "Endpoint": {
                                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                                "Port": 6379
                            }
                        }
                    ],
                    "NumberOfNodes": 2
                }
            ],
            "ClusterEndpoint": {
                "Address": "clustercfg.my-cluster.xxxxxx.memorydb.us-
east-1.amazonaws.com",
                "Port": 6379
            },
            "NodeType": "db.r6g.large",
            "EngineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "ParameterGroupName": "default.memorydb-redis6",
            "ParameterGroupStatus": "in-sync",
            "SubnetGroupName": "my-sg",
            "TLSEnabled": true,
            "ARN": "arn:aws:memorydb:us-east-1:xxxxxxexamplearn:cluster/my-cluster",
            "SnapshotRetentionLimit": 0,
            "MaintenanceWindow": "wed:03:00-wed:04:00",
            "SnapshotWindow": "04:30-05:30",
            "ACLName": "my-acl",
            "DataTiering": "false",
            "AutoMinorVersionUpgrade": true
        }
    ]
```

```
}
```

有关更多信息，请参阅《命令参考》中的 update-cluster。 AWS CLI

移除分片（MemoryDB API）

您可以通过 `UpdateCluster` 操作使用 MemoryDB API 在线重新配置 MemoryDB 集群中的分片。

以下过程介绍了如何通过使用 MemoryDB API 删除分片的方法重新配置 MemoryDB 集群中的分片。

> **Important**
>
> 在从集群中移除分片之前，MemoryDB 可确保所有数据适合其余分片。如果数据适合，将根据要求从集群中删除分片，并将其密钥空间映射到其余分片。如果数据不适合剩余的分片，则过程将终止，并且集群的分片配置将保留为与发出请求之前相同。

您可以使用 MemoryDB API 从 MemoryDB 集群中移除一个或多个分片。您无法移除集群中的所有分片。而是必须删除集群。有关更多信息，请参阅 步骤 5：删除集群。

在`UpdateCluster`中使用以下参数：

参数

- `ClusterName` – 必需。指定在哪个集群上执行分片重新配置操作。
- `ShardConfiguration` – 必需。允许使用 `ShardCount` 属性设置分片数量：

  `ShardCount` – 设置此属性指定所需要的分片数量。

通过修改节点类型来在线纵向扩展

通过对 MemoryDB 使用在线纵向扩展，您可以在最短停机的情况下动态扩展集群。这样，即使在扩展时，您的集群也可以处理请求。

> **Note**
>
> 不支持在使用数据分层功能的集群（例如，使用 r6gd 节点类型的集群）和不使用数据分层功能的集群（例如，使用 r6g 节点类型的集群）之间扩缩。有关更多信息，请参阅 数据分层。

您可执行以下操作：

- 纵向扩展 – 通过调整 MemoryDB 集群的节点类型以使用较大的节点类型来增加读取和写入容量。

  MemoryDB 动态调整集群大小，同时保持在线并处理请求。
- 缩减 – 通过向下调整节点类型以使用较小节点来减少读写容量。同样，MemoryDB 动态调整集群大小，同时保持在线并处理请求。在这种情况下，您可以通过缩小节点来降低成本。

> **Note**
>
> 扩展和缩减过程依赖于使用新选择的节点类型创建集群并将新节点与先前节点同步。为确保秤 up/down 流畅无阻，请执行以下操作：
>
> - 虽然纵向扩展过程旨在保持完全在线，但它确实依赖于在旧节点和新节点之间同步数据。我们建议您在预期数据流量最小时启动扩展/缩减。
> - 尽可能在生产前调试环境中测试扩展期间的应用程序行为。

在线纵向扩展

主题

- 纵向扩展 MemoryDB 集群（控制台）
- 扩展 MemoryDB 集群 (CLI AWS)
- 纵向扩展 MemoryDB 集群（MemoryDB API）

纵向扩展 MemoryDB 集群（控制台）

以下过程介绍如何使用 AWS 管理控制台纵向扩展 MemoryDB 集群。在此过程中，MemoryDB 集群将继续处理请求，且停机时间降至最短。

纵向扩展集群（控制台）

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 从集群列表中，选择集群。
3. 选择 Actions（操作），然后选择 Modify（修改）。
4. 在修改集群对话框中：

- 从 Node type 列表中选择您希望扩展到的节点类型。要扩展，请选择大于现有节点的节点类型。

5. 选择保存更改。

   集群状态更改为正在修改。当状态变为 available 时，即表示修改完成，您可以开始使用新集群。

扩展 MemoryDB 集群 (CLI AWS)

以下过程介绍如何使用 AWS CLI纵向扩展 MemoryDB 集群。在此过程中，MemoryDB 集群将继续处理请求，且停机时间降至最短。

向上扩展 MemoryDB 集群 (CLI AWS)

1. 通过运行带有以下参数的 AWS CLI `list-allowed-node-type-updates`命令来确定可以扩展到的节点类型。

   对于 Linux、macOS 或 Unix：

   ```
   aws memorydb list-allowed-node-type-updates \
        --cluster-name my-cluster-name
   ```

   对于 Windows：

   ```
   aws memorydb list-allowed-node-type-updates ^
        --cluster-name my-cluster-name
   ```

   以上命令的输出类似于此处所示（JSON 格式）。

   ```
   {
        "ScaleUpNodeTypes": [
            "db.r6g.2xlarge",
            "db.r6g.large"
        ],
        "ScaleDownNodeTypes": [
            "db.r6g.large"
        ],
   }
   ```

   有关更多信息，请参阅《AWS CLI 参考资料》中的 list-allowed-node-type-up dates。

2. 使用 AWS CLI `update-cluster`命令和以下参数修改集群以向上扩展到新的更大的节点类型。

   - `--cluster-name` – 要纵向扩展的集群的名称。
   - `--node-type` – 要扩展集群的新节点类型。此值必须是步骤 1 中由 `list-allowed-node-type-updates` 命令返回的节点类型之一。

   对于 Linux、macOS 或 Unix：

   ```
   aws memorydb update-cluster  \
    --cluster-name my-cluster \
    --node-type db.r6g.2xlarge
   ```

   对于 Windows：

   ```
   aws memorydb update-cluster ^
        --cluster-name my-cluster ^
        --node-type db.r6g.2xlarge ^
   ```

   有关更多信息，请参阅 update-cluster。

纵向扩展 MemoryDB 集群（MemoryDB API）

以下过程使用 MemoryDB API 将集群从其当前节点类型扩展为较大的新节点类型。在此过程中，MemoryDB 会更新 DNS 条目使其指向新的节点。您可以在该集群继续保持在线并处理传入请求时扩展启用自动失效转移的集群。

纵向扩展为较大的节点类型所需的时间因节点类型和当前集群中的数据量不同而异。

纵向扩展 MemoryDB 集群（MemoryDB API）

1. 使用带以下参数的 MemoryDB API `ListAllowedNodeTypeUpdates` 操作确定您可纵向扩展的节点类型。

   - `ClusterName` – 集群的名称。使用此参数可描述特定集群而非所有集群。

   ```
   https://memory-db.us-east-1.amazonaws.com/
       ?Action=ListAllowedNodeTypeUpdates
   ```

```
    &ClusterName=MyCluster
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T192317Z
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅《M emoryDB API 参考》ListAllowedNodeTypeUpdates中的。

2. 使用带以下参数的 UpdateCluster MemoryDB API 操作将当前集群扩展为新的节点类型。

   - ClusterName – 集群的名称。
   - NodeType – 此集群中集群的较大的新节点类型。此值必须是步骤 1 中由 ListAllowedNodeTypeUpdates 操作返回的实例类型之一。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateCluster
    &NodeType=db.r6g.2xlarge
    &ClusterName=myCluster
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

有关更多信息，请参阅 UpdateCluster。

在线缩减

主题

- 缩减 MemoryDB 集群（控制台）
- 缩小 MemoryDB 集群 (CLI AWS)
- 缩减 MemoryDB 集群（MemoryDB API）

缩减 MemoryDB 集群（控制台）

以下过程介绍如何使用 AWS 管理控制台缩减 MemoryDB 集群。在此过程中，MemoryDB 集群将继续处理请求，且停机时间降至最短。

缩减 MemoryDB 集群（控制台）

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 从集群列表中，选择首选集群。
3. 选择 Actions（操作），然后选择 Modify（修改）。
4. 在修改集群对话框中：
   - 从 Node type 列表中选择您希望扩展到的节点类型。要缩减，请选择小于现有节点的节点类型。请注意，并不是可缩减到所有节点类型。
5. 选择保存更改。

   集群状态更改为正在修改。当状态变为 available 时，即表示修改完成，您可以开始使用新集群。

缩小 MemoryDB 集群 (CLI AWS)

以下过程介绍如何使用 AWS CLI缩减 MemoryDB 集群。在此过程中，MemoryDB 集群将继续处理请求，且停机时间降至最短。

缩小 MemoryDB 集群 (CLI AWS)

1. 通过运行带有以下参数的 AWS CLI `list-allowed-node-type-updates`命令来确定可以缩减到的节点类型。

   对于 Linux、macOS 或 Unix：

   ```
   aws memorydb list-allowed-node-type-updates \
        --cluster-name my-cluster-name
   ```

   对于 Windows：

   ```
   aws memorydb list-allowed-node-type-updates ^
        --cluster-name my-cluster-name
   ```

以上命令的输出类似于此处所示（JSON 格式）。

```
{
     "ScaleUpNodeTypes": [
         "db.r6g.2xlarge",
         "db.r6g.large"
     ],
     "ScaleDownNodeTypes": [
         "db.r6g.large"
     ],
}
```

有关更多信息，请参阅 list-allowed-node-type-updates。

2. 使用 `update-cluster` 命令和以下参数修改集群以缩减为较小的新节点类型。

   - `--cluster-name` – 要缩减的集群的名称。
   - `--node-type` – 要扩展集群的新节点类型。此值必须是步骤 1 中由 `list-allowed-node-type-updates` 命令返回的节点类型之一。

   对于 Linux、macOS 或 Unix：

```
aws memorydb update-cluster  \
     --cluster-name my-cluster \
     --node-type db.r6g.large
```

   对于 Windows：

```
aws memorydb update-cluster ^
     --cluster-name my-cluster ^
     --node-type db.r6g.large
```

   有关更多信息，请参阅 update-cluster。

缩减 MemoryDB 集群（MemoryDB API）

以下过程使用 MemoryDB API 将集群从其当前节点类型扩展为较小的新节点类型。在此过程中，MemoryDB 集群将继续处理请求，且停机时间降至最短。

缩减为较小的节点类型所需的时间因节点类型和当前集群中的数据量而异。

缩减（MemoryDB API）

1. 使用带有以下参数的 ListAllowedNodeTypeUpdatesAPI 来确定可以缩减为哪些节点类型：

   - ClusterName – 集群的名称。使用此参数可描述特定集群而非所有集群。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=ListAllowedNodeTypeUpdates
    &ClusterName=MyCluster
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210802T192317Z
    &X-Amz-Credential=<credential>
```

2. 使用带有以下参数的 UpdateClusterAPI 将当前集群缩小到新的节点类型。

   - ClusterName – 集群的名称。
   - NodeType – 此集群中集群的较小的新节点类型。此值必须是步骤 1 中由 ListAllowedNodeTypeUpdates 操作返回的实例类型之一。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateCluster
    &NodeType=db.r6g.2xlarge
    &ClusterName=myReplGroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

使用参数组配置引擎参数

MemoryDB 使用参数控制节点和集群的运行时属性。通常，更新的引擎版本包含用于支持更新功能的其他参数。有关参数表，请参阅引擎特定参数。

正如您所预期的，某些参数值（例如 `maxmemory`）由引擎和节点类型决定。有关由节点类型决定的这些参数值的表，请参阅MemoryDB 节点类型特定的参数。

主题

- 参数管理
- 参数组层
- 创建参数组
- 按名称列出参数组
- 列出参数组的值
- 修改参数组
- 删除参数组
- 引擎特定参数

参数管理

参数被分组到命名的参数组中，以便更轻松地管理参数。参数组表示在启动时传递给引擎软件的参数的特定值组合。这些值确定每个节点上的引擎进程在运行时的行为方式。特定参数组中的参数值应用于与该组关联的所有节点（不论这些节点属于哪个集群）。

要优化集群的性能，您可以修改某些参数值或更改集群的参数组。

- 您无法修改或删除默认参数组。如果您需要自定义参数值，必须创建自定义参数组。
- 参数组系列和您要分配给的集群必须兼容。例如，如果您的集群运行 Redis OSS 版本 6，您只能使用 memorydb_redis6 系列中的参数组（默认或自定义）。
- 当您更改集群的参数时，更改会立即应用到集群。无论您是更改集群的参数组本身，还是更改集群参数组内的参数值，都是如此。

参数组层

MemoryDB 参数组层

全局默认值

用于区域中所有 MemoryDB 客户的顶级根参数组。

全局默认参数组：

- 预留供 MemoryDB 使用，对客户不可用。

客户默认值

创建供用户使用的全局默认参数组的副本。

客户默认参数组：

- 由 MemoryDB 创建和拥有。
- 可供客户用作参数组，用于运行此参数组所支持引擎版本的任意集群。
- 无法由客户编辑。

客户拥有

客户默认参数组的副本。客户拥有的参数组在客户创建参数组时创建。

客户拥有的参数组：

- 由客户创建并拥有。
- 可以分配给任意客户兼容的集群。
- 可由客户修改用于创建自定义参数组。

  并非所有参数值均可修改。有关更多信息，请参阅 引擎特定参数。

创建参数组

如果存在一个或多个要从默认值更改的参数值，则需要创建新参数组。您可以使用 MemoryDB 控制台 AWS CLI、或 MemoryDB API 创建参数组。

创建参数组（控制台）

以下过程介绍了如何使用 MemoryDB 控制台创建参数组。

使用 MemoryDB 控制台创建参数组

1. 登录AWS 管理控制台并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 要查看所有可用的参数组列表，请在导航窗格左侧选择Parameter Groups。
3. 要创建参数组，请选择 Create parameter group。

   创建参数组页面将显示。
4. 在 Name 框中，键入此参数组的唯一名称。

   在创建集群或修改集群的参数组时，您将按参数组的名称选择参数组。因此，建议名称具有信息性，并且以某种方法标识该参数组的系列。

   参数组命名约束如下：

   - 必须以 ASCII 字母开头。
   - 只能包含 ASCII 字母、数字和连字符。
   - 长度必须介于 1 到 255 个字符之间。
   - 不能包含两个连续连字符。
   - 不能以连字符结束。
5. 在 Description 框中，键入参数组的说明。
6. 在引擎版本兼容性框中，选择与参数组对应的引擎版本。
7. 在标签中，可以选择添加标签以搜索和筛选参数组或跟踪AWS成本。
8. 要创建参数组，请选择 Create。

   要在不创建参数组的情况下终止此过程，请选择 Cancel。
9. 创建参数组后，它将具有系列的默认值。要更改默认值，您必须修改参数组。有关更多信息，请参阅 修改参数组。

创建参数组 (AWSCLI)

要使用创建参数组AWS CLI，请使用`create-parameter-group`带有这些参数的命令。

- `--parameter-group-name` – 参数组的名称。

  参数组命名约束如下：

  - 必须以 ASCII 字母开头。
  - 只能包含 ASCII 字母、数字和连字符。
  - 长度必须介于 1 到 255 个字符之间。
  - 不能包含两个连续连字符。
  - 不能以连字符结束。
- `--family` – 参数组的引擎和版本系列。
- `--description` – 用户提供的参数组描述。

Example

以下示例使用 memorydb_redis6 系列作为模板来创建名为 myRedis6x 的参数组。

对于 Linux、macOS 或 Unix：

```
aws memorydb create-parameter-group \
    --parameter-group-name myRedis6x  \
    --family memorydb_redis6 \
    --description "My first parameter group"
```

对于 Windows：

```
aws memorydb create-parameter-group ^
    --parameter-group-name myRedis6x  ^
    --family memorydb_redis6 ^
    --description "My first parameter group"
```

该命令的输出内容应类似如下所示。

```
{
    "ParameterGroup": {
        "Name": "myRedis6x",
        "Family": "memorydb_redis6",
        "Description": "My first parameter group",
        "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x"
    }
```

```
}
```

创建参数组后，它将具有系列的默认值。要更改默认值，您必须修改参数组。有关更多信息，请参阅修改参数组。

有关更多信息，请参阅 create-parameter-group。

创建参数组（MemoryDB API）

要使用 MemoryDB API 创建参数组，请使用带以下参数的 CreateParameterGroup 操作。

- ParameterGroupName – 参数组的名称。

  参数组命名约束如下：

  - 必须以 ASCII 字母开头。
  - 只能包含 ASCII 字母、数字和连字符。
  - 长度必须介于 1 到 255 个字符之间。
  - 不能包含两个连续连字符。
  - 不能以连字符结束。
- Family – 参数组的引擎和版本系列。例如 memorydb_redis6。
- Description – 用户提供的参数组描述。

Example

以下示例使用 memorydb_redis6 系列作为模板来创建名为 myRedis6x 的参数组。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=CreateParameterGroup
   &Family=memorydb_redis6
   &ParameterGroupName=myRedis6x
   &Description=My%20first%20parameter%20group
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

来自此操作的响应应类似如下所示。

```
<CreateParameterGroupResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
  <CreateParameterGroupResult>
    <ParameterGroup>
      <Name>myRedis6x</Name>
      <Family>memorydb_redis6</Family>
      <Description>My first parameter group</Description>
      <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>
    </ParameterGroup>
  </CreateParameterGroupResult>
  <ResponseMetadata>
    <RequestId>d8465952-af48-11e0-8d36-859edca6f4b8</RequestId>
  </ResponseMetadata>
</CreateParameterGroupResponse>
```

创建参数组后，它将具有系列的默认值。要更改默认值，您必须修改参数组。有关更多信息，请参阅修改参数组。

有关更多信息，请参阅 CreateParameterGroup。

按名称列出参数组

您可以使用 MemoryDB 控制台AWS CLI、或 MemoryDB API 列出参数组。

按名称列出参数组（控制台）

以下过程介绍了如何使用 MemoryDB 控制台查看参数组列表。

使用 MemoryDB 控制台列出参数组

1. 登录AWS 管理控制台并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 要查看所有可用的参数组列表，请在导航窗格左侧选择Parameter Groups。

按名称列出参数组 (AWSCLI)

要使用生成参数组列表AWS CLI，请使用命令`describe-parameter-groups`。如果提供了参数组的名称，将只会列出该参数组。如果未提供参数组的名称，将列出最多 `--max-results` 个参数组。在任一情况下，都会列出参数组的名称、系列和描述。

Example

以下示例代码列出了参数组 myRedis6x。

对于 Linux、macOS 或 Unix：

```
aws memorydb describe-parameter-groups \
    --parameter-group-name myRedis6x
```

对于 Windows：

```
aws memorydb describe-parameter-groups ^
    --parameter-group-name myRedis6x
```

该命令的输出内容将类似如下所示，列出参数组的名称、系列和描述。

```
{
    "ParameterGroups": [
     {
         "Name": "myRedis6x",
```

```
        "Family": "memorydb_redis6",
        "Description": "My first parameter group",
              "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/
myredis6x"
     }
    ]
}
```

Example

以下示例代码列出了在 Valkey 或 Redis OSS 引擎 5.0.6 和更高版本上运行的参数组 myRedis6x。

对于 Linux、macOS 或 Unix：

```
aws memorydb describe-parameter-groups \
    --parameter-group-name myRedis6x
```

对于 Windows：

```
aws memorydb describe-parameter-groups ^
    --parameter-group-name myRedis6x
```

该命令的输出内容将类似如下所示，列出参数组的名称、系列和描述。

```
{
    "ParameterGroups": [
     {
        "Name": "myRedis6x",
        "Family": "memorydb_redis6",
        "Description": "My first parameter group",
              "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/
myredis6x"
     }
    ]
}
```

Example

以下示例代码列出最多 20 个参数组。

```
aws memorydb describe-parameter-groups --max-results 20
```

该命令的 JSON 输出内容将类似如下所示，列出每个参数组的名称、系列和描述。

```
{
    "ParameterGroups": [
        {
            "ParameterGroupName": "default.memorydb-redis6",
            "Family": "memorydb_redis6",
            "Description": "Default parameter group for memorydb_redis6",
            "ARN": "arn:aws:memorydb:us-east-1:012345678912:parametergroup/
default.memorydb-redis6"
        },
        ...
    ]
}
```

有关更多信息，请参阅 describe-parameter-groups。

按名称列出参数组（MemoryDB API）

要使用 MemoryDB API 生成参数组的列表，请使用 DescribeParameterGroups 操作。如果提供了参数组的名称，将只会列出该参数组。如果未提供参数组的名称，将列出最多 MaxResults 个参数组。在任一情况下，都会列出参数组的名称、系列和描述。

Example

以下示例代码列出最多 20 个参数组。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeParameterGroups
   &MaxResults=20
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

来自此操作的响应应类似如下所示，列出每个参数组 memorydb_redis6 时的名称、系列和描述。

```
<DescribeParameterGroupsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
  <DescribeParameterGroupsResult>
    <ParameterGroups>
```

```
      <ParameterGroup>
        <Name>myRedis6x</Name>
        <Family>memorydb_redis6</Family>
        <Description>My custom Redis OSS 6 parameter group</Description>
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>
      </ParameterGroup>
       <ParameterGroup>
        <Name>default.memorydb-redis6</Name>
        <Family>memorydb_redis6</Family>
        <Description>Default parameter group for memorydb_redis6</Description>
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/default.memorydb-
redis6</ARN>
      </ParameterGroup>
    </ParameterGroups>
  </DescribeParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeParameterGroupsResponse>
```

Example

以下示例代码列出了参数组 myRedis6x。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeParameterGroups
   &ParameterGroupName=myRedis6x
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

来自此操作的响应将类似如下所示，列出名称、系列和描述。

```
<DescribeParameterGroupsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
  <DescribeParameterGroupsResult>
    <ParameterGroups>
      <ParameterGroup>
        <Name>myRedis6x</Name>
        <Family>memorydb_redis6</Family>
        <Description>My custom Redis OSS 6 parameter group</Description>
```

```
        <ARN>arn:aws:memorydb:us-east-1:012345678912:parametergroup/myredis6x</ARN>
      </ParameterGroup>
    </ParameterGroups>
  </DescribeParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeParameterGroupsResponse>
```

有关更多信息，请参阅 DescribeParameterGroups。

列出参数组的值

您可以使用 MemoryDB 控制台、或 MemoryDB API 列出参数组的AWS CLI参数及其值。

列出参数组的值（控制台）

以下过程介绍了如何使用 MemoryDB 控制台列出参数组的参数及其值。

使用 MemoryDB 控制台列出参数组的参数及其值

1. 登录AWS 管理控制台并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 要查看所有可用的参数组列表，请在导航窗格左侧选择Parameter Groups。
3. 通过选择参数组名称的名称（而不是名称旁边的框）来选择要列出其中包含的参数及其值的参数组。

   屏幕底部将列出这些参数及其值。由于参数的数量，您可能需要上下滚动来查找所需的参数。

列出参数组的值 (AWSCLI)

要使用列出参数组的参数及其值AWS CLI，请使用命令`describe-parameters`。

Example

以下示例代码列出了参数组 myRedis6x 的所有参数及其值。

对于 Linux、macOS 或 Unix：

```
aws memorydb describe-parameters \
    --parameter-group-name myRedis6x
```

对于 Windows：

```
aws memorydb describe-parameters ^
    --parameter-group-name myRedis6x
```

有关更多信息，请参阅 `describe-parameters`。

列出参数组的值（MemoryDB API）

要使用 MemoryDB API 列出参数组的参数及其值，请使用 `DescribeParameters` 操作。

有关更多信息，请参阅 DescribeParameters。

修改参数组

> **Important**
>
> 您无法修改任何默认参数组。

您可以修改参数组中的某些参数值。这些参数值应用于与参数组关联的集群。有关参数值更改何时应用于参数组的更多信息，请参阅引擎特定参数。

修改参数组（控制台）

以下过程介绍了如何使用 MemoryDB 控制台更改参数值。您可以使用相同的过程来更改任意参数的值。

使用 MemoryDB 控制台更改参数值

1. 登录AWS 管理控制台并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 要查看所有可用的参数组列表，请在导航窗格左侧选择Parameter Groups。
3. 通过选择参数组名称左侧的单选按钮来选择要修改的参数组。

   选择操作，然后选择查看详细信息。或者选择参数组名称以转到详细信息页面。
4. 要修改参数，请选择编辑。所有可编辑的参数都将启用编辑功能。您可能需要跨页移动才能找到待更改参数。或者在搜索框中按名称、值或类型搜索参数。
5. 对参数进行任何必要的修改。
6. 要保存您的更改，请选择保存更改。
7. 如果您修改了跨页显示的参数值，则可以通过选择预览更改来查看所有更改。要确认更改，请选择保存更改。要进行更多修改，请选择返回。
8. 参数详细信息页面还允许您选择重置为默认值。要重置为默认值，请选择重置为默认值。复选框将出现在所有参数的左侧。您可以选择待重置项，然后选择进行重置以确认。

   选择确认以确认对话框中的重置操作。
9. 参数详细信息页面允许您设置要每页可查看的参数数量。使用右侧的齿轮进行这些更改。您还 enable/disable 可以在详细信息页面上添加所需的列。这些更改将持续到控制台的会话中。

要查找您更改的参数名称，请参阅引擎特定参数。

修改参数组 (AWSCLI)

要使用更改参数的值AWS CLI，请使用命令`update-parameter-group`。

要查找您要更改的参数名称和允许的值，请参阅引擎特定参数

有关更多信息，请参阅 update-parameter-group。

修改参数组（MemoryDB API）

要使用 MemoryDB API 更改参数组的参数值，请使用 `UpdateParameterGroup` 操作。

要查找您要更改的参数名称和允许的值，请参阅引擎特定参数

有关更多信息，请参阅 `UpdateParameterGroup`。

删除参数组

您可以使用 MemoryDB 控制台、或 MemoryDB API AWS CLI 删除自定义参数组。

如果参数组与任何集群关联，则无法将其删除。也无法删除任一默认参数组。

删除参数组（控制台）

以下过程介绍了如何使用 MemoryDB 控制台删除参数组。

使用 MemoryDB 控制台删除参数组

1. 登录AWS 管理控制台并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 要查看所有可用的参数组列表，请在导航窗格左侧选择Parameter Groups。
3. 通过选择参数组名称左侧的单选按钮来选择要删除的参数组。

   选择 操作，然后选择 删除。
4. Delete Parameter Groups 确认屏幕随即出现。
5. 要删除参数组，请在确认文本框中输入删除。

   要保留参数组，请选择 Cancel。

删除参数组 (AWSCLI)

要使用删除参数组AWS CLI，请使用命令`delete-parameter-group`。对于要删除的参数组，由 `--parameter-group-name` 指定的参数组不能具有与之关联的任何集群，也不能是默认参数组。

以下示例代码删除 myRedis6x 参数组。

Example

对于 Linux、macOS 或 Unix：

```
aws memorydb delete-parameter-group \
    --parameter-group-name myRedis6x
```

对于 Windows：

```
aws memorydb delete-parameter-group ^
```

```
    --parameter-group-name myRedis6x
```

有关更多信息，请参阅 delete-parameter-group。

删除参数组（MemoryDB API）

要使用 MemoryDB API 删除参数组，请使用 DeleteParameterGroup 操作。对于要删除的参数组，由 ParameterGroupName 指定的参数组不能具有与之关联的任何集群，也不能是默认参数组。

Example

以下示例代码删除 myRedis6x 参数组。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DeleteParameterGroup
   &ParameterGroupName=myRedis6x
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 DeleteParameterGroup。

引擎特定参数

如果您没有为 Valkey 或 Redis OSS 集群指定参数组，则将使用适合您引擎版本的默认参数组。您无法更改默认参数组中的任何参数的值。但是，您可以随时创建自定义参数组并将其分配给集群，只要可按条件修改的参数的值在两个参数组中相同。有关更多信息，请参阅 创建参数组。

主题

- Valkey 7 和 Redis OSS 7 参数更改
- Redis OSS 6 参数
- MemoryDB 节点类型特定的参数

Valkey 7 和 Redis OSS 7 参数更改

> **Note**
>
> MemoryDB 引入了向量搜索，其中包括一个新的不可变参数组 `default.memorydb-valkey7.search`。此参数组在 MemoryDB 控制台中可用，也可以在使用 create -cluster CLI 命令创建新 vector-search-enabled集群时使用。该预览版在以下AWS地区推出：美国东部（弗吉尼亚北部）、美国东部（俄亥俄州）、美国西部（俄勒冈）、亚太地区（东京）和欧洲（爱尔兰）。

参数组系列：memorydb_valkey7

Valkey 7 和 Redis OSS 7 中增加的参数如下所示。

Name	Details	说明
`latency-tracking`	允许的值：`yes`、`no` 默认值：no 类型：字符串 可修改：是	设置为“yes”（是）时，将跟踪每个命令的延迟，并允许通过 `INFO` 延迟统计命令导出百分位数分布，并通过 `LATENCY` 命令导出累积延迟分布（直方图）。

Name	Details	说明
	更改生效：立即跨集群中的所有节点生效。	
hash-max-listpack-entries	允许的值：0+ 默认值：512 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	压缩数据集所需的最大哈希条目数。
hash-max-listpack-value	允许的值：0+ 默认值：64 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	压缩数据集所需的最大哈希条目数的阈值。
zset-max-listpack-entries	允许的值：0+ 默认值：128 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	压缩数据集所需的最大已排序集合条目数。

Name	Details	说明
zset-max-listpack-value	允许的值：0+ 默认值：64 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	压缩数据集所需的最大已排序集合条目数的阈值。
search-enabled	允许的值：yes, no 默认值：no 类型：字符串 可修改：是 更改生效：仅适用于新集群。 最低引擎版本：7.1	如果设置为“是”，则会启用搜索功能。
search-query-timeout-ms	允许的值：1 - 60,000 默认值：10,000 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。 最低引擎版本：7.1	允许搜索查询运行的最长时间（以毫秒为单位）。

Redis OSS 7 中更改的参数如下所示。

Name	Details	说明
activerehashing	可修改：no。在 Redis OSS 7 中，默认情况下，此参数处于隐藏和已启用状态。为了禁用此参数，您需要创建一个支持案例。	可修改：是。

Redis OSS 7 中删除的参数如下所示。

Name	Details	说明
hash-max-ziplist-entries	允许的值：0+ 默认值：512 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	使用 listpack 而非 ziplist 来表示小哈希编码
hash-max-ziplist-value	允许的值：0+ 默认值：64 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	使用 listpack 而非 ziplist 来表示小哈希编码
zset-max-ziplist-entries	允许的值：0+ 默认值：128	使用 listpack 而非 ziplist 来表示小哈希编码。

Name	Details	说明
	类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	
zset-max-ziplist-value	允许的值：0+ 默认值：64 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	使用 listpack 而非 ziplist 来表示小哈希编码。

Redis OSS 6 参数

> **Note**
>
> 在 Redis OSS 引擎 6.2 版中，在引入 r6gd 节点系列以支持 数据分层 功能时，r6gd 节点类型仅支持 noeviction、volatile-lru 和 allkeys-lru max-memory 策略。

参数组系列： memorydb_redis6

Redis OSS 6 中增加的参数如下所示。

Name	Details	说明
maxmemory-policy	类型：字符串 允许的值：volatile-lru,allkeys-lru,volatile-lfu,allkeys-lfu	达到最大内存使用率时密钥的移出策略。 有关将 Valkey 或 Redis OSS 用作 LRU 缓存的更多信息，请参阅键驱逐。

Name	Details	说明
	,volatile-random,allkeys-random,volatile-ttl,noeviction 默认值：noeviction	
`list-compress-depth`	类型：整数 允许的值：0- 默认：0	压缩深度是要从压缩中排除的列表各端的 quicklist ziplist 节点的数目。始终不会压缩列表的首尾以便执行快速推送和弹出操作。设置为： • 0：禁止所有压缩。 • 1：从首尾开始将第一个节点压缩到列表中。 [head]->node->node->...->node->[tail] 压缩除 [head] 和 [tail] 以外的所有节点。 • 2：从首尾开始将第二个节点压缩到列表中。 [head]->[next]->node->node->...->node->[prev]->[tail] [head]、[next]、[prev]、[tail] 不压缩。压缩所有其他节点。 • 等等

Name	Details	说明
hll-sparse-max-bytes	类型：整数 允许的值：1-16000 默认值：3000	HyperLogLog 稀疏表示字节限制。限制包括 16 个字节的标头。当 HyperLogLog 使用稀疏表示超过此限制时，它将转换为密集表示。 不建议使用超过 16000 的值，因为此时密集表现形式具有更高的内存效率。 我们建议使用 3000 左右的值来获得空间效率较高的编码，同时不会过多地降低 PFADD 效率，即稀疏编码的复杂度为 O（N）。当不考虑CPU，而是空间时，该值可以提高到大约 10000，并且数据集由许多 HyperLogLogs 基数在 0-15000 范围内的数据集组成。
lfu-log-factor	类型：整数 允许的值：1- 默认值：10	LFU 驱逐策略递增密钥计数器的日志因数。
lfu-decay-time	类型：整数 允许的值：0- 默认：1	减少 LFU 驱逐策略的键计数器的时间，以分钟为单位。
active-defrag-max-scan-fields	类型：整数 允许的值：1-1000000 默认值：1000	在主动碎片整理期间，将从主字典扫描中处理的最大set/hash/zset/list字段数。

Name	Details	说明
active-defrag-threshold-upper	类型：整数 允许的值：1-100 默认值：100	我们使用最大精力的碎片最高百分比。
client-output-buffer-limit-pubsub-hard-limit	类型：整数 允许的值：0- 默认值：33554432	对于 Redis OSS publish/subscribe 客户端：如果客户端的输出缓冲区达到指定的字节数，则客户端将断开连接。
client-output-buffer-limit-pubsub-soft-limit	类型：整数 允许的值：0- 默认值：8388608	对于 Redis OSS publish/subscribe 客户端：如果客户端的输出缓冲区达到指定的字节数，则客户端将断开连接，但前提是这种情况持续存在 client-output-buffer-limit-pubsub-soft-seconds.
client-output-buffer-limit-pubsub-soft-seconds	类型：整数 允许的值：0- 默认值：60	对于 Redis OSS publish/subscribe 客户端：如果客户端的输出缓冲区保持client-output-buffer-limit-pubsub-soft-limit 字节的时间超过此秒数，则客户端将断开连接。
timeout	类型：整数 允许的值：0,20- 默认：0	节点在超时之前等待的秒数。值为： • 0 – 从不断开空闲客户端。 • 1-19 – 无效值。 • >=20 – 节点在断开空闲客户端之前等待的秒数。

Name	Details	说明
notify-keyspace-events	类型：字符串 允许的值：NULL 默认值：NULL	Redis OSS 要通知 Pub/Sub 客户端的密钥空间事件。默认情况下，所有通知处于禁用状态。
maxmemory-samples	类型：整数 允许的值：1- 原定设置值：3	对于 least-recently-used(LRU)和time-to-live （TTL） 计算，此参数表示要检查的密钥的样本量。默认情况下，Redis OSS 选择 3 个键并使用最近最少使用的一个键。
slowlog-max-len	类型：整数 允许的值：0- 默认值：128	Redis OSS 慢速日志的最大长度。此长度没有限制。请注意，它会消耗内存。您可以通过 SLOWLOG RESET. 回收慢日志使用的内存
activerehashing	类型：字符串 允许的值：yes、no 默认值：yes	主哈希表每秒重新哈希十次；每个重新哈希操作消耗 1 毫秒的 CPU 时间。 在创建参数组时设置此值。向集群分配新参数组时，此值在旧参数组和新参数组中必须相同。
client-output-buffer-limit-normal-hard-limit	类型：整数 允许的值：0- 默认：0	如果客户端的输出缓冲区达到指定字节数，则客户端将断开连接。默认值为零（没有硬限制）。
client-output-buffer-limit-normal-soft-limit	类型：整数 允许的值：0- 默认：0	如果客户端的输出缓冲区达到指定字节数，则客户端将断开连接，但是仅当此条件保持 client-output-buffer-limit-normal-soft-seconds 时间时。默认值为零（没有软限制）。

Name	Details	说明
client-output-buffer-limit-normal-soft-seconds	类型：整数 允许的值：0- 默认：0	如果客户端的输出缓冲区保持 client-output-buffer-limit-normal-soft-limit 字节的时间长于此秒数，则客户端将断开连接。默认值为零（没有时间限制）。
tcp-keepalive	类型：整数 允许的值：0- 默认：300	如果此参数设置为非零值（N），则节点客户端会每 N 秒轮询一次，以确保它们仍然连接。对于默认设置 0，不进行这种轮询。
active-defrag-cycle-min	类型：整数 允许的值：1-75 默认：5	用于碎片整理的最少精力，以 CPU 百分比为单位。
stream-node-max-bytes	类型：整数 允许的值：0- 默认值：4096	流数据结构是节点的基数树，这些节点对内部的多个项进行编码。使用此配置指定基数树中单个节点的最大大小（以字节为单位）。如果设置为 0，则树节点的大小是不受限制的。
stream-node-max-entries	类型：整数 允许的值：0- 默认值：100	流数据结构是节点的基数树，这些节点对内部的多个项进行编码。使用此配置指定在追加新的流条目时切换到新节点之前单个节点可包含的项的最大数目。如果设置为 0，则树节点中的项数是不受限制的。

Name	Details	说明
lazyfree-lazy-eviction	类型：字符串 允许的值：yes、no 默认值：no	对移出执行异步删除。
active-defrag-ignore-bytes	类型：整数 允许的值：1048576- 默认值：104857600	启动有效碎片整理的碎片垃圾最低量。
lazyfree-lazy-expire	类型：字符串 允许的值：yes、no 默认值：no	对已过期密钥执行异步删除。
active-defrag-threshold-lower	类型：整数 允许的值：1-100 默认值：10	启动有效碎片整理的碎片最低百分比。
active-defrag-cycle-max	类型：整数 允许的值：1-75 默认值：75	用于碎片整理的最多精力，以 CPU 百分比为单位。
lazyfree-lazy-server-del	类型：字符串 允许的值：yes、no 默认值：no	对更新值的命令执行异步删除。

Name	Details	说明
slowlog-log-slower-than	类型：整数 允许的值：0- 默认值：10000	Redis OSS Slow Log 特征记录的命令的最长执行时间（单位：微秒）。请注意，负数表示禁用慢日志，而值为零则表示强制记录每个命令。
hash-max-ziplist-entries	类型：整数 允许的值：0- 默认值：512	确定用于哈希的内存量。条目少于指定数量的哈希使用节省空间的特殊编码进行存储。
hash-max-ziplist-value	类型：整数 允许的值：0- 默认值：64	确定用于哈希的内存量。条目小于指定字节数的哈希使用节省空间的特殊编码进行存储。
set-max-intset-entries	类型：整数 允许的值：0- 默认值：512	确定用于特定类型的集（在 64 位有符号整数的范围内，以 10 为基数的整数表示的字符串）的内存量。条目少于指定数量的这类集使用节省空间的特殊编码进行存储。
zset-max-ziplist-entries	类型：整数 允许的值：0- 默认值：128	确定用于排序集的内存量。元素少于指定数量的排序集使用节省空间的特殊编码进行存储。
zset-max-ziplist-value	类型：整数 允许的值：0- 默认值：64	确定用于排序集的内存量。条目小于指定字节数的排序集使用节省空间的特殊编码进行存储。

Name	Details	说明
tracking-table-max-keys	类型：整数 允许的值：1-100000000 默认值：1000000	为了帮助客户端缓存，Redis OSS 支持跟踪哪些客户端访问了哪些键。 当所跟踪的密钥被修改后，会向所有客户端发送失效消息，通知它们缓存的值不再有效。此值允许您指定此表的上限。
acllog-max-len	类型：整数 允许的值：1-10000 默认值：128	ACL 日志的最大条目数。
active-expire-effort	类型：整数 允许的值：1-10 默认：1	Redis OSS 会通过两种机制删除超过键自身存活时间的键。一种机制是，访问密钥并发现其已过期。另一种机制是，周期性任务对密钥进行采样，并使那些超过其存活时间的密钥过期。此参数定义 Redis OSS 用于在周期性任务中使项目过期的工作量。 默认值 1 用于避免 10% 以上的过期密钥仍存在于内存中。其还用于避免 25% 以上的总内存被消耗及增加系统的延迟。您可以将此值增加到 10，以提高用在过期密钥上的工作量。需要权衡的是，当 CPU 更高时，延迟也可能会更高。我们建议将值设为 1，除非您发现内存使用率较高，并且可以容忍 CPU 使用率升高。
lazyfree-lazy-user-del	类型：字符串 允许的值：yes、no 默认值：no	指定 DEL 命令的默认行为是否与 UNLINK 相同。

Name	Details	说明
activedefrag	类型：字符串 允许的值：yes、no 默认值：no	已启用有效的内存碎片整理。
maxclients	类型：整数 允许的值：65000 默认值：65000	可以一次连接的最大客户端连接数。不可修改。
client-query-buffer-limit	类型：整数 允许的值：1048576-1073741824 默认值：1073741824	单个客户端查询缓冲区的最大大小。立即发生更改。
proto-max-bulk-len	类型：整数 允许的值：1048576-536870912 默认值：536870912	单个元素请求的最大大小。立即发生更改。

MemoryDB 节点类型特定的参数

虽然大多数参数具有单个值，但是某些参数根据使用的节点类型具有不同的值。下表显示了每种节点类型的 maxmemory 的默认值。maxmemory 的值是节点上可供您使用（数据和其他用途）的最大字节数。

节点类型	Maxmemory
db.r7g.large	14037181030

节点类型	Maxmemory
db.r7g.xlarge	28261849702
db.r7g.2xlarge	56711183565
db.r7g.4xlarge	113609865216
db.r7g.8xlarge	225000375228
db.r7g.12xlarge	341206346547
db.r7g.16xlarge	450000750456
db.r6gd.xlarge	28261849702
db.r6gd.2xlarge	56711183565
db.r6gd.4xlarge	113609865216
db.r6gd.8xlarge	225000375228
db.r6g.large	14037181030
db.r6g.xlarge	28261849702
db.r6g.2xlarge	56711183565
db.r6g.4xlarge	113609865216
db.r6g.8xlarge	225000375228
db.r6g.12xlarge	341206346547
db.r6g.16xlarge	450000750456
db.t4g.small	1471026299
db.t4g.medium	3317862236

> **Note**
>
> 所有 MemoryDB 实例类型必须在 Amazon 虚拟私有云（VPC）中创建。

受限命令

为了提供托管服务体验，MemoryDB 限制了对某些需要高级特权的特定命令的访问。以下命令不可用：

- `acl deluser`
- `acl load`
- `acl save`
- `acl setuser`
- `bgrewriteaof`
- `bgsave`
- `cluster addslot`
- `cluster delslot`
- `cluster setslot`
- `config`
- `debug`
- `migrate`
- `module`
- `psync`
- `replicaof`
- `save`
- `shutdown`
- `slaveof`
- `sync`

教程：配置 Lambda 函数，以便在 Amazon VPC 中访问 MemoryDB

在本教程中，您可以学习如何：

- 在 us-east-1 区域中的默认 Amazon Virtual Private Cloud（Amazon VPC）中创建 MemoryDB 集群。
- 创建 Lambda 函数以访问集群。创建 Lambda 函数时，您需要在您的 Amazon VPC IDs 中提供子网和一个 VPC 安全组，以允许 Lambda 函数访问您的 VPC 中的资源。在本教程的图示中，Lambda 函数生成 UUID，将其写入到集群，然后从集群中检索。
- 手动调用 Lambda 函数，并确认它访问了您的 VPC 中的集群。
- 清理为本教程设置的 Lambda 函数、集群和 IAM 角色。

主题

- 步骤 1：创建集群
- 第 2 步：创建 Lambda 函数
- 步骤 3：测试 Lambda 函数
- 步骤 4：清除（可选）

步骤 1：创建集群

要创建集群，请执行以下步骤。

创建集群

在此步骤中，您将使用 (CLI) 在账户的 us-east-1 区域的默认 Amazon VPC 中创建一个集群。 AWS Command Line Interface 有关使用 MemoryDB 控制台或 API 创建集群的信息，请参阅步骤 2：创建集群。

```
aws memorydb create-cluster --cluster-name cluster-01 --engine-version 7.0 --acl-name
 open-access \
--description "MemoryDB IAM auth application" \
--node-type db.r6g.large
```

请注意，“状态”字段的值设置为 CREATING。MemoryDB 需要几分钟时间来完成集群的创建。

复制集群端点

使用 `describe-clusters` 命令确认 MemoryDB 已完成集群的创建。

```
aws memorydb describe-clusters \
```

```
--cluster-name cluster-01
```

复制输出中显示的集群端点地址。在为 Lambda 函数创建部署包时，您将需要此地址。

创建 IAM 角色

1. 为您的角色创建 IAM 信任政策文档，如下所示，允许您的账户承担新角色。将策略保存到名为 trust-policy.json 的文件中。请务必将本策略中的 account_id 123456789012 替换为您的 account_id。

   JSON

   ```
   {
   "Version":"2012-10-17",
       "Statement": [{
           "Effect": "Allow",
           "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
           "Action": "sts:AssumeRole"
       },
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "lambda.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }]
   }
   ```

2. 创建 IAM 策略文档，如下所示。将策略保存到名为 policy.json 的文件中。请务必将本策略中的 account_id 123456789012 替换为您的 account_id。

   JSON

   ```
   {
   "Version":"2012-10-17",
     "Statement": [
       {
         "Effect" : "Allow",
         "Action" : [
           "memorydb:Connect"
   ```

```
      ],
      "Resource" : [
        "arn:aws:memorydb:us-east-1:123456789012:cluster/cluster-01",
        "arn:aws:memorydb:us-east-1:123456789012:user/iam-user-01"
      ]
    }
  ]
}
```

3. 创建 IAM 角色。

```
aws iam create-role \
--role-name "memorydb-iam-auth-app" \
--assume-role-policy-document file://trust-policy.json
```

4. 创建 IAM 策略。

```
aws iam create-policy \
  --policy-name "memorydb-allow-all" \
  --policy-document file://policy.json
```

5. 向角色附加 IAM 策略。请务必将这个 policy-arn 中的 account_id 123456789012 替换为您的 account_id。

```
aws iam attach-role-policy \
 --role-name "memorydb-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"
```

创建访问控制列表（ACL）

1. 创建启用 IAM 的新用户。

```
aws memorydb create-user \
  --user-name iam-user-01 \
--authentication-mode Type=iam \
--access-string "on ~* +@all"
```

2. 创建 ACL 并将其附加到集群中。

```
aws memorydb create-acl \
  --acl-name iam-acl-01 \
```

```
  --user-names iam-user-01

aws memorydb update-cluster \
  --cluster-name cluster-01 \
  --acl-name iam-acl-01
```

第 2 步：创建 Lambda 函数

要创建 Lambda 函数，请执行以下步骤。

创建部署程序包

在本教程中，我们为您的 Lambda 函数提供了 Python 中的示例代码。

Python

以下示例 Python 代码在 MemoryDB 集群中读取和写入项目。复制代码，并将其保存到名为 app.py 的文件中。请确保将代码中的 cluster_endpoint 值替换为您在之前的步骤中复制的端点地址。

```
from typing import Tuple, Union
from urllib.parse import ParseResult, urlencode, urlunparse

import botocore.session
import redis
from botocore.model import ServiceId
from botocore.signers import RequestSigner
from cachetools import TTLCache, cached
import uuid

class MemoryDBIAMProvider(redis.CredentialProvider):
    def __init__(self, user, cluster_name, region="us-east-1"):
        self.user = user
        self.cluster_name = cluster_name
        self.region = region

        session = botocore.session.get_session()
        self.request_signer = RequestSigner(
            ServiceId("memorydb"),
            self.region,
            "memorydb",
            "v4",
            session.get_credentials(),
```

```
            session.get_component("event_emitter"),
        )

    # Generated IAM tokens are valid for 15 minutes
    @cached(cache=TTLCache(maxsize=128, ttl=900))
    def get_credentials(self) -> Union[Tuple[str], Tuple[str, str]]:
        query_params = {"Action": "connect", "User": self.user}

        url = urlunparse(
            ParseResult(
                scheme="https",
                netloc=self.cluster_name,
                path="/",
                query=urlencode(query_params),
                params="",
                fragment="",
            )
        )
        signed_url = self.request_signer.generate_presigned_url(
            {"method": "GET", "url": url, "body": {}, "headers": {}, "context": {}},
            operation_name="connect",
            expires_in=900,
            region_name=self.region,
        )
        # RequestSigner only seems to work if the URL has a protocol, but
        # MemoryDB only accepts the URL without a protocol
        # So strip it off the signed URL before returning
        return (self.user, signed_url.removeprefix("https://"))

def lambda_handler(event, context):
    username = "iam-user-01" # replace with your user id
    cluster_name = "cluster-01" # replace with your cache name
    cluster_endpoint = "clustercfg.cluster-01.xxxxxx.memorydb.us-east-1.amazonaws.com"
 # replace with your cluster endpoint
    creds_provider = MemoryDBIAMProvider(user=username, cluster_name=cluster_name)
    redis_client = redis.Redis(host=cluster_endpoint, port=6379,
 credential_provider=creds_provider, ssl=True, ssl_cert_reqs="none")

    key='uuid'
    # create a random UUID - this will be the sample element we add to the cluster
    uuid_in = uuid.uuid4().hex
    redis_client.set(key, uuid_in)
    result = redis_client.get(key)
    decoded_result = result.decode("utf-8")
```

```
    # check the retrieved item matches the item added to the cluster and print
    # the results
    if decoded_result == uuid_in:
        print(f"Success: Inserted {uuid_in}. Fetched {decoded_result} from MemoryDB.")
    else:
        raise Exception(f"Bad value retrieved. Expected {uuid_in}, got
 {decoded_result}")

    return "Fetched value from MemoryDB"
```

此代码使用 Python redis-py 库将项目放入集群并检索它们。此代码使用 cachetools 将生成的 IAM 身份验证令牌缓存 15 分钟。要创建包含 redis-py 和 cachetools 的部署包，请执行以下步骤。

在包含 app.py 源代码文件的项目目录中，创建一个文件夹包，用于在其中安装 redis-py 和 cachetools 库。

```
mkdir package
```

使用 pip 安装 redis-py 和 cachetools。

```
pip install --target ./package redis
pip install --target ./package cachetools
```

创建包含 redis-py 和 cachetools 库的 .zip 文件。在 Linux 和 MacOS 中，运行以下命令。在 Windows 中，使用首选 zip 实用工具创建一个 .zip 文件，并将 redis-py 和 cachetools 库置于根目录下。

```
cd package
zip -r ../my_deployment_package.zip .
```

将您的函数代码添加到 .zip 文件。在 Linux 和 macOS 中，运行以下命令：在 Windows 中，使用首选 zip 实用工具将 app.py 添加到 .zip 文件的根目录下。

```
cd ..
zip my_deployment_package.zip app.py
```

创建 IAM 角色（执行角色）

将名为的 AWS 托管策略附加AWSLambdaVPCAccessExecutionRole到角色。

```
aws iam attach-role-policy \
 --role-name "memorydb-iam-auth-app" \
 --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"
```

上传部署包（创建 Lambda 函数）

在此步骤中，您将使用创建函数命令创建 Lambda 函数 (AccessMemoryDB)。 AWS CLI

在包含您的部署包 .zip 文件的项目目录中，运行以下 Lambda CLI `create-function` 命令。

对于角色选项，请使用您在上一步中创建的执行角色的 ARN。对于 vpc-config，请输入默认 VPC 子网的列表和默认 VPC 安全组 ID 的列表，以逗号分隔。您还可以在 Amazon VPC 控制台中找到这些值。要查找您的默认 VPC 子网，请选择您的 VPCs，然后选择您 AWS 账户的默认 VPC。要查找此 VPC 的安全组，请转到安全，然后选择安全组。请确保您选择了 us-east-1 区域。

```
aws lambda create-function \
--function-name AccessMemoryDB  \
--region us-east-1 \
--zip-file fileb://my_deployment_package.zip \
--role arn:aws:iam::123456789012:role/memorydb-iam-auth-app \
--handler app.lambda_handler \
--runtime python3.12  \
--timeout 30 \
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-
security-group-id
```

步骤 3：测试 Lambda 函数

在此步骤中，您将使用调用命令手动调用 Lambda 函数。当 Lambda 函数执行时，它会生成一个 UUID 并将其写入您在 Lambda 代码中指定的 ElastiCache 缓存中。然后，Lambda 函数将从缓存中检索项目。

1. 使用调用命令 AWS Lambda 调用 Lambda 函数 (AccessMemoryDB)。

   ```
   aws lambda invoke \
   --function-name AccessMemoryDB  \
   --region us-east-1 \
   output.txt
   ```

2. 按以下过程验证 Lambda 函数是否已成功执行：

- 查看 output.txt 文件。
- 打开 CloudWatch 控制台并选择函数的日志组 (/aws/lambda/AccessRedis)，验证日志中的 CloudWatch 结果。日志流应包含类似于以下内容的输出：

```
Success: Inserted 826e70c5f4d2478c8c18027125a3e01e. Fetched
 826e70c5f4d2478c8c18027125a3e01e from MemoryDB.
```

- 在 AWS Lambda 控制台中查看结果。

步骤 4：清除（可选）

要进行清理，请执行以下步骤。

删除 Lambda 函数

```
aws lambda delete-function \
 --function-name AccessMemoryDB
```

删除 MemoryDB 集群

请删除集群。

```
aws memorydb delete-cluster \
 --cluster-name cluster-01
```

移除用户和 ACL

```
aws memorydb delete-user \
 --user-id iam-user-01

aws memorydb delete-acl \
 --acl-name iam-acl-01
```

移除 IAM 角色和策略

```
aws iam detach-role-policy \
 --role-name "memorydb-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"
```

```
aws iam detach-role-policy \
--role-name "memorydb-iam-auth-app" \
--policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"

aws iam delete-role \
 --role-name "memorydb-iam-auth-app"

 aws iam delete-policy \
  --policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"
```

向量搜索

MemoryDB 向量搜索扩展了 MemoryDB 的功能。向量搜索可以与现有的 MemoryDB 功能结合使用。不使用向量搜索的应用程序不受此功能的影响。向量搜索在提供 MemoryDB 的所有区域均可用。

向量搜索可简化应用程序架构，同时提供高速向量搜索。适用于 MemoryDB 的向量搜索非常适合将峰值性能和规模作为最重要选择标准的用例。您可以使用现有 MemoryDB 数据，或 Valkey 或 Redis OSS API 来构建机器学习和生成式人工智能使用场景。这包括检索增强生成、异常检测、文献检索和实时推荐。

截至2024年6月26日，在流行的矢量 AWS 数据库中，MemoryDB以最高的召回率提供了最快的矢量搜索性能。 AWS

主题

- 向量搜索概述
- 使用案例
- 向量搜索功能和限制
- 创建已启用向量搜索的集群
- 向量搜索命令

向量搜索概述

向量搜索在索引创建、维护和使用的基础上建立。每个向量搜索操作都指定一个单一索引，其操作被限定于该索引，也就是说，对一个索引的操作不受任何其他索引操作的影响。除了创建和销毁索引的操作之外，可以随时对任何索引执行任意数量的操作，这意味着在集群级别，可以同时对多个索引执行多个操作。

各个索引是存在于独特命名空间中的命名对象，该命名空间与其他 Valkey 和 Redis OSS 命名空间（例如键、函数等）分开。从概念上讲，每个索引都类似于传统的数据库表，它的结构有两个维度：列和行。表中的每行对应一个键。索引中的每一列对应该键的一个成员或部分。在本文档中，术语“键”、“行”和“记录”的含义相同，可以互换使用。同样，术语“列”、“字段”、“路径”和“成员”的含义在本质上是相同的，可以互换使用。

没有用于添加、删除或修改索引数据的特殊命令。然而，使用现有的 HASH 或 JSON 命令对索引中的键进行修改也会自动更新索引。

主题

- 索引以及 Valkey 和 Redis OSS 键空间
- 索引字段类型
- 向量索引算法
- 向量搜索查询表达式
- INFO 命令
- 向量搜索安全

索引以及 Valkey 和 Redis OSS 键空间

在 Valkey 和 Redis OSS 键空间子集的基础上构造和维护索引。多个索引可以不受限制地选择不相交或不重叠的键空间子集。每个索引的键空间由创建索引时提供的键前缀列表定义。前缀列表是可选的，如果省略前缀列表，则整个键空间将成为该索引的一部分。索引还具有类型，并且仅涵盖具有匹配类型的键。当前，仅支持 JSON 和哈希索引。哈希索引仅为前缀列表所涵盖的哈希键编制索引，同样，JSON 索引仅为其前缀列表所涵盖的 JSON 键编制索引。在索引的键空间前缀列表中，未指定类型的键将被忽略，并且也不会影响搜索操作。

当 HASH 或 JSON 命令修改索引键空间内的键时，该索引就会更新。此过程包括提取每个索引的已声明字段，并使用新值更新索引。更新过程在后台线程中完成，这意味着索引可能需要过一段时间后，才能最终与其键空间内容保持一致。因此，插入或更新键后，可能短时间内不会出现在搜索结果中。在系统负载繁重的时期，数据发生 and/or 大量突变，可见性延迟可能会变得更长。

索引的创建是一个多步骤过程。第一步是执行定义索引的 FT.CREATE 命令。成功执行创建命令后会自动启动第二步：回填。回填过程在后台线程中运行，并会扫描键空间，查找位于新索引前缀列表中的键。找到的每个键都会添加到索引中。最终，整个键空间都会接受扫描，完成索引创建过程。请注意，在回填过程运行时，允许对索引键进行变更并且没有任何限制，索引回填过程只有在所有键都正确编制索引后才会完成。在索引回填时，不允许尝试查询操作，否则回填终止并显示错误。可以通过执行 `FT.INFO` 查看索引的输出（“backfill_status”）来确定回填过程的完成情况。

索引字段类型

索引的每个字段（列）都有创建索引时声明的特定类型，并且有一个键内位置。对于哈希键，位置是哈希中的字段名称。对于 JSON 键，位置是 JSON 路径描述。修改键时，系统会提取与已声明字段关联的数据，将其转换为声明的类型并存储在索引中。如果数据丢失或无法成功转换为声明的类型，则该字段将从索引中省略。有四种类型的字段，如下所述：

- 数值字段包含一个数字。对于 JSON 字段，必须遵守 JSON 数字规则。对于哈希，字段应包含以固定或浮点数标准格式编写的数值 ASCII 文本。无论键内的表示形式如何，字段都将转换为在索引中

存储的 64 位浮点数。数值字段可以与范围搜索运算符搭配使用。基础数字以浮点数形式存储并且有精度限制，因此适用于浮点数比较的常用规则对于数值字段也适用。

- 标签字段包含零个或多个标签值，编码为单个 UTF-8 字符串。字符串解析为使用分隔符（默认为英文逗号，但可更改）分隔的标签值，并删除前导和尾随空格。单个标签字段中可以包含任意数量的标签值。对于标签字段，可以在查询时对标签值进行筛选，并且可以选择区分大小写或不区分大小写。
- 文本字段包含一组字节，不一定符合 UTF-8 标准。对于文本字段，可使用与应用程序相关的值对查询结果进行修饰。例如，URL 或文档内容等。
- 向量字段包含一个数字向量，也称为嵌入。向量字段支持使用指定的算法和距离度量对固定大小的向量进行 K 近邻搜索（KNN）。对于哈希索引，该字段应包含以二进制格式（小端序 IEEE 754）编码的整个向量。对于 JSON 键，路径应引用大小正确且填充数字的数组。请注意，当将 JSON 数组用作向量字段时，JSON 键中数组的内部表示形式会转换为所选算法所需的格式，从而减少内存消耗和精度。使用 JSON 命令进行的后续读取操作所产生值的精度会降低。

向量索引算法

系统提供两种向量索引算法：

- Flat – Flat 算法是对索引中每个向量进行暴力线性处理，提供距离计算精度范围内的精确答案。由于索引采用线性处理，因此对于大型索引，此算法的运行时间可能非常长。
- HNSW（分层可导航小世界）– HNSW 算法是一种替代方案，它能够提供正确答案的近似值，从而极大程度缩短执行时间。该算法由 `M`、`EF_CONSTRUCTION` 和 `EF_RUNTIME` 三个参数控制。前两个参数在创建索引时指定，无法更改。`EF_RUNTIME` 参数的默认值在创建索引时指定，但之后可以在任何单独的查询操作中更改。这三个参数相互作用，在摄取和查询操作期间平衡内存和 CPU 消耗，并控制精确 KNN 搜索近似值的质量（称为查准率）。

两种向量搜索算法（Flat 和 HNSW）都支持可选的 `INITIAL_CAP` 参数。指定此参数时，它会为索引预先分配内存，从而减少内存管理开销并提高向量摄取速度。

类似于 HNSW 的向量搜索算法可能无法高效处理对先前插入向量的删除或覆盖。使用这些操作可能会导致索引内存消耗过多，从而 and/or 降低召回质量。重新索引是恢复最佳内存使用 and/or 回调的一种方法。

向量搜索查询表达式

FT.SEARCH 和 FT.AGGREGATE 命令需要查询表达式。该表达式是一个单字符串参数，由一个或多个运算符组成。每个运算符使用索引中的一个字段来标识索引中键的子集。可以使用布尔组合器和括号将多个运算符组合起来，以进一步增强或限制收集的键集合（或结果集）。

通配符

通配符运算符，即星号（“*”），可匹配索引中的所有键。

数值范围

数值范围运算符使用以下语法：

```
<range-search> ::= '@' <numeric-field-name> ':' '[' <bound> <bound> ']'
<bound>  ::= <number> | '(' <number>
<number> ::= <integer> | <fixed-point> | <floating-point> | 'Inf' | '-Inf' | '+Inf'
```

< numeric-field-name **>** 必须是声明的类型字段NUMERIC。默认情况下包含边界值，但可以使用前导左圆括号 [“(”] 来排除边界值。通过使用 Inf、+Inf 或 -Inf 作为边界之一，可以将范围搜索转换为单个关系比较（**<**、**<=**、**>**、**>=** ）。无论指定哪种数值格式（整数、固定点、浮点、无穷大），数字都会转换为 64 位浮点数进行比较，并相应地降低精度。

Example示例

```
@numeric-field:[0 10]                    // 0   <= <value> <= 10
@numeric-field:[(0 10]                   // 0   <  <value> <= 10
@numeric-field:[0 (10]                   // 0   <= <value> <  10
@numeric-field:[(0 (10]                  // 0   <  <value> <  10
@numeric-field:[1.5 (Inf]                // 1.5 <= value
```

标签比较

标签比较运算符使用以下语法：

```
<tag-search> ::= '@' <tag-field-name> ':' '{' <tag> [ '|' <tag> ]* '}'
```

如果运算符中的任何标签与记录的标签字段中的任何标签匹配，则该记录将包含在结果集中。采用 <tag-field-name> 设计的字段必须是用 TAG 类型声明的索引字段。标签比较的示例包括：

```
@tag-field:{ atag }
@tag-field: { tag1 | tag2 }
```

布尔值组合

可以使用布尔逻辑组合数字运算符或标签运算符的结果集：and/or. Parentheses can be used to group operators and/or更改计算顺序。布尔逻辑运算符的语法为：

```
<expression> ::= <phrase> | <phrase> '|' <expression> | '(' <expression> ')'
<phrase> ::= <term> | <term> <phrase>
<term> ::= <range-search> | <tag-search> | '*'
```

将多个术语组合成短语时使用“and”运算。使用竖线（“|”）组合多个短语时使用“or”运算。

向量搜索

向量索引支持两种不同的搜索方法：近邻和范围。近邻搜索可找到在索引中与提供的（参考）向量最接近的数量为 K 的向量，这通常称为 KNN，表示“K”最近邻。KNN 搜索的语法是：

```
<vector-knn-search> ::= <expression> '=>[KNN' <k> '@' <vector-field-name> '$'
 <parameter-name> <modifiers> ']'
<modifiers> ::= [ 'EF_RUNTIME' <integer> ] [ 'AS' <distance-field-name>]
```

向量 KNN 搜索仅适用于满足的向量，<expression>这些向量可以是上面定义的运算符的任意组合：通配符、范围搜索、标签搜索 and/or 布尔值组合。

- <k> 是一个整数，指定要返回的近邻向量的数量。
- <vector-field-name> 必须指定类型为 VECTOR 的已声明字段。
- <parameter-name> 字段指定 FT.SEARCH 或 FT.AGGREGATE 命令 PARAM 表中的一个条目。此参数是距离计算的参考向量值。向量的值以小端序 IEEE 754 二进制格式编码到 PARAM 值中（与哈希向量字段的编码相同）。
- 对于 HNSW 类型的向量索引，可以使用可选的 EF_RUNTIME 子句覆盖创建索引时建立的 EF_RUNTIME 参数的默认值。
- 可选的 <distance-field-name> 为结果集提供了一个字段名称，用于包含参考向量和定位键之间的计算距离。

范围搜索可找到距离参考向量指定距离（半径）内的所有向量。范围搜索的语法是：

```
<vector-range-search> ::= ‘@’ <vector-field-name> ‘:’ ‘[’ ‘VECTOR_RANGE’ ( <radius> |
 ‘$’ <radius-parameter> )  $<reference-vector-parameter> ‘]’ [ ‘=’ ‘>’ ‘{’ <modifiers>
 ‘}’ ]
<modifiers> ::= <modifier> | <modifiers>, <modifier>
<modifer> ::= [ ‘$yield_distance_as’ ‘:’ <distance-field-name> ] [ ‘$epsilon’ ‘:’
 <epsilon-value> ]
```

其中：

- <vector-field-name> 是要搜索的向量字段的名称。
- <radius> or $<radius-parameter> 是搜索的数字距离限制。
- $<reference-vector-parameter> 是包含参考向量的参数的名称。向量的值以小端序 IEEE 754 二进制格式编码到 PARAM 值中（与哈希向量字段的编码相同）。
- 可选的 <distance-field-name> 为结果集提供了一个字段名称，用于包含参考向量和各个键之间的计算距离。
- 可选的 <epsilon-value> 控制搜索操作的边界，遍历距离 <radius> * (1.0 + <epsilon-value>) 内的向量以寻找候选结果。默认值为 0.01。

INFO 命令

向量搜索对 Valkey 和 Redis OSS INFO 命令进行了补充，增加了几个统计数据和计数器部分。请求检索 SEARCH 部分将检索以下所有部分：

`search_memory` 部分

名称	说明
search_used_memory_bytes	所有搜索数据结构消耗的内存字节数
search_used_memory_human	上述内存消耗的人类可读版本

`search_index_stats` 部分

名称	说明
search_number_of_indexes	已创建的索引数量
search_num_fulltext_indexes	所有索引中非向量字段的数量
search_num_vector_indexes	所有索引中向量字段的数量
search_num_hash_indexes	哈希类型键索引数量
search_num_json_indexes	JSON 类型键索引数量
search_total_indexed_keys	所有索引键总数

名称	说明
search_total_indexed_vectors	所有索引中的向量总数
search_total_indexed_hash_keys	所有索引中的哈希类型键总数
search_total_indexed_json_keys	所有索引中的 JSON 类型键总数
search_total_index_size	所有索引使用的字节数
search_total_fulltext_index_size	非向量索引结构使用的字节数
search_total_vector_index_size	向量索引结构使用的字节数
search_max_index_lag_ms	上次摄取批量更新期间的摄取延迟

`search_ingestion` 部分

名称	说明
search_background_indexing_status	摄取状态。NO_ACTIVITY 表示闲置。其他值表示有键正在进行摄取。
search_ingestion_paused	除了在重新启动期间，此值始终应为“no”。

`search_backfill` 部分

> **Note**
>
> 本部分中记录的某些字段只有在当前存在回填操作时才可见。

Name	说明
search_num_active_backfills	当前回填活动数量
search_backfills_paused	除非内存不足，否则此值始终应为“no”。

Name	说明
search_current_backfill_progress_percentage	当前回填的完成百分比（0-100）

`search_query` 部分

名称	说明
search_num_active_queries	当前正在执行的 `FT.SEARCH` 和 `FT.AGGREGATE` 命令的数量

向量搜索安全

针对命令和数据访问的 ACL（访问控制列表）安全机制已扩展到控制搜索工具。系统完全支持针对单个搜索命令进行 ACL 控制。提供了一个新的 ACL 类别 `@search`，并更新了许多现有类别（`@fast`、`@read`、`@write` 等），以包含新命令。搜索命令不会修改键数据，这意味着将保留现有的 ACL 写入访问机制。哈希和 JSON 操作的访问规则不因索引的存在而改变；这些命令仍然受到普通键级别访问控制的约束。

带索引的搜索命令也可以通过 ACL 进行访问控制。访问检查在整个索引级别执行，而不是在每个键级别执行。这意味着，只有当用户有权访问该索引键空间前缀列表中所有可能的键时，系统才会向该用户授予对该索引的访问权限。换句话说，索引的实际内容并不能控制访问权限。用于安全检查的是前缀列表定义的索引理论内容。很容易造成这样一种情况：用户对密钥具有读 and/or 写权限，但无法访问包含该密钥的索引。请注意，创建或使用索引只需要具有对键空间的读取访问权限，而不考虑是否有写入访问权限。

有关与 MemoryDB ACLs 配合使用的更多信息，请参阅使用访问控制列表对用户进行身份验证()。ACLs

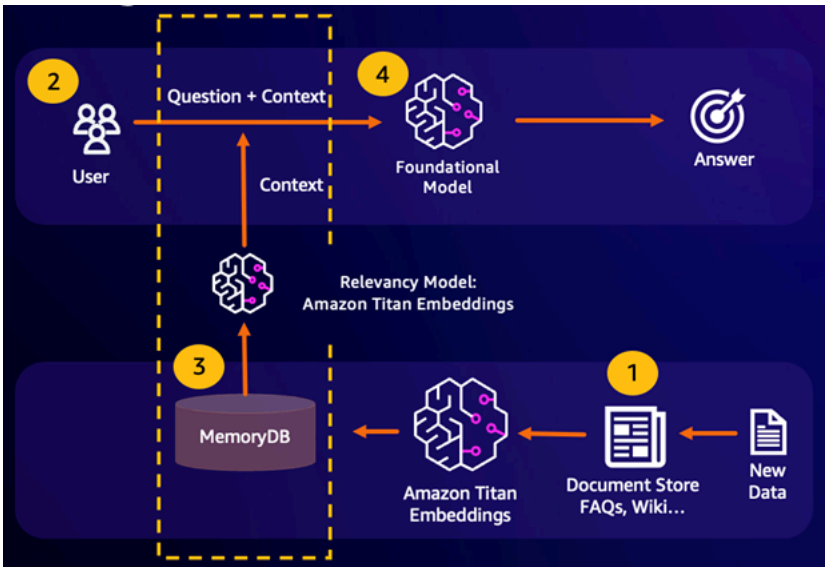
使用案例

以下是向量搜索使用案例。

检索增强生成（RAG）

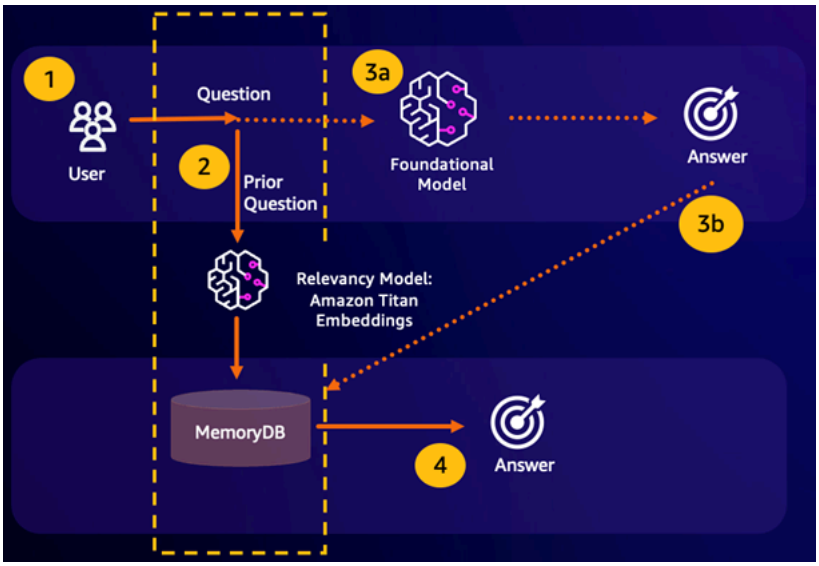
检索增强生成（RAG）利用向量搜索从大型数据语料库中检索相关段落，以增强大型语言模型（LLM）。具体而言，编码器会将输入上下文和搜索查询嵌入向量，然后使用近似的最近邻搜索找到语

义相似段落。这些检索到的段落会与原始上下文连接在一起，为 LLM 提供额外相关信息，从而向用户返回更准确的响应。



持久语义缓存

语义缓存是通过存储来自 FM 的先前结果来降低计算成本的过程。通过重复使用先前推断的结果而不是重新计算这些结果，语义缓存减少了通过推理期间所需的计算量。 FMsMemoryDB 支持持久语义缓存，从而避免丢失过往推断的数据。这使您的生成式人工智能应用程序能够在几毫秒内响应，提供先前语义相似问题的答案，同时通过避免不必要的 LLM 推断来降低成本。

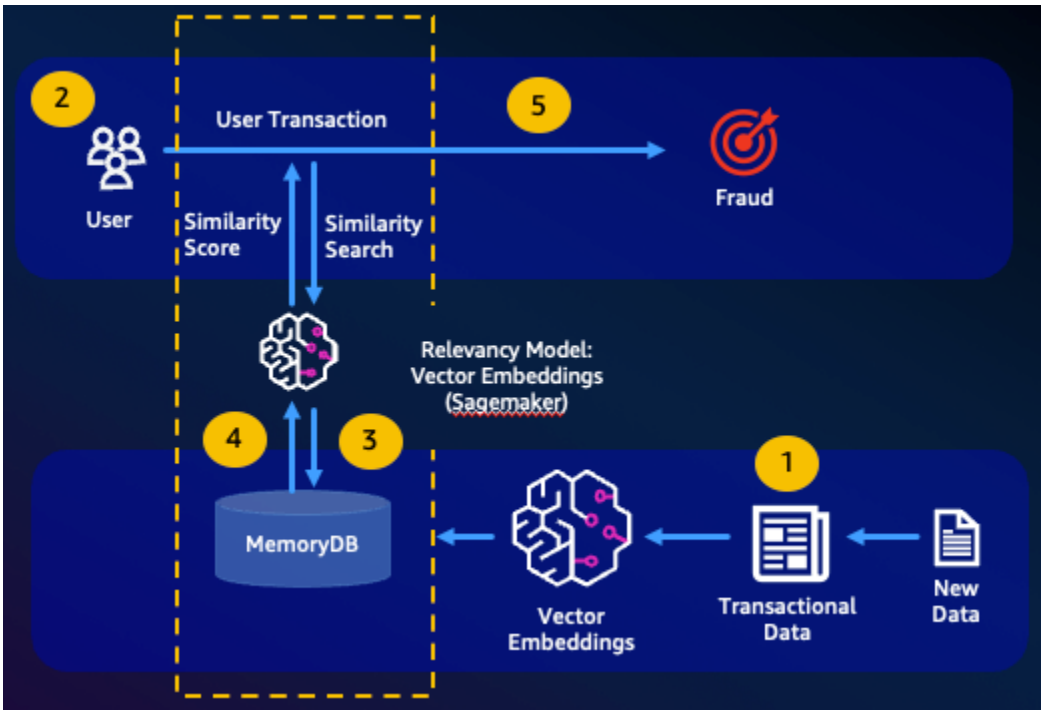


- 语义搜索命中 – 如果根据定义的相似度分数，客户的查询在语义上与前一个问题相似，FM 缓冲区内存（MemoryDB）会在第 4 步中返回前一个问题的答案，并且不会通过第 3 步调用 FM。这将避免基础模型（FM）延迟和相关成本，为客户提供更快的体验。

- 语义搜索未命中 – 如果根据定义的相似度分数，客户的查询在语义上与之前的查询不相似，则客户将在第 3a 步中调用 FM 提供对客户的响应。然后，从 FM 生成的响应将作为向量存储到 MemoryDB 中，供将来的查询使用（第 3b 步），以最大限度地降低语义相似问题上的 FM 成本。在此流程中，由于原始查询没有语义上相似的问题，因此不会调用第 4 步。

欺诈侦测

欺诈检测是一种异常检测形式，在它将有效事务表示为向量，并与全新事务的向量表示进行比较。当这些全新事务的向量与表示有效事务数据的向量相似性较低时，就会检测到欺诈。这样就可以通过对正常行为进行建模来检测欺诈，而不必试图预测每一个可能的欺诈实例。通过 MemoryDB，组织能够在高吞吐量时期执行此操作，最大程度减少误报，并保持毫秒级低延迟。



其他使用案例

- 推荐引擎可以通过将内容表示为向量，来为用户找到相似产品或内容。向量是通过对属性和模式进行分析而创建的。根据用户模式和属性，通过查找与已获得用户正面评价的内容相似度最高的向量，可以向用户推荐之前未看到过的新内容。
- 文档搜索引擎将文本文档表示为密集数字向量并捕捉语义。在搜索时，引擎会将搜索查询转换为向量，并使用近似近邻搜索来查找与查询向量相似度最高的文档。这种向量相似度方法允许根据语义而不仅仅是根据关键字的匹配来匹配文档。

向量搜索功能和限制

向量搜索可用性

R6g、r7g 和 t4g 节点类型支持支持向量搜索的 MemoryDB 配置，并且适用于所有可用 MemoryDB 的区域。 AWS

无法修改现有集群来启用搜索。但是，可以根据已禁用搜索的集群的快照来创建已启用搜索的集群。

参数限制

下表列出了各种向量搜索项目的限制：

Item	最大值
向量中的维数	32768
可以创建的索引数量	10
索引中的字段数量	50
FT.SEARCH 和 FT.AGGREGATE TIMEOUT 子句（毫秒）	10000
FT.AGGREGATE 命令中的管道阶段数量	32
FT.AGGREGATE LOAD 子句中的字段数量	1024
FT.AGGREGATE GROUPBY 子句中的字段数量	16
FT.AGGREGATE SORTBY 子句中的字段数量	16
FT.AGGREGATE PARAM 子句中的参数数量	32
HNSW M 参数	512
HNSW EF_CONSTRUCTION 参数	4096
HNSW EF_RUNTIME 参数	4096

扩展限制

MemoryDB 的向量搜索目前仅限于单个分片，不支持水平扩展。向量搜索支持垂直扩展和副本扩展。

操作限制

索引持久性和回填

向量搜索特征保留索引定义和索引内容。这意味着，在任何导致节点启动或重启的操作请求或事件中，索引定义和内容将从最新的快照中恢复，并从多可用区事务日志中读取任何待处理的交易。无需用户执行任何操作即可启动此操作。数据恢复后，重建过程将作为回填操作执行。这在功能上等同于系统自动为每个定义的索引执行 FT.CREATE 命令。请注意，数据恢复后，很可能在索引回填完成之前，节点便可用于应用程序操作，这意味着应用程序将再次看到回填，例如，使用回填索引的搜索命令可能会遭到拒绝。有关回填的更多信息，请参阅向量搜索概述。

索引回填的完成在主索引和副本之间不同步。应用程序可能会意外看到这种不同步的情况，因此建议应用程序在启动搜索操作之前，先在主副本和所有副本上验证回填完成情况。

快照 import/export 和实时迁移

搜索索引存在于 RDB 文件中，这会限制数据兼容传输。只有另一个启用了 MemoryDB 向量的集群才能理解由 MemoryDB 向量搜索功能定义的向量索引格式。此外，预览集群中的 RDB 文件可以由 GA 版本的 MemoryDB 集群导入，它将在加载 RDB 文件时重建索引内容。

但是，不包含索引的 RDB 文件不受这种限制。因此，只要在导出前删除索引，就可以将预览群集中的数据导出到非预览群集。

内存消耗

内存消耗基于向量数、维度数、M 值和非向量数据量，例如与向量关联的元数据或实例中存储的其他数据。

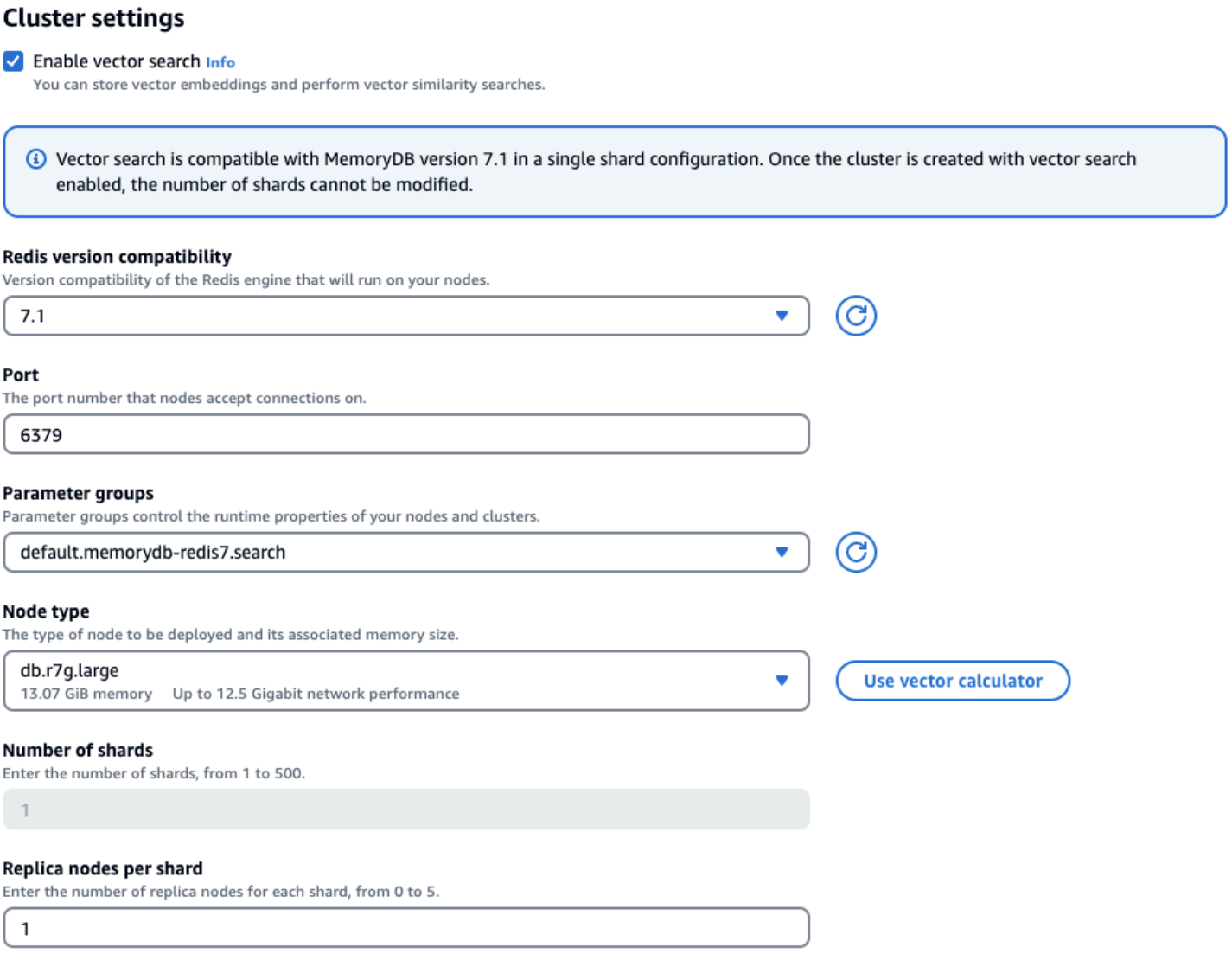
所需的总存储空间是实际向量数据所需空间和向量索引所需空间的组合。计算向量数据所需空间时，需要测量在 HASH 或 JSON 数据结构中存储向量所需的实际容量，以及为实现优化内存分配而需向最近的内存块增加的额外开销。每个向量索引都使用对存储在这些数据结构中的向量数据的引用，并使用有效的内存优化来删除索引中向量数据的任何重复副本。

向量的数量取决于您将数据表示为向量的方式。例如，您可以选择将单个文档表示成几个块，其中每个块表示一个向量。或者，您可以选择将整个文档表示为单个向量。

向量的维数取决于您选择的嵌入模型。例如，如果您选择使用 AWS Titan 嵌入模型，则维数将为 1536。

M 参数表示在索引构造期间为每个新元素创建的双向链接数。MemoryDB 将此值默认为 16；但您可以重写此设置。较高的 M 参数更适合高维度 and/or 高召回率要求，而低 M 参数更适合低维度 and/or 低召回要求。M 值会随着索引变大而增加内存消耗，从而使得总体内存消耗变得更高。

在控制台体验中，MemoryDB 提供了一种在集群设置下选中“启用向量搜索”后，根据向量工作负载的特性选择正确实例类型的简便方法。

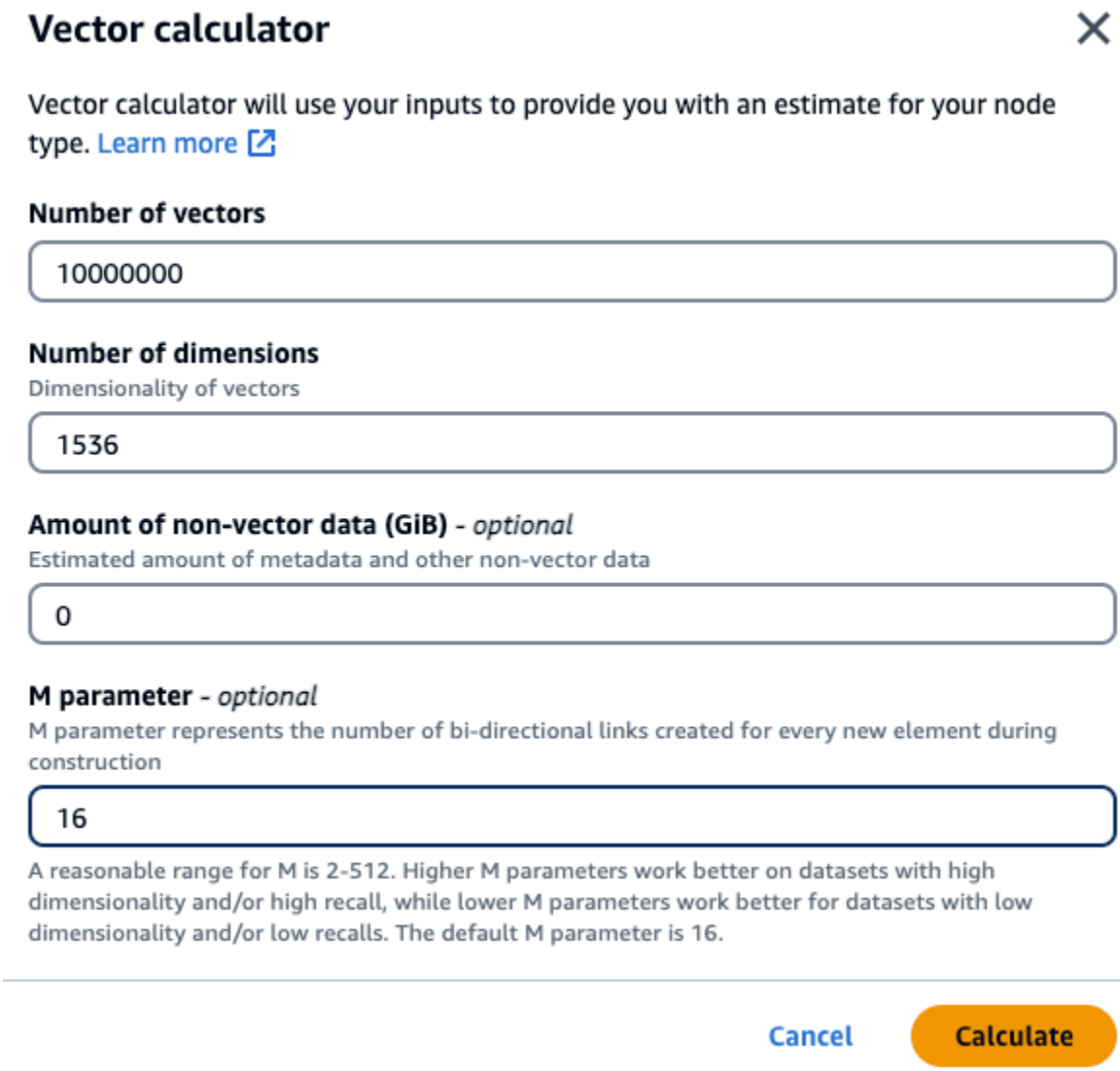


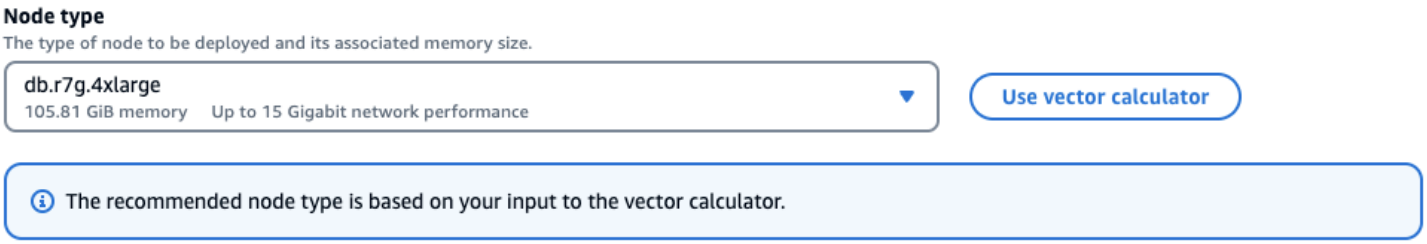
示例工作负载

客户希望以其内部财务文件为基础构建一个语义搜索引擎。他们目前持有 100 万份财务文档，使用具有 1536 个维度的 Titan 嵌入模型将每个文档分成 10 个向量，并且没有非向量数据。客户决定使用默认值 16 作为 M 参数。

- 向量：100 万 * 10 个块 = 1000 万个向量
- 维度：1536
- 非向量数据（GB）：0 GB
- M 参数：16

有了这些数据，客户可以在控制台中单击使用向量计算器按钮，根据其参数获取推荐的实例类型：





在此示例中，向量计算器将根据提供的参数寻找最小的 MemoryDB r7g 节点类型，该类型可以容纳存储向量所需的存储空间。请注意，这是一个近似值，您应该测试该实例类型以确保它符合您的要求。

根据上述计算方法和示例工作负载中的参数，此向量数据将需要 104.9 GB 来存储数据和单个索引。在这种情况下，因为 `db.r7g.4xlarge` 实例类型有 105.81 GB 的可用存储空间，所以建议使用这种实例类型。下一个更小的节点类型太小，无法承载向量工作负载。

由于每个向量索引都使用对所存储向量数据的引用，并且不会在向量索引中创建向量数据的额外副本，因此索引消耗的空间也相对较少。这在创建多个索引时以及部分向量数据已删除的情况下很有用，并且重建 HNSW 图将有助于建立适宜的节点连接，从而实现高质量的向量搜索结果。

在回填期间内存不足

与 Valkey 和 Redis OSS 写入操作类似，索引回填会受到限制。 out-of-memory如果引擎内存在回填过程中已满，则所有回填都将暂停。有可用内存后，回填过程会恢复。当由于内存不足导致回填暂停时，也可以删除内容和编制索引。

事务

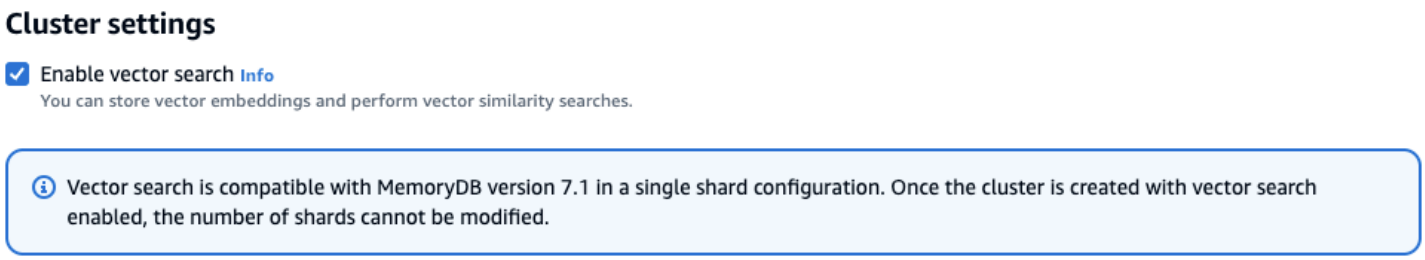
命令`FT.CREATE`、、、`FT.DROPINDEX`、`FT.ALIASADD`、`FT.ALIASDEL`、和`FT.ALIASUPDATE`不能在事务上下文中执行，也就是说，不能在 MULTI/EXEC 区块内或在 LUA 或 FUNCTION 脚本中执行。

创建已启用向量搜索的集群

您可以使用 AWS 管理控制台、或，创建启用矢量搜索的集群 AWS Command Line Interface。根据方法的不同，必须考虑启用向量搜索。

使用 AWS 管理控制台

要在控制台中创建支持向量搜索的集群，您需要在集群设置下启用向量搜索。在单个分片配置，向量搜索适用于 MemoryDB 版本 7.1。



有关在中使用矢量搜索的更多信息 AWS 管理控制台，请参阅创建集群（控制台）。

使用 AWS Command Line Interface

要创建启用向量搜索的 MemoryDB 集群，可以使用 MemoryDB create-cluster 命令，通过传递不可变参数组 `default.memorydb-redis7.search` 来启用向量搜索功能。

```
aws memorydb create-cluster \
  --cluster-name <value> \
  --node-type <value> \
  --engine redis \
  --engine-version 7.1 \
  --num-shards 1 \
  --acl-name <value> \
  --parameter-group-name default.memorydb-redis7.search
```

或者，您还可以创建新的参数组来启用向量搜索，如以下示例所示。您可以在此处了解有关参数组的更多信息。

```
aws memorydb create-parameter-group \
  --parameter-group-name my-search-parameter-group \
  --family memorydb_redis7
```

接下来，在新创建的参数组中将参数 search-enabled 更新为“是”。

```
aws memorydb update-parameter-group \
  --parameter-group-name my-search-parameter-group \
  --parameter-name-values "ParameterName=search-enabled,ParameterValue=yes"
```

现在，您可以使用此自定义参数组代替默认参数组，在 MemoryDB 集群上启用向量搜索。

向量搜索命令

以下是受支持的向量搜索命令列表。

主题

- FT.CREATE
- FT.SEARCH
- FT.AGGREGATE
- FT.DROPINDEX
- FT.INFO
- FT._LIST
- FT.ALIASADD
- FT.ALIASDEL
- FT.ALIASUPDATE
- FT._ALIASLIST
- FT.PROFILE
- FT.EXPLAIN
- FT.EXPLAINCLI

FT.CREATE

创建索引并启动该索引的回填。有关更多信息，请参阅向量搜索概述，了解索引构造的详细信息。

语法

```
FT.CREATE <index-name>
ON HASH | JSON
[PREFIX <count> <prefix1> [<prefix2>...]]
SCHEMA
(<field-identifier> [AS <alias>]
  NUMERIC
| TAG [SEPARATOR <sep>] [CASESENSITIVE]
| TEXT
| VECTOR [HNSW|FLAT] <attr_count> [<attribute_name> <attribute_value>])
```

```
)+
```

架构

- 字段标识符：
  - 对于哈希键，字段标识符是一个字段名称。
  - 对于 JSON 密钥，字段标识符是一个 JSON 路径。

  有关更多信息，请参阅 索引字段类型。

- 字段类型：
  - TAG：有关更多信息，请参阅标签。
  - NUMERIC：字段包含一个数字。
  - TEXT：字段包含任何数据块。
  - VECTOR：支持向量搜索的向量字段。
    - 算法 – 可以是 HNSW（分层可导航小世界）或 FLAT（暴力）。
    - `attr_count` – 将作为算法配置传递的属性数量，包括名称和值。
    - `{attribute_name} {attribute_value}`— 定义索引配置的特定算法 key/value 对。

      对于 FLAT 算法，属性为：

      必需：
      - DIM – 向量中的维度数。
      - DISTANCE_METRIC – 可以是 [L2 | IP | COSINE] 之一。
      - TYPE – 向量类型。`FLOAT32` 是一种受支持的类型。

      可选：
      - INITIAL_CAP – 索引中影响索引内存分配大小的初始向量容量。

      对于 HNSW 算法，属性为：

      必需：
      - TYPE – 向量类型。`FLOAT32` 是一种受支持的类型。
      - DIM – 向量维度，以正整数形式指定。最大值：32768
      - DISTANCE_METRIC – 可以是 [L2 | IP | COSINE] 之一。

可选：

- INITIAL_CAP – 索引中影响索引内存分配大小的初始向量容量。默认为 1024。
- M – 图中每层中每个节点允许的最大出站边缘数量。在零层，出站边缘的最大数目将为 2M。默认值为 16，最大值为 512。
- EF_CONSTRUCTION – 控制索引构建期间检查的向量数量。此参数的值越大，查准率就越高，但索引创建时间也会越长。默认值为 200。最大值为 4096。
- EF_RUNTIME – 控制查询操作期间检查的向量数量。此参数的值越大，查全率就越高，但查询时间也会越长。可以在查询时覆盖此参数的值。默认值为 10。最大值为 4096。

Return

返回简单的字符串 OK 消息或错误响应。

示例

> **Note**
>
> 以下示例使用 valkey-cli 的原生参数，例如在将数据发送到 Valkey 或 Redis OSS 之前，对数据去引号和去转义。要使用其他编程语言客户端（Python、Ruby、C# 等），请遵循这些环境处理字符串和二进制数据的处理规则。有关支持的客户端的更多信息，请参阅构建工具 AWS

Example 1：创建一些索引

为大小为 2 的向量创建索引

```
FT.CREATE hash_idx1 ON HASH PREFIX 1 hash: SCHEMA vec AS VEC VECTOR HNSW 6 DIM 2 TYPE
 FLOAT32 DISTANCE_METRIC L2
OK
```

使用 HNSW 算法创建 6 维 JSON 索引：

```
FT.CREATE json_idx1 ON JSON PREFIX 1 json: SCHEMA $.vec AS VEC VECTOR HNSW 6 DIM 6 TYPE
 FLOAT32 DISTANCE_METRIC L2
OK
```

Example示例 2：填入一些数据

对以下命令进行格式化，使其可以作为 redis-cli 终端程序的参数执行。使用编程语言客户端（例如，Python、Ruby、C#）的开发人员需要遵循其环境有关处理字符串和二进制数据的处理规则。

创建一些哈希和 json 数据：

```
HSET hash:0 vec "\x00\x00\x00\x00\x00\x00\x00\x00"
HSET hash:1 vec "\x00\x00\x00\x00\x00\x00\x80\xbf"
JSON.SET json:0 . '{"vec":[1,2,3,4,5,6]}'
JSON.SET json:1 . '{"vec":[10,20,30,40,50,60]}'
JSON.SET json:2 . '{"vec":[1.1,1.2,1.3,1.4,1.5,1.6]}'
```

注意以下几点：

- 哈希和 JSON 数据的键具有索引定义的前缀。
- 向量位于索引定义的相应路径上。
- 哈希向量以十六进制数据形式输入，而 JSON 数据以数字形式输入。
- 向量的长度适当，二维哈希向量条目包含两个浮点数的十六进制数据，六维 JSON 向量条目包含六个数字。

Example示例 3：删除并重新创建索引

```
FT.DROPINDEX json_idx1
OK

FT.CREATE json_idx1 ON JSON PREFIX 1 json: SCHEMA $.vec AS VEC VECTOR FLAT 6 DIM 6 TYPE
 FLOAT32 DISTANCE_METRIC L2
OK
```

请注意，新的 JSON 索引使用 FLAT 算法而不是 HNSW 算法。另请注意，将重新为现有的 JSON 数据编制索引：

```
FT.SEARCH json_idx1 "*=>[KNN 100 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
 DIALECT 2
1) (integer) 3
2) "json:2"
3) 1) "__VEC_score"
```

```
   2) "11.11"
   3) "$"
   4) "[{\"vec\":[1.1, 1.2, 1.3, 1.4, 1.5, 1.6]}]"
4) "json:0"
5) 1) "__VEC_score"
   2) "91"
   3) "$"
   4) "[{\"vec\":[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]}]"
6) "json:1"
7) 1) "__VEC_score"
   2) "9100"
   3) "$"
   4) "[{\"vec\":[10.0, 20.0, 30.0, 40.0, 50.0, 60.0]}]"
```

FT.SEARCH

使用提供的查询表达式查找索引中的键。找到后，可以返回这些键中索引字段的计数 and/or 内容。有关更多信息，请参阅向量搜索查询表达式。

要创建在这些示例中使用的数据，请参阅 FT.CREATE 命令。

语法

```
FT.SEARCH <index-name> <query>
[RETURN <token_count> (<field-identifier> [AS <alias>])+]
[TIMEOUT timeout]
[PARAMS <count> <name> <value> [<name> <value>]]
[LIMIT <offset> <count>]
[COUNT]
```

- RETURN：此子句确定返回键的哪些字段。每个字段的可选 AS 子句会在结果中覆盖该字段的名称。只能指定已为此索引声明的字段。
- LIMIT: <offset><count>：此子句提供分页功能，仅返回满足偏移量和计数值的键。如果省略此子句，则默认为“LIMIT 0 10”，即最多返回 10 个键。
- PARAMS：键值对数量的两倍。可以在查询表达式中引用参数 key/value 对。有关更多信息，请参阅向量搜索查询表达式。
- COUNT：此子句不返回键的内容，只返回键的数量。这就相当于使用“LIMIT 0 0”。

Return

返回数组或错误响应。

- 如果操作成功完成，则返回一个数组。第一个元素是与查询匹配的键的总数。其余元素是键名称和字段列表配对。字段列表是另一个数组，包含字段名称和对应值的配对。
- 如果索引正在回填，该命令会立即返回错误响应。
- 如果超时，该命令将返回错误响应。

示例：进行一些搜索

> **Note**
>
> 以下示例使用 valkey-cli 的原生参数，例如在将数据发送到 Valkey 或 Redis OSS 之前，对数据去引号和去转义。要使用其他编程语言客户端（Python、Ruby、C# 等），请遵循这些环境处理字符串和二进制数据的处理规则。有关支持的客户端的更多信息，请参阅构建工具 AWS

哈希值搜索

```
FT.SEARCH hash_idx1 "*=>[KNN 2 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00" DIALECT 2
1) (integer) 2
2) "hash:0"
3) 1) "__VEC_score"
   2) "0"
   3) "vec"
   4) "\x00\x00\x00\x00\x00\x00\x00\x00"
4) "hash:1"
5) 1) "__VEC_score"
   2) "1"
   3) "vec"
   4) "\x00\x00\x00\x00\x00\x00\x80\xbf"
```

将生成两个结果，按分数排序，即按与查询向量的距离（以十六进制输入）排序。

JSON 搜索

```
FT.SEARCH json_idx1 "*=>[KNN 2 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
 DIALECT 2
1) (integer) 2
```

```
2) "json:2"
3) 1) "__VEC_score"
   2) "11.11"
   3) "$"
   4) "[{\"vec\":[1.1, 1.2, 1.3, 1.4, 1.5, 1.6]}]"
4) "json:0"
5) 1) "__VEC_score"
   2) "91"
   3) "$"
   4) "[{\"vec\":[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]}]"
```

将生成两个最接近的结果，按分数排序，请注意，JSON 向量值会转换为浮点数，查询向量仍然是向量数据。另请注意，由于 KNN 参数为 2，因此只返回两个结果。值越大，返回的结果越多：

```
FT.SEARCH json_idx1 "*=>[KNN 100 @VEC $query_vec]" PARAMS 2 query_vec
 "\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\x00\
 DIALECT 2
1) (integer) 3
2) "json:2"
3) 1) "__VEC_score"
   2) "11.11"
   3) "$"
   4) "[{\"vec\":[1.1, 1.2, 1.3, 1.4, 1.5, 1.6]}]"
4) "json:0"
5) 1) "__VEC_score"
   2) "91"
   3) "$"
   4) "[{\"vec\":[1.0, 2.0, 3.0, 4.0, 5.0, 6.0]}]"
6) "json:1"
7) 1) "__VEC_score"
   2) "9100"
   3) "$"
   4) "[{\"vec\":[10.0, 20.0, 30.0, 40.0, 50.0, 60.0]}]"
```

FT.AGGREGATE

FT.SEARCH 命令的超集，允许对查询表达式选择的键进行大量额外处理。

语法

```
FT.AGGREGATE index query
  [LOAD * | [count field [field ...]]]
```

```
  [TIMEOUT timeout]
  [PARAMS count name value [name value ...]]
  [FILTER expression]
  [LIMIT offset num]
  [GROUPBY count property [property ...] [REDUCE function count arg [arg ...] [AS name]
[REDUCE function count arg [arg ...] [AS name] ...]] ...]]
  [SORTBY count [ property ASC | DESC [property ASC | DESC ...]] [MAX num]]
  [APPLY expression AS name]
```

- FILTER、LIMIT、GROUPBY、SORTBY 和 APPLY 子句可以任意次数重复，并且可以自由混合使用。它们按指定的顺序应用，其中一个子句的输出作为下一个子句的输入。
- 在上述语法中，“属性”要么是在 FT.CREATE 命令中为此索引声明的字段，要么是之前的 APPLY 子句或 REDUCE 函数的输出。
- LOAD 子句仅限于加载已在索引中声明的字段。“LOAD *”将加载索引中声明的所有字段。
- 支持以下 reducer 函数：
  COUNT、COUNT_DISTINCTISH、SUM、MIN、MAX、AVG、STDDEV、QUANTILE、TOLIST、FIRST_
  和 RANDOM_SAMPLE。有关更多信息，请参阅聚合。
- LIMIT <offset><count>：保留从 <offset> 开始的记录，并持续保留 <count> 条记录，所有其他记录都将丢弃。
- PARAMS：键值对数量的两倍。可以在查询表达式中引用参数 key/value 对。

Return

返回数组或错误响应。

- 如果操作成功完成，则返回一个数组。第一个元素是没有特殊含义的整数（应忽略）。其余元素是最后一个阶段输出的结果。每个元素都是由字段名称和值对组成的数组。
- 如果索引正在回填，该命令会立即返回错误响应。
- 如果超时，该命令将返回错误响应。

FT.DROPINDEX

删除一个索引。将删除索引定义和相关内容。键不受影响。

语法

```
FT.DROPINDEX <index-name>
```

Return

返回一条简单的字符串 OK 消息或错误响应。

FT.INFO

语法

```
FT.INFO <index-name>
```

FT.INFO 页面的输出是由键值对组成的数组，如下表所述：

键	值类型	说明
index_name	字符串	索引的名称
creation_timestamp	整数	创建时间的 Unix 样式时间戳
key_type	字符串	哈希或 JSON
key_prefixes	字符串数组	此索引的键前缀
fields	字段信息数组	此索引的字段
space_usage	整数	此索引使用的内存字节数
fullext_space_usage	整数	非向量字段使用的内存字节数
vector_space_usage	整数	向量字段使用的内存字节数
num_docs	整数	当前包含在索引中的键数
num_indexed_vectors	整数	当前包含在索引中的向量数
current_lag	整数	最近的摄取延迟（毫秒）
backfill_status	字符串	其中之一：已完成 InProgres、已暂停或失败

下表描述了每个字段的信息：

Key	值类型	说明
identifier	字符串	字段名称
field_name	字符串	哈希成员名称或 JSON 路径
类型	字符串	以下类型之一：“Numeric”、“Tag”、“Text”或“Vector”
option	字符串	忽略

如果字段的类型为 Vector，则根据算法的不同，可能会包含额外信息。

HNSW 算法：

Key	值类型	说明
algorithm	字符串	HNSW
data_type	字符串	FLOAT32
distance_metric	字符串	以下值之一：“L2”、“IP”或“Cosine”
initial_capacity	整数	向量字段索引的初始大小
current_capacity	整数	向量字段索引的当前大小
maximum_edges	整数	创建时的 M 参数
ef_construction	整数	创建时的 EF_CONSTRUCTION 参数
ef_runtime	整数	创建时的 EF_RUNTIME 参数

FLAT 算法：

Key	值类型	说明
algorithm	字符串	FLAT
data_type	字符串	FLOAT32
distance_metric	字符串	以下值之一：“L2”、“IP”或“Cosine”
initial_capacity	整数	向量字段索引的初始大小
current_capacity	整数	向量字段索引的当前大小

FT._LIST

列出所有索引。

语法

```
FT._LIST
```

Return

返回索引名称数组

FT.ALIASADD

为索引添加别名。新别名可以在需要索引名称的任何位置使用。

语法

```
FT.ALIASADD <alias> <index-name>
```

Return

返回一条简单的字符串 OK 消息或错误响应。

FT.ALIASDEL

删除索引的现有别名。

语法

```
FT.ALIASDEL <alias>
```

Return

返回一条简单的字符串 OK 消息或错误响应。

FT.ALIASUPDATE

更新现有别名以指向不同的物理索引。该命令只影响将来对别名的引用。当前正在进行的操作（FT.SEARCH、FT.AGGREGATE）不受此命令的影响。

语法

```
FT.ALIASUPDATE <alias> <index>
```

Return

返回一条简单的字符串 OK 消息或错误响应。

FT._ALIASLIST

列出索引别名。

语法

```
FT._ALIASLIST
```

Return

返回一个数组，其大小与当前别名的数量相同。此数组的每个元素都是别名索引对。

FT.PROFILE

运行查询并返回有关该查询的配置文件信息。

语法

```
FT.PROFILE
```

```
<index>
SEARCH | AGGREGATE
[LIMITED]
QUERY <query ....>
```

Return

由两个元素组成的数组。第一个元素是分析 FT.SEARCH 或 FT.AGGREGATE 命令的结果。第二个元素是由性能和分析信息组成的数组。

FT.EXPLAIN

解析查询并返回有关如何解析该查询的信息。

语法

```
FT.EXPLAIN <index> <query>
```

Return

包含解析结果的字符串。

FT.EXPLAINCLI

与 FT.EXPLAIN 命令相同，除了结果以更适合 redis-cli 的不同格式显示。

语法

```
FT.EXPLAINCLI <index> <query>
```

Return

包含解析结果的字符串。

MemoryDB 多区域

MemoryDB 多区域是一个完全托管的、主动-主动的多区域数据库，使您能够构建具有高达 99.999% 可用性、微秒级读取和个位数毫秒写入延迟的多区域应用程序。您可以从区域降级中提高可用性和弹性，同时受益于多区域应用程序的低延迟本地读取和写入。

使用 MemoryDB 多区域，您可以构建高可用的多区域应用程序，以提高弹性。它提供主动-主动复制，因此您可以从最接近客户的区域本地提供读取和写入，具有微秒级读取和个位数毫秒写入延迟。MemoryDB 多区域在区域之间异步复制数据，数据通常在一秒内传播。它会自动解决更新冲突并纠正数据分歧问题，使您能够专注于您的应用程序。

目前，以下区域支持 MemoryDB 多 AWS 区域：美国东部（弗吉尼亚北部和俄亥俄州）、美国西部（俄勒冈、加利福尼亚北部）、欧洲（爱尔兰、法兰克福和伦敦）和亚太地区（东京、悉尼、孟买、首尔和新加坡）。

只需点击几下即可轻松开始使用 MemoryDB 多区域， AWS 管理控制台 或者使用最新的 AWS SDK，或者。 AWS CLI

主题

- 先决条件和限制
- 工作原理
- 一致性和冲突解决
- 通过控制台使用 MemoryDB 多区域
- 通过 CLI 使用 MemoryDB 多区域
- 监控 MemoryDB 多区域
- MemoryDB 多区域扩缩
- 支持与不支持的命令

先决条件和限制

在开始使用 MemoryDB 多区域之前，请注意以下事项：

- MemoryDB 多区域在您选择的区域之间复制数据：通过创建多区域集群，您理解并同意数据将在您选择的区域之间移动。

从多区域组中移除一个区域也会删除该区域中的区域集群。

- 区域可用性-以下区域支持 MemoryDB 多 AWS 区域：美国东部（弗吉尼亚北部和俄亥俄州）、美国西部（俄勒冈、加利福尼亚北部）、欧洲（爱尔兰、法兰克福和伦敦）和亚太地区（东京、悉尼、孟买、首尔和新加坡）。
- 行为与设置：所有多区域的区域集群将具有相同数量的分片、实例类型、Valkey 引擎版本、TLS 和参数组设置。您可以为每个区域集群选择不同的 IAM 身份验证 ACLs、快照窗口、标签、客户托管密钥 (CMKs) 和维护窗口。

  借助 MemoryDB 多区域，不同区域中的集群可以拥有不同数量的副本。
- 支持的节点类型：MemoryDB 多区域支持 XL 及更大尺寸的 R7g 节点。

  MemoryDB 多区域支持 Valkey 引擎版本 7.3 及以上。
- 支持的数据类型：MemoryDB 多区域目前支持大多数 Redis OSS 或 Valkey 数据类型，我们将在未来添加对更多数据类型的支持。支持的数据类型包括字符串、哈希、集合和有序集合，但并非所有操作这些数据类型的命令都受支持。
- 区域总数-使用 MemoryDB 多区域，您最多可以在五个区域之间自动复制 MemoryDB 集群数据。AWS
- 支持的选项-MemoryDB 多区域支持 horizontal/vertical 扩展、IAM 集成 ACLs、自动和按需快照、自动软件修补和监控。
- 备份与恢复：您可以创建快照来备份多区域区域集群的数据。您可以手动创建快照，或者使用 MemoryDB 的自动快照调度程序，在您为每个区域集群单独指定的时间每天拍摄新快照。
- 迁移-您可以选择恢复任何 MemoryDB 或 Redis OSS/Valkey RDB 格式的备份。要从备份迁移数据，请创建一个新的 MemoryDB 多区域区域集群，并指定 Amazon S3 中的快照位置。如果是 MemoryDB 快照，您也可以指定其名称。MemoryDB 多区域将使用该快照中的数据创建区域集群。由于 MemoryDB 多区域支持字符串、哈希、集合、有序集合数据类型，您只能迁移这些受支持数据类型的快照数据。如果备份文件包含不受支持的 Redis OSS 数据类型，MemoryDB 多区域默认将使迁移操作失败。
- 资源预留：MemoryDB 多区域旨在保障区域可用性。每个节点上会永久保留部分资源，以确保本地读写请求的处理能够独立于对等区域的工作负载。这些资源也用于在对等区域发生事件（包括区域隔离事件及其恢复过程）期间保障本地可用性。与单区域 MemoryDB 相比，这会导致不同的性能特征。MemoryDB 多区域支持水平扩缩和垂直扩缩，以增加可用资源。
- 不是 RPO/RTO SLAs-MemoryDB 多区域不提供规定的服务等级协议。 RPO/RTO 它将继续接受与其他 AWS 区域隔离的 AWS 区域中的写入，这可能会无限期地增加交叉复制延迟。我们希望客户使

用 “MultiRegionClusterReplicationLag” 指标检测隔离，并根据他们想要的 RPO 将其应用程序流量重定向到另一个区域。

- 无单一端点或自动失效转移：发生区域性中断时，您必须手动将客户流量重定向到其他区域中的应用堆栈。您必须确保已为其正确配置对 MemoryDB 集群的多区域访问。
- 不支持 TTL - MemoryDB 多区域不支持 TTL（生存时间）。
- 不支持数据分层或向量搜索 - MemoryDB 多区域不支持向量搜索和数据分层功能。
- MemoryDB 多区域不支持 read-modify-write命令（APPEND、RENAMENX 等）。
- MemoryDB 多区域不保证 Redis OSS 事务的原子性和一致性。
- 授权模型：可以从任何受支持的区域调用 MemoryDB 多区域 API 操作。通过在 IAM 策略中指定多区域集群的 ARN，可以限制权限范围。多区域集群 ARN 的格式为 `arn:aws:memorydb::<account-id>:multiregioncluster/multi-region-cluster-name`。ARN 中不包含区域信息。
- 吞吐量限制-MemoryDB Multi-Region 在一个区域中每个节点最多可支持 1.3 个 GB/s 读取吞吐量，每个分片最多可支持 50 个 MB/s 全局聚合写入吞吐量。
- AWS p olicy-该 AWS ReadOnlyAccess 策略提供对 AWS 服务和资源的只读访问权限，但不会自动检索有关一个或多个多区域集群的详细信息。要检索一个或多个多区域集群的详细信息，请使用 AmazonMemoryDBReadOnlyAccess 策略或创建 IAM 客户管理型策略。
- 删除区域集群-删除区域集群时，任何关联的客户托管密钥 (CMKs) 都必须保持有效，直到区域集群完成删除。这确保了剩余的区域集群能够收敛到一致状态。

工作原理

MemoryDB 多区域的工作原理如下。

- 概念

  多区域集群是由一个或多个区域集群组成的集合，全部归一个 AWS 账户所有。

  区域集群是指作为多区域集群一部分的 AWS区域中的单个集群。每个区域集群存储相同的数据集。任何给定的多区域集群每个 AWS 区域只能有一个区域集群。

  当您创建多区域集群时，它包含多个区域集群（每个区域一个），MemoryDB 将其视为一个单元。当应用程序向任何区域集群写入数据时，MemoryDB 会自动异步地将该数据复制到多区域集群内的所有其他区域集群。您可以将区域集群添加到多区域集群中，使其可以在更多区域中使用。您将能够在最多五个区域之间自动复制 MemoryDB 集群数据。

- 可用性和耐久性

  在极不可能发生的区域隔离或区域降级事件中，您可以更新全局 DNS，将应用程序的流量重定向到其他健康的区域，而无需进行任何数据库重新配置，从而简化维护应用程序高可用性的过程。MemoryDB 在多可用区事务日志中持久存储来自所有区域的所有写入，以确保区域内数据无丢失。MemoryDB 多区域会跟踪所有已在区域内确认但尚未复制到所有成员集群的写入。如果某个区域被隔离或降级，它仍将继续接受本地写入。当被隔离的区域重新连接到多区域集群时，已确认但尚未复制到其他区域的写入将被复制到多区域集群中的所有区域。MemoryDB 多区域还会使用 CRDT 机制，自动将这些待处理的写入与中断期间在其他区域可能发生的任何更新进行协调。

- 连接到 MemoryDB 多区域集群

  要向您的区域集群写入数据和从中读取数据，请使用支持的 Redis OSS/Valkey 客户端（包括 Valkey GLIDE）连接到该集群。每个区域集群都有一个您的 Redis OSS/Valkey 客户端可以连接的终端节点。您可以使用 AWS 控制台、CLI 或 API 检索您的区域集群终端节点。然后，您可以在应用程序中使用（或配置）此终端节点来获取来自区域集群 read/write 的数据。

一致性和冲突解决

对某个区域集群中的键进行的任何更新，通常会在一秒内异步传播到多区域集群中的其他区域集群。如果某个区域被隔离或降级，MemoryDB 多区域会跟踪所有已执行但尚未传播到所有成员集群的写入。当该区域恢复在线时，MemoryDB 多区域会恢复将该区域中任何待处理的写入传播到其他区域的成员集群。它也会恢复将其他成员集群的写入传播到现已恢复在线的区域。无论该区域被隔离多长时间，所有以前成功的写入都将最终得到传播。

如果您的应用程序大约同时在不同区域更新相同的键，则可能产生冲突。MemoryDB 多区域使用无冲突复制数据类型（CRDT）来协调冲突的并发写入。CRDT 是一种可以独立、并发更新而无需协调的数据结构。这意味着写入-写入冲突会在每个副本上独立合并，并最终实现一致性。

具体来说，MemoryDB 使用两级“最后写入者胜出（LWW）”来解决冲突。对于字符串数据类型，LWW 在键级别解决冲突。对于其他数据类型，LWW 在子键级别解决冲突。冲突解决完全由系统托管，并在后台进行，不会对应用程序的可用性产生任何影响。以下是哈希数据类型的一个示例：

区域 A 在时间戳 T1 执行“HSET K F1 V1”；区域 B 在时间戳 T2 执行“HSET K F2 V2”；复制后，区域 A 和 B 都将拥有包含两个字段的键 K。当不同区域并发更新同一集合中的不同子键时，由于 MemoryDB 对哈希数据类型在子键级别解决冲突，这两次更新不会相互冲突。因此，最终数据将包含这两次更新的效果。

时间	区域 A	区域 B
T1	HSET K F1 V1	
T2		HSET K F2 V2
T3	同步	同步
T4	K: {F1:V1, F2:V2}	K: {F1:V1, F2:V2}

CRDT 及示例

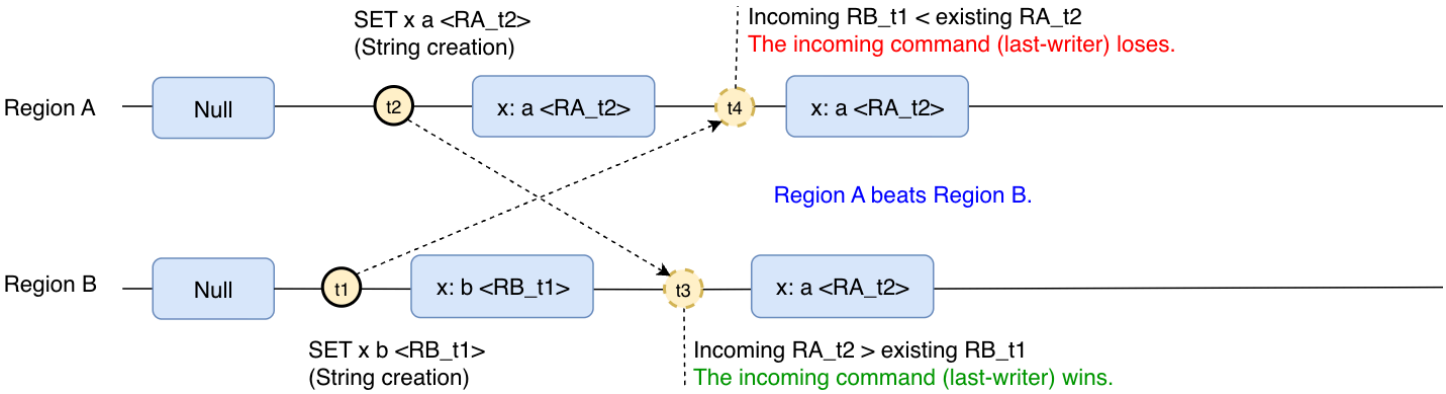
MemoryDB 多区域实现了无冲突复制数据类型（CRDT）来解决来自多个区域的并发写入冲突。CRDT 允许不同区域在最终接收到相同操作集后，无论顺序如何，都能独立达成最终一致性。

当单个键在多个区域中同时被更新时，需要解决写入-写入冲突以实现数据一致性。MemoryDB 多区域使用“最后写入者胜出（LWW）”策略来确定获胜操作，最终只能观察到“后发生”的操作效果。如果操作 op1 的效果在操作 op2 执行时，已在原始执行 op1 的区域中得到应用，则我们说操作 op1“发生在”操作 op2“之前”。

对于集合（哈希、集合和 SortedSet）MemoryDB 多区域解决元素级别的冲突。这允许 MemoryDB Multi-Region 使用 LWW 来解决每个元素上的 write/write 冲突。例如，从多个区域同时向同一集合添加不同元素，将导致该集合包含所有元素。

并发执行：最后写入者胜出

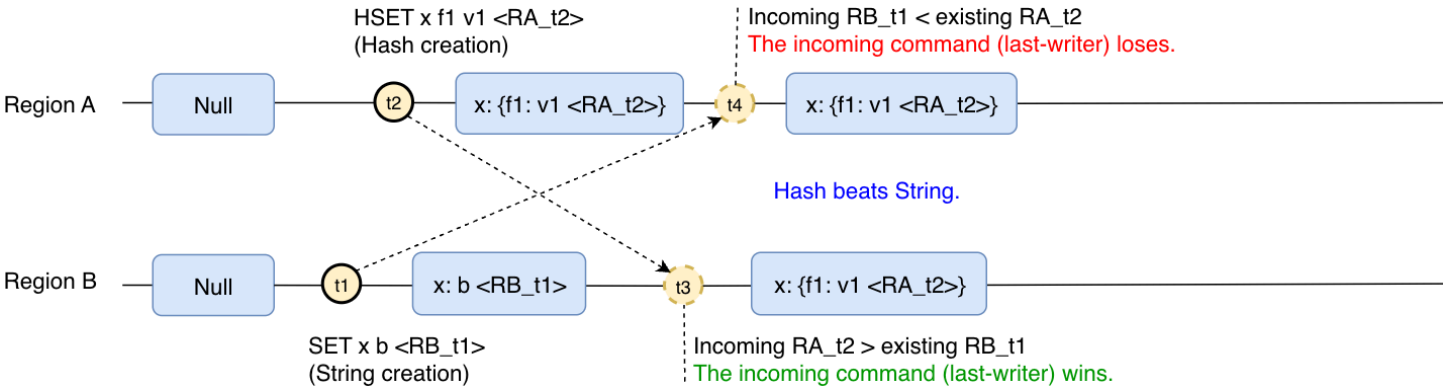
在 MemoryDB 多区域中，当存在键的并发创建时，在任何区域上执行的最后一个操作将决定该键的结果。例如：



键 x 在区域 B 上创建，值为“b”，但此后同一键在区域 A 上创建，值为“a”。由于区域 A 的操作是最后执行的操作，该键最终将收敛为值“a”。

数据类型冲突的并发执行：最后写入者胜出

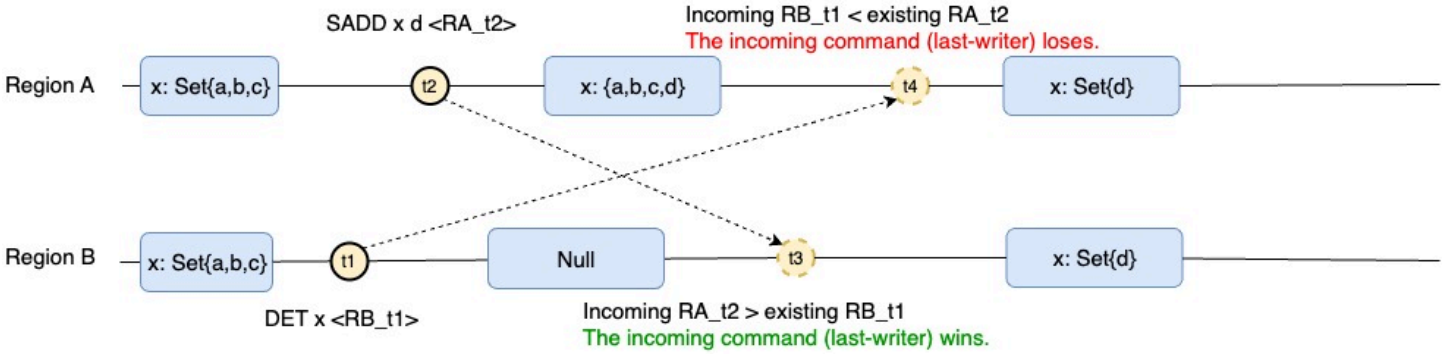
在前面的示例中，该键在两个区域中都是以相同类型创建的。如果该键是以不同数据类型创建的，也会观察到类似的行为：



键 x 在区域 B 上创建为字符串类型，值为“b”。但在此之后，且在该操作复制到区域 A 之前，同一键在区域 A 上被创建为哈希类型。由于区域 A 的操作是最后执行的操作，该键最终将收敛为在区域 A 上创建的哈希。

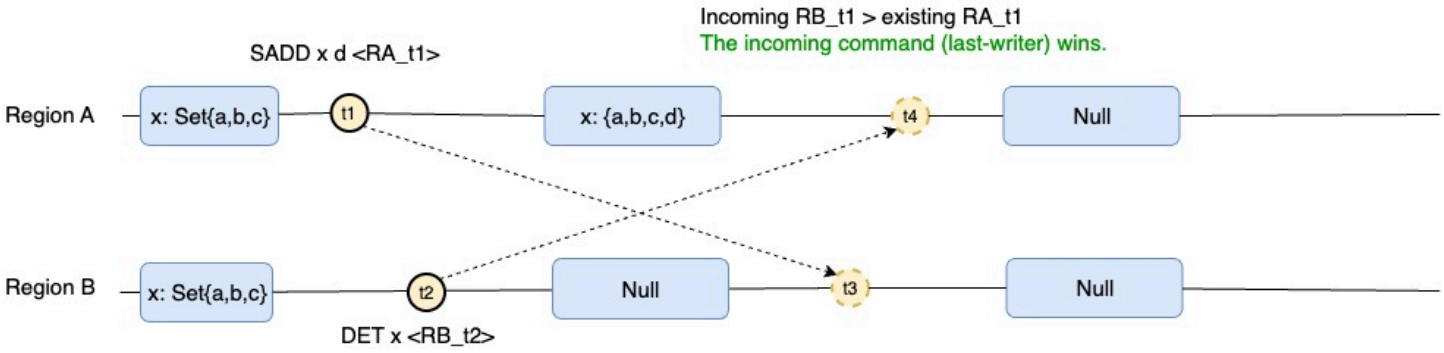
并发创建-删除：最后写入者胜出

在同时删除和 “创建”（意思是 of 值） replacement/addition 的场景中，最后执行的操作将获胜。最终结果将由删除操作的顺序决定。如果删除操作先发生：



区域 B 删除了 Set 类型的键 x。之后，区域 A 向该键添加了新成员。由于区域 A 的操作是最后执行的操作，该键最终将收敛为包含在区域 A 添加的唯一元素的 Set。

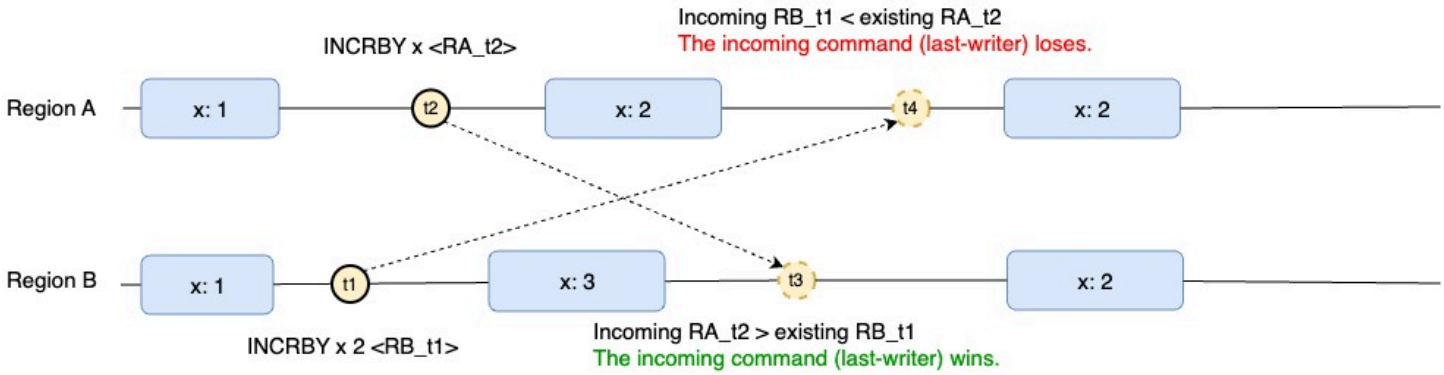
如果删除操作后发生：



区域 A 向 Set 类型的键 x 添加了新成员。之后，该键在区域 B 被删除。由于区域 B 的操作是最后执行的操作，最终该键将被删除。

计数器、并发操作：采用完整值复制与最后写入者胜出

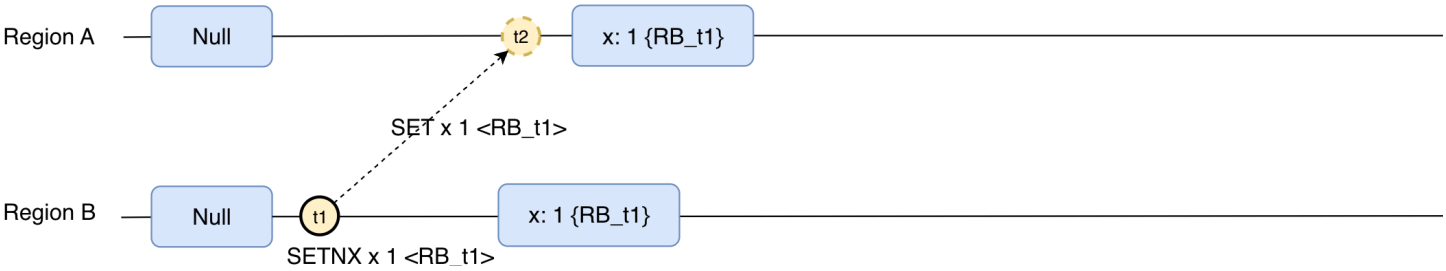
通过执行全值复制和应用，MemoryDB Multi-Region 中的计数器的行为与非计数器类型类似。 last-writer-strategy并发操作不会合并，而是最后一个操作将胜出。例如：



在此场景中，键 x 的初始值为 1。然后区域 B 将计数器 x 增加 2，之后不久区域 A 将计数器增加 1。由于区域 A 的操作是最后执行的操作，并且增加 1 是最后执行的操作，键 x 最终将收敛为值 2。

非确定性命令作为确定性命令复制

为保证不同区域间值的一致性，在 MemoryDB 多区域中，非确定性命令作为确定性命令复制。非确定性命令是指那些依赖于外部因素的命令，例如 SETNX。SETNX 依赖于键是否存在，而该键可能存在于远程区域，但不存在于接收命令的本地区域。因此，原本非确定性的命令会作为完整值复制进行复制。对于字符串类型，它将作为 SET 命令复制。



总之，所有对字符串类型的操作都作为 SET 或 DEL 复制；所有对哈希类型的操作都作为 HSET 或 HDEL 复制；所有对集合类型的操作都作为 SADD 或 SREM 复制；所有对有序集合的操作都作为 ZADD 或 ZREM 复制。

通过控制台使用 MemoryDB 多区域

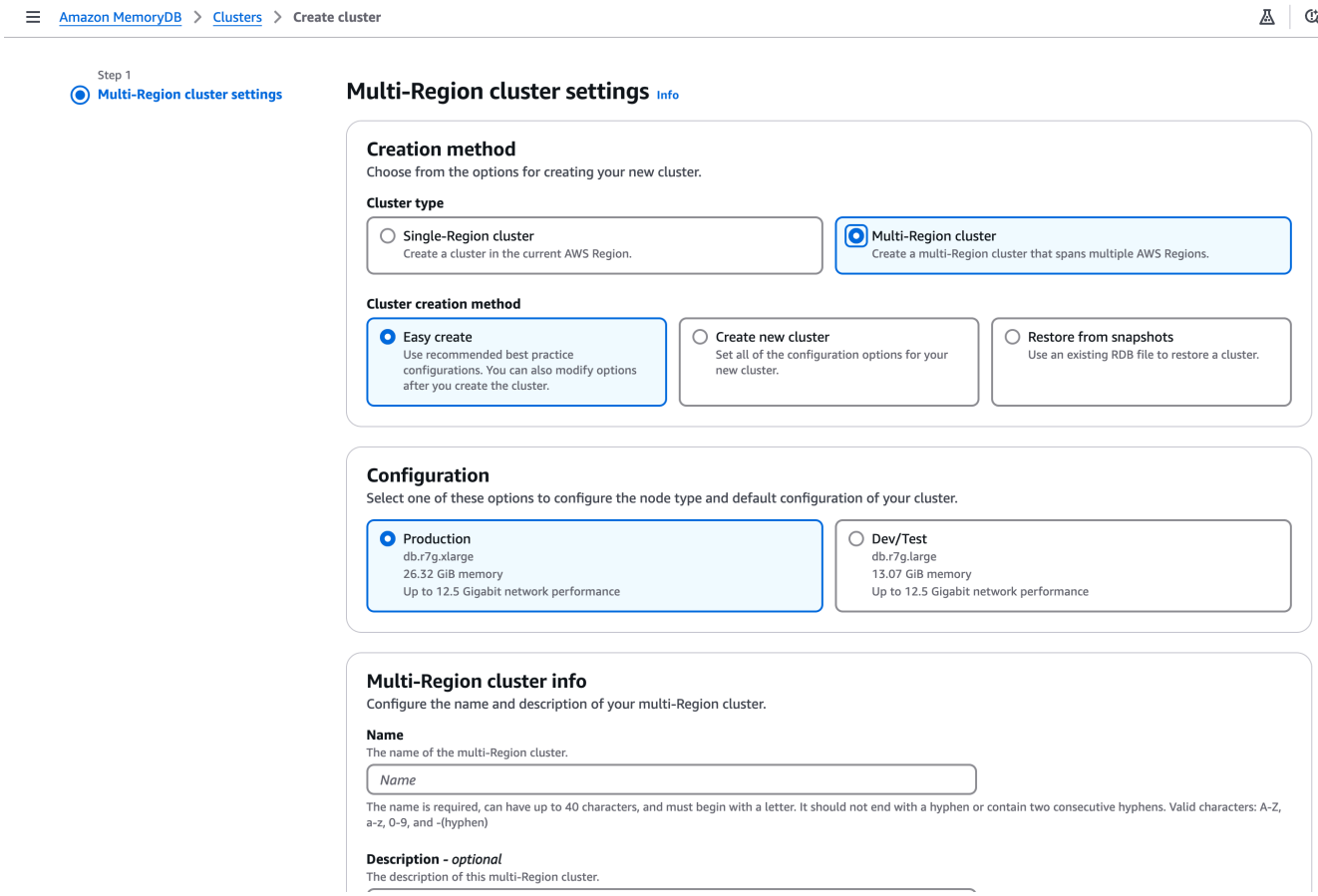
以下是通过控制台使用 MemoryDB 多区域的方法。

主题

- 在 MemoryDB 多区域中创建新集群
- 将快照还原到多区域集群内的新集群或现有集群
- 修改 MemoryDB 多区域中的集群
- 删除 MemoryDB 多区域中的集群

在 MemoryDB 多区域中创建新集群

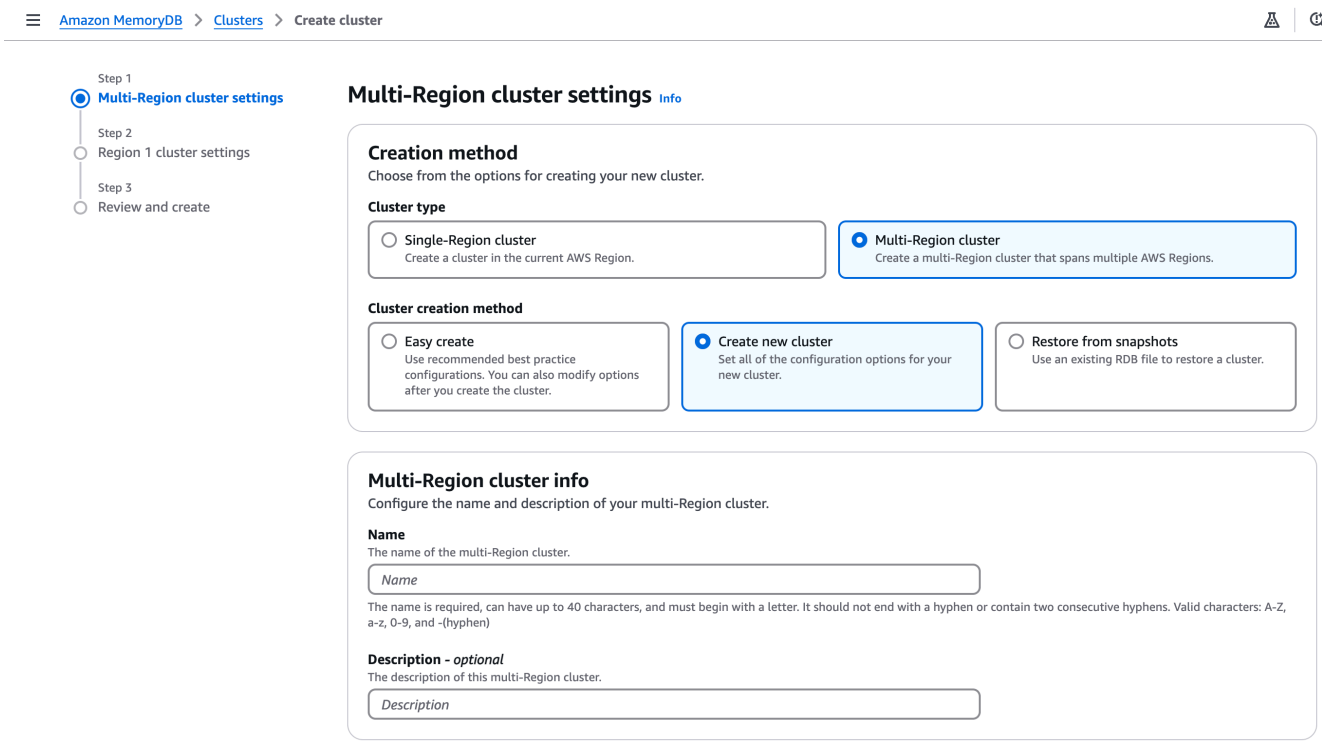
1. 从集群列表或仪表板导航至创建集群部分。



2. 在集群类型字段中，选择多区域集群。
3. 在集群创建方法字段中，选择简易创建。
4. 填写名称和描述，验证默认值并选择创建。

创建并配置集群

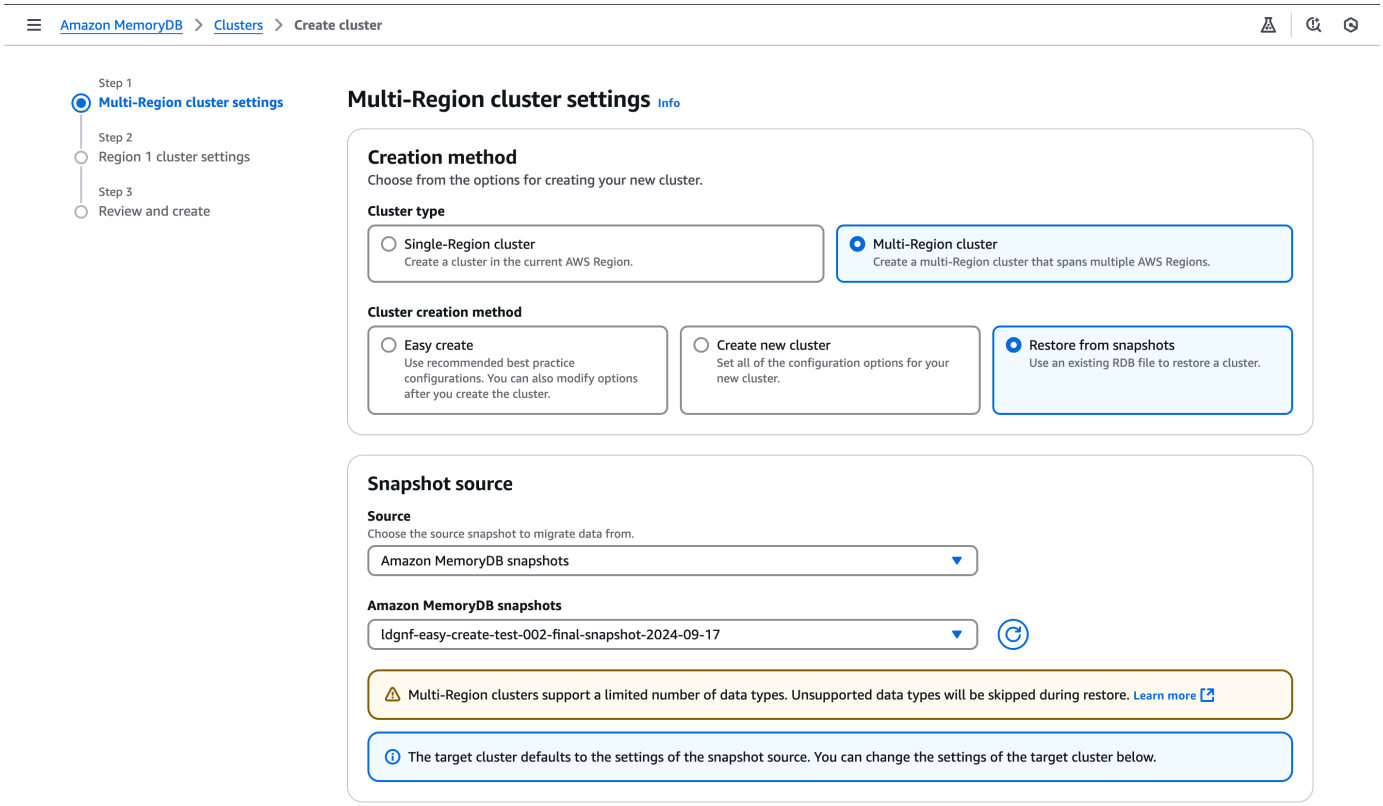
1. 从集群列表或仪表板导航至创建集群部分。



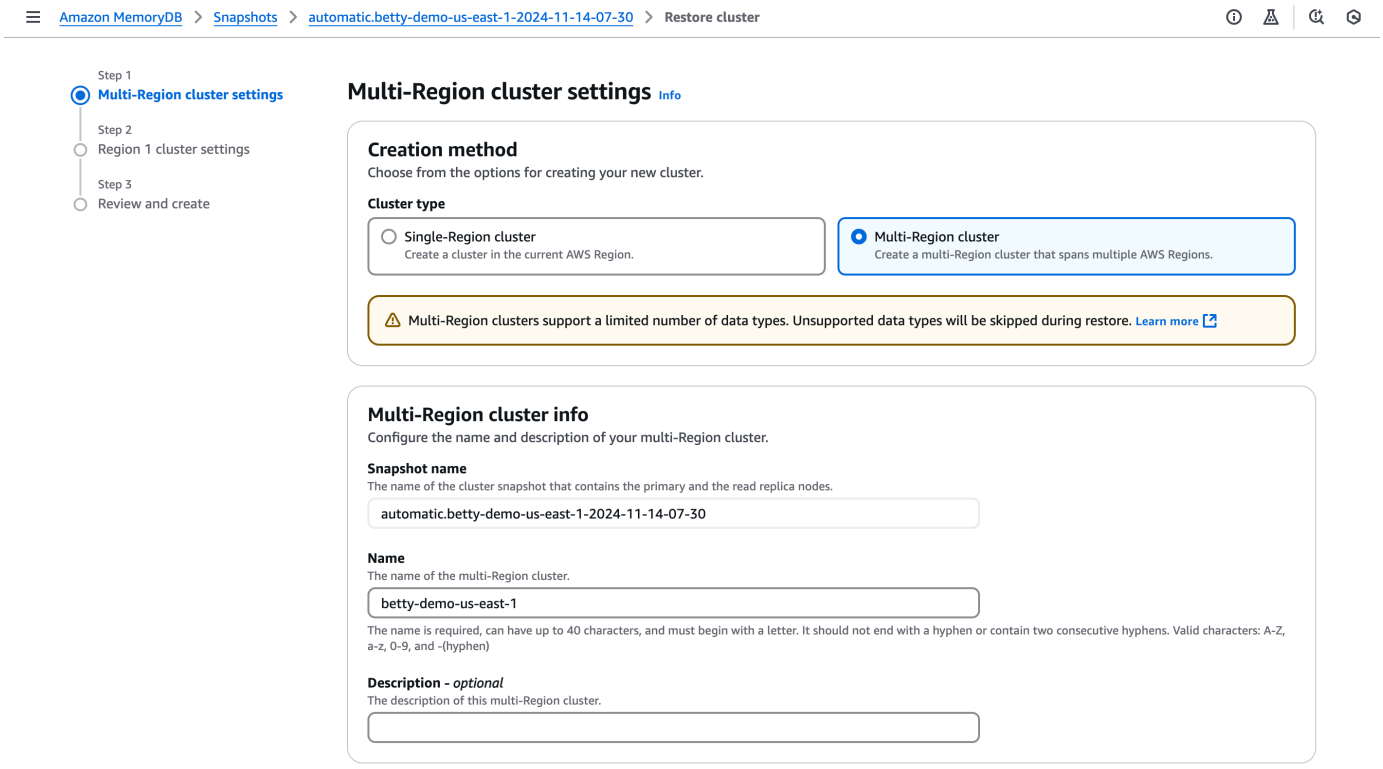
2. 在集群类型字段中，选择多区域集群。
3. 在集群创建方法字段中，选择创建新集群。
4. 填写名称和描述，验证值并选择创建。

将快照还原到多区域集群内的新集群或现有集群

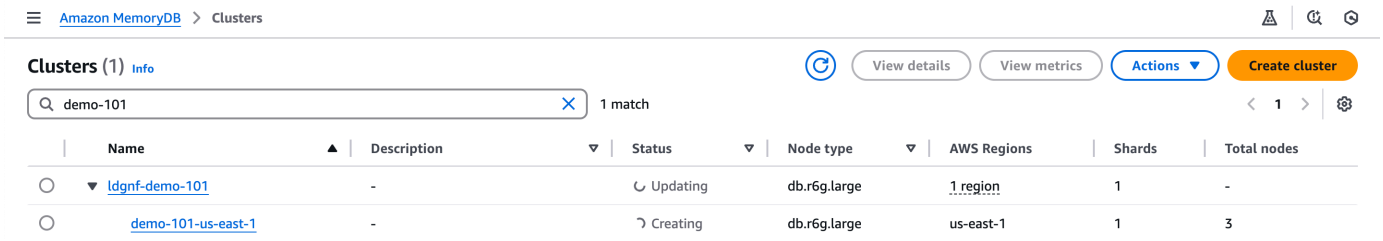
1. 从集群列表或仪表板导航至创建集群部分。



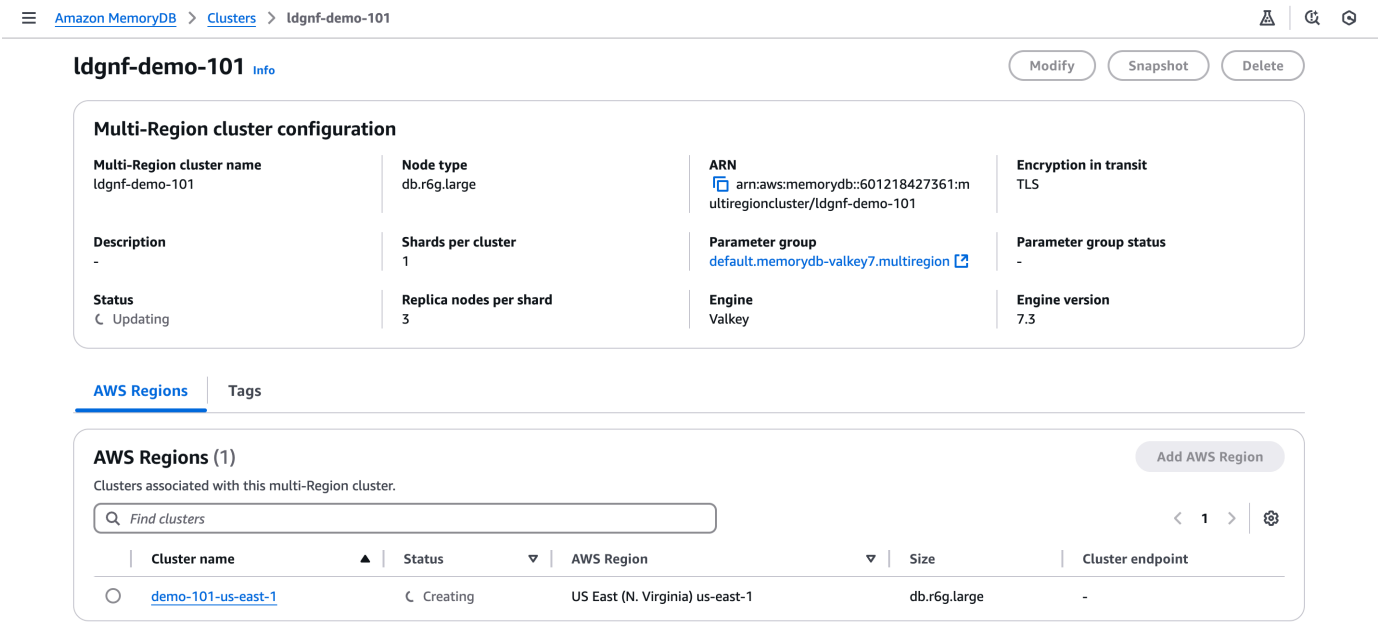
2. 在集群类型字段中，选择多区域集群。
3. 在集群创建方法字段中，选择从快照还原。
4. 选择源快照，然后填写必填字段。查看您的选择，然后选择还原。



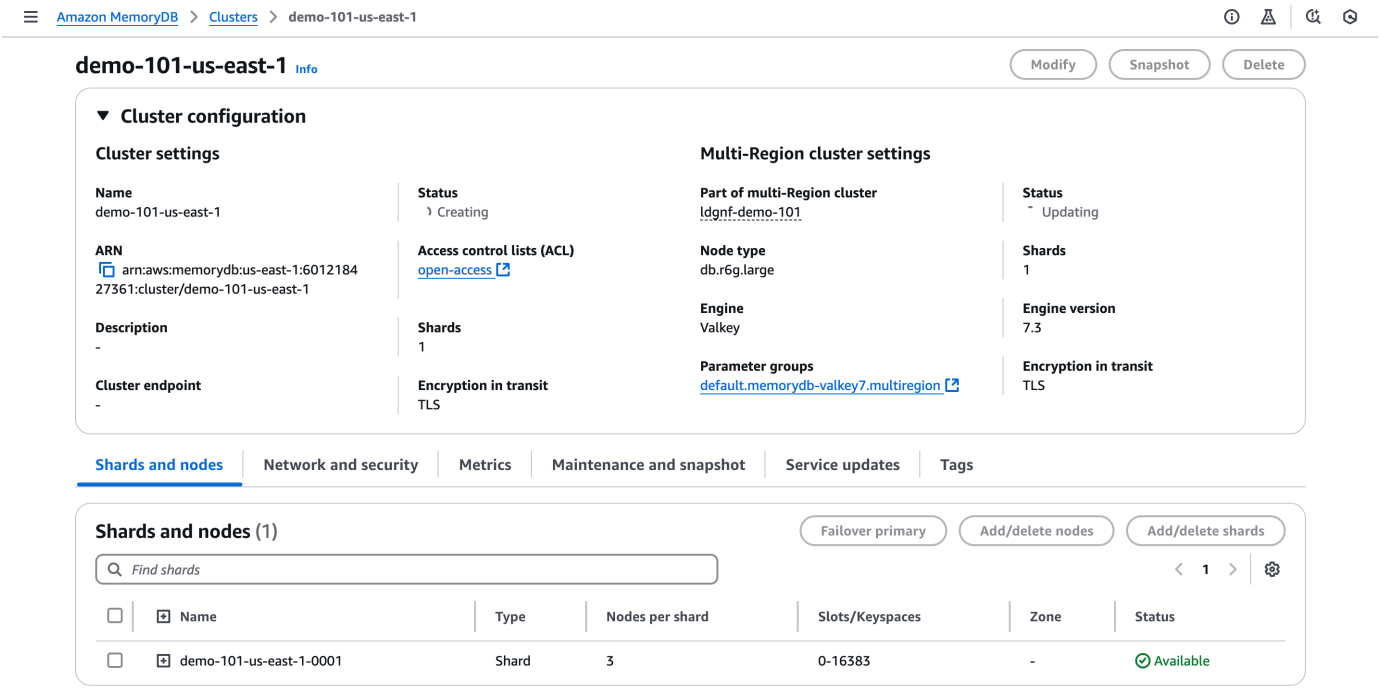
5. 要查看您的多区域集群，请导航至集群部分：



6. 现在选择目标多区域集群名称。

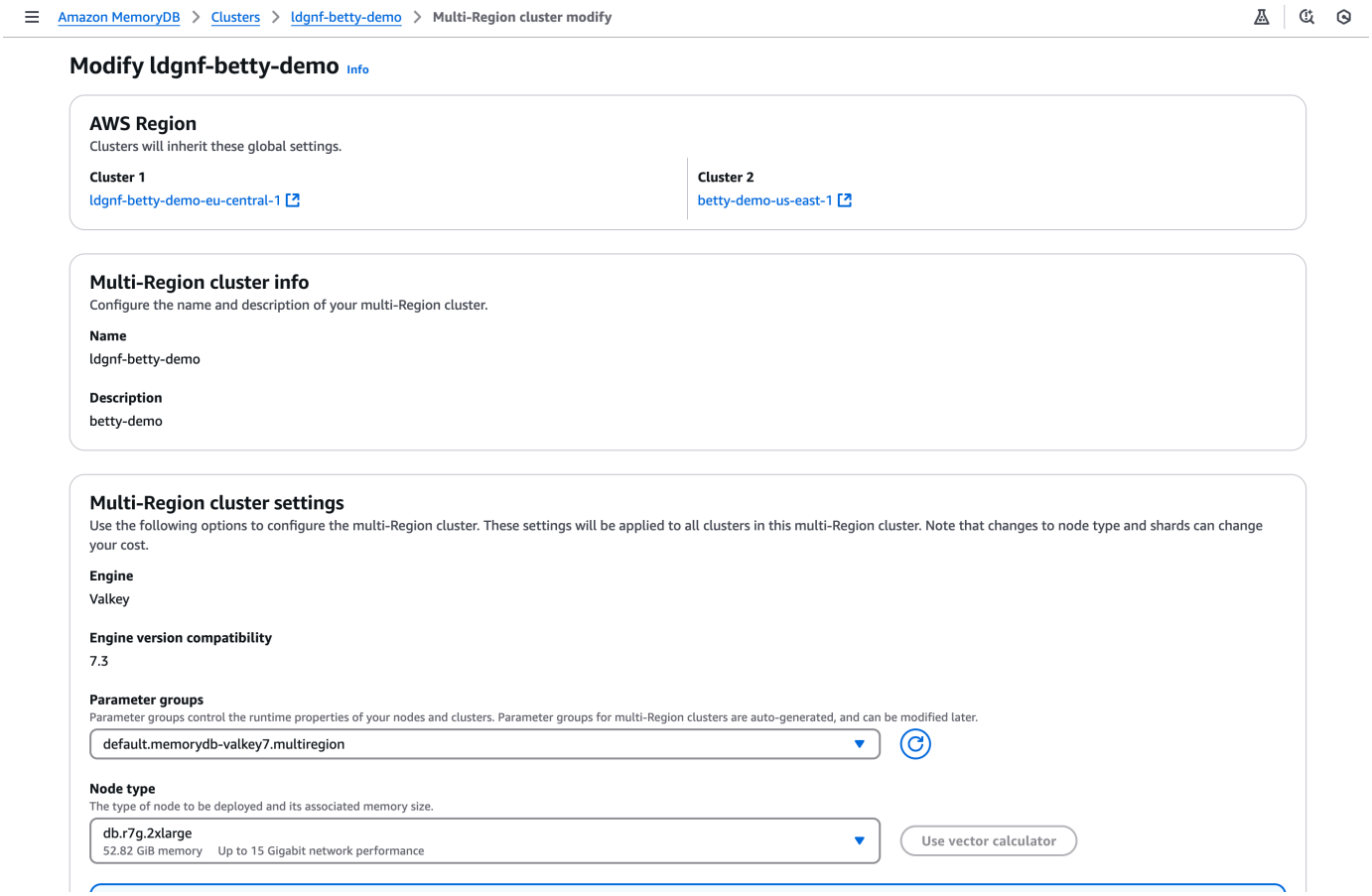


7. 现在选择目标区域集群名称。



修改 MemoryDB 多区域中的集群

1. 导航至集群部分。您应能看到所有当前集群。

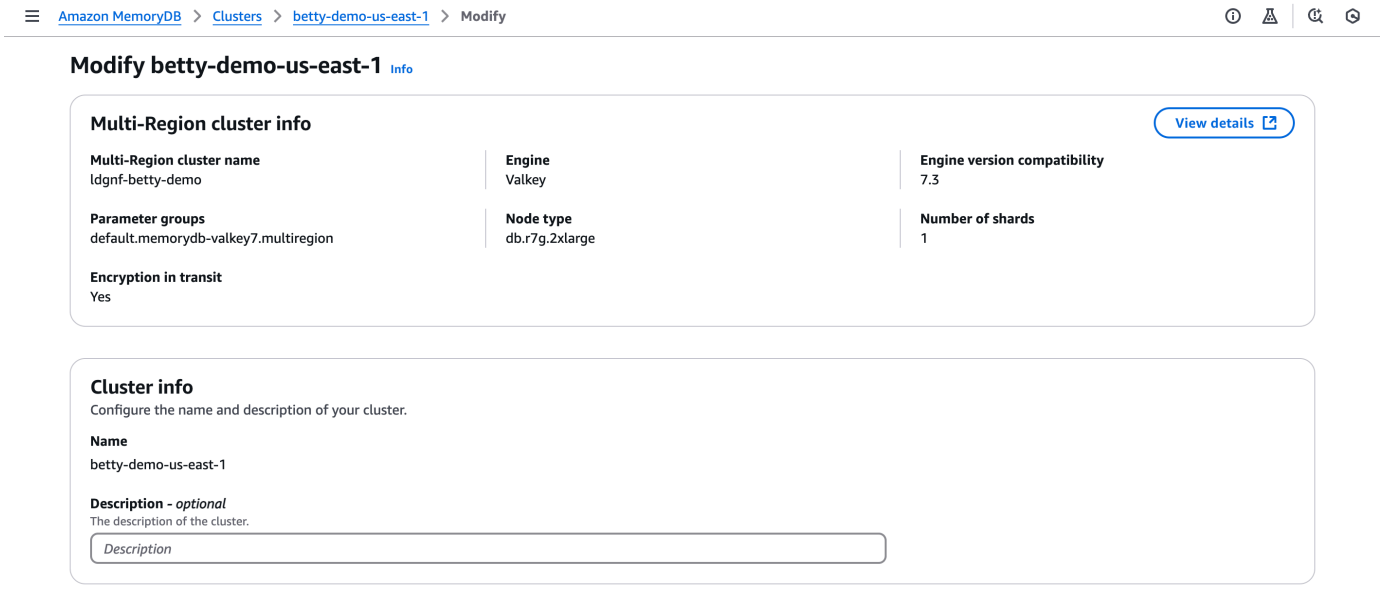


然后根据您要修改的集群类型，从以下步骤中选择。

2. 要修改多区域集群中的单个集群，请先选择其所属的多区域集群。然后在操作中选择编辑按钮（右上角）。接着选择目标单个集群。您也可以从详细信息页面修改此集群。

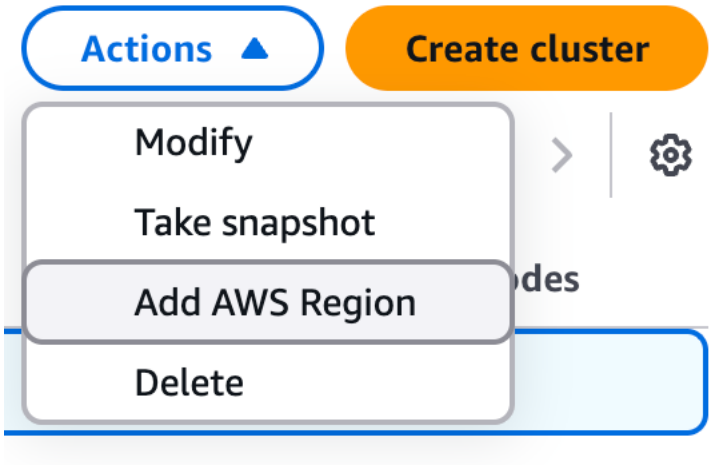
修改区域集群

1. 要修改多区域集群，请选择目标多区域集群名称。

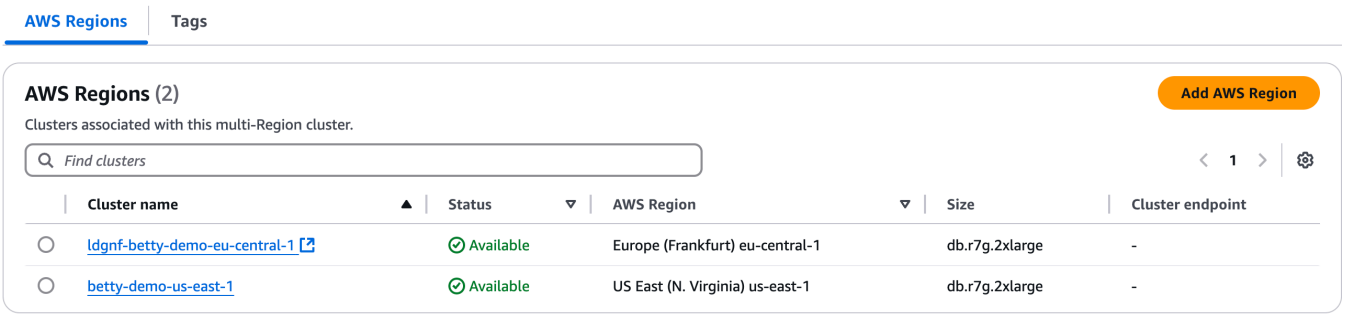


然后选择该集群，并在操作中（右上角）或从详细信息页面选择编辑按钮。

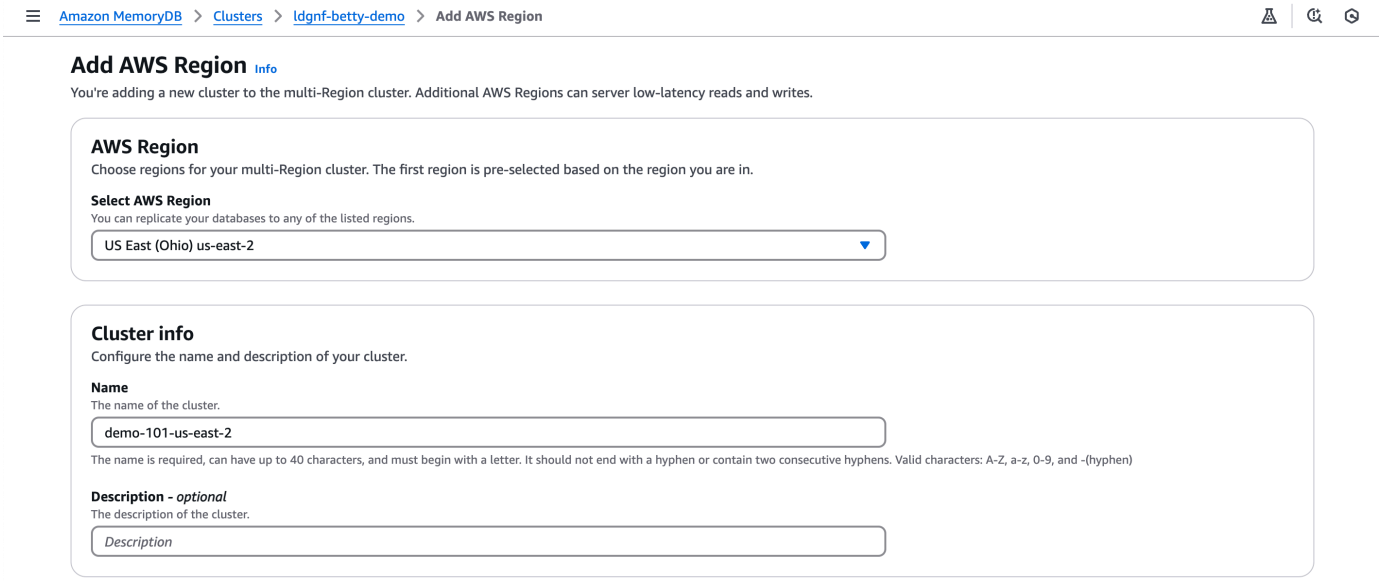
2. 要添加区域群集，请选择所选的目标多区域群集，然后转到操作下拉列表并选择添加 AWS 区域。您也可以前往 AWS 区域的详细信息页面，选择目标多区域集群，然后从那里进行添加。



3. 要添加区域，请选择目标区域。然后填写所需信息并选择添加 AWS 区域。

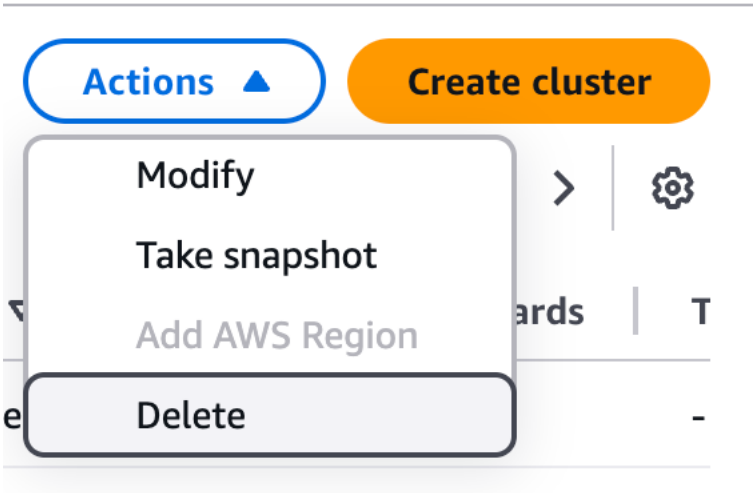


4. 要向空的多区域集群添加新的区域集群，您将看到与创建多区域集群时相同的选项。唯一的区别是多区域集群信息已经存在。

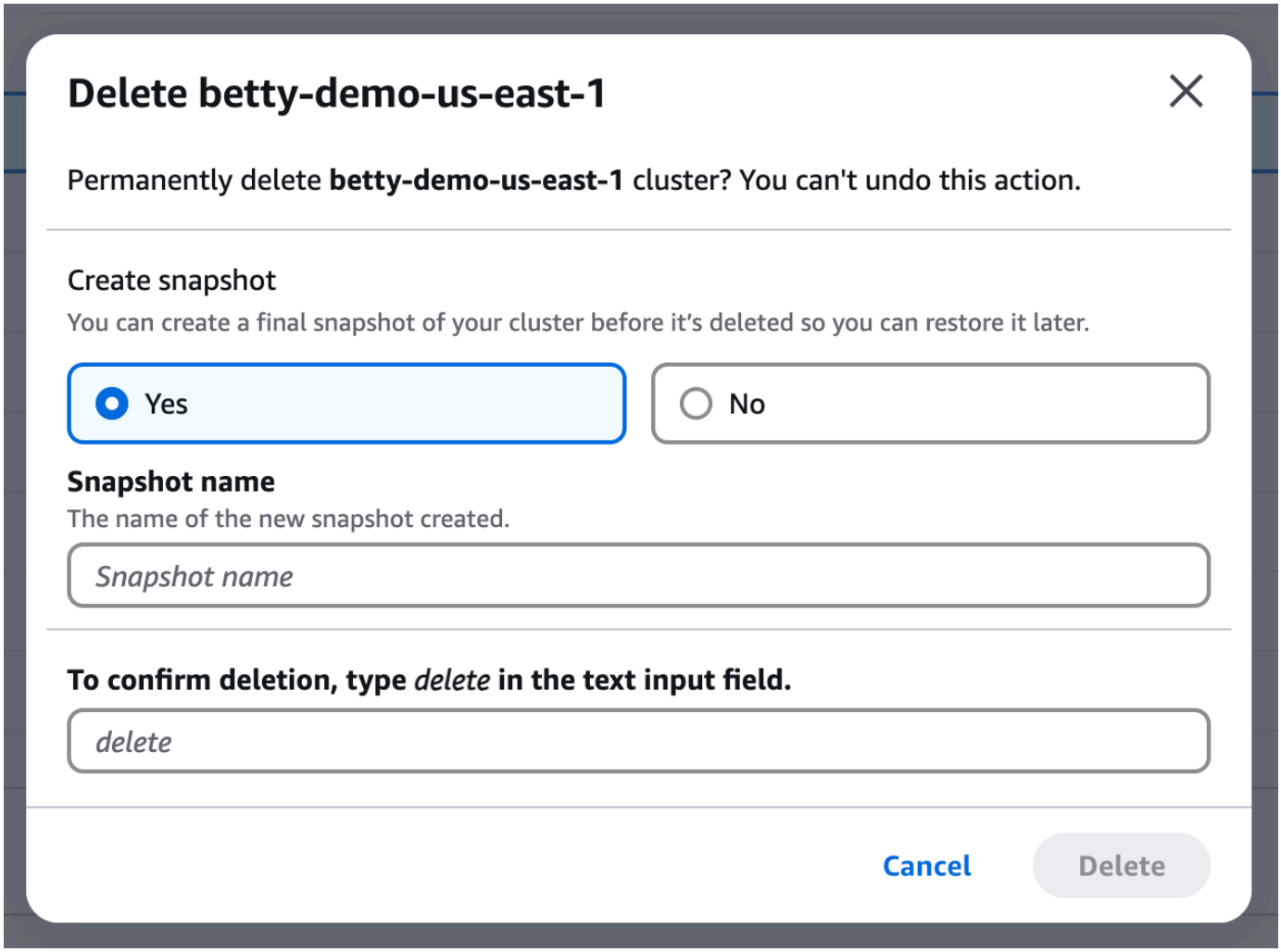


删除 MemoryDB 多区域中的集群

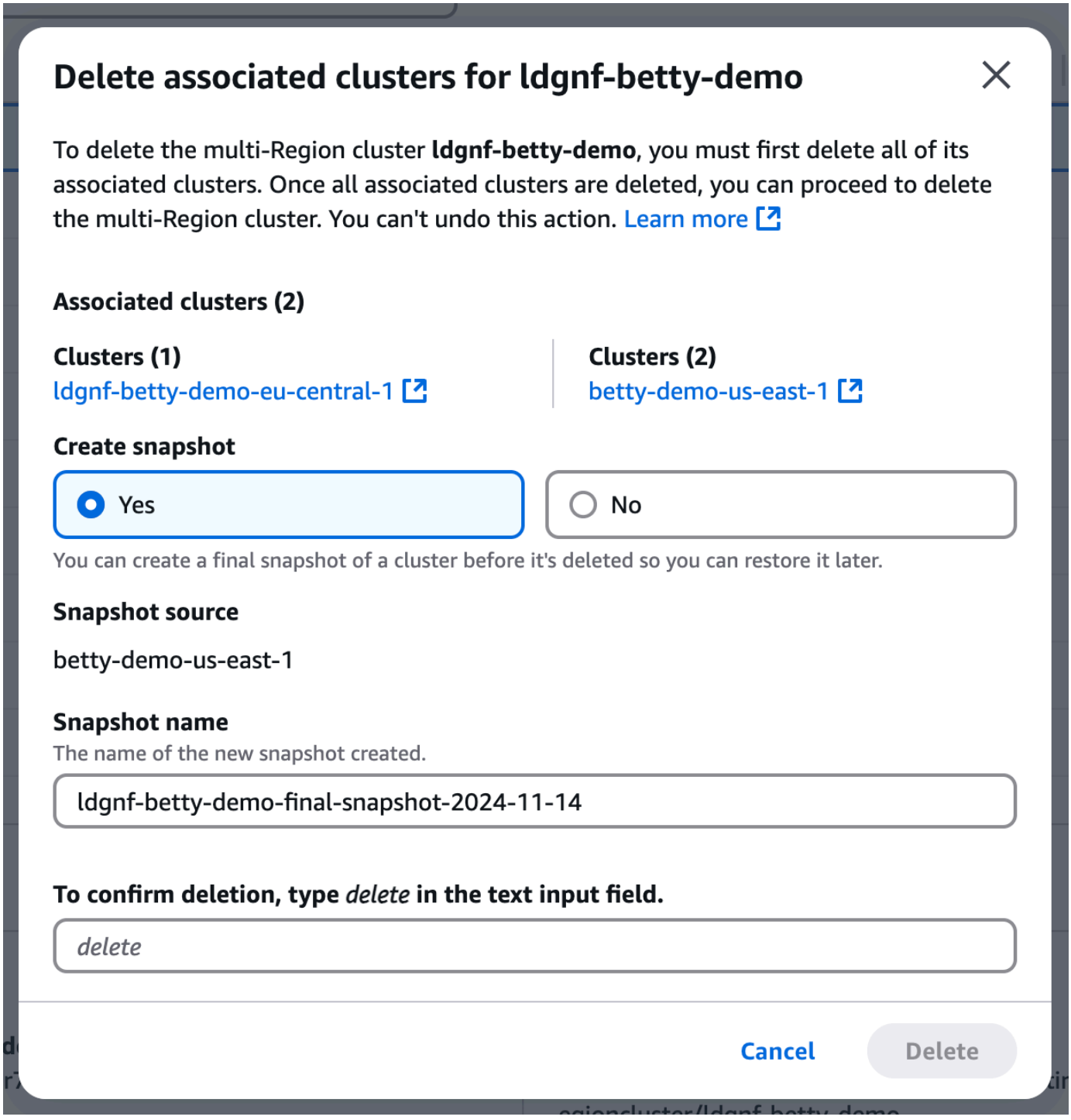
1. 要删除某个区域中的单个集群，请选择目标区域集群。然后转到操作菜单下拉列表，选择单个集群，并选择删除。



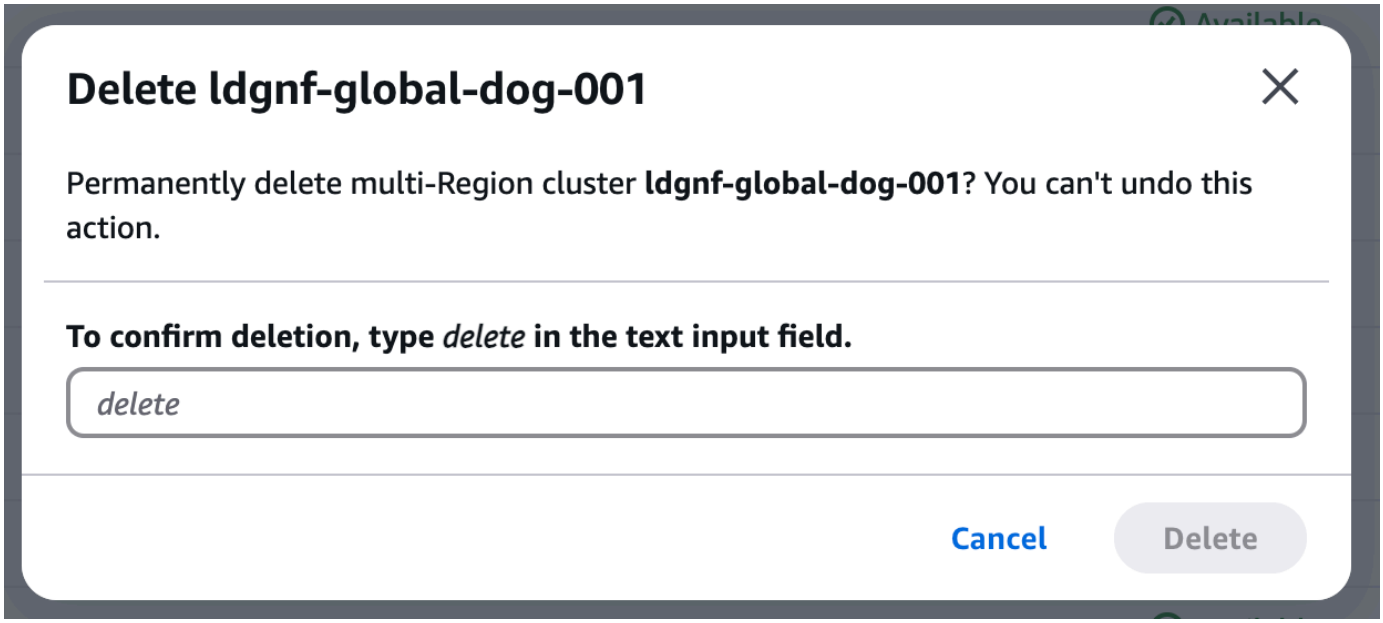
系统将显示一个确认窗口，包括在删除前创建快照的选项。如果您仍要删除，请在文本字段中输入“delete”，然后选择删除。



2. 要删除多区域集群的所有关联区域集群，请选择包含一个或多个集群的目标多区域集群。选中目标多区域集群后，转到操作菜单下拉列表并选择删除。



3. 要删除整个多区域集群，请选择目标空多区域集群。然后转到操作菜单下拉列表并选择删除。



通过 CLI 使用 MemoryDB 多区域

以下是通过 CLI 使用 MemoryDB 多区域的方法

> **Note**
>
> MemoryDB 多区域仅支持节点类型 db.r7g.xlarge 及以上。

创建带有内存DBMulti 区域的集群

创建多区域集群

```
aws memorydb create-multi-region-cluster \
 --multi-region-cluster-name-suffix my-multi-region-cluster \
 --node-type db.r7g.xlarge \
 --engine valkey \
 --region us-east-1
```

在美国东部（弗吉尼亚州北部）区域创建区域集群

```
aws memorydb create-cluster \
 --cluster-name my-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
```

```
 --node-type db.r7g.xlarge \
 --acl-name open-access \
 --region us-east-1 \
```

在欧洲地区（爱尔兰）区域创建区域集群

```
aws memorydb create-cluster \
 --cluster-name my-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --node-type db.r7g.xlarge \
 --acl-name open-access \
 --region eu-west-1 \
```

从任何区域描述多区域集群

```
aws memorydb describe-multi-region-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --region eu-west-1
```

更新多区域集群

修改节点类型

```
aws memorydb update-multi-region-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --node-type db.r7g.4xlarge \
 --region us-east-1
```

修改分片数量

```
aws memorydb update-multi-region-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --shard-configuration \
 ShardCount=3 \
 --update-strategy COORDINATED \
 --region us-east-1
```

扩展 MemoryDB 集群

首先，使用 `list-allowed-node-type-updates` 命令列出可以纵向扩展或横向扩展的节点：

```
aws memorydb list-allowed-node-type-updates \
 --cluster-name my-cluster-name
```

这将提供可以扩容或缩容的节点列表。然后要更新它们，您可以使用 update-cluster 命令：

```
aws memorydb update-cluster  \
 --cluster-name my-cluster \
 --node-type db.r6g.2xlarge
```

有关多区域扩缩的更多信息，请参阅 MemoryDB 多区域扩缩。

正在删除 MemoryDB 多区域中的集群

删除区域集群

```
aws memorydb delete-cluster \
 --cluster-name my-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --region us-east-1
```

删除多区域集群

```
aws memorydb delete-multi-region-cluster \
 --multi-region-cluster-name my-multi-region-cluster \
 --region us-east-1
```

监控 MemoryDB 多区域

您可以使用 Amazon CloudWatch 来监控多区域集群的行为和性能。MemoryDB 为多区域集群内的每个区域集群发布 MultiRegionClusterReplicationLag 指标。

MultiRegionClusterReplicationLag 显示从更新被写入远程多区域区域集群的多可用区事务日志，到该更新被写入本地多区域区域集群的主节点之间所经过的时间。该指标以毫秒为单位表示，在分片级别为每个源区域和目标区域对发出。

在正常操作期间，MultiRegionClusterReplicationLag 应相当恒定。MultiRegionClusterReplicationLag 数值升高可能表明来自某个区域集群的更新未能及时传播到其他区域集群。随着时间的推移，这可能导致其他区域集群落后，因为它们无法持续接收更新。

MultiRegionClusterReplicationLag如果某个 AWS 区域变得孤立或降级，并且您在该区域中有一个区域集群，则可能会增加。在这种情况下，您可以暂时将应用程序的读取和写入活动重定向到另一个运行状况良好的 AWS 区域。

MemoryDB 多区域扩缩

随着集群需求的变化，您可能决定通过更改节点类型或 MemoryDB 集群中的分片数量来提升性能或降低成本。扩展 MemoryDB 多区域集群会同时扩缩其中的所有区域集群。MemoryDB 多区域集群支持在线重分片。MemoryDB 多区域集群不支持离线重分片。

您决定重新调节集群的情况包括以下几种：

- 记忆压力

  如果您的区域集群中的节点存在内存压力，您可能会决定进行横向扩展或纵向扩展，以获得更多资源来更好地存储数据和处理请求。

  您可以通过监控以下指标来确定您的节点是否承受内存压力： FreeableMemory、 SwapUsage、 BytesUsedForMemory DB 和 MultiRegionClusterReplicationLag
- CPU 或网络瓶颈

  如果 latency/throughput 问题困扰着您的集群，则可能需要向外扩展或向上扩展以解决问题。

  您可以通过监控以下指标来监控延迟和吞吐量水平：CPUUtilization、NetworkBytesIn、NetworkBytesOut、CurrConnections、 NewConnections、 and MultiRegionClusterReplicationLag。
- 您的集群规模过大

  当前集群的需求使得横向缩减或纵向缩减不会损害性能并能降低成本。

您可以使用以下指标监控集群的使用情况，以确定是否可以安全地缩减或缩小规模： FreeableMemory SwapUsage BytesUsedForMemory、 CPUUtilization、 NetworkBytesIn、 NetworkBytesOut、 CurrConnections、 NewConnections 和 MultiRegionClusterReplicationLag

MemoryDB 多区域集群有两种扩展方式：水平扩缩和垂直扩缩。

- 水平扩缩您通过添加或删除分片来更改 MemoryDB 多区域集群中的分片数量。在线重新分片过程允许在区域集群继续为传入请求提供服务的 in/out 同时进行扩展。

- 纵向扩展通过更改节点类型来调整 MemoryDB 多区域集群的规模。在线垂直扩展允许在区域集群继续为传入请求提供服务的 up/down 同时进行扩展。

扩展默认使用“协调式”更新策略。这意味着要么所有区域集群成功扩展，要么所有区域集群均不扩展。

横向扩展操作也支持“非协调式”更新策略。这意味着某些区域集群可能成功横向扩展，而某些区域集群的横向扩展尝试可能失败。如果某个区域集群横向扩展成功，则所有其他区域集群会继续重试横向扩展，直到所有其他横向扩展也都成功。

如果所有区域集群均未能成功横向扩展，则多区域集群的“非协调式”横向扩展将失败。

> **Note**
>
> 当区域集群在不同时间进行横向扩展时，“非协调式”横向扩展可能导致区域集群之间长时间存在容量不平衡。这可能会导致 MultiRegionClusterReplicationLag 指标增加，而区域集群数据可能会长期存在差异。

MemoryDB 多区域集群的区域集群可以配置不同数量的副本节点，但区域集群中的所有分片必须具有相同数量的副本节点。

如果您要通过缩小或缩小规模 MemoryDB 多区域集群的大小和内存容量，请确保新配置有足够的内存和可用 IPs 空间来存放您的数据、足够的引擎开销，并且区域集群的 MultiRegionClusterReplicationLag 指标在几秒或一分钟范围内。

您可以使用、和 MemoryDB API 水平和垂直扩展 MemoryDB 多区域集群。 AWS 管理控制台 AWS CLI

支持与不支持的命令

支持的命令

> **Note**
>
> - SET 命令当前不支持 EX、PX、EXAT、PXAT 和 KEEPTTL 选项。
> - RESTORE 命令不支持将 TTL 设置为非零值。同样不支持 ABSTTL、IDLETIME 和 FREQ 选项。

数据类型	命令
字符串	SET*、DECR、DECRBY、GET、GETRANGE、SUBSTR、GETDEL、GETSET、INCR、INCRBY、INCRBYFLOAT、MGET、MSET、MSETNX、SETNX、STRLEN、LCS
哈希	HINCRBY、HINCRBYFLOAT、HDEL、HSET、HMSET、HGET、HEXISTS、HLEN、HKEYS、HVALS、HGETALL、HMGET、HSTRLEN、HSETNX、HRANDFIELD、HSCAN
设置	SADD、SREM、SISMEMBER、SMISMEMBER、SCARD、SMEMBERS、SRANDMEMBER、SSCAN、SUNION、SINTERCARD、SINTER、SDIFF、SPOP
有序集合	ZADD、ZINCRBY、ZSCORE、ZMSCORE、ZCARD、ZRANK、ZREVRANK、ZRANGE、ZRANGEBYSCORE、ZRANGEBYLEX、ZREVRANGE、ZREVRANGEBYLEX、ZREVRANGEBYSCORE、ZREMRANGEBYLEX、ZREMRANGEBYSCORE、ZREMRANGEBYRANK、ZUNION、ZINTER、ZINTERCARD、ZDIFF、ZLEXCOUNT、ZCOUNT、ZREM、ZMPOP、ZPOPMIN、ZPOPMAX、ZSCAN、ZRANDMEMBER
通用	SCAN、DEL、UNLINK、DUMP、RESTORE**、EXISTS、KEYS、RANDOMKEY、TYPE

不支持的命令

不支持的命令的通用类别包括：不支持的数据类型（位图、Hyperloglog、列表、地理空间和流）、TTL 相关命令、阻塞命令以及函数相关命令。完整列表如下：

数据类型	命令
字符串	APPEND、GETEX、SETEX、SETRANGE
Bitmap	BITCOUNT、BITFIELD、BITFIELD_RO、BITOP、BITPOS、GETBIT、SETBIT
Hyperloglog	PFADD、PFCOUNT、PFDEBUG、PFMERGE、PFSELFTEST
列表	BLMOVE、BLMPOP、BLPOP、BRPOP、BRPOPLPUSH、LINDEX、LINSERT、LLEN、LMOVE、LMPOP、LPOP、LPOS、PUSH、LPUSHX、LRANGE、LREM、LSET、LTRIM、RPOP、RPOPLPUSH、RPUSH、RPUSHX
设置	SMOVE、SUNIONSTORE、SDIFFSTORE、SINTERSTORE
有序集合	BZMPOP、BZPOPMAX、BZPOPMIN、ZDIFFSTORE、ZINTERSTORE、ZRANGESTORE、ZUNIONSTORE
地理空间	GEOADD、GEODIST、GEOHASH、GEOPOS、GEORADIUS、GEORADIUS_RO、GEORADIUSBYMEMBER、GEORADIUSBYMEMBER_RO、GEOSEARCH、GEOSEARCHSTORE
流	XACK、XADD、XAUTOCLAIM、XCLAIM、XDEL、XLEN、XPENDING、XRANGE、XREAD、XREADGROUP、XREVRANGE、XSETID、XTRIM、XGROUP、XINFO

数据类型	命令
通用	COPY、FLUSHDB、FLUSHALL、MOVE、RENAME、RENAMENX、SORT、SORT_RO、SWAPDB、OBJECT、FUNCTION、FCALL、FCALL_RO、EXPIRE、EXPIREAT、EXPIRETIME、PERSIST、PEXPIRE、PEXPIREAT、PEXPIRETIME、PSETEX、PTTL、TTL

MemoryDB 中的安全性

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将此描述为云的安全性和云中的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为AWS 合规计划合规计划合规计划合的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 MemoryDB 的合规性计划，请参阅按合规计划划分的范围内AWS 服务按合规计划划分。
- 云端安全-您的责任由您使用的 AWS 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、您公司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 MemoryDB 时应用责任共担模式。它说明了如何配置 MemoryDB 以实现您的安全性和合规性目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 MemoryDB 资源。

内容

- MemoryDB 中的数据保护
- MemoryDB 中的身份和访问管理
- 日志记录和监控
- 适用于 MemoryDB 的合规性验证
- MemoryDB 中的基础设施安全性
- 互联网络流量隐私
- MemoryDB 中的服务更新

MemoryDB 中的数据保护

分 AWS 担责任模型适用于中的数据保护。如本模型所述 AWS ，负责保护运行所有内容的全球基础架构 AWS Cloud。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 AWS 服务 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。有关欧洲数据保护的信息，请参阅 AWS Security Blog 上的 AWS Shared Responsibility Model and GDPR 博客文章。

出于数据保护目的，我们建议您保护 AWS 账户 凭证并使用 AWS IAM Identity Center 或 AWS Identity and Access Management (IAM) 设置个人用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 用于 SSL/TLS 与 AWS 资源通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 使用设置 API 和用户活动日志 AWS CloudTrail。有关使用 CloudTrail 跟踪捕获 AWS 活动的信息，请参阅《AWS CloudTrail 用户指南》中的使用跟 CloudTrail 踪。
- 使用 AWS 加密解决方案以及其中的所有默认安全控件 AWS 服务。
- 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 Amazon S3 中的敏感数据。
- 如果您在 AWS 通过命令行界面或 API 进行访问时需要经过 FIPS 140-3 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 140-3 版》https://aws.amazon.com/compliance/fips/。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括您使用控制台、API 或以其他 AWS 服务 方式使用控制台 AWS CLI、API 或时 AWS SDKs。在用于名称的标签或自由格式文本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，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

MemoryDB 中的数据安全性

为了帮助确保数据安全，MemoryDB 和 Amazon EC2 提供了禁止未经授权访问服务器上数据的机制。

MemoryDB 还为集群中的数据提供加密功能：

- 传输中加密可对从一个位置移动到另一个位置的数据进行加密，例如在集群中的节点之间或在集群与应用程序之间移动数据。
- 静态加密可在快照操作期间对事务日志和磁盘上的数据进行加密。

您还可以使用 使用访问控制列表对用户进行身份验证 () ACLs 控制用户对集群的访问。

主题

- MemoryDB 中的静态加密
- MemoryDB 传输中加密（TLS）

- 使用访问控制列表对用户进行身份验证 () ACLs
- 使用 IAM 进行身份验证

MemoryDB 中的静态加密

为了帮助保护您的数据，MemoryDB 和 Amazon S3 提供了不同的方法来限制对集群中的数据的访问。有关更多信息，请参阅MemoryDB 和 Amazon VPC和MemoryDB 中的身份和访问管理。

MemoryDB 静态加密始终通过加密永久数据来提高数据安全性。它对以下方面进行加密：

- 事务日志中的数据
- 同步、快照和交换操作期间的磁盘
- 存储在 Amazon S3 中的快照

MemoryDB 提供默认（托管服务）的静态加密，以及在AWS 密钥管理服务（KMS） 中使用您自己的对称客户托管式根密钥的功能。

在预设情况下，在启用数据分层的集群中存储在 SSD（固态硬盘）上的数据始终加密。

有关传输中加密的信息，请参阅MemoryDB 传输中加密（TLS）

主题

- 使用 AWS KMS 中的客户托管式密钥
- 另请参阅

使用 AWS KMS 中的客户托管式密钥

MemoryDB 支持对称的客户托管式根密钥（KMS 密钥），用于静态加密。客户自主管理型 KMS 密钥是您在自己的 AWS 账户中创建、拥有并管理的加密密钥。有关更多信息，请参阅AWS密钥管理服务开发人员指南中的客户根密钥。必须先在 AWS KMS 中创建密钥，然后才能将其与 MemoryDB 一起使用。

要了解如何创建 AWS KMS 根密钥，请参阅 AWS Key Management Service 开发人员指南中的创建密钥。

MemoryDB 允许您与 AWS KMS 集成。有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的使用授权。无需任何客户操作即可实现 MemoryDB 与 AWS KMS 的集成。

`kms:ViaService` 条件键将 AWS KMS 密钥的使用限制为来自指定的 AWS 服务的请求。要将 `kms:ViaService` 与 MemoryDB 结合使用，请将两个 ViaService 名称包含在条件键值中：`memorydb.amazon_region.amazonaws.com`。有关更多信息，请参阅 kms:ViaService。

您可以使用 AWS CloudTrail 来跟踪 MemoryDB 代表您向 AWS Key Management Service 发送的请求。对 AWS Key Management Service 发出的与客户自主管理型密钥相关的所有 API 调用都具有相应的 CloudTrail 日志。您还可以通过调用 ListGrants KMS API 调用来查看 MemoryDB 创建的授权。

使用客户托管式密钥对集群进行加密后，集群的所有快照都将按如下方式进行加密：

- 使用与集群关联的客户托管式密钥对每日自动快照进行加密。
- 删除集群时创建的最终快照也使用与集群关联的客户托管式密钥进行加密。
- 默认情况下，使用与集群关联的密钥对手动创建的快照进行加密。您可以通过选择其他客户自主管理型密钥来覆此行为。
- 复制快照将默认使用与源快照关联的客户托管式密钥。您可以通过选择其他客户自主管理型密钥来覆此行为。

> **Note**
>
> - 将快照导出到所选的 Amazon S3 存储桶时，无法使用客户托管式密钥。但是，导出到 Amazon S3 的所有快照都将使用服务器端加密进行加密。您可以选择将快照文件复制到新的 S3 对象并使用客户托管式 KMS 密钥进行加密，将文件复制到使用 KMS 密钥通过默认加密设置的另一个 S3 存储桶，或者更改文件本身中的加密选项。
> - 对于未使用客户托管式密钥进行加密的手动创建快照，您还可以使用客户托管式密钥对其进行加密。使用此选项，即使未在原始集群上加密数据，也可以使用 KMS 密钥对存储在 Amazon S3 中的快照文件进行加密。
>
> 从快照还原允许您从可用的加密选项中进行选择，类似于创建新集群时可用的加密选项。

- 如果删除密钥或禁用密钥并为用于加密集群的密钥撤销授权，则集群将变得不可恢复。换句话说，复制组在硬件故障后无法修改或恢复。AWSKMS 在至少七天的等待期限之后才会删除根密钥。删除密钥后，您可以使用其他客户托管式密钥创建快照以用于存档目的。
- 自动密钥轮换将保留 AWS KMS 根密钥的属性，因此轮换不会影响您访问 MemoryDB 数据的能力。加密 MemoryDB 集群不支持手动密钥轮换，手动密钥轮换涉及创建新的根密钥和将任何参考更新到旧密钥。要了解详情，请参阅 AWS 密钥管理服务开发人员指南中的轮换客户根密钥。
- 使用 KMS 密钥加密 MemoryDB 集群需要每个集群具有一个授权。在集群的整个生命周期中使用此授权。此外，在快照创建期间使用每个快照一个授权。在创建快照后，此授权将停用。
- 有关 AWS KMS 授权和限制的更多信息，请参阅 AWS 密钥管理服务开发人员指南中的配额。

另请参阅

- MemoryDB 传输中加密（TLS）
- MemoryDB 和 Amazon VPC
- MemoryDB 中的身份和访问管理

MemoryDB 传输中加密（TLS）

为了帮助确保数据安全，MemoryDB 和 Amazon EC2 提供了禁止未经授权访问服务器上数据的机制。通过传输中加密功能，MemoryDB 为您提供了在不同位置之间移动数据时用来保护数据的工具。例如，您可能从集群中的主节点向只读副本节点移动数据，或在集群与应用程序之间移动数据。

主题

- 传输中加密概览
- 另请参阅

传输中加密概览

MemoryDB 传输中加密是一项在数据最脆弱的时候（从一个位置传输到另一个位置时）提高数据安全性的特征。

MemoryDB 传输中加密可实现以下功能：

- 加密连接 – 服务器和客户端连接都采用传输层安全性协议（TLS）加密。
- 加密复制 – 对在主节点与副本节点之间移动的数据进行加密。
- 服务器身份验证 – 客户端可通过身份验证确定它们连接到正确的服务器。

自 2023 年 7 月 20 日起，TLS 1.2 是新集群和现有集群的最低支持版本。使用此链接 了解有关 TLS 1.2 的更多信息，网址：AWS。

有关连接到 MemoryDB 集群的更多信息，请参阅 使用 redis-cli 连接到 MemoryDB 节点。

另请参阅

- MemoryDB 中的静态加密
- 使用访问控制列表（ACL）对用户进行身份验证

- MemoryDB 和 Amazon VPC
- MemoryDB 中的身份和访问管理

使用访问控制列表对用户进行身份验证 () ACLs

您可以使用访问控制列表 (ACLs) 对用户进行身份验证。

ACLs 使您能够通过对用户进行分组来控制集群访问权限。这些访问控制列表旨在作为一种集群访问组织方式。

使用 ACLs，您可以创建用户并使用访问字符串为其分配特定权限，如下一节所述。您可以将用户分配到与特定角色（管理员、人力资源）一致的访问控制列表，然后将这些访问控制列表部署到一个或多个 MemoryDB 集群。这样，您可以在使用相同 MemoryDB 集群的客户端之间建立安全边界，并阻止客户端彼此访问数据。

ACLs 旨在支持在 Redis OSS 6 中引入 ACL。 ACLs 与 MemoryDB 集群一起使用时，存在一些限制：

- 不能在访问字符串中指定密码。您可以通过CreateUser或UpdateUser呼叫来设置密码。
- 对于用户权利，您可以将 `on` 和 `off` 作为访问字符串的一部分进行传递。如果两者在访问字符串中都未指定，则将为用户分配 `off` 并且用户没有访问集群的权限。
- 您不能使用已禁止命令。如果您指定了已禁止命令，则会引发异常。有关此类命令列表，请参阅 受限命令。
- 您不能将 `reset` 命令作为访问字符串的一部分。您可以使用 API 参数指定密码，MemoryDB 会管理这些密码。因此，您不能使用 `reset`，因为此命令会移除用户的所有密码。
- Redis OSS 6 引入了 ACL LIST 命令。此命令将返回用户列表以及应用于各用户的 ACL 规则。MemoryDB 支持 `ACL LIST` 命令，但不像 Redis OSS 那样支持密码哈希。使用 MemoryDB，您可以使用该DescribeUsers操作来获取类似的信息，包括访问字符串中包含的规则。但是，DescribeUsers无法检索用户密码。

  MemoryDB 支持的其他只读命令包括 ACL WHOAMI、ACL USERS 和 ACL CAT。MemoryDB 不支持任何其他基于写入的 ACL 命令。

下文将详细介绍如何 ACLs 与 MemoryDB 配合使用。

主题

- 使用访问字符串指定权限

- 向量搜索功能
- ACLs 向集群申请 MemoryDB

使用访问字符串指定权限

要指定对 MemoryDB 集群的权限，您可以使用或创建访问字符串并将其分配给用户。 AWS CLI AWS 管理控制台

根据定义，访问字符串是指应用于用户的、以空格分隔的规则列表。它们定义了用户可以执行的命令以及用户可以对其进行操作的密钥。要执行命令，用户必须有权访问正在执行的命令以及命令访问的所有密钥。规则从左到右累积应用，如果提供的字符串中存在冗余，则可以使用更简单的字符串代替提供的字符串。

有关 ACL 规则的语法的信息，请参阅 ACL。

在以下示例中，访问字符串表示具有所有可用密钥和命令访问权限的活动用户。

```
on ~* &* +@all
```

访问字符串语法分解如下：

- on – 用户是活动用户。
- ~* – 具有对所有可用密钥的访问权限。
- &* – 具有对所有发布/订阅频道的访问权限。
- +@all – 具有对所有可用命令的访问权限。

上述设置的限制性最小。您可以修改这些设置以使其更加安全。

在以下示例中，访问字符串表示一个用户，其访问权限限于对以“app::”键空间开头的键进行读取访问

```
on ~app::* -@all +@read
```

您可以通过列出用户有权访问的命令来进一步优化这些权限：

+*command1* – 用户对命令的访问被限制为 *command1*。

+@category – 用户的访问被限制为某个类别的命令。

有关向用户分配访问字符串的信息，请参阅 使用控制台和 CLI 创建用户和访问控制列表。

如果要将现有工作负载迁移到 MemoryDB，则可以通过调用 ACL LIST（不包括用户和任何哈希密码）来检索访问字符串。

向量搜索功能

对于向量搜索，所有搜索命令都属于该 @search 类别，而现有类别 @read、@write、@fast 和 @slow 则会更新为包括搜索命令。如果用户无权访问某个类别，那么该用户也无权访问该类别中的任何命令。例如，如果用户无权访问 @search，那么该用户也无法执行任何与搜索相关的命令。

下表指示搜索命令到相应类别的映射。

VSS 命令	@read	@write	@fast	@slow
FT.CREATE		Y	Y	
FT.DROPINDEX		Y	Y	
FT.LIST	Y			Y
FT.INFO	Y		Y	
FT.SEARCH	Y			Y
FT.AGGREGATE	Y			Y
FT.PROFILE	Y			Y
FT.ALIASADD		Y	Y	
FT.ALIASDEL		Y	Y	
FT.ALIASUPDATE		Y	Y	

VSS 命令	@read	@write	@fast	@slow
FT._ALIASLIST	Y			Y
FT.EXPLAIN	Y		Y	
FT.EXPLAINCLI	Y		Y	
FT.CONFIG	Y		Y	

ACLs 向集群申请 MemoryDB

要使用 MemoryDB ACLs，您需要执行以下步骤：

1. 创建一个或多个用户。
2. 创建 ACL 并将用户添加到此列表中。
3. 将 ACL 分配到集群。

下方详细地说明了这些步骤。

主题

- 使用控制台和 CLI 创建用户和访问控制列表
- 使用控制台和 CLI 管理访问控制列表
- 将访问控制列表分配到集群

使用控制台和 CLI 创建用户和访问控制列表

用户的用户信息是 ACLs 用户名，也可以是密码和访问字符串（可选）。访问字符串提供对密钥和命令的权限级别。用户名对于用户是唯一的，是传递给引擎的内容。

确保您提供的 ACL 符合用户组的预期目的。例如，如果您创建一个名为 `Administrators` 的 ACL，则添加到该组的任何用户都应将其访问字符串设置为对密钥和命令具有完全访问权限。对于 `e-commerce` ACL 中的用户，您可以将其访问字符串设置为只读访问。

MemoryDB 为每个账户和用户名 "default" 自动配置一个默认用户。它未与任何集群关联，除非明确添加到 ACL 中。您无法修改或删除该用户。该用户旨在与以前 Redis OSS 版本的默认行为兼容，并具有一个访问字符串，允许它调用所有命令并访问所有密钥。

将为每个包含默认用户的账户创建一个不可变“开放访问”ACL。这是默认用户可加入的唯一的 ACL。当您创建一个集群时，必须选择一个与该集群关联的 ACL。虽然您可以选择对默认用户应用“开放访问”ACL，但我们强烈建议为权限仅限于其业务需求的用户创建 ACL。

未启用 TLS 的集群必须使用“开放访问”ACL 提供开放身份验证。

ACLs 可以在没有用户的情况下创建。空 ACL 无权访问集群，只能与启动 TLS 的集群关联。

创建用户时，最多可以设置两个密码。修改密码时，将保持与集群之间的所有现有连接。

特别是，在用于 MemoryDB 时，请注意以下用户密码限制： ACLs

- 密码必须是 16-128 个可打印字符。
- 不允许使用以下非字母数字字符：, "" / @。

使用控制台和 CLI 管理用户

创建用户（控制台）

在控制台上创建用户

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在左侧导航窗格中，选择用户。
3. 选择创建用户
4. 在创建用户页面上，输入名称。

   集群命名约束如下：

   - 必须包含 1 – 40 个字母数字字符或连字符。
   - 必须以字母开头。
   - 不能包含两个连续连字符。
   - 不能以连字符结束。
5. 在密码下，最多可以输入两个密码。
6. 在访问字符串下，输入访问字符串。访问字符串设置允许用户使用的密钥和命令的权限级别。

7. 对于标签，您可以选择应用标签来搜索和筛选用户或跟踪您的 AWS 费用。
8. 选择创建。

使用创建用户 AWS CLI

使用 CLI 创建用户

- 使用 create-user 命令可创建新的用户。

  对于 Linux、macOS 或 Unix：

```
aws memorydb create-user \
  --user-name user-name-1 \
  --access-string "~objects:* ~items:* ~public:*" \
  --authentication-mode \
        Passwords="abc",Type=password
```

  对于 Windows：

```
aws memorydb create-user ^
  --user-name user-name-1 ^
  --access-string "~objects:* ~items:* ~public:*" ^
  --authentication-mode \
        Passwords="abc",Type=password
```

修改用户（控制台）

在控制台上修改用户

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在左侧导航窗格中，选择用户。
3. 选择要修改的用户旁边的单选按钮，然后选择操作 -> 修改
4. 如果要修改密码，请选择修改密码单选按钮。请注意，如设有两个密码，则修改其中之一时必须同时输入两个密码。
5. 如果要更新访问字符串，则请输入新的访问字符串。

6. 选择 Modify(修改)。

使用修改用户 AWS CLI

使用 CLI 修改用户

1. 使用 update-user 命令修改用户。
2. 当修改用户时，与该用户关联的访问控制列表以及与 ACL 关联的任何集群将进行更新。将会保持所有现有连接。示例如下。

   对于 Linux、macOS 或 Unix：

   ```
   aws memorydb update-user \
     --user-name user-name-1 \
     --access-string "~objects:* ~items:* ~public:*"
   ```

   对于 Windows：

   ```
   aws memorydb update-user ^
     --user-name user-name-1 ^
     --access-string "~objects:* ~items:* ~public:*"
   ```

查看用户详细信息（控制台）

在控制台查看用户详细信息

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在左侧导航窗格中，选择用户。
3. 在用户名下选择用户或使用搜索框查找用户。
4. 在用户设置下，查看用户的访问字符串、密码数量、状态和 Amazon 资源名称（ARN）。
5. 在访问控制列表（ACL）下，查看用户所属的 ACL。
6. 在标签下，查看与用户关联的任何标签。

使用查看用户详细信息 AWS CLI

使用 describe-users 命令查看用户的详细信息。

```
aws memorydb describe-users \
  --user-name my-user-name
```

删除 用户（控制台）

在控制台上删除用户

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在左侧导航窗格中，选择用户。
3. 选择要修改的用户旁边的单选按钮，然后选择操作 -> 删除
4. 要进行确认，请在确认文本框中输入 `delete`，然后选择确认。
5. 要取消，请选择取消。

使用删除用户 AWS CLI

使用 CLI 删除用户

- 使用 delete-user 命令删除用户。

  此账户将被删除并从其所属的任何访问控制列表中移除。示例如下：

  对于 Linux、macOS 或 Unix：

  ```
  aws memorydb delete-user \
    --user-name user-name-2
  ```

  对于 Windows：

  ```
  aws memorydb delete-user ^
    --user-name user-name-2
  ```

使用控制台和 CLI 管理访问控制列表

您可以创建访问控制列表来组织和控制用户对一个或多个集群的访问，如下所示。

使用以下步骤通过控制台管理访问控制列表。

创建访问控制列表（ACL）（控制台）

使用控制台创建访问控制列表

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在左侧导航窗格中，选择访问控制列表（ACL）。
3. 选择创建 ACL。
4. 在创建访问控制列表（ACL）页面上，输入 ACL 名称。

   集群命名约束如下：

   - 必须包含 1 – 40 个字母数字字符或连字符。
   - 必须以字母开头。
   - 不能包含两个连续连字符。
   - 不能以连字符结束。
5. 在选定用户下，执行以下操作之一：

   a. 通过选择创建用户创建新用户

   b. 添加用户，方法是选择管理，再从管理用户对话框中选择用户，然后选择选择。
6. 对于标签，您可以选择应用标签来搜索和筛选您的费用 ACLs 或跟踪您的 AWS 费用。
7. 选择创建。

使用创建访问控制列表 (ACL) AWS CLI

通过 CLI 使用以下过程创建访问控制列表。

使用 CLI 创建新 ACL 并添加用户

- 使用 create-acl 命令创建 ACL。

  对于 Linux、macOS 或 Unix：

```
aws memorydb create-acl \
  --acl-name "new-acl-1" \
  --user-names "user-name-1" "user-name-2"
```

对于 Windows：

```
aws memorydb create-acl ^
  --acl-name "new-acl-1" ^
  --user-names "user-name-1" "user-name-2"
```

修改访问控制列表（ACL）（控制台）

使用控制台修改访问控制列表

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在左侧导航窗格中，选择访问控制列表（ACL）。
3. 选择要修改的 ACL，然后选择 修改
4. 在修改页面的所选用户下，执行下列操作之一：
   a. 通过选择创建用户添加到 ACL。
   b. 添加或移除用户，方法是选择管理，再从管理用户对话框中选择或取消选中用户，然后选择选择。
5. 在创建访问控制列表（ACL）页面上，输入 ACL 名称。

   集群命名约束如下：

   - 必须包含 1 – 40 个字母数字字符或连字符。
   - 必须以字母开头。
   - 不能包含两个连续连字符。
   - 不能以连字符结束。
6. 在选定用户下，执行以下操作之一：
   a. 通过选择创建用户创建新用户
   b. 添加用户，方法是选择管理，再从管理用户对话框中选择用户，然后选择选择。

7. 选择修改保存更改，或选择取消放弃更改。

使用修改访问控制列表 (ACL) AWS CLI

使用 CLI 通过添加新用户或移除当前成员来修改 ACL

- 使用 update-acl 命令修改 ACL。

  对于 Linux、macOS 或 Unix：

```
aws memorydb update-acl --acl-name new-acl-1 \
--user-names-to-add user-name-3 \
--user-names-to-remove user-name-2
```

  对于 Windows：

```
aws memorydb update-acl --acl-name new-acl-1 ^
--user-names-to-add user-name-3 ^
--user-names-to-remove user-name-2
```

> **Note**
>
> 此命令将结束属于从 ACL 中移除的用户的任何打开的连接。

查看访问控制列表（ACL）详细信息（控制台）

在控制台上查看 ACL 详细信息

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在左侧导航窗格上，选择访问控制列表（ACL）。
3. 在 ACL 名称下选择 ACL 或使用搜索框查找 ACL。
4. 在用户下，查看与 ACL 关联的用户列表。
5. 在关联集群下，查看 ACL 所属的集群。
6. 在标签下，查看与 ACL 关联的任何标签。

使用查看访问控制列表 (ACL) AWS CLI

使用 describe-acls 命令查看 ACL 详细信息。

```
aws memorydb describe-acls \
  --acl-name test-group
```

删除访问控制列表（ACL）（控制台）

使用控制台删除访问控制列表

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在左侧导航窗格中，选择访问控制列表（ACL）。
3. 选择要修改的 ACL，然后选择删除
4. 在删除页面上，在确认框中输入 `delete`，并选择删除或取消，以避免删除 ACL。

将删除 ACL 本身，而不是属于该组的用户。

使用删除访问控制列表 (ACL) AWS CLI

使用 CLI 删除 ACL

- 使用 delete-acl 命令删除 ACL。

  对于 Linux、macOS 或 Unix：

  ```
  aws memorydb delete-acl /
     --acl-name
  ```

  对于 Windows：

  ```
  aws memorydb delete-acl ^
     --acl-name
  ```

  上述示例将返回以下响应。

  ```
  aws memorydb delete-acl --acl-name "new-acl-1"
  {
  ```

```
    "ACLName": "new-acl-1",
    "Status": "deleting",
    "EngineVersion": "6.2",
    "UserNames": [
        "user-name-1",
        "user-name-3"
    ],
    "clusters": [],
    "ARN":"arn:aws:memorydb:us-east-1:493071037918:acl/new-acl-1"
}
```

将访问控制列表分配到集群

创建 ACL 并添加用户后，实施的最后一步 ACLs 是将 ACL 分配给集群。

使用控制台将访问控制列表分配到集群

要使用向集群添加 ACL AWS 管理控制台，请参阅创建 MemoryDB 集群。

为群集分配访问控制列表使用 AWS CLI

以下 AWS CLI 操作创建一个启用传输中加密 (TLS) 且acl-name参数值为的集群*my-acl-name*。用已存在的子网组替换子网组 subnet-group。

关键参数

- **--engine-version** – 必须是 6.2。
- **--tls-enabled** – 用于身份验证和关联 ACL。
- **--acl-name** – 此值提供由具有集群指定访问权限的用户组成的访问控制列表。

对于 Linux、macOS 或 Unix：

```
aws memorydb create-cluster \
    --cluster-name "new-cluster" \
    --description "new-cluster" \
    --engine-version "6.2" \
    --node-type db.r6g.large \
    --tls-enabled \
    --acl-name "new-acl-1" \
    --subnet-group-name "subnet-group"
```

对于 Windows：

```
aws memorydb create-cluster ^
    --cluster-name "new-cluster" ^
    --cluster-description "new-cluster" ^
    --engine-version "6.2" ^
    --node-type db.r6g.large ^
    --tls-enabled ^
    --acl-name "new-acl-1" ^
    --subnet-group-name "subnet-group"
```

以下 AWS CLI 操作修改了启用传输中加密 (TLS) 且acl-name参数值new-acl-2为的集群。

对于 Linux、macOS 或 Unix：

```
aws memorydb update-cluster \
    --cluster-name cluster-1 \
    --acl-name "new-acl-2"
```

对于 Windows：

```
aws memorydb update-cluster ^
    --cluster-name cluster-1 ^
    --acl-name "new-acl-2"
```

使用 IAM 进行身份验证

主题

- 概述
- 限制
- 设置
- 连接

概述

使用 IAM 身份验证，当您的集群配置为使用 Valkey 或 Redis OSS 版本 7 或更高版本时，您可以使用 AWS IAM 身份验证与 MemoryDB 的连接。这使您可以增强安全模型并简化许多管理安全任务。通过 IAM 身份验证，您可以为每个单独的 MemoryDB 集群和 MemoryDB 用户配置精细的访问控制，并遵

循最低权限权限原则。MemoryDB 的 IAM 身份验证的工作原理是在 AUTH 或 HELLO 命令中提供有效期很短的 IAM 身份验证令牌，而不是有效期很长的 MemoryDB 用户密码。有关 IAM 身份验证令牌的更多信息，请参阅《 AWS 通用参考指南》中的签名版本 4 签名流程和下面的代码示例。

您可以使用 IAM 身份及其关联策略进一步限制 Valkey 或 Redis OSS 访问权限。您还可以直接向来自联合身份提供商的用户授予对 MemoryDB 集群的访问权限。

要 AWS 将 IAM 与 MemoryDB 配合使用，您首先需要创建一个身份验证模式设置为 IAM 的 MemoryDB 用户，然后才能创建或重复使用 IAM 身份。IAM 身份需要关联策略才能向 MemoryDB 集群和 MemoryDB 用户授予 `memorydb:Connect` 操作权限。配置完成后，您可以使用 IAM 用户或角色的 AWS 证书创建 IAM 身份验证令牌。最后，在连接到 MemoryDB 集群节点时，您需要在 Valkey 或 Redis OSS 客户端中提供有效期较短的 IAM 身份验证令牌作为密码。支持凭证提供程序的客户端可以为每个新连接自动生成临时凭证。MemoryDB 将对启用 IAM 的 MemoryDB 用户的连接请求执行 IAM 身份验证，并将通过 IAM 验证连接请求。

限制

使用 IAM 身份验证时，以下限制适用：

- 使用 Valkey 或 Redis OSS 引擎版本 7.0 或更高版本时，IAM 身份验证可用。
- IAM 身份验证令牌的有效期为 15 分钟。对于长时间的连接，建议使用支持凭证提供程序接口的 Redis OSS 客户端。
- 经过 IAM 身份验证的 MemoryDB 连接将在 12 小时后自动断开。通过使用新 IAM 身份验证令牌发送 AUTH 或 HELLO 命令，可以将连接延长 12 小时。
- `MULTI EXEC` 命令不支持 IAM 身份验证。
- 目前，IAM 身份验证并不支持所有的全局条件上下文键。有关全局条件上下文键的更多信息，请参阅《IAM 用户指南》中的 AWS 全局条件上下文键。

设置

要设置 IAM 身份验证，请执行以下操作：

1. 创建集群

```
aws memorydb create-cluster \
    --cluster-name cluster-01 \
    --description "MemoryDB IAM auth application"
    --node-type db.r6g.large \
    --engine-version 7.0 \
```

```
    --acl-name open-access
```

2. 为您的角色创建 IAM 信任政策文档，如下所示，允许您的账户承担新角色。将策略保存到名为 trust-policy.json 的文件中。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
        "Action": "sts:AssumeRole"
    }
}
```

3. 创建 IAM 策略文档，如下所示。将策略保存到名为 policy.json 的文件中。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect" : "Allow",
      "Action" : [
        "memorydb:connect"
      ],
      "Resource" : [
        "arn:aws:memorydb:us-east-1:123456789012:cluster/cluster-01",
        "arn:aws:memorydb:us-east-1:123456789012:user/iam-user-01"
      ]
    }
  ]
}
```

4. 创建 IAM 角色。

```
aws iam create-role \
  --role-name "memorydb-iam-auth-app" \
  --assume-role-policy-document file://trust-policy.json
```

5. 创建 IAM 策略。

```
aws iam create-policy \
  --policy-name "memorydb-allow-all" \
  --policy-document file://policy.json
```

6. 向角色附加 IAM 策略。

```
aws iam attach-role-policy \
 --role-name "memorydb-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/memorydb-allow-all"
```

7. 创建启用 IAM 的新用户。

```
aws memorydb create-user \
  --user-name iam-user-01 \
  --authentication-mode Type=iam \
  --access-string "on ~* +@all"
```

8. 创建 ACL 并附加用户。

```
aws memorydb create-acl \
  --acl-name iam-acl-01 \
  --user-names iam-user-01

aws memorydb update-cluster \
  --cluster-name cluster-01 \
  --acl-name iam-acl-01
```

连接

使用令牌作为密码进行连接

您首先需要使用 AWS SigV4 预签名请求生成有效期较短的 IAM 身份验证令牌。之后，您需要在连接到 MemoryDB 集群时提供 IAM 身份验证令牌作为密码，如下例所示。

```
String userName = "insert user name"
String clusterName = "insert cluster name"
String region = "insert region"

// Create a default AWS Credentials provider.
```

```
// This will look for AWS credentials defined in environment variables or system
 properties.
AWSCredentialsProvider awsCredentialsProvider = new
 DefaultAWSCredentialsProviderChain();

// Create an IAM authentication token request and signed it using the AWS credentials.
// The pre-signed request URL is used as an IAM authentication token for MemoryDB.
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userName,
 clusterName, region);
String iamAuthToken =
 iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());

// Construct URL with IAM Auth credentials provider
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
    .withSsl(ssl)
    .withAuthentication(userName, iamAuthToken)
    .build();

// Create a new Lettuce client
RedisClusterClient client = RedisClusterClient.create(redisURI);
client.connect();
```

以下为 IAMAuthTokenRequest 的定义。

```
public class IAMAuthTokenRequest {
    private static final HttpMethodName REQUEST_METHOD = HttpMethodName.GET;
    private static final String REQUEST_PROTOCOL = "http://";
    private static final String PARAM_ACTION = "Action";
    private static final String PARAM_USER = "User";
    private static final String ACTION_NAME = "connect";
    private static final String SERVICE_NAME = "memorydb";
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final String userName;
    private final String clusterName;
    private final String region;

    public IAMAuthTokenRequest(String userName, String clusterName, String region) {
        this.userName = userName;
        this.clusterName = clusterName;
        this.region = region;
```

```
    }

    public String toSignedRequestUri(AWSCredentials credentials) throws
 URISyntaxException {
        Request<Void> request = getSignableRequest();
        sign(request, credentials);
        return new URIBuilder(request.getEndpoint())
            .addParameters(toNamedValuePair(request.getParameters()))
            .build()
            .toString()
            .replace(REQUEST_PROTOCOL, "");
    }

    private <T> Request<T> getSignableRequest() {
        Request<T> request  = new DefaultRequest<>(SERVICE_NAME);
        request.setHttpMethod(REQUEST_METHOD);
        request.setEndpoint(getRequestUri());
        request.addParameters(PARAM_ACTION, Collections.singletonList(ACTION_NAME));
        request.addParameters(PARAM_USER, Collections.singletonList(userName));
        return request;
    }

    private URI getRequestUri() {
        return URI.create(String.format("%s%s/", REQUEST_PROTOCOL, clusterName));
    }

    private <T> void sign(SignableRequest<T> request, AWSCredentials credentials) {
        AWS4Signer signer = new AWS4Signer();
        signer.setRegionName(region);
        signer.setServiceName(SERVICE_NAME);

        DateTime dateTime = DateTime.now();
        dateTime = dateTime.plus(Duration.standardSeconds(TOKEN_EXPIRY_SECONDS));

        signer.presignRequest(request, credentials, dateTime.toDate());
    }

    private static List<NameValuePair> toNamedValuePair(Map<String, List<String>> in) {
        return in.entrySet().stream()
            .map(e -> new BasicNameValuePair(e.getKey(), e.getValue().get(0)))
            .collect(Collectors.toList());
    }
}
```

使用凭证提供程序进行连接

以下代码显示了如何使用 IAM 身份验证凭证提供程序通过 MemoryDB 进行身份验证。

```
String userName = "insert user name"
String clusterName = "insert cluster name"
String region = "insert region"

// Create a default AWS Credentials provider.
// This will look for AWS credentials defined in environment variables or system
 properties.
AWSCredentialsProvider awsCredentialsProvider = new
 DefaultAWSCredentialsProviderChain();

// Create an IAM authentication token request. Once this request is signed it can be
 used as an
// IAM authentication token for MemoryDB.
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userName,
 clusterName, region);

// Create a credentials provider using IAM credentials.
RedisCredentialsProvider redisCredentialsProvider = new
 RedisIAMAuthCredentialsProvider(
    userName, iamAuthTokenRequest, awsCredentialsProvider);

// Construct URL with IAM Auth credentials provider
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
    .withSsl(ssl)
    .withAuthentication(redisCredentialsProvider)
    .build();

// Create a new Lettuce cluster client
RedisClusterClient client = RedisClusterClient.create(redisURI);
client.connect();
```

以下是 Lettuce 集群客户端的示例，该客户端将封装在凭证提供程序 IAMAuthTokenRequest 中，以便在需要时自动生成临时证书。

```
public class RedisIAMAuthCredentialsProvider implements RedisCredentialsProvider {
    private static final long TOKEN_EXPIRY_SECONDS = 900;
```

```
    private final AWSCredentialsProvider awsCredentialsProvider;
    private final String userName;
    private final IAMAuthTokenRequest iamAuthTokenRequest;
    private final Supplier<String> iamAuthTokenSupplier;

    public RedisIAMAuthCredentialsProvider(String userName,
        IAMAuthTokenRequest iamAuthTokenRequest,
        AWSCredentialsProvider awsCredentialsProvider) {
        this.userName = userName;
        this.awsCredentialsProvider = awsCredentialsProvider;
        this.iamAuthTokenRequest = iamAuthTokenRequest;
        this.iamAuthTokenSupplier =
Suppliers.memoizeWithExpiration(this::getIamAuthToken, TOKEN_EXPIRY_SECONDS,
TimeUnit.SECONDS);
    }

    @Override
    public Mono<RedisCredentials> resolveCredentials() {
        return Mono.just(RedisCredentials.just(userName, iamAuthTokenSupplier.get()));
    }

    private String getIamAuthToken() {
        return
iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());
    }
```

MemoryDB 中的身份和访问管理

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制可以通过身份验证（登录）和授权（具有权限）使用 MemoryDB 资源的人员。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

- 受众
- 使用身份进行身份验证
- 使用策略管理访问
- MemoryDB 如何与 IAM 协同工作
- 适用于 MemoryDB 的基于身份的策略示例

- 排查 MemoryDB 的身份和访问权限
- 访问控制
- 管理 MemoryDB 资源的访问权限的概述

受众

您的使用方式 AWS Identity and Access Management (IAM) 因您的角色而异：

- 服务用户：如果您无法访问功能，请从管理员处请求权限（请参阅排查 MemoryDB 的身份和访问权限）
- 服务管理员：确定用户访问权限并提交权限请求（请参阅MemoryDB 如何与 IAM 协同工作）
- IAM 管理员：编写用于管理访问权限的策略（请参阅适用于 MemoryDB 的基于身份的策略示例）

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份进行身份验证 AWS 账户根用户，或者通过担任 IAM 角色进行身份验证。

您可以使用来自身份源的证书 AWS IAM Identity Center（例如（IAM Identity Center）、单点登录身份验证或 Google/Facebook 证书，以联合身份登录。有关登录的更多信息，请参阅《AWS 登录 用户指南》中的如何登录您的 AWS 账户。

对于编程访问，AWS 提供 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的适用于 API 请求的AWS 签名版本 4。

AWS 账户 root 用户

创建时 AWS 账户，首先会有一个名为 AWS 账户 root 用户的登录身份，该身份可以完全访问所有资源 AWS 服务 和资源。我们强烈建议不要使用根用户进行日常任务。有关需要根用户凭证的任务，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户使用与身份提供商的联合身份验证才能 AWS 服务 使用临时证书进行访问。

联合身份是指来自您的企业目录、Web 身份提供商的用户 Directory Service ，或者 AWS 服务 使用来自身份源的凭据进行访问的用户。联合身份代入可提供临时凭证的角色。

要集中管理访问权限，建议使用。 AWS IAM Identity Center有关更多信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

IAM 用户和群组

IAM 用户是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅 IAM 用户指南中的要求人类用户使用身份提供商的联合身份验证才能 AWS 使用临时证书进行访问。

IAM 组指定一组 IAM 用户，便于更轻松地对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的 IAM 用户使用案例。

IAM 角色

IAM 角色是具有特定权限的身份，可提供临时凭证。您可以通过从用户切换到 IAM 角色（控制台）或调用 AWS CLI 或 AWS API 操作来代入角色。有关更多信息，请参阅《IAM 用户指南》中的担任角色的方法。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon EC2 上运行的应用程序非常有用。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略定义了与身份或资源关联时的权限。 AWS 在委托人提出请求时评估这些政策。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概述。

管理员使用策略，通过定义哪个主体可以在什么条件下对哪些资源执行哪些操作来指定谁有权访问什么。

默认情况下，用户和角色没有权限。IAM 管理员创建 IAM 策略并将其添加到角色中，然后用户可以担任这些角色。IAM 策略定义权限，与执行操作所用的方法无关。

基于身份的策略

基于身份的策略是您附加到身份（用户、组或角色）的 JSON 权限策略文档。这些策略控制身份可以执行什么操作、对哪些资源执行以及在什么条件下执行。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

基于身份的策略可以是内联策略（直接嵌入到单个身份中）或托管策略（附加到多个身份的独立策略）。要了解如何在托管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。示例包括 IAM 角色信任策略和 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。您必须在基于资源的策略中指定主体。

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 IAM 中的 AWS 托管策略。

其他策略类型

AWS 支持其他策略类型，这些策略类型可以设置更常见的策略类型授予的最大权限：

- 权限边界 – 设置基于身份的策略可以授予 IAM 实体的最大权限。有关更多信息，请参阅《 IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCPs)-在中指定组织或组织单位的最大权限 AWS Organizations。有关更多信息，请参阅《AWS Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs)-设置账户中资源的最大可用权限。有关更多信息，请参阅《AWS Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略 – 在为角色或联合用户创建临时会话时，作为参数传递的高级策略。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。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

MemoryDB 如何与 IAM 协同工作

在使用 IAM 管理对 MemoryDB 的访问之前，您应该了解哪些 IAM 功能可用于 MemoryDB。

可以与 MemoryDB 一起使用的 IAM 特征

IAM 功能	MemoryDB 支持
基于身份的策略	是
基于资源的策略	否

IAM 功能	MemoryDB 支持
策略操作	是
策略资源	是
策略条件键	支持
ACLs	是
ABAC（策略中的标签）	是
临时凭证	是
主体权限	是
服务角色	是
服务关联角色	是

要全面了解 MemoryDB 和其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的AWS 服务。

适用于 MemoryDB 的基于身份的策略

支持基于身份的策略：是

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》中的使用客户管理型策略定义自定义 IAM 权限。

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。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

适用于 MemoryDB 的基于身份的策略示例

要查看 MemoryDB 基于身份的策略的示例，请参阅 适用于 MemoryDB 的基于身份的策略示例。

MemoryDB 内基于资源的策略

支持基于资源的策略：否

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 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。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户访问，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

适用于 MemoryDB 的策略操作

支持策略操作：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。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。在策略中包含操作以授予执行关联操作的权限。

要查看 MemoryDB 操作的列表，请参阅《服务授权参考》中的 MemoryDB 定义的操作。

MemoryDB 中的策略操作在操作前使用以下前缀：

```
MemoryDB
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "MemoryDB:action1",
      "MemoryDB:action2"
         ]
```

您也可以使用通配符（*）指定多个操作。例如，要指定以单词 `Describe` 开头的所有操作，包括以下操作：

```
"Action": "MemoryDB:Describe*"
```

要查看 MemoryDB 基于身份的策略的示例，请参阅 适用于 MemoryDB 的基于身份的策略示例。

适用于 MemoryDB 的策略资源

支持策略资源：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。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。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于不支持资源级权限的操作，请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

要查看 MemoryDB 资源类型及其列表 ARNs，请参阅《服务授权参考》中的 MemoryDB 定义的资源。要了解您可以在哪些操作中指定每个资源的 ARN，请参阅 MemoryDB 定义的操作。

要查看 MemoryDB 基于身份的策略的示例，请参阅 适用于 MemoryDB 的基于身份的策略示例。

MemoryDB 的策略条件键

支持特定于服务的策略条件键：是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。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。

`Condition` 元素根据定义的条件指定语句何时执行。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。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

要查看 MemoryDB 基于身份的策略的示例，请参阅 适用于 MemoryDB 的基于身份的策略示例。

使用条件键

您可以指定决定 IAM 策略如何生效的条件。在 MemoryDB 中，您可以使用 JSON 策略的 `Condition` 元素将请求上下文中的键与您在策略中指定的键值进行比较。有关更多信息，请参阅 IAM JSON 策略元素：条件。

有关 MemoryDB 条件键的列表，请参阅《服务授权参考》中的 MemoryDB 的条件键。

有关全局条件键的列表，请参阅 AWS 全局条件上下文键。

指定条件：使用条件键

要实现精细控制，可以编写 IAM 权限策略，用于指定控制某些请求上的单独参数集的条件。然后，可以将该策略应用于您使用 IAM 控制台创建的 IAM 用户、组或角色。

要应用条件，请将条件信息添加到 IAM 策略语句。例如，要禁止创建任何禁用了 TLS 的 MemoryDB 集群，可以在策略声明中指定以下条件。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "memorydb:CreateCluster"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "Bool": {
          "memorydb:TLSEnabled": "false"
        }
      }
    }
  ]
}
```

有关标记的更多信息，请参阅标记 MemoryDB 资源。

有关使用策略条件运算符的更多信息，请参阅 MemoryDB API 权限：操作、资源和条件参考。

策略示例：使用条件实现精细参数控制

此部分介绍对之前列出的 MemoryDB 参数实现精细访问控制的示例策略。

1. memorydb: TLSEnabled — 指定仅在启用 TLS 的情况下创建集群。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
              {
            "Effect": "Allow",
            "Action": [
                "memorydb:CreateCluster"
            ],
            "Resource": [
                "arn:aws:memorydb:*:*:parametergroup/*",
                "arn:aws:memorydb:*:*:subnetgroup/*",
                "arn:aws:memorydb:*:*:acl/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "memorydb:CreateCluster"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "Bool": {
                    "memorydb:TLSEnabled": "true"
                }
            }
        }
    ]
}
```

2. memorydbUserAuthenticationMode:: — 指定可以使用特定类型的身份验证模式（例如 IAM）创建用户。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "memorydb:Createuser"
            ],
            "Resource": [
                "arn:aws:memorydb:*:*:user/*"
            ],
            "Condition": {
                "StringEquals": {
                    "memorydb:UserAuthenticationMode": "iam"
                }
            }
        }
    ]
}
```

如果您要设置基于 “拒绝” 的策略，则无论情况如何，都建议使用StringEqualsIgnoreCase运算符来避免使用特定用户身份验证模式类型的所有呼叫。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "memorydb:CreateUser"
      ],
      "Resource": "*",
      "Condition": {
        "StringEqualsIgnoreCase": {
          "memorydb:UserAuthenticationMode": "password"
        }
      }
    }
  ]
}
```

MemoryDB 中的访问控制列表 (ACLs)

支持 ACLs：是

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

使用 MemoryDB 的基于属性的访问权限控制（ABAC）

支持 ABAC（策略中的标签）：是

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于称为标签的属性来定义权限。您可以将标签附加到 IAM 实体和 AWS 资源，然后设计 ABAC 策略以允许在委托人的标签与资源上的标签匹配时进行操作。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`aws:RequestTag/*key-name* 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的使用 ABAC 授权定义权限。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

将临时凭证用于 MemoryDB

支持临时凭证：是

临时证书提供对 AWS 资源的短期访问权限，并且是在您使用联合身份或切换角色时自动创建的。AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的临时安全凭证和使用 IAM 的。AWS 服务

MemoryDB 的跨服务主体权限

支持转发访问会话（FAS）：是

转发访问会话 (FAS) 使用调用主体的权限 AWS 服务，再加上 AWS 服务 向下游服务发出请求的请求。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

MemoryDB 的服务角色

支持服务角色：是

服务角色是由一项服务担任、代表您执行操作的 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 AWS 服务委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会破坏 MemoryDB 的功能。仅当 MemoryDB 提供相关指导时才编辑服务角色。

MemoryDB 的服务相关角色

支持服务关联角色：是

服务相关角色是一种与服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务关联角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅能够与 IAM 搭配使用的AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

适用于 MemoryDB 的基于身份的策略示例

默认情况下，用户和角色没有创建或修改 MemoryDB 资源的权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略（控制台）。

有关 MemoryDB 定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅《服务授权参考》中的 MemoryDB 的操作、资源和条件键。 ARNs

主题

- 策略最佳实践
- 使用 MemoryDB 控制台
- 允许用户查看他们自己的权限

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 MemoryDB 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 AWS 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的AWS 托管策略。它们在你的版本中可用 AWS 账户。我们建议您通过定义针对您的用例的 AWS 客户托管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AWS 托管策略或工作职能的AWS 托管策略。
- 应用最低权限：在使用 IAM 策略设置权限时，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 策略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。如果服务操作是通过特定 AWS 服务的（例如）使用的，则也可以使用条件来授予对服务操作的访问权限 CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，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 策略语言（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》中的使用 IAM Access Analyzer 验证策略。
- 需要多重身份验证 (MFA)-如果 AWS 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。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》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 MemoryDB 控制台

要访问 MemoryDB 控制台，您必须拥有一组最低的权限。这些权限必须允许您列出和查看有关您的 MemoryDB 资源的详细信息。 AWS 账户如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 AWS CLI 或 AWS API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍然可以使用 MemoryDB 控制台，还需要将 MemoryDB `ConsoleAccess` 或`ReadOnly` AWS 托管策略附加到实体。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

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或使用 AWS CLI 或 AWS API 以编程方式完成此操作的权限。

```
{
    "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": "*"
        }
    ]
}
```

排查 MemoryDB 的身份和访问权限

使用以下信息可帮助您诊断和修复在使用 MemoryDB 和 IAM 时可能遇到的常见问题。

主题

- 我无权在 MemoryDB 中执行操作
- 我无权执行 iam：PassRole

- 我想允许 AWS 账户之外的人访问我的 MemoryDB 资源

我无权在 MemoryDB 中执行操作

如果 AWS 管理控制台 告诉您您无权执行某项操作，则必须联系管理员寻求帮助。管理员是指提供用户名和密码的人员。

当 mateojackson 用户尝试使用控制台查看有关虚构 *my-example-widget* 资源的详细信息，但不拥有虚构 MemoryDB:*GetWidget* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 MemoryDB:GetWidget on resource: my-example-widget
```

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 MemoryDB:*GetWidget* 操作访问 *my-example-widget* 资源。

我无权执行 iam：PassRole

如果您收到一个错误，表明您无权执行 iam:PassRole 操作，则必须更新策略以允许您将角色传递给 MemoryDB。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的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 marymajor 的 IAM 用户尝试使用控制台在 MemoryDB 中执行操作时，会发生以下示例错误。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 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许 AWS 账户之外的人访问我的 MemoryDB 资源

您可以创建一个角色，以便其他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 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解 MemoryDB 是否支持这些特征，请参阅 MemoryDB 如何与 IAM 协同工作。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）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 中的跨账户资源访问。

访问控制

您可以使用有效的凭证来对自己的请求进行身份验证，但您还必须拥有权限才能创建或访问 MemoryDB 资源。例如，您必须拥有创建 MemoryDB 集群的权限。

下面几节介绍如何管理 MemoryDB 的权限。我们建议您先阅读概述。

- 管理 MemoryDB 资源的访问权限的概述
- 为 MemoryDB 使用基于身份的策略（IAM 策略）

管理 MemoryDB 资源的访问权限的概述

每个 AWS 资源都归一个 AWS 账户所有，创建或访问资源的权限受权限策略的约束。账户管理员可以向 IAM 身份（即：用户、组和角色）附加权限策略。此外，MemoryDB 还支持向资源附加权限策略。

> **Note**
>
> 账户管理员（或管理员用户）是具有管理员权限的用户。有关更多信息，请参阅《IAM 用户指南》中的 IAM 最佳实操。

要提供访问权限，请为您的用户、组或角色添加权限：

- 中的用户和群组 AWS IAM Identity Center：

  创建权限集合。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。
- 通过身份提供者在 IAM 中托管的用户：

  创建适用于身份联合验证的角色。按照《IAM 用户指南》中针对第三方身份提供者创建角色（联合身份验证）的说明进行操作。
- IAM 用户：
  - 创建您的用户可以担任的角色。按照《IAM 用户指南》中为 IAM 用户创建角色的说明进行操作。
  - （不推荐使用）将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中向用户添加权限（控制台）中的说明进行操作。

主题

- MemoryDB 资源和操作
- 了解资源所有权
- 管理对 资源的访问
- 为 MemoryDB 使用基于身份的策略（IAM 策略）
- 资源级权限
- 使用 MemoryDB 的服务相关角色
- AWS MemoryDB 的托管策略
- MemoryDB API 权限：操作、资源和条件参考

MemoryDB 资源和操作

在 MemoryDB 中，主要资源是集群。

这些资源具有与之关联的唯一 Amazon 资源名称 (ARNs)，如下所示。

> **Note**
>
> 为了使资源级权限生效，ARN 字符串上的资源名称应该为小写。

资源类型	ARN 格式
用户	arn: aws: memorydb:: user/user1 *us-east-1 :123456789012*
访问控制列表（ACL）	arn: aws: memorydb:: acl/myacl *us-east-1 :123456789012*
Cluster	arn: aws: memorydb:: cluster/my-cluster *us-east-1:123456789012*
快照	arn: aws: memorydb:: snapshot/my-snashot *us-east-1:123456789012*
参数组	arn: aws: memorydb:: parametergroup/ *us-east-1:123456789012* my-parameter-group
子网组	arn: aws: memorydb:: subnetgroup/ *us-east-1:123456789012* my-subnet-group

MemoryDB 提供一组操作用来处理 MemoryDB 资源。有关可用操作的列表，请参阅 MemoryDB 操作。

了解资源所有权

资源所有者是创建资源的 AWS 账户。也就是说，资源所有者是对创建资源的请求进行身份验证的委托人实体的 AWS 账户。委托人实体可以是根账户、IAM 用户或 IAM 角色。以下示例说明了它的工作原理：

- 假设您使用账户的根 AWS 账户凭证创建集群。在这种情况下，您的 AWS 账户就是资源的所有者。在 MemoryDB 中，该资源为集群。
- 假设您在 AWS 账户中创建了一个 IAM 用户，并向该用户授予创建集群的权限。在这种情况下，用户可以创建集群。但是，该用户所属的您的 AWS 账户拥有群集资源。
- 假设您在 AWS 账户中创建了一个拥有创建集群权限的 IAM 角色。在这种情况下，任何可以代入该角色的人都可以创建集群。该角色所属的您的 AWS 账户拥有群集资源。

管理对 资源的访问

权限策略规定谁可以访问哪些内容。下一节介绍创建权限策略时的可用选项。

> **Note**
>
> 本节讨论如何在 MemoryDB 范围内使用 IAM。这里不提供有关 IAM 服务的详细信息。有关完整的 IAM 文档，请参阅《IAM 用户指南》中的什么是 IAM？。有关 IAM 策略语法和说明的信息，请参阅《IAM 用户指南》中的 AWS IAM 策略参考。

附加到 IAM 身份的策略称为基于身份的策略（IAM 策略）。附加到资源的策略称为基于资源的策略。

主题

- 基于身份的策略（IAM 策略）
- 指定策略元素：操作、效果、资源和主体
- 在策略中指定条件

基于身份的策略（IAM 策略）

您可以向 IAM 身份附加策略。例如，您可以执行以下操作：

- 向您账户中的用户或组附加权限策略 – 账户管理员可以使用与特定用户关联的权限策略来授予权限。在这种情况下，权限可供该用户创建 MemoryDB 资源，例如集群、参数组或安全组。

- 向角色附加权限策略（授予跨账户权限）：您可以向 IAM 角色附加基于身份的权限策略，以授予跨账户的权限。例如，账户 A 中的管理员可以创建一个角色来向另一个 AWS 账户（例如账户 B）或 AWS 服务授予跨账户权限，如下所示：

  1. 账户 A 管理员可以创建一个 IAM 角色，然后向该角色附加授予其访问账户 A 中资源的权限策略。
  2. 账户 A 管理员可以把信任策略附加至用来标识账户 B 的角色，账户 B 由此可以作为主体代入该角色。
  3. 然后，账户 B 管理员可以向账户 B 中的任何用户委派担任该角色的权限。这样，账户 B 中的用户就可以创建或访问账户 A 中的资源。在某些情况下，您可能需要向 AWS 服务授予代入该角色的权限。为支持此方法，信任策略中的委托人也可以是 AWS 服务委托人。

  有关使用 IAM 委托权限的更多信息，请参阅《IAM 用户指南》中的访问权限管理。

以下是允许用户对您的 AWS 账户执行`DescribeClusters`操作的策略示例。MemoryDB 还支持使用用 ARNs 于 API 操作的资源来识别特定资源。（此方法也称为资源级权限。）

有关对 MemoryDB 使用基于身份的策略的更多信息，请参阅 为 MemoryDB 使用基于身份的策略（IAM 策略）。有关用户、组、角色和权限的更多信息，请参阅 IAM 用户指南中的身份（用户、组和角色）。

指定策略元素：操作、效果、资源和主体

对于每个 MemoryDB 资源（请参阅 MemoryDB 资源和操作），该服务都定义了一组 API 操作（请参阅操作）。为授予这些 API 操作的权限，MemoryDB 定义了一组您可以在策略中指定的操作。例如，对于 MemoryDB 集群资源，定义了以下操作：`CreateCluster`、`DeleteCluster` 和 `DescribeClusters`。执行一个 API 操作可能需要多个操作的权限。

以下是最基本的策略元素：

- 资源：在策略中，您可以使用 Amazon Resource Name（ARN）标识策略应用到的资源。有关更多信息，请参阅 MemoryDB 资源和操作。
- 操作 – 您可以使用操作关键字标识要允许或拒绝的资源操作。例如，根据指定的 `Effect`，`memorydb:CreateCluster` 权限允许或拒绝执行 MemoryDB `CreateCluster` 操作的用户权限。
- 效果：您可以指定当用户请求特定操作（可以是允许或拒绝）时的效果。如果没有显式授予（允许）对资源的访问权限，则隐式拒绝访问。您也可显式拒绝对资源的访问。例如，您可以执行此操作，以确保用户无法访问资源，即使有其他策略授予了访问权限也是如此。

- 主体：在基于身份的策略（IAM 策略）中，附加了策略的用户是隐式主体。对于基于资源的策略，您可以指定要接收权限的用户、账户、服务或其他实体（仅适用于基于资源的策略）。

有关 IAM 策略语法和描述的更多信息，请参阅《IAM 用户指南》中的 AWS IAM 策略参考。

有关显示所有 MemoryDB API 操作的表，请参阅 MemoryDB API 权限：操作、资源和条件参考。

在策略中指定条件

当您授予权限时，可使用 IAM 策略语言来指定规定策略何时生效的条件。例如，您可能希望策略仅在特定日期后应用。有关使用策略语言指定条件的更多信息，请参阅《IAM 用户指南》中的条件。

为 MemoryDB 使用基于身份的策略（IAM 策略）

本主题提供了基于身份的策略的示例，在这些策略中，账户管理员可以向 IAM 身份（即：用户、组和角色）附加权限策略。

> **Important**
>
> 我们建议您先阅读说明了以下方面的基本概念和选项的主题：管理对 MemoryDB 资源的访问。有关更多信息，请参阅 管理 MemoryDB 资源的访问权限的概述。

本主题的各个部分涵盖以下内容：

- 使用 MemoryDB 控制台所需的权限
- 适用于 MemoryDB 的AWS托管（预定义）策略
- 客户管理型策略示例

下面介绍权限策略示例。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [{
       "Sid": "AllowClusterPermissions",
       "Effect": "Allow",
       "Action": [
          "memorydb:CreateCluster",
          "memorydb:DescribeClusters",
          "memorydb:UpdateCluster"],
       "Resource": "*"
       },
       {
         "Sid": "AllowUserToPassRole",
         "Effect": "Allow",
         "Action": [ "iam:PassRole" ],
         "Resource": "arn:aws:iam::123456789012:role/EC2-roles-for-cluster"
       }
   ]
```

```
}
```

该策略包含两条语句：

- 第一条语句授予对账户所拥有的任何集群的 MemoryDB 操作（`memorydb:CreateCluster`、`memorydb:DescribeClusters` 和 `memorydb:UpdateCluster`）权限。
- 第二条语句授予对 `Resource` 值末尾指定的 IAM 角色名称的 IAM 操作（`iam:PassRole`）的权限。

该策略不指定 `Principal` 元素，因为在基于身份的策略中，您未指定获取权限的委托人。附加了策略的用户是隐式委托人。向 IAM 角色附加权限策略后，该角色的信任策略中标识的主体将获取权限。

有关显示所有 MemoryDB API 操作及其适用的资源的表，请参阅 MemoryDB API 权限：操作、资源和条件参考。

使用 MemoryDB 控制台所需的权限

权限参考表列出 MemoryDB API 操作并显示每个操作所需的权限。有关 MemoryDB API 操作的更多信息，请参阅 MemoryDB API 权限：操作、资源和条件参考。

要使用 MemoryDB 控制台，请首先授予执行其他操作的权限，如以下权限策略中所示。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "MinPermsForMemDBConsole",
        "Effect": "Allow",
        "Action": [
            "memorydb:Describe*",
            "memorydb:List*",
            "ec2:DescribeAvailabilityZones",
            "ec2:DescribeVpcs",
            "ec2:DescribeAccountAttributes",
            "ec2:DescribeSecurityGroups",
            "cloudwatch:GetMetricStatistics",
            "cloudwatch:DescribeAlarms",
            "s3:ListAllMyBuckets",
```

```
            "sns:ListTopics",
            "sns:ListSubscriptions" ],
        "Resource": "*"
        }
    ]
}
```

MemoryDB 控制台出于以下原因需要上述其他权限：

- MemoryDB 操作权限使控制台可以显示账户中的 MemoryDB 资源。
- 控制台需要ec2操作权限才能查询 Amazon EC2，这样它才能显示可用区 VPCs、安全组和账户属性。
- cloudwatch操作权限使控制台能够检索 Amazon CloudWatch 指标和警报，并将其显示在控制台中。
- sns 操作权限使控制台可以检索 Amazon Simple Notification Service（Amazon SNS）主题和订阅，并将其显示在控制台中。

客户管理型策略示例

如果您未使用默认策略并选择使用自定义托管策略，请确保以下两项之一。您应该有权调用 iam:createServiceLinkedRole（有关更多信息，请参阅示例 4：允许用户调用 IAM CreateServiceLinkedRole API）。或者您应该已经创建了 MemoryDB 服务相关角色。

本节中的示例策略与使用 MemoryDB 控制台所需的最低权限相结合时，将授予其他权限。这些示例也与 AWS SDKs 和有关 AWS CLI。有关使用 MemoryDB 控制台所需的权限的更多信息，请参阅 使用 MemoryDB 控制台所需的权限。

有关设置 IAM 用户和组的说明，请参阅 IAM 用户指南中的创建您的第一个 IAM 用户和管理员组。

> **Important**
>
> 在生产中使用 IAM 策略之前，请始终全面测试这些策略。当您使用 MemoryDB 控制台时，一些看起来简单的 MemoryDB 操作可能需要其他操作来支持它们。例如，memorydb:CreateCluster 授予创建 MemoryDB 集群的权限。但是，为执行此操作，MemoryDB 控制台使用一些 Describe 和 List 操作来填充控制台列表。

示例

- 示例 1：允许用户对 MemoryDB 资源进行只读访问
- 示例 2：允许用户执行常见的 MemoryDB 系统管理员任务
- 示例 3：允许用户访问所有 MemoryDB API 操作
- 示例 4：允许用户调用 IAM CreateServiceLinkedRole API

示例 1：允许用户对 MemoryDB 资源进行只读访问

以下策略授予允许用户列出资源 MemoryDB 操作权限。通常，您将此类型的权限策略挂载到管理人员组。

JSON

```
{
   "Version":"2012-10-17",
   "Statement":[{
      "Sid": "MemDBUnrestricted",
      "Effect":"Allow",
      "Action": [
          "memorydb:Describe*",
          "memorydb:List*"],
      "Resource":"*"
      }
   ]
}
```

示例 2：允许用户执行常见的 MemoryDB 系统管理员任务

常见的系统管理员任务包括：修改集群、参数和参数组。系统管理员还可能需要获得有关 MemoryDB 事件的信息。以下策略授予执行这些常见系统管理员任务的 MemoryDB 操作的用户权限。通常，您将此类型的权限策略挂载到系统管理员组。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "MDBAllowSpecific",
```

```
            "Effect": "Allow",
            "Action": [
                "memorydb:UpdateCluster",
                "memorydb:DescribeClusters",
                "memorydb:DescribeEvents",
                "memorydb:UpdateParameterGroup",
                "memorydb:DescribeParameterGroups",
                "memorydb:DescribeParameters",
                "memorydb:ResetParameterGroup"
            ],
            "Resource": "*"
        }
    ]
}
```

示例 3：允许用户访问所有 MemoryDB API 操作

以下策略允许用户访问所有 MemoryDB 操作。建议您仅向管理员用户授予此类型的权限策略。

JSON

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

示例 4：允许用户调用 IAM CreateServiceLinkedRole API

以下策略允许用户调用 IAM `CreateServiceLinkedRole` API。我们建议您对调用变化 MemoryDB 操作的用户应用此类型的权限策略。

JSON

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"CreateSLRAllows",
      "Effect":"Allow",
      "Action":[
        "iam:CreateServiceLinkedRole"
      ],
      "Resource":"*",
      "Condition":{
        "StringLike":{
          "iam:AWS ServiceName":"memorydb.amazonaws.com"
        }
      }
    }
  ]
}
```

资源级权限

您可以通过在 IAM 策略中指定资源来限制权限范围。许多 AWS CLI API 操作支持的资源类型因操作的行为而异。每条 IAM 策略语句为对一个资源执行的一个操作授予权限。如果操作不对指定资源执行操作，或者您授予对所有资源执行操作的权限，则策略中资源的值为通配符（*）。对于许多 API 操作，可以通过指定资源的 Amazon 资源名称（ARN）或与多个资源匹配的 ARN 模式来限制用户可修改的资源。要按资源限制权限，请指定资源的 ARN。

MemoryDB 资源 ARN 格式

> **Note**
>
> 为了使资源级权限生效，ARN 字符串中的资源名称应为小写。

- 用户 — arn: aws: memorydb:: user/user1 *us-east-1:123456789012*
- ACL — arn: aws: memorydb:: acl/my-acl *us-east-1:123456789012*
- 集群 — arn: aws: memorydb:: cluster/my-cluster *us-east-1:123456789012*

- 快照 — arn: aws: memorydb:: snapshot/my-snapshot *us-east-1:123456789012*
- 参数组 — arn: aws: memorydb:: parametergroup/ *us-east-1:123456789012* my-parameter-group
- 子网组 — arn: aws: memorydb:: subnetgroup/ *us-east-1:123456789012* my-subnet-group

示例

- 示例 1：允许用户完全访问特定 MemoryDB 资源类型
- 示例 2：拒绝用户访问集群。

示例 1：允许用户完全访问特定 MemoryDB 资源类型

以下策略明确允许指定的 account-id 完全访问子网组、安全组和集群类型的所有资源。

```
{
        "Sid": "Example1",
        "Effect": "Allow",
        "Action": "memorydb:*",
        "Resource": [
             "arn:aws:memorydb:us-east-1:account-id:subnetgroup/*",
             "arn:aws:memorydb:us-east-1:account-id:securitygroup/*",
             "arn:aws:memorydb:us-east-1:account-id:cluster/*"
        ]
}
```

示例 2：拒绝用户访问集群。

以下示例明确拒绝对特定集群进行指定 account-id 访问。

```
{
        "Sid": "Example2",
        "Effect": "Deny",
        "Action": "memorydb:*",
        "Resource": [
                "arn:aws:memorydb:us-east-1:account-id:cluster/name"
        ]
}
```

使用 MemoryDB 的服务相关角色

MemoryDB 使用 AWS Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种独特的 IAM 角色，直接链接到 AWS 服务，例如 MemoryDB。MemoryDB 服务相关角色由 MemoryDB 预定义。它们包含该服务代表您的集群调用 AWS 服务所需的一切权限。

使用服务相关角色时，您不必手动添加必要的权限，所以您可以更轻松地设置 MemoryDB。这些角色已存在于您的 AWS 账户中，但已关联到 MemoryDB 用例并具有预定义的权限。只有 MemoryDB 可以代入这些角色，并且只有这些角色可以使用预定义的权限策略。只有先删除角色的相关资源，才能删除角色。这将保护您的 MemoryDB 资源，因为您不会无意中删除访问资源的必要权限。

有关支持服务相关角色的其他服务的信息，请参阅使用 IAM 的AWS 服务并查找服务相关角色列中显示为是的服务。选择是和链接，查看该服务的服务关联角色文档。

目录

- MemoryDB 的服务相关角色权限
- 创建服务相关角色（IAM）
  - 创建服务相关角色（IAM 控制台）
  - 创建服务相关角色（IAM CLI）
  - 创建服务相关角色（IAM API）
- 编辑 MemoryDB 的服务相关角色的描述
  - 编辑服务相关角色描述（IAM 控制台）
  - 编辑服务相关角色描述（IAM CLI）
  - 编辑服务相关角色描述（IAM API）
- 删除 MemoryDB 的服务相关角色
  - 清除服务相关角色
  - 删除服务相关角色（IAM 控制台）
  - 删除服务相关角色（IAM CLI）
  - 删除服务相关角色（IAM API）

MemoryDB 的服务相关角色权限

MemoryDB 使用名为 AWSServiceRoleForMemoryDB 的服务相关角色 — 此策略允许 MemoryDB 在必要时代表您管理管理集群的 AWS 资源。

AWSServiceRoleForMemory数据库服务相关角色权限策略允许 MemoryDB 对指定资源完成以下操作：

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateTags"
   ],
   "Resource": "arn:aws-cn:ec2:*:*:network-interface/*",
   "Condition": {
    "StringEquals": {
     "ec2:CreateAction": "CreateNetworkInterface"
    },
    "ForAllValues:StringEquals": {
     "aws:TagKeys": [
      "AmazonMemoryDBManaged"
     ]
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateNetworkInterface"
   ],
   "Resource": [
    "arn:aws-cn:ec2:*:*:network-interface/*",
    "arn:aws-cn:ec2:*:*:subnet/*",
    "arn:aws-cn:ec2:*:*:security-group/*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DeleteNetworkInterface",
    "ec2:ModifyNetworkInterfaceAttribute"
   ],
   "Resource": "arn:aws-cn:ec2:*:*:network-interface/*",
```

```
    "Condition": {
     "StringEquals": {
      "ec2:ResourceTag/AmazonMemoryDBManaged": "true"
     }
    }
   },
   {
    "Effect": "Allow",
    "Action": [
     "ec2:DeleteNetworkInterface",
     "ec2:ModifyNetworkInterfaceAttribute"
    ],
    "Resource": "arn:aws-cn:ec2:*:*:security-group/*"
   },
   {
    "Effect": "Allow",
    "Action": [
     "ec2:DescribeSecurityGroups",
     "ec2:DescribeNetworkInterfaces",
     "ec2:DescribeAvailabilityZones",
     "ec2:DescribeSubnets",
     "ec2:DescribeVpcs"
    ],
    "Resource": "*"
   },
   {
    "Effect": "Allow",
    "Action": [
     "cloudwatch:PutMetricData"
    ],
    "Resource": "*",
    "Condition": {
     "StringEquals": {
      "cloudwatch:namespace": "AWS/MemoryDB"
     }
    }
   }
  ]
}
```

有关更多信息，请参阅 AWS 托管策略：内存 DBService RolePolicy。

允许 IAM 实体创建 AWSServiceRoleForMemory数据库服务相关角色

向该 IAM 实体的权限中添加以下策略声明：

```
{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:PutRolePolicy"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB*",
    "Condition": {"StringLike": {"iam:AWS ServiceName": "memorydb.amazonaws.com"}}
}
```

允许 IAM 实体删除 AWSServiceRoleForMemory数据库服务相关角色

向该 IAM 实体的权限中添加以下策略声明：

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB*",
    "Condition": {"StringLike": {"iam:AWS ServiceName": "memorydb.amazonaws.com"}}
}
```

或者，您可以使用 AWS 托管策略来提供对 MemoryDB 的完全访问权限。

创建服务相关角色（IAM）

您可以使用 IAM 控制台、CLI 或 API 创建服务相关角色。

创建服务相关角色（IAM 控制台）

您可使用 IAM 控制台创建服务相关角色。

创建服务相关角色（控制台）

1. 登录 AWS 管理控制台 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的左侧导航窗格中，选择 Roles。然后选择创建新角色。

3. 在 Select type of trusted entity（选择受信任实体的类型）下，选择 AWS Service（亚马逊云科技服务）。
4. 在或选择一个服务以查看其用例中，选择 MemoryDB。
5. 选择下一步: 权限。
6. 在 策略名称下，请注意此角色需要 `MemoryDBServiceRolePolicy`。选择 Next:Tags（下一步: 标签）。
7. 请注意，服务相关角色不支持标签。选择下一步: 审核。
8. （可选）对于 Role description，编辑新服务相关角色的描述。
9. 检查角色，然后选择创建角色。

创建服务相关角色（IAM CLI）

您可以使用中的 IAM 操作 AWS Command Line Interface 来创建服务相关角色。此角色可以包括服务代入角色时所需的信任策略和内联策略。

创建服务相关角色（CLI）

使用以下操作：

```
$ aws iam create-service-linked-role --aws-service-name memorydb.amazonaws.com
```

创建服务相关角色（IAM API）

您可以使用 IAM API 创建服务相关角色。此角色可以包括服务代入角色时所需的信任策略和内联策略。

创建服务相关角色（API）

使用 CreateServiceLinkedRole API 调用。在请求中，指定 `memorydb.amazonaws.com` 的服务名称。

编辑 MemoryDB 的服务相关角色的描述

MemoryDB 不允许您编辑 AWSServiceRoleForMemory数据库服务相关角色。创建服务关联角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。

编辑服务相关角色描述（IAM 控制台）

您可以使用 IAM 控制台编辑服务相关角色的描述。

编辑服务相关角色的描述（控制台）

1. 在 IAM 控制台的左侧导航窗格中，选择 Roles。
2. 以下代码示例显示如何将 IAM 策略附加到用户。
3. 在 Role description 的最右侧，选择 Edit。
4. 在框中输入新描述，然后选择 Save（保存）。

编辑服务相关角色描述（IAM CLI）

您可以使用中的 IAM 操作 AWS Command Line Interface 来编辑与服务相关的角色描述。

更改服务相关角色的描述（CLI）

1. （可选）要查看角色的当前描述，请使用 for AWS CLI IAM 操作get-role。

   Example

   ```
   $ aws iam get-role --role-name AWSServiceRoleForMemoryDB
   ```

   通过 CLI 操作使用角色名称（并非 ARN）指向角色。例如，如果一个角色的 ARN 为 `arn:aws:iam::123456789012:role/myrole`，则应将角色称为 **myrole**。
2. 要更新服务相关角色的描述，请使用 for I AWS CLI AM 操作update-role-description。

   对于 Linux、macOS 或 Unix：

   ```
   $ aws iam update-role-description \
       --role-name AWSServiceRoleForMemoryDB \
       --description "new description"
   ```

   对于 Windows：

   ```
   $ aws iam update-role-description ^
       --role-name AWSServiceRoleForMemoryDB ^
       --description "new description"
   ```

编辑服务相关角色描述（IAM API）

您可以使用 IAM API 编辑服务相关角色描述。

更改服务相关角色的描述（API）

1. （可选）要查看角色的当前描述，请使用 IAM API 操作 GetRole。

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=GetRole
      &RoleName=AWSServiceRoleForMemoryDB
      &Version=2010-05-08
      &AUTHPARAMS
   ```

2. 要更新角色的描述，请使用 IAM API 操作 UpdateRoleDescription。

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=UpdateRoleDescription
      &RoleName=AWSServiceRoleForMemoryDB
      &Version=2010-05-08
      &Description="New description"
   ```

删除 MemoryDB 的服务相关角色

如果不再需要使用某个需要服务关联角色的功能或服务，我们建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，您必须先清除您的服务相关角色，然后才能将其删除。

MemoryDB 不会删除您的服务相关角色。

清除服务相关角色

您必须先确认该角色没有与之关联的资源（集群），然后才能使用 IAM 删除服务相关角色。

在 IAM 控制台中检查服务相关角色是否具有活动会话

1. 登录 AWS 管理控制台 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的左侧导航窗格中，选择 Roles。然后选择 AWSServiceRoleForMemory数据库角色的名称（不是复选框）。
3. 在所选角色的 Summary 页面上，选择 Access Advisor 选项卡。
4. 在访问顾问选项卡查看服务相关角色的近期活动。

删除需要数据库的 MemoryDB 资源（ AWSServiceRoleForMemory控制台 ）

- 要删除集群，请参阅以下内容：
  - 使用 AWS 管理控制台
  - 使用 AWS CLI
  - 使用 MemoryDB API

删除服务相关角色（IAM 控制台）

您可以使用 IAM 控制台删除服务相关角色。

删除服务相关角色 (控制台)

1. 登录 AWS 管理控制台 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的左侧导航窗格中，选择 Roles。然后，选中要删除的角色名称旁边的复选框，而不是名称或行本身。
3. 对于页面顶部的角色操作，请选择删除角色。
4. 在确认页面中，查看上次访问服务的数据，该数据显示了每个选定角色上次访问 AWS 服务的时间。这样可帮助您确认角色当前是否处于活动状态。如果要继续，请选择 Yes, Delete 以提交服务相关角色进行删除。
5. 监视 IAM 控制台通知，以监控服务相关角色的删除进度。由于 IAM 服务相关角色删除是异步的，因此，在您提交角色进行删除后，删除任务可能成功，也可能失败。如果任务失败，您可以从通知中选择 View details 或 View Resources 以了解删除失败的原因。

删除服务相关角色（IAM CLI）

您可以使用中的 IAM 操作 AWS Command Line Interface 来删除服务相关角色。

删除服务相关角色（CLI）

1. 如果您不知道要删除的服务相关角色的名称，请输入以下命令。此命令列出了您账户中的角色及其 Amazon 资源名称 (ARNs)。

   ```
   $ aws iam get-role --role-name role-name
   ```

   通过 CLI 操作使用角色名称（并非 ARN）指向角色。例如，如果某个角色具有 ARN `arn:aws:iam::123456789012:role/myrole`，则将该角色称为 **`myrole`**。

2. 如果服务相关角色正被使用或具有关联的资源，则无法删除它，因此您必须提交删除请求。如果不满足这些条件，该请求可能会被拒绝。您必须从响应中捕获 deletion-task-id 以检查删除任务的状态。输入以下命令以提交服务相关角色的删除请求。

```
$ aws iam delete-service-linked-role --role-name role-name
```

3. 输入以下命令以检查删除任务的状态。

```
$ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
```

删除任务的状态可能是 NOT_STARTED、IN_PROGRESS、SUCCEEDED 或 FAILED。如果删除失败，则调用会返回失败的原因，以便您进行问题排查。

删除服务相关角色（IAM API）

您可以使用 IAM API 删除服务相关角色。

删除服务相关角色（API）

1. 要提交服务相关角色的删除请求，请调用 DeleteServiceLinkedRole。在请求中，指定角色名称。

   如果服务相关角色正被使用或具有关联的资源，则无法删除它，因此您必须提交删除请求。如果不满足这些条件，该请求可能会被拒绝。您必须从响应中捕获 DeletionTaskId 以检查删除任务的状态。

2. 要检查删除的状态，请调用 GetServiceLinkedRoleDeletionStatus。在请求中，指定 DeletionTaskId。

   删除任务的状态可能是 NOT_STARTED、IN_PROGRESS、SUCCEEDED 或 FAILED。如果删除失败，则调用会返回失败的原因，以便您进行问题排查。

AWS MemoryDB 的托管策略

要向用户、群组和角色添加权限，使用 AWS 托管策略比自己编写策略要容易得多。创建仅为团队提供所需权限的 IAM 客户管理型策略需要时间和专业知识。要快速入门，您可以使用我们的 AWS 托管策略。这些政策涵盖常见用例，可在您的 AWS 账户中使用。有关 AWS 托管策略的更多信息，请参阅 IAM 用户指南中的AWS 托管策略。

AWS 服务维护和更新 AWS 托管策略。您无法更改 AWS 托管策略中的权限。服务偶尔会向 AWS 托管式策略添加额外权限以支持新特征。此类更新会影响附加策略的所有身份（用户、组和角色）。当启动新特征或新操作可用时，服务最有可能会更新 AWS 托管式策略。服务不会从 AWS 托管策略中移除权限，因此策略更新不会破坏您的现有权限。

此外，还 AWS 支持跨多个服务的工作职能的托管策略。例如，ReadOnlyAccess AWS 托管策略提供对所有 AWS 服务和资源的只读访问权限。当服务启动一项新功能时， AWS 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AWS 托管式策略。

AWS 托管策略：内存 DBService RolePolicy

您无法将内存DBServiceRolePolicy AWS 托管策略附加到账户中的身份。此策略是 M AWS emoryDB 服务相关角色的一部分。此角色允许服务管理您账户中的网络接口和安全组。

MemoryDB 使用此策略中的权限来管理 EC2 安全组和网络接口。这是管理 MemoryDB 集群所必需的。

权限详细信息

该策略包含以下权限。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateTags"
   ],
   "Resource": "arn:aws-cn:ec2:*:*:network-interface/*",
   "Condition": {
    "StringEquals": {
     "ec2:CreateAction": "CreateNetworkInterface"
```

```
    },
    "ForAllValues:StringEquals": {
     "aws:TagKeys": [
      "AmazonMemoryDBManaged"
     ]
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:CreateNetworkInterface"
   ],
   "Resource": [
    "arn:aws-cn:ec2:*:*:network-interface/*",
    "arn:aws-cn:ec2:*:*:subnet/*",
    "arn:aws-cn:ec2:*:*:security-group/*"
   ]
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DeleteNetworkInterface",
    "ec2:ModifyNetworkInterfaceAttribute"
   ],
   "Resource": "arn:aws-cn:ec2:*:*:network-interface/*",
   "Condition": {
    "StringEquals": {
     "ec2:ResourceTag/AmazonMemoryDBManaged": "true"
    }
   }
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DeleteNetworkInterface",
    "ec2:ModifyNetworkInterfaceAttribute"
   ],
   "Resource": "arn:aws-cn:ec2:*:*:security-group/*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "ec2:DescribeSecurityGroups",
```

```
    "ec2:DescribeNetworkInterfaces",
    "ec2:DescribeAvailabilityZones",
    "ec2:DescribeSubnets",
    "ec2:DescribeVpcs"
   ],
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": [
    "cloudwatch:PutMetricData"
   ],
   "Resource": "*",
   "Condition": {
    "StringEquals": {
     "cloudwatch:namespace": "AWS/MemoryDB"
    }
   }
  }
 ]
}
```

适用于 MemoryDB 的AWS托管（预定义）策略

AWS 通过提供由创建和管理的独立 IAM 策略来解决许多常见用例 AWS。托管策略可针对常见使用案例授予必要权限，因此，您无需自行调查具体需要哪些权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

以下 AWS 托管策略是特定于 MemoryDB 的，您可以将其附加到账户中的用户：

AmazonMemoryDBReadOnlyAccess

您可以将 `AmazonMemoryDBReadOnlyAccess` 策略附加到 IAM 身份。此策略授予允许只读访问所有 MemoryDB 资源的管理权限。

AmazonMemoryDBReadOnlyAccess-授予对 MemoryDB 资源的只读访问权限。

JSON

```
{
 "Version":"2012-10-17",
```

```
 "Statement": [{
  "Effect": "Allow",
  "Action": [
   "memorydb:Describe*",
   "memorydb:List*"
  ],
  "Resource": "*"
 }]
}
```

AmazonMemoryDBFull访问权限

您可以将 AmazonMemoryDBFullAccess 策略附加到 IAM 身份。此策略授予允许完全访问 MemoryDB 资源的管理权限。

AmazonMemoryDBFull访问权限-授予对 MemoryDB 资源的完全访问权限。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [{
   "Effect": "Allow",
   "Action": "memorydb:*",
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": "iam:CreateServiceLinkedRole",
   "Resource": "arn:aws:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB",
   "Condition": {
    "StringLike": {
     "iam:AWSServiceName": "memorydb.amazonaws.com"
    }
   }
  }
 ]
}
```

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [{
   "Effect": "Allow",
   "Action": "memorydb:*",
   "Resource": "*"
  },
  {
   "Effect": "Allow",
   "Action": "iam:CreateServiceLinkedRole",
   "Resource": "arn:aws-cn:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB",
   "Condition": {
    "StringLike": {
     "iam:AWSServiceName": "memorydb.amazonaws.com"
    }
   }
  }
 ]
}
```

您还可以创建自定义 IAM 策略，以授予执行 MemoryDB API 操作的相关权限。您可以将这些自定义策略附加到需要这些权限的 IAM 用户或组。

MemoryDB 对托管策略的 AWS 更新

查看自该服务开始跟踪这些更改以来，MemoryDB AWS 托管策略更新的详细信息。有关此页面更改的自动提示，请订阅 MemoryDB 文档历史记录页面上的 RSS 源。

更改	描述	日期
AWS 托管策略：内存 DBService RolePolicy – 添加策略	Memory DBService RolePolicy 添加了 memorydb: 的权限。ReplicateMultiRegionClusterData此权限将允许服务关	2024 年 1 月 12 日

更改	描述	日期
	联角色为 MemoryDB 多区域集群复制数据。	
AmazonMemoryDBFull访问权限 – 添加策略	MemoryDB 添加了描述和列出受支持资源的新权限。MemoryDB 需要这些权限，才能查询账户中的所有支持资源。	10/07/2021
AmazonMemoryDBReadOnlyAccess – 添加策略	MemoryDB 添加了描述和列出受支持资源的新权限。MemoryDB 需要这些权限，才能通过查询账户中的所有受支持资源来创建基于账户的应用程序。	10/07/2021
MemoryDB 开始跟踪更改	服务启动	8/19/2021

MemoryDB API 权限：操作、资源和条件参考

在设置访问控制以及编写可附加到 IAM 策略的权限策略（基于身份或基于资源）时，可将下表作为参考。此表列出每个 MemoryDB API 操作及您可授予执行该操作的权限的对应操作。您可以在策略的 `Action` 字段中指定这些操作，并在策略的 `Resource` 字段中指定资源值。除非另有说明，否则需要该资源。某些字段同时包含必需资源和可选资源。如果没有资源 ARN，则策略中的资源为通配符（*）。

> **Note**
>
> 要指定操作，请在 API 操作名称之前使用 `memorydb:` 前缀（例如，`memorydb:DescribeClusters`）。

日志记录和监控

监控是保持 MemoryDB 和您的其他 AWS 解决方案的可靠性、可用性和性能的重要方面。AWS 提供了以下一些监控工具来监控 MemoryDB、在出现错误时进行报告并适时自动采取措施：

- Amazon CloudWatch 实时监控您的 AWS 资源以及在 AWS 上运行的应用程序。您可以收集和跟踪指标，创建自定义的控制平面，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。例如，您可以使用 CloudWatch 跟踪 Amazon EC2 实例的 CPU 使用率或其他指标并且在需要时自动启动新实例。有关更多信息，请参阅《Amazon CloudWatch 用户指南》https://docs.aws.amazon.com/AmazonCloudWatch/latest/monitoring/。
- Amazon CloudWatch Logs 使您能够监控、存储和访问来自 Amazon EC2 实例、CloudTrail 和其他来源的日志文件。CloudWatch Logs 可以监控日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南。
- AWS CloudTrail 捕获由您的 AWS 账户或代表该账户发出的 API 调用和相关事件，并将日志文件传输到您指定的 Amazon S3 存储桶。您可以标识哪些用户和账户调用了 AWS、发出调用的源 IP 地址以及调用的发生时间。有关更多信息，请参阅《AWS CloudTrail 用户指南》。

使用 Amazon CloudWatch 监控 MemoryDB

您可以使用 CloudWatch 监控 MemoryDB，CloudWatch 会收集原始数据并将其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《Amazon CloudWatch 用户指南》。

以下部分列出了 MemoryDB 的指标和维度。

主题

- 主机级指标
- MemoryDB 的指标
- 应监控哪些指标？
- 选择指标统计数据和周期
- 监控 CloudWatch 指标

主机级指标

AWS/MemoryDB 命名空间包含各个节点的以下主机级指标。

另请参阅

- MemoryDB 的指标

指标	描述	单位
CPUUtilization	整个主机的 CPU 使用率百分比。由于 Valkey 和 Redis OSS 是单线程的，所以我们建议您监控有 4 个或更多 vCPU 的节点的 EngineCPU Utilization 指标。	百分比
FreeableMemory	主机上可用的闲置内存量。此数字源自操作系统报告为可释放的 RAM 和缓冲区中的内存。	字节
NetworkBytesIn	主机已从网络读取的字节数。	字节
NetworkBytesOut	实例在所有网络接口上发送的字节数。	字节
NetworkPacketsIn	实例在所有网络接口上收到的数据包的数量。此指标依据单个实例上的数据包数量来标识传入流量的量。	计数

指标	描述	单位
NetworkPacketsOut	实例在所有网络接口上发送的数据包的数量。此指标依据单个实例上的数据包数量标识传出流量的量。	计数
NetworkBandwidthInAllowanceExceeded	因入站聚合带宽超过实例的最大值而形成的数据包的数量。	计数
NetworkConntrackAllowanceExceeded	由于连接跟踪超过实例的最大值且无法建立新连接而形成的数据包的数量。这可能会导致进出实例的流量丢失数据包。	计数
NetworkBandwidthOutAllowanceExceeded	因出站聚合带宽超过实例的最大值而形成的数据包的数量。	计数
NetworkPacketsPerSecondAllowanceExceeded	因双向每秒数据包数量超过实例的最大值而形成的数据包数量。	计数
NetworkMaxBytesIn	每分钟内接收字节的最大每秒突增量。	字节
NetworkMaxBytesOut	每分钟内传输字节的最大每秒突增量。	字节
NetworkMaxPacketsIn	每分钟内接收数据包的最大每秒突增量。	计数
NetworkMaxPacketsOut	每分钟内传输数据包的最大每秒突增量。	计数
SwapUsage	主机上的交换区使用量。	字节

MemoryDB 的指标

AWS/MemoryDB 命名空间包括以下指标。

除 ReplicationLag、EngineCPUUtilization、SuccessfulWriteRequestLatency 和 SuccessfulReadRequestLatency 外，这些指标源自 Valkey 和 Redis OSS 的 info 命令。每项指标都是按照节点级计算的。

有关 INFO 命令的完整文档，请参阅 INFO。

另请参阅。

- 主机级指标

指标	描述	单位
ActiveDefragHits	活动碎片整理进程每分钟执行的值重新分配数。这是根据 INFO 的 active_defrag_hits 统计数据得出的。	数字
AuthenticationFailures	使用 AUTH 命令进行身份验证时的总失败尝试次数。您可以使用 ACL LOG 命令查找有关个人身份验证失败的更多信息。我们建议为此设置告警以检测未经授权的访问尝试。	计数
BytesUsedForMemoryDB	MemoryDB 为所有目的（包括数据集、缓冲区等）分配的字节的总数。	字节
	Dimension: Tier=SSD （对于使用 数据分层 功能的集群）：SSD 所使用的总字节数。	字节
	Dimension: Tier=Memory （对于使用 数据分层 功能的集群）：内存所使用的总字节数。这是 INFO 的 used_memory 统计数据的值。	字节
BytesReadFromDisk	每分钟从磁盘读取的总字节数。仅支持使用数据分层功能的集群。	字节
BytesWrittenToDisk	每分钟写入磁盘的总字节数。仅支持使用数据分层功能的集群。	字节
CommandAuthorizationFailures	用户运行其无权限调用的命令的失败尝试次数。您可以使用 ACL LOG 命令查找有关个人身份验证失败的更多信息。我们建议为此设置告警以检测未经授权的访问尝试。	计数
CurrConnections	客户端连接数，不包括来自只读副本的连接。MemoryDB 使用 2 至 4 个连接来监控各种	计数

指标	描述	单位
	情况下的集群。这是根据 INFO 的 connected_clients 统计数据得出的。	
CurrItems	缓存中的项目数。这是根据 keyspace 统计数据得出的，方法是计算整个键空间中所有键的总和。	计数
	Dimension: Tier=Memory （对于使用数据分层功能的集群）。内存中的项目数。	计数
	Dimension: Tier=SSD （固态硬盘）（对于使用功能的 Redis 集群）数据分层。SSD 中的项目数。	计数
DatabaseMemoryUsagePercentage	正在使用的集群的可用内存的百分比。这是使用 used_memory/maxmemory 根据 INFO 计算得出的。	百分比
DatabaseCapacityUsagePercentage	集群的总数据容量中正在使用的百分比。 在数据分层实例上，指标的计算方式为 (used_memory - mem_not_counted_for_evict + SSD used) / (maxmemory + SSD total capacity) ，其中 used_memory 和 maxmemory 取自 INFO。 在所有其他情况下，使用 used_memory/maxmemory 计算指标。	百分比
DB0AverageTTL	根据 INFO 命令中的 keyspace 统计数据公开 DBO 的 avg_ttl。	毫秒

指标	描述	单位
EngineCPUUtilization	提供 Valkey 或 Redis OSS 引擎线程的 CPU 使用率。由于引擎为单线程，您可以使用该指标来分析该进程本身的负载。EngineCPUUtilization 指标更精确地呈现了该进程。您可以将其与 CPUUtilization 指标配合使用。CPUUtilization 公开服务器实例整体的 CPU 使用率，包括其他操作系统和管理流程。对于有四个或更多 vCPU 的较大节点类型，可使用 EngineCPUUtilization 指标来监控和设置扩展阈值。 **Note** 在 MemoryDB 主机上，后台进程将监控主机以提供托管式数据库体验。这些后台进程可能会占用很大一部分 CPU 工作负载。这在具有两个以上 vCPU 的大型主机上影响不大，但在 vCPU 个数不超过 2 个的小型主机上影响较大。如果仅监控 EngineCPUUtilization 指标，您将无法发现因 Valkey 或 Redis OSS 引擎或后台监控进程的 CPU 使用率过高而导致主机过载情况。因此，我们建议对于具有不超过两个 vCPU 的主机，还需要监控 CPUUtilization 指标。	百分比
Evictions	由于 maxmemory 限制而被驱逐的密钥数。这是根据 INFO 的 evicted_keys 统计数据得出的。	计数
IsPrimary	指示节点是否为当前分片的主节点。指标可以是 0（非主节点）或 1（主节点）。	计数

指标	描述	单位
KeyAuthorizationFailures	用户访问其无权限访问的密钥的失败尝试次数。您可以使用 ACL LOG 命令查找有关个人身份验证失败的更多信息。我们建议为此设置告警以检测未经授权的访问尝试。	计数
KeyspaceHits	主字典中成功的只读键查找次数。这是根据 INFO 的 keyspace_hits 统计数据得出的。	计数
KeyspaceMisses	主字典中失败的只读键查找次数。这是根据 INFO 的 keyspace_misses 统计数据得出的。	计数
KeysTracked	键跟踪所跟踪的键数占 tracking-table-max-keys 的百分比。密钥跟踪用于帮助客户端侧缓存，并在修改密钥时通知客户端。	计数
MaxReplicationThroughput	观测到的最大吞吐量。吞吐量在短时间间隔内采样以识别流量突发。报告采样值中的最大值。采样频率为 1 分钟。例如，如果在 10 毫秒期间写入了 1MB 数据，则此指标的值将为 100 MB/秒。请注意，当此指标超过 100 MB/秒时，由于写入吞吐量限制，可能会观察到较高的写入延迟。	每秒字节数
MemoryFragmentationRatio	指示 Valkey 或 Redis OSS 引擎的内存分配效率。某些阈值将表示不同的行为。建议的值是让碎片化大于 1.0。这是根据 INFO 的 mem_fragmentation_ratio statistic 计算得出的。	数字
MultiRegionClusterReplicationLag	在 MemoryDB 多区域集群中，MultiRegionClusterReplicationLag 测量从更新写入某个区域集群的多可用区事务日志，到该更新写入多区域集群中另一个区域集群的主节点之间所经过的时间。该指标在分片级别为每个源区域和目标区域对发出。	毫秒

指标	描述	单位
NewConnections	在此期间，服务器接受的连接总数。这是根据 INFO 的 total_connections_received 统计数据得出的。	计数
NumItemsReadFromDisk	每分钟从磁盘检索的项目总数。仅支持使用数据分层功能的集群。	计数
NumItemsWrittenToDisk	每分钟写入磁盘的项目总数。仅支持使用数据分层功能的集群。	计数
PrimaryLinkHealthStatus	此状态有两个值：0 或 1。值为 0 表示 MemoryDB 主节点中的数据未与 EC2 上的 Valkey 或 Redis OSS 引擎同步。值为 1 表示数据已同步。	布尔值
Reclaimed	密钥过期事件的总数。这是根据 INFO 的 expired_keys 统计数据得出的。	计数
ReplicationBytes	对于重复配置中的节点，ReplicationBytes 报告主项向其所有副本发送的字节数。此指标代表集群上的写入负载。这是根据 INFO 的 master_repl_offset 统计数据得出的。	字节
ReplicationDelayedWriteCommands	由于同步复制而延迟的写入命令数。复制会由于各种因素而延迟，例如网络拥塞或超过最大复制吞吐量。	计数
ReplicationLag	该指标仅适用于作为只读副本运行的节点。它代表副本在应用主节点的改动方面滞后的时间（以秒为单位）。	秒
SuccessfulWriteRequestLatency	成功写入请求的延迟。 有效统计量：平均值、总和、最小值、最大值、样本数、p0 到 p100 之间的任何百分位数。样本数仅包括成功执行的命令。可从 Valkey 7.2 开始使用。	微秒

指标	描述	单位
SuccessfulReadRequestLatency	成功读取请求的延迟。 有效统计量：平均值、总和、最小值、最大值、样本数、p0 到 p100 之间的任何百分位数。样本数仅包括成功执行的命令。可从 Valkey 7.2 开始使用。	微秒
ErrorCount	指定时间段内失败命令的总数。 有效统计量：平均值、总和、最小值、最大值	计数

以下是一些类型的命令的集合，派生自 info commandstats。commandstats 部分根据命令类型提供统计信息，包括调用次数。

有关可用命令的完整列表，请参阅命令。

指标	描述	单位
EvalBasedCmds	基于 eval 的命令的命令总数。这是根据 commandstats 统计数据得出的，方式是计算 eval 与 evalsha 的总和。	计数
GeoSpatialBasedCmds	基于地理空间的命令的命令总数。这是根据 commandstats 统计数据得出的。它是通过汇总所有地理类型的命令的总和得出的：geoadd、geodist、geohash、geopos、georadius 和 georadiusbymember。	计数
GetTypeCmds	read-only 类型命令的总数。这是根据 commandstats 统计数据得出的，方法是计算所有 read-only 类型命令（get、hget、scard、lrange 等）的总和。	计数
HashBasedCmds	基于哈希的命令总数。这是根据 commandstats 统计数据得出的，方法是计算	计数

指标	描述	单位
	所有作用于一个或多个哈希的命令（hget、hkeys、hvals、hdel 等）的总和。	
HyperLogLogBasedCmds	基于 HyperLogLog 的命令的总数。这是根据 commandstats 统计数据得出的，方法是计算所有 pf 类型的命令（pfadd、pfcount、pfmerge 等）的总和。	计数
JsonBasedCmds	基于 JSON 的命令总数。这是根据 commandstats 统计数据得出的，方式是计算所有作用于一个或多个 JSON 文档对象的命令的总和。	计数
KeyBasedCmds	基于密钥的命令总数。这是根据 commandstats 统计数据得出的，方法是计算所有作用于多个数据结构中的一个或多个键的命令（del、expire、rename 等）的总和。	计数
ListBasedCmds	基于列表的命令总数。这是根据 commandstats 统计数据得出的，方法是计算所有作用于一个或多个列表的命令（lindex、lrange、lpush、ltrim 等）的总和。	计数
PubSubBasedCmds	用于发布/订阅功能的命令总数。这是根据 commandstats 统计数据得出的，方式是计算所有用于发布/订阅功能的命令（psubscribe、publish、pubsub、punsubscribe、subscribe 和 unsubscribe）的总和。	计数
SearchBasedCmds	二级索引和搜索命令的总数，包括读取和写入命令。这是根据 commandstats 统计数据得出的，方式是计算所有作用于二级索引的搜索命令的总和。	计数
SearchBasedGetCmds	二级索引和搜索只读命令的总数。这是根据 commandstats 统计数据得出的，方式是计算所有二级索引和搜索获取命令的总和。	计数

指标	描述	单位
SearchBasedSetCmds	二级索引和搜索写入命令的总数。这是根据 commandstats 统计数据得出的，方式是计算所有二级索引和搜索集命令的总和。	计数
SearchNumberOfIndexes	索引的总数。	计数
SearchNumberOfIndexedKeys	已编入索引的键的总数	计数
SearchTotalIndexSize	所有索引占用的内存（字节）。	字节
SetBasedCmds	基于设置的命令总数。这是根据 commandstats 统计数据得出的，方法是计算所有作用于一个或多个集合的命令（scard、sdiff、sadd、sunion 等）的总和。	计数
SetTypeCmds	write 类型命令的总数。这是根据 commandstats 统计数据得出的，方法是计算对数据执行操作的所有 mutative 类型的命令（set、hset、sadd、lpop 等）的总和。	计数
SortedSetBasedCmds	基于设置的已排序命令总数。这是根据 commandstats 统计数据得出的，方法是计算所有作用于一个或多个已排序集合的命令（zcount、zrange、zrank、zadd 等）的总和。	计数
StringBasedCmds	基于字符串的命令总数。这是根据 commandstats 统计数据得出的，方法是计算所有作用于一个或多个字符串的命令（strlen、setex、setrange 等）的总和。	计数
StreamBasedCmds	基于流的命令总数。这是根据 commandstats 统计数据得出的，方法是计算所有作用于一个或多个流数据类型的命令（xrange、xlen、xadd、xdel 等）的总和。	计数

应监控哪些指标？

通过以下 CloudWatch 指标可深入了解 MemoryDB 性能。在许多情况下，我们建议对这些指标设置 CloudWatch 告警，以便您可以在性能问题出现之前采取纠正措施。

监控指标

- CPUUtilization
- EngineCPUUtilization
- SwapUsage
- 移出
- 当前连接
- 内存
- 网络
- 延迟
- 复制

CPUUtilization

这是以百分比形式报告的主机级指标。有关更多信息，请参阅 主机级指标。

对于有 2 个或更少 vCPU 的较小节点类型，可使用 `CPUUtilization` 指标来监控工作负载。

一般来说，我们建议您将阈值设置为可用 CPU 的 90%。因为 Valkey 和 Redis OSS 是单线程的，实际阈值应计算为节点总容量的一小部分。例如，假设您使用具有两个核心的节点类型。在这种情况下，CPU 使用率的阈值为 90/2，或 45%。要查找您的节点类型具有的核心（vCPU）数量，请参阅 MemoryDB 定价。

您需要根据所使用的节点中的核心数，来确定自己的阈值。如果超过此阈值，并且主要工作负载来自读取请求，则请通过添加只读副本来扩展集群。如果主要工作负载来自写入请求，我们建议您添加更多分片，以在更多主节点中分配写入工作负载。

> **Tip**
>
> 您可能可以使用向您报告有关 Valkey 或 Redis OSS 引擎核心的使用率百分比的指标 `EngineCPUUtilization`，而不是使用主机级指标 `CPUUtilization`。要了解此指标在您的节点上是否可用并了解更多信息，请参阅 MemoryDB 的指标。

对于有 4 个或更多 vCPU 的较大节点类型，您可能希望使用 EngineCPUUtilization 指标，该指标可以向您报告 Valkey 或 Redis OSS 引擎核心的使用率百分比。要了解此指标在您的节点上是否可用并了解更多信息，请参阅 MemoryDB 的指标。

EngineCPUUtilization

对于有 4 个或更多 vCPU 的较大节点类型，您可能希望使用 EngineCPUUtilization 指标，该指标可以向您报告 Valkey 或 Redis OSS 引擎核心的使用率百分比。要了解此指标在您的节点上是否可用并了解更多信息，请参阅 MemoryDB 的指标。

SwapUsage

这是以字节为单位报告的主机级指标。有关更多信息，请参阅 主机级指标。

如果 FreeableMemory CloudWatch 指标接近 0（即低于 100MB），或者 SwapUsage 指标大于 FreeableMemory 指标，则节点可能正承受内存压力。

移出

这是引擎指标。我们建议您根据应用程序需求，为此指标确定自己的警报阈值。

当前连接

这是引擎指标。我们建议您根据应用程序需求，为此指标确定自己的警报阈值。

当前连接的数量不断增加，可能表示应用程序出现问题；您需要调查应用程序行为以解决此问题。

内存

内存是 Valkey 和 Redis OSS 的核心。了解集群的内存利用率对于避免数据丢失和适应数据集的未来增长是必要的。有关节点内存利用率的统计信息可在 INFO 命令的内存部分中找到。

网络

集群网络带宽容量的决定因素之一是您选择的节点类型。有关节点的网络容量的更多信息，请参阅 Amazon MemoryDB 定价。

延迟

延迟指标 SuccessfulWriteRequestLatency 和 SuccessfulReadRequestLatency 测量 Valkey 引擎的 MemoryDB 响应请求所需的总时间。

> **Note**
>
> 当在 Valkey 客户端上启用 CLIENT REPLY 并使用 Valkey 流水线技术时，可能会出现 `SuccessfulWriteRequestLatency` 和 `SuccessfulReadRequestLatency` 指标数值偏高的情况。Valkey 流水线技术是一种通过一次性发出多个命令而不等待每个单独命令的响应来提高性能的技术。为避免数值偏高，我们建议将您的 Redis 客户端配置为使用 CLIENT REPLY OFF 来流水线化命令。

复制

可通过 `ReplicationBytes` 指标了解被复制的数据量。您可以根据复制吞吐量监控 `MaxReplicationThroughput`。建议在达到最大复制吞吐量时添加更多分片。

`ReplicationDelayedWriteCommands` 还可以提示工作负载是否超过最大复制吞吐量。有关在 MemoryDB 中使用复制的更多信息，请参阅了解 MemoryDB 复制

选择指标统计数据和周期

虽然 CloudWatch 将允许您为每个指标选择统计数据和周期，但并非所有的组合都有用。例如，CPU 利用率的平均、最小和最大统计数据均有用，但求和统计数据却无用。

适用于每个单独节点之所有 MemoryDB 示例的发布时间都为 60 秒持续时间。对于任何 60 秒期间，节点的度量标准将只包含一个单一示例。

监控 CloudWatch 指标

MemoryDB 和 CloudWatch 集成在一起，因此您可收集多种指标。您可使用 CloudWatch 监控这些指标。

> **Note**
>
> 下列示例需要使用 CloudWatch 命令行工具。有关 CloudWatch 的更多信息和下载开发工具，请参阅 CloudWatch 产品页面。

以下程序将介绍如何使用 CloudWatch 收集过去一小时内集群的存储空间统计数据。

> **Note**
>
> 下述示例中提供的 `StartTime` 和 `EndTime` 值都供说明之用。确保针对您的节点使用适当的开始和结束时间值替代示例中的相应值。

有关 MemoryDB 限制的信息，请参阅 MemoryDB 的AWS 服务限制。

监控 CloudWatch 指标（控制台）

收集集群的 CPU 利用率统计数据

1. 登录到 AWS 管理控制台 并打开 MemoryDB 控制台，网址：https://console.aws.amazon.com/memorydb/。
2. 选择您希望查看其度量标准的节点。

   > **Note**
   >
   > 若选择 20 个以上的节点，则将禁用在控制台上查看指标。

a. 在 AWS 管理控制台的集群页面上，单击一个或多个集群的名称。

显示集群的详情页面。

b. 单击位于窗口顶部的 Nodes 选项卡。

c. 在详情窗口的 Nodes 选项卡上，选择您希望查看其度量标准的节点。

一份可用 CloudWatch 度量标准列表会显示在控制台窗口的底部。

d. 单击 CPU 利用率 度量标准。

CloudWatch 控制台将打开，其中会显示您选择的指标。您可以使用统计数据和周期下拉列表框以及时间范围 选项卡来更改所显示的指标。

使用 CloudWatch CLI 监控 CloudWatch 指标

收集集群的 CPU 利用率统计数据

- 使用以下参数调用 CloudWatch 命令 aws cloudwatch get-metric-statistics（请注意，此处的开始和结束时间仅作为示例；您需要替换为适合您自己的开始和结束时间）：

对于 Linux、macOS 或 Unix：

```
aws cloudwatch get-metric-statistics CPUUtilization \
    --dimensions=ClusterName=mycluster,NodeId=0002" \
    --statistics=Average \
    --namespace="AWS/MemoryDB" \
    --start-time 2013-07-05T00:00:00 \
    --end-time 2013-07-06T00:00:00 \
    --period=60
```

对于 Windows：

```
mon-get-stats CPUUtilization ^
    --dimensions=ClusterName=mycluster,NodeId=0002" ^
    --statistics=Average ^
    --namespace="AWS/MemoryDB" ^
    --start-time 2013-07-05T00:00:00 ^
    --end-time 2013-07-06T00:00:00 ^
    --period=60
```

使用 CloudWatch API 监控 CloudWatch 指标

收集集群的 CPU 利用率统计数据

- 使用以下参数调用 CloudWatch API `GetMetricStatistics`（请注意，此处的开始和结束时间仅作为示例；您需要替换为适合您自己的开始和结束时间）：
  - `Statistics.member.1=Average`
  - `Namespace=AWS/MemoryDB`
  - `StartTime=2013-07-05T00:00:00`
  - `EndTime=2013-07-06T00:00:00`
  - `Period=60`
  - `MeasureName=CPUUtilization`
  - `Dimensions=ClusterName=mycluster,NodeId=0002`

  Example

```
http://monitoring.amazonaws.com/
    ?SignatureVersion=4
    &Action=GetMetricStatistics
    &Version=2014-12-01
    &StartTime=2013-07-16T00:00:00
    &EndTime=2013-07-16T00:02:00
    &Period=60
    &Statistics.member.1=Average
    &Dimensions.member.1="ClusterName=mycluster"
    &Dimensions.member.2="NodeId=0002"
    &Namespace=Amazon/memorydb
    &MeasureName=CPUUtilization
    &Timestamp=2013-07-07T17%3A48%3A21.746Z
    &AWS;AccessKeyId=<&AWS; Access Key ID>
    &Signature=<Signature>
```

监控 MemoryDB 事件

当集群上发生重大事件时，MemoryDB 会将通知发送到特定 Amazon SNS 主题。示例可以包括添加节点失败、添加节点成功、修改安全组等内容。通过监控关键事件，您可以了解集群的当前状态，并且能够根据事件采取相应的纠正措施。

主题

- 管理 MemoryDB Amazon SNS 通知
- 查看 MemoryDB 事件
- 事件通知和 Amazon SNS

管理 MemoryDB Amazon SNS 通知

您可以配置 MemoryDB 以使用 Amazon Simple Notification Service（Amazon SNS）发送重要集群事件的通知。在这些示例中，您将使用 Amazon SNS 主题的 Amazon 资源名称（ARN）配置集群，以便接收通知。

> **Note**
>
> 此主题假设您已经注册 Amazon SNS，同时已设置并订阅 Amazon SNS 主题。有关如何执行此操作的信息，请参阅 Amazon Simple Notification Service 开发人员指南。

添加 Amazon SNS 主题

以下部分说明了如何使用 AWS 控制台、AWS CLI 或 MemoryDB API 添加 Amazon SNS 主题。

添加 Amazon SNS 主题（控制台）

以下过程说明了如何为集群添加 Amazon SNS 主题。

> **Note**
>
> 此过程还可用于修改 Amazon SNS 主题。

为集群添加或修改 Amazon SNS 主题（控制台）

1. 登录到 AWS 管理控制台 并打开 MemoryDB 控制台，网址：https://console.aws.amazon.com/memorydb/。
2. 在 Clusters（集群）中，选择要为其添加或修改 Amazon SNS 主题 ARN 的集群。
3. 选择 Modify(修改)。
4. 在 Topic for SNS Notification（SNS 通知的主题）下的 Modify Cluster（修改集群）中，选择要添加的 SNS 主题，或选择 Manual ARN input（手动 ARN 输入）并键入 Amazon SNS 主题的 ARN。
5. 选择 Modify(修改)。

添加 Amazon SNS 主题（AWS CLI）

要为集群添加或修改 Amazon SNS 主题，请使用 AWS CLI 命令 `update-cluster`。

以下代码示例会将 Amazon SNS 主题 ARN 添加到 my-cluster。

对于 Linux、macOS 或 Unix：

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --sns-topic-arn arn:aws:sns:us-east-1:565419523791:memorydbNotifications
```

对于 Windows：

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --sns-topic-arn arn:aws:sns:us-east-1:565419523791:memorydbNotifications
```

有关更多信息，请参阅 UpdateCluster。

添加 Amazon SNS 主题（MemoryDB API）

若要为集群添加或更新 Amazon SNS 主题，请使用下列参数调用 `UpdateCluster` 操作：

- `ClusterName=my-cluster`
- `SnsTopicArn=arn%3Aaws%3Asns%3Aus-east-1%3A565419523791%3AmemorydbNotifications`

若要为集群添加或更新 Amazon SNS 主题，请调用 `UpdateCluster` 操作。

有关更多信息，请参阅 UpdateCluster。

启用和禁用 Amazon SNS 通知

您可以打开或关闭针对集群的通知。下面将介绍如何禁用 Amazon SNS 通知。

启用和禁用 Amazon SNS 通知（控制台）

使用 AWS 管理控制台 禁用 Amazon SNS 通知

1. 登录到 AWS 管理控制台 并打开 MemoryDB 控制台，网址：https://console.aws.amazon.com/memorydb/。
2. 选择要修改其通知的集群左侧的单选按钮。
3. 选择 Modify(修改)。
4. 在 Topic for SNS Notification 下的 Modify Cluster 中，选择 Disable Notifications。
5. 选择 Modify(修改)。

启用和禁用 Amazon SNS 通知（AWS CLI）

若要禁用 Amazon SNS 通知，请使用包含以下参数的命令 `update-cluster`：

对于 Linux、macOS 或 Unix：

```
aws memorydb update-cluster \
    --cluster-name my-cluster \
    --sns-topic-status inactive
```

对于 Windows：

```
aws memorydb update-cluster ^
    --cluster-name my-cluster ^
    --sns-topic-status inactive
```

启用和禁用 Amazon SNS 通知（MemoryDB API）

若要禁用 Amazon SNS 通知，请使用下列参数调用 `UpdateCluster` 操作：

- `ClusterName=my-cluster`

- SnsTopicStatus=inactive

此调用返回类似于下述信息的输出：

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateCluster
    &ClusterName=my-cluster
    &SnsTopicStatus=inactive
    &Version=2021-01-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20210801T220302Z
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Date=20210801T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

查看 MemoryDB 事件

MemoryDB 记录与您的集群、安全组和参数组有关的事件。此信息包括事件的日期和时间、事件的源名称和源类型以及事件描述。通过使用 MemoryDB 控制台、AWS CLI `describe-events` 命令或 MemoryDB API 操作 `DescribeEvents`，您可以轻松从日志中检索事件。

以下过程说明了如何查看过去 24 小时（1440 分钟）内的所有 MemoryDB 事件。

查看 MemoryDB 事件（控制台）

以下过程显示了使用 MemoryDB 控制台的事件。

使用 MemoryDB 控制台查看事件

1. 登录到 AWS 管理控制台 并打开 MemoryDB 控制台，网址：https://console.aws.amazon.com/memorydb/。
2. 在左侧导航窗格中，选择事件。

   事件屏幕显示了所有可用事件列表。列表的每一行表示一个事件，并显示事件源、事件类型（例如 cluster、parameter-group、acl、security-group 或 subnet group）、事件的 GMT 时间及事件的描述。

   通过使用 Filter，您可以指定是要查看事件列表中的所有事件，还是仅查看特定类型的事件。

查看 MemoryDB 事件（AWS CLI）

要使用 AWS CLI 生成 MemoryDB 事件的列表，请使用命令 `describe-events`。您可以使用可选参数来控制所列事件的类型、所列事件的时间范围、要列出的事件的最大数目等。

以下代码列出最多 40 个集群事件。

```
aws memorydb describe-events --source-type cluster --max-results 40
```

以下代码列出了过去 24 小时（1440 分钟）内的所有事件。

```
aws memorydb describe-events --duration 1440
```

`describe-events` 命令的输出类似于此处所示。

```
{
```

```
    "Events": [
        {
            "Date": "2021-03-29T22:17:37.781Z",
            "Message": "Added node 0001 in Availability Zone us-east-1a",
            "SourceName": "memorydb01",
            "SourceType": "cluster"
        },
        {
            "Date": "2021-03-29T22:17:37.769Z",
            "Message": "cluster created",
            "SourceName": "memorydb01",
            "SourceType": "cluster"
        }
    ]
}
```

有关更多信息（如可用参数和允许的参数值），请参阅 describe-events。

查看 MemoryDB 事件（MemoryDB API）

要使用 MemoryDB API 生成 MemoryDB 事件的列表，请使用 DescribeEvents 操作。您可以使用可选参数来控制所列事件的类型、所列事件的时间范围、要列出的事件的最大数目等。

以下代码列出了 40 个最新的集群事件。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeEvents
   &MaxResults=40
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cluster
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

以下代码列出了过去 24 小时（1440 分钟）内的集群事件。

```
https://memory-db.us-east-1.amazonaws.com/
   ?Action=DescribeEvents
   &Duration=1440
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
```

```
   &SourceType=cluster
   &Timestamp=20210802T192317Z
   &Version=2021-01-01
   &X-Amz-Credential=<credential>
```

以上操作应生成类似于以下内容的输出。

```
<DescribeEventsResponse xmlns="http://memory-db.us-east-1.amazonaws.com/
doc/2021-01-01/">
    <DescribeEventsResult>
        <Events>
            <Event>
                <Message>cluster created</Message>
                <SourceType>cluster</SourceType>
                <Date>2021-08-02T18:22:18.202Z</Date>
                <SourceName>my-memorydb-primary</SourceName>
            </Event>

 (...output omitted...)

        </Events>
    </DescribeEventsResult>
    <ResponseMetadata>
        <RequestId>e21c81b4-b9cd-11e3-8a16-7978bb24ffdf</RequestId>
    </ResponseMetadata>
</DescribeEventsResponse>
```

有关更多信息（如可用参数和允许的参数值），请参阅 DescribeEvents。

事件通知和 Amazon SNS

当集群上发生重要事件时，MemoryDB 可以使用 Amazon Simple Notification Service（SNS）发布消息。此功能可用于在连接到集群的各个节点端点的客户端计算机上刷新服务器列表。

> **Note**
>
> 有关 Amazon Simple Notification Service（SNS）的更多信息（包括定价信息和 Amazon SNS 文档链接），请参阅 Amazon SNS 产品页面。

通知会发布到指定 Amazon SNS 主题。下面是通知的要求：

- 只能为 MemoryDB 通知配置一个主题。
- 拥有 Amazon SNS 主题的 AWS 账户必须是拥有已启用通知的集群的同一账户。

MemoryDB 事件

以下 MemoryDB 事件会触发 Amazon SNS 通知：

事件名称	消息	描述
MemoryDB:AddNodeComplete	`"Modified number of nodes from %d to %d"`	节点已添加到集群，并准备就绪，可供可用。
由于空闲 IP 地址不足导致的 MemoryDB:AddNodeFailed	`"Failed to modify number of nodes from %d to %d due to insufficient free IP addresses"`	因为没有足够的可用 IP 地址，所以无法添加节点。
MemoryDB:ClusterParametersChanged	`"Updated parameter group for the cluster"` 在执行创建操作的情况下，还发送 `"Updated to use a ParameterGroup %s"`	一个或多个集群参数已更改。
MemoryDB:ClusterProvisioningComplete	`"Cluster created."`	集群预配置已完成，并且集群中的节点准备就绪，可供使用。
由于不兼容网络状态导致的 MemoryDB:ClusterProvisioningFailed	`"Failed to create cluster due to incompatible network state. %s"`	尝试将新集群启动到不存在的虚拟私有云（VPC）中。
MemoryDB:ClusterRestoreFailed	`"Restore from %s failed for node %s. %s"`	MemoryDB 无法使用快照数据填充集群。这可能是由于 Amazon S3 中不存在快照文件，或者该文件的权限不正

事件名称	消息	描述
		确。要对集群加以说明，状态将是 `restore-failed` 。您需要删除集群，重新开始。 有关更多信息，请参阅 使用外部创建的快照为新集群做种。
MemoryDB:ClusterScalingComplete	`"Succeeded applying modification to node type to %s."`	已成功纵向扩展集群。
MemoryDB:ClusterScalingFailed	`"Failed applying modification to node type to %s."`	对集群的纵向扩展操作已失败。
MemoryDB:NodeReplaceStarted	`"Recovering node %s"`	MemoryDB 已检测到运行节点的主机性能下降或无法访问，并已开始节点的替换工作。 **Note** 针对替换之节点的 DNS 分录未发生变化。 在大多数情况下，您无需在此事件发生时刷新适用于您的客户端的服务器列表。然而，某些客户端库可能停止使用节点，即使在 MemoryDB 已替换节点之后，亦是如此；在这种情况下，应用程序应该在此事件发生时刷新服务器列表。

事件名称	消息	描述
MemoryDB:NodeReplaceComplete	`"Finished recovery for node %s"`	MemoryDB 已检测到运行节点的主机性能下降或无法访问，并已完成节点的替换工作。 **Note** 针对替换之节点的 DNS 分录未发生变化。 在大多数情况下，您无需在此事件发生时刷新适用于您的客户端的服务器列表。然而，某些客户端库可能停止使用节点，即使在 MemoryDB 已替换节点之后，亦是如此；在这种情况下，应用程序应该在此事件发生时刷新服务器列表。
MemoryDB:CreateClusterComplete	`"Cluster created"`	成功创建集群。
MemoryDB:CreateClusterFailed	`"Failed to create cluster due to unsuccessful creation of its node(s)." "and""Deleting all nodes belonging to this cluster."`	未创建集群。
MemoryDB:DeleteClusterComplete	`"Cluster deleted."`	已完成集群和所有关联节点的删除工作。
MemoryDB:FailoverComplete	`"Failover to replica node %s completed"`	已成功故障转移至副本节点。

事件名称	消息	描述
MemoryDB:NodeReplacementCanceled	`"The replacement of node %s which was scheduled during the maintenance window from start time: %s, end time: %s has been canceled"`	计划替换的集群中的节点不再计划替换。
MemoryDB:NodeReplacementRescheduled	`"The replacement in maintenance window for node %s has been re-scheduled from previous start time: %s, previous end time: %s to new start time: %s, new end time: %s"`	之前计划替换的集群中的节点已计划在通知中所述的新时段内替换。 有关您可以执行的操作的信息，请参阅 替换节点。
MemoryDB:NodeReplacementScheduled	`"The node %s is scheduled for replacement during the maintenance window from start time: %s to end time: %s"`	您集群中的节点计划在通知所述的时段内替换。 有关您可以执行的操作的信息，请参阅 替换节点。
MemoryDB:RemoveNodeComplete	`"Removed node %s"`	节点已从集群中移除。
MemoryDB:SnapshotComplete	`"Snapshot %s succeeded for node %s"`	快照已成功完成。

事件名称	消息	描述
MemoryDB:SnapshotFailed	`"Snapshot %s failed for node %s"`	快照失败。有关失败原因的详细信息，请参阅该集群的事件。 要对快照加以说明，请参阅 DescribeSnapshots，状态将是 `failed`。

使用 AWS CloudTrail 记录 MemoryDB API 调用

MemoryDB 与 AWS CloudTrail 集成，后者是一项服务，可在 MemoryDB 中提供用户、角色或 AWS 服务所采取操作的记录。CloudTrail 将对 MemoryDB 的所有 API 调用都作为事件捕获，包括来自 MemoryDB 控制台的调用、来自对 MemoryDB API 操作的代码调用。如果您创建跟踪记录，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 MemoryDB 的事件）。如果您不配置跟踪，则仍可在 CloudTrail 控制台中的事件历史记录中查看最新事件。使用 CloudTrail 收集的信息，您可以确定向 MemoryDB 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《AWS CloudTrail 用户指南》https://docs.aws.amazon.com/awscloudtrail/latest/userguide/。

CloudTrail 中的 MemoryDB 信息

在您创建 AWS 账户时，将在该账户上启用 CloudTrail。当 MemoryDB 发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 AWS 服务事件一同保存在事件历史记录中。您可以在 AWS 账户中查看、搜索和下载最新事件。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 AWS 账户中的事件（包括 MemoryDB 的事件），请创建跟踪记录。通过跟踪，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有区域。此跟踪记录在 AWS 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 Amazon S3 存储桶。此外，您可以配置其他 AWS 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，请参阅下列内容：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知

- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

所有 MemoryDB 操作都由 CloudTrail 记录。例如，对 `CreateCluster`、`DescribeClusters` 和 `UpdateCluster` 操作的调用会在 CloudTrail 日志文件中生成条目。

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 IAM 用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 AWS 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 MemoryDB 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日记账条目。一个事件表示来自任何源的一个请求，包括有关请求的操作、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。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `CreateCluster` 操作。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T17:56:46Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "CreateCluster",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.create-cluster",
    "requestParameters": {
        "clusterName": "memorydb-cluster",
```

```
        "nodeType": "db.r6g.large",
        "subnetGroupName": "memorydb-subnet-group",
        "aCLName": "open-access"
    },
    "responseElements": {
        "cluster": {
            "name": "memorydb-cluster",
            "status": "creating",
            "numberOfShards": 1,
            "availabilityMode": "MultiAZ",
            "clusterEndpoint": {
                "port": 6379
            },
            "nodeType": "db.r6g.large",
            "engineVersion": "6.2",
            "enginePatchVersion": "6.2.6",
            "parameterGroupName": "default.memorydb-redis6",
            "parameterGroupStatus": "in-sync",
            "subnetGroupName": "memorydb-subnet-group",
            "tLSEnabled": true,
            "aRN": "arn:aws:memorydb:us-east-1:123456789012:cluster/memorydb-cluster",
            "snapshotRetentionLimit": 0,
            "maintenanceWindow": "tue:06:30-tue:07:30",
            "snapshotWindow": "09:00-10:00",
            "aCLName": "open-access",
            "dataTiering": "false",
            "autoMinorVersionUpgrade": true
        }
    },
    "requestID": "506fc951-9ae2-42bb-872c-98028dc8ed11",
    "eventID": "2ecf3dc3-c931-4df0-a2b3-be90b596697e",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```


下面的示例显示了一个 CloudTrail 日志条目，该条目演示了 `DescribeClusters` 操作。请注意，对于所有的 MemoryDB 描述和列出调用（`Describe*` 和 `List*`），`responseElements` 部分将被移除并显示为 `null`。


```
{
```

```
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T18:39:51Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "DescribeClusters",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.describe-clusters",
    "requestParameters": {
        "maxResults": 50,
        "showShardDetails": true
    },
    "responseElements": null,
    "requestID": "5e831993-52bb-494d-9bba-338a117c2389",
    "eventID": "32a3dc0a-31c8-4218-b889-1a6310b7dd50",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```


下面的示例显示了一个 CloudTrail 日志条目，该条目记录了 `UpdateCluster` 操作。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T19:23:20Z",
```

```
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "UpdateCluster",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.update-cluster",
    "requestParameters": {
        "clusterName": "memorydb-cluster",
        "snapshotWindow": "04:00-05:00",
        "shardConfiguration": {
            "shardCount": 2
        }
    },
    "responseElements": {
        "cluster": {
            "name": "memorydb-cluster",
            "status": "updating",
            "numberOfShards": 2,
            "availabilityMode": "MultiAZ",
            "clusterEndpoint": {
                "address": "clustercfg.memorydb-cluster.cde8da.memorydb.us-
east-1.amazonaws.com",
                "port": 6379
            },
            "nodeType": "db.r6g.large",
            "engineVersion": "6.2",
            "EnginePatchVersion": "6.2.6",
            "parameterGroupName": "default.memorydb-redis6",
            "parameterGroupStatus": "in-sync",
            "subnetGroupName": "memorydb-subnet-group",
            "tLSEnabled": true,
            "aRN": "arn:aws:memorydb:us-east-1:123456789012:cluster/memorydb-cluster",
            "snapshotRetentionLimit": 0,
            "maintenanceWindow": "tue:06:30-tue:07:30",
            "snapshotWindow": "04:00-05:00",
            "autoMinorVersionUpgrade": true,
            "DataTiering": "false"
        }
    },
    "requestID": "dad021ce-d161-4365-8085-574133afab54",
    "eventID": "e0120f85-ab7e-4ad4-ae78-43ba15dee3d8",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
```

```
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```

下面的示例显示了一个 CloudTrail 日志条目，该条目演示了 `CreateUser` 操作。请注意，对于包含敏感数据的 MemoryDB 调用，该数据将在相应的 CloudTrail 事件中进行编辑，如以下 `requestParameters` 部分所示。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "EKIAUAXQT3SWDEXAMPLE",
        "arn": "arn:aws:iam::123456789012:user/john",
        "accountId": "123456789012",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "userName": "john"
    },
    "eventTime": "2021-07-10T19:56:13Z",
    "eventSource": "memorydb.amazonaws.com",
    "eventName": "CreateUser",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.01",
    "userAgent": "aws-cli/2.2.29 Python/3.9.6 Darwin/19.6.0 source/x86_64 prompt/off
 command/memorydb.create-user",
    "requestParameters": {
        "userName": "memorydb-user",
        "authenticationMode": {
            "type": "password",
            "passwords": [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "accessString": "~* &* -@all +@read"
    },
    "responseElements": {
        "user": {
            "name": "memorydb-user",
            "status": "active",
            "accessString": "off ~* &* -@all +@read",
            "aCLNames": [],
            "minimumEngineVersion": "6.2",
            "authentication": {
```

```
                "type": "password",
                "passwordCount": 1
            },
            "aRN": "arn:aws:memorydb:us-east-1:123456789012:user/memorydb-user"
        }
    },
    "requestID": "ae288b5e-80ab-4ff8-989a-5ee5c67cd193",
    "eventID": "ed096e3e-16f1-4a23-866c-0baa6ec769f6",
    "readOnly": false,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "eventCategory": "Management"
}
```


适用于 MemoryDB 的合规性验证

作为多个 AWS 合规性计划的一部分，第三方审核员将评测 MemoryDB 的安全性和合规性。这包括：

- 支付卡行业数据安全标准（PCI DSS）。有关更多信息，请参阅 PCI DSS。
- 《健康保险流通与责任法案》商业伙伴协议（HIPAA BAA）。有关更多信息，请参阅 HIPAA 合规性。
- 系统和组织控制（SOC）1、2 和 3。有关更多信息，请参阅 SOC。
- 联邦风险与授权管理项目（FedRAMP）中级。有关更多信息，请参阅 FedRAMP。
- ISO/IEC 27001:2013、27017:2015、27018:2019 和 ISO/IEC 9001:2015。有关更多信息，请参阅 AWS ISO 和 CSA STAR 认证和服务。

有关特定合规性计划范围内的 AWS 服务的列表，请参阅按合规性计划提供的范围内 AWS 服务。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅在 AWS Artifact 中下载报告、。

您使用 MemoryDB 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。AWS 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 - 这些部署指南讨论了架构注意事项，并提供了在AWS上部署基于安全性和合规性的基准环境的步骤。
- AWS 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地点。

- 《AWS Config 开发人员指南》中的 Evaluating Resources with Rules – AWS Config 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。
- AWS Security Hub CSPM – 此AWS服务提供了AWS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操。
- AWS 审计管理器 – 此 AWS 服务可帮助您持续审计 AWS 使用情况，以简化您管理风险和符合法规及行业标准的方式。

MemoryDB 中的基础设施安全性

作为一项托管式服务，MemoryDB 由 Amazon Web Services：安全流程概览白皮书中所述的 AWS 全球网络安全程序提供保护。

您可以使用 AWS 发布的 API 调用通过网络访问 MemoryDB。客户端必须支持传输层安全性（TLS）1.2 或更高版本。我们建议使用 TLS 1.3 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。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 AWS Security Token Service（AWS STS）生成临时安全凭证来对请求进行签名。

互联网络流量隐私

MemoryDB 使用以下技术保护您的数据免受未经授权的访问：

- MemoryDB 和 Amazon VPC 说明了安装所需的安全组类型。
- MemoryDB API 和接口 VPC 端点（AWS PrivateLink） 让您可以在 VPC 和 MemoryDB API 端点之间建立私有连接。
- MemoryDB 中的身份和访问管理 用于授予和限制用户、组和角色的操作。

MemoryDB 和 Amazon VPC

Amazon Virtual Private Cloud（Amazon VPC）服务定义一个与传统数据中心非常相似的虚拟网络。在通过 Amazon VPC 配置您的虚拟私有云（VPC）时，您可以选择它的 IP 地址范围、创建子网并配置路由表、网关和安全设置。您还可以将集群添加到虚拟网络，并使用 Amazon VPC 安全组控制对集群的访问。

本部分说明如何在 VPC 中手动配置 MemoryDB 集群。这些信息适用于希望更深入地了解 MemoryDB 和 Amazon VPC 如何协同工作的用户。

主题

- 了解 MemoryDB 和 VPC
- 用于访问 Amazon VPC 中 MemoryDB 集群的访问模式
- 创建虚拟私有云（VPC）

了解 MemoryDB 和 VPC

MemoryDB 已与 Amazon VPC 完全集成。对于 MemoryDB 用户，这具有以下意义：

- MemoryDB 始终在 VPC 中启动您的集群。
- 如果您不熟悉 AWS，则系统将为您自动创建一个默认 VPC。
- 如果您有默认 VPC 并且在启动集群时未指定子网，该集群会启动到您的默认 Amazon VPC 中。

有关更多信息，请参阅检测支持的平台以及是否具有默认 VPC。

使用 Amazon VPC，您可以在非常类似于传统数据中心的 AWS 云中创建虚拟网络。您可以配置您的 VPC，包括选择其 IP 地址范围、创建子网以及配置路由表、网关和安全设置。

MemoryDB 可以管理软件升级、修补、故障检测和恢复。

VPC 中的 MemoryDB 概述

- VPC 是 AWS 云的一个独立部分，分配有自己的 IP 地址数据块。
- 互联网网关将您的 VPC 直接连接到互联网，并提供对其他 AWS 资源 [例如在 VPC 外部运行的 Amazon Simple Storage Service（Amazon S3）] 的访问。
- Amazon VPC 子网是 VPC 的 IP 地址范围的一部分，在其中您可以根据您的安全和操作需求隔离 AWS 资源。
- Amazon VPC 安全组可以控制 MemoryDB 集群和 Amazon EC2 实例的入站和出站流量。
- 您可以在子网中启动 MemoryDB 集群。节点具有子网地址范围内的私有 IP 地址。
- 您也可以在子网中启动 Amazon EC2 实例。每个 Amazon EC2 实例都具有一个在子网地址范围内的私有 IP 地址。Amazon EC2 实例可以连接到同一子网中的任何节点。
- 要可以从互联网访问您的 VPC 中的 Amazon EC2 实例，您需要为此实例分配静态公有地址（称为弹性 IP 地址）。

先决条件

要在 VPC 中创建 MemoryDB 集群，您的 VPC 必须满足下列要求：

- 您的 VPC 必须允许非专用 Amazon EC2 实例。不能在为专用实例租赁配置的 VPC 中使用 MemoryDB。
- 必须为您的 VPC 定义子网组。MemoryDB 使用该子网组选择与节点关联的子网和子网中的 IP 地址。

- 必须为您的 VPC 定义一个安全组，或者您可以使用提供的默认缓存安全组。
- 每个子网的 CIDR 块必须足够大，以便为 MemoryDB 提供可在维护活动期间使用的备用 IP 地址。

路由和安全性

您可以在 VPC 中配置路由，以控制流量的流向（例如，流向互联网网关或虚拟私有网关）。使用互联网网关，您的 VPC 可以直接访问不在您的 VPC 中运行的其他 AWS 资源。如果您选择只使用一个虚拟专用网关连接至贵组织的本地网络，那么您可以通过 VPN 设置您的互联网入口流量路由，并使用本地安全策略和防火墙来控制出口。在此种情况下，当您通过互联网访问 AWS 资源时，便会产生额外的带宽费用。

您可以使用 Amazon VPC 安全组来帮助保护 Amazon VPC 中的 MemoryDB 集群和 Amazon EC2 实例。安全组在实例级上（而非子网级上）与防火墙的功能类似。

> **Note**
>
> 我们强烈建议您使用 DNS 名称连接到您的节点，因为基础 IP 地址可能会随时间而改变。

Amazon VPC 文档

Amazon VPC 有一套专门文档，介绍如何创建和使用您的 Amazon VPC。下表显示了在 Amazon VPC 指南何处查找信息。

描述	文档
如何开始使用 Amazon VPC	Amazon VPC 入门
如何通过 AWS 管理控制台 使用 Amazon VPC	Amazon VPC User Guide
所有 Amazon VPC 命令的完整描述	Amazon EC2 命令行参考（Amazon VPC 命令可在 Amazon EC2 参考中找到）
Amazon VPC API 操作、数据类型和错误的完整描述	Amazon EC2 API 参考（Amazon VPC API 操作可在 Amazon EC2 参考中找到）
关于需要在您终止可选的 IPsec VPN 连接时对网关进行配置的网络管理员之信息	AWS Site-to-Site VPN 是什么？

有关 Amazon Virtual Private Cloud 的更多详细信息，请参阅 Amazon Virtual Private Cloud。

用于访问 Amazon VPC 中 MemoryDB 集群的访问模式

MemoryDB 支持以下场景来访问 Amazon VPC 中的集群：

目录

- 当 MemoryDB 集群和 Amazon EC2 实例位于同一 Amazon VPC 中时访问该集群
- 当 MemoryDB 集群和 Amazon EC2 实例位于不同 Amazon VPC 时访问该集群
  - 当 MemoryDB 集群和 Amazon EC2 实例位于同一区域的不同 Amazon VPC 时访问该集群
    - 使用 Transit Gateway
  - 当 MemoryDB 集群和 Amazon EC2 实例位于不同区域的不同 Amazon VPC 时访问该集群
    - 使用 Transit VPC
- 从在客户的数据中心中运行的应用程序访问 MemoryDB 集群
  - 使用 VPN 连接从在客户的数据中心中运行的应用程序访问 MemoryDB 集群
  - 使用 Direct Connect 从在客户的数据中心中运行的应用程序访问 MemoryDB 集群

当 MemoryDB 集群和 Amazon EC2 实例位于同一 Amazon VPC 中时访问该集群

最常见的使用场景是，当 EC2 实例上部署的应用程序需要连接到同一 VPC 中的集群时。

要管理同一 VPC 中 EC2 实例与集群之间的访问，最简单方法如下所示：

1. 为集群创建 VPC 安全组。此安全组可用于限制对集群的访问权限。例如，可为此安全组创建自定义规则，允许使用您创建集群时分配给该集群的端口以及将用来访问集群的 IP 地址进行 TCP 访问。

   MemoryDB 集群的默认端口为 6379。
2. 为 EC2 实例（Web 和应用程序服务器）创建 VPC 安全组。如果需要，此安全组可允许通过 VPC 的路由表从 Internet 访问 EC2 实例。例如，您可设置此安全组的规则以允许通过端口 22 对 EC2 实例进行 TCP 访问。
3. 在集群的安全组中创建自定义规则，允许从为 EC2 实例创建的安全组连接。这将允许安全组的任何成员均可访问集群。

在 VPC 安全组中创建允许从另一安全组连接的规则

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc。

2. 在左侧导航窗格中，选择安全组。
3. 选择或创建一个要用于集群的安全组。在入站规则下，选择编辑入站规则，然后选择添加规则。此安全组将允许访问其他安全组的成员。
4. 从 Type 中选择 Custom TCP Rule。

   a. 对于 Port Range，指定在创建集群时使用的端口。

      MemoryDB 集群的默认端口为 6379。

   b. 在 Source 框中，开始键入安全组的 ID。从列表中选择要用于 Amazon EC2 实例的安全组。
5. 完成后选择 Save。

当 MemoryDB 集群和 Amazon EC2 实例位于不同 Amazon VPC 时访问该集群

当您的集群与您用来访问它的 EC2 实例位于不同 VPC 中时，可通过多种方式访问集群。如果集群和 EC2 实例位于不同的 VPC 中但位于相同区域中，则可使用 VPC 对等。如果集群和 EC2 实例位于不同的区域中，您可以在两个区域之间创建 VPN 连接。

主题

- 当 MemoryDB 集群和 Amazon EC2 实例位于同一区域的不同 Amazon VPC 时访问该集群
- 当 MemoryDB 集群和 Amazon EC2 实例位于不同区域的不同 Amazon VPC 时访问该集群

当 MemoryDB 集群和 Amazon EC2 实例位于同一区域的不同 Amazon VPC 时访问该集群

集群由同一区域中不同 Amazon VPC 中的 Amazon EC2 实例访问 – VPC 对等连接

VPC 对等连接是两个 VPC 之间的网络连接，通过此连接，您可以使用私有 IP 地址在这两个 VPC 之间路由流量。这两个 VPC 中的实例可以彼此通信，就像它们在同一网络中一样。您可以在您自己的 Amazon VPC 之间创建 VPC 对等连接，也可以在它们与同一区域内其他 AWS 账户中的 Amazon VPC 之间创建该连接。要了解有关 Amazon VPC 对等连接的更多信息，请参阅 VPC 文档。

通过对等连接访问不同 Amazon VPC 中的集群

1. 确保两个 VPC 的 IP 范围不重叠，否则无法使其对等。
2. 使两个 VPC 对等。有关更多信息，请参阅创建并接受 Amazon VPC 对等连接。
3. 更新路由表。有关更多信息，请参阅为 VPC 对等连接更新路由表

4. 修改 MemoryDB 集群的安全组，以允许来自对等 VPC 中的应用程序安全组的入站连接。有关更多信息，请参阅引用对等 VPC 安全组。

通过对等连接访问集群将产生额外的数据传输费用。

使用 Transit Gateway

通过中转网关，您可以连接同一 AWS 区域中的 VPC 与 VPN 连接并在它们之间路由流量。中转网关可跨 AWS 账户发挥作用，您可以使用 AWS Resource Access Manager 与其他账户共享您的中转网关。在您与另一个 AWS 账户共享中转网关后，账户拥有者可以将其 VPC 挂载到您的中转网关。任一账户的用户都可以随时删除此挂载。

您可以在中转网关上启用多播，然后创建一个中转网关多播域，允许通过与域关联的 VPC 挂载，将多播流量从多播源发送到多播组成员。

您还可以在不同 AWS 区域的中转网关之间创建对等连接挂载。这使您能够跨不同区域在中转网关的挂载之间路由流量。

有关更多信息，请参阅中转网关。

当 MemoryDB 集群和 Amazon EC2 实例位于不同区域的不同 Amazon VPC 时访问该集群

使用 Transit VPC

创建一个可充当全球网络中转中心的中转 VPC 是使用 VPC 对等连接的另一种方法，同时也是连接多个地理位置分散的 VPC 和远程网络的另一种常见策略。传输 VPC 可简化网络管理，并最大程度地减少连接多个 VPC 和远程网络时所需的连接数。此设计可以节省时间和工作量并降低成本，因为它的实施几乎消除了在托管传输中心建立实体办事处或部署物理网络设备时所需的传统费用。

跨不同区域中的不同 VPC 进行连接

建立 Transit Amazon VPC 后，在一个区域中的“辐射型”VPC 中部署的应用程序可以连接到另一个区域中的“辐射型”VPC 中的 MemoryDB 集群。

访问在不同 AWS 区域的不同 VPC 中的集群

1. 部署传输 VPC 解决方案。有关更多信息，请参阅 AWS Transit Gateway。
2. 更新应用和 VPC 中的路由表，以通过 VGW（虚拟专用网关）和 VPN 设备路由流量。对于使用边界网关协议（BGP）的动态路由，可自动传播您的路由。

3. 修改 MemoryDB 集群的安全组，以允许来自该应用程序实例 IP 范围的入站连接。请注意，这种情况下，无法引用该应用程序服务器安全组。

跨区域访问集群将造成网络连接延迟并产生其他跨区域数据传输费用。

从在客户的数据中心中运行的应用程序访问 MemoryDB 集群

另一种可能的方案是混合架构，客户数据中心内的客户端或应用程序可能需要访问 VPC 中的 MemoryDB 集群。此方案也受支持，前提是客户的 VPC 和数据中心之间已通过 VPN 或 Direct Connect 建立连接。

主题

- 使用 VPN 连接从在客户的数据中心中运行的应用程序访问 MemoryDB 集群
- 使用 Direct Connect 从在客户的数据中心中运行的应用程序访问 MemoryDB 集群

使用 VPN 连接从在客户的数据中心中运行的应用程序访问 MemoryDB 集群

从数据中心通过 VPN 连接到 MemoryDB

通过 VPN 连接从本地应用程序中访问 VPC 中的集群

1. 通过向 VPC 中添加硬件虚拟专用网关来建立 VPN 连接。有关更多信息，请参阅在您的 VPC 中添加硬件虚拟专用网关。
2. 更新部署 MemoryDB 集群时所在子网的 VPC 路由表，以允许来自本地应用程序服务器的流量。对于使用 BGP 的动态路由，可自动传播您的路由。
3. 修改 MemoryDB 集群的安全组，以允许来自本地应用程序服务器的入站连接。

通过 VPN 连接访问集群将造成网络连接延迟并产生其他数据传输费用。

使用 Direct Connect 从在客户的数据中心中运行的应用程序访问 MemoryDB 集群

从数据中心或通过 Direct Connect 连接到 MemoryDB

使用 Direct Connect 从在您的网络中运行的应用程序访问 MemoryDB 集群

1. 建立 Direct Connect 连接。有关更多信息，请参阅 AWS Direct Connect 入门。

2. 修改 MemoryDB 集群的安全组，以允许来自本地应用程序服务器的入站连接。

通过 DX 连接访问集群可能会造成网络连接延迟并产生其他数据传输费用。

创建虚拟私有云（VPC）

在本示例中，您基于 Amazon VPC 服务创建一个 虚拟私有云（VPC），其中每个可用区域都有一个私有子网。

创建 VPC（控制台）

在 Amazon Virtual Private Cloud 内部创建 MemoryDB 集群

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
2. 在 VPC 控制面板上，选择 Create VPC（创建 VPC）。
3. 在要创建的 Resources（资源）下，选择 VPC and more（VPC 等）。
4. 在可用区（AZ）数量下，选择要在其中启动子网的可用区数量。
5. 在 Number of public subnets（公有子网数量）下，选择要添加到 VPC 的公有子网数量。
6. 在 Number of private subnets（私有子网数量）下，选择要添加到 VPC 的私有子网数量。

   > **Tip**
   >
   > 记录您的子网标识符，以及哪个是公有的，哪个是私有的。稍后，当您启动集群以及向 Amazon VPC 添加 Amazon EC2 实例时，您将需要此类信息。

7. 创建 Amazon VPC 安全组。您将对集群和 Amazon EC2 实例使用此安全组。

   a. 在 AWS 管理控制台 的左侧导航窗格中，选择安全组。

   b. 选择创建安全组。

   c. 在相应的框内，为您的安全组输入名称和描述。对于 VPC，选择 VPC 标识符。

   d. 根据需要完成所有设置后，选择 Yes, Create。

8. 为您的安全组定义一个网络入口规则。此规则将允许您使用 Secure Shell（SSH）连接至 Amazon EC2 实例。

   a. 在左侧导航窗格中，选择安全组。

   b. 在列表中找到您的安全组，然后选择它。

   c. 在 Security Group 下，选择 Inbound 选项卡。在 Create a new rule 框中，选择 SSH，然后选择 Add Rule。

      为新入站规则设置以下值以允许 HTTP 访问：

- 类型：HTTP
- 来源：0.0.0.0/0

d. 为新入站规则设置以下值以允许 HTTP 访问：

- 类型：HTTP
- 来源：0.0.0.0/0

选择 Apply Rule Changes。

现在，您已准备就绪，可在 VPC 中创建子网组并创建集群。

子网和子网组

子网组是您可为在 Amazon Virtual Private Cloud（VPC）环境中运行的集群指定的子网（通常为私有子网）集合。

在 Amazon VPC 中创建集群时，请指定一个子网组或使用提供的默认子网组。MemoryDB 使用该子网组选择与节点关联的子网和子网中的 IP 地址。

本部分介绍如何创建和利用子网以及子网组来管理对 MemoryDB 资源的访问。

有关 Amazon VPC 环境中子网组使用情况的更多信息，请参阅 步骤 3：授予对集群的访问权限。

支持的 MemoryDB 可用区 ID

区域名称/区域	支持的 AZ ID		
美国东部（俄亥俄州）区域 us-east-2	use2-az1, use2-az2, use2-az3		
美国东部（弗吉尼亚州北部）区域 us-east-1	use1-az1, use1-az2, use1-az4, use1-az5, use1-az6		
美国西部（北加利福尼亚）区域	usw1-az1, usw1-az2, usw1-az3		

区域名称/区域	支持的 AZ ID		
us-west-1			
美国西部（俄勒冈州）区域 us-west-2	usw2-az1, usw2-az2, usw2-az3, usw2-az4		
加拿大（中部）区域 ca-central-1	cac1-az1, cac1-az2, cac1-az4		
亚太地区（香港）区域 ap-east-1	ape1-az1, ape1-az2, ape1-az3		
亚太地区（孟买）区域 ap-south-1	aps1-az1, aps1-az2, aps1-az3		
Asia Pacific（Tokyo）Region ap-northeast-1	apne1-az1, apne1-az2, apne1-az4		
亚太地区（首尔）区域 ap-northeast-2	apne2-az1, apne2-az2, apne2-az3		
亚太地区（新加坡）区域 ap-southeast-1	apse1-az1, apse1-az2, apse1-az3		

区域名称/区域	支持的 AZ ID		
亚太地区（悉尼）区域 ap-southeast-2	apse2-az1, apse2-az2, apse2-az3		
欧洲地区（法兰克福）区域 eu-central-1	euc1-az1, euc1-az2, euc1-az3		
欧洲地区（爱尔兰）区域 eu-west-1	euw1-az1, euw1-az2, euw1-az3		
欧洲地区（伦敦）区域 eu-west-2	euw2-az1, euw2-az2, euw2-az3		
欧洲地区（巴黎）区域 eu-west-3	euw3-az1, euw3-az2, euw3-az3		
欧洲地区（斯德哥尔摩）区域 eu-north-1	eun1-az1, eun1-az2, eun1-az3		
欧洲地区（米兰） eu-south-1	eus1-az1, eus1-az2, eus1-az3		
南美洲（圣保罗）区域 sa-east-1	sae1-az1, sae1-az2, sae1-az3		

区域名称/区域	支持的 AZ ID		
中国（北京）区域 cn-north-1	cnn1-az1, cnn1-az2		
中国（宁夏）区域 cn-northwest-1	cnw1-az1, cnw1-az2, cnw1-az3		
us-gov-east-1	usge1-az1, usge1-az2, usge1-az3		
us-gov-west-1	usgw1-az1, usgw1-az2, usgw1-az3		
欧洲地区（西班牙）区域 eu-south-2	eus2-az1, eus2-az2, eus2-az3		

主题

- MemoryDB 与 IPV6
- 创建子网组
- 更新子网组
- 查看子网组详细信息
- 删除子网组

MemoryDB 与 IPV6

您可以通过提供具有双栈和仅支持 ipv6 子网的子网组，使用 Valkey 和 Redis OSS 引擎创建新的双栈和仅支持 ipv6 的集群。您不能更改现有集群的网络类型。

通过此功能，您可以：

- 在双栈子网上创建仅支持 ipv4 和双栈集群。
- 在仅支持 ipv6 的子网上创建仅支持 ipv6 的集群。
- 创建新的子网组以支持仅支持 ipv4、双栈和仅支持 ipv6 的子网。
- 修改现有子网组以包含基础 VPC 中的其他子网。
- 修改子网组中的现有子网
  - 向配置为支持 IPv6 的子网组添加仅支持 IPv6 的子网
  - 向配置为支持 IPv4 和双栈的子网组添加 IPv4 或双栈子网
- 对于双栈和 ipv6 集群，通过引擎发现命令发现集群中所有具有 ipv4 或 ipv6 地址的节点。这些发现命令包括 `redis_info`、`redis_cluster` 及类似命令。
- 对于双栈和 ipv6 集群，通过 DNS 发现命令发现集群中所有节点的 ipv4 和 ipv6 地址。

创建子网组

创建新的子网组时，请注意可用 IP 地址的数量。如果子网的可用 IP 地址非常少，您可能会在可以向集群添加更多节点方面受到限制。要解决此问题，您可以将一个或多个子网分配给子网组，以便在集群的可用区中拥有足够数量的 IP 地址。之后，便可向您的集群中添加更多节点。

以下过程演示如何创建名为 `mysubnetgroup` 的子网组（控制台、AWS CLI 和 MemoryDB API）。

创建子网组（控制台）

以下过程介绍如何创建子网组（控制台）。

创建子网组（控制台）

1. 登录 AWS 管理控制台并通过以下网址打开 MemoryDB 控制台：https://console.aws.amazon.com/memorydb/。
2. 在左侧导航窗格中，选择子网组。
3. 选择 Create Subnet Group。
4. 在创建子网组页面中，执行以下操作：

a. 在 Name 框中，为子网组键入名称。

集群命名约束如下：

- 必须包含 1 – 40 个字母数字字符或连字符。
- 必须以字母开头。
- 不能包含两个连续连字符。
- 不能以连字符结束。

b. 在 Description 框中，为子网组键入描述。

c. 在 VPC ID 框中，选择您创建的 Amazon VPC。如果您尚未创建 VPC，请选择创建 VPC 按钮，然后按照步骤创建一个。

d. 在选定子网中，选择私有子网的可用区和 ID，然后选择选择。

5. 对于标签，请选择性地应用标签来搜索和筛选子网或跟踪 AWS 成本。
6. 根据需要完成所有设置后，选择创建。
7. 在出现的确认信息中，选择 Close。

您的新子网组显示在 MemoryDB 控制台的子网组列表中。您可以在窗口底部选择子网组以查看详细信息，例如与此组关联的所有子网。

创建子网组（AWS CLI）

在命令提示符处，使用命令 `create-subnet-group` 创建子网组。

对于 Linux、macOS 或 Unix：

```
aws memorydb create-subnet-group \
    --subnet-group-name mysubnetgroup \
    --description "Testing" \
    --subnet-ids subnet-53df9c3a
```

对于 Windows：

```
aws memorydb create-subnet-group ^
    --subnet-group-name mysubnetgroup ^
    --description "Testing" ^
    --subnet-ids subnet-53df9c3a
```

该命令应该生成类似于下述信息的输出：

```
    {
        "SubnetGroup": {
            "Subnets": [
                {
                    "Identifier": "subnet-53df9c3a",
                    "AvailabilityZone": {
                    "Name": "us-east-1a"
                    }
                }
            ],
            "VpcId": "vpc-3cfaef47",
            "Name": "mysubnetgroup",
            "ARN": "arn:aws:memorydb:us-east-1:012345678912:subnetgroup/
mysubnetgroup",
            "Description": "Testing"
        }
    }
```

有关更多信息，请参阅 AWS CLI 主题 create-subnet-group。

创建子网组（MemoryDB API）

通过使用 MemoryDB API，调用带以下参数的 `CreateSubnetGroup`：

- `SubnetGroupName=`*`mysubnetgroup`*
- `Description=`*`Testing`*
- `SubnetIds.member.1=`*`subnet-53df9c3a`*

更新子网组

您可以更新子网组的描述，或者修改与子网组关联的子网 ID 列表。如果集群目前正在使用子网组中的子网，那么您不能删除该子网的 ID。

以下过程介绍如何更新子网组。

更新子网组（控制台）

更新子网组

1. 登录到 AWS 管理控制台 并打开 MemoryDB 控制台，网址：https://console.aws.amazon.com/memorydb/。
2. 在左侧导航窗格中，选择子网组。
3. 在子网组列表中，选择您希望修改的一个子网组。
4. 名称、VPCId 和描述字段不可修改。
5. 在选定子网部分，单击管理对子网所需可用区进行任何更改。要保存您的更改，请选择 Save（保存）。

更新子网组（AWS CLI）

在命令提示符处，使用命令 `update-subnet-group` 更新子网组。

对于 Linux、macOS 或 Unix：

```
aws memorydb update-subnet-group \
    --subnet-group-name mysubnetgroup \
    --description "New description" \
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

对于 Windows：

```
aws memorydb update-subnet-group ^
    --subnet-group-name mysubnetgroup ^
    --description "New description" ^
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

该命令应该生成类似于下述信息的输出：

```
{
```

```
    "SubnetGroup": {
        "VpcId": "vpc-73cd3c17",
        "Description": "New description",
        "Subnets": [
            {
                "Identifier": "subnet-42dcf93a",
                "AvailabilityZone": {
                    "Name": "us-east-1a"
                }
            },
            {
                "Identifier": "subnet-48fc12a9",
                "AvailabilityZone": {
                    "Name": "us-east-1a"
                }
            }
        ],
        "Name": "mysubnetgroup",
        "ARN": "arn:aws:memorydb:us-east-1:012345678912:subnetgroup/mysubnetgroup",
    }
}
```

有关更多信息，请参阅 AWS CLI 主题 update-subnet-group。

更新子网组（MemoryDB API）

通过使用 MemoryDB API，调用带以下参数的 UpdateSubnetGroup：

- SubnetGroupName=*mysubnetgroup*
- 要更改其值的任何其他参数。此示例使用 Description=*New%20description* 更改子网组的描述。

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateSubnetGroup
    &Description=New%20description
    &SubnetGroupName=mysubnetgroup
    &SubnetIds.member.1=subnet-42df9c3a
    &SubnetIds.member.2=subnet-48fc21a9
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
```

```
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20141201T220302Z
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Signature=<signature>
    &X-Amz-SignedHeaders=Host
```

> **Note**
>
> 当您创建新的子网组时，请记下可用 IP 地址的数量。如果子网的可用 IP 地址非常少，您可能会在可以向集群添加更多节点方面受到限制。要解决此问题，您可以将一个或多个子网分配给子网组，以便在集群的可用区中拥有足够数量的 IP 地址。之后，便可向您的集群中添加更多节点。

查看子网组详细信息

以下过程介绍如何查看子网组的详细信息。

查看子网组详细信息（控制台）

查看子网组详细信息（控制台）

1. 登录到 AWS 管理控制台 并打开 MemoryDB 控制台，网址：https://console.aws.amazon.com/memorydb/。
2. 在左侧导航窗格中，选择子网组。
3. 在子网组页面上，在名称下选择子网组，或者在搜索栏中输入子网组名称。
4. 在子网组页面上，在名称下选择子网组，或者在搜索栏中输入子网组名称。
5. 在子网组设置下，查看子网组的名称、描述、VPC ID 和 Amazon 资源名称（ARN）。
6. 在子网下，查看子网组的可用区、子网 ID 和 CIDR 块
7. 在标签下，查看与子网组关联的任何标签。

查看子网组详细信息（AWS CLI）

在命令提示符处，使用命令 `describe-subnet-groups` 查看指定子网组的详细信息。

对于 Linux、macOS 或 Unix：

```
aws memorydb describe-subnet-groups \
    --subnet-group-name mysubnetgroup
```

对于 Windows：

```
aws memorydb describe-subnet-groups ^
    --subnet-group-name mysubnetgroup
```

该命令应该生成类似于下述信息的输出：

```
{
  "subnetgroups": [
    {
      "Subnets": [
        {
          "Identifier": "subnet-060cae3464095de6e",
          "AvailabilityZone": {
            "Name": "us-east-1a"
          }
        },
        {
          "Identifier": "subnet-049d11d4aa78700c3",
          "AvailabilityZone": {
            "Name": "us-east-1c"
          }
        },
        {
          "Identifier": "subnet-0389d4c4157c1edb4",
          "AvailabilityZone": {
            "Name": "us-east-1d"
          }
        }
      ],
      "VpcId": "vpc-036a8150d4300bcf2",
      "Name": "mysubnetgroup",
      "ARN": "arn:aws:memorydb:us-east-1:53791xzzz7620:subnetgroup/mysubnetgroup",
      "Description": "test"
    }
  ]
}
```

要查看所有子网组的详细信息，请使用相同的命令，但无需指定子网组名称。

```
aws memorydb describe-subnet-groups
```

有关更多信息，请参阅 AWS CLI 主题 describe-subnet-groups。

查看子网组（MemoryDB API）

通过使用 MemoryDB API，调用带以下参数的 DescribeSubnetGroups：

SubnetGroupName=*mysubnetgroup*

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=UpdateSubnetGroup
    &Description=New%20description
    &SubnetGroupName=mysubnetgroup
    &SubnetIds.member.1=subnet-42df9c3a
    &SubnetIds.member.2=subnet-48fc21a9
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20211801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20210801T220302Z
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Signature=<signature>
    &X-Amz-SignedHeaders=Host
```

删除子网组

如果您决定不再需要您的子网组，则可删除它。如果集群目前正在使用某个子网组，则无法删除该子网组。如果删除启用了多可用区的集群上的某个子网组会使集群保留的子网少于两个，您也无法删除该子网组。您必须先取消勾选多可用区，然后删除子网。

以下过程介绍如何删除子网组。

删除子网组（控制台）

删除子网组

1. 登录到 AWS 管理控制台 并打开 MemoryDB 控制台，网址：https://console.aws.amazon.com/memorydb/。
2. 在左侧导航窗格中，选择子网组。
3. 在子网组列表中，选择要删除的子网组，再选择操作，然后选择删除。

> **Note**
>
> 您无法删除默认子网组或与任何集群关联的子网组。

4. 删除子网组确认屏幕随即出现。
5. 要删除子网组，请在确认文本框中输入 `delete`。要保留子网组，请选择取消。

删除子网组（AWS CLI）

通过使用 AWS CLI，调用带以下参数的命令 delete-subnet-group：

- `--subnet-group-name` *`mysubnetgroup`*

对于 Linux、macOS 或 Unix：

```
aws memorydb delete-subnet-group \
    --subnet-group-name mysubnetgroup
```

对于 Windows：

```
aws memorydb delete-subnet-group ^
    --subnet-group-name mysubnetgroup
```

有关更多信息，请参阅 AWS CLI 主题 delete-subnet-group。

删除子网组（MemoryDB API）

通过使用 MemoryDB API，调用带以下参数的 DeleteSubnetGroup：

- SubnetGroupName=*mysubnetgroup*

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DeleteSubnetGroup
    &SubnetGroupName=mysubnetgroup
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20210801T220302Z
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20210801T220302Z
    &X-Amz-Expires=20210801T220302Z
    &X-Amz-Signature=<signature>
    &X-Amz-SignedHeaders=Host
```

此命令不生成任何输出。

有关更多信息，请参阅 MemoryDB API 主题 DeleteSubnetGroup。

MemoryDB API 和接口 VPC 端点（AWS PrivateLink）

您可以通过创建接口 VPC 端点，从而在 VPC 和 Amazon MemoryDB API 端点之间建立私有连接。接口端点由提供支持AWS PrivateLink。 AWS PrivateLink 允许您在没有互联网网关、NAT 设备、VPN 连接或 Direct AWS Connect 连接的情况下私密访问 MemoryDB API 操作。

VPC 中的实例不需要公有 IP 地址便可与 MemoryDB 端点进行通信。您的实例也不需要公有 IP 地址即可使用任何可用的 MemoryDB API 操作。您的 VPC 和 MemoryDB 之间的流量不会脱离 Amazon 网络。每个接口终端节点均由子网中的一个或多个弹性网络接口表示。有关弹性网络接口的更多信息，请参阅《Amazon EC2 用户指南》中的弹性网络接口。

- 有关 VPC 终端节点的更多信息，请参阅 Amazon VPC 用户指南中的接口 VPC 终端节点 (AWS PrivateLink)。

- 有关 MemoryDB API 操作的更多信息，请参阅 MemoryDB API 操作。

创建接口 VPC 终端节点后，如果您为该终端节点启用私有 DNS 主机名，则为默认 MemoryDB 终端节点 (https://memorydb。 *Region*.amazonaws.com) 解析到您的 VPC 终端节点。如果您尚未启用私有 DNS 主机名，则 Amazon VPC 将提供一个您可以使用的 DNS 端点名称，格式如下：

```
VPC_Endpoint_ID.memorydb.Region.vpce.amazonaws.com
```

有关更多信息，请参阅 Amazon VPC 用户指南中的接口 VPC 端点 (AWS PrivateLink)。MemoryDB 支持调用您的 VPC 中的所有 API 操作。

> **Note**
>
> 只能为 VPC 中的一个 VPC 端点启用私有 DNS 主机名。如果要创建额外的 VPC 端点，则应为其禁用私有 DNS 主机名。

VPC 端点注意事项

在为 MemoryDB API 端点设置接口 VPC 端点之前，请务必查看《Amazon VPC 用户指南》中的接口端点属性和限制。可以从使用 AWS PrivateLink的 VPC 中获取所有与管理 MemoryDB 资源相关的 MemoryDB API 操作。MemoryDB API 端点支持 VPC 端点策略。默认情况下，允许通过端点对 MemoryDB API 操作进行完全访问。有关更多信息，请参阅《Amazon VPC User Guide》中的 Controlling access to services with VPC endpoints。

为 MemoryDB API 创建接口 VPC 端点

您可以使用 Amazon VPC 控制台或 AWS CLI为 MemoryDB API 创建 VPC 端点。有关更多信息，请参阅《Amazon VPC User Guide》中的 Creating an interface endpoint。

在创建接口 VPC 端点后，您可以为端点启用私有 DNS 主机名。当你这样做时，默认的 MemoryDB 端点 (https://memorydb。 *Region*.amazonaws.com) 解析到您的 VPC 终端节点。有关更多信息，请参阅《Amazon VPC 用户指南》中的通过接口端点访问服务。

为 Amazon MemoryDB API 创建 VPC 端点策略

您可以为 VPC 端点附加控制对 MemoryDB API 的访问的端点策略。此策略指定以下内容：

- 可执行操作的主体。

- 可执行的操作。
- 可对其执行操作的资源。

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用 VPC 端点控制对服务的访问。

Example MemoryDB API 操作的 VPC 端点策略

下面是用于 MemoryDB API 的端点策略示例。当附加到端点时，此策略会向所有委托人授予对列出的针对所有资源的 MemoryDB API 操作的访问权限。

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "memorydb:CreateCluster",
   "memorydb:UpdateCluster",
   "memorydb:CreateSnapshot"
  ],
  "Resource": "*"
 }]
}
```

Example拒绝来自指定 AWS 账户的所有访问的 VPC 终端节点策略

以下 VPC 终端节点策略拒绝 AWS 账户使用该终端节点访问*123456789012*所有资源。此策略允许来自其他账户的所有操作。

```
{
 "Statement": [{
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*",
   "Principal": "*"
  },
  {
   "Action": "*",
   "Effect": "Deny",
   "Resource": "*",
   "Principal": {
    "AWS": [
```

```
      "123456789012"
    ]
   }
  }
 ]
}
```

常见漏洞与暴露（CVE）：MemoryDB 中已解决的安全漏洞

常见漏洞和风险（CVE）是公开已知网络安全漏洞的条目列表。每个条目都是一个链接，其中包含一个标识号、一段描述和至少一个公共参考。您可以在本页找到 MemoryDB 中已解决的安全漏洞列表。

我们建议您始终升级到最新的 MemoryDB 版本，以防范已知漏洞。MemoryDB 公开了 PATCH 组件。PATCH 版本用于向后兼容的错误修复、安全修复和非功能性更改。

您可以使用下表来验证特定版本的 MemoryDB 是否包含对特定安全漏洞的修复。如果您的 MemoryDB 缓存待服务更新，它可能容易受到以下列出的安全漏洞之一的影响。我们建议您应用此服务更新。有关受支持的 MemoryDB 引擎版本及升级方式的更多信息，请参阅 引擎版本。

> **Note**
>
> - 如果某个 CVE 在某个 MemoryDB 版本中得到解决，则意味着在更新的版本中也已解决。
> - 下表中的星号 (*) 表示您必须已为运行指定版本的 MemoryDB 集群应用了最新的服务更新，才能解决该安全漏洞。有关如何验证您已为集群运行的 MemoryDB 版本应用了最新服务更新的更多信息，请参阅 管理服务更新。

MemoryDB 版本	CVEs 已解决
Valkey 7.3 及所有旧版 Valkey Redis OSS 7.1 及所有旧版 Redis OSS	CVE-2025-49844*、CVE-2025-46817*、CVE-2025-46818*、CVE-2025-46819*
Valkey 7.2 和 7.3	CVE-2025-21607*、CVE-2025-21605*、CVE-2024-31449*、CVE-2024-31227*、CVE-2024-31228*
Valkey 7.2.7	CVE-2024-51741

MemoryDB 版本	CVEs 已解决
Redis OSS 7.1 和 6.2	CVE-2025-21605*、CVE-2024-31449*、CVE-2024-31227*、CVE-2024-31228*、CVE-2023-41056
Redis OSS 7.0.7	CVE-2023-41056*
Redis OSS 6.2.7	CVE-2024-46981
Redis OSS 6.2.6	CVE-2022-24834*、CVE-2022-35977*、CVE-2022-36021*、CVE-2023-22458、CVE-2023-25155、CVE-2023-28856 CVE-2023-45145：请注意，此 CVE 已在 Redis OSS 6.2 和 7.0 中解决，但未在 Redis OSS 7.1 中解决。
Redis OSS 6.0.5	CVE-2022-24735*、CVE-2022-24736*

MemoryDB 中的服务更新

MemoryDB 会自动监控您的集群和节点实例集，以便在有服务更新可用时应用服务更新。通常，您可以设置预定义的维护时段，以便 MemoryDB 可以应用这些更新。然而，在某些情况下，您可能会发现这种方法过于僵化，并且可能限制您的业务流程。

使用 MemoryDB 中的服务更新，您可以控制更新的应用时间和内容。您还可以实时监控对所选 MemoryDB 集群执行这些更新的进度。

管理服务更新

MemoryDB 服务更新将定期发布。如果您有一个或多个符合这些服务更新的合格集群，则更新发布后，您将通过电子邮件、SNS、Personal Health Dashboard (PHD) 和亚马逊 CloudWatch 事件收到通知。更新也会显示在 MemoryDB 控制台的服务更新页面上。利用此控制面板，您可以查看 MemoryDB 实例集的所有服务更新及其状态。

您可以控制在自动更新开始前应用更新的时间。我们强烈建议您尽快应用任何安全更新类型的更新，以确保您的 MemoryDB 始终安装最新的安全补 up-to-date丁。

以下各节详细探索了这些选项。

主题

- Amazon MemoryDB 托管维护和服务更新概述

Amazon MemoryDB 托管维护和服务更新概述

我们会频繁升级 MemoryDB 实例集，无缝地将补丁和升级应用到实例上。我们通过以下两种方式之一进行：

1. 持续托管维护。
2. 服务更新。

这些维护和服务更新是应用那些增强安全性、可靠性和操作性能的升级所必需的。

持续托管维护会不定期在您的维护时段内直接进行，无需您采取任何行动。需要注意的是，维护时段对所有客户都是强制性的，您无法选择退出。我们强烈建议在这些已建立的维护时段内避免任何关键或重要的活动。此外，请注意，关键更新不能跳过，以确保系统的安全性和最佳性能。

服务更新使您能够灵活地自行应用它们。它们是有时间限制的，并且可能在截止日期过后被移到维护时段中由我们应用。

您可以通过在方便时尽早应用更新或通过替换节点来管理更新，因为在替换时会自动应用更新。如果在即将到来的维护时段之前更新已应用到所有节点，则在维护时段内将没有更新活动。

服务更新

MemoryDB 中的服务更新 使您能够自行决定应用某些服务更新。这些更新可以是以下类型：安全补丁或次要软件更新。这些更新有助于增强集群的安全性、可靠性和操作性能。

这些服务更新的价值在于您可以控制何时应用更新（例如，当有需要 MemoryDB 集群 24x7 可用性的重要业务事件时，您可以延迟应用服务更新）。

如果您有一个或多个符合这些服务更新的合格集群，则更新发布后，您将通过电子邮件、Amazon SNS、AWS Health 控制面板和 A mazon Events CloudWatch 事件收到通知。更新也会显示在 MemoryDB 控制台的服务更新页面上。利用此控制面板，您可以查看 MemoryDB 实例集的所有服务更新及其状态。

您可以控制在自动更新开始前应用更新的时间。我们强烈建议您尽快应用任何安全更新类型的更新，以确保您的 MemoryDB 始终安装最新的安全补 up-to-date丁。

您的集群可能是不同服务更新的一部分。大多数更新不需要您单独应用它们。对您的集群应用一个更新将在适用的情况下将其他更新标记为已完成。如果状态没有自动更改为“已完成”，您可能需要分别对同一集群应用多个更新。

服务更新的影响和停机时间

当您或 Amazon MemoryDB 对一个或多个 MemoryDB 集群应用服务更新时，更新在每个分片内一次仅应用于一个节点，直到所有选定集群都更新完毕。正在更新的节点将经历几秒钟的停机时间，而集群的其余部分将继续处理流量。

- 集群配置不会发生任何变化。
- 你会看到你的 CloudWatch 指标会有延迟，会尽快赶上。

节点替换对我的应用程序有何影响？ - 对于 MemoryDB 节点，替换过程旨在保证持久性和可用性。对于单节点 MemoryDB 集群，MemoryDB 会动态启动一个副本，从我们的持久性组件恢复数据，然后失效转移到该副本。对于由多个节点组成的复制组，MemoryDB 会替换现有副本，并将数据从我们的持久性组件同步到新副本。仅当节点数多于 1 个时，MemoryDB 才具有多可用区特性。因此在此场景下，替换主节点会触发到读取副本的失效转移。计划内的节点替换在集群处理传入写入请求的同时完成。如果只有一个节点，MemoryDB 会替换主节点，然后从我们的持久性组件同步数据。在此期间，主节点不可用，会导致写入中断时间较长。

为了确保顺利的替换体验并最大程度减少数据丢失，我应遵循哪些最佳实践？ - 在 MemoryDB 中，数据具有高持久性，即使在单节点部署中预计也不会发生数据丢失。但仍建议实施多可用区和备份策略，以在极不可能发生的故障事件中最大限度地减少丢失的可能性。为确保顺利的替换体验，我们会尝试一次仅替换同一集群中足够数量的节点，以保持集群稳定。您可以通过启用多可用区，在不同的可用区中配置主节点和读取副本。在这种情况下，当某个节点被替换时，主节点角色将失效转移到分片中的某个副本。该分片随后将开始处理流量，并且数据将从其持久性组件中恢复。如果您的配置每个分片仅包含一个主节点和一个副本，我们建议在修补前添加额外的副本。这将防止在修补过程中可用性降低。我们建议在传入写入流量较低的时段安排替换。

我应该遵循哪些客户端配置最佳实践以最小化维护期间的应用程序中断？ - 在 MemoryDB 中，集群模式配置始终启用，这在托管或非托管操作期间提供了最佳可用性。副本节点的各个节点端点可用于所有读取操作。在 MemoryDB 中，集群中始终启用自动失效转移，这意味着主节点可能会更改。因此，应用程序应确认节点的角色并更新所有读取端点，以确保不会给主节点带来重大负载。同样，在维护时段内，避免使副本因读取请求而过载。实现此目的的一种方法是确保您至少有两个读取副本，以避免维护期间发生任何读取中断。

测试客户端应用程序以确认它们符合 Redis/Valkey 集群协议非常重要，并且请求可以正确地跨节点重定向。建议实施退避和重试策略，以避免在维护和替换活动期间使 MemoryDB 节点过载。

重新安排 - 您可以通过更改维护时段来推迟服务更新。仅当计划日期与集群的维护时段匹配时，计划更新才会应用到集群。一旦您更改了维护时段并且计划日期已过，服务更新将被重新安排到接下来几周内新指定的时段。您将在新日期到达前一周收到新通知。

安全 AWS 是一项共同的责任。我们强烈建议您尽早应用更新。

选择退出服务更新 - 您可以通过验证“自动更新开始日期”属性的值来确定是否可以退出服务更新。如果服务更新的“自动更新开始日期”属性值已设置，MemoryDB 将在即将到来的维护时段为任何剩余的集群安排服务更新，并且无法选择退出。尽管如此，如果您在维护时段之前将服务更新应用到剩余集群，MemoryDB 将不会在维护时段期间重新应用该服务更新。有关更多信息，请参阅 应用服务更新。

为什么服务更新不能由 MemoryDB 在维护时段直接应用？ - 请注意，服务更新的目的是让您灵活掌握应用时间。未参与 MemoryDB 支持的合规性计划的集群可以选择不应用这些更新，或在全年以较低的频率应用它们。但建议应用更新以保持符合法规要求。这仅在未设置服务更新的“自动更新开始日期”属性值时才成立。有关更多信息，请参阅 适用于 MemoryDB 的合规性验证。

在维护时段应用的更新与服务更新有何不同？ - 通过持续托管维护应用的更新会直接在您的维护时段中安排，无需您采取任何行动。服务更新是有时间限制的，并让您可以通过“自动更新开始日期”来控制何时应用。如果届时仍未应用，MemoryDB 可能会在您的维护时段安排这些更新。

持续托管维护更新

这些更新是强制性的，会直接在您的维护时段中应用，无需您采取任何行动。这些更新与服务更新提供的更新是分开的。

持续维护的影响和停机时间

节点替换需要多长时间？ - 替换通常会在 30 分钟内完成。在某些实例配置和流量模式下，替换可能需要更长时间。

节点替换对我的应用程序有何影响？ - 持续托管维护更新的应用方式与“服务更新”相同，都是通过节点替换进行的。详情请参阅上文的服务更新影响和停机时间部分。

我如何自行管理节点替换？ - 您还可以选择在计划节点替换时段之前的任意时间自己管理这些替换。如果您选择自行管理替换，可以根据您的使用案例采取各种操作。

- 替换具有一个或多个分片的集群中的节点：您可以使用备份和恢复，或者先扩展再缩减以替换节点。
- 更改您的维护时段：此外，您可以更改集群的维护时段。要将维护时段更改为以后更方便的时间，您可以使用 UpdateCluster API、更新集群 CLI 或在 MemoryD B 管理控制台中单击 “修改”。一旦您更改了维护时段，MemoryDB 将在新指定的时段内为您的节点安排维护。

为了解其实际工作方式，假设当前是星期四 11/09 的 1500，下一个维护时段是星期五 11/10 的 1700。以下是 3 种场景：

- 您将维护时段更改为星期五 1600（在当前日期时间之后，且在下一个计划维护时段之前）。节点将于 11 月 10 日星期五 16 点替换。
- 您将维护时段更改为星期六 1600（在当前日期时间之后，且在下一个计划维护时段之后）。节点将于 11 月 11 日星期六 16 点替换。
- 您将维护时段更改为星期三 1600（在本周内早于当前日期时间）。节点将于 11 月 15 日星期三 16 点替换。

有关更多信息，请参阅 管理维护。

请注意，如果您为这些集群配置的维护时段相同，那么来自不同区域的不同集群中的节点可以同时被替换。

我如何了解即将到来的计划替换？ -您应该在健康控制面板上收到 AWS 健康通知。您还可以使用 DescribeServiceUpdates API 查看不同服务升级的状态。请注意，我们会尽一切努力主动通知客户可预见的替换。但是，在发生不可预测故障等特殊情况下，可能会出现未事先通知的替换。

我能否将计划维护更改为更合适的时间？ - 是的，您可以通过更改维护时段将计划维护推迟到更合适的时间。

为什么要执行这些节点替换？ - 这些替换是向您的基础主机应用强制性软件更新所必需的。这些更新有助于增强我们的安全性、可靠性和操作性能。

这些替换是否会同时影响我在多个可用区中的节点以及来自不同区域的集群？ - 替换可以在多个可用区或区域中并行运行，具体取决于集群的维护时段。

应用服务更新

您可以从服务更新具有 available（可用）状态起开始向实例集应用服务更新。服务更新为累积更新。换句话说，您尚未应用的任何更新都包含在您的最新更新中。

如果服务更新启用了自动更新，则可以选择在其可用时不执行任何操作。MemoryDB 将安排在自动更新开始日期之后的集群维护时段内应用更新。您将收到更新每个阶段的相关通知。

> **Note**
>
> 您只能应用那些具有 available（可用）或 scheduled（已安排）状态的服务更新。

有关查看并向适用的 MemoryDB 集群应用任何特定于服务的更新的更多信息，请参阅 使用控制台应用服务更新。

当您的一个或多个 MemoryDB 集群有新的服务更新可用时，您可以使用 MemoryDB 控制台、API 或 AWS CLI 来应用更新。以下各节说明了可用于应用更新的选项。

使用控制台应用服务更新

要查看可用服务更新的列表和其他信息，请转到控制台中的 Service Updates（服务更新）页面。

1. 登录 AWS 管理控制台 并打开 MemoryDB 控制台，网址为。https://console.aws.amazon.com/memorydb/
2. 在导航窗格中，选择 Service Updates（服务更新）。

在服务更新详细信息下，您可以查看以下内容：

- Service update name（服务更新名称）：服务更新的唯一名称
- 更新描述：提供有关服务更新的详细信息
- 自动更新开始日期：如果设置了此属性，则在此日期之后，MemoryDB 将安排在适当维护时段中自动更新集群。您将提前收到确切的计划维护时段的通知，这可能不是自动更新开始日期之后的即时通知。您仍然可以随时对您的集群应用更新。如果未设置该属性，则服务更新将不会启用自动更新，且 MemoryDB 亦不会自动更新集群。

在 Cluster update status（集群更新状态）部分中，您可以查看尚未应用服务更新或最近才应用服务更新的集群的列表。您可以查看每个集群的以下内容：

- Cluster name（集群名称）：集群的名称
- Nodes updated（已更新节点）：特定集群中已更新或仍对特定服务更新可用的各个节点的比率。
- Update Type（更新类型）：服务更新的类型，可以是 security-update 或 engine-update
- Status（状态）：集群服务更新的状态，为下列状态之一：
  - available（可用）：更新适用于必需的集群。
  - in-progres（进行中）：正在对此集群应用更新。
  - 已计划：已计划更新日期。
  - 完成：已成功应用更新。完成状态的集群将在完成后显示 7 天。

如果您选择任何或所有具有 available（可用）或 scheduled（已安排）状态的集群，然后选择 Apply now（立即应用），将开始对这些集群应用更新。

使用应用服务更新 AWS CLI

在收到服务更新可用的通知后，您可以使用 AWS CLI检测和应用这些更新：

- 要检索可用的服务更新的描述，请运行以下命令：

  ```
  aws memorydb describe-service-updates --status available
  ```

  有关更多信息，请参阅 describe-service-updates。
- 要对集群列表应用服务更新，请运行以下命令：

  ```
  aws memorydb batch-update-cluster --service-update
  ServiceUpdateNameToApply=sample-service-update --cluster-names cluster-1
  cluster2
  ```

  有关更多信息，请参阅 batch-update-cluster。

参考

本部分中的主题涵盖了有关使用 MemoryDB API 和 AWS CLI 的 MemoryDB 部分的信息。本部分还包含常见错误消息和服务通知。

- 使用 MemoryDB API
- MemoryDB API 参考
- AWS CLI 参考的 MemoryDB 部分

使用 MemoryDB API

本部分提供一些针对任务的说明，介绍如何使用和实施 MemoryDB 操作。有关这些操作的完整描述，请参阅 MemoryDB API 参考。

主题

- 使用查询 API
- 可用的库
- 对应用程序进行问题排查

使用查询 API

查询参数

HTTP 基于查询的请求是指使用 HTTP 动作 GET 或 POST 的 HTTP 请求，查询参数的名称为 `Action`。

每个查询请求必须包括一些通用参数，以处理操作的身份验证和选择事宜。

有些操作会使用参数列表。这些列表都是使用 `param.`*n* 表示法指定的。*n* 值是从 1 开始的整数。

查询请求身份验证

您只可以通过 HTTP 发送查询请求，并且每个查询请求中必须包含您的签名。本部分描述了如何创建签名。以下过程中说明的方法称为签名版本 4。

下面介绍了对发送至 AWS 的请求进行身份验证的基本步骤。其中假定您注册了 AWS，并且有一个访问密钥 ID 和秘密访问密钥。

查询身份验证流程

1. 发件人构建一个将要发送至 AWS 的请求。
2. 发件人计算请求签名，即带有一个 SHA-1 哈希函数的键控式哈希信息验证码（HMAC），如本主题下一部分中所定义的那样。
3. 该请求的发件人将请求数据、签名和访问密钥 ID（即所使用的秘密访问密钥的密钥标识符）发送至 AWS。
4. AWS 使用访问密钥 ID 来查询秘密访问密钥。
5. AWS 使用与计算请求中签名所用的相同算法根据请求数据和秘密访问密钥生成一个签名。

6. 如果签名匹配，那么请求将被视为可信。如果比较签名这一操作失败，那么请求将被丢弃，同时 AWS 将返回错误响应。

> **Note**
>
> 如果请求包含一个 `Timestamp` 参数，那么针对请求计算的签名将在被赋予值后的 15 分钟失效。
> 如果请求包含一个 `Expires` 参数，那么签名将在 `Expires` 参数指定的时间失效。

计算请求签名

1. 创建标准化的查询字符串，您在此过程的稍后部分需要用到它：

   a. 根据参数名称、按照自然字节排序对 UTF-8 查询字符串组成部分进行分类。参数可取自 GET URI 或 POST 正文（当内容类型为 application/x-www-form-urlencoded 时）。

   b. URL 根据以下规则对参数名称和值进行编码：

      i. 不对任何由 RFC 3986 定义的非预留字符进行 URL 编码。这些未预留字符是 A–Z、a–z、0–9、连字符（-）、下划线（_）、句点（.）和波形符（~）。

      ii. 使用 %XY 对所有其他参数进行百分比编码，其中“X”和“Y”分别代表十六进制字符 0-9 和大写字母 A-F。

      iii. 以 %XY%ZA... 格式对扩展的 UTF-8 字符进行百分号编码。

      iv. 将空白字符百分号编码为 %20（不是普通编码方案中的 +）。

   c. 使用等号（ = ）（ASCII 字符 61）将编码的参数名称与它们的编码值分隔开，即使参数值为空，亦应如此。

   d. 使用“和”符号（&）（ASCII 代码 38）隔开名称/值对。

2. 依照下列伪语法创建用以签名的字符串（“\n”代表 ASCII 换行）。

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

   HTTPRequestURI 组件是 URI 的 HTTP 绝对路径组件，但不包括查询字符串。如果 HTTPRequestURI 为空，则使用正斜杠（ / ）。

3. 利用您刚创建的字符串计算符合 RFC 2104 的 HMAC，将您的秘密访问密钥当作密钥，并将 SHA256 或 SHA1 作为哈希算法。

   有关更多信息，请参阅 https://www.ietf.org/rfc/rfc2104.txt。

4. 将结果值转换为 base64。
5. 将此值作为请求中的 `Signature` 参数值。

例如，下面是一个示例请求（为清晰起见，添加了换行符）。

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=DescribeClusters
    &ClusterName=myCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2021-01-01
```

对于前述的查询字符串，您将要计算下述字符串的 HMAC 签名。

```
GET\n
    memory-db.amazonaws.com\n
    Action=DescribeClusters
    &ClusterName=myCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE%2F20140523%2Fus-east-1%2Fmemorydb%2Faws4_request
    &X-Amz-Date=20210801T223649Z
    &X-Amz-SignedHeaders=content-type%3Bhost%3Buser-agent%3Bx-amz-content-sha256%3Bx-
amz-date
        content-type:
        host:memory-db.us-east-1.amazonaws.com
        user-agent:ServicesAPICommand_Client
    x-amz-content-sha256:
    x-amz-date:
```

结果是下面的已签名请求。

```
https://memory-db.us-east-1.amazonaws.com/
```

```
    ?Action=DescribeClusters
    &ClusterName=myCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2021-01-01
    &X-Amz-Algorithm=Amazon4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20141201/us-east-1/memorydb/aws4_request
    &X-Amz-Date=20210801T223649Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=2877960fced9040b41b4feaca835fd5cfeb9264f768e6a0236c9143f915ffa56
```

有关签名流程和计算请求签名的详细信息，请参阅主题签名版本 4 签名流程及其副主题。

可用的库

AWS 为喜欢使用特定于语言的 API（而不是查询 API）构建应用程序的软件开发人员提供了软件开发工具包（SDK）。这些开发工具包提供了一些基本功能（未包括在 API 中），如请求身份验证、请求重试和错误处理，以便您更轻松地开始工作。现已推出适用以下编程语言的开发工具包和其他资源：

- Java：
- Windows 和 .NET
- PHP
- Python *
- Ruby

有关其他语言的信息，请参阅示例代码和库。

对应用程序进行问题排查

MemoryDB 提供具体的描述性错误，以帮助您在与 MemoryDB API 互动时排查问题。

检索错误

通常，在您花费任何时间处理错误结果之前，您都会希望您的应用程序检查某个请求是否生成错误。查明是否出现错误的最简单方法是寻找 MemoryDB API 中做出响应的 `Error` 节点。

XPath 语法规则不仅提供了一种搜索 `Error` 节点存在情况的简单方法，而且提供了一种检索错误代码和信息的简单方法。下面的代码片段采用 Perl 和 XML::XPath 模块来确定在请求期间是否出现错误。如果出现了错误，那么代码会刊载第一个错误代码和响应信息。

```
use XML::XPath;
my $xp = XML::XPath->new(xml =>$response);
if ( $xp->find("//Error") )
{print "There was an error processing your request:\n", " Error code: ",
$xp->findvalue("//Error[1]/Code"), "\n", " ",
$xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

故障排除技巧

我们建议采用下列流程来诊断和解析 MemoryDB API 问题。

- 验证 MemoryDB 是否正确运行。

  如要执行此操作，只需打开一个浏览器窗口，然后提交一个查询请求至 MemoryDB 服务（例如 https://memory-db.us-east-1.amazonaws.com）。MissingAuthenticationTokenException 或 UnknownOperationException 可确认服务有效并对请求做出响应。
- 检查您的请求结构。

  每个 MemoryDB 操作都在 MemoryDB API Reference 中有一个参考页面。复查您正在使用的参数是否正确。为了给予您关于潜在错误内容的意见，请考虑示例请求或用户场景，以查看这些示例是否正在执行类似操作。
- 检查论坛。

  MemoryDB 有一个论坛，您可以在其中搜索他人在开发过程中遇到的问题的解决方案。如要查看论坛，请参阅

  https://forums.aws.amazon.com/。

MemoryDB 的配额

您的 AWS 账户为每项 AWS 服务设定了默认限额（以前称为限制）。除非另有说明，否则，每个配额是区域特定的。您可以请求增加某些配额，但其他一些配额无法增加。

要请求提高配额，请参阅《Service Quotas 用户指南》中的请求提高配额。如果限额在服务限额中尚不可用，请使用提高限制表格。

您的 AWS 账户具有以下与 MemoryDB 相关的配额。

名称	默认值	描述	指标名称
每个区域的节点数	300	一个区域中所有 MemoryDB 集群的最大节点数。此配额适用于给定区域内的预留节点和非预留节点。您可在同一区域中拥有最多 300 个预留节点和 300 个非预留节点。	NodesPerRegion
每个集群的节点数（Redis OSS 集群模式已启用）	90	MemoryDB 中单个 Redis OSS 集群的最大节点数。	NodesPerCluster
每个区域的参数组数	300	您可以在区域中创建的最大参数组数量。	ParameterGroup
每个区域的子网组数	300	您可以在区域中创建的最大子网组数量。	SubnetGroup
每个子网组的子网数	20	您可以为子网组定义的最大子网数量。	SubnetsPerSubnetGroup
每个区域的用户数	2000	您可以在一个区域中创建的最大用户数。	用户

名称	默认值	描述	指标名称
每个区域的用户组数	200	您可以在一个区域中创建的最大用户组数。	UserGroup
每个用户组的用户数	100	您可以为用户组定义的最大用户数。	UsersPerUserGroup

MemoryDB 用户指南的文档历史记录

下表介绍了 MemoryDB 的文档版本。

变更	说明	日期
MemoryDB 多区域已启动。	MemoryDB 多区域已启动。	2024 年 12 月 1 日
MemoryDB 多区域的 IAM 和安全策略更新。	IAM 和安全策略已更新。有关更多信息，请参阅使用服务关联角色和使用服务关联角色。	2024 年 12 月 1 日
MemoryDB 现在支持 Valkey。	MemoryDB 现在支持 Valkey。	2024 年 10 月 8 日
MemoryDB 现在支持使用 IAM 对用户进行身份验证	IAM 身份验证让您可以使用 AWS Identity and Access Management 身份对与 MemoryDB 的连接进行身份验证。这使您可以增强安全模型并简化许多管理安全任务。有关更多信息，请参阅使用 IAM 进行身份验证。	2023 年 5 月 10 日
MemoryDB 现在支持 Redis OSS 7	此版本为 MemoryDB 提供了多项新功能：Redis OSS 功能、ACL 改进和分片发布/订阅和增强型 I/O 多路复用。有关更多信息，请参阅 Redis OSS 引擎版本。	2023 年 5 月 9 日
MemoryDB 现在提供预留节点	相比按需节点定价，预留节点可以提供大幅折扣。预留节点不是物理节点，而是对账户中使用的按需型节点所应用的账单折扣。有关更多信息，请参阅 MemoryDB 预留节点。	2022 年 12 月 27 日

MemoryDB 现在支持数据分层	MemoryDB 数据分层。借助数据分层功能，您将能够以更低的成本将集群容量最高扩展到数百 TB。有关更多信息，请参阅数据分层。	2022 年 11 月 3 日
MemoryDB 现在支持原生 JavaScript 对象表示法（JSON）格式	原生 JavaScript 对象表示法（JSON）格式是在 Redis OSS 集群中对复杂数据集进行编码的一种简单的无模式方法。您可以使用 JavaScript 对象表示法（JSON）格式在 Redis OSS 集群中进行数据的本地存储和访问，并更新在这些集群中存储的 JSON 数据，而无需管理自定义代码来对其进行序列化和反序列化。有关更多信息，请参阅 JSON 入门。	2022 年 5 月 25 日
MemoryDB 现在支持 AWS PrivateLink	AWS PrivateLink 使您可以私密访问 MemoryDB API 操作，而无需互联网网关、NAT 设备、VPN 连接或 AWS Direct Connect 连接。有关更多信息，请参阅 MemoryDB API 和接口 VPC 端点（AWS PrivateLink）。	2022 年 1 月 24 日
初始版本。	首次发布 MemoryDB 用户指南。有关更多信息，请参阅什么是 MemoryDB？	2021 年 8 月 19 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。