Amazon MemoryDB for Redis

开发人员指南

亚马逊云科技

Amazon MemoryDB for Redis: 开发人员指南

Amazon Web Services 文档中描述的 Amazon Web Services 服务或功能可能因区域而异。要查看适用于中国区域的差异，请参阅中国的 Amazon Web Services 服务入门。

Table of Contents

什么是内存 DB 对 Redis？

MemoryDB of Redis 是一种持久的内存中数据库服务，可提供超快的性能。它专为具有微服务架构的现代应用程序而构建。

MemoryDB 与 Redis 兼容，这是一个受欢迎的开源数据存储，使您能够使用他们目前已经使用的同样灵活友好的 Redis 数据结构、API 和命令快速构建应用程序。使用 Memory DB，您的所有数据都存储在内存中，这使您能够实现微秒读取和单位数毫秒的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，以实现快速故障切换、数据库恢复和节点重启。

Memory DB 既具有内存中的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，从而无需分别管理缓存和持久数据库。

主题

内存 DB 的功能

MemoryDB of Redis 是一种持久的内存中数据库服务，可提供超快的性能。MemoryDB 的功能包括：

- 主节点的一致性强，并保证副本节点的最终一致性。有关更多信息，请参阅 一致性 (p. 95)。
- 微秒读取和一位数毫秒写入延迟，每个集群高达 1.6 亿 TPS。
- 灵活友好的 Redis 数据结构和 API。几乎无需修改即可轻松构建新应用程序或迁移现有 Redis 应用程序。
- 使用多可用区事务日志实现数据持久性，提供快速数据库恢复和重启。
- 具有自动故障切换功能的多可用区可用性，检测节点故障并从中恢复。
- 通过添加和删除节点轻松水平扩展，或者通过移动到较大或较小的节点类型进行垂直 您可以通过添加分片来扩展写吞吐量，并通过添加副本来扩展读取吞吐量。
- Read-after-write主节点的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、静态加密和通过使用访问控制列表 (ACL) 验证用户 (p. 172).
- Amazon S3 中的自动快照，保留时间长达 35 天。
- Support 每个群集多达 500 个节点和超过 100 TB 的存储空间（每个分片有 1 个副本）。
- 使用 TLS 进行传输中加密和静态加密Amazon KMS钥匙。
- Redis 的用户身份验证和授权使用访问控制列表 (ACL) 验证用户 (p. 172).
- 对该项的支持AmazonGraviton2 实例类型。
- 与其他公司集成Amazon服务，例如CloudWatch，Amazon VPC，CloudTrail，以及用于监控、安全和通知的 Amazon SNS。
- 完全托管的软件修补和升级。
- Amazon针对管理 API 的 Identity in Access Management (IAM) 集成和基于标签的访问控制。

MemoryDB 核心组件

下面，您可以找到 Memory 部署的主要组件概述。

主题

集群

集群是为单个数据集提供服务的一个或多个节点的集合。将 MemoryDB 数据集分为分片，每个分片有一个主节点和最多 5 个可选副本节点。主节点提供读取和写入请求，而副本只提供读取请求。主节点可以故障切换到副本节点，从而将该副本提升到该分片的新主节点。MemoryDB 将 Redis 作为其数据库引擎运行，创建集群时，您可以为集群指定 Redis 版本。您可以使用Amazon CLI、MemoryDB API 或Amazon Web Services Management Console.

每个 MemoryDB 集群都运行 Redis 引擎版本。每个 Redis 引擎版本都有其自身支持的功能。此外，每个 Redis 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其节点类型决定。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 (p. 22)。

> 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 Memory 结合使用的更多信息，请参阅MoryDB 和Amazon VPC (p. 221).

许多 MemoryDB 操作面向集群：

- 创建集群
- 修改集群
- 为集群拍摄快照
- 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 (p. 24) 和 管理节点 (p. 21)

  有关集群、节点和相关操作的信息。
- 用于 Redis 的 MemoryDB 中的弹性 (p. 94)

有关增强集群的容错能力的信息。

节点

一个节点是 Memory 部署的最小构建块，并使用 Amazon EC2 实例运行。每个节点都运行创建集群时选择的 Redis 版本。节点属于属于群集的分片。

每个节点都以您创建集群时选择的版本运行引擎实例。如果需要，您可以将集群中的节点纵向扩展或缩减到不同的类型。有关更多信息，请参阅扩展 (p. 134)。

集群中的每个节点都具有相同的节点类型。支持多种类型的节点，每种类型的节点具有不同的内存量。有关受支持的节点类型的列表，请参阅 受支持的节点类型 (p. 22)。

有关节点的更多信息，请参阅管理节点 (p. 21)。

分片

分片是由 1 至 6 个节点组成的组合，其中一个用作主写入节点，另外 5 个节点用作只读副本。MemoryDB 集群始终至少有一个分片。

MemoryDB 集群最多可以有 500 个分片，您的数据分区在分片之间。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，请参阅了解内存 DB 复制 (p. 95)。

有关分片的更多信息，请参阅使用分区 (p. 42)。

参数组

参数组是为集群上管理 Redis 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，并且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，请参阅使用参数组配置引擎参数 (p. 148).

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，您可以指定子网组或使用提供的默认子网组。MemoryDB 使用该子网组选择一个子网和该子网内的 IP 地址，以便与您的节点关联。

有关 MemoryDB 子网组的更多详细信息，请参阅子网和子网组 (p. 228).

访问控制列表

访问控制列表是一个或多个用户的集合。访问字符串跟随 RedisACL 规则授权用户访问 Redis 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 (ACL) 验证用户 (p. 172).

用户

用户拥有用户名和密码，用于访问 MemoryDB 集群上的数据和发出命令。用户是访问控制列表 (ACL) 的成员，您可以使用该列表来确定该用户在 MemoryDB 集群上的权限。有关更多信息，请参阅 。使用访问控制列表 (ACL) 验证用户 (p. 172)

相关 服务

ElastiCache对于 Redis

在决定是否将 MemoryDB 用于 Redis 时还是ElastiCache对于 Redis，请考虑以下比较：

- MemoryDB of Redis 是一个持久的内存中数据库，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提供超快性能的持久数据库（微秒读取和单位数毫秒写入延迟），则应考虑使用 Memory DB。如果您想使用 Redis 数据结构和带有主持久数据库的 API 构建应用程序，则 Memory DB 也可能非常适合您的使用案例。最后，您应该考虑使用 MemoryDB 来简化应用程序体系结构并降低成本，方法是将数据库的使用替换为缓存以实现持久性和性能。
- ElastiCache对于 Redis 是一项通常用于缓存使用 Redis 的其他数据库和数据存储中的数据的服务。您应该考虑ElastiCache适用于 Redis，用于缓存您希望使用现有主数据库或数据存储加速数据访问的工作负载（微秒读取和写入性能）。您还应该考虑ElastiCache对于 Redis，适用于希望使用 Redis 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案例。

选择区域和可用区

Amazon 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。

Amazon 区域是指大型、分布范围广泛的单独地理位置。可用区是 Amazon 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 Amazon 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 MobilyDB 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终端节点。有关服务终端节点，请参阅支持的区域和终端节点 (p. 6)。

找到您的节点

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

通过将节点放置在不同的可用区中，MemoryDB 可以消除一个可用区中的故障（例如停电）导致可用性丧失的可能性。

- 创建内存数据库集群 (p. 13)
- 修改 MemoryDB 集群 (p. 30)

支持的区域和终端节点

MobilyDB for RDK 提供多个Amazon区域。这意味着，您可在满足您要求的位置启动 MobilyDB 集群。例如，您可以在最靠近您客户的 Amazon 区域或者满足某些法律要求的特定 Amazon 区域中启动。

默认情况下，Amazon开发工具包，Amazon CLI、MobilyDB API 和 MobilyDB 控制台引用了美国东部（弗吉尼亚北部）区域。随着 MobilyDB 不断向新区域扩展，这些区域的新端点同样可以在您的 HTTP 请求中使用，即Amazon开发工具包，Amazon CLI和控制台。

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)。在不同的可用区内启动节点，可以实现可能的最大容错。有关区域和可用区的更多信息，请参阅选择区域和可用区 (p. 4)在本主题的开头。

支持 MobilyDB 的区域

区域名称/区域	端点	协议	
美国东部（俄亥俄）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（加利福尼亚北部）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	
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（孟买）区域 ap-south-1	memory-db.ap-south-1.amazonaws.com	HTTPS	
亚太区域（东京） ap-northeast-1	memory-db.ap-northeast-1.amazonaws.com	HTTPS	
亚太区域（首尔） ap-northeast-2	memory-db.ap-northeast-2.amazonaws.com	HTTPS	
亚太区域（新加坡） ap-southeast-1	memory-db.ap-southeast-1.amazonaws.com	HTTPS	

区域名称/区域	端点	协议	
亚太区域（悉尼） ap-southeast-2	memory-db.ap-southeast-2.amazonaws.com	HTTPS	
欧洲（法兰克福）区域 eu-central-1	memory-db.eu-central-1.amazonaws.com	HTTPS	
欧洲（爱尔兰）区域 eu-west-1	memory-db.eu-west-1.amazonaws.com	HTTPS	
欧洲（伦敦）区域 eu-west-2	memory-db.eu-west-2.amazonaws.com	HTTPS	
欧洲（巴黎）区域 eu-west-3	memory-db.eu-west-3.amazonaws.com	HTTPS	
欧洲（斯德哥尔摩）区域 eu-north-1	memory-db.eu-north-1.amazonaws.com	HTTPS	
Europe (Milan) Region eu-south-1	memory-db.eu-south-1.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	
中国（宁夏）区域 cn-northwest-1	memory-db.cn-northwest-1.amazonaws.com.cn	HTTPS	

换一张桌子Amazon按地区划分的产品和服务，请参阅按地区划分的产品和服务.

有关区域内支持的可用区的表，请参阅子网和子网组 (p. 228).

访问内存 DB

每个 MemoryDB 群集终端节点都包含一个地址和一个端口。此群集终端节点支持 Redis Cluster 协议，允许客户端发现群集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将副本升级到位时，您可以连接到群集终端节点以使用 Redis Cluster 协议发现新的主节点。

您需要连接到群集终端节点才能使用cluster nodes要么cluster slots命令。找到密钥的正确节点后，您可以直接连接到该节点以进行读/写请求。Redis 客户端可以使用群集终端节点自动连接到正确的节点。

要对群集中的特定节点进行故障排除，您也可以使用特定于节点的终端节点，但这些节点对于正常使用并不必需。

要查找集群的终端节点，请参阅：

- 查找 MemoryDB 集群的终端节点 (AmazonCLI) (p. 40)
- 查找 MemoryDB 集群的终端节点 (MemoryDB API) (p. 42)

要连接到节点或群集，请参阅使用 redis-cli 连接到 MemoryDB 节点 (p. 18).

内存 DB 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AmazonIdentity and Access Management (IAM)。当你连接到Amazon使用 IAM 凭证，Amazon账户必须拥有 IAM 策略来授予执行操作所需的权限。有关更多信息，请参阅 。Redis MemoryDB 中的身份和权限管理 (p. 180)
- 要控制对群集的访问级别，您可以创建具有指定权限的用户并将其分配给访问控制列表 (ACL)。然后，ACL 将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 (ACL) 验证用户 (p. 172)。
- 必须基于 Amazon VPC 服务在 Virtual Private Cloud (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 集群的终端节点和端口的连接，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到 MemoryDB 集群的连接。有关 VPC 的更多信息，请参阅 MoryDB 和Amazon VPC (p. 221)。

有关配置安全性的信息，请参阅内存中的安全性 For Redis (p. 168)。

开始前的准备工作

如果未完成这些任务，以下主题描述开始使用 Redis 时必须采取的一次性操作。

主题

注册Amazon

如果您还没有 Amazon Web Services 账户，请完成以下步骤创建一个。

注册 Amazon Web Services 账户

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

   在注册时，您将接到一通电话，要求您使用电话键盘输入一个验证码。

创建 IAM 用户

自行创建管理员用户并将该用户添加到管理员组（控制台）

1. 选择 Root user（根用户）并输入您的 Amazon Web Services 账户 电子邮件地址，以账户拥有者身份登录 IAM 控制台。在下一页上，输入您的密码。

   Note

   强烈建议您遵守以下使用 **`Administrator`** IAM 用户的最佳实践，妥善保存根用户凭证。只在执行少数账户和服务管理任务时才作为根用户登录。
2. 在导航窗格中，选择 Users（用户），然后选择 Add users（添加用户）。
3. 对于 User name（用户名），输入 **`Administrator`**。
4. 选中 Amazon Web Services Management Console access (Amazon Web Services Management Console 管理控制台访问）旁边的复选框。然后选择自定义密码，并在文本框中输入新密码。
5. （可选）默认情况下，Amazon要求新用户在首次登录时创建新密码。您可以清除 User must create a new password at next sign-in（用户必须在下次登录时创建新密码）旁边的复选框以允许新用户在登录后重置其密码。
6. 选择 Next:。Permissions (下一步：权限)。
7. 在设置权限下，选择将用户添加到组。
8. 选择创建组。
9. 在 Create group（创建组）对话框中，对于 Group name（组名称），输入 **`Administrators`**。
10. 选择 Filter policies（筛选策略），然后选择 Amazon managed - job function（Amazon 托管 – 工作职能）以筛选表内容。
11. 在策略列表中，选中 AdministratorAccess 的复选框。然后选择 Create group（创建组）。

Note

您必须先激活 IAM 用户和角色对账单的访问权限，然后才能使用 `AdministratorAccess` 权限访问 Amazon Billing and Cost Management 控制台。为此，请按照"向账单控制台委派访问权限"教程第 1 步中的说明进行操作。

12. 返回到组列表中，选中您的新组所对应的复选框。如有必要，选择 Refresh（刷新）以在列表中查看该组。
13. 选择 Next:。标签。
14. (可选) 通过以键值对的形式附加标签来向用户添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 IAM 用户指南中的标记 IAM 实体。
15. 选择 Next:。审核查看要添加到新用户的组成员资格的列表。如果您已准备好继续，请选择 Create user（创建用户）。

您可使用这一相同的流程创建更多组和用户，并允许您的用户访问 Amazon Web Services 账户 资源。要了解有关使用策略限制用户对特定 Amazon 资源的权限的信息，请参阅访问管理和示例策略。

完成上述操作后，您可以找到有关设置 MemoryDB 特定权限和访问权限的更多信息，请参阅管理您的 MemoryDB 资源的访问权限概览 (p. 182).

开始使用 MemoryDB

本练习将引导您完成使用 MemoryDB 管理控制台创建、授予访问权限、连接和最终删除 MemoryDB 集群的步骤。

主题

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。

主题

获取 Amazon 访问密钥

在您可以通过编程方式或通过Amazon命令行界面 (AmazonCLI)，必须拥有Amazon访问密钥。如果您计划仅使用 MemoryDB 控制台，则无需访问密钥。访问密钥包含访问密钥 ID 和秘密访问密钥，用于签署对发出的编程请求Amazon 如果没有访问密钥，您可以从 Amazon 管理控制台创建访问密钥。作为最佳实践，请勿在不必要时对任何任务使用Amazon账户根用户访问密钥。而是为自己创建一个具有访问密钥的新管理员 IAM 用户。仅当创建访问密钥时，您才能查看或下载秘密访问密钥。以后您无法恢复它们。不过，您随时可以创建新的访问密钥。您还必须拥有执行所需 IAM 操作的权限。有关更多信息，请参阅 IAM 用户指南中的访问 IAM 资源所需的权限。

为 IAM 用户创建访问密钥

1. 登录 Amazon 管理控制台，并通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Users (用户)。
3. 选择要为其创建访问密钥的用户的名称，然后选择 Security credentials (安全凭证) 选项卡。
4. 在 Access keys(访问密钥) 部分，选择 Create access key (创建访问密钥)。
5. 要查看新访问密钥对，请选择 Show (显示)。关闭此页面后，您将无法再次访问该私有访问密钥。您的凭证与下面类似：
   - 访问密钥 ID：AKIAIOSFODNN7EXAMPLE
   - 秘密访问密钥：wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
6. 要下载密钥对，请选择 Download .csv file (下载 .csv 文件)。将密钥存储在安全位置。关闭此页面后，您将无法再次访问该私有访问密钥。

7. 请对密钥保密以保护您的Amazon账户，切勿通过电子邮件发送密钥。请勿对企业外部共享密钥，即使有来自 Amazon 或 Amazon.com 的询问。合法代表 Amazon 的任何人永远都不会要求您提供密钥。
8. 下载 csv 格式文件之后，选择 Close（关闭）。在创建访问密钥时，预设情况下，密钥对处于活动状态，并且您可以立即使用此密钥对。

相关主题:

- IAM 用户指南中的什么是 IAM？
- Amazon 一般参考中的 Amazon 安全凭证。

配置您的凭证

在您可以通过编程方式或通过AmazonCLI，必须配置凭证以便为您的应用程序启用授权。

可通过多种方式执行此操作。例如，您可以手动创建凭证文件以存储您的访问密钥 ID 和秘密访问密钥。您还可以使用 Amazon CLI 的 `aws configure` 命令自动创建文件。或者，您也可以使用环境变量。有关配置您的凭证的更多信息，请参阅用于在 Amazon 上进行构建的工具中特定于编程的 Amazon 开发工具包开发人员指南。

下载和配置 Amazon CLI

http://aws.amazon.com/cli 提供 Amazon CLI。它在 Windows、MacOS 和 Linux 上运行。下载 Amazon CLI 后，可执行以下步骤安装和配置：

1. 转到 Amazon Command Line Interface 用户指南。
2. 按照安装 Amazon CLI 和配置 Amazon CLI 的说明操作。

设置权限（仅限新的 MemoryDB 用户）

ReReis MemyDB 会创建并使用服务相关角色以预配置资源并访问其他角色。Amazon代表您的资源和服务。要使用 MemoryDB 为您创建服务相关角色，请使用Amazon-名为的托管策略`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，请确保您具有调用的权限。`iam:createServiceLinkedRole`或者您已经创建了 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)
- Amazon适用于 Redis 的 MemoryDB 托管（预定义）策略 (p. 197)
- 对 Redis 使用 Amazon MemoryDB 的服务相关角色 (p. 189)

第 1 步：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 (p. 25) 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 使用率的更多信息，请参阅MemoryDB.

在 Virtual Private Cloud (VPC) 中基于 Amazon VPC 服务启动集群。

创建内存数据库集群

以下示例演示如何使用Amazon Web Services Management Console、Amazon CLI和 MemoryDB API。

创建集群 (控制台)

使用 MemoryDB 控制台创建集群的步骤

1. 登录到Amazon Web Services Management Console然后打开适用于 Redis 的 MemoryDB 控制台https://console.aws.amazon.com/memorydb/.
2. 选择集群在左侧导航窗格中，然后选择创建集群.
3. 完成集群信息部分。

   a. 在 Name (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 Description (描述) 框中，输入此集群的描述。
4. 完成子网组部分:

   - 适用于子网组，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。如果你正在创建一个新的：
     - 输入名称
     - 输入说明
     - 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 (p. 228)。
     - 如果您正在创建新的子网组，但没有现有 VPC，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？。
5. 完成集群设置部分:

   a. 适用于Redis 版本兼容性，请接受默认值`6.2`.

   b. 适用于端口，请接受默认 Redis 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. 适用于参数组，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 Redis 特定的参数 (p. 162)。

   d. 适用于节点类型，为所需节点类型选择一个值（及其关联的内存大小）。

e. 适用于分片数量，选择该集群所需的分区数量。为了提高集群的可用性，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区数量。有关更多信息，请参阅 扩展 MemoryDB 集群 (p. 135)。

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 Next（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。您可以为 VPC 使用默认安全组或创建新安全组。

      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，您有以下选项：

      - 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密钥以外的加密密钥，方法是选择客户托管Amazon拥有的 KMS 密钥然后选择密钥。

      - Encryption in-transit（传输中加密）– 对传输中的数据启用加密。如果选择不加密，则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 (ACL) 验证用户 (p. 172)。

   iii. 适用于快照，可以选择指定快照保留期和快照窗口。默认情况下，启用自动快照已预先选择。

   iv. 适用于维护时段可选择指定维护时段。这些区域有：维护时段是每周 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 为您的维护时段选择日期和时间 (无首选项），或者，您可以自行选择日期、时间和持续时间 [Specify指定维护时段)。如果您在列表中选择 Specify maintenance window，则为您的维护时段选择 Start day、Start time 和 Duration（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 (p. 91)。

   v. 对于 Notifications（通知），选择现有 Amazon Simple Notification Service (Amazon SNS) 主题，或选择“Manual ARN input（手动 ARN 输入）”，然后输入主题的 Amazon Resource Name (ARN)。Amazon SNS 允许将通知推送到与 Internet 连接的智能设备。默认设置是禁用通知。有关更多信息，请参阅 。https://aws.amazon.com/sns/.

   vi. 适用于标签，您可以有选择地应用标签来搜索和筛选您的集群，或跟踪Amazon成本.

i. 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 Create cluster (创建集群) 启动集群或选择 Cancel (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 。第 2 步：授予集群的访问权限 (p. 17)

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 第 4 步：删除集群 (p. 19)。

创建集群 (AmazonCLI)

使用 创建群集Amazon 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 \
    --subnet-group my-sg
```

对于 Windows：

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

你应该得到以下 JSON 响应：

```
{
    "Cluster": {
        "Name": "my-cluster",
        "Status": "creating",
        "NumberOfShards": 1,
        "AvailabilityMode": "MultiAZ",
        "ClusterEndpoint": {
            "Port": 6379
        },
        "NodeType": "db.r6g.large",
        "EngineVersion": "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:xxxxxxxxxxxxxx:cluster/my-cluster",
        "SnapshotRetentionLimit": 0,
        "MaintenanceWindow": "wed:03:00-wed:04:00",
        "SnapshotWindow": "04:30-05:30",
        "ACLName": "my-acl",
        "AutoMinorVersionUpgrade": true
    }
}
```

集群的状态变为后，您就可以开始使用集群了`available`.

> **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 第 4 步：删除集群 (p. 19)。

创建集群 (MemoryDB API)

要使用 MemoryDB API 创建集群，请使用CreateClusteraction.

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 第 4 步：删除集群 (p. 19)。

第 2 步：授予集群的访问权限

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。Amazon 弹性容器服务中运行的容器化或无服务器应用程序也可以访问它们，或者Amazon Lambda. 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 集群的默认端口是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. 为集群的安全组创建自定义规则，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 Amazon 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 Security Groups (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 集群的默认端口是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。

有关从其他 Amazon VPC 访问您的 MemoryDB 集群的信息，Amazon区域或企业网络，请参阅：

- 用于访问 Amazon VPC 中 MemoryDB 集群的访问模式 (p. 224)
- 从外部访问 MemoryDB 资源Amazon (p. 35)

第 3 步：连接到集群

在继续之前，请完成第 2 步：授予集群的访问权限 (p. 17)。

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。

查找您的集群终端节点

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，您必须先确定终端节点。

要进一步了解如何查找您的终端节点，请参阅：

- 查找 MemoryDB 集群的终端节点 (Amazon Web Services Management Console) (p. 40)
- 查找 MemoryDB 集群的终端节点 (AmazonCLI) (p. 40)
- 查找 MemoryDB 集群的终端节点 (MemoryDB API) (p. 42)

Connect 到 MemoryDB 集群 (Linux)

现在您有了所需的终端节点，便可以登录 EC2 实例并连接到集群。在以下示例中，您使用cli实用工具连接到使用 Ubuntu 22 的集群。最新版本的 cli 还支持 SSL/TLS 用于连接启用加密/身份验证的集群。

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。您也可以在 Amazon Linux 和 Amazon Linux 2 上使用具有 TLS/SL 的 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 软件发布版中。
2. 在 EC2 实例的命令提示符处，键入以下命令：

   Amazon Linux 2

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

   Amazon Linux

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

3. 然后，建议您运行可选的 `make test` 命令。
4. 在 EC2 实例的命令提示符处，键入以下命令，并使用您的集群和端口的终端节点替换此示例中显示的相应内容。

```
src/redis-cli -c -h Cluster Endpoint --tls -p 6379
```

第 4 步：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 (p. 108)。

使用 Amazon Web Services Management Console

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. 登录到Amazon Web Services Management Console然后打开适用于 Redis 的 MemoryDB 控制台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名称旁边的单选按钮。在这种情况下，是您在 第 1 步：创建 集群 (p. 13) 创建的集群的名称。
3. 对于 Actions (操作)，选择 Delete (删除)。
4. 在删除集群之前，先选择是否创建集群的快照，然后输入`delete`在确认框中和Delete以删除集群，或选择Cancel来保留集群。

   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 Amazon 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 第 1 步：创建 集群 (p. 13) 中创建的集群的名称。

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

这些区域有：`delete-cluster`CLI 操作仅删除一个集群。要删除多个集群，请调用`delete-cluster`针对要删除的每个集群。在删除一个集群之前，您无需等待删除另一个集群完成。

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

以下代码删除集群 my-cluster。在这种情况下，将 my-cluster 替换为您在 第 1 步：创建 集群 (p. 13) 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>
```

这些区域有：DeleteClusterAPI 操作仅删除一个集群。要删除多个集群，请调用DeleteCluster针对要删除的每个集群。在删除一个集群之前，您无需等待删除另一个集群完成。

有关更多信息，请参阅 。DeleteCluster.

接下来该做什么？

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：

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

管理节点

节点是 MemoryDB for Redis 部署中的最小构建数据块。节点属于属于集群的分片。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

涉及节点的一些重要操作如下：

MemoryDB 节点和分区

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。在分片中，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 支持一个集群内的多个分片。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列出了包含成员节点的分片、节点名称、端点以及其他信息。

在 MemoryDB 集群创建后，可以对其进行修改（扩展或缩减）。有关更多信息，请参阅 扩展 (p. 134) 和 替换节点 (p. 23)。

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、引擎版本或从亚马逊迁移，这样做会很有用 ElastiCache 对于 Redis。有关更多信息，请参阅 手动创建快照 (p. 111) 和 从快照还原 (p. 125)。

受支持的节点类型

MemoryDB 支持以下节点类型。

- 通用型

  最新一代：

vCPU	内存 (GiB)	网络性能
db.t4g.small	1.37	最高 5Gb
db.t4g.medium	3.09	最高 5Gb

- 内存优化

  最新一代：

vCPU	内存 (GiB)	网络性能
db.r6g.large	13.07	最高 10 Gb
db.r6g.xlarge	26.32	最高 10 Gb
db.r6g.2xlarge	52.82	最高 10 Gb
db.r6g.4xlarge	105.81	最高 10 Gb
db.r6g.8xlarge	209.55	12 Gb
db.r6g.12xlarge	317.77	20 Gb
db.r6g.16xlarge	419.10	25 Gb

适用于Amazon区域可用性，请参阅MemoryDB for Redis 定价

所有节点类型都是在虚拟私有云 (VPC) 中创建的。

替换节点

MemoryDB 频繁升级其机群，补丁和升级通常是无缝的。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**
>
> MemoryDB for Redis 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。

  如果节点是多可用区集群的成员，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，替换完成。
- 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 (p. 30)。

  > **Note**
  >
  > 仅当 MemoryDB 通知包括维护时段时，您才可以通过移动维护时段的方式更改替换时段。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 (p. 91)。

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。此标识符指定了在与 MemoryDB API 交互时的特殊集群，Amazon CLI命令。在 Amazon 区域中，集群标识符对于该用户必须是唯一的。

MemyDB 集群设计为通过 Amazon EC2 实例进行访问。您只能在 Virtual Private Cloud (VPC) 中基于 Amazon VPC 服务启动 MemoryDB 集群，但您可以从外部进行访问。Amazon. 有关更多信息，请参阅从外部访问 MemoryDB 资源Amazon (p. 35)。

准备集群

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

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

准备工作

了解以下问题的答案有助于使集群的创建更加流畅：

- 开始创建集群前，请务必在相同 VPC 中创建子网组。或者，您可以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 (p. 228)。

  MemoryDB 旨在从内部访问Amazon使用 Amazon EC2。但是，如果您在 VPC 中启动基于 Amazon VPC 的 VPC，则可以提供从外部进行访问的权限。Amazon. 有关更多信息，请参阅从外部访问 MemoryDB 资源Amazon (p. 35)。
- 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 (p. 150)。
- 您是否需要创建 VPC 安全组？

  有关更多信息，请参阅 。VPC 中的安全性.
- 您想如何实现容错？

  有关更多信息，请参阅缓解故障 (p. 94)。

主题

内存和处理器要求

Redis 的 MememoryDB 的基本构建基块是节点。配置分片节点以形成集群。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 集群包含 1 到 500 个分片。MemyDB 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终端节点的网络地址与 MemoryDB 集群连接。除了节点终端节点外，MemoryDB 集群本身还具有一个称为群集端节点. 您的应用程序可以使用此终端节点来读取或写入集群，从而最多可确定要从哪个节点中读取或写入的节点。

扩展要求

所有集群都可以扩展更大的节点类型。当您扩展 MemoryDB 集群时，可以在线执行此操作，以便集群保持可用，或者可以从快照中为新集群添加种子，并避免新集群开始为空。

有关更多信息，请参阅本指南中的 扩展 (p. 134)。

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用户帐户。因此，必须先授权对集群的入口授权，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 第 2 步：授予集群的访问权限 (p. 17)。

区域和可用区

通过将 MemoryDB 集群定位在Amazon您可以降低延迟，接近应用程序的区域。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 (p. 4)
- 缓解故障 (p. 94)

查看集群的详细信息

您可以使用 MemiryDB 控制台查看有关一个或多个集群的详细信息，Amazon CLI，或 MemoryDB API。

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. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB of Redis 控制台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以查看集群详细信息页面。

   这些区域有：簇详细信息页面显示有关集群的详细信息，包括集群终端节点。您可以使用中提供的多个选项卡查看更多详细信息簇详细信息页.
3. 选择分片和节点选项卡可查看集群分片的列表和每个分片中的节点数。
4. 要查看有关节点的特定信息，请展开下表中的分片。或者，您还可以使用搜索框搜索分片。

   执行此操作将显示有关每个节点的信息，包括其可用区、插槽/密钥空间和状态。
5. 选择指标选项卡来监控他们各自的进程，例如CPU 利用率和引擎 CPU 利用率. 有关更多信息，请参阅MemoryDB (p. 200)。
6. 选择网络和安全性选项卡上可查看子网组和安全组的详细信息。

   a. In子网组，您可以看到子网组的名称、子网所属 VPC 的链接以及子网组的 Amazon 资源名称 (ARN)。

   b. In安全组，你可以看到安全组 ID、名称和描述。
7. 选择维护和快照选项卡以查看快照设置的详细信息。

   a. In快照，您可以看到是否启用了自动快照、快照保留期和快照窗口。

   b. In快照，您将看到此集群的任何快照的列表，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 (p. 108)。
8. 选择维护和快照选项卡以查看维护时段的详细信息以及任何待处理的 ACL、重新分片或服务更新。有关更多信息，请参阅管理维护 (p. 91)。
9. 选择服务更新选项卡以查看适用于此群集的任何服务更新的详细信息。有关更多信息，请参阅用于 Redis 的 MemoryDB 中的服务更新 (p. 238)。
10. 选择标签选项卡以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，请参阅为快照添加标签 (p. 132)。

查看集群的详细信息（AmazonCLI)

您可以使用 Amazon 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 输出显示了响应：

```
{
    "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": "6.2.4",
            "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",
            "AutoMinorVersionUpgrade": true
        }
```

有关更多信息，请参阅 。Amazon CLI对于 MemoryDB 主题describe-clusters.

查看集群的详细信息（MemoryDB API）

您可以使用 MemoryDB API 查看集群的详细信息DescribeClustersaction. 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

除了在集群中添加或移除节点外，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，更改将立即应用于集群。无论是更改集群的参数组本身还是更改集群参数组中的参数值，都是如此。

您还可以更新群集的引擎版本。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。有关更多信息，请参阅升级引擎版本 (p. 47)。

使用 Amazon Web Services Management Console

修改集群

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB of Redis 控制台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 Amazon 区域。
3. 从左侧导航中，转到集群. 从群集细节中，使用单选按钮选择群集然后转到操作然后修改.
4. 这些区域有：修改此时将显示页。
5. 在修改窗口中，根据需要做出修改。选项包括：
   - 描述
   - 子网组
   - VPC 安全组
   - 节点类型
   - Redis 版本兼容性
   - 启用自动快照
   - 快照保留周期
   - 快照窗口
   - 维护时段
   - SNS 主题通知
6. 选择 Save changes（保存更改）。

您还可以在簇详细信息在页面上单击修改对集群进行修改。如果要修改群集的特定部分，可以转到簇详细信息在页面上单击修改.

使用 Amazon CLI

您可以使用 Amazon 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 。更新集群中的Amazon CLI命令参考命令。

使用 MemoryDB API

您可以使用 MemoryDB API 修改现有集群UpdateClusteroperation. 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>
```

从集群中添加/删除节点

您可以使用在集群中添加或删除节点Amazon Web Services Management Console，Amazon CLI，或MemoryDB API。

使用 Amazon Web Services Management Console

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB of Redis 控制台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上，选择添加/删除节点
4. In新节点数，输入希望的节点数量。
5. 选择 Confirm（确认）。

   Important

   如果将节点数设置为 1，则将不再启用多可用区。您还可以选择启用自动故障移.

使用 Amazon 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 (p. 27)。
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`要从其中移除节点的集群的 ID。
   - `--replica-configuration`— 允许您设置副本数量：
     - `ReplicaCount`— 设置此属性指定希望的副本节点数。
   - `--region` 指定要从其中删除节点的集群的 Amazon 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 Amazon CLI 主题 `update-cluster`。

使用 MemoryDB API

要使用 MemoryDB API 删除节点，请调用`UpdateCluster`包含集群名称和要删除节点列表的 API 操作，如下所示：

- `ClusterName`要从其中移除节点的集群的 ID。
- `ReplicaConfiguration`— 允许您设置副本数量：
  - `ReplicaCount`— 设置此属性指定希望的副本节点数。
- `Region` 指定要从其中删除节点的集群的 Amazon 区域。

有关更多信息，请参阅 。UpdateCluster.

访问您的集群

Redis MemoryDB 实例设计用于通过 Amazon EC2 实例进行访问。

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

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 Amazon Web Services Management Console，然后通过以下网址打开 Amazon EC2 控制台：https://console.aws.amazon.com/ec2/。
2. 在左侧导航窗格中，在网络和安全，选择个安全组.
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，否则此安全组将被命名为。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 Add rule。
   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。Redis 的默认端口是**6379**.
   e. 在源框中，选择Anywhere其端口范围为 (0.0.0/0)，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 公开集群时，不会在 Internet 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 Save (保存)。

当您在 Amazon VPC 中启动 Amazon EC2 实例时，该实例将能够连接到您的 MemoryDB 集群。

从外部访问 MemoryDB 资源Amazon

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AmazonClient 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 Amazon 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

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 Amazon Certificate Manager (ACM)：
  - 在 ACM 控制台上，选择 Certificate Manager (证书管理器)。
  - 选择 Import Certificate (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 Certificate body（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 Certificate chain (证书链) 字段中。
  - 选择 Review and import (查看并导入)。
  - 选择 Import (导入)。

  要使用 Amazon 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 Amazon 客户端 VPN 组件

使用 Amazon 控制台

在 Amazon 控制台上，选择 Services（服务），然后选择 VPC。

在 Virtual Private Network (虚拟专用网) 下，选择 Client VPN Endpoints (客户端 VPN 终端节点) 并执行以下操作：

配置 Amazon Client VPN 组件

- 选择 Create Client VPN Endpoint (创建客户端 VPN 终端节点)。
- 指定以下选项：
  - Client IPv4 CIDR (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。例如：10.0.0/22。
  - 在 Server certificate ARN (服务器证书 ARN) 中，选择之前导入的证书的 ARN。
  - 选择 Use mutual authentication (使用双向身份验证)。
  - 在 Client certificate ARN (客户端证书 ARN) 中，选择之前导入的证书的 ARN。
  - 选择 Create Client VPN Endpoint (创建客户端 VPN 终端节点)。

使用 Amazon 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:us-
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 Associations (关联) 选项卡。
- 选择 Associate (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，请在子网组在 MemoryDB 仪表板上。
  - 选择 Associate (关联)。如有必要，请为其余网络重复执行这些步骤。

使用 Amazon 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 终端节点设置中定义）到服务端口上的 MemyDB 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 Apply Security Group (应用安全组)。
- 选择新的安全组。

使用 Amazon 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 Authorization (授权) 选项卡上，选择 Authorize Ingress (授权入口) 并指定以下内容：

- 启用访问的目标网络：或者使用 0.0.0/0 以允许访问任何网络（包括 Internet），或限制 MemoryDB 网络/主机。
- 在 Grant access to: (授予访问权限：) 下，选择 Allow access to all users (允许访问所有用户)。
- 选择 Add Authorization Rules (添加授权规则)。

使用 Amazon 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 Route Table (路由表) 选项卡，然后单击 Create Route (创建路由)。

- 路线目的地：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 Create Route (创建路由)。

使用 Amazon 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 Amazon 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 Amazon 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 Amazon Client VPN：

- 在 Amazon Web Services Management Console上，选择 Services (服务)，然后选择 VPC。
- 选择 Client VPN Endpoints (客户端 VPN 终端节点)。
- 选择客户端 VPN 终端节点，然后选择 Actions (操作) -> Import Client Certificate CRL (导入客户端证书 CRL)。
- 查看 `crl.pem` 文件的内容。

使用 Amazon 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

您的应用程序使用终端节点连接到集群。终端节点是集群的唯一的地址。使用集群的集群终端节点适用于所有操作。

以下部分将引导您搜索所需的终端节点。

查找 MemoryDB 集群的终端节点 (Amazon Web Services Management Console)

查找 MemoryDB 集群的终端节点

1. 登录到Amazon Web Services Management Console然后打开适用于 Redis 的 MemoryDB 控制台https://console.aws.amazon.com/memorydb/.
2. 从导航窗格中，选择 Clusters (集群)。

   集群屏幕随即出现，其中显示集群的列表。选择要连接的集群。
3. 要查找集群的终端节点，请选择集群的名称（而不是单选按钮）。
4. 这些区域有：集群终端节点显示在集群详细信息. 要复制它，请选择复制图标位于端点左侧。

查找 MemoryDB 集群的终端节点 (AmazonCLI)

您可以使用`describe-clusters`命令来搜索集群的终端节点。命令返回集群的终端节点。

以下操作检索终端节点，在本示例中该终端节点表示为*##*，用于群集`mycluster`.

它返回以下 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 集群的终端节点 (MemoryDB API)

您可以使用适用于 Redis 的 MemoryDB API 来搜索集群的终端节点。

查找 MemoryDB 集群的终端节点 (MemoryDB API)

您可以使用 MemoryDB API 来搜索集群的终端节点`DescribeClusters`action. 操作返回集群的终端节点。

以下操作检索集群的集群终端节点`mycluster`.

```
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.

使用分区

分片是由 1 到 6 个节点组成的集合。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 5 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。

当您使用Amazon Web Services Management Console，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，请参阅创建内存数据库集群 (p. 13)。

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 的离线重新分片和分片重新平衡 (p. 135) 和 MemoryDB 的在线重新分片和分片重新平衡 (p. 136)。

寻找碎片的名字

您可以使用Amazon Web Services Management Console，Amazon CLI或者 MemoryDB API。

使用 Amazon Web Services Management Console

以下过程使用Amazon Web Services Management Console找到 MemoryDB 集群的分片名称。

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB and Redis 控制台https://console.aws.amazon.com/memorydb/.
2. 在左侧导航窗格中，选择集群.
3. 在下选择集群名称你想找到谁的碎片名字。
4. 在分片和节点选项卡中，查看下面的分片列表名称. 您还可以展开每个节点以查看其节点的详细信息。

使用 Amazon CLI

要查找 MemoryDB 集群的分片（分片）名称，请使用Amazon CLI手术`describe-clusters`使用以下可选参数。

- **--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**— 返回分片的详细信息，包括它们的名字。

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
```

其中返回以下 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": "6.2.4",
            "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",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

使用 MemoryDB API

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

- **`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`**— 返回分片的详细信息，包括它们的名字。

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>
```

管理你的 MemoryDB 实现

本节介绍了有关如何管理 MemoryDB 实施的各种组件的详细信息。

主题

引擎版本和升级

本部分介绍支持的 Redis 引擎版本以及如何升级。

主题

支持的 Redis 版本

MemoryDB for Redis 版本（加强版）

MemoryDB 推出新版本的 Redis 引擎，其中包括使用访问控制列表 (ACL) 验证用户 (p. 172)、自动版本升级支持、客户端缓存和重大操作改进。

此版本还推出了对原生的支持 JavaScript 对象表示法 (JSON) 格式，这是一种在 Redis 集群中编码复杂数据集的简单无模式方式。借助 JSON 支持，您可以将性能和 Redis API 用于通过 JSON 运行的应用程序。有关更多信息，请参阅 JSON 入门 (p. 48)。还包括与 JSON 相关的指标`JsonBasedCmds`这被纳入 CloudWatch 监控此数据类型的使用情况。有关更多信息，请参阅 MemoryDB (p. 200)。

借助Redis 6，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。这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 还将自动管理正在运行状态中的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 用于 Redis 的 MemoryDB 中的服务更新 (p. 238)。

如果您在创建时没有指定引擎版本，MemoryDB 将自动为您选择首选的 Redis 版本。另一方面，如果您使用指定引擎版本`6.2`，MemoryDB 会自动调用可用的 Redis 6.2 首选补丁版本。

例如，当您创建一个集群时，您将`--engine-version`Parameter`6.2`. 则在集群创建时，系统将会使用当前可用的 Patch 版本启动集群。任何包含完整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。

打电话时`DescribeEngineVersions`API，`EngineVersion`Parameter calue 会设置为 6.2，实际引擎版本会在`EnginePatchVersion`字段中返回的子位置类型。

有关 Redis 6.2 发布说明Redis 6.2 发布说明在 Redis GitHub.

升级引擎版本

默认情况下，MemoryDB 通过服务更新自动管理正在运行的集群的补丁版本。此外 auto 如果您将`AutoMinorVersionUpgrade`集群的属性设置为 false。但是，您不能选择退出auto 补丁版本升级。

在启动auto 升级之前，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 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。

您可以通过以下方式启动集群的引擎版本升级：

- 通过更新它并指定新的引擎版本。有关更多信息，请参阅 修改 MemoryDB 集群 (p. 30)。
- 正在为相应的引擎版本应用服务更新。有关更多信息，请参阅 用于 Redis 的 MemoryDB 中的服务更新 (p. 238)。

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。
- 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 Redis OSS 次要版本提供单一版本，而不提供多个补丁版本。这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 还将自动管理正在运行状态中的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 用于 Redis 的 MemoryDB 中的服务更新 (p. 238)。
- 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。
- 我们建议在传入的写流量较低期间执行引擎升级。

  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。

如何升级引擎版本

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

- 使用 Amazon Web Services Management Console (p. 30)
- 使用 Amazon CLI (p. 30)
- 使用 MemoryDB API (p. 31)

解决被阻止的 Redis 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 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

待处理的操作	阻止的操作
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

JSON 入门

MemoryDB 支持原生语法 JavaScript 对象表示法 (JSON) 格式，这是对 Redis 集群内复杂数据集群进行编码的简单方法。您可以使用本地存储和访问数据 JavaScript 在 Redis 集群中设置对象表示法 (JSON) 格式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即可对其进行序列化和反序列化。

除了将 Redis API 用于通过 JSON 运行的应用程序之外，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。您也可以使用搜索您的 JSON 文档内容Goessner style `JSONPath`查询。

使用受支持的引擎版本创建集群后，JSON 数据类型和关联命令将自动可用。这与 RedisJSON 模块的版本 2 兼容 API 且兼容 RDB，因此您可以轻松地将现有的基于 JSON 的 Redis 应用程序迁移到 MemoryDB 中。有关支持的 Redis 命令的更多信息，请参阅。支持的命令 (p. 55).

JSON 相关指标`JsonBasedCmds`被合并到 CloudWatch 监视此数据类型的使用情况。有关更多信息，请参阅 。内存 DB 的指标.

主题

Redis JSON 数据类型概述

MemoryDB 支持许多 Redis 命令来处理 JSON 数据类型。以下是 JSON 数据类型的概述以及受支持的 Redis 命令的详细列表。

术语

租期	描述
JSON 文档	指的是 Redis JSON 密钥的值
JSON 值	指的是 JSON 文档的子集，包括表示整个文档的根目录。值可以是容器或容器中的条目
JSON 元素	等同于 JSON 值的

支持的 JSON 标准

JSON 格式符合RFC 7159和ECMA-404JSON 数据交换标准。UTF-8统一码支持 JSON 中的文本。

Root 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 文档以经过优化的格式存储在内部，可以快速访问和修改。与相同文档的等效序列化表示相比，这种格式通常会消耗更多的内存。单个 JSON 文档消耗的内存限制为 64MB，这是内存中数据结构的大小，而不是 JSON 字符串的大小。JSON 文档使用的内存量可通过使用`JSON.DEBUG MEMORY`命令。

JSON ACL

- JSON 数据类型完全集成到中Redis 访问控制列表 (ACL)功能。与现有的每数据类型类别（@string、@hash 等）类似，添加了一个新类别 @json，以简化对 JSON 命令和数据的访问的管理。没有其他现有的 Redis 命令属于 @json 类别。所有 JSON 命令都强制执行任何密钥空间或命令限制和权限。
- 有五个现有的 Redis ACL 类别已更新为包含新的 JSON 命令：@read、@write、@fast、@slow 和 @admin。下表显示了 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.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.OBJLEN	y		y		
JSON.RESP	y		y		
JSON.SET		y		y	
JSON.STRAPPEND		y	y		
JSON.STRLEN	y		y		
JSON.STRLEN	y		y		

JSON 命令	@read	@write	@fast	@slow	@admin
JSON.TOGGLE		y	y		
JSON.TYPE	y		y		
JSON.NUMINCRBY		y	y		

嵌套深度限制

当 JSON 对象或数组的元素本身就是另一个 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。最大嵌套深度限制为 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，并显示错误。

命令语法

大多数命令要求将 Redis 键名作为第一个参数。有些命令还有 path 参数。如果 path 参数是可选的且未提供，则默认为根。

表示语法：

- 必需参数括在尖括号内，例如 <key>
- 可选参数括在方括号内，例如 [path]
- 其他可选参数由... 表示，例如 [json...]

路径语法

JSON-redis 支持两种路径语法：

- 增强语法— 遵循描述的 jsonPath 语法Goessner，如下表所示。为了清楚起见，我们对表中的描述进行了重新排序和修改。
- 受限语法— 具有有限的查询能力。

> **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，则使用增强的语法。否则，将使用受限语法。

增强语法

符号/表达式样式	描述
$	根元素
. 或 []	子运算符
..	递归式下降
*	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。
[,]	UNion 运算符
[start: end: step]	数组切片运算符

符号/表达式样式	描述
?()	将过滤器（脚本）表达式应用于当前数组或对象
()	筛选条件表达式
@	在过滤器表达式中使用，表示当前正在处理的节点
==	等于，在筛选器表达式中使用。
!=	不等于，在筛选器表达式中使用。
>	大于，在筛选器表达式中使用。
>=	大于或等于，在筛选表达式中使用。
<	小于，在筛选器表达式中使用。
<=	小于或等于，在筛选表达式中使用。
&&	逻辑 AND，用于组合多个筛选器表达式。
||	逻辑 OR，用于组合多个筛选器表达式。

示例

以下示例是基于Goessner示例 XML 数据，我们通过添加其他字段对其进行了修改。

```
{ "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	描述
$.store.book [*] .author	商店里所有书籍的作者
$。作者	所有作者
$.store。 *	商店的所有成员
$ ["存储"] . *	商店的所有成员
$.store.. price	商店里所有东西的价格
$..*	JSON 结构的所有递归成员
$.. book [*]	所有书籍
$.. book [0]	第一本书
$.. book [-1]	最后一本书
$.. book [0:2]	前两本书
$. book [0,1]	前两本书
$.. book [0:4]	从索引 0 到 3 的图书（不包括结尾索引）
$.. book [0:4:2]	位于索引 0、2 的图书
$.. book [？ (@.isbn)]	所有带有 isbn 编号的图书
$.. book [？ (@.price<10)]	所有书籍都低于 10 美元
'$.. book [？ (@.price < 10)] '	所有书都比10美元便宜。（如果路径包含空格，则必须用引号括起来）
'$.. book [？ (@ ["price"] < 10)] '	所有书籍都低于 10 美元
'$.. book [？ (@。 ["价格"] < 10)] '	所有书籍都低于 10 美元
$.. book [？ (@.price>=10&&@.price<=100)]	价格在 10 美元到 100 美元之间的所有图书（含）
'$.. book [？ (@.price>=10 &&@.price<=100)] '	价格在 10 美元到 100 美元之间（含）的所有图书。（如果路径包含空格，则必须用引号括起来）
$.. book [？ (@.sold==true||@.in-stock=false)]	所有图书已售出或缺货
'$.. book [？ (@.sold == true || @.in-stock == false)] '	所有图书已售出或缺货。（如果路径包含空格，则必须用引号括起来）
'$.store.book [？ (@。 ["类别"] == “小说”)] '	小说类中的所有图书
'$.store.book [？ (@。 ["类别"]! = “小说”)] '	非小说类的所有图书

更多过滤器表达式示例：

```
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> ON.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]"
```

受限语法

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

示例

路径	描述
.store.book [0] .author	第一本书的作者
.store.book [-1] .author	最后一本书的作者
.adress.city	城市名称
["store"] ["book"] [0] ["title"]	第一本书的标题
["store"] ["book"] [-1] ["title"]	最后一本书的标题

> **Note**
>
> 全部Goessner本文档中引用的内容受知识语法.

常见错误前缀

每条错误消息都有一个前缀。以下是常见错误前缀的列表：

Prefix	描述
呃	常规错误
LIMIT	超出大小限制错误。例如，超过文档大小限制或嵌套深度限制
不存在的	键或路径不存在
超出界限	数组索引超出界限
语法错误	语法错误
错误的类型	值类型不正确

JSON 相关指标

提供了以下 JSON 信息指标：

Info	描述
json_total_memory_bytes	分配给 JSON 对象的总内存量
json_num_documents	Redis 中的文档总数

要查询核心指标，请运行 Redis 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 数据类型进行交互。

运算符优先顺

在评估条件表达式以进行筛选时，&s 优先考虑，然后求值 ||s，这在大多数语言中很常见。括号内的操作将首先执行。

最大路径嵌套限制行为

MemoryDB 的最大路径嵌套限制为 128。所以一个值像`$.a.b.c.d...`只能达到 128 个等级。

处理数字值

JSON 没有单独的整数和浮点数数据类型。它们都是被叫的号码。

数字语法：

当输入时收到 JSON 数字时，它将转换为两种内部二进制表示形式之一：64 位有符号整数或 64 位 IEEE 双精度浮点数。原始字符串及其任何格式都不会保留。因此，当数字作为 JSON 响应的一部分输出时，它会使用通用格式规则从内部二进制表示形式转换为可打印字符串，这可能会导致生成的字符串与收到的字符串不同。

算运命令`NUMINCRBY`和`NUMMULTBY`：

- 如果两个数字都是整数，并且结果超出了`int64`，它将自动变为 64 位 IEEE 双精度浮点数。
- 如果至少有一个数字是浮点数，则结果将是 64 位 IEEE 双精度浮点数。
- 如果结果超出 64 位 IEEE double 的范围，则该命令将返回`OVERFLOW`错误消息。

有关可用命令的详细列表，请参阅。支持的命令 (p. 55).

直接语法

MemoryDB 直接过滤数组对象。

对于类似的数据`[0,1,2,3,4,5,6]`和类似 $[? (@<4)]，或者类似的数据`{"my_key": [0,1,2,3,4,5,6]}`还有一个路径查询`$.my_key[?(@<4)]`，在这两种情况下，MemoryDB 都会返回 [1,2,3]。

数组索引行为

MemoryDB 允许数组的正索引和负索引。对于长度为 5 的数组，0 将查询第一个元素，1 将查询第二个元素，依此类推。负数从数组的末尾开始，因此 -1 将查询第五个元素，-2 将查询第四个元素，依此类推。

为了确保客户的行为具有可预测性，MemoryDB 不会向下或向上舍入数组索引，因此，如果您的数组长度为 5，则调用索引 5 或更高、或 -6 或更低的索引将不会产生结果。

严格语法

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。这是为了让我们的客户保持正确的行为。

支持的命令

支持以下 Redis JSON 命令：

主题

- JSON.STRLEN (p. 80)
- JSON.TOGGLE (p. 81)
- JSON.TYPE (p. 82)

JSON.ARRAPPEND

将一个或多个值附加到路径处的数组值。

语法

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

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（必需）— 一个 JSON 路径
- json（必需）— 要附加到数组的 JSON 值

返回值

如果路径是增强的语法：

- 整数数组，表示每条路径处数组的新长度。
- 如果值不是数组，则其对应的返回值为 null。
- SYNTAXERR如果其中一个输入 json 参数不是有效的 JSON 字符串，则会出现错误。
- NONEXISTENT如果路径不存在，则错误。

如果路径是受限语法：

- 整数，数组的新长度。
- 如果选择了多个数组值，则该命令将返回上次更新的数组的新长度。
- WRONGTYPE如果路径中的值不是数组，则为错误。
- SYNTAXERR如果其中一个输入 json 参数不是有效的 JSON 字符串，则会出现错误。
- NONEXISTENT如果路径不存在，则错误。

示例

增强路径语法：

```
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

在路径的数组中搜索首次出现的标量 JSON 值。

- 超出范围错误的处理方法是将索引四舍五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]]
```

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（必需）— 一个 JSON 路径
- json-scalar（必需）— 要搜索的标量值；JSON 标量指的是不是对象或数组的值。也就是说，字符串、数字、布尔值和 null 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，不包括。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 表示包含最后一个元素。

返回值

如果路径是增强的语法：

- 整数数组。每个值都是路径处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。

如果路径是受限语法：

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

示例

增强路径语法：

```
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

在索引之前的路径处向数组值插入一个或多个值。

语法

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

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（必需）— 一个 JSON 路径
- index（必需）— 在其中插入值的数组索引。
- json（必需）— 要附加到数组的 JSON 值

返回值

如果路径是增强的语法：

- 整数数组，表示每条路径处数组的新长度。
- 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。
- `OUTOFBOUNDARIES`如果索引参数超出了界限，则错误。

如果路径是受限语法：

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

示例

增强路径语法：

```
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

获取路径处数组值的长度。

语法

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

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（可选）— 一个 JSON 路径。如果未提供，则默认为 root

返回值

如果路径是增强的语法：

- 整数数组，表示每条路径的数组长度。
- 如果值不是数组，则其对应的返回值为 null。
- 如果文档密钥对不存在，则为空。

如果路径是受限语法：

- 批量字符串数组。每个元素都是对象中的键名。
- 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。
- WRONGTYPE如果路径中的值不是数组，则为错误。
- WRONGTYPE如果路径不存在，则错误。
- 如果文档密钥对不存在，则为空。

示例

增强路径语法：

```
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。弹出一个空数组返回 null。

语法

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

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（可选）— 一个 JSON 路径。如果未提供，则默认为 root
- index（可选）— 数组中要开始弹出的位置。
  - 如果未提供，则默认为 -1，这意味着最后一个元素。
  - 负值表示距离最后一个元素的位置。
  - 越界索引四舍五入到其各自的数组边界。

返回值

如果路径是增强的语法：

- 批量字符串数组，表示每个路径处的弹出值。
- 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。

如果路径是受限语法：

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

示例

增强路径语法：

```
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

在路径处修剪数组，使其成为子数组 [start, end]，两者都包括在内。

- 如果数组为空，则无效，返回 0。
- 如果 start <0，则将其视为 0。
- 如果 end >= 大小（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>
```

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（必需）— 一个 JSON 路径
- start（必需）— 起始索引（含）。
- end（必需）— 结束索引，包含。

返回值

如果路径是增强的语法：

- 整数数组，表示每条路径处数组的新长度。
- 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。
- `OUTOFBOUNDARIES`如果索引参数超出了界限，则错误。

如果路径是受限语法：

- 整数，数组的新长度。
- 如果数组为空，则为空。
- WRONGTYPE如果路径中的值不是数组，则为错误。
- OUTOFBOUNDARIES如果索引参数超出了界限，则错误。

示例

增强语法：

```
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]
```

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（可选）— 一个 JSON 路径。如果未提供，则默认为 root

返回值

- 整数，已清理的容器数量。
- 清除空数组或对象会导致清除 1 个容器。
- 清除非容器值返回 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) 7
127.0.0.1:6379> JSON.CLEAR k1  $[*]
```

```
(integer) 4
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：

- 内存 <key>[路径] — 报告 JSON 值的内存使用情况（以字节为单位）。如果未提供 path，则默认为根。
- FIELDS <key>[路径] — 报告指定文档路径中的字段数。如果未提供 path，则默认为根。每个非容器 JSON 值都算作一个字段。对象和数组以递归方式为其包含 JSON 值的每个字段计数一个字段。除根容器外，每个容器值都算作一个附加字段。
- HELP — 打印命令的帮助消息。

语法

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

取决于子命令：

记忆

- 如果路径是增强的语法：
  - 返回一个整数数组，表示每个路径上的 JSON 值的内存大小（以字节为单位）。
  - 如果 Redis 密钥对不存在，则返回空数组。
- 如果路径是受限语法：
  - 返回一个整数，内存大小为 JSON 值（以字节为单位）。
  - 如果 Redis 密钥对不存在，则返回空值。

FIELDS

- 如果路径是增强的语法：
  - 返回一个整数数组，表示每个路径上具有 JSON 值的字段数。
  - 如果 Redis 密钥对不存在，则返回空数组。
- 如果路径是受限语法：
  - 返回整数，JSON 值的字段数。
  - 如果 Redis 密钥对不存在，则返回空值。

HELP — 返回一组帮助消息。

示例

增强语法：

```
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

删除文档密钥中路径处的 JSON 值。如果路径是根，则相当于从 Redis 中删除密钥。

语法

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

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（可选）— 一个 JSON 路径。如果未提供，则默认为 root

返回值

- 已删除元素的数量。
- 如果 Redis 密钥对不存在，则为 0。
- 如果 JSON 路径无效或不存在，则返回 0。

示例

增强语法：

```
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

别名JSON.DEL (p. 64)

JSON.GET

在一个或多个路径中返回序列化的 JSON。

语法

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

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 缩进/换行符/空格（可选）— 控制返回的 JSON 字符串的格式，即 “漂亮打印”。每一个的默认值都是空字符串。它们可以任意组合被覆盖。可以按任意顺序指定它们。

- NOESCAPE-可选，允许存在以实现旧版兼容性，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。路径参数必须放在末尾。

返回值

增强语法：

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，该命令将返回空数组。

如果给出了多条路径：

- 返回一个字符串化的 JSON 对象，其中每个路径都是一个键。
- 如果混合了增强路径语法和受限路径语法，则结果符合增强的语法。
- 如果路径不存在，则对不存在，其对不存在，其对不存在，其对不存在，其对不存在

示例

增强语法：

```
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

从多个文档密钥的路径中获取序列化的 JSON。对于不存在的密钥或 JSON 路径返回 null。

语法

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

- key（必需）— 一个或多个文档类型的 Redis 密钥。
- 路径（必需）— 一个 JSON 路径

返回值

- 批量字符串数组。数组的大小等于命令中的键数。数组的每个元素都填充 (a) 路径所处的序列化的 JSON，或 (b) 如果密钥不存在或路径在文档中不存在或路径无效（语法错误），则填充为 Null。
- 如果存在任何指定的密钥且不是 JSON 密钥，则命令返回`WRONGTYPE`错误消息。

示例

增强语法：

```
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一个给定的数字。

语法

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

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（必需）— 一个 JSON 路径
- 数字（必填）— 一个数字

返回值

如果路径是增强的语法：

- 批量字符串数组，表示每个路径的结果值。
- 如果值不是数字，则其对应的返回值为 null。
- WRONGTYPE如果无法解析该数字，则为错误。
- OVERFLOW如果结果超出 64 位 IEEE 双精度范围，则为错误。
- NONEXISTENT如果文档密钥对不存在，则无效。

如果路径是受限语法：

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

示例

增强语法：

```
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

将路径上的数字值乘以给定的数字。

语法

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

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（必需）— 一个 JSON 路径
- 数字（必填）— 一个数字

返回值

如果路径是增强的语法：

- 批量字符串数组，表示每个路径的结果值。
- 如果值不是数字，则其对应的返回值为 null。
- `WRONGTYPE`如果无法解析该数字，则为错误。
- `OVERFLOW`如果结果超出 64 位 IEEE 双精度范围，则为错误。
- `NONEXISTENT`如果文档密钥对不存在，则无效。

如果路径是受限语法：

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

示例

增强语法：

```
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

获取路径中对象值中的键数。

语法

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

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（可选）— 一个 JSON 路径。如果未提供，则默认为 root

返回值

如果路径是增强的语法：

- 整数数组，表示每条路径的对象长度。
- 如果值不是对象，则其对应的返回值为 null。
- 如果文档密钥对不存在，则为空。

如果路径是受限语法：

- 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。
- WRONGTYPE如果路径中的值不是对象，则为错误。
- WRONGTYPE如果路径不存在，则错误。
- 如果文档密钥对不存在，则为空。

示例

增强语法：

```
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

在路径的对象值中获取键名。

语法

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

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（可选）— 一个 JSON 路径。如果未提供，则默认为 root

返回值

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的数组。
- 如果值不是对象，则其对应的返回值为空值。
- 如果文档密钥对不存在，则为空。

如果路径是受限语法：

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

示例

增强语法：

```
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 Redis 序列化协议 (RESP) 中给定路径处的 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 空批量字符串。
- 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 [，后面是数组的元素。
- JSON 对象表示为 RESP 数组，其中第一个元素是简单字符串 {，后面是键值对，每个都是 RESP 批量字符串。

语法

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

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（可选）— 一个 JSON 路径。如果未提供，则默认为 root

返回值

如果路径是增强的语法：

- 数组。每个数组元素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。

如果路径是受限语法：

- 数组，表示路径中值的 RESP 形式。
- 如果文档密钥对不存在，则为空。

示例

增强语法：

```
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

在路径处设置 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 NOPLIIT。
- 如果父元素存在但不是对象，则该命令将返回 ERROR。
- 如果父元素存在并且是对象：
  - 如果该成员不存在，则当且仅当父对象是路径中的最后一个子级时，才会将新成员附加到父对象。否则，该命令将返回不存在。
  - 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 NOPLIIT。
- 如果父元素存在但不是数组，则该命令将返回 ERROR。
- 如果父元素存在但索引超出界限，则该命令将返回 OUTOFBOUNDIARES 错误。

- 如果父元素存在且索引有效，则该元素将被新的 JSON 值替换。

如果路径调用对象或数组，则该值（对象或数组）将被新的 JSON 值替换。

语法

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

[NX | XX] 在哪里可以有 0 或 1 个 [NX | XX] 个标识符

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（必需）— JSON 路径。对于新的 Redis 密钥，JSON 路径必须是根 “.”。
- NX（可选）— 如果路径是根路径，则仅在 Redis 键不存在时才设置该值，即插入新文档。如果路径不是根路径，则仅在路径不存在时才设置值，即在文档中插入值。
- XX（可选）— 如果路径是根目录，则仅在 Redis 键存在时才设置该值，即替换现有文档。如果路径不是根路径，则仅在路径存在时才设置值，即更新现有值。

返回值

- 成功时使用简单字符串 “OK”。
- 如果未满足，则无效。

示例

增强语法：

```
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中的 JSON 字符串。

语法

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

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（可选）— 一个 JSON 路径。如果未提供，则默认为 root
- json_string（必需）— 字符串的 JSON 表示形式。请注意，JSON 字符串必须用引号括起来，即 '"foo"'。

返回值

如果路径是增强的语法：

- 整数数组，表示每个路径处字符串的新长度。
- 如果路径中的值不是字符串，则其对应的返回值为 null。
- `SYNTAXERR` 如果输入 json 参数不是有效的 JSON 字符串，则会出现错误。
- `NONEXISTENT` 如果路径不存在，则错误。

如果路径是受限语法：

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

示例

增强语法：

```
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]
```

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（可选）— 一个 JSON 路径。如果未提供，则默认为 root

返回值

如果路径是增强的语法：

- 整数数组，表示每个路径处的字符串值的长度。
- 如果值不是字符串，则相应返回值为 null。
- 如果文档密钥对不存在，则为空。

如果路径是受限语法：

- 整数，字符串的长度。
- 如果选择了多个字符串值，则该命令将返回第一个字符串的长度。
- `WRONGTYPE`如果路径中的值不是字符串，则为错误。
- `NONEXISTENT`如果路径不存在，则错误。
- 如果文档密钥对不存在，则为空。

示例

增强语法：

```
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

在路径处在 true 和 false 之间切换布尔值。

语法

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

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（可选）— 一个 JSON 路径。如果未提供，则默认为 root

返回值

如果路径是增强的语法：

- 整数数组（0-false，1-true），表示每个路径处的结果布尔值。
- 如果值不是布尔值，则其对应的返回值为 null。
- `NONEXISTENT`如果文档密钥对不存在，则无效。

如果路径是受限语法：

- 表示结果布尔值的字符串 (“true”/“false”) 表示结果的布尔值。
- `NONEXISTENT`如果文档密钥对不存在，则无效。
- `WRONGTYPE`如果路径中的值不是布尔值，则为错误。

示例

增强语法：

```
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

给定路径的值的报告类型。

语法

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

- 密钥（必需）— JSON 文档类型的 Redis 密钥
- 路径（可选）— 一个 JSON 路径。如果未提供，则默认为 root

返回值

如果路径是增强的语法：

- 字符串数组，表示每个路径的值的类型。类型是 {"null"、“布尔”、“字符串”、“数字”、“整数”、“对象” 和 “数组”} 之一。
- 如果路径不存在，则其对应的返回值为 null。

- 如果文档键不存在，则为空数组。

如果路径是受限语法：

- 字符串，值的类型
- 如果文档密钥对不存在，则为空。
- 如果 JSON 路径无效或不存在，则为空。

示例

增强语法：

```
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
```

标记内存 DB 资源

为了帮助您管理集群和其他 MemoryDB 资源，您可通过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 Amazon 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。

有关标签的基本知识

标签是为Amazon资源分配的标记。每个标签都包含您定义的一个键 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 Amazon 资源进行分类。例如，您可以为账户中的 MemoryDB 群集定义一组标签，以跟踪每个群集的拥有者和用户组。

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Amazon为最佳实践添加标记白皮书.

标签对于内存没有任何语义意义，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。

您可以使用使用来处理标签Amazon Web Services Management Console，Amazon CLI，以及 MemoryDB API。

如果您使用的是 IAM，则可以控制 Amazon 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 (p. 188)。

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内存资源。下表列出了支持标记的资源。如果您使用的是 Amazon Web Services Management Console，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。控制台可能会根据名称标签，但此标签对于 MemoryDB 服务没有任何语义意义。

此外，某些资源创建操作允许您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 Amazon 内存 DB API，则AmazonCLI 或Amazon开发工具包，您可以使用`Tags`用于应用标签的相关 MemoryDB API 操作的参数。它们是：

- `CreateCluster`
- `CopySnapshot`
- `CreateParameterGroup`
- `CreateSubnetGroup`
- `CreateSnapshot`
- `CreateACL`
- `CreateUser`

下表描述了可以标记的 MemoryDB 资源以及可在创建时使用 MemoryDB API 标记的资源。AmazonCLI 或Amazon发工具包。

标记支持用于 MemoryDB 资源的支持

支持标签	支持在创建时标记
是 parametergroup	是
是 subnetgroup	是
是 集群	是

支持标签	支持在创建时标记
是	是
是	是
是	是

对于支持在创建时标记的 MemoryDB API 操作，您可以在 IAM 策略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 (p. 86)。

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 (p. 86)。

为群集和快照添加标签

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，则将标记集群。
  - 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：

  标签将从资源中添加/删除。

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)。

- 尽管 MemoryDB 允许在标签中使用任何字符，但其他服务可能具有严格限制。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 Amazon 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 资源的更多信息，请参阅您可以为之添加标签的资源 (p. 84).

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，即使群集包含标签，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 in Redis 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。

MemyDB 成本分配标签是您定义的键-值对，此标签与 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 Amazon 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 Amazon 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。然后，您可以按区域查看总体内存成本细分。有关更多信息，请参阅 Amazon Billing 用户指南中的使用成本分配标签。

您可以向 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`
- 标签密钥是标签的名称，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`。

Amazon 不会对您的标签应用任何语义意义。标签严格按字符串进行解释。Amazon不会自动在任何内存 DB 资源上设置任何标签。

使用 Amazon CLI 管理成本分配标签

您可以使用 Amazon 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 Amazon CLI 列出标签

您可以使用Amazon CLI使用列出现有内存资源上的标签，方法是使用列出标签operation.

下面的代码使用Amazon CLI列出 MemoryDB 集群上的标签`my-cluster`在 us-east-1 区域中。

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，则输出空TagList.

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

有关更多信息，请参阅 。Amazon CLI适用于内存 DB列出标签.

使用 Amazon CLI 添加标签

您可以使用Amazon CLI使用向现有 MemoryDB 资源添加标签，方法是使用tag-resourceCLI 操作。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Amazon CLI添加钥匙`Service`和`Region`有这些价值`memorydb`和`us-east-1`分别到群集`my-cluster`在 us-east-1 区域中。

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 。Amazon CLI适用于内存 DBtag-resource.

您也可以使用Amazon CLI使用操作，在创建新集群时向集群添加标签create-cluster.

使用 Amazon CLI 修改标签

您可以使用Amazon CLI修改内存集群上的标签。

修改标签：

- 使用tag-resource添加新标签和值，或更改与现有标签关联的值。
- 使用untag-resource从资源中删除指定标签。

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 Amazon CLI 删除标签

您可以使用Amazon CLI使用从 MemoryDB 集群中删除现有标签，方法是使用untag-resourceoperation.

下面的代码使用Amazon CLI用钥匙删除标签`Service`和`Region`来自群集`my-cluster`在 us-east-1 区域。

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 。Amazon CLI适用于内存 DBuntag-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集群的内存 DB。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

- 使用内存 API 列出标签 (p. 90)

使用内存 API 列出标签

您可以使用 MemoryDB API 列出现有资源上的标签，方法是使用ListTagsoperation.

以下代码使用内存 API 列出资源上的标签。`my-cluster`在 us-east-1 区域。

```
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>
```

使用内存 API 添加标签

您可以使用内存 API 向现有 MemoryDB 集群添加标签，方法是使用TagResourceoperation. 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 MemoryDB API 来添加密钥`Service`和`Region`有这些价值`memorydb`和`us-east-1`分别到资源`my-cluster`在 us-east-1 区域。

```
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.

使用内存 API 修改标签

您可以使用内存 API 修改内存集群上的标签。

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。
- 使用 UntagResource 可删除资源的标签。

以上任意操作的输出将是指定资源上标签及其值的列表。

使用内存 API 删除标签

您可以使用内存 API 从现有 MemoryDB 集群中删除标签，方法是使用UntagResourceoperation.

以下代码使用 MemoryDB API 来删除带有密钥的标签`Service`和`Region`来自群集`my-cluster`在 us-east-1 区域中。

```
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 会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	亚太区域（首尔）	12:00–20:00 UTC
ap-south-1	亚太（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（法兰克福）区域	23:00–07:00 UTC
eu-west-1	欧洲（爱尔兰）区域	22:00–06:00 UTC
eu-west-2	欧洲（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（弗吉尼亚北部）区域	03:00–11:00 UTC
us-east-1	美国东部（俄亥俄）区域	04:00–12:00 UTC
us-west-1	美国西部（加利福尼亚北部）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 (p. 23) – 管理节点替换
- 修改 MemoryDB 集群 (p. 30) – 更改集群的维护时段

最佳实践

以下您可以找到使用的最佳实践：遵循最佳实践可改进您集群的性能和可靠性。

主题

受限 Redis 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`

用于 Redis 的 MemoryDB 中的弹性

Amazon全球基础设施围绕Amazon区域和可用区构建。Amazon区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 Amazon 区域和可用区的更多信息，请参阅 Amazon 全球基础设施。

除了AmazonRedis 的 MemoryDB 全球基础设施提供了多种功能，以帮助支持您的数据弹性和快照需求。

主题

缓解故障

规划用于 Redis 实施的 MemoryDB 时，您应做好计划以最大程度地减小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：MailyDB 集群集群

MemoryDB 集群由应用程序可从中读取和写入的主节点以及 0 至 5 个只读副本节点组成的只读副本节点。但是，我们强烈建议至少使用 1 个副本以实现高可用性。在向主节点写入数据时，都将保留到事务日志中，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. MailyDB 检测到发生故障的副本。
2. MailyDB 使发生故障的节点脱机。
3. 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。

MailyDB 多可用区

如果在 MemoryDB 集群上启动多可用区，将自动检测并替换发生故障的主集群。

1. MailyDB 检测到主节点故障。
2. 在确保副本与失败的主副本一致后，MemoryDB 会故障转移到副本。
3. 在发生故障的主集群的可用区中启动副本。
4. 新节点将与事务日志同步。

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 (p. 97)。

最佳实践：在线集群大小调整

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区期间测试应用程序行为。
- 获取扩展问题的提前通知 – 重新分区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`、`Freeable` 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，请确保要保留的分区上的可用空余内存至少是您计划删除的分区上已用内存的 1.5 倍。
- 在非高峰时间启动重新分区 – 此做法有助于减少重新分区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。
- `FLUSHALL`和`FLUSHDB`重新分片期间，Lua 脚本内不支持和命令。

了解内存 DB 复制

内存 DB 实施复制，具有跨多达 500 个分片的数据进行复制。

集群中的每个分区包装一个读/写主节点和最多 5 个只读副本节点。每个主节点可以维持高达 100 MB/s。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 5 个副本，这是允许的最大副本数。

一致性

在 MemoryDB 中，主节点是强烈的一致性。在返回客户端之前，成功的写入操作会持久存储在分布式多可用区事务日志中。对初选的读取操作总是返回最多up-to-date反映来自所有先前成功的写入操作的影响的数据。在主故障转移之间保持了这种强大的一致性。

在内存 DB 中，副本节点具有最终一致性。从副本中读取操作（使用`READONLY`命令）可能并不总是反映最近成功写入操作的影响，延迟指标发布到CloudWatch. 但是，来自单个副本的读取操作依次一致。成功的写入操作按照在主副本上执行的顺序对每个副本生效。

集群中的复制

分片中的每个只读副本都保留一个分片主节点数据的副本。使用使用事务日志的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。只读副本增强读取可扩展性。由于 MemoryDB 将数据存储在持久的事务日志中，因此没有数据丢失的风险。数据在内存 DB 集群中的分片上进行分区。

应用程序使用 MemoryDB 集群集群终端节以连接集群中的节点。有关更多信息，请参阅 查找连接端点 (p. 39)。

MemoryDB 集群是区域性的，只能包含来自一个区域的节点。若要增强容错能力，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。

强烈建议对所有 MemoryDB 集群使用复制功能，它为您提供了多可用区。有关更多信息，请参阅 利用多可用区最大限度地减少 MemoryDB (p. 97)。

利用多可用区最大限度地减少 MemoryDB

在许多情况下，MemoryDB 可能需要替换主节点；其中包括某些类型的计划维护以及出现不太可能出现的主节点或可用区故障的事件。

对节点故障的响应取决于哪个节点发生了故障。但是，在任何情况下，MemoryDB 都可以确保在节点更换或故障转移期间不会丢失任何数据。例如，如果副本出现故障，则会替换出现故障的节点，并从事务日志中同步数据。如果主节点出现故障，将触发到一致副本的故障转移，从而确保在故障转移期间不会丢失任何数据。现在，写入操作将从新的主节点处理。然后替换旧的主节点并从事务日志中同步。

如果主节点在单个节点分片（没有副本）上出现故障，MemoryDB 将停止接受写入操作，直到主节点被替换并从事务日志同步。

节点更换可能会导致群集停机时间，但如果多可用区处于活动状态，则停机时间将最小化。主节点的角色将自动故障转移到其中一个副本。无需创建和预配置新的主节点，因为 MemoryDB 将以透明的方式处理此问题。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节点替换，请注意，在集群处理传入的写请求时，将注意计划节点替换已完成。

MemoryDB 集群上的多可用区提高了容错能力。在集群的主节点出于任何原因变得无法访问或发生故障时，此情况尤其如此。MemoryDB 群集上的多可用区要求每个分片都必须有多个节点，并自动启用。

主题

故障情形及多可用区响应

如果多可用区处于活动状态，发生故障的主节点故障转移到可用副本。该副本将自动与事务日志同步，并成为主节点，这比创建和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

如果只有主节点发生故障，则副本将自动成为主节点。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配置替换副本。

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. 最新的副本会自动成为主副本。

   一旦故障转移过程完成（通常只需几秒钟的时间），写入操作就会恢复。
3. 启动和预配置替换副本。

   在发生故障的主节点所在的可用区中启动替换副本，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 集群的终端节点 (MemoryDB API) (p. 42)

主节点和某些副本发生故障时的故障情况

如果主节点和至少一个副本发生故障，则最新的副本将提升为主集群。在与发生故障的节点相同的可用区域中创建和预配置新的副本。

当主节点和一些副本发生故障时，MemoryDB 多可用区将执行以下操作：

1. 发生故障的主节点和发生故障的副本脱机。
2. 可用的副本将成为主节点。

   一旦故障转移完成（通常只需几秒钟的时间），写入操作就会恢复。
3. 创建和预配置替换副本。

   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 集群的终端节点 (AmazonCLI) (p. 40)
- 查找 MemoryDB 集群的终端节点 (MemoryDB API) (p. 42)

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。

在这种情况下不会丢失数据，因为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. 创建和预配置替换主节点，将与事务日志同步。
3. 创建和预配置替换副本，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 集群的终端节点 (AmazonCLI) (p. 40)
- 查找 MemoryDB 集群的终端节点 (MemoryDB API) (p. 42)

测试自动故障转移

您可以使用 MemoryDB 控制台测试自动故障切换Amazon CLI，以及 MemoryDB API。

在测试时，请注意以下内容：

- 在每个集群的任意 24 小时滚动期间，您最多可以使用此操作五次。
- 如果在不同集群的分片上调用此操作，您可以同时进行调用。
- 在某些情况下，您可能会在同一个 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，请使用主分区控制台检查事件，Amazon CLI，或 MemoryDB API。查找以下与`FailoverShard`，此处按可能发生的顺序列出：
  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中的内存分区 API 参考
- 此 API 旨在测试应用程序在 MemoryDB 故障转移时的行为。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，在大型运营活动等特定情况下，Amazon 可能会阻止此 API。

主题

使用 Amazon Web Services Management Console 测试自动故障转移

使用以下过程测试通过控制台进行自动故障转移。

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB and Redis 控制台https://console.aws.amazon.com/memorydb/.
2. 选择要测试的集群左侧的单选按钮。此集群必须至少具有一个副本节点。
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 (p. 30)。
4. 选择集群的名称。
5. 在存储库的分片和节点页面上，对于要测试故障转移的分片，选择分片的名称。
6. 对于节点，选择故障转移主.
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 Amazon CLI 测试自动故障转移

您可以在任何启用多可用区的集群上测试自动故障转移。Amazon CLI手术故障转移分片.

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Amazon CLI要打电话`failover-shard`在碎片`0001`在 MemoryDB 集群中`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 Amazon CLI `describe-events` 操作。

它将返回以下 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：

- 故障转移分片
- describe-events

使用 MemoryDB API 测试自动故障转移

以下示例调用：`FailoverShard`在碎片`0003`在集群中`memorydb00`.

Example 测试自动故障转移

```
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>
```

要跟踪故障转移的进度，请使用 MemoryDB`DescribeEvents`API 操作。

有关更多信息，请参阅下列内容：

- 故障转移分片
- DescribeEvents

更改副本数量

您可以使用 “” 动态地增加或减少 MememReplicyDB 集群中的只读副本数量。Amazon Web Services Management Console，Amazon CLI，或 MemoryDB API。所有分片必须具有相同数量的副本。

增加集群中的副本数量

您可以将 MemreyDB 集群中的副本数量最多增加到每个分片 5 个。您可以使用Amazon Web Services Management Console，Amazon CLI，或 MemoryDB API。

主题

使用 Amazon Web Services Management Console

要增加内存 DB 集群（控制台）中的副本数量，请参阅从集群中添加/删除节点 (p. 32).

使用 Amazon CLI

要增加内存 DB 集群中的副本数量，请使用`update-cluster`命令具有以下参数：

- `--cluster-name` – 必需。确定要在其中增加副本数量的集群。
- `--replica-configuration` – 必需。允许您设置副本数量。要增加副本数量，请将`ReplicaCount`属性到您希望在此操作结束时此分片中所具有副本数量。

Example

以下示例增加了集群中的副本数量`my-cluster`到 2.

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
```

它将返回以下 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": "6.2.4",
        "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",
        "AutoMinorVersionUpgrade": true
    }
}
```

更新后的集群的详细信息更新到可用中，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
```

它将返回以下 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": "6.2.4",
            "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",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

有关使用 CLI 增加副本数量的更多信息，请参阅更新中的Amazon CLI命令参考。

使用内存 DB API

要增加内存 DB 分片中的副本数量，请使用`UpdateCluster`具有以下参数的操作：

- `ClusterName` – 必需。确定要在其中增加副本数量的集群。
- `ReplicaConfiguration` – 必需。允许您设置副本数量。要增加副本数量，请将`ReplicaCount`属性到您希望在此操作结束时此分片中所具有副本数量。

Example

以下示例增加了集群中的副本数量`sample-cluster`到 3 个。完成此示例后，每个分片中将有三个副本。无论这是具有单个分片的 MemoryDB 集群还是具有多个分片的 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

您可以减少内存 DB 的集群中的副本数量。您可以将副本数量减少到 0，但是如果您的主节点发生故障，则无法故障转移到副本。

您可以使用Amazon Web Services Management Console，Amazon CLI或者使用内存 DB API 减少集群中的副本数量。

主题

使用 Amazon Web Services Management Console

要减少内存 DB 集群（控制台）中的副本数量，请参阅从集群中添加/删除节点 (p. 32).

使用 Amazon CLI

要减少内存 DB 集群中的副本数量，请使用`update-cluster`命令具有以下参数：

- `--cluster-name` – 必需。确定要在其中减少副本数量的集群。
- `--replica-configuration` – 必需。

  `ReplicaCount`— 设置此属性以指定希望的副本节点数。

Example

以下示例使用`--replica-configuration`减少集群中的副本数量`my-cluster`到指定的值。

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 ^
```

它将返回以下 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": "6.2.4",
        "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",
        "AutoMinorVersionUpgrade": true
    }
}
```

更新后的集群的详细信息更新到可用中，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
```

它将返回以下 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": "6.2.4",
            "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",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

有关使用 CLI 减少副本数量的更多信息，请参阅更新中的Amazon CLI命令参考。

使用内存 DB API

要减少内存 DB 集群中的副本数量，请使用`UpdateCluster`具有以下参数的操作：

- `ClusterName` – 必需。确定要在其中减少副本数量的集群。
- `ReplicaConfiguration` – 必需。允许您设置副本数量。

  `ReplicaCount`— 设置此属性以指定希望的副本节点数。

Example

以下示例使用`ReplicaCount`减少集群中的副本数量`sample-cluster`对一个。完成此示例后，每个分片中将有一个副本。无论这是具有单个分片的 MemoryDB 集群还是具有多个分片的 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

适用于 Redis 集群的 MemoryDB 会自动将数据备份到多可用区事务日志，但您可以选择创建 point-in-time 定期或按需集群的快照。这些快照可用于在之前的某个时间点重新创建集群或为全新的集群播种。快照包

含集群的元数据以及集群中的所有数据。所有快照都会写入亚马逊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，您可以使用Amazon Web Services Management Console，Amazon Command Line Interface(Amazon CLI) 和 MemoryDB API。

主题

快照约束

在计划或创建快照时考虑以下约束：

- 对于 MemoryDB 集群，快照和还原适用于所有受支持的节点类型。
- 在任何连续的 24 小时期间，每个集群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 始终从主节点获取快照。这可确保您在删除集群之前捕获最新数据。

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，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Amazon地区。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，您可启用自动快照。启用自动快照后，MemoryDB 会每天为集群创建快照。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 (p. 125)。

当您计划自动快照时，应规划以下设置：

- 快照窗口— 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创建快照时段。

  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，则当天获取的快照将保留 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。即使禁用了自动快照，MemoryDB 数据仍然完全持久。

您可在创建 MemoryDB 集群时使用 MemoryDB 控制台，启用或禁用自动快照Amazon CLI，或 MemoryDB API。您可以在创建 MemoryDB 集群时启用自动快照，方法是选中启用自动备份在处登录快照部分. 有关更多信息，请参阅 创建内存数据库集群 (p. 13)。

手动创建快照

除了自动快照之外，您还可以创建一个手册请随时快照。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 (p. 117)— 源快照是自动还是手动创建并不重要。
- 创建最终快照 (p. 113)— 创建快照，然后立即删除集群。

其他重要主题

- 快照约束 (p. 109)
- 快照费用 (p. 109)

您可以使用创建节点的手动快照Amazon Web Services Management Console，Amazon CLI，或MemoryDB API。

创建手动快照（控制台）

创建集群的快照 (控制台)

1. 登录到Amazon Web Services Management Console然后打开适用于 Redis 的 MemoryDB 控制台https://console.aws.amazon.com/memorydb/.
2. 从左侧导航窗格中选择集群.

   此时将出现 MemoryDB 群集屏幕。
3. 选择要备份的 MemoryDB 集群名称左侧的单选按钮。
4. 选择操作然后拍摄快照.
5. 在快照窗口中，输入快照的名称快照名称。建议该名称指明所备份的集群以及创建快照的日期和时间。

   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. UNDER加密中，选择是使用默认加密密钥还是客户管理的密钥。有关更多信息，请参阅 传输中加密 (TLS) (p. 171)。
7. UNDER标签，或者，添加标签来搜索和筛选您的快照，或跟踪Amazon成本.
8. 选择 Take snapshot (拍摄快照)。

   集群的状态将变为快照。当状态变回可用快照已完成。

创建手动快照 (AmazonCLI)

创建集群的手动快照Amazon CLI，请使用`create-snapshot` Amazon CLI操作使用以下参数：

- `--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《Amazon CLI 命令参考》中的 `create-snapshot`。

创建手动快照（内存数据库 API）

要使用 MemoryDB API 创建集群的手动快照，请使用`CreateSnapshot`使用以下参数的 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.

创建最终快照

您可以使用 MemoryDB 控制台，Amazon CLI，或 MemoryDB API。

创建最终快照（控制台）

您可以在使用 MemoryDB 控制台删除 MemoryDB 集群时创建最终快照。

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给快照起个名字第 4 步：删除集群 (p. 19).

创建最终快照 (AmazonCLI)

在删除 MemoryDB 集群时，您可以使用Amazon CLI.

删除 MemoryDB 集群集群集群时

要在删除集群时创建最终快照，请使用`delete-cluster` Amazon CLI操作，参数如下：

- `--cluster-name` – 要删除的集群的名称。
- `--final-snapshot-name`— 最终快照的名称。

以下代码生成最终快照`bkup-20210515-final`删除集群时`myCluster`.

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 。delete-cluster中的Amazon CLI命令参考.

创建最终快照（内存数据库 API）

您可在删除 MemoryDB 集群时使用 MemoryDB API 创建最终快照。

删除 MemoryDB 集群集群集群时

要创建最终快照，请使用`DeleteCluster`使用以下参数的 MemoryDB API 操作。

- `ClusterName` – 要删除的集群的名称。
- `FinalSnapshotName`— 快照的名称。

以下 MemoryDB API 操作创建快照`bkup-20210515-final`删除集群时`myCluster`.

```
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)

使用显示快照Amazon Web Services Management Console

1. 登录控制台
2. 从左侧导航窗格中选择快照.
3. 使用搜索进行筛选手册、自动，或者所有快照。
4. 要查看特定快照的详细信息，请选择快照名称左侧的单选按钮。选择操作然后查看详细信息.
5. （可选）在查看详细信息页面中，您可以执行其他快照操作，例如复制、恢复要么删除. 您也可以向快照添加标签

描述快照 (AmazonCLI)

要显示快照列表及特定快照的可选详情，请使用`describe-snapshots`命令行界面操作。

示例

以下操作使用参数`--max-results`以列出与您的账户关联的最多 20 个快照。省略参数`--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`仅列出与该集群关联的快照`my-cluster`.

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

以下操作使用参数`--snapshot-name`显示快照的详细信息`my-snapshot`.

```
aws memorydb describe-snapshots --snapshot-name my-snapshot
```

有关更多信息，请参阅 。descee.

描述快照（内存数据库 API）

要显示快照列表，请使用`DescribeSnapshots`operation.

示例

以下操作使用参数`MaxResults`以列出与您的账户关联的最多 20 个快照。省略参数`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`列出与该集群相关联的所有快照`MyCluster`.

```
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`显示快照的详细信息`MyBackup`.

```
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。复制快照时，除非特别覆盖，否则将使用与源相同的 KMS 加密密钥作为目标。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 (p. 119).

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. 登录到Amazon Web Services Management Console然后打开适用于 Redis 的 MemoryDB 控制台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. 从快照列表中，选择要复制的快照名称左侧的单选按钮。
4. 选择操作然后选择Copy.
5. 在复制快照” 页面中，请执行以下操作：
   a. 在新快照名称框中，键入新快照的名称。
   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 (p. 119).
   c. 选择是否使用默认值Amazon KMS加密密钥或使用自定义密钥。有关更多信息，请参阅 传输中加密 (TLS) (p. 171)。
   d. 您可以选择将标签添加到快照副本。
   e. 选择 Copy（复制）。

复制快照 (AmazonCLI)

要复制快照，请使用`copy-snapshot`operation.

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 (p. 119)。

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 。copyyet.

复制快照（内存数据库 API）

要复制快照，请使用`copy-snapshot`操作使用以下参数：

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 (p. 119)。

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

适用于 Redis 的 MemoryDB 支持将 MemoryDB 快照导出到 Amazon Simple Storage Service (Amazon S3) 存储桶，因此您可以从 MemoryDB 外部访问该快照。导出的 MemoryDB 快照与开源 Redis 完全兼容，并且可以使用相应的 Redis 版本或工具进行加载。您可以使用 MemoryDB 控制台导出快照，Amazon CLI，或 MemoryDB API。

当您需要在其他Amazon区域。您可以导出您在某个 Amazon 区域中的数据，将 .rdb 文件复制到新的 Amazon 区域，然后使用该 .rdb 文件为新集群设定种子而不用等待新集群在使用过程中填充。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设定种子 (p. 128)。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 快照以及您希望将该快照复制到其中的 Amazon S3 存储桶必须位于同一个Amazon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。

您必须具有一个 Amazon S3 存储桶，才能将快照导出到 Amazon S3 存储桶Amazon区域作为快照。授予 MemoryDB 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。
> - 其他人有权使用 CopySnapshot API 操作。
>
>   具有使用权限的用户或组`CopySnapshot`API 操作可以创建自己的 Amazon S3 存储桶并将快照复制到其中。要控制对快照的访问，请使用Amazon Identity and Access Management(IAM) 策略来控制谁可以使用`CopySnapshot`API。有关使用 IAM 控制 MemoryDB API 操作使用的更多信息，请参阅Redis MemoryDB 中的身份和权限管理 (p. 180)中的内存 DB 用户指南.

主题

第 1 步：创建 Amazon S3 存储桶

以下步骤使用 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. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 选择 Create Bucket（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Region（区域）列表中，选择 Amazon S3 存储桶的 Amazon 区域。该Amazon区域必须相同Amazon区域作为您要导出的 MemoryDB 快照。

c. 选择Create（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。

第 2 步：授予 MemoryDB 对 Amazon S3 存储桶的访问权限

默认情况下，在 2019 年 3 月 20 日之前推出的 Amazon 区域为已启用状态。您可以立即开始在这些 Amazon 区域中工作。2019 年 3 月 20 日之后推出的区域默认情况下处于禁用状态。您必须按照所述，先启用或选择加入这些区域，然后才能使用它们。管理Amazon地区.

授予 MemoryDB 对 AU 中的 S3 存储桶的访问权限Amazon区域

要对中的 Amazon S3 存储桶创建适当的权限Amazon区域，请按以下步骤进行操作。

授予 MemoryDB 对 S3 存储桶的访问权限

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Simple Storage Service（Amazon S3）控制台：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。这应该是您在第 1 步：创建 Amazon S3 存储桶 (p. 119)中创建的 S3 存储桶。
3. 选择Permissions (权限)选项卡和下方Permissions (权限)，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。

```
{
    "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:::example-bucket",
                "arn:aws:s3:::example-bucket/*"
            ]
        }
    ]
}
```

第 3 步：导出 MemoryDB 快照

现在，您已经创建了 S3 存储桶并向授予 MemoryDB 访问它的权限。将 S3 对象所有权更改为已启用 ACL — 存储桶拥有者首选. 接下来，您可以使用 MemoryDB 控制台AmazonCLI 或 MemoryDB API 来将您的快照导出到存储桶。下面假设您拥有以下附加的 S3 特定 IAM 权限。

```
{
 "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 控制台将快照导出到 Amazon S3 存储桶，以便从 MemoryDB 外部访问它。Amazon S3 存储桶必须位于同一Amazon区域作为 MemoryDB 快照。

将 MemoryDB 快照导出到 Amazon S3 存储桶

1. 登录到Amazon Web Services Management Console然后打开适用于 Redis 的 MemoryDB 控制台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. 从快照列表中，选择要导出的快照名称左侧的单选按钮。
4. 选择 Copy（复制）。
5. 在 Create a Copy of the Backup? (创建备份副本?) 中，执行以下操作：

    a. In新快照名称框中，键入新快照的名称。

    名称必须在 1 到 1000 个字符之间，并能够以 UTF-8 编码。

    MemoryDB 添加了一个分片标识符和`.rdb`对您在此处输入的值进行修改。例如，如果您输入`my-exported-snapshot`，内存 DB 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（您在中创建的存储桶）的名称第 1 步：创建 Amazon S3 存储桶 (p. 119))。

    这些区域有：目标 S3 位置必须是快照中的 Amazon S3 存储桶Amazon具有以下权限的区域，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 第 2 步：授予 MemoryDB 对 Amazon S3 存储桶的访问权限 (p. 120)。

c. 选择 Copy（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，则您会收到以下某个错误消息。返回到第 2 步：授予 MemoryDB 对 Amazon S3 存储桶的访问权限 (p. 120)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：AddRead存储桶的权限访问权限。

如果想要将您的快照复制到其他存储Amazon区域，使用 Amazon S3 来复制它。有关更多信息，请参阅 。复制对象中的Amazon Storage Service.

导出 memoryDB 快照 (AmazonCLI)

使用将快照导出到 Amazon S3 存储桶`copy-snapshot`使用以下参数的 CLI 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。

  名称必须在 1 到 1000 个字符之间，并能够以 UTF-8 编码。

  MemoryDB 添加了一个分片标识符和`.rdb`对您在此处输入的值进行修改。例如，如果您输入`my-exported-snapshot`，内存 DB 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。

  这些区域有：`--target-bucket`必须是快照中的 Amazon S3 存储桶Amazon具有以下权限的区域，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 第 2 步：授予 MemoryDB 对 Amazon S3 存储桶的访问权限 (p. 120)。

以下操作将快照复制到 my-s3-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 my-s3-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 my-s3-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，则您会收到以下某个错误消息。返回到第 2 步：授予 MemoryDB 对 Amazon S3 存储桶的访问权限 (p. 120)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：AddRead存储桶的权限访问权限。

有关更多信息，请参阅《Amazon CLI 命令参考》中的 `copy-snapshot`。

如果想要将您的快照复制到其他存储Amazon区域，使用 Amazon S3 的复制操作。有关更多信息，请参阅 。复制对象中的Amazon Storage Service.

导出内存数据库快照 (MemoryDB API)

使用将快照导出到 Amazon S3 存储桶`CopySnapshot`使用以下参数进行 API 操作。

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 添加了一个分片标识符和`.rdb`对您在此处输入的值进行修改。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。

  这些区域有：`TargetBucket`必须是快照中的 Amazon S3 存储桶Amazon具有以下权限的区域，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 第 2 步：授予 MemoryDB 对 Amazon S3 存储桶的访问权限 (p. 120)。

以下示例将自动快照复制到 Amazon S3 存储桶`my-s3-bucket`.

Example

```
https://memory-db.us-east-1.amazonaws.com/
    ?Action=CopySnapshot
    &SourceSnapshotName=automatic.my-primary-2021-06-27-03-15
    &TargetBucket=my-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，则您会收到以下某个错误消息。返回到第 2 步：授予 MemoryDB 对 Amazon S3 存储桶的访问权限 (p. 120)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：AddRead存储桶的权限访问权限。

有关更多信息，请参阅 。CopySnapshot.

如果想要将您的快照复制到其他存储Amazon区域，使用 Amazon S3 的复制操作将导出的快照复制到其他 Amazon S3 存储桶Amazon区域。有关更多信息，请参阅 。复制对象中的Amazon Storage Service.

从快照还原

您可以从 MemoryDB 还原数据 ElastiCache 将 Redis .rdb 快照文件存储到新集群。

适用于 Redis 的 MemoryDB 还原流程支持以下操作：

- 从您创建的一个或多个 .rdb 快照文件 ElastiCache 将 Redis 转换为 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 Redis 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，当还原到 MemoryDB 时，如果 .rdb 文件引用多个数据库，还原将会失败。

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，当使用 MemoryDB 控制台进行还原时。使用时，可以通过设置参数值来进行这些选择Amazon CLI或要还原的 MemoryDB API。

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 (p. 111)。
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设定种子 (p. 128).

以下过程向您演示如何使用 MemoryDB 控制台，将快照还原到新集群。Amazon CLI，或 MemoryDB API。

从快照还原 (控制台)

将快照还原到新集群 (控制台)

1. 登录到Amazon Web Services Management Console然后打开适用于 Redis 的 MemoryDB 控制台https://console.aws.amazon.com/memorydb/.
2. 在导航窗格上，选择快照.
3. 在快照列表中，选择您要从中进行还原的快照名称旁的按钮。
4. 选择操作然后选择还原
5. UNDER集群集配置，输入以下内容：

   a. 集群名称— 必需。新集群的名称。

   b. 说明— 可选。新集群的描述。
6. 完成子网组部分:

   - 适用于子网组，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。如果你正在创建一个新的：

     - 输入名称

- 输入说明
- 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 (p. 228)。
- 如果您正在创建新的子网组，但没有现有 VPC，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. 适用于Redis 版本兼容性，接受默认为`6.0`.

   b. 适用于端口，接受默认 Redis 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. 适用于参数组，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 Redis 特定的参数 (p. 162)。

   d. 适用于节点类型，为所需节点类型（及其关联的内存大小）选择一个值。

   e. 适用于分片数量，为此集群选择所需的分区数量。

      您可以动态更改集群中的分区数量。有关更多信息，请参阅 扩展 MemoryDB 集群 (p. 135)。

   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 Next（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。您可以为 VPC 使用默认安全组或创建新安全组。

         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

           您可以通过选择 Customer Managed Amazon KMS key（客户托管式 Amazon KMS key）并选择该密钥来提供不同的加密密钥。

         - Encryption in-transit（传输中加密）– 对传输中的数据启用加密。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 (ACL) 验证用户 (p. 172)。

      iii. 适用于快照（可选）指定快照保留期和快照时段。默认为启用自动快照被选中。

      iv. 适用于维护时段（可选）指定维护时段。这些区域有：维护时段是每周 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 为您的维护时段选择日期和时间 [NCE（无首选项），或者您可以自行选择日期、时间和持续时间 [NCE（指定维护时段）。如果您在列表中选择 Specify maintenance window，则为您的维护时段选择 Start day、Start time 和 Duration（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 (p. 91)。

      v. 对于 Notifications（通知），选择现有 Amazon Simple Notification Service (Amazon SNS) 主题，或选择“Manual ARN input（手动 ARN 输入）”，然后输入主题的 Amazon Resource

Name (ARN)。Amazon SNS 允许将通知推送到与 Internet 连接的智能设备。默认设置是禁用通知。有关更多信息，请参阅 。https://aws.amazon.com/sns/.

i. 适用于标签，您可以有选择地应用标签来搜索和筛选集群或跟踪Amazon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 Create cluster (创建集群) 启动集群或选择 Cancel (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 第 2 步：授予集群的访问权限 (p. 17) 和 第 3 步：连接到集群 (p. 18)。

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 第 4 步：删除集群 (p. 19)。

从快照还原 (AmazonCLI)

当使用任一`create-cluster`操作，请务必包含参数`--snapshot-name`要么`--snapshot-arns`使用快照中的数据为新集群做种时使用。

有关更多信息，请参阅下列内容：

- 创建集群 (AmazonCLI) (p. 15)中的内存 DB 用户指南.
- create-cluster中的Amazon CLI命令参考。

从快照还原（MemoryDB API）

您可以使用 MemoryDB API 操作还原 MemoryDB 快照`CreateCluster`.

使用`CreateCluster`操作，请务必包含参数`SnapshotName`要么`SnapshotArns`使用快照中的数据为新集群做种时使用。

有关更多信息，请参阅下列内容：

- 创建集群 (MemoryDB API) (p. 15)中的内存 DB 用户指南.
- CreateCluster中的MemoryDB API 参考.

使用外部创建的快照为新集群设定种子

创建新 MemoryDB 集群时，可以使用 Redis .rdb 快照文件中的数据为其做种。

要从 MemoryDB 快照为新的 MemoryDB 集群播种，或者 ElastiCache 有关 Redis 快照，请参阅从快照还原 (p. 125).

使用 Redis .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。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 Redis 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，请参阅特定于 MemeryDB 节点类型的参数 (p. 167).
  - 您只能使用 Amazon S3 服务器端加密 (SSE-S3) 对 Redis .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。

第 1 步：在外部集群上创建 redis 快照

创建快照以播种 MemoryDB 集群

1. 连接到您的现有 Redis 实例。
2. 运行 Redis`BGSAVE`要么`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 Redis 网站上的 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 Redis 网站上的 SAVE。

有关创建快照的其他信息，请参阅Redis 持久性在 Redis 网站上查看。

第 2 步：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 第 3 步：将您的快照上载到 Amazon S3 (p. 129)。

创建 Amazon S3 存储桶

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Simple Storage Service（Amazon S3）控制台：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）。

我们强烈建议您在相同的 Amazon S3 存储桶中创建Amazon区域作为您的新 MemoryDB 集群。此方式将确保当 MemoryDB 从 Amazon S3 读取 .rdb 文件时，数据传输速度最快。

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. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Simple Storage Service（Amazon S3）控制台：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Save（保存）。

记录存储桶名称和文件夹名称。

第 3 步：将您的快照上载到 Amazon S3

现在，上传您在第 1 步：在外部集群上创建 redis 快照 (p. 128)中创建的 .rdb 文件。将其上传到您在 第 2 步：创建 Amazon S3 存储桶和文件夹 (p. 128) 中创建的 Amazon S3 存储桶和文件夹。有关此任务的更多信息，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. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Simple Storage Service（Amazon S3）控制台：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. 请选择 Upload（上传）。
5. 选择 Add files。
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 Amazon 中列出了正确的文件上传页面，然后选择上传.

记下 .rdb 文件的路径。例如，如果存储桶名称为 `myBucket` 并且路径为 `myFolder/redis.rdb`，请输入 `myBucket/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，请参阅存储桶命名规则中的Amazon Storage Service.

第 4 步：授予 MemoryDB 对 .rdb 文件的读取访问权限

默认情况下，在 2019 年 3 月 20 日之前推出的 Amazon 区域为已启用状态。您可以立即开始在这些 Amazon 区域中工作。2019 年 3 月 20 日之后推出的区域默认情况下处于禁用状态。您必须按照所述，先启用或选择加入这些区域，然后才能使用它们。管理Amazon地区.

授予 MemoryDB 对 .rdb 文件的读取访问权限

授予 MemoryDB 对快照文件的读取访问权限

1. 登录到 Amazon Web Services Management Console，然后通过以下网址打开 Simple Storage Service（Amazon S3）控制台：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. 请选择 Permissions 选项卡。
6. UNDERPermissions (权限)，选择存储桶策略然后选择编辑.
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。

```
{
    "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:::example-bucket",
                "arn:aws:s3:::example-bucket/snapshot1.rdb",
                "arn:aws:s3:::example-bucket/snapshot2.rdb"
            ]
        }
    ]
}
```

8. 选择Save（保存）。

第 5 步：通过 .rdb 文件数据为 MemoryDB 集群设定种子

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，请按照中的说明操作创建内存数据库集群 (p. 13).

您用来告知 MemoryDB 在何处查找已上传到 Amazon S3 的 Redis 快照的方法取决于您创建集群时采用的方法：

通过 .rdb 文件数据为 MemoryDB 集群设定种子

- 使用 MemoryDB 控制台

  选择 Redis 引擎之后，展开 Advanced Redis settings 部分，然后找到 Import data to cluster。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 `myBucket/myFolder/myBackupFilename.rdb`。

- 使用 Amazon 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:::myBucket/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

  如果您将`CreateCluster`或者`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:::myBucket/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AmazonMemoryDB 命令行界面或 MemoryDB API，用于获取事件消息。

为快照添加标签

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 (p. 84)。

成本分配标签是一种通过按标签值对发票上的费用进行分组来跨多种 Amazon 服务跟踪成本的方式。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，Amazon CLI或 MemoryDB API，您可以在快照上添加、列出、修改、删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 (p. 86)。

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它们的唯一方法。

您可以使用 MemoryDB 控制台Amazon CLI，或 MemoryDB API。

删除快照 (控制台)

以下过程演示使用 MemoryDB 控制台删除快照。

如何删除快照

1. 登录到Amazon Web Services Management Console然后打开适用于 Redis 的 MemoryDB 控制台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，其中包含您的快照的列表。
3. 选择要删除的快照名称左侧的单选按钮。
4. 选择 Actions，然后选择 Delete 。
5. 如果要删除此快照，请输入`delete`在文本框中，然后选择Delete. 要取消删除，请选择Cancel. 状态将变为正在删除。

删除快照 (AmazonCLI)

使用删除快照Amazon CLI使用以下参数删除快照。

- `--snapshot-name`— 要删除的快照的名称。

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 Amazon CLI 命令参考中的 delete-snapshot。

删除快照（内存数据库 API）

使用`DeleteSnapshot`使用以下参数删除快照。

- `SnapshotName`— 要删除的快照的名称。

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，这种需求高峰将产生很高的 通过使用 MemyDB，您可以扩展以满足当前需求，而只需支付所用的项目。

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

操作	MemoryDB	
横向扩展	MemoryDB 的在线重新分片和分片重新平衡 (p. 136)	
更改节点类型	通过修改节点类型来在线纵向扩展 (p. 143)	
更改分片数量	扩展 MemoryDB 集群 (p. 135)	

扩展 MemoryDB 集群

由于对集群的需求发生变化，您可能决定通过更改 Memory 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存在内存压力：FreeableMemory、SwapUsage, 和用于存储数据库的字节.
- CPU 或网络瓶颈：

  如果延迟/吞吐量问题给您的集群带来麻烦，您可能需要进行横向扩展来解决这些问题。

  您可以通过监控以下指标来监控您的延迟和吞吐量水平：CPU 使用率、NetworkBytesIn、NetworkBytesOut、当前连接, 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，以便确定是否可以安全地进行缩减：FreeableMemory、SwapUsage、用于存储数据库的字节、CPU 使用率、NetworkBytesIn、NetworkBytesOut、当前连接, 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 MemyDB 集群：横向和纵向扩展。

- 通过添加或删除分片，您可以通过添加或删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时进行缩减/扩展。
- 纵向扩展 – 更改节点类型以调整集群大小。在线纵向扩展允许在集群继续处理传入请求时进行扩展/缩减。

如果要通过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 Redis 开销。

MemoryDB 的离线重新分片和分片重新平衡

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。

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。

要离线重新配置分片 MemyDB 集群

1. 创建现有 MemyDB 集群的手动快照。有关更多信息，请参阅手动创建快照 (p. 111)。
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 (p. 125)。
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 (p. 39)。

MemoryDB 的在线重新分片和分片重新平衡

通过对 Memory Db 使用在线重新分片和分片重新平衡，您可以在无需停机的情况下动态扩展 Memory DB。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：

- 扩展— 通过向 MemyDB 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。
- 扩展— 通过删除 Memory 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 MemyDB 在线重新分片：

- 槽或密钥空间和大型项目存在以下限制：

  如果分片中的任何密钥包含一个大型项，在横向扩展或重新平衡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：在线集群大小调整 (p. 95)。

您可以使用水平扩展或重新平衡 MemyDB 集群。Amazon Web Services Management Console，Amazon CLI，以及 MemoryDB API。

通过在线重新分区功能添加分区

您可以使用将分片添加到 MemyDB 集群中。Amazon Web Services Management Console、Amazon CLI，或 MemoryDB API。

添加分区（控制台）

您可以使用Amazon Web Services Management Console将一个或多个分片添加到 MemyDB 集群中。以下步骤描述了这个过程。

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB of Redis 控制台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上，选择添加/删除分片
4. In分片数量新，输入所需分片的数量。
5. 选择确认保留更改或Cancel丢弃。

添加分区 (Amazon CLI)

以下过程介绍了如何通过使用添加分片的方法重新配置 Memory Db 集群中的分片。Amazon CLI.

在update-cluster中使用以下参数：

参数

- --cluster-name – 必需。指定在哪个集群（集群）上执行分片重新配置操作。
- --shard-configuration – 必需。允许您设置分片数量。
  - ShardCount— 设置此属性以指定所需分片的数量。

Example

以下示例修改集群中的分片数。my-cluster到 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
```

它返回以下 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": "6.2.4",
        "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",
        "AutoMinorVersionUpgrade": true
    }
}
```

要在更新集群的状态从更改后，查看更新集群的详细信息更新到可用，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
```

它将返回下面的 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": "6.2.4",
            "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",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

有关更多信息，请参阅 。更新集群中的Amazon CLI命令参考。

添加分片（MemoryDB API）

您可以通过使用 Memory DB API 在线重新配置 Memory 集群中的分片。`UpdateCluster`operation.

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。允许您设置分片数量。
  - `ShardCount`— 设置此属性以指定所需分片的数量。

有关更多信息，请参阅 。UpdateCluster.

通过在线重新分区功能删除分区

您可以使用从 MemyDB 集群中删除分片Amazon Web Services Management Console、Amazon CLI，或MemoryDB API。

删除分区（控制台）

以下过程介绍了如何通过使用删除分片的方法重新配置 Memory Db 集群中的分片。Amazon Web Services Management Console.

Important

在从集群中删除分片之前，Memory DB 可确保所有数据将适合其余分片。如果数据适合，分片将根据要求从集群中删除分片。如果数据不适合剩余分片，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Amazon Web Services Management Console以从 MemyDB 集群中删除一个或多个分片。您无法删除集群中的所有分片。而是必须删除集群。有关更多信息，请参阅第 4 步：删除集群 (p. 19)。以下过程介绍删除一个或多个分片的过程。

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB of Redis 控制台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上，选择添加/删除分片
4. In分片数量新中，输入你想要的分片数量（至少有 1 个）。
5. 选择确认保留更改或Cancel丢弃。

删除分区 (Amazon CLI)

以下过程介绍了如何通过使用删除分片的方法重新配置 Memory Db 集群中的分片。Amazon CLI.

Important

在从集群中删除分片之前，Memory DB 可确保所有数据将适合其余分片。如果数据将适合，分片将根据要求从集群中删除，并将其密钥空间映射到其余分片。如果数据不适合剩余分片，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Amazon CLI以从 MemyDB 集群中删除一个或多个分片。您无法删除集群中的所有分片。而是必须删除集群。有关更多信息，请参阅第 4 步：删除集群 (p. 19)。

在`update-cluster`中使用以下参数：

参数

- `--cluster-name` – 必需。指定在哪个集群（集群）上执行分片重新配置操作。
- `--shard-configuration` – 必需。允许您使用设置分片的数量。`ShardCount`属性：

  `ShardCount`— 设置此属性以指定所需分片的数量。

Example

以下示例修改集群中的分片数。`my-cluster`到 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
```

它返回以下 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": "6.2.4",
        "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",
        "AutoMinorVersionUpgrade": true
    }
}
```

要在更新集群的状态从更改后，查看更新集群的详细信息更新到可用，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
```

它将返回下面的 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": "6.2.4",
            "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",
            "AutoMinorVersionUpgrade": true
        }
    ]
}
```

有关更多信息，请参阅 。更新集群中的Amazon CLI命令参考。

删除分片（MemoryDB API）

您可以通过使用 Memory DB API 在线重新配置 Memory 集群中的分片。`UpdateCluster`operation.

以下过程介绍了如何通过使用 Memory DB API 删除分片的方法重新配置 Memory 集群中的分片。

> **Important**
>
> 在从集群中删除分片之前，Memory DB 可确保所有数据将适合其余分片。如果数据将适合，分片将根据要求从集群中删除，并将其密钥空间映射到其余分片。如果数据不适合剩余分片，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 MemyDB API 从 MemyDB 集群中删除一个或多个分片。您无法删除集群中的所有分片。而是必须删除集群。有关更多信息，请参阅第 4 步：删除集群 (p. 19)。

在`UpdateCluster`中使用以下参数：

参数

- `ClusterName` – 必需。指定在哪个集群（集群）上执行分片重新配置操作。
- `ShardConfiguration` – 必需。允许您使用设置分片的数量。`ShardCount`属性：

  `ShardCount`— 设置此属性以指定所需分片的数量。

通过修改节点类型来在线纵向扩展

通过对 Memory DB 使用在线纵向扩展，您可以在最短的停机时间的情况下动态扩展集群。这样，即使在扩展时，集群也可以处理请求。

您可执行以下操作：

- 纵向扩展— 通过调整 Memory 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。

  MemyDB 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，MemyDB 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。要确保平滑的扩展/缩减流程，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

扩展 MemoryDB 集群（控制台）

以下过程介绍如何使用扩展 MemyDB 集群。Amazon Web Services Management Console. 在此过程中，MemyDB 集群将继续处理请求，且停机时间降至最短。

扩展集群（控制台）

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB of Redis 控制台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：
   - 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 Save changes（保存更改）。

   集群的状态将变为修改的. 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 (AmazonCLI)

以下过程介绍如何使用扩展 MemyDB 集群。Amazon CLI. 在此过程中，MemyDB 集群将继续处理请求，且停机时间降至最短。

要扩展 MemoryDB 集群（AmazonCLI)

1. 通过运行带以下参数的 Amazon CLI `list-allowed-node-type-updates` 命令，确定您可向上扩展到的节点类型。

   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 。列表允许的节点类型更新中的Amazon CLI参考.

2. 使用修改集群以扩展为较大的新节点类型。Amazon CLI `update-cluster`命令和以下参数。

   - `--cluster-name`— 要向上扩展为的集群的名称。
   - `--node-type`— 要将集群扩展为的新节点类型。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 。更新集群.

扩展 MemoryDB 集群（MemoryDB API）

以下过程使用 Memory DB 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，MemyDB 会更新 DNS 条目，使其指向新的节点。您可以在集群继续保持在线并处理传入请求时扩展启用自动故障转移的集群。

向上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。

向上扩展 MemoryDB 集群（MemoryDB API）

1. 使用 MemyDB API 确定您可纵向扩展为的节点类型。`ListAllowedNodeTypeUpdates`具有以下参数的操作。

   - `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 。列表允许节点类型更新中的用于 Redis API 参考的 MemoryDB.

2. 使用将当前集群向上扩展为新的节点类型。`UpdateCluster`MemyDB API 操作和以下参数。

   - `ClusterName`— 集群的名称。
   - `NodeType`— 此集群中的集群较大的新节点类型。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

- 缩小 MemoryDB 集群（控制台）(p. 146)
- 缩小 MemoryDB 集群 (AmazonCLI) (p. 146)
- 缩小 MemoryDB 集群（MemoryDB API）(p. 147)

缩小 MemoryDB 集群（控制台）

以下过程介绍如何使用缩减 MemyDB 集群。Amazon Web Services Management Console. 在此过程中，Memory 集群将继续处理请求，且停机时间降至最短。

缩减 MemoryDB 集群

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB of Redis 控制台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：
   - 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 Save changes（保存更改）。

   集群的状态将变为修改的. 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 (AmazonCLI)

以下过程介绍如何使用缩减 MemyDB 集群。Amazon CLI. 在此过程中，Memory 集群将继续处理请求，且停机时间降至最短。

要缩减 MemoryDB 集群（AmazonCLI)

1. 通过运行带以下参数的 Amazon CLI `list-allowed-node-type-updates` 命令，确定您可缩减为的节点类型。

   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 。列表允许的节点类型更新.

2. 使用修改集群以缩减为较小的新节点类型。`update-cluster`命令和以下参数。

   - `--cluster-name`— 要缩减为的集群的名称。
   - `--node-type`— 要将集群扩展为的新节点类型。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 。更新集群.

缩小 MemoryDB 集群（MemoryDB API）

以下过程使用 Memory DB 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，Memory 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不同而异。

缩小（MemoryDB API）

1. 使用确定您可缩减为的节点类型。列表允许节点类型更新使用以下参数 API：

   - `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. 使用将当前集群向下扩展为新的节点类型。UpdateClusterAPI 具有以下参数。

   - `ClusterName`— 集群的名称。
   - `NodeType`— 此集群中的集群较小的新节点类型。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

Redis MemyDB 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Redis 特定的参数 (p. 162)。

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，请参阅特定于 MemeryDB 节点类型的参数 (p. 167)。

主题

参数管理

参数已分组到指定的参数组中，以便更轻松地管理参数。参数组表示在启动期间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，则必须创建自定义参数组。
- 参数组系列与您分配给参数组的集群必须兼容。例如，如果您的集群运行 Redis 版本 6，您只能使用 Memydb_redis6 系列中的参数组 (默认或自定义)。
- 当您更改集群的参数时，更改将立即应用于集群。无论是更改集群的参数组本身还是更改集群参数组中的参数值，都是如此。

参数组层

Redis 参数组层

全局默认值

用于区域中 Redis 客户的所有 MemoryDB 的顶级根参数组。

全局默认参数组：

- 预留供 MemroyDB 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 MemroyDB 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。客户创建参数组时创建客户拥有的参数组。

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。
- 客户可以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 Redis 特定的参数 (p. 162)。

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 MemroyDB 控制台、Amazon CLI，或 MemoryDB API。

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 MemroyDB 控制台创建参数组

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB for Redis 控制台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，请选择创建参数组.

   这些区域有：创建参数组此时将显示页。
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Redis 版本兼容性框中，选择此参数组对应的引擎版本。
7. 在标签，可选择添加标签以搜索和筛选参数组或跟踪Amazon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 (p. 158)。

创建参数组 (AmazonCLI)

要使用 Amazon CLI 创建参数组，请使用带以下参数的命令 `create-parameter-group`。

- `--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

以下示例创建一个名为的参数组myredis6x使用 memorydb_redis6 系列作为模板。

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 (p. 158)。

有关更多信息，请参阅 `create-parameter-group`。

创建参数组（MemroyDB API）

要使用 MemroyDB API 创建参数组，请使用`CreateParameterGroup`使用这些参数进行操作。

- `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

以下示例创建一个名为的参数组myredis6x使用 memorydb_redis6 系列作为模板。

```
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 (p. 158)。

有关更多信息，请参阅 CreateParameterGroup。

按名称列出参数组

您可以使用 MemroyDB 控制台、Amazon CLI，或 MemoryDB API。

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 MemroyDB 控制台列出参数组

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB for Redis 控制台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 (AmazonCLI)

要使用 Amazon CLI 生成参数组的列表，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参数组myredis6x对于在 Redis 引擎 5.0.6 和更高版本上运行的参数组。

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 MemroyDB API 生成参数组的列表，请使用DescribeParameterGroupsaction. 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>
```

此操作所得到的响应将类似如下所示，列出每个参数组的名称、系列和描述，以及在 memorydb_redis6 的情况下。

```
<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 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 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 控制台列出参数组的参数及其值。Amazon CLI，或 MemoryDB API。

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. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB for Redis 控制台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 (不是旁边的框)，选择要列出其中包含的参数和值的参数组。

   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 (AmazonCLI)

要使用 Amazon CLI 列出参数组的参数及其值，请使用命令 `describe-parameters`。

Example

以下示例代码列出了参数组的所有参数及其值。myredis6x.

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（MemroyDB API）

要使用 MemoryDB API 列出参数组的参数及其值，请使用`DescribeParametersaction`.

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Redis 特定的参数 (p. 162)。

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 MemroyDB 控制台更改参数值

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB for Redis 控制台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，选择要修改的参数组。

   选择操作然后查看详细信息. 或者，您也可以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. 将启用所有可编辑的参数以进行编辑。您可能需要跨页面移动才能找到要更改的参数。或者，您可以按名称、值或在搜索框中的类型搜索参数。
5. 对参数进行任何必需的修改。
6. 要保存您的更改，请选择 Save Changes (保存更改)。
7. 如果您在页数中修改了参数值，则可以通过选择以下方法查看所有更改预览更改. 要确认更改，请选择保存更改. 要进行更多修改，请选择背部.
8. 这些区域有：参数详情页面还为您提供了重置为默认值的选项。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。您可以选择要重置的那个，然后选择继续重置以确认。

   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在每个页面上查看的参数数量。使用右侧的齿轮进行这些更改。您还可以在详细信息页面上启用/禁用所需的列。这些更改持续在控制台的会话中进行。

   要查找您更改的参数名称，请参阅Redis 特定的参数 (p. 162)。

修改参数组 (AmazonCLI)

要使用 Amazon CLI 更改参数值，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Redis 特定的参数 (p. 162)

有关更多信息，请参阅 。update-parameter-group.

修改参数组（ MemroyDB API ）

要使用 MemoryDB API 更改参数组的参数值，请使用`UpdateParameterGroup`action.

要查找您要更改的参数名称和允许的值，请参阅Redis 特定的参数 (p. 162)

有关更多信息，请参阅 `UpdateParameterGroup`。

删除参数组

您可以使用 MemroyDB 控制台、Amazon CLI，或 MemoryDB API。

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 MemroyDB 控制台删除参数组

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB for Redis 控制台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，选择要删除的参数组。

   选择 Actions，然后选择 Delete 。
4. Delete Parameter Groups 确认屏幕随即出现。
5. 要删除参数组，请输入Delete在确认文本框中。

   要保留参数组，请选择 Cancel。

删除参数组 (AmazonCLI)

要使用 Amazon CLI 删除参数组，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 MemroyDB API 删除参数组，请使用`DeleteParameterGroup`action. 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。

Redis 特定的参数

如果您没有为 Redis 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 (p. 150)。

主题

Redis 6 参数

参数组系列：memorydb_redis6

Redis 6 中添加的参数如下所示。

名称	详细信息	说明
`maxmemory-policy`	类型: STRING 允许的值：volatile-lru、allkeys-lru、volatile-lfu、allkeys-lfu、挥发性随机、ALLKEYS-随机、volatile-ttl、noeviction 默认值：noeviction	达到最大内存使用率时密钥的移出策略。 有关更多信息，请参阅将 Redis 用作 LRU 缓存。将 Redis 用作 LRU 缓存.
`list-compress-depth`	类型: INTEGER 允许的值：—0 默认值：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： • ：禁用所有压缩。 • 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
`hll-sparse-max-bytes`	类型: INTEGER 允许的值：1-16000 默认值：3000	HyperLog日志稀疏表示形式字节限制。限制包括 16 个字节的标头。当HyperLog使用稀疏表示形式的日志超出此限制，它会转换成密集表现形式。 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，同时不会降低效率。`PFADD`太多了，这是 O (N) 使用稀疏编码。如果不存在 CPU 问题，但空间

名称	详细信息	说明
		较为严重，且数据集由许多组成，则可将该值提高到 10000 左右。HyperLog基数在 0-15000 范围内的日志。
lfu-log-factor	类型: INTEGER 允许的值：–1 默认值：10	用于增加 LFU 驱逐策略的密钥计数器的日志因子。
lfu-decay-time	类型: INTEGER 允许的值：—0 默认值：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: INTEGER 允许的值：1-1000000 默认值：1000	在活动碎片整理过程中，将从主字典扫描中处理的最大 sethash/zset/list 字段数。
active-defrag-threshold-upper	类型: INTEGER 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: INTEGER 允许的值：—0 默认值：33554432	对于 Redis 发布/订阅客户端：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。
client-output-buffer-limit-pubsub-soft-limit	类型: INTEGER 允许的值：—0 默认值：8388608	对于 Redis 发布/订阅客户端：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时间时。client-output-buffer-limit-pubsub-soft-seconds.
client-output-buffer-limit-pubsub-soft-seconds	类型: INTEGER 允许的值：—0 默认值：60	对于 Redis 发布/订阅客户端：如果客户端的输出缓冲区保持 client-output-buffer-limit-pubsub-soft-limit 字节的时间长于此秒数，则客户端将断开连接。
timeout	类型: INTEGER 允许的值：0,20- 默认值：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。

名称	详细信息	说明
notify-keyspace-events	类型: STRING 允许的值：NULL 默认值：NULL	Redis 用于通知 Pub/Sub 客户关于的密钥空间事件。默认情况下，禁用所有通知。
maxmemory-samples	类型: INTEGER 允许的值：–1 默认值：3	适用于least-recently-used (LRU)和time-to-live (TTL)计算，此参数表示要检查的密钥的采样大小。默认情况下，Redis 选择 3 个密钥并使用最近最少使用的一个密钥。
slowlog-max-len	类型: INTEGER 允许的值：—0 默认值：128	Redis 慢速日志的最大长度。这个长度没有限制。请注意，它会消耗内存。你可以使用回收慢日志使用的内存SLOWLOG RESET.
activerehashing	类型: STRING 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: INTEGER 允许的值：—0 默认值：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: INTEGER 允许的值：—0 默认值：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）。
client-output-buffer-limit-normal-soft-seconds	类型: INTEGER 允许的值：—0 默认值：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: INTEGER 允许的值：—0 默认值：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: INTEGER 允许的值：1-75 默认值：5	用于碎片整理的最少精力，以 CPU 百分比为单位。

名称	详细信息	说明
stream-node-max-bytes	类型: INTEGER 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: INTEGER 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。
lazyfree-lazy-eviction	类型: STRING 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: INTEGER 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: STRING 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: INTEGER 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: INTEGER 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: STRING 允许的值：Yes、no 默认值：no	对更新值的命令执行异步删除。
slowlog-log-slower-than	类型: INTEGER 允许的值：—0 默认值：10000	Redis 记录命令的最大执行时间（单位：微秒）。Slow Log功能。请注意，负数禁用慢日志，而值为零则强制记录每个命令。
hash-max-ziplist-entries	类型: INTEGER 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。

名称	详细信息	说明
hash-max-ziplist-value	类型: INTEGER 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: INTEGER 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: INTEGER 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: INTEGER 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。
tracking-table-max-keys	类型: INTEGER 允许的值：1-100000000 默认值：1000000	为了帮助客户端缓存，Redis 支持跟踪哪些客户端访问了哪些密钥。 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: INTEGER 允许的值：1-10000 默认值：128	ACL 日志中的最大条目数。
active-expire-effort	类型: INTEGER 允许的值：1-10 默认值：1	Redis 会通过两种机制删除超过密钥自身存活时间的密钥。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 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: STRING 允许的值：Yes、no 默认值：no	指定的默认行为DEL命令的行为与UNLINK.

名称	详细信息	说明
activedefrag	类型: STRING 允许的值：Yes、no 默认值：no	已启用活动内存碎片整理。

特定于 MemeryDB 节点类型的参数

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maxmemory对于每种节点类型。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.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

AutoryDB 所有实例类型必须在 Amazon Virtual Private Cloud VPC 中创建。

内存中的安全性 For Redis

Amazon 十分重视云安全性。作为 Amazon 客户，您将从专为满足大多数安全敏感型企业的要求而打造的数据中心和网络架构中受益。

安全性是Amazon和您的共同责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云的安全性 – Amazon 负责保护在 Amazon 云中运行 Amazon 服务的基础设施。Amazon 还向您提供可安全使用的服务。第三方审核员定期测试和验证我们的安全性的有效性，作为 Amazon Compliance Programs 的一部分. 要了解适用于的合规性计划，请参阅适用于MemoryDB的合规性计划，请参阅Amazon合规性计划范围内的服务.
- 云中的安全性 - 您的责任由您使用的 Amazon 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您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 for Redis 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。您还将了解如何使用其他Amazon服务。帮助您监控和保护您的服务。

目录

Redis 中的 MemoryDB 中的数据保护

这些区域有：Amazon 责任共担模式适用于中的数据保护。如该模式中所述，Amazon 负责保护运行所有 Amazon Web Services 云 的全球基础设施。您负责维护对托管在此基础设施上的内容的控制。此内容包括您所使用的Amazon Web Services的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。

出于数据保护目的，我们建议您保护 Amazon Web Services 账户 凭证并使用 Amazon Identity and Access Management (IAM) 设置单独的用户账户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护您的数据：

- 对每个账户使用 Multi-Factor Authentication (MFA)。
- 使用 SSL/TLS 与 Amazon 资源进行通信。建议使用 TLS 1.2 或更高版本。
- 使用 Amazon CloudTrail 设置 API 和用户活动日志记录。
- 使用 Amazon 加密解决方案以及 Amazon 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Simple Storage Service（Amazon S3）中的个人数据。
- 如果在通过命令行界面或 API 访问 Amazon 时需要经过 FIPS 140-2 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准 (FIPS) 第 140-2 版》。

我们强烈建议您切勿将机密信息或敏感信息（例如您客户的电子邮件地址）放入标签或自由格式字段（例如名称字段）。这包括使用或其他时间Amazon使用控制台、API、Amazon CLI，或者Amazon开发工具包。

您在用于名称的标签或自由格式字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，我们强烈建议您不要在 URL 中包含凭证信息来验证您对该服务器的请求。

Redis MemoryDB 中的数据安全

为了帮助确保数据安全，MemoryDB in Redis 和 Amazon EC2 提供了禁止未经授权来访问服务器上数据的机制。

MemoryDB 还为集群上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 (ACL) 验证用户 (p. 172)为了控制用户对您的集群的访问。

主题

MemoryDB 中的静态加密

为了帮助保护您的数据，MemoryDB for Redis 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 MoryDB 和Amazon VPC (p. 221) 和 Redis MemoryDB 中的身份和权限管理 (p. 180)。

始终启用 MemoryDB 静态加密，以通过加密持久数据来提高数据的安全性。它加密了以下几个方面：

- 事务日志中的数据
- 同步、快照和交换操作期间的磁盘
- Amazon S3 中存储的快照

MemoryDB 提供默认（服务管理）的静态加密，以及在中使用您自己的对称的客户管理的客户根密钥的能力。AmazonKey Management Service (KMS).

有关传输中加密的信息，请参阅传输中加密 (TLS) (p. 171)

主题

- 使用客户管理的密钥Amazon自杀 (p. 170)
- 另请参阅 (p. 171)

使用客户管理的密钥Amazon自杀

MemoryDB 支持对称的客户管理的根密钥（KMS KMS Key）进行静态加密。客户托管式 KMS 密钥是您在自己的 Amazon 账户中创建、拥有并管理的加密密钥。有关更多信息，请参阅 。客户根密钥中的Amazon键管理服务开发人员指南. 密钥必须在AmazonKMS 之前可以将其与 MemoryDB 一起使用。

要了解如何创建 Amazon KMS 根密钥，请参阅 Amazon Key Management Service 开发人员指南中的创建密钥。

MemoryDB 允许您与AmazonKMS. 有关更多信息，请参阅 Amazon 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AmazonKMS.

这些区域有：`kms:ViaService`条件密钥限制使用Amazon指定请求的 KMS 密钥Amazon服务。使用`kms:ViaService`使用 MemoryDB，在条件键值中包含两个 ViaService 名称：`memorydb.amazon_region.amazonaws.com`. 有关更多信息，请参阅 。kms:ViaService.

您可以使用AmazonCloudTrail跟踪 MemoryDB to Redis 发送到的请求Amazon Key Management Service代表您。对 Amazon Key Management Service 发出的与客户托管式密钥相关的所有 API 调用都具有相应的 CloudTrail 日志。您还可以通过调用ListGrantsKMS API 调用。

使用客户托管密钥加密集群后，集群的所有快照都将按如下方式加密：

- 使用与集群关联的客户管理的密钥对自动每日快照进行加密。
- 删除集群时创建的最终快照也使用与集群关联的客户管理的密钥进行加密。
- 默认情况下，手动创建的快照将使用与集群关联的 KMS 密钥。您可以通过选择其他客户托管式密钥来覆此行为。
- 默认情况下，复制快照使用与源快照关联的客户管理的密钥。您可以通过选择其他客户托管式密钥来覆此行为。

> Note
>
> - 将快照导出到所选的 Amazon S3 存储桶时，无法使用客户管理的密钥。但是，导出到 Amazon S3 的所有快照都使用服务器端加密。您可以选择将快照文件复制到新的 S3 对象并使用客户管理

的 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。

- 您还可以使用客户管理的密钥对手动创建的未使用客户管理的密钥进行加密的快照进行加密。使用此选项，即使未在原始群集上加密数据，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。

- 如果你删除密钥或禁用键和撤消授权对于用于加密集群的密钥，集群将变得无法恢复。换句话说，复制组在硬件故障后无法修改或恢复。AmazonKMS 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管理的密钥创建快照以用于存档目的。
- 自动密钥轮换可保留您的属性AmazonKMS 根密钥，因此轮换不会影响您访问 MemoryDB 数据的能力。加密的 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更新对旧密钥的任何引用。要了解更多信息，请参阅轮换客户根密钥中的Amazon键管理服务开发人员指南.
- 使用 KMS 密钥加密 MemoryDB 集群需要每个集群一次授权。在集群的整个生命周期中使用此授权。此外，在创建快照期间使用每个快照一个授权。创建快照后，此授权将停用。
- 有关AmazonKMS 授权和限制，请参阅配额中的Amazon键管理服务开发人员指南.

另请参阅

- 传输中加密 (TLS) (p. 171)
- MoryDB 和Amazon VPC (p. 221)
- Redis MemoryDB 中的身份和权限管理 (p. 180)

传输中加密 (TLS)

为了帮助确保数据安全，MemoryDB in Redis 和 Amazon EC2 提供了禁止未经授权来访问服务器上数据的机制。通过传输中加密功能，MemoryDB 为您提供了在从一个位置之间移动数据时用来保护数据的工具。例如，您可能从群集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

MemoryDB in Redis 传输中加密是一项功能，可以提高数据在最易受攻击点的安全性，当数据从一个位置传输到另一个位置时。

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均采用安全套接字层 (SSL) 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。

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 (p. 18).

另请参阅

- MemoryDB 中的静态加密 (p. 170)

使用访问控制列表 (ACL) 验证用户

您可以使用访问控制列表 (ACL) 对用户进行身份验证。

ACL 使您能够通过对用户进行分组来控制群集访问。这些访问控制列表旨在作为一种组织对集群的访问权限的方式。

如下一节所述，您可以使用访问字符串创建用户并为其分配特定权限。您可以将用户分配给与特定角色（管理员、人力资源）匹配的访问控制列表，然后将这些用户部署到一个或多个 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。

ACL 旨在支持引入Redis ACL在 Redis 6 中。在将 ACL 与 MemoryDB 集群一起使用时，有一些限制。

- 不能在访问字符串中指定密码。你用设置密码CreateUser要么UpdateUser调用。
- 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均未指定，则将为用户分配 1`off`并且没有对集群的访问权限。
- 您不能使用已禁止的命令。如果您指定了已禁止的命令，则会引发异常。有关这些命令的列表，请参阅受限 Redis 命令 (p. 93).
- 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 管理密码。因此，您不能使用 `reset`，因为此命令会移除用户的所有密码。
- Redis 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。支持内存 DB`ACL LIST`命令，但不像 Redis 那样支持哈希密码。使用 MemoryDB，您可以使用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不会检索用户密码。

  MemoryDB 支持的其他只读命令包括ACL WHOAMI、ACL 用户, 和ACL 猫. MemoryDB 不支持任何其他基于写入的 ACL 命令。

下面将更详细地介绍如何将更详细地介绍如何将为 MemoryDB

主题

使用访问字符串指定权限

要指定对 MemoryDB 集群的权限，请使用Amazon CLI要么Amazon Web Services Management Console.

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。
- `+@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#” 开头的密钥进行读取访问 键空间

```
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 (p. 173)。

如果要将现有工作负载迁移到 MemoryDB，则可以通过调用来检索访问字符串。`ACL LIST`，不包括用户和任何密码哈希值。

将 ACL 应用于 MemoryDB 的集群

要使用 MemoryDB ACL，请执行以下步骤：

1. 创建一个或多个用户。
2. 创建 ACL 并将用户添加到列表中。
3. 将 ACL 分配给集群。

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

ACL 用户的用户信息为用户名以及可选的密码和访问字符串。访问字符串提供对密钥和命令的权限级别。此名称对于用户是唯一的，也是传递给引擎的内容。

确保您提供的用户权限符合 ACL 的预期目的。例如，如果您创建了一个名为`Administrators`，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。对于中的用户`e-commerce`ACL，您可以将其访问字符串设置为只读访问。

MemoryDB 使用用户名自动为每个账户配置默认用户`"default"`. 除非明确添加到 ACL 中，否则它不会与任何集群关联。您无法修改或删除该用户。该用户旨在与以前 Redis 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。

将为包含默认用户的每个账户创建不可变的 “开放访问” ACL。这是默认用户可加入的唯一 ACL。创建集群时，必须选择要与该集群关联的 ACL。虽然您确实可以选择对默认用户应用 “开放访问” ACL，但我们强烈建议您创建具有权限限限制于其业务需求的用户的 ACL。

未启用 TLS 的群集必须使用 “开放访问” ACL 来提供开放式身份验证。

可以在没有用户的情况下创建 ACL。空 ACL 将无法访问群集，只能与启用了 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，在将 ACL 用于 MemoryDB 时，请注意以下用户密码限制：

- 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. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB and Redis 控制台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. 在存储库的创建用户页面中，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. UDER密码，您最多可输入两个密码。
6. UDER访问字符串中，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. 适用于标签，您可以有选择地应用标签来搜索和筛选用户或跟踪您的Amazon成本.
8. 选择 Create（创建）。

使用创建用户Amazon CLI

使用 CLI 创建用户

- 使用创建用户命令创建用户。

  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. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB and Redis 控制台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，如果您有两个密码，则在修改其中一个密码时必须输入两个密码。
5. 如果要更新访问字符串，请输入新的访问字符串。
6. 选择修改。

使用修改用户Amazon CLI

使用 CLI 修改用户

1. 使用更新用户命令修改用户。
2. 修改用户后，将更新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）

在控制台上查看用户详细信息

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB and Redis 控制台https://console.aws.amazon.com/memorydb/.
2. 在左侧导航窗格中，选择。用户.
3. 选择下面的用户用户名或者使用搜索框来查找用户。
4. UDER用户设置您可以查看用户的访问字符串、密码计数、状态和 Amazon 资源名称 (ARN)。
5. UDER访问控制列表 (ACL)您可以查看用户属于的 ACL。
6. UDER标签您可以查看与用户关联的任何标签。

使用命令查看用户详细信Amazon CLI

使用描述用户命令来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. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB and Redis 控制台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->Delete
4. 要确认，请输入`delete`在确认文本框中，然后选择Delete.
5. 要取消，请选择 Cancel (取消)。

使用删除用户Amazon CLI

使用 CLI 删除用户

- 使用删除用户命令删除用户。

  用户帐户将被删除并从其所属的任何访问控制列表中移除。以下是示例。

  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权限，如下所示。

使用控制台通过以下过程管理访问控制列表。

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. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB and Redis 控制台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. 在存储库的创建访问控制列表 (ACL)页面中，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. UDER选定的用户请执行下列操作之一：

   a. 选择创建新用户创建用户

   b. 通过选择添加用户Manage然后从管理用户对话框然后选择选择.

6. 适用于标签，您可以有选择地应用标签来搜索和筛选 ACL 或跟踪Amazon成本.

7. 选择 Create（创建）。

使用命令创建访问控制列表 (ACL)Amazon CLI

使用 CLI 通过以下过程创建访问控制列表。

使用 CLI 创建新 ACL 并添加用户

- 使用创建-ACL命令来创建 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. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB and Redis 控制台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. 在存储库的修改页面，下选定的用户请执行下列操作之一：

   a. 选择创建新用户创建用户以添加到 ACL 中。

   b. 选择或删除用户Manage然后从管理用户对话框然后选择选择.

5. 在存储库的创建访问控制列表 (ACL)页面中，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. UDER选定的用户请执行下列操作之一：

   a. 选择创建新用户创建用户

   b. 通过选择添加用户Manage然后从管理用户对话框然后选择选择.

7. 选择修改要保存您的更改或Cancel丢弃它们。

使用命令修改访问控制列表 (ACL)Amazon CLI

使用 CLI 通过添加新用户或移除当前成员来修改 ACL

- 使用更新 ACL命令来修改 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. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB and Redis 控制台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. 选择下面的 ACLACL 名称或者使用搜索框来查找 ACL。
4. UDER用户您可以查看与 ACL 关联的用户列表。
5. UDER关联的群集你可以查看 ACL 所属的集群。
6. UDER标签您可以查看与 ACL 关联的任何标签。

使用命令查看访问控制列表 (ACL)Amazon CLI

使用描述 ACL命令来查看 ACL 的详细信息。

```
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. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB and Redis 控制台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Delete
4. 在存储库的Delete页面，输入`delete`在确认框中并选择Delete要么Cancel以避免删除 ACL。

将删除 ACL 本身，而不是属于该组的用户。

使用命令删除访问控制列表 (ACL)Amazon CLI

使用 CLI 删除 ACL

- 使用删除 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给群集

创建 ACL 并添加用户后，实施 ACL 的最后步骤是将 ACL 分配给集群。

使用控制台将访问控制列表分配给群集

要将 ACL 添加到群集中，请使用Amazon Web Services Management Console请参阅创建内存数据库集群 (p. 13).

将访问控制列表分配给群集使用Amazon CLI

以下Amazon CLI操作创建一个启用了传输中加密 (TLS) 的集群并acl-name具有值的参数*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Amazon CLI操作修改启用了传输中加密 (TLS) 的集群并acl-name具有值的参数new-acl-2.

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"
```

Redis MemoryDB 中的身份和权限管理

访问 Redis 的 MemoryDB 需要符合以下条件的证书Amazon可以用来验证您的请求。这些凭证必须有权访问Amazon资源，例如 MemoryDB 集群或 Amazon Elastic Compute Cloud (Amazon EC2) 实例。以下各节提供了有关如何使用的详细信息Amazon Identity and Access Management（IAST-1）和 MemoryDB，通过控制可以访问资源的对象来保护资源。

身份验证

您可以以下面任一类型的身份访问 Amazon：

- Amazon Web Services 账户 根用户

  当您创建 Amazon Web Services 账户时，最初使用的是一个对账户中所有 Amazon Web Services和资源拥有完全访问权限的登录身份。此身份称为 Amazon Web Services 账户根用户，使用您创建账户时所用的电子邮件地址和密码登录，即可获得该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅《Amazon 一般参考》中的需要根用户凭证的任务。
- IAM 用户和组

IAM 用户是Amazon Web Services 账户内对某个人员或应用程序具有特定权限的一个身份。在可能的情况下，我们建议依赖于临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，我们建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的使用场景定期轮换访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可能具有一个名为 IAMAdmins 的组，并为该组授予权限以管理 IAM 资源。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅 IAM 用户指南中的何时创建 IAM 用户（而不是角色）。

- IAM 角色

  IAM 角色是可在账户中创建的一种具有特定权限的 IAM 身份。IAM 角色类似于 IAM 用户，因为它是一个 Amazon 身份，具有确定其在 Amazon 中可执行和不可执行的操作的权限策略。但是，角色旨在让需要它的任何人代入，而不是唯一地与某个人员关联。此外，角色没有关联的标准长期凭证（如密码或访问密钥）。相反，当您代入角色时，它会为您提供角色会话的临时安全凭证。具有临时凭证的 IAM 角色在以下情况下很有用：

  - Federated user access（联合用户访问）– 要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关联合身份验证的角色的信息，请参阅《IAM 用户指南》中的为第三方身份提供商创建角色。
  - Amazon Web Service 访问 – 服务角色是一个 IAM 角色，服务代入该角色以代表您执行操作。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅 IAM 用户指南中的创建向 Amazon Web Service 委派权限的角色。
  - 在 Amazon EC2 上运行的应用程序 – 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 Amazon CLI 或 Amazon API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 Amazon 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅 IAM 用户指南中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

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为 Redis 资源创建或访问 MemoryDB。例如，您必须有权创建 MemoryDB 集群。

下面几节介绍如何管理 Redis 的 MemoryDB 的权限。我们建议您先阅读概述。

- 管理您的 MemoryDB 资源的访问权限概览 (p. 182)
- 为适用于 Redis 的 MemoryDB 使用基于身份的策略（IAM 策略）(p. 185)

管理您的 MemoryDB 资源的访问权限概览

每个 Amazon 资源都归某个 Amazon 账户所有，创建和访问资源的权限由权限策略进行管理。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，适用于 Redis 的 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 最佳实践。

在授予权限时，您可以决定谁获得权限。您还可以决定他们获得对哪些资源的权限以及允许对这些资源执行的具体操作。

主题

Redis 资源和操作的 MemoryDB

在 Redis 的 MemoryDB 中，主要资源是聚集.

这些资源具有关联的唯一 Amazon 资源名称 (ARN)，如下表所示。

> 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:*us-east-1#*:user/user1
访问控制列表 (ACL)	arn: aws: memorydb:*us-east-1#*:acl/myacl
Cluster	arn: aws: memorydb:*us-east-1#*:cluster/my-cluster
快照	arn: aws: memorydb:*us-east-1#*:快照/我的快照
参数组	arn: aws: memorydb:*us-east-1#*:parametergroup/my-parameter-group
子网组	arn: aws: memorydb:*us-east-1#*:子网组/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 Redis 的 MemoryDB操作.

了解资源所有权

资源所有者是创建资源的 Amazon 账户。也就是说，资源拥有者是委托人实体的 Amazon 账户，可对创建相应资源的请求进行身份验证。一个主要实体可以是根账户、IAM 用户或 IAM 角色。以下示例说明了它的工作原理：

- 假设您使用的 root 账户凭证Amazon用于创建集群的帐户。在这种情况下，您的 Amazon 账户是资源的拥有者。在 MemoryDB 中，该资源为集群。
- 假设您在自己的中创建了一个 IAM 用户Amazon帐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，您的Amazon该用户所属的账户拥有该集群资源。
- 假设您在中创建 IAM 角色Amazon一个有权创建集群的账户。在这种情况下，能够担任该角色的任何人都可以创建集群。你的Amazon该角色所属的账户拥有该集群资源。

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 Redis 的 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 policy 语法和说明的信息，请参阅 IAM 用户指南中的 Amazon IAM 策略参考。

附加到 IAM 身份的策略称为基于身份的策略（IAM policy）。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 policy）(p. 183)
- 指定策略元素：操作、效果、资源和主体 (p. 184)
- 在策略中指定条件 (p. 184)

基于身份的策略（IAM policy）

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，以向其他Amazon账户（如账户 B）或某项Amazon服务授予跨账户权限，如下所述：
  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. 之后，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 中的用户可以创建或访问账户 A 中的资源。在一些情况下，您可能需要向 Amazon 服务授予担任该角色的权限。为支持此方法，信任策略中的委托人也可以是 Amazon 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 Amazon 账户执行 `DescribeClusters` 操作的示例策略。MemoryDB 还支持使用 API 操作的资源 ARN 来标识特定资源。（此方法也称为资源级权限。）

```
{
   "Version": "2012-10-17",
   "Statement": [{
```

```
      "Sid": "DescribeClusters",
      "Effect": "Allow",
      "Action": [
         "memorydb:DescribeClusters"],
      "Resource": resource-arn
      }
   ]
}
```

有关对 MemoryDB 使用基于身份的策略的更多信息，请参阅为适用于 Redis 的 MemoryDB 使用基于身份的策略（IAM 策略）(p. 185). 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 Redis 资源的每个 MemoryDB 资源（请参阅Redis 资源和操作的 MemoryDB (p. 182)），该服务定义了一组 API 操作（请参阅操作)。为授予执行这些 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`，以及`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：

- Resource（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 Redis 资源和操作的 MemoryDB (p. 182)。
- 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，取决于指定的`Effect`，`memorydb:CreateCluster`权限允许或拒绝执行适用于 Redis 的 MemoryeDB 的用户权限`CreateCluster`操作。
- Effect（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 policy）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 policy 语法和描述的更多信息，请参阅 IAM 用户指南中的Amazon IAM policy 参考。

有关显示所有 Redis API 操作的 MemoryDB 的表，请参阅MemoryDB API 权限：操作、资源和条件参考 (p. 199).

在策略中指定条件

当您授予权限时，可使用 IAM policy 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。

为适用于 Redis 的 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**
>
> 我们建议您首先阅读说明管理对 Redis 资源的访问的基本概念和选项的主题。有关更多信息，请参阅 管理您的 MemoryDB 资源的访问权限概览 (p. 182)。

本主题的各个部分涵盖以下内容：

- 使用 Redis 控制台的 MemoryDB 所需的权限 (p. 185)
- Amazon适用于 Redis 的 MemoryDB 托管（预定义）策略 (p. 197)
- 客户托管的策略示例 (p. 186)

下面介绍权限策略示例。

```
{
   "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：

- 第一条语句为 Redis 操作授予了 MemoryDB 的权限 (`memorydb:CreateCluster`,`memorydb:DescribeClusters`，以及`memorydb:UpdateCluster`) 在该账户拥有的任何集群上。
- 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委托人将获取权限。

有关显示所有 Redis API 操作及它们适用的资源的表，请参阅MemoryDB API 权限：操作、资源和条件参考 (p. 199).

使用 Redis 控制台的 MemoryDB 所需的权限

权限参考表列出 Redis API 操作的 MemoryDB 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 (p. 199).

要使用 MemoryDB for Redis 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。

```
{
```

```
    "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` 操作权限来查询 Amazon EC2，以便可以显示可用区、VPC、安全组及账户属性。
- 权限`cloudwatch`操作使控制台能够检索亚马逊 CloudWatch 指标和告警并在控制台中显示它们。
- `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。

客户托管的策略示例

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 (p. 187)）。或者您应该已经创建了 MemoryDB 服务相关角色。

与使用 MemoryDB for Redis 控制台所需的最低权限相结合时，本节中的示例策略将授予其他权限。这些示例还与 Amazon 开发工具包 和 Amazon CLI 相关。有关使用 MemoryDB 控制台所需的权限的更多信息，请参阅使用 Redis 控制台的 MemoryDB 所需的权限 (p. 185).

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`填充控制台列表的操作。

示例

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的 MemoryDB 操作的权限。通常，您将此类型的权限策略挂载到管理人员组。

```
{
   "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。

```
{
   "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。

```
{
   "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。

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
```

```
      "Sid":"CreateSLRAllows",
      "Effect":"Allow",
      "Action":[
        "iam:CreateServiceLinkedRole"
      ],
      "Resource":"*",
      "Condition":{
        "StringLike":{
          "iam:AmazonServiceName":"memorydb.amazonaws.com"
        }
      }
    }
  ]
}
```

资源级权限

您可以通过在 IAM policy 中指定资源来限制权限范围。很多Amazon CLIAPI 操作支持的资源类型因操作行为而有所不同。每条 IAM policy 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 Resource Name (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 字符串上的资源名称应该为小写。

- 用户 — arn: aws: memorydb:*us-east-1#*:user/user1
- ACL — arn: aws: memorydb:*us-east-1#*:acl/my-acl
- 集群 — arn: aws: memorydb:*us-east-1#*:cluster/my-cluster
- 快照 — arn: aws: memorydb:*us-east-1#*:快照/我的快照
- 参数组 — arn: aws: memorydb:*us-east-1#*:parametergroup/my-parameter-group
- 子网组 — arn: aws: memorydb:*us-east-1#*:子网组/my-subnet-group

示例

- 示例 1：允许用户完全访问特定 MemoryDB 资源类型 (p. 188)
- 示例 2：拒绝用户访问集群。 (p. 188)

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对子网组、安全组和集群类型的所有资源的完全访问权限。

```
{
        "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。

以下示例明确拒绝指定的account-id访问特定集群。

```
{
        "Sid": "Example2",
        "Effect": "Deny",
        "Action": "memorydb:*",
        "Resource": [
                "arn:aws:memorydb:us-east-1:account-id:cluster/name"
        ]
}
```

对 Redis 使用 Amazon MemoryDB 的服务相关角色

Amazon MemoryDB of RedisAmazon Identity and Access Management(IAM)服务相关角色. 服务相关角色是一种独特类型的 IAM 角色，它与直接相关Amazon服务，例如 Amazon MemoryDB of Redis。Amazon MemoryDB of Redis 服务相关角色由 Amazon MemoryDB of Redis 预定义。它们包含该服务代表您的集群调用 Amazon 服务所需的一切权限。

服务相关角色可让您更轻松地设置 Amazon MemoryDB of Redis，因为您不必手动添加必要的权限。你的角色已经存在于你的Amazon但与 Amazon MemoryDB 针对 Redis 使用案例关联并有预定义的权限。只有 Amazon MemoryDB of Redis 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 Amazon MemoryDB 用于 Redis 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它服务的信息，请参阅使用 IAM 的 Amazon 服务并查找 Service-Linked Role（服务相关角色）列中显示为 Yes（是）的服务。请选择 Yes 与查看该服务的服务相关角色文档的链接。

目录

Amazon MemoryDB of Redis 的服务相关角色权限

Amazon MemoryDB of Redis 使用名为的服务相关角色AWSServiceRoleForMemoryDB— 此策略允许 MemoryDB 管理Amazon代表您管理集群所需的资源。

这些区域有：AWSServiceRoleForMemoryDB服务相关角色权限策略允许 Amazon MemoryDB of Redis 对指定资源完成以下操作：

```
{
 "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 Amazon托管策略：Amazon 内存 DBServiceRole策略 (p. 195)。

要允许 IAM 实体创建AWSServiceRoleForMemoryDB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-cn:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB*",
    "Condition": {"StringLike": {"iam:AmazonServiceName": "memorydb.amazonaws.com"}}
}
```

允许删除 IAM 实体AWSServiceRoleForMemoryDB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-cn:iam::*:role/aws-service-role/memorydb.amazonaws.com/
AWSServiceRoleForMemoryDB*",
    "Condition": {"StringLike": {"iam:AmazonServiceName": "memorydb.amazonaws.com"}}
}
```

此外，也可以使用Amazon托管策略以提供对 Amazon MemoryDB of Redis 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 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色. 然后选择 Create new role (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 Amazon Service（亚马逊云科技服务）。
4. UNDER或选择服务以查看其使用案例，选择内存 DB.
5. 选择 Next:。Permissions (下一步：权限)。

6. 在 Policy name（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 Create role。

创建服务相关角色 (IAM CLI)

您可以从 Amazon Command Line Interface 中使用 IAM 操作创建服务相关角色。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 Redis 的 Amazon MemoryDB 的服务相关角色的描述

Amazon MemoryDB of Redis 不允许您编辑AWSServiceRoleForMemoryDB服务相关角色。创建服务相关角色后，将无法更改角色名称，因为可能有多个实体引用该角色。但是可以使用 IAM 编辑角色说明。

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 控制台的导航窗格中，选择角色.
2. 选择要修改的角色的名称。
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)

您可以从 Amazon Command Line Interface 使用 IAM 操作来编辑服务相关角色的描述。

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 Amazon CLI 执行 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 Amazon CLI 执行 IAM 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 Redis 的 Amazon MemoryDB 的服务相关角色

如果不再需要使用某个需要服务相关角色的功能或服务，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。但是，您必须先清除您的服务相关角色，然后才能将其删除。

Amazon MemoryDB of Redis 不会删除您的服务相关角色。

清除服务相关角色

在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 Amazon Web Services Management Console，打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色. 然后选择 AWSServiceRoleForMemoryDB 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 Amazon MemoryDB，以获取需要的 Redis 资源AWSServiceRoleForMemoryDB(console)

- 要删除集群，请参阅以下内容：
  - 使用 Amazon Web Services Management Console (p. 19)
  - 使用 Amazon CLI (p. 19)
  - 使用 MemoryDB (p. 20)

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 Amazon Web Services Management Console，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色. 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，查看上次访问服务数据，该数据显示每个选定角色上次访问的时间Amazon服务。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)

您可以从 Amazon Command Line Interface 中使用 IAM 操作删除服务相关角色。

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。此命令会列出您账户中的角色及其 Amazon 资源名称 (ARN)。

   ```
   $ 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。

Amazon适用于 Redis MemoryDB 的托管策略

要向用户、组和角色添加权限，与自己编写策略相比，使用 Amazon 托管策略更简单。创建仅为团队提供所需权限的 IAM 客户托管策略需要时间和专业知识。要快速入门，您可以使用我们的 Amazon 托管策略。这些策略涵盖常见使用案例，可在您的Amazon账户中使用。有关 Amazon 托管策略的更多信息，请参阅 IAM 用户指南中的 Amazon 托管策略。

Amazon 服务负责维护和更新 Amazon 托管策略。您无法更改 Amazon 托管策略中的权限。服务偶尔会向 Amazon 托管策略添加额外权限以支持新功能。此类更新会影响附加策略的所有身份（用户、组和角色）。当启动新功能或新操作可用时，服务最有可能会更新 Amazon 托管策略。服务不会从 Amazon 托管策略中删除权限，因此策略更新不会破坏您的现有权限。

此外，Amazon 还支持跨多种服务的工作职能的托管策略。例如，ReadOnly访问 Amazon托管策略提供对所有的只读访问Amazon服务和资源。当服务启动新功能时，Amazon会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅 IAM 用户指南中的适用于工作职能的 Amazon 托管策略。

Amazon托管策略：Amazon 内存 DBServiceRole策略

你不能附加 MemoryDBServiceRole策略Amazon针对账户中身份的托管策略。此策略是的一部分 AmazonAmazon MemoryDB 服务相关角色。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

此策略包含以下权限。

```
{
 "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"
    }
   }
  }
 ]
}
```

Amazon适用于 Redis 的 MemoryDB 托管（预定义）策略

Amazon通过提供由创建和管理的独立 IAM 策略来满足许多常用案例的要求。Amazon托管式策略可授予常用案例的必要权限，因此，您可以免去调查都需要哪些权限的工作。有关更多信息，请参阅 IAM 用户指南中的Amazon托管策略。

以下Amazon托管策略（您可以将它们附加到您账户中的用户）是特定于 MemoryDB 的：

AmazonMemoryDBReadOnly访问

您可以将 `AmazonMemoryDBReadOnlyAccess` 策略附加得到 IAM 身份。此策略授予管理权限，允许对所有 MemoryDB 资源进行只读访问。

AmazonMemoryDBReadOnly访问-授予对 MemoryDB 的 Redis 资源的只读访问权限。

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

AmazonMemoryDBFullAccess

您可以将 `AmazonMemoryDBFullAccess` 策略附加得到 IAM 身份。此策略授予管理权限，允许完全访问所有 MemoryDB 资源。

AmazonMemoryDBFullAccess-授予对 Redis 资源的 MemoryDB 的完全访问权限。

```
{
 "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"
    }
   }
  }
 ]
```

```
}
```

Note

您可以通过登录到 IAM 控制台并在该控制台中搜索特定策略来查看这些权限策略。

此外，您还可以创建自定义 IAM 策略，以授予执行 Redis API 操作的 MemoryDB 的权限。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 更新到Amazon托管策略

查看有关更新的详细信息Amazon从该服务开始跟踪这些更改开始，MemoryDB 的托管策略。要查看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
AmazonMemoryDBFullAccess (p. 197)— 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受支持资源。	10/07/2021
AmazonMemoryDBReadOnly访问 (p. 197)— 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 已开启跟踪更改	服务启动	8/19/2021

MemoryDB API 权限：操作、资源和条件参考

当你设置时访问控制 (p. 181)并编写可附加到 IAM 策略的权限策略（基于身份或基于资源），可将下表作为参考。此表列出每个 MemoryDB for Redis 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

监控是保持适用于 Redis 和其他的 MemoryDB 的可靠性、可用性和性能的重要方面Amazon解决方案。Amazon提供以下监控工具来监控 MemoryDB、在出现错误时进行报告并在适当的时候采取自动化措施：

- 亚马逊 CloudWatch监控Amazon资源以及您在上运行的应用程序Amazon实时。您可以收集和跟踪指标，创建自定义的控制面板，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 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 。亚马逊 CloudWatch 用户指南.
- 亚马逊 CloudWatch 日志使您能够监控、存储和访问来自 Amazon EC2 实例的日志文件， CloudTrail和其他来源的数据。 CloudWatch 日志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 。亚马逊 CloudWatch 日志用户指南.
- Amazon CloudTrail 捕获由您的 Amazon 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 Simple Storage Service（Amazon S3）存储桶。您可以标识哪些用户和账户调用了Amazon、从中发出调用的源 IP 地址以及调用的发生时间。有关更多信息，请参阅 Amazon CloudTrail 用户指南。

使用Amazon 监控 DB for Redis CloudWatch

你可以使用监控 Redis 的 MemoryDB CloudWatch，此工具可收集原始数据，并将数据处理为便于读取的近乎实时的指标。这些统计数据会保存 15 个月，从而使您能够访问历史信息，并能够更好地了解您的 Web 应用程序或服务的执行情况。此外，可以设置用于监测特定阈值的警报，并在达到相应阈值时发送通知或执行操作。有关更多信息，请参阅 。亚马逊 CloudWatch 用户指南.

以下部分列出了 MemoryDB 的指标和维度。

主题

主机级指标

这些区域有：`Amazon/MemoryDB`命名空间包含各个节点的以下主机级指标。

另请参阅

- MemoryDB (p. 200)

指标	描述	单位
CPUUtilization	整个主机的 CPU 使用率百分比。由于 Redis 是单线程的，因此，我们建议您监控EngineCPUUtilization有 4 个或更多 vCPUs 的节点的指标。	百分比
FreeableMemory	主机上可用的闲置内存量。这是从内存和缓冲区中派生出来的，并且操作系统报告为空闲的。	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
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
NetworkLinkLocalAllowanceExceeded	由于到本地代理服务的流量的 PPS 超出网络接口的最大值而形成的数据包数量。这会影响流向 DNS 服务、实例元数据服务和 Amazon Time Sync Service 的流量。	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
SwapUsage	主机上的交换区使用量。	字节

MemoryDB

Amazon/memorydb 命名空间包括以下 Redis 指标。

除 ReplicationLag 和 EngineCPUUtilization 之外，这些指标均源自 Redis info 命令。每项指标均是按照节点级计算的。

如需 Redis info 命令的完整文档，请参阅 http://redis.io/commands/info。

另请参阅

- 主机级指标 (p. 199)

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。这是从 Redis INFO 的 active_defrag_hits 统计数据中得出的。	数字
AuthenticationFailures	使用 AUTH 命令向 Redis 进行身份验证的失败尝试总次数。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	Redis 为所有目的（包括数据集、缓冲区等）分配的字节的总数。这是从 Redis INFO 的 used_memory 统计数据中得出的。	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 使用两到四个连接来监控各种情况下的集群。这是根据 Redis INFO 中的 connected_clients 统计数据得出的。	计数
CurrItems	键空间中的项目数。此值根据以下方法获得的 Redis keyspace 统计数据得出：计算整个密钥空间中所有密钥的总和。	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 从 Redis INFO 计算得来的。	百分比
DB0AverageTTL	根据 Redis INFO 命令中的 keyspace 统计数据中公开 DBO 的 avg_ttl。	毫秒
EngineCPUUtilization	提供 Redis 引擎线程的 CPU 使用率。由于 Redis 是单线程的，您可以使用该指标来分析 Redis 进程本身的负载。EngineCPUUtilization 指标更精确地呈现了 Redis 流程。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 Redis 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，	百分比

指标	描述	单位
	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。	
Evictions	由于 maxmemory 限制而被驱逐的密钥数。这是根据 Redis INFO 中的 evicted_keys 统计数据得出的。	计数
IsPrimary	指示节点是否为当前分区的主节点。指标可以是 0（非主节点）或 1（主节点）。	计数
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。这是从 Redis INFO 的 keyspace_hits 统计数据中得出的。	计数
KeyspaceMisses	主字典中失败的只读键查找次数。这是从 Redis INFO 的 keyspace_misses 统计数据中得出的。	计数
KeysTracked	Redis 密钥跟踪所跟踪的密钥数所占 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	在上一个测量周期内观察到的最大复制吞吐量。	每秒字节数
MemoryFragmentationRatio	指示 Redis 引擎的内存分配的效率。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 Redis INFO 中的 mem_fragmentation_ratio statistic 计算得来的。	数字
NewConnections	在此期间，服务器接受的连接总数。这是根据 Redis INFO 中的 total_connections_received 统计数据得出的。	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 Redis 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 Redis INFO 中的 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 Redis INFO 中的 master_repl_offset 统计数据得出的。	字节
ReplicationDelayedWriteCommands	由于超过最大复制吞吐量而延迟的命令数。	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

以下是一些类型的命令的集合，派生自 info commandstats。commandstats 部分提供基于命令类型的统计数据，包括调用次数。

有关可用命令的完整列表，请参阅 Redis 文档中的 Redis 命令。

指标	描述	单位
EvalBasedCmds	基于 eval 的命令的命令总数。这是从 Redis commandstats 统计数据派生的。这是根据 Redis commandstats 统计数据通过计算 eval、evalsha 的总和得出的。	计数
GeoSpatialBasedCmds	基于地理空间的命令的命令总数。这是从 Redis commandstats 统计数据派生的。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 Redis commandstats 统计数据得出的，方式是计算所有 read-only 类型的命令（get、hget、scard、lrange 等）的总和。	计数
HashBasedCmds	基于哈希的命令总数。此值是根据 Redis commandstats 统计数据得出的，方式是计算所有作用于一个或多个哈希的命令（hget、hkeys、hvals、hdel 等）的总和。	计数
HyperLogLogBasedCmds	基于 HyperLogLog 的命令的总数。这是根据 Redis commandstats 统计数据得出的，方式是计算所有 pf 类型的命令（pfadd、pfcount、pfmerge 等）的总和。	计数
JsonBasedCmds	基于 JSON 的命令总数。这是从 Redis 派生的commandstats统计数据，方法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 Redis commandstats 统计数据得出的，方式是计算作用于多个数据结构中的一个或多个键的所有命令（del、expire、rename 等）的总和。	计数
ListBasedCmds	基于列表的命令总数。此值根据 Redis commandstats 统计数据得出，方式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。这是从 Redis commandstats 统计数据得出的，方法是对以下用于发布/订阅功能的所有命令进行求和：psubscribe、publish、pubsub、punsubscribe、subscribe 和 unsubscribe。	计数
SetBasedCmds	基于设置的命令总数。此值根据 Redis commandstats 统计数据得出，方式是计算所有作用于一个或多个设置的命令（scard、sdiff、sadd、sunion 等）的总和。	计数
SetTypeCmds	write 类型命令的总数。这是根据 Redis commandstats 统计数据得出的，方式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

指标	描述	单位
SortedSetBasedCmds	基于设置的已排序命令总数。此值根据 Redis commandstats 统计数据得出，方式是计算所有作用于一个或多个已排序设置的命令（zcount、zrange、zrank、zadd 等）的总和。	计数
StringBasedCmds	基于字符串的命令总数。此值根据 Redis commandstats 统计数据得出，方式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 Redis commandstats 统计数据得出的，方式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？

以下 CloudWatch 指标可深入了解 MemoryDB 性能。在大多数情况下，我们建议您设置 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

CPU 利用率

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 (p. 199)。

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 Redis 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 (vCPUs) 数量，请参阅内存 DB.

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区，在更多主节点中分配写入工作负载。

> Tip
>
> 而不是使用主机级别指标`CPUUtilization`，您也许能够使用 Redis 指标`EngineCPUUtilization`，其中会向您报告 Redis 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.

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 Redis 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.

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 Redis 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.

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 (p. 199)。

此指标不应超过 50 MB。

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。

CurrConnections

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。

越来越多的CurrConnections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 Redis 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 Redis INFO 命令的内存部分中找到。

Network

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 内存 DB.

复制

可通过 `ReplicationBytes` 指标了解被复制的数据量。您可监控`MaxReplicationThroughput`与复制容量吞吐量对比。建议在达到最大复制容量吞吐量时添加更多分片。

`ReplicationDelayedWriteCommands`还可以指示工作负载是否超过最大复制容量吞吐量。有关在 MemoryDB 中复制的更多信息，请参阅了解 DB

选择指标统计数据和周期

WHIL CloudWatch 将允许您为每个指标选择任何统计数据和周期，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。

对于每个单独节点，发布所有 MemoryDB 示例的持续时间均为 60 秒。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

内存 DB CloudWatch 集成在一起，因此可收集多种指标。可用监控这些指标 CloudWatch.

> Note
>
> 以下示例需要使用 CloudWatch 命令行工具。有关 的更多信息 CloudWatch 要下载开发人员工具，请参阅 CloudWatch 产品页面.

下面将介绍如何使用 CloudWatch 以收集过去一小时内集群的存储空间统计数据。

> Note
>
> 这些区域有：`StartTime`和`EndTime`下述示例中提供的值均供说明之用。确保为节点替换相应的开始时间和结束时间值。

有关 DBAmazon服务限制for 内存 DB for Memor

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Amazon Web Services Management Console然后打开适用于 Redis 的 MemoryDB 控制台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. 在存储库的集群的页面Amazon管理控制台，单击一个或多个集群的名称。

      显示群集的详情页面。

   b. 单击位于窗口顶部的 Nodes 选项卡。

   c. 在存储库的节点选项卡上，选择您希望查看其度量标准的节点。

      可用的列表 CloudWatch 度量标准显示在控制台窗口的底部。

   d. 单击 CPU 利用率 度量标准。

      这些区域有： CloudWatch 控制台将打开，其中会显示您选择的指标。您可以使用 Statistic (统计数据) 和 Period (周期) 下拉列表框以及 Time Range (时间范围) 选项卡来更改所显示的指标。

监控 CloudWatch 使用指标 CloudWatch CLI

收集集群的 CPU 利用率统计数据

- 使用 CloudWatch 命令aws cloudwatch get-metric-statistics使用以下参数（请注意，开始和结束时间仅作为示例显示；您需要替换为自己的适当开始和结束时间）：

对于 Linux、macOS 或 Unix：

```
aws cloudwatch get-metric-statistics CPUUtilization \
    --dimensions=ClusterName=mycluster,NodeId=0002" \
    --statistics=Average \
    --namespace="Amazon/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="Amazon/MemoryDB" ^
    --start-time 2013-07-05T00:00:00 ^
    --end-time 2013-07-06T00:00:00 ^
    --period=60
```

监控 CloudWatch 使用指标 CloudWatch API

收集集群的 CPU 利用率统计数据

- 调用 CloudWatch APIGetMetricStatistics使用以下参数（请注意，开始和结束时间仅作为示例显示；您需要替换为自己的适当开始和结束时间）：
  - Statistics.member.1=Average
  - Namespace=Amazon/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 DB for Redis 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 Amazon SNS

您可以配置 MemoryDB 以使用 Amazon Simple Notification (Amazon Simple Notification) 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 Resource Name (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 Amazon SNS 主题Amazon控制台、Amazon CLI，或 MemoryDB API。

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Amazon Web Services Management Console然后打开适用于 Redis 的 MemoryDB 控制台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. 选择修改。

添加 Amazon SNS 主题 (AmazonCLI)

要为集群添加或修改 Amazon SNS 主题，请使用 Amazon 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 (内存 DB)

若要为集群添加或更新 Amazon SNS 主题，请调用`UpdateCluster`操作具有以下参数：

- `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`action.

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 Amazon Web Services Management Console 禁用 Amazon SNS 通知

1. 登录到Amazon Web Services Management Console然后打开适用于 Redis 的 MemoryDB 控制台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. 选择修改。
4. 在 Topic for SNS Notification 下的 Modify Cluster 中，选择 Disable Notifications。
5. 选择修改。

启用和禁用 Amazon SNS 通知 (AmazonCLI)

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 (内存数据库 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 查看

MemoryDB 会记录与集群、安全组和参数组有关的事件。此类信息包括事件的数据和时间、事件的源名称和源类型，以及事件的描述。您可以使用 MemoryDB 控制台轻松从日志中检索事件Amazon CLI `describe-events`命令或 MemoryDB API 操作`DescribeEvents`.

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 DB 查看 (控制台)

以下过程演示了使用 MemoryDB 控制台查看事件。

要查看使用 MemoryDB 使用

1. 登录到Amazon Web Services Management Console然后打开适用于 Redis 的 MemoryDB 控制台https://console.aws.amazon.com/memorydb/.
2. 在左侧导航窗格中，选择事件.

   这些区域有：事件将显示列出所有可用事件的窗口。列表中的每一行代表一个事件，并显示事件源、事件类型（例如集群、参数组、ACL、安全组或子网组）、事件的 GMT 时间和事件的描述。

   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 (AmazonCLI)

要使用Amazon CLI，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`。

查看内存数据库事件（内存数据库 API）

要使用 MemoryDB API 生成 MemoryDB 事件的列表，请使用`DescribeEvents`action. 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-cluster 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

当集群上发生重大事件时，MemeryDB 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终端节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。
- 这些区域有：Amazon拥有 Amazon SNS 主题的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
内存 DB：addNode完成	`"Modified number of nodes from %d to %d"`	已将节点添加到集群，并准备就绪，可供使用。
由于空闲 IP 地址不足导致的:addNodeFailed	`"Failed to modify number of nodes from %d to %d due to insufficient free IP addresses"`	由于没有足够的可用 IP 地址，所以无法添加节点。
MemoryDB：簇参数已更改	`"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。
内存 DB：集群配置完成	`"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:缓存数据库预配置失败	`"Failed to create cluster due to incompatible network state. %s"`	已尝试将新集群启动到不存在的虚拟私有云 (VPC) 中。
内存 DB：cluster RestoResto 失败	`"Restore from %s failed for node %s. %s"`	MemoryDB 无法使用 Redis 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确。如果您对集群进行描述，状态将是`restore-failed`. 您需要删除集群，从头再来。 有关更多信息，请参阅使用外部创建的快照为新集群设定种子 (p. 128)。
内存 DB：群集扩展完成	`"Succeeded applying modification to node type to %s."`	已成功纵向扩展集群。
内存 DB：群集扩展失败	`"Failed applying modification to node type to %s."`	对集群的纵向扩展操作已失败。
内存 DB：已修改群集安全组	`"Modified security group for cluster."`	发生下列事件之一： • 已修改授权用于集群的安全组列表。

事件名称	消息	描述
		• 已在与集群相关的任何安全组上授权一个或多个新的 EC2 安全组。 • 已从与集群相关的任何安全组中撤销一个或多个 EC2 安全组。
内存 DB：节点替换已开始	`"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。
内存 DB：节点替换完成	`"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。
内存 DB：创建群集完成	`"Cluster created"`	已成功创建集群。
内存 DB：创建群集失败	`"Failed to create cluster due to unsuccessful creation of its node(s)."` 和 `"Deleting all nodes belonging to this cluster."`	群集未创建。
内存 DB：删除群集完成	`"Cluster deleted."`	已完成集群和所有关联节点的删除工作。
内存 DB：故障转移完成	`"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
内存 DB：节点替换已取消	`"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：节点替换已重新安排	`"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窗口期间替换。 有关您可以执行的操作的信息，请参阅替换节点 (p. 23)。
内存 DB：已计划节点替换	`"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窗口期间替换。 有关您可以执行的操作的信息，请参阅替换节点 (p. 23)。
内存 DB：删除节点完成	`"Removed node %s"`	已从集群中移除节点。
内存 DB: Snapshot 完成	`"Snapshot %s succeeded for node %s"`	快照已成功完成。
内存 DB：快照失败	`"Snapshot %s failed for node %s"`	快照失败。有关失败原因的详细信息，请参阅集群的事件。 如果您对快照加以说明，请参阅DescribeSnapshots，状态将为`failed`.

使用记录 redis API 调用的 MemoryDBAmazon CloudTrail

redis 的 MemoryDB 与Amazon CloudTrail，该服务提供了用户、角色或Amazon在 MemoryDB 中为 Redis 提供服务。CloudTrail 将对 Redis 的 MemoryDB 的所有 API 调用作为事件捕获，包括来自 Memory DB 的 Redis API 操作的代码调用。如果您创建跟踪，则可以将 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 Redis 的 Memory DB 事件）。如果您不配置跟踪记录，则仍可在 CloudTrail 控制台中的 Event history（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、请求的发出时间以及其他详细信息。

如需了解有关 CloudTrail 的更多信息，请参阅 Amazon CloudTrail 用户指南。

CloudTrail 中的 Redis 信息的 MemoryDB

在您创建 Amazon 账户时，将在该账户上启用 CloudTrail。当 MemoryDB 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他事件一起保存在 CloudTrail 事件Amazon服务事件历史记录. 您可以在 Amazon 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。

要持续记录您的事件Amazon要创建跟踪，包括 MemoryDB 的事件，请创建跟踪。通过跟踪记录，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 Amazon 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Amazon S3 存储桶。此外，您可以配置其他 Amazon 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅下列内容：

- 创建跟踪记录概述
- 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

CloudTrail 将记录所有 MemoryDB 操作。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身份用户的临时安全凭证发出的。
- 请求是否由其他 Amazon 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 Redis 日志文件条目的 MemoryDB

跟踪记录是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 存储桶。CloudTrail 日志文件包含一个或多个日志条目。一个事件表示来自任何源的一个请求，包括有关请求的操作、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": "6.2.4",
            "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",
            "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，该条目说明了 `DescribeClusters` 操作。请注意，对于 Redis 的所有 MemoryDB 描述和列出调用（`Describe*`和`List*`），`responseElements`部分已删除并显示为`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"
}
```

下面示例显示了一个 CloudTrail 日志条目，该条目记录了`UpdateCluster`action.

```
{
    "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": "6.2.4",
            "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
        }
    },
    "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，该条目说明了 `CreateUser` 操作。请注意，对于包含敏感数据的 MemoryDB to Redis 调用，该数据将在相应的 CloudTrail 事件中进行编辑，如`requestParameters`在下面的部分。


```
{
    "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"
}
```


Redis Amazon MemoryDB 中的基础设施安全性

作为托管服务，MemoryDB 受Amazon中描述的全局网络安全程序Amazon Web Services：安全过程概述白皮书。

你用Amazon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) 1.0 或更高版本。建议使用 TLS 1.2 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 委托人关联的秘密访问密钥来对请求进行签名。或者，您可以使用 Amazon Security Token Service (Amazon STS) 生成临时安全凭证来对请求进行签名。

的合规性验证

作为多个计划的一部分，第三方审计员将评估适用于 Redis 的 MemoryDB 的安全性和合规性验证Amazon合规性计划。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。

有关特定合规性计划范围内的Amazon服务的列表，请参阅。

您可以使用 Amazon Artifact 下载第三方审计报告。有关更多信息，请参阅下载 Amazon Artifact 中的报告。

您在使用 MemoryDB 时的合规性责任由您的数据的敏感性、您公司的合规性目标以及适用的法律法规决定。Amazon提供以下资源来帮助实现合规性：

- 安全性与合规性 Quick Start 指南安全性与合规性 Quick Start 指南 - 这些部署指南讨论了架构注意事项，并提供了在 Amazon 上部署基于安全性和合规性的基准环境的步骤。
- Amazon合规性资源 – 此业务手册和指南集合可能适用于您的行业和位置。
- 使用规则评估资源中的Amazon Config开发人员指南–Amazon 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。
- Amazon Security Hub – 此 Amazon 服务提供了 Amazon 中安全状态的全面视图，可帮助您检查是否符合安全行业标准和最佳实践。
- Amazon Audit Manager – 此 Amazon服务帮助您持续审计您的Amazon使用情况，以简化管理风险以及与相关法规与行业标准的合规性的方式。

互联网络流量隐私

Redis MemoryDB for Redis 使用以下技术保护您的数据免受未经授权的访问：

- MoryDB 和Amazon VPC (p. 221) 说明了安装所需的安全组类型。
- Redis API 和接口 VPC 终端节点 (Amazon PrivateLink) (p. 237)允许您在 VPC 和For Redis API 终端节点内存数据库之间建立私有连接。
- Redis MemoryDB 中的身份和权限管理 (p. 180) 用于授予和限制用户、组和角色的操作。

MoryDB 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。使用 Amazon VPC 配置虚拟私有云 (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 安全组控制对该集群的访问。

本部分说明如何在 VPC 中手动配置 MemoryDB 集群。此类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 (p. 222)
- 用于访问 Amazon VPC 中 MemoryDB 集群的访问模式 (p. 224)
- 创建 Virtual Private Cloud (VPC) (p. 227)

了解 MemoryDB 和 VPC

MemoryDB 与 Amazon VPC 完全集成在一起。对于 MemoryDB 用户，这具有以下意义：

- MemoryDB 始终在 VPC 中启动您的集群。
- 如果您是的新用户Amazon，默认 VPC 会为您自动创建。
- 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Amazon非常类似于传统数据中心的云。您可以配置您的 VPC，包括选择它的 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 中的内存数据库概述

	VPC 是Amazon分配有自己的 IP 地址块的云。
	互联网网关将您的 VPC 直接连接到互联网，并提供访问其他Amazon资源（如 Amazon S3）等）在 VPC 外部运行的资源。
	Amazon VPC 子网是 VPC 的 IP 地址范围的一部分，在其中您可以隔离Amazon根据您的安全和运营需求提供资源。
	您 VPC 中的路由表可定向子网与 Internet 之间的网络流量。亚马逊 VPC 有一个隐含的路由器。
	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 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 内创建 MemoryDB 集群，您的 VPC 必须满足以下要求：

- 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）。通过互联网网关，您的 VPC 可以直接访问其他Amazon未在您的 VPC 中运行的资源。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 Internet 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，当您访问Amazon通过 Internet 获取资源。

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 名称连接到节点，因为基本 IP 地址可能会随着时间的推移而变化。

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。下表显示在 Amazon VPC 指南中的位置。

描述	文档
如何开始使用 Amazon VPC	Amazon VPC 入门
如何通过 Amazon Web Services Management Console 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	Amazon 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

适用于 Redis 的 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

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 集群的默认端口是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. 为集群的安全组创建自定义规则，该规则允许从为 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 Amazon 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 Security Groups (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 集群的默认端口是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 中但位于同一区域中，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

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 Amazon 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 Amazon 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 Amazon 账户发挥作用，您可以使用 Amazon Resource Access Manager 与其他账户共享您的中转网关。在您与另一个 Amazon 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 Amazon 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 Amazon 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 Amazon 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。

从运行于客户数据中心中的应用程序访问 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 连接从运行于客户数据中心的应用程序访问 MemoryDB 集群

从数据中心或通过 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 从运行于客户数据中心的应用程序访问 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 Amazon 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 Virtual Private Cloud (VPC)

在此示例中，您基于 Amazon VPC 服务创建一个 Virtual Private Cloud (VPC)，并为每个可用区设置一个私有子网。

创建 VPC（控制台）

在 Amazon Virtual Private Cloud 内部创建 MoryDB

1. 登录 Amazon 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 控制面板中，选择创建 VPC.
3. 在要创建的 Resources（资源）下，选择 VPC and more（VPC 等）。
4. 在 Number of Availability Zones (AZs)（可用区数量）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，以及哪些是公有的，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。您将对您的集群和 Amazon EC2 实例使用这个安全组。

   a. 在左侧导航窗格中的Amazon Web Services Management Console，选择个安全组.
   b. 选择 Create Security Group（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。适用于VPC，选择您 VPC 的标识符。
   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 Security Groups (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
      - 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
      - 源：0.0.0.0/0

      选择 Apply Rule Changes。

您现在可以创建子网组和创建集群您的 VPC 中的。

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，您可以指定子网组或使用提供的默认子网组。MemyDB 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 第 2 步：授予集群的访问权限 (p. 17)。

受支持的 MemoryDB AZ ID

区域名称/区域	支持的 AZ ID		
美国东部（俄亥俄）区域 `us-east-2`	`use2-az1, use2-az2, use2-az3`		
美国东部（弗吉尼亚北部）区域 `us-east-1`	`use1-az2, use1-az4, use1-az6`		
美国西部（加利福尼亚北部）区域 `us-west-1`	`usw1-az1, usw1-az2, usw1-az3`		
美国西部（俄勒冈）区域 `us-west-2`	`usw2-az1, usw2-az2, usw2-az3`		
加拿大（中部）区域 `ca-central-1`	`cac1-az1, cac1-az2, cac1-az3`		
亚太地区（香港）区域 `ap-east-1`	`ape1-az1, ape1-az2, ape1-az3`		
亚太（孟买）区域 `ap-south-1`	`aps1-az1, aps1-az2, aps1-az3`		
亚太区域（东京） `ap-northeast-1`	`apne1-az1, apne1-az2, apne1-az4`		
亚太区域（首尔） `ap-northeast-2`	`apne2-az1, apne2-az2, apne2-az3`		
Asia Pacific (Singapore) Region `ap-southeast-1`	`apse1-az1, apse1-az2, apse1-az3`		

区域名称/区域	支持的 AZ ID		
亚太区域（悉尼） ap-southeast-2	apse2-az1, apse2-az2, apse2-az3		
欧洲（法兰克福）区域 eu-central-1	euc1-az1, euc1-az2, euc1-az3		
Europe (Ireland) Region eu-west-1	euw1-az1, euw1-az2, euw1-az3		
欧洲（伦敦）区域 eu-west-2	euw2-az1, euw2-az2, euw2-az3		
欧洲（斯德哥尔摩）区域 eu-north-1	eun1-az1, eun1-az2, eun1-az3		
南美洲（圣保罗）区域 sa-east-1	sae1-az1、sae1-az2、sae1-az3		
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		

主题

- 创建子网组 (p. 230)
- 更新子网组 (p. 232)
- 查看子网组详细信息 (p. 233)
- 删除子网组 (p. 236)

创建子网组

当您创建新的子网组时，请记下可用 IP 地址的数量。如果子网拥有的空闲 IP 地址寥寥无几，则您还可以向集群中添加的节点数可能会受限制。要解决此问题，您可以对某一子网组分配一个或多个子网，这样集群的可用区中便会有充足数量的 IP 地址。之后，便可向您的集群中添加更多节点。

以下过程介绍如何创建一个名为的子网组。`mysubnetgroup`（控制台）、Amazon 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. 登录到Amazon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，请执行以下操作：
   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。如果你还没有创建一个，请选择创建 VPC按钮并按照步骤创建。
   d. In选定子网中，选择您的私有子网的可用区和 ID，然后选择选择.
5. 适用于标签，您可以有选择地应用标签来搜索和筛选子网或跟踪Amazon成本.
6. 当所有设置都按照需要进行后，选择Create.
7. 在出现的确认信息中，选择 Close。

您的新子网组将显示在子网组MemoryDB 控制台的列表。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 (AmazonCLI)

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 Amazon CLI 主题 create-subnet-group。

创建子网组（MemoryDB API）

使用 MemoryDB API 调用`CreateSubnetGroup`使用以下参数调用：

- `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Amazon Web Services Management Console然后在以下位置打开 MemoryDB and Redis 控制台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和说明字段不可修改。
5. 在选定子网部分点击Manage以对子网所需的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 (AmazonCLI)

在命令提示符处，使用命令`update-subnet-group`以更新子网组。

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 。Amazon CLI话题更新子网组.

更新子网组（MemoryDB API）

使用 MemoryDB API 调用`UpdateSubnetGroup`使用以下参数调用：

- `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。如果子网拥有的空闲 IP 地址寥寥无几，则您还可以向集群中添加的节点数可能会受限制。要解决此问题，您可以对某一子网组分配一个或多个子网，这样集群的可用区中便会有充足数量的 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。

查看子网组的详细信息（控制台）

查看子网组的详细信息 (控制台)

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB and Redis 控制台https://console.aws.amazon.com/memorydb/.
2. 在左侧导航窗格中，选择子网组.
3. 在存储库的子网组页面中，选择下面的子网组名称或者在搜索栏中输入子网组的名称。
4. 在存储库的子网组页面中，选择下面的子网组名称或者在搜索栏中输入子网组的名称。
5. UDER子网组设置您可以查看子网组的名称、描述、VPC ID 和 Amazon Resource Name (ARN)。
6. UDERSubnets您可以查看子网组的可用区、子网 ID 和 CIDR 块

7. UDER标签您可以查看与子网组关联的任何标签。

查看子网组详细信息 (AmazonCLI)

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，但不指定子网组名称。

```
aws memorydb describe-subnet-groups
```

有关更多信息，请参阅 Amazon CLI 主题 describe-subnet-groups。

查看子网组（MemoryDB API）

使用 MemoryDB API 调用`DescribeSubnetGroups`使用以下参数调用：

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。你必须先取消选中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Amazon Web Services Management Console然后在以下位置打开 MemoryDB and Redis 控制台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，选择操作然后选择Delete.

   Note

   您无法删除默认子网组或与任何集群关联的子网组。
4. 这些区域有：删除子网组此时会显示确认屏幕。
5. 要删除子网组，请输入`delete`在确认文本框中。要保留子网组，请选择取消。

删除子网组 (AmazonCLI)

通过使用 Amazon 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 。Amazon CLI话题删除子网组.

删除子网组（MemoryDB API）

使用 MemoryDB API 调用`DeleteSubnetGroup`使用以下参数调用：

- `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
```

此命令不会生成任何输出。

有关更多信息，请参阅 MemoryDB API 主题DeleteSubnetGroup.

Redis API 和接口 VPC 终端节点 (Amazon PrivateLink)

您可以在 VPC 和之间建立私有连接。亚马逊 用于 Redis API 端点的 MemoryDB 通过创建接口 VPC 终端节点. 接口终端节点支持提供支持 Amazon PrivateLink.Amazon PrivateLink 允许您可以私下访问内存 DB，而无需互联网网关、NAT 设备、VPN 连接或AmazonDirect Connect 连接。

VPC 中的实例即使没有公有 IP 地址也可与 Redis API 终端节点进行通信。您的实例也不需要公有 IP 地址即可使用任何可用的内存 DB API 操作。Redis VPC 和内存 DB 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，请参阅接口 VPC 终端节点 (Amazon PrivateLink) 中的Amazon VPC User Guide.
- 有关内存 DB API 操作的更多信息，请参阅Amazon 内存 DB API 操作.

在创建接口 VPC 终端节点后，如果启用私有 DNS终端节点的主机名，默认 MemoryDB 终端节点 (https://memorydb.##.amazonaws.com) 解析为您的 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 。接口 VPC 终端节点 (Amazon PrivateLink) 中的Amazon VPC User Guide. 支持调用它的所有内存 DBAPI 操作在您的 VPC 内。

> **Note**
>
> 只能为 VPC 中的一个 VPC 终端节点启用私有 DNS 主机名。如果您想创建额外的 VPC 终端节点，则应为其禁用私有 DNS 主机名。

VPC 终端节点注意事项

在为 Redis API 终端节点设置内存 DB 接口 VPC 终端节点之前，请务必查看。接口终端节点属性和限制中的Amazon VPC User Guide. 所有内存 DB API 操作那是可以从使用的 VPC 中获取与管理 Redis 资源相关的内存 DB。Amazon PrivateLink. 内存 DB API 终端节点支持 VPC 终端节点策略。默认情况下，允许通过终端节点对内存 DB API 操作进行完全访问。有关更多信息，请参阅《Amazon VPC 用户指南》中的使用 VPC 终端节点控制对服务的访问权限。

为 创建接口 VPC 终端节点这API 内存 DB

您可以使用 Amazon VPC 控制台或Amazon CLI. 有关更多信息，请参阅 Amazon VPC 用户指南.中的创建接口端点

在创建接口 VPC 终端节点后，您可以为端点启用私有 DNS 主机名。当你这样做时，Redis 端点的默认 MemoryDB (https://memorydb.##.amazonaws.com) 解析为您的 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 创建 VPC 终端节点策略这Amazon 内存 DB API

您可以为 VPC 终端节点附加控制对内存 DB API 的访问的终端节点策略。此策略指定以下内容：

- 可执行操作的委托人。
- 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 终端节点控制对服务的访问。

Example 用于内存 DB API 操作的 VPC 终端节点策略

下面是用于内存 DB API 的终端节点策略示例。当附加到终端节点时，此策略会向所有委托人授予对列出的针对所有资源的内存 DB API 操作的访问权限。

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "memorydb:CreateCluster",
   "memorydb:UpdateCluster",
   "memorydb:CreateSnapshot"
  ],
  "Resource": "*"
 }]
}
```

Example 拒绝来自指定的所有访问的 VPC 终端节点策略Amazon帐户

以下 VPC 终端节点策略拒绝 Amazon 账户 *123456789012* 所有使用终端节点访问资源的权限。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"
    ]
   }
  }
 ]
}
```

用于 Redis 的 MemoryDB 中的服务更新

MemoryDB in Redis 可自动监控您的集群和节点队列，以便在有服务更新可用时进行应用。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。但是，在某些情况下，您可能会发现此方法过于僵化，可能会限制您的业务流程。

利用服务更新，您可以控制何时应用更新和应用哪些更新。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条件的集群，则将在发布更新时将通过电子邮件、Personal Health Dashboard (PHD) 和 Amazon CloudWatch 事件接收通知。更新也会显示在服务更新在 MemoryDB 控制台上显示的页面。通过使用此控制面板，您可以查看 MemoryDB 队列的所有服务更新及其状态。

您可以控制何时应用更新，然后再开始自动更新。强烈建议您应用任何类型的更新。更新安全更新尽快确保您的 MemoryDB 始终使用最新的安全修补程序保持最新。

以下各节详细探索了这些选项。

主题

应用服务更新

您可以从服务更新具有可用status。服务更新是累积的。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，则可以选择在服务更新可用时不采取任何操作。MemoryDB 将安排在群集的维护时段内应用更新自动更新开始日期. 您将收到更新的每个阶段的相关通知。

> Note
>
> 你只能应用那些具有可用要么计划的status。

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 (p. 239).

在新的服务更新适用于一个或多个 MemoryDB 集群时，您可以使用 MemoryDB 控制台、API 或Amazon 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

要查看可用的服务更新列表以及其他信息，请转到服务更新控制台中的页面。

1. 登录到Amazon Web Services Management Console然后在以下位置打开 MemoryDB for Redis 控制台https://console.aws.amazon.com/memorydb/.
2. 在导航窗格中，选择服务更新.

UNDER更新详细信息您可以查看以下内容：

- 服务更新名称：服务更新的唯一名称
- 更新说明：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，MemoryDB 将在此日期之后开始安排集群在适当的维护窗口中自动更新。您将在确切的计划维护时段提前收到通知，但在自动更新开始日期. 您仍然可以随时选择将更新应用于集群。如果未设置该属性，则服务更新不会启用自动更新，MemoryDB 不会自动更新您的集群。

在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刚应用服务更新的群集的列表。对于每个集群，您可以查看以下内容：

- 集群名称：集群的名称

- 节点已更新：特定集群中已更新或仍可用于特定服务更新的各个节点的比率。
- 更新类型：服务更新的类型，它为更新安全更新要么更新引擎更新
- 状态：集群上服务更新的状态，它为下列状态之一：
  - 可用：此更新适用于必需的集群。
  - 正在进行：正在对此集群应用更新。
  - 计划的：已计划更新日期。
  - 完成：更新已成功应用。状态完成的集群将在完成后 7 天内显示。

  如果您选择了任何或所有集群可用要么计划的状态，然后选择立即申请，更新将开始在这些群集上应用。

使用 Amazon CLI 应用服务更新

在收到服务更新可用的通知后，您可以使用 Amazon 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 。描述-服务更新.
- 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 。批量更新集群.

参考

本部分中的主题涵盖了有关使用的主题涵盖了有关使用的 MemoryDB API 和的部分的信息。Amazon CLI. 本部分还包含常见错误消息和服务通知。

- 使用 MemoryDB API (p. 242)
- 的记忆数据库 API 参考
- 的 MemoryDB 部分Amazon CLI参考

使用 MemoryDB API

本部分提供有关如何使用和实施 MemoryDB 操作的说明。有关这些操作的完整介绍，请参阅MemoryDB API 参考.

主题

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 Amazon 的请求进行身份验证的基本步骤。其中假定您注册了 Amazon，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 Amazon 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 Amazon。
4. Amazon 使用访问密钥 ID 来查询秘密访问密钥。
5. Amazon 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 Amazon 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

Amazon 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，帮助您在与 MemoryDB API 进行交互时排查问题。

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一个`Error`来自 MemoryDB API 中做出响应的节点。

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 的运行是否正常。

  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 或未知操作例外，确认服务是否可用，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 参考. 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以及解决方案。如要查看论坛，请参阅

  https://forums.aws.amazon.com/.

Redis 的亚马逊 MemoryDB 的配额

您的 Amazon 账户对于每项 Amazon 服务都具有默认配额（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。如果配额在 Service Quotas 中尚不可用，请使用提高限制表格。

您的Amazon您的账户具有以下与 MemoryDB 相关的配额。

资源	默认值
每个区域的节点数	300
每个实例类型的节点	90
每个分片的节点	6
每个区域的参数组数	150
每个区域的子网组数	150
每个子网组的子网数	20

《内存 DB 用户指南》的文档历史记录

下表介绍了内存 DB 的文档版本。

变更	说明	日期
内存 DB 现在支持本机版本 JavaScript 对象表示法 (JSON) 格式 (p. 247)	本机人 JavaScript 对象表示法（JSON）格式是在 Redis 集群中对复杂数据集进行编码的一种简单的无模式的方法。您可以使用本机存储和访问数据 JavaScript 在 Redis 集群中进行对象表示法（JSON）格式并更新在这些集群中存储的 JSON 数据-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
内存 DB 现在支持Amazon PrivateLink (p. 247)	Amazon PrivateLink 使您可以私下访问 MemoryDB API 操作，而无需互联网网关、VPN 连接或AmazonDirecConnect 有关更多信息，请参阅 。内存 DB API 和接口 VPC 终端节点 (Amazon PrivateLink).	2022 年 1 月 24 日
首次发布 (p. 247)	内存 DB 用户指南首次发布。有关更多信息，请参阅 。什么是内存 DB？	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。