aws

用户指南

ElastiCache 适用于 Redis 的 Amazon

API 版本 2015-02-02

ElastiCache 适用于 Redis 的 Amazon: 用户指南

Table of Contents

适用于 Redis ElastiCache 的 Amazon 是什么？

欢迎阅读亚马逊 Red ElastiCache is 版用户指南。Amazon ElastiCache 是一项 Web 服务，可以轻松地在云中设置、管理和扩展分布式内存数据存储或缓存环境。它可以提供高性能、可扩展且具有成本效益的缓存解决方案。同时，它可以帮助消除与部署和管理分布式缓存环境相关的复杂性。

您可以用两种 ElastiCache 格式运营亚马逊。您可以从无服务器缓存开始入手，也可以选择设计自己的缓存群集。

> **Note**
>
> 亚马逊同时 ElastiCache 使用 Redis 和 Memcached 引擎。使用您感兴趣的引擎的指南。如果您不确定要使用哪个引擎，请参阅本指南中的比较 Memcached 和 Redis 自行设计缓存。

无服务器缓存

ElastiCache for Redis 提供无服务器缓存，可简化为应用程序添加和操作基于 Redis 的缓存。ElastiCache for Redis Serverless 使您能够在不到一分钟的时间内创建高可用性缓存，并且无需预置实例或配置节点或集群。开发人员可以通过使用 ElastiCache 控制台、SDK 或 CLI 指定缓存名称来创建无服务器缓存。

ElastiCache 对于 Redis Serverless 来说，还无需规划和管理缓存容量。 ElastiCache for Redis 会持续监控应用程序使用的缓存内存、计算和网络带宽，并根据应用程序的需求进行扩展。 ElastiCache for Redis 通过抽象底层缓存基础设施和集群设计，为开发人员提供了简单的端点体验。 ElastiCache for Redis 可以自动透明地管理硬件配置、监控、节点更换和软件修补，因此您可以专注于应用程序开发，而不是操作缓存。

ElastiCache 适用于 Redis Serverless 与 Redis 7.1 及更高版本兼容。

ElastiCache 为 Redis 集群设计自己的集群

如果您需要精细控制您 ElastiCache 的 for Redis 集群，则可以选择使用设计自己的 Redis 集群。ElastiCache ElastiCache 允许您通过为集群选择节点类型、节点数量和跨 AWS 可用区的节点放置来设计集群。由于 ElastiCache 是一项完全托管的服务，因此它可以自动管理集群的硬件配置、监控、节点更换和软件修补。

为 Redi ElastiCache s 集群设计自己的集群可以提高集群的灵活性和控制力。例如，在运行集群时，您可以根据需要选择单可用区可用性或多可用区可用性。您也可以选择在集群模式下运行 Redis 以启用水平扩展，或者不选择集群模式以进行垂直扩展。在设计自己的集群时，您需要负责选择正确的节点类型和数量，以确保您的缓存具有应用程序所需的足够容量。您还可以选择何时对 Redis 集群应用新的软件补丁。

在为 Redis 集群设计自己的 ElastiCache 集群时，您可以选择运行 Redis 3.0 及更高版本。

相关服务

Amazon MemoryDB for Redis

在决定是使用 ElastiCache for Redis 还是适用于 Redis 的 Amazon MemoryDB 时，请考虑以下比较结果：

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

Amazon RDS

ElastiCache for Redis 可以将经常访问的数据存储在缓存中，帮助您节省数据库费用。如果您的应用程序有较高的读取吞吐量要求，则可以使用 ElastiCache（而不是扩展底层数据库）来实现大规模、高速性能和更低的数据存储成本。

工作方式

在这里，您可以找到 for Redis 部署的主要组件 ElastiCache 的概述。

缓存和缓存引擎

缓存是一种内存中的数据存储，可用于存储缓存的数据。通常，您的应用程序会将经常访问的数据缓存在缓存中，以优化响应时间。 ElastiCache for Redis 提供两种部署选项：无服务器集群和自行设计的集群。请参阅 选择部署选项。

> **Note**
>
> 亚马逊同时 ElastiCache 使用 Redis 和 Memcached 引擎。使用您感兴趣的引擎的指南。如果您不确定要使用哪个引擎，请参阅本指南中的比较 Memcached 和 Redis 自行设计缓存。

主题

- ElastiCache 对于 Redis 来说是如何工作的
- 定价维度
- ElastiCache 用于 Redis 备份

ElastiCache 对于 Redis 来说是如何工作的

ElastiCache 适用于 Redis 无服务器

ElastiCache for Redis Serverless 使您能够创建缓存，而不必担心容量规划、硬件管理或集群设计。您只需为缓存提供一个名称，即可收到一个端点，可以在 Redis 客户端中配置此端点以开始访问缓存。

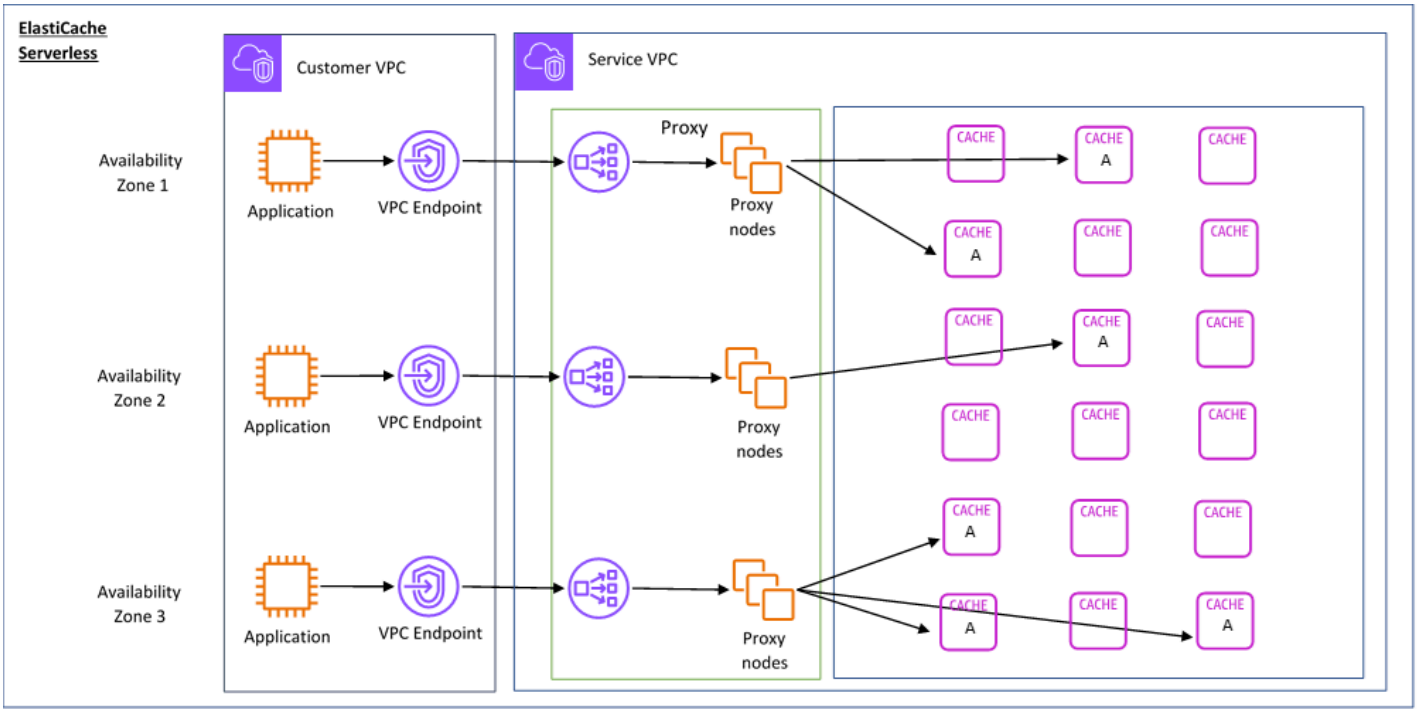
> **Note**
>
> ElastiCache for Redis Serverless 在集群模式下运行 Redis，并且仅与同时支持 TLS 和 Redis 集群协议的 Redis 客户端兼容。

主要优势

- 无需进行容量规划：ElastiCache 无服务器让您无需规划容量。 ElastiCache Serverless 持续监控缓存的内存、计算和网络带宽利用率，并可纵向和横向扩展。它可以增大缓存节点，同时并行启动横向扩展操作，以确保缓存能够扩展以始终满足您的应用程序需求。
- Pay-per-use：使用 ElastiCache Serverless，您需要为缓存中的工作负载所存储的数据和使用的计算付费。请参阅 定价维度。

- 高可用性：ElastiCache Serverless 会自动跨多个可用区 (AZ) 复制您的数据，以实现高可用性。它会自动监控底层缓存节点，并在出现故障时将其替换。它为每个缓存提供 99.99% 可用性 SLA。
- 自动软件升级：ElastiCache Serverless会自动将您的缓存升级到最新的次要版本和补丁软件版本，而不会对您的应用程序的可用性产生任何影响。当有新的 Redis 主版本可用时， ElastiCache 将向您发送通知。
- 安全性：无服务器始终对传输中数据和静态数据进行加密。您可以使用服务托管密钥或您自己的客户自主管理型密钥，对静态数据进行加密。

下图说明了 ElastiCache 无服务器的工作原理。



创建新的无服务器缓存时， ElastiCache 将在您的 VPC 中您选择的子网中创建一个虚拟私有云 (VPC) 终端节点。您的应用程序可以通过这些 VPC 端点连接到缓存。

使用 ElastiCache Serverless，您可以收到应用程序连接到的单个 DNS 端点。当您请求与端点建立新连接时， ElastiCache Serverless 会通过代理层处理所有缓存连接。代理层有助于减少复杂的客户端配置，因为在底层集群发生变化时，客户端无需重新发现集群拓扑。代理层是一组使用网络负载均衡器处理连接的代理节点。当应用程序创建新的缓存连接时，网络负载均衡器会将请求发送到代理节点。当应用程序执行缓存命令时，连接到应用程序的代理节点会在缓存中的缓存节点上执行请求。代理层从客户端提取缓存群集拓扑和节点。这使您 ElastiCache 能够智能地进行负载平衡、横向扩展和添加新的缓存节点、在缓存节点出现故障时更换缓存节点以及更新缓存节点上的软件，所有这些都不会影响应用程序的可用性，也不必重置连接。

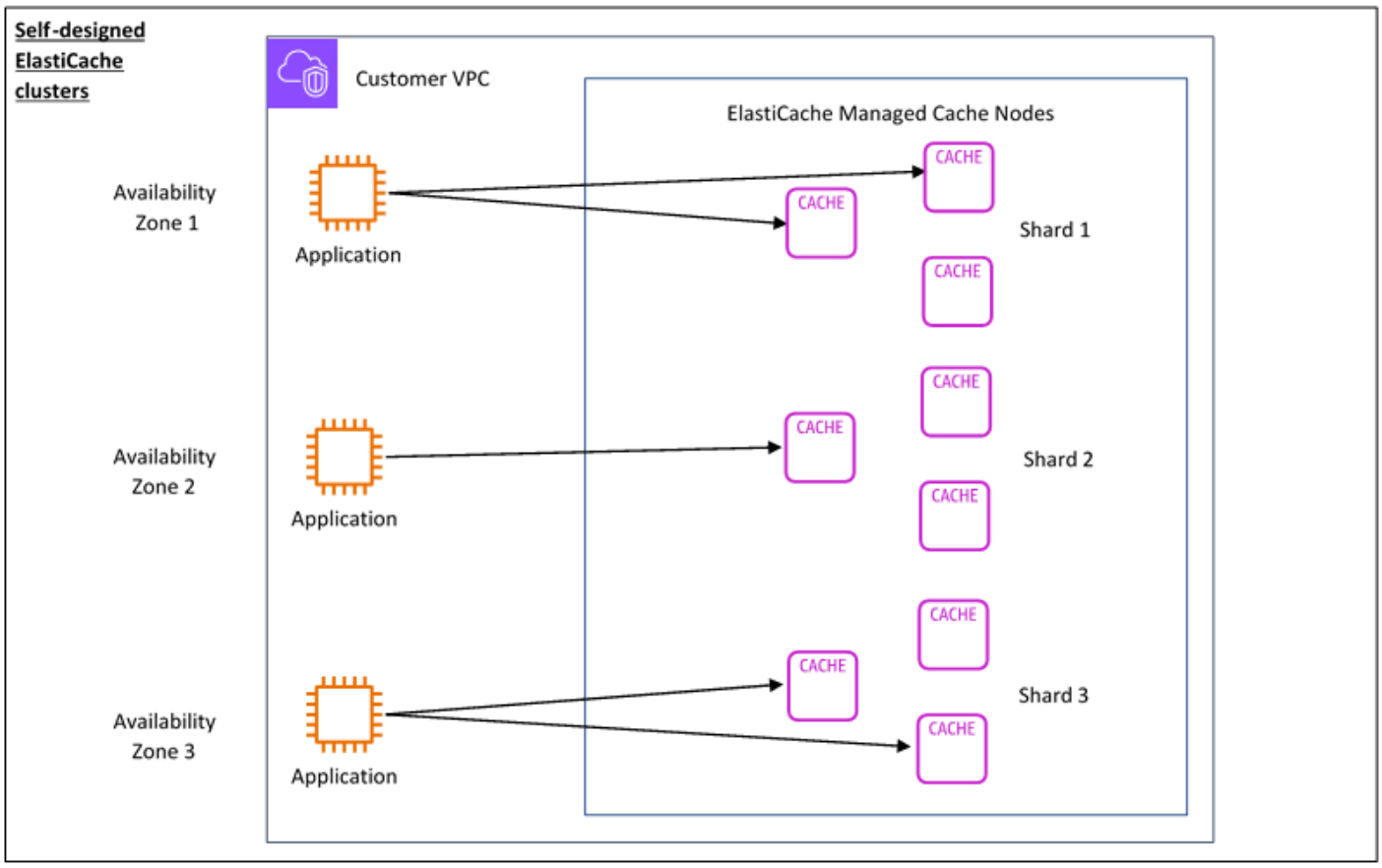
自行设计 ElastiCache 的集群

您可以通过为 ElastiCache 集群选择缓存节点系列、大小和节点数量来选择设计自己的集群。设计自己的集群可让您实施更精细的控制，并让您选择缓存中的分片数和每个分片中的节点（主节点和副本）数。您可以选择通过创建带多个分片的集群，以便在集群模式下操作 Redis，或者在非集群模式下使用单个分片来操作 Redis。

主要优势

- 设计自己的集群：使用 ElastiCache，您可以设计自己的集群并选择要放置缓存节点的位置。例如，如果您的应用程序需要放弃高可用性来换取低延迟，则可以选择在单个可用区中部署缓存节点。或者，在设计集群时，您可以使用跨多个可用区的节点，从而实现高可用性。
- 精细控制：在设计自己的集群时，您可以通过微调缓存上的设置来实现更多控制。例如，您可以使用 Redis 特定的参数 配置缓存引擎。
- 垂直和水平扩展：在需要时，您可以选择增大或减小缓存节点大小来手动扩缩集群。您也可以通过添加新分片或向分片添加更多副本来水平扩展。您还可以使用 Auto-Scaling 功能根据计划配置扩展，或者根据缓存上的 CPU 和内存使用率等指标进行扩展。

下图说明了 ElastiCache 自行设计的集群的工作原理。



定价维度

您可以通过两个部署 ElastiCache 选项进行部署。在部署 ElastiCache Serverless 时，您需要为以 GB 小时存储的数据和按 ElastiCache 处理单元 (ECPU) 计算的数据的使用量付费。选择为 Redis 集群设计自己的 ElastiCache 集群时，您需要按每小时的缓存节点使用量付费。请参阅此处的定价详细信息。

数据存储

您需要为存储在 ElastiCache 无服务器中的数据付费，按千兆字节小时 (GB-Hr) 计费。 ElastiCache Serverless 持续监控存储在缓存中的数据，每分钟采样多次，并计算每小时平均值以确定缓存的数据存储使用量（以 GB-Hr 为单位）。每个 ElastiCache 无服务器缓存按流量计量存储至少 1 GB 的数据。

ElastiCache 处理单元 (ECPU)

您需要为应用程序在 ElastiCache 无服务器 ElastiCache 处理单元 (ecPU) 上执行的 Redis 请求付费，该单位包括 vCPU 时间和传输的数据。

- 对于传输的每 KB 数据，简单读取和写入需要 1 个 ECPU。例如，传输最多 1 KB 数据的 GET 命令将使用 1 个 ECPU。传输 3.2 KB 数据的 SET 请求将使用 3.2 个 ECPU。
- 需要额外的 vCPU 时间的命令将按比例消耗更多 ECPU。例如，如果您的应用程序使用 Redis HMGET 命令，并且使用的 vCPU 时间是简单 SET/GET 命令的 3 倍，那么它将使用 3 个 ECPU。
- 使用更多 vCPU 时间和传输更多数据的命令会根据两个维度中的较高者使用 ECPU。例如，如果您的应用程序使用 HMGET 命令，并且使用的 vCPU 时间是简单 SET/GET 命令的 3 倍，并传输 3.2 KB 数据，那么它将使用 3.2 个 ECPU。或者，如果它仅传输 2 KB 数据，则将使用 3 个 ECPU。

ElastiCache Serverless 会发出一个名为的新指标`ElastiCacheProcessingUnits`，该指标可帮助您了解工作负载消耗的 ECPU。

节点小时数

您可以通过选择 EC2 节点系列、大小、节点数和跨可用区放置，来设计自己的 Redis 缓存群集。在自行设计集群时，您需要按小时为每个缓存节点付费。

ElastiCache 用于 Redis 备份

备份是 Redis 缓存的 point-in-time 副本。 ElastiCache 使您可以随时备份数据或设置自动备份。备份可用于还原现有缓存或为新集群制作种子。备份包含缓存中的所有数据和某些元数据。有关更多信息，请参阅 快照和还原。

选择部署选项

Amazon ElastiCache 有两个部署选项：

- 无服务器缓存
- 自行设计的集群

有关两者支持的命令列表，请参阅支持和限制使用的 Redis 命令。

无服务器缓存

Amazon ElastiCache Serverless 简化了缓存的创建并可即时扩展以支持客户要求最苛刻的应用程序。借 ElastiCache 助 Serverless，您可以在不到一分钟的时间内创建高度可用且可扩展的缓存，无需预置、规划和管理缓存集群容量。 ElastiCache Serverless 自动在三个可用区之间冗余存储数据，并提供 99.99% 的可用性服务级别协议 (SLA)。备份是交叉兼容的，可以导出到自己设计的集群，也可以从中恢复。

自行设计的集群

如果您需要精细控制您 ElastiCache 的 for Redis 集群，则可以选择使用设计自己的 Redis 集群。ElastiCache ElastiCache 允许您通过为集群选择节点类型、节点数量和跨 AWS 可用区的节点放置来操作基于节点的集群。由于 ElastiCache 是一项完全托管的服务，因此它可以帮助管理集群的硬件配置、监控、节点更换和软件修补。自行设计的集群可以设计为提供高达 99.99% 的可用性 SLA。备份是交叉兼容的，可以导出到 Serveless 缓存并从中恢复。

选择部署选项

在以下情况下，选择无服务器缓存：

- 您正在为新的或难以预测的工作负载创建缓存。
- 您具有不可预测的应用程序流量。
- 您想以最轻松的方式开始使用缓存。

在以下情况下，选择设计自己的 ElastiCache 集群：

- 您已经在运行 ElastiCache Serverless，想要更精细地控制运行 Redis 的节点类型、节点数量和节点的位置。
- 您期望应用程序流量相对可预测，并且希望对性能、可用性和成本进行精细控制。
- 您可以预测容量要求以控制成本。

比较无服务器缓存和自行设计的集群

功能	无服务器缓存	自行设计的集群
缓存设置	在不到一分钟的时间内创建一个只有名字的缓存	提供对缓存集群设计的精细控制。用户可以选择节点类型、节点数量和跨 AWS 可用区域的位置
支持 Red ElastiCache is 版本	ElastiCache 适用于 Redis 版本 7.1 及更高版本	ElastiCache 适用于 Redis 版本 4.0 及更高版本
集群模式	`cluster mode enabled`仅在中操作 Redis。Redis 客户	可以配置为在启用集群模式或禁用集群模式下运行。

功能	无服务器缓存	自行设计的集群
	端必须支持`cluster mode enabled`才能连接到 ElastiCache 无服务器。	
扩展	无需任何容量管理即可自动纵向和横向扩展。	提供对扩展的控制，同时还需要监控以确保当前容量足以满足需求。 您可以选择垂直扩展，方法是在需要时增加或减少缓存节点的大小。您还可以通过向分片添加新分片或向分片中添加更多副本来进行水平扩展。 借助自动缩放功能，您还可以根据计划配置扩展，或者根据缓存上的 CPU 和内存使用率等指标进行扩展。
客户端连接	客户端连接到单个端点。这使得底层缓存节点拓扑（扩展、替换和升级）可以在不断开客户端连接的情况下进行更改。	客户端连接到每个单独的缓存节点。如果更换了节点，客户端会重新发现集群拓扑并重新建立连接。
可配置性	没有精细的配置可用。客户可以配置基本设置，包括可以访问缓存的子网、是开启还是关闭自动备份以及最大缓存使用限制。	自行设计的集群提供了精细的配置选项。客户可以使用参数组进行精细控制。有关由节点类型决定的这些参数值的表，请参阅特定于 Redis 节点类型的参数。
多可用区	数据在多个可用区之间异步复制，以提高可用性并缩短读取延迟。	提供在单个可用区或跨多个可用区 (AZ) 设计集群的选项。对于多可用区集群，数据在多个可用区之间异步复制，以提高可用性并缩短读取延迟。

功能	无服务器缓存	自行设计的集群
静态加密	始终启用。客户可以在中使用 AWS 托管式密钥 或客户管理的密钥 AWS KMS。	启用或禁用静态加密的选项。启用后，客户可以在中使用 AWS 托管式密钥 或客户管理的密钥 AWS KMS。
传输中的加密 (TLS)	始终启用。客户端必须支持 TLS 连接。	启用或禁用的选项。
备份	支持自动和手动备份缓存，不会影响性能。 备份是交叉兼容的，可以还原到 ElastiCache 无服务器缓存或自行设计的集群中。	支持自动和手动备份。集群可能会受到一些性能影响，具体取决于可用的预留内存。有关更多信息，请参阅管理预留内存。 备份是交叉兼容的，可以还原到 ElastiCache 无服务器缓存或自行设计的集群中。
监控	Support 缓存级别指标，包括缓存命中率、缓存失误率、数据大小和消耗的 ECPU。 ElastiCache Serverless 使用缓存中发生重大事件 EventBridge 时发送事件。您可以选择使用 Amazon EventBridge 监控、摄取、转换和 ElastiCache 处理事件。有关更多信息，请参阅无服务器缓存事件。	ElastiCache 自行设计的集群会在每个节点级别发布指标，包括主机级别的指标和缓存指标。 自行设计的集群会针对重大事件发出 SNS 通知。请参阅 Redis 的指标。
可用性	99.99% 可用性服务等级协议 (SLA)	自行设计的集群可以设计为实现高达 99.99% 的可用性服务级别协议 (SLA)，具体取决于配置。

功能	无服务器缓存	自行设计的集群
软件升级和修补	自动将缓存软件升级到最新的次要版本和补丁版本，而不会影响应用程序。客户会收到主要版本升级的通知，客户可以在需要时升级到最新的主要版本。	自行设计的集群为次要版本和补丁版本升级以及主要版本升级提供支持客户的自助服务。托管更新会在客户定义的维护时段内自动应用。客户还可以选择按需应用次要版本或补丁版本升级。
全球数据存储	不支持	支持 Global Data Store，支持通过单区域写入和多区域读取实现跨区域复制
数据分层	不支持	使用 r6gd 系列节点设计的集群的数据在内存和本地 SSD（固态驱动器）存储之间分层。除了将数据存储在内存中外，数据分层还通过在每个群集节点中使用成本较低的固态硬盘 (SSD)，为 Redis 工作负载提供了一种性价比高的选择。
定价模型	Pay-per-use，基于以 GB 小时为单位存储的数据和 ElastiCache 处理单元 (ECPU) 中的请求。请参阅此处的定价详细信息。	Pay-per-hour，基于缓存节点的使用情况。请参阅此处的定价详细信息。

相关主题:

- 设计和管理自己的 ElastiCache 集群

Amazon ElastiCache 资源

我们建议您先阅读以下部分，并在需要时参阅此部分内容：

- 服务亮点和定价 – 产品详细信息页面提供涵盖 ElastiCache 服务亮点和定价的一般产品概览。
- ElastiCache 视频 – ElastiCache 视频 部分包含向您介绍 Amazon ElastiCache 的视频。这些视频包含 ElastiCache 的常见使用案例以及说明如何使用 ElastiCache 减少延迟并提高应用程序吞吐量的演示。
- 入门 – 开始使用适用于 Redis ElastiCache 的 Amazon 部分包括有关创建缓存集群的信息。它还包括如何授权访问缓存集群、连接到缓存节点以及删除缓存集群。
- 规模性能 – 利用 Amazon ElastiCache 实现规模性能白皮书介绍了能够帮助您的应用程序在处理大规模负载时良好运行的缓存策略。

如果要使用 AWS Command Line Interface (AWS CLI)，您可以利用以下文档帮助您开始使用：

- AWS Command Line Interface 文档

  此部分提供有关下载 AWS CLI、使 AWS CLI 在系统上工作以及提供 AWS 凭证的信息。
- ElastiCache 的 AWS CLI 文档

  此单独文档的内容涵盖所有 AWS CLI for ElastiCache 命令（包括语法和示例）。

您可以使用各种常用的编程语言编写应用程序，以利用 ElastiCache API。下面是一些资源：

- 用于 Amazon Web Services 的工具

  Amazon Web Services 提供了许多支持 ElastiCache 的软件开发工具包（SDK）。您可以使用 Java、.NET、PHP、Ruby 和其他语言为 ElastiCache 编写代码。这些开发工具包可以格式化面向 ElastiCache 的请求、分析响应并提供重试逻辑和错误处理，从而极大简化应用程序开发。
- 使用 ElastiCache API

  如果您不想使用 AWS 开发工具包，可以直接使用查询 API 与 ElastiCache 交互。您可以在此部分中查找有关创建请求和验证请求身份以及处理响应的问题排查提示和信息。
- Amazon ElastiCache API 参考

  此单独文档的内容涵盖所有 ElastiCache API 操作（包括语法和示例）。

AWS区域和可用区

Amazon 云计算资源安置在世界不同地区 (例如：北美、欧洲或亚洲) 的高度可用的数据中心设施内。每个数据中心位置称为一个 AWS 区域。

每个 AWS 区域包含很多称为可用区或 AZ 的不同位置。每个可用区设计为可隔离其他可用区的故障。每个可用区都设计为向同一 AWS 区域中的其他可用区提供低成本、低延迟的网络连接。通过启动独立可用区内的实例，您可以保护您的应用程序不受单一位置故障的影响。有关更多信息，请参阅选择区域和可用区。

您可以在多个可用区创建集群，此选项称为多可用区部署。当您选择此选项时，Amazon 会自动在不同的可用区预置和维护辅助备用节点实例。主节点实例可以跨可用区异步复制到辅助实例。此方法帮助提供数据冗余和故障转移支持，消除 I/O 冻结，并在系统备份期间将延迟峰值降至最小。有关更多信息，请参阅通过多可用区最大程度地减少 ElastiCache for Redis 中的停机时间。

ElastiCache 的常见使用案例以及 ElastiCache 能够如何帮助您

无论是提供最新资讯、前 10 位的排行榜、产品目录还是销售活动的门票，速度都是关键。传输内容的速度对您的网站和业务的成功有很大的影响。

《纽约时报》曾报道："对于没有耐心的 Web 用户而言，一眨眼的功夫都显得太长，"用户会记下竞争网站之间的 250 毫秒（1/4 秒）的差异。用户会离开速度较慢的网站并转至速度较快的网站。亚马逊开展了一项测试（引自网页加载时间与访客流失率的相关性），结果表明，加载时间每增加 100 毫秒 (1/10 秒)，销售额就会减少 1%。

如果有人需要数据，在该数据已缓存的情况下，您可以更快地传输该数据。无论是网页还是推动业务决策的报告，都是如此。您的公司是否能在不缓存网页的情况下以可能最短的延迟传输网页？

直观上显著的一点是，您需要缓存请求次数最多的项目。但您为何不缓存请求次数极少的项目？ 甚至最优化的数据库查询或远程 API 调用的速度也比从内存中的缓存检索平面密钥的速度慢得多。显著变慢是导致客户流失的原因。

下面的示例演示了使用 ElastiCache 提高应用程序的总体性能的一些方式。

主题

- 内存中的数据存储
- 游戏排行榜 (Redis 排序集)
- 消息发送 (Redis Pub/Sub)
- 推荐数据（Redis 哈希）
- 其他 Redis 用法
- ElastiCache 客户评价

内存中的数据存储

内存中密钥值存储的主要目的是，提供对数据副本的超快 (毫秒级延迟) 的、低成本的访问。大部分数据存储具有经常访问但很少更新的数据区域。此外，查询数据库将始终比在密钥值对缓存中查找密钥更慢且成本更高。某些数据库查询的执行成本特别高。例如，涉及跨多个表连接的查询或含有密集型计算的查询。通过缓存此类查询结果，您只需为查询支付一次费用。然后，您可以快速地多次检索数据，而无需重新执行查询。

我应对哪些数据进行缓存？

在决定要缓存的数据时，请考虑这些因素：

速度和费用 – 与从缓存中获取数据相比，从数据库中获取数据始终更慢且费用更高。一些数据库查询原本就比其他查询更慢且费用更高。例如，在多个表上执行连接的查询要比简单的单表查询更慢且费用更高。如果要使用速度慢且费用高的查询来获取所需数据，则这种情况适合使用缓存。如果要使用相对快速且简单的查询来获取数据，则这种情况可能也适合使用缓存，但具体取决于其他因素。

数据和访问模式 – 确定要缓存的数据还需要了解数据本身及其访问模式。例如，对快速变化或很少访问的数据进行缓存是没有意义的。为了使缓存具有真实的益处，数据应相对静态且被频繁访问。例如，社交媒体网站上的个人资料。另一方面，如果对数据进行缓存不会带来速度或成本优势，那么您将无需缓存数据。例如，对返回搜索结果的网页进行缓存是没有意义的，因为这些查询与结果通常是唯一的。

过时性– 根据定义，缓存的数据是过时的数据。即使在某些情况下它不是过时的，它也应该始终被认为是过时的并按过时数据处理。在判断数据是否适合使用缓存时，请确定应用程序对过时数据的容忍性。

您的应用程序也许能够在一种环境中容忍过时数据，但不能在另一种环境中容忍过时数据。例如，假设您的站点提供公开交易的股票价格。您的客户可能会接受一些过时性，并且免责声明价格可能存在 n 分钟延迟。但是，在向卖出或买入股票的经纪人提供股票的价格时，您需要实时数据。

在以下情况下，需要考虑对数据进行缓存：

- 与缓存检索相比，获取数据的速度更慢且费用更高。
- 用户经常访问您的数据。
- 您的数据保持相对相同，或者如果数据迅速发生变化，过时性不是一个大问题。

有关更多信息，请参阅下列内容：

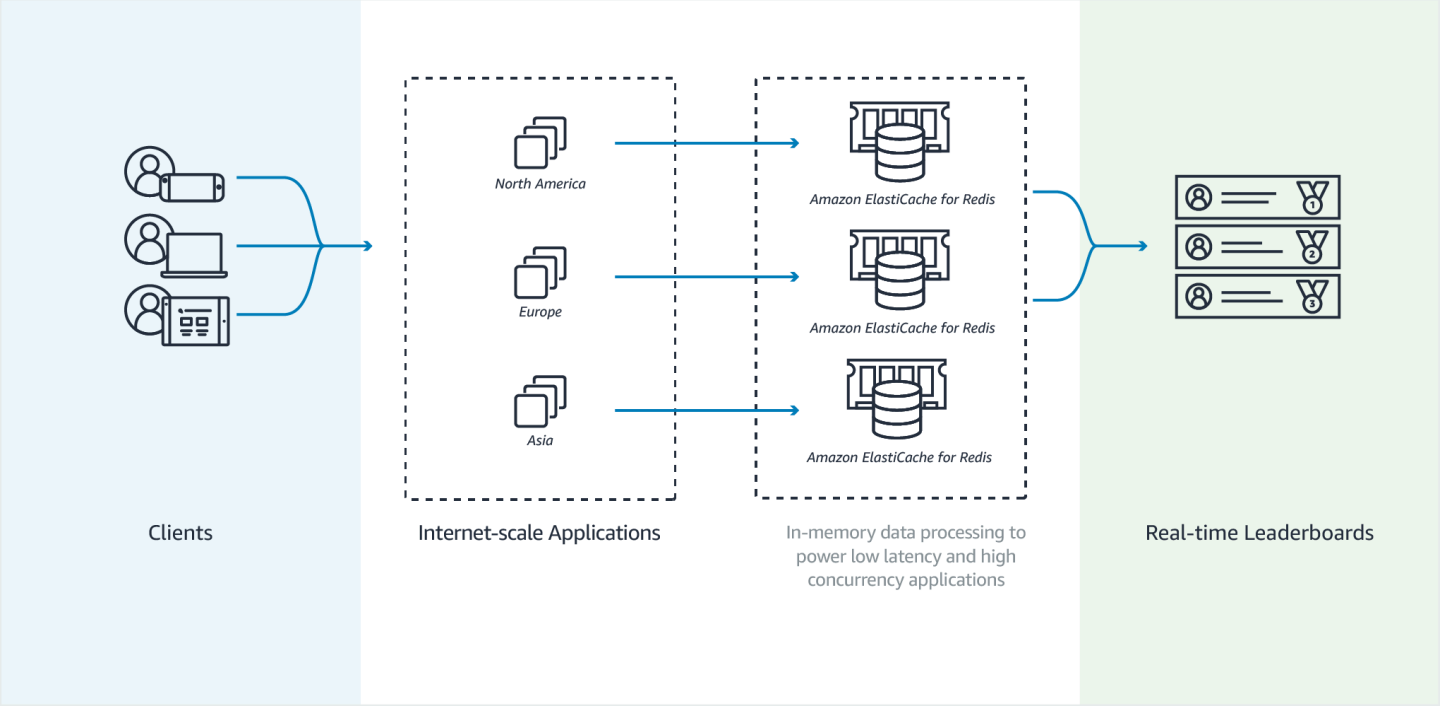
- ElastiCache for Redis 用户指南中的缓存策略

游戏排行榜 (Redis 排序集)

Redis 排序集将与排行榜的计算复杂性从应用程序移至 Redis 集群。

排行榜（例如，游戏的排名前 10 位的分数）的计算量比较大。当有大量并发玩家和不断变化的分数时，这一点尤其如此。Redis 排序集可确保唯一性和元素排序。使用 Redis 排序集时，每次将新元素添加到排序集时，它都会实时重新排序。然后以正确的数字顺序添加到集合内。

在下图中，您可以看到 ElastiCache for Redis 游戏排行榜如何工作。



Example - Redis 排行榜

在本示例中，使用 ZADD 将 4 个游戏玩家及其分数输入排序后的列表中。命令 ZREVRANGEBYSCORE 按玩家的分数列出玩家（从高到低）。紧接着，使用 ZADD 覆盖现有条目来更新 June 的分数。最终，ZREVRANGEBYSCORE 按玩家的分数列出玩家（从高到低）。该列表显示，June 的排名已上升。

```
ZADD leaderboard 132 Robert
ZADD leaderboard 231 Sandra
ZADD leaderboard 32 June
ZADD leaderboard 381 Adam

ZREVRANGEBYSCORE leaderboard +inf -inf
1) Adam
2) Sandra
3) Robert
4) June

ZADD leaderboard 232 June

ZREVRANGEBYSCORE leaderboard +inf -inf
1) Adam
2) June
3) Sandra
4) Robert
```

使用以下命令，June 可以获知自己在所有玩家中的排名。由于排名是从零开始的，因此 ZREVRANK 会为排名第二的 June 返回 1。

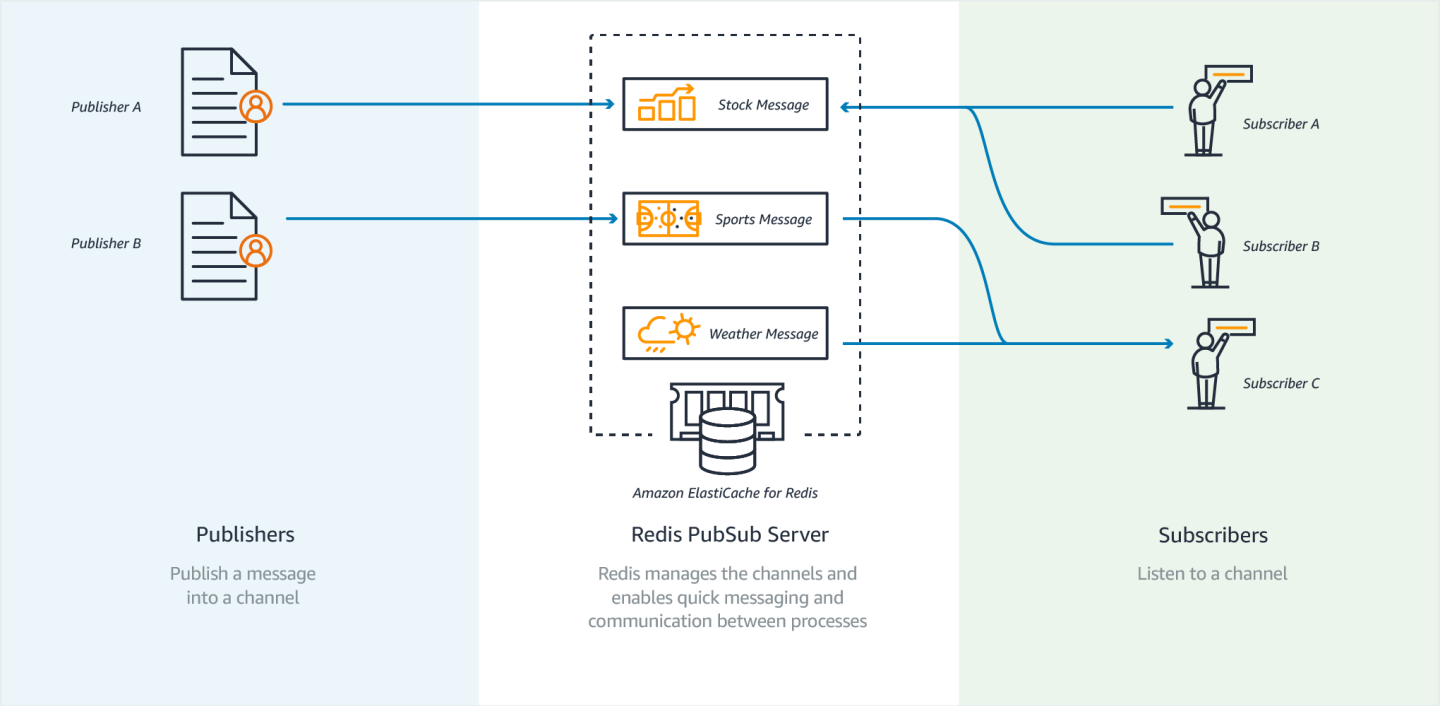
```
ZREVRANK leaderboard June
1
```

有关更多信息，请参阅有关排序集的 Redis 文档。

消息发送 (Redis Pub/Sub)

当您发送电子邮件时，可将它发送给一个或多个指定收件人。在 pub/sub 范式中，您在不知道接收者 (如果有) 的情况下将消息发送到特定频道。消息的接收者是订阅该频道的人。例如，假设您订阅了 news.sports.golf 频道。您和所有其他订阅 news.sports.golf 频道的人会收到所有发布到 news.sports.golf 的消息。

Redis pub/sub 功能与任何密钥空间都没有关联。因此，它不会影响任何级别。在下图中，您可以找到 ElastiCache for Redis 消息收发的示意图。



订阅

要接收某个频道的消息，请订阅该频道。您可以订阅单个频道、多个指定频道或者与某个模式匹配的所有频道。要取消订阅，您可以取消订阅在订阅时指定的频道。或者，如果您使用模式匹配订阅，则可以使用之前使用的相同模式取消订阅。

Example - 订阅单个频道

要订阅单个频道，可以使用 SUBSCRIBE 命令并指定您要订阅的频道。在以下示例中，客户将订阅 news.sports.golf 频道。

```
SUBSCRIBE news.sports.golf
```

过段时间以后，客户使用 UNSUBSCRIBE 命令并指定要取消订阅的频道，来取消对该频道的订阅。

```
UNSUBSCRIBE news.sports.golf
```

Example - 订阅多个指定的频道

要订阅多个特定频道，请使用 SUBSCRIBE 命令列出频道。在以下示例中，客户将订阅 news.sports.golf、news.sports.soccer 和 news.sports.skiing 频道。

```
SUBSCRIBE news.sports.golf news.sports.soccer news.sports.skiing
```

要取消对特定频道的订阅，请使用 UNSUBSCRIBE 命令并指定要取消订阅的频道。

```
UNSUBSCRIBE news.sports.golf
```

要取消对多个频道的订阅，请使用 UNSUBSCRIBE 命令并指定要取消订阅的频道。

```
UNSUBSCRIBE news.sports.golf news.sports.soccer
```

要取消所有订阅，请使用 UNSUBSCRIBE 并指定每个频道。或使用 UNSUBSCRIBE 并且无需指定频道。

```
UNSUBSCRIBE news.sports.golf news.sports.soccer news.sports.skiing
```

或者

```
UNSUBSCRIBE
```

Example - 使用模式匹配进行订阅

客户可以使用 PSUBSCRIBE 命令订阅与某个模式匹配的所有频道。

在以下示例中，客户将订阅所有体育频道。您不需要像使用 SUBSCRIBE 那样单个列出所有体育频道。相反，使用 PSUBSCRIBE 命令，您可以使用模式匹配。

```
PSUBSCRIBE news.sports.*
```

Example 取消订阅

要取消对这些频道的订阅，请使用 PUNSUBSCRIBE 命令。

```
PUNSUBSCRIBE news.sports.*
```

> **Important**
>
> 发送到 [P]SUBSCRIBE 命令与 [P]UNSUBSCRIBE 命令的频道字符串必须匹配。您不能 PSUBSCRIBE 到 new.* 和从 news.sports*PUNSUBSCRIBE 或从 news.sports.golfUNSUBSCRIBE。

发布

要向某个频道的所有订阅者发送消息，请使用 PUBLISH 命令，并指定频道和消息。以下示例将发布一条消息：“It's Saturday and sunny。I'm headed to the links.” 到 news.sports.golf 频道。

```
PUBLISH news.sports.golf "It's Saturday and sunny. I'm headed to the links."
```

客户无法向其订阅的频道发布消息。

有关更多信息，请参阅 Redis 文档中的 Pub/Sub。

推荐数据（Redis 哈希）

在 Redis 中使用 INCR 或 DECR 简化了编译推荐。每当用户对产品“给予好评”时，您就会增加一个 item:productID:like 计数器。每当用户对产品“给予差评”时，您就会增加一个 item:productID:dislike 计数器。借助 Redis 哈希，您还可以维护一个已对产品给予好评或差评的人员的列表。

Example - 给予好评和给予差评

```
INCR item:38923:likes
HSET item:38923:ratings Susan 1
```

```
INCR item:38923:dislikes
HSET item:38923:ratings Tommy -1
```

其他 Redis 用法

Salvatore Sanfilippo 的博客文章 How to take advantage of Redis just adding it to your stack（如何充分利用刚刚添加到您堆栈的 Redis）讨论了许多常见的数据库疑虑以及如何使用 Redis 轻松解决这些问题。此方法从数据库中删除负载并提高性能。

ElastiCache 客户评价

要了解像 Airbnb、PBS、Esri 这样的企业和其他使用 Amazon ElastiCache 的企业如何改善客户体验，从而拓展业务，请参阅其他人如何使用 Amazon ElastiCache。

您还可以观看教程视频以了解更多 ElastiCache 客户使用案例。

开始使用适用于 Redis ElastiCache 的 Amazon

使用本节中的动手教程来帮助您入门并了解有关 Redis ElastiCache 的更多信息。

主题

设置

要进行设置，请执行 ElastiCache以下操作：

主题

注册获取 AWS 账户

如果您没有 AWS 账户，请完成以下步骤来创建一个。

要注册 AWS 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，将接到一通电话，要求使用电话键盘输入一个验证码。

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

AWS 注册过程完成后会向您发送一封确认电子邮件。在任何时候，您都可以通过转至 https://aws.amazon.com/ 并选择我的账户来查看当前的账户活动并管理您的账户。

创建具有管理访问权限的用户

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

保护你的 AWS 账户根用户

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

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

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

创建具有管理访问权限的用户

1. 启用 IAM Identity Center

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

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

以具有管理访问权限的用户身份登录

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

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

将访问权限分配给其他用户

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

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的创建权限集。
2. 将用户分配到一个组，然后为该组分配单点登录访问权限。

   有关说明，请参阅《AWS IAM Identity Center 用户指南》中的添加组。

授权以编程方式访问

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

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
人力身份 （在 IAM Identity Center 中管理的用户）	使用临时证书签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关的 AWS CLI，请参阅《AWS Command Line Interface 用户指南》 AWS IAM Identity Center中的 “配置 AWS CLI 要使用”。 • 有关 AWS 软件开发工具包、工具和 AWS API，请参阅《软件开发工具包和AWS 工具参考指南》中的 IAM 身份中心身份验证。

哪个用户需要编程式访问权限？	目的	方式
IAM	使用临时证书签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照 IAM 用户指南中的将临时证书与 AWS 资源配合使用中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署向 AWS CLI、 AWS 软件开发工具包或 AWS API 发出的编程请求。	按照您希望使用的界面的说明进行操作。 • 有关信息 AWS CLI，请参阅用户指南中的使用 IAM 用户证书进行身份验证。AWS Command Line Interface • 有关 AWS SDK 和工具，请参阅 S AWS DK 和工具参考指南中的使用长期凭证进行身份验证。 • 有关 AWS API，请参阅 IAM 用户指南中的管理 IAM 用户的访问密钥。

相关主题:

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

设置您的权限（仅限新 ElastiCache 用户）

要提供访问权限，请为您的用户、组或角色添加权限：

- 中的用户和群组 AWS IAM Identity Center：

  创建权限集合。按照《AWS IAM Identity Center 用户指南》中创建权限集的说明进行操作。

- 通过身份提供商在 IAM 中托管的用户：

创建适用于身份联合验证的角色。按照《IAM 用户指南》中为第三方身份提供商创建角色（联合身份验证）的说明进行操作。

- IAM 用户：
  - 创建您的用户可以担任的角色。按照《IAM 用户指南》中为 IAM 用户创建角色的说明进行操作。
  - （不推荐使用）将策略直接附加到用户或将用户添加到用户组。按照《IAM 用户指南》中向用户添加权限（控制台）中的说明进行操作。

Amazon ElastiCache 创建并使用服务相关角色代表您配置 AWS 资源和访问其他资源和服务。 ElastiCache 要为您创建服务相关角色，请使用名为的 AWS托管策略。`AmazonElastiCacheFullAccess`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，请确保您具有调用权限`iam:createServiceLinkedRole`或已创建 ElastiCache服务相关角色。

有关更多信息，请参阅下列内容：

- 创建新策略（IAM）
- 适用于 Amazon ElastiCache 的 AWS 托管策略
- 将服务相关角色用于 Amazon ElastiCache

设置 EC2

您需要设置一个 EC2 实例，并从该实例连接到缓存。

- 如果您还没有 EC2 实例，请在此处了解如何设置 EC2 实例：EC2 入门。
- 您的 EC2 实例必须与缓存位于同一 VPC，并具有相同的安全组设置。默认情况下，Amazon ElastiCache 会在您的默认 VPC 中创建缓存并使用默认安全组。在学习本教程时，请确保您的 EC2 实例位于默认 VPC 中并且具有默认安全组。

授予从 Amazon VPC 安全组到您的缓存的网络访问权限

ElastiCache 自行设计的集群使用端口 6379 来执行 Redis 命令，而 ElastiCache 无服务器则同时使用端口 6379 和端口 6380。为了成功连接并从您的 EC2 实例执行 Redis 命令，您的安全组必须允许根据需要访问这些端口。

1. 登录 AWS Command Line Interface 并打开 Amazon EC2 控制台。
2. 在导航窗格中的 Network & Security 下，选择 Security Groups。
3. 从安全组列表中，为 Amazon VPC 选择安全组。除非您创建了供 ElastiCache 使用的安全组，否则该安全组将被命名为 de fault。
4. 选择“入站”选项卡，然后：
   a. 选择 Edit（编辑）。
   b. 选择 添加规则。
   c. 在“类型”列中，选择自定义 TCP 规则。
   d. 在端口范围框中，键入 6379。
   e. 在源框中，选择端口范围为（0.0.0.0/0）的任何位置，这样，从您 Amazon VPC 中启动的任何 Amazon EC2 实例都可以连接到您的缓存。
   f. 如果您使用的是 ElastiCache 无服务器，请通过选择添加规则来添加其他规则。
   g. 在 Type 列中，选择 Custom TCP rule。
   h. 在端口范围框中，键入 6380。
   i. 在源框中，选择端口范围为（0.0.0.0/0）的任何位置，这样，从您 Amazon VPC 中启动的任何 Amazon EC2 实例都可以连接到您的缓存。
   j. 选择保存

下载并设置 redis-cli

1. 使用您选择的连接实用工具连接到 Amazon EC2 实例。有关如何连接到 Amazon EC2 实例的说明，请参阅 Amazon EC2 入门指南。
2. 根据您的设置运行相应命令，下载并安装 redis-cli 实用工具。

   Amazon Linux 2023

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

   Amazon Linux 2

   ```
   sudo amazon-linux-extras install epel -y
   sudo yum install gcc jemalloc-devel openssl-devel tcl tcl-devel -y
   sudo wget http://download.redis.io/redis-stable.tar.gz
   ```

```
sudo tar xvzf redis-stable.tar.gz
cd redis-stable
sudo make BUILD_TLS=yes
```

> **Note**
>
> - 当您安装 redis6 程序包时，它会安装 redis6-cli 并默认提供加密支持。
> - 安装 redis-cli 时，必须为 TLS 提供构建支持。 ElastiCache 只有启用 TLS 后，才能访问无服务器。
> - 如果您要连接到未加密的集群，则不需要 `Build_TLS=yes` 选项。

第 1 步：创建缓存

在此步骤中，您将在 Amazon ElastiCache 中创建一个新的缓存。

AWS Management Console

要使用 ElastiCache 控制台创建新缓存，请执行以下操作：

1. 登录到 AWS Management Console 并打开 https://console.aws.amazon.com/connect/。
2. 在控制台左侧的导航窗格中，选择 Redis 缓存。
3. 在控制台的右侧，选择创建 Redis 缓存。
4. 在缓存设置中输入名称。您还可以选择为缓存输入描述。
5. 保持选中默认设置。
6. 单击创建以创建缓存。
7. 缓存处于“活动”状态后，您可以开始在缓存上写入和读取数据。

AWS CLI

以下 AWS CLI 示例使用 create-serverless-cache 创建新缓存。

Linux

```
aws elasticache create-serverless-cache \
    --serverless-cache-name CacheName \
```

```
    --engine redis
```

Windows

```
aws elasticache create-serverless-cache ^
    --serverless-cache-name CacheName ^
    --engine redis
```

请注意，“状态”字段的值设置为 CREATING。

要验证 ElastiCache 是否已完成缓存创建过程，请使用 describe-serverless-caches 命令。

Linux

```
aws elasticache describe-serverless-caches --serverless-cache-name CacheName
```

Windows

```
aws elasticache describe-serverless-caches --serverless-cache-name CacheName
```

创建新缓存后，继续执行第 2 步：对缓存读取和写入数据。

第 2 步：对缓存读取和写入数据

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。

此部分还假设您已为从中连接到缓存的 EC2 实例设定 VPC 访问权限和安全组设置，并已在 EC2 实例上设置 redis-cli。有关该步骤的更多信息，请参阅设置。

查找缓存端点

AWS Management Console

要使用 ElastiCache 控制台查找缓存的终端节点，请执行以下操作：

1. 登录 AWS Management Console 并打开亚马逊 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在控制台左侧的导航窗格中，选择 Redis 缓存。
3. 在控制台的右侧，单击刚刚创建的缓存的名称。

4. 在缓存详细信息中，找到并复制缓存端点。

AWS CLI

以下 AWS CLI 示例说明如何使用 describe-serverless-caches 命令查找新缓存的终端节点。运行命令后，查找“端点”字段。

Linux

```
aws elasticache describe-serverless-caches \
  --serverless-cache-name CacheName
```

Windows

```
aws elasticache describe-serverless-caches ^
  --serverless-cache-name CacheName
```

连接到您的 Redis 缓存（Linux）

现在您有了所需的端点，便可以登录到您的 EC2 实例并连接到集群。在以下示例中，您使用 redis-cli 实用工具连接到集群。可使用以下命令连接到缓存（注：将 cache-endpoint 替换为您在上一步中检索到的端点）。

```
src/redis-cli -h cache-endpoint --tls -p 6379
set a "hello"          // Set key "a" with a string value and no expiration
OK
get a                  // Get value for key "a"
"hello"
```

连接到您的 Redis 缓存（Windows）

现在您有了所需的端点，便可以登录到您的 EC2 实例并连接到集群。在以下示例中，您使用 redis-cli 实用工具连接到集群。可使用以下命令连接到缓存。打开命令提示符，切换到 Redis 目录并运行命令（注：将 Cache_Endpoint 替换为您在上一步中检索到的端点）。

```
c:\Redis>redis-cli -h Redis_Cluster_Endpoint --tls -p 6379
set a "hello"          // Set key "a" with a string value and no expiration
OK
get a                  // Get value for key "a"
"hello"
```

您现在可以继续执行第 3 步：（可选）清除。

第 3 步：（可选）清除

如果您不再需要已创建的 Amazon ElastiCache 缓存，可以将其删除。此步骤有助于确保不会为未使用的资源付费。您可以使用 ElastiCache 控制台、AWS CLI 或 ElastiCache API 删除缓存。

AWS Management Console

要使用控制台删除缓存，请执行以下操作：

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 在控制台左侧的导航窗格中，选择 Redis 缓存。
3. 选择要删除的缓存旁边的单选按钮。
4. 选择右上角的操作，然后选择删除。
5. （可选）您可以选择在删除缓存之前拍摄最终快照。
6. 在删除确认屏幕上，重新输入缓存名称并发选择删除以删除集群，或选择取消以保留集群。

当缓存进入正在删除状态之后，您便不再需要支付缓存费用。

AWS CLI

以下 AWS CLI 示例使用 delete-serverless-cache 命令删除缓存。

Linux

```
aws elasticache delete-serverless-cache \
  --serverless-cache-name CacheName
```

Windows

```
aws elasticache delete-serverless-cache ^
  --serverless-cache-name CacheName
```

请注意，状态字段的值设置为正在删除。

您现在可以继续执行后续步骤。

后续步骤

有关 ElastiCache 的更多信息，请参阅以下页面：

- 与 ElastiCache
- 针对 Redis ElastiCache 进行扩展
- Amazon ElastiCache 中的日志记录和监控
- ElastiCache 最佳实践和缓存策略
- 快照和还原
- Amazon SNS 监控 ElastiCache 事件

ElastiCache 和 AWS 开发工具包入门

本节包含帮助您了解 Amazon ElastiCache 的实践教程。我们鼓励您通读一份特定于语言的教程。

> **Note**
>
> AWS SDK 适用于各种语言。有关完整列表，请参阅用于 Amazon Web Services 的工具。

Python 和 ElastiCache

在本教程中，您可以使用 AWS SDK for Python (Boto3) 编写几个简单的程序，以执行以下 ElastiCache 操作：

- 创建 ElastiCache 集群（已启用集群模式和已禁用集群模式）
- 检查用户或用户组是否存在，如不存在，请创建用户或用户组（仅限 Redis 6.0 及以上版本）
- 连接到 ElastiCache
- 执行操作，如设置和获取字符串，读取和写入流，以及从 Pub/Sub 频道发布和订阅。

在学习本教程时，您可以参考 AWS SDK for Python (Boto) 文档。以下部分特定于 ElastiCache：ElastiCache 低级别客户端

教程的先决条件

- 设置 AWS 访问密钥使用 AWS SDK。有关更多信息，请参阅设置。

- 安装 Python 3.0 或更高版本。有关更多信息，请参阅 https://www.python.org/downloads。有关说明，请参阅 Boto 3 文档中的快速入门。

创建 ElastiCache 集群和用户

以下示例使用 boto3 SDK 进行 ElastiCache 管理操作（创建集群或用户），使用 redis-p redis-py-cluster y/ 进行数据处理。

主题

- 创建已禁用集群模式的集群
- 创建采用 TLS 和 RBAC 且已禁用集群模式的集群
- 创建已启用集群模式的集群
- 创建采用 TLS 和 RBAC 且已启用集群模式的集群
- 检查用户/用户组是否存在，如不存在，请创建用户/用户组

创建已禁用集群模式的集群

复制以下程序并将其粘贴到名为 CreateClusterModeDisabledCluster.py 的文件中。

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def
 create_cluster_mode_disabled(CacheNodeType='cache.t3.small',EngineVersion='6.0',NumCacheCluste
 cache cluster',ReplicationGroupId=None):
    """Creates an ElastiCache Cluster with cluster mode disabled

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
 cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/
CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
 used.
```

```
    :param NumCacheClusters: Number of nodes in the cluster. Minimum 1 (just a primary
 node) and maximun 6 (1 primary and 5 replicas).
    If not specified, cluster will be created with 1 primary and 1 replica.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Name for the cluster
    :return: dictionary with the API results

    """
    if not ReplicationGroupId:
        return 'ReplicationGroupId parameter is required'

    response = client.create_replication_group(
        AutomaticFailoverEnabled=True,
        CacheNodeType=CacheNodeType,
        Engine='redis',
        EngineVersion=EngineVersion,
        NumCacheClusters=NumCacheClusters,
        ReplicationGroupDescription=ReplicationGroupDescription,
        ReplicationGroupId=ReplicationGroupId,
        SnapshotRetentionLimit=30,
    )
    return response


if __name__ == '__main__':

    # Creates an ElastiCache Cluster mode disabled cluster, based on cache.m6g.large
 nodes, Redis 6, one primary and two replicas
    elasticacheResponse = create_cluster_mode_disabled(
        #CacheNodeType='cache.m6g.large',
        EngineVersion='6.0',
        NumCacheClusters=3,
        ReplicationGroupDescription='Redis cluster mode disabled with replicas',
        ReplicationGroupId='redis202104053'
        )

    logging.info(elasticacheResponse)
```

要运行该程序，请输入以下命令：

```
python CreateClusterModeDisabledCluster.py
```

有关更多信息，请参阅 管理集群。

创建采用 TLS 和 RBAC 且已禁用集群模式的集群

为确保安全性，在创建已禁用集群模式的集群时，您可以使用传输层安全性 (TLS) 和基于角色的访问控制 (RBAC)。与 Redis AUTH（如果对客户端令牌进行身份验证，则所有经过身份验证的客户端都对复制组具有完全的访问权限）不同的是，RBAC 使您能够通过用户组控制集群访问权限。这些用户组旨在作为一种管理对复制组的访问权限的方式。有关更多信息，请参阅 基于角色的访问控制（RBAC）。

复制以下程序并将其粘贴到名为 ClusterModeDisabledWithRBAC.py 的文件中。

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def
 create_cluster_mode_disabled_rbac(CacheNodeType='cache.t3.small',EngineVersion='6.0',NumCacheC
 cache cluster',ReplicationGroupId=None, UserGroupIds=None,
 SecurityGroupIds=None,CacheSubnetGroupName=None):
    """Creates an ElastiCache Cluster with cluster mode disabled and RBAC

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
 cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/
CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
 used.
    :param NumCacheClusters: Number of nodes in the cluster. Minimum 1 (just a primary
 node) and maximun 6 (1 primary and 5 replicas).
    If not specified, cluster will be created with 1 primary and 1 replica.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Mandatory name for the cluster.
    :param UserGroupIds: The ID of the user group to be assigned to the cluster.
    :param SecurityGroupIds: List of security groups to be assigned. If not defined,
 default will be used
    :param CacheSubnetGroupName: subnet group where the cluster will be placed. If not
 defined, default will be used.
    :return: dictionary with the API results

    """
```

```
    if not ReplicationGroupId:
        return {'Error': 'ReplicationGroupId parameter is required'}
    elif not isinstance(UserGroupIds,(list)):
        return {'Error': 'UserGroupIds parameter is required and must be a list'}

    params={'AutomaticFailoverEnabled': True,
            'CacheNodeType': CacheNodeType,
            'Engine': 'redis',
            'EngineVersion': EngineVersion,
            'NumCacheClusters': NumCacheClusters,
            'ReplicationGroupDescription': ReplicationGroupDescription,
            'ReplicationGroupId': ReplicationGroupId,
            'SnapshotRetentionLimit': 30,
            'TransitEncryptionEnabled': True,
            'UserGroupIds':UserGroupIds
        }

    # defaults will be used if CacheSubnetGroupName or SecurityGroups are not explicit.
    if isinstance(SecurityGroupIds,(list)):
        params.update({'SecurityGroupIds':SecurityGroupIds})
    if CacheSubnetGroupName:
        params.update({'CacheSubnetGroupName':CacheSubnetGroupName})

    response = client.create_replication_group(**params)
    return response

if __name__ == '__main__':

    # Creates an ElastiCache Cluster mode disabled cluster, based on cache.m6g.large
 nodes, Redis 6, one primary and two replicas.
    # Assigns the existent user group "mygroup" for RBAC authentication

    response=create_cluster_mode_disabled_rbac(
        CacheNodeType='cache.m6g.large',
        EngineVersion='6.0',
        NumCacheClusters=3,
        ReplicationGroupDescription='Redis cluster mode disabled with replicas',
        ReplicationGroupId='redis202104',
        UserGroupIds=[
            'mygroup'
        ],
        SecurityGroupIds=[
            'sg-7cc73803'
        ],
```

```
        CacheSubnetGroupName='default'
    )

    logging.info(response)
```

要运行该程序，请输入以下命令：

```
python ClusterModeDisabledWithRBAC.py
```

有关更多信息，请参阅 管理集群。

创建已启用集群模式的集群

复制以下程序并将其粘贴到名为 ClusterModeEnabled.py 的文件中。

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def
 create_cluster_mode_enabled(CacheNodeType='cache.t3.small',EngineVersion='6.0',NumNodeGroups=1
 ReplicationGroupDescription='Sample cache with cluster mode
 enabled',ReplicationGroupId=None):
    """Creates an ElastiCache Cluster with cluster mode enabled

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
 cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/
CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
 used.
    :param NumNodeGroups: Number of shards in the cluster. Minimum 1 and maximun 90.
    If not specified, cluster will be created with 1 shard.
    :param ReplicasPerNodeGroup: Number of replicas per shard. If not specified 1
 replica per shard will be created.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Name for the cluster
    :return: dictionary with the API results

    """
```

```
    if not ReplicationGroupId:
        return 'ReplicationGroupId parameter is required'

    response = client.create_replication_group(
        AutomaticFailoverEnabled=True,
        CacheNodeType=CacheNodeType,
        Engine='redis',
        EngineVersion=EngineVersion,
        ReplicationGroupDescription=ReplicationGroupDescription,
        ReplicationGroupId=ReplicationGroupId,
    #   Creates a cluster mode enabled cluster with 1 shard(NumNodeGroups), 1 primary
 node (implicit) and 2 replicas (replicasPerNodeGroup)
        NumNodeGroups=NumNodeGroups,
        ReplicasPerNodeGroup=ReplicasPerNodeGroup,
        CacheParameterGroupName='default.redis6.0.cluster.on'
    )

    return response


# Creates a cluster mode enabled
response = create_cluster_mode_enabled(
    CacheNodeType='cache.m6g.large',
    EngineVersion='6.0',
    ReplicationGroupDescription='Redis cluster mode enabled with replicas',
    ReplicationGroupId='redis20210',
#   Creates a cluster mode enabled cluster with 1 shard(NumNodeGroups), 1 primary
 (implicit) and 2 replicas (replicasPerNodeGroup)
    NumNodeGroups=2,
    ReplicasPerNodeGroup=1,
)

logging.info(response)
```

要运行该程序，请输入以下命令：

```
python ClusterModeEnabled.py
```

有关更多信息，请参阅 管理集群。

创建采用 TLS 和 RBAC 且已启用集群模式的集群

为确保安全性，在创建已启用集群模式的集群时，您可以使用传输层安全性 (TLS) 和基于角色的访问控制 (RBAC)。与 Redis AUTH（如果对客户端令牌进行身份验证，则所有经过身份验证的客户端

都对复制组具有完全的访问权限）不同的是，RBAC 使您能够通过用户组控制集群访问权限。这些用户组旨在作为一种管理对复制组的访问权限的方式。有关更多信息，请参阅 基于角色的访问控制（RBAC）。

复制以下程序并将其粘贴到名为 ClusterModeEnabledWithRBAC.py 的文件中。

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def
 create_cluster_mode_enabled(CacheNodeType='cache.t3.small',EngineVersion='6.0',NumNodeGroups=1
 ReplicationGroupDescription='Sample cache with cluster
 mode enabled',ReplicationGroupId=None,UserGroupIds=None,
 SecurityGroupIds=None,CacheSubnetGroupName=None,CacheParameterGroupName='default.redis6.0.clus
    """Creates an ElastiCache Cluster with cluster mode enabled and RBAC

    Returns a dictionary with the API response

    :param CacheNodeType: Node type used on the cluster. If not specified,
 cache.t3.small will be used
    Refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/
CacheNodes.SupportedTypes.html for supported node types
    :param EngineVersion: Engine version to be used. If not specified, latest will be
 used.
    :param NumNodeGroups: Number of shards in the cluster. Minimum 1 and maximun 90.
    If not specified, cluster will be created with 1 shard.
    :param ReplicasPerNodeGroup: Number of replicas per shard. If not specified 1
 replica per shard will be created.
    :param ReplicationGroupDescription: Description for the cluster.
    :param ReplicationGroupId: Name for the cluster.
    :param CacheParameterGroupName: Parameter group to be used. Must be compatible with
 the engine version and cluster mode enabled.
    :return: dictionary with the API results

    """
    if not ReplicationGroupId:
        return 'ReplicationGroupId parameter is required'
    elif not isinstance(UserGroupIds,(list)):
        return {'Error': 'UserGroupIds parameter is required and must be a list'}
```

```
    params={'AutomaticFailoverEnabled': True,
            'CacheNodeType': CacheNodeType,
            'Engine': 'redis',
            'EngineVersion': EngineVersion,
            'ReplicationGroupDescription': ReplicationGroupDescription,
            'ReplicationGroupId': ReplicationGroupId,
            'SnapshotRetentionLimit': 30,
            'TransitEncryptionEnabled': True,
            'UserGroupIds':UserGroupIds,
            'NumNodeGroups': NumNodeGroups,
            'ReplicasPerNodeGroup': ReplicasPerNodeGroup,
            'CacheParameterGroupName': CacheParameterGroupName
        }

    # defaults will be used if CacheSubnetGroupName or SecurityGroups are not explicit.
    if isinstance(SecurityGroupIds,(list)):
        params.update({'SecurityGroupIds':SecurityGroupIds})
    if CacheSubnetGroupName:
        params.update({'CacheSubnetGroupName':CacheSubnetGroupName})

    response = client.create_replication_group(**params)
    return response

if __name__ == '__main__':
    # Creates a cluster mode enabled cluster
    response = create_cluster_mode_enabled(
        CacheNodeType='cache.m6g.large',
        EngineVersion='6.0',
        ReplicationGroupDescription='Redis cluster mode enabled with replicas',
        ReplicationGroupId='redis2021',
    #   Creates a cluster mode enabled cluster with 1 shard(NumNodeGroups), 1 primary
 (implicit) and 2 replicas (replicasPerNodeGroup)
        NumNodeGroups=2,
        ReplicasPerNodeGroup=1,
        UserGroupIds=[
            'mygroup'
        ],
        SecurityGroupIds=[
            'sg-7cc73803'
        ],
        CacheSubnetGroupName='default'

    )
```

```
    logging.info(response)
```

要运行该程序，请输入以下命令：

```
python ClusterModeEnabledWithRBAC.py
```

有关更多信息，请参阅 管理集群。

检查用户/用户组是否存在，如不存在，请创建用户/用户组

通过 RBAC，您可以使用访问字符串创建用户并为其分配特定权限。您可以将用户分配到与特定角色（管理员、人力资源）对应的用户组，然后将这些用户组部署到一个或多 ElastiCache 个 Redis 复制组中。这样，您可以在使用相同 Redis 复制组的客户端之间建立安全边界，并阻止客户端彼此访问数据。有关更多信息，请参阅 基于角色的访问控制（RBAC）。

复制以下程序并将其粘贴到名为 UserAndUserGroups.py 的文件中。更新用于提供凭证的机制。此示例中的凭证显示为可替换凭证，并分配了未声明的项目。避免对凭证进行硬编码。

```
import boto3
import logging

logging.basicConfig(level=logging.INFO)
client = boto3.client('elasticache')

def check_user_exists(UserId):
    """Checks if UserId exists

    Returns True if UserId exists, otherwise False
    :param UserId: ElastiCache User ID
    :return: True|False
    """
    try:
        response = client.describe_users(
            UserId=UserId,
        )
        if response['Users'][0]['UserId'].lower() == UserId.lower():
            return True
    except Exception as e:
        if e.response['Error']['Code'] == 'UserNotFound':
            logging.info(e.response['Error'])
            return False
        else:
            raise
```

```
def check_group_exists(UserGroupId):
    """Checks if UserGroupID exists

    Returns True if Group ID exists, otherwise False
    :param UserGroupId: ElastiCache User ID
    :return: True|False
    """

    try:
        response = client.describe_user_groups(
            UserGroupId=UserGroupId
        )
        if response['UserGroups'][0]['UserGroupId'].lower() == UserGroupId.lower():
            return True
    except Exception as e:
        if e.response['Error']['Code'] == 'UserGroupNotFound':
            logging.info(e.response['Error'])
            return False
        else:
            raise

def create_user(UserId=None,UserName=None,Password=None,AccessString=None):
    """Creates a new user

    Returns the ARN for the newly created user or the error message
    :param UserId: ElastiCache user ID. User IDs must be unique
    :param UserName: ElastiCache user name. ElastiCache allows multiple users with the
 same name as long as the associated user ID is unique.
    :param Password: Password for user. Must have at least 16 chars.
    :param AccessString: Access string with the permissions for the user. For
 details refer to https://docs.aws.amazon.com/AmazonElastiCache/latest/red-ug/
Clusters.RBAC.html#Access-string
    :return: user ARN
    """
    try:
        response = client.create_user(
            UserId=UserId,
            UserName=UserName,
            Engine='Redis',
            Passwords=[Password],
            AccessString=AccessString,
            NoPasswordRequired=False
        )
```

```
        return response['ARN']
    except Exception as e:
        logging.info(e.response['Error'])
        return e.response['Error']

def create_group(UserGroupId=None, UserIds=None):
    """Creates a new group.
    A default user is required (mandatory) and should be specified in the UserIds list

    Return: Group ARN
    :param UserIds: List with user IDs to be associated with the new group. A default
 user is required
    :param UserGroupId: The ID (name) for the group
    :return: Group ARN
    """
    try:
        response = client.create_user_group(
            UserGroupId=UserGroupId,
            Engine='Redis',
            UserIds=UserIds
        )
        return response['ARN']
    except Exception as e:
        logging.info(e.response['Error'])


if __name__ == '__main__':

    groupName='mygroup2'
    userName = 'myuser2'
    userId=groupName+'-'+userName

    # Creates a new user if the user ID does not exist.
    for tmpUserId,tmpUserName in [ (userId,userName), (groupName+'-
default','default')]:
        if not check_user_exists(tmpUserId):
            response=create_user(UserId=tmpUserId,
 UserName=EXAMPLE,Password=EXAMPLE,AccessString='on ~* +@all')
            logging.info(response)
        # assigns the new user ID to the user group
    if not check_group_exists(groupName):
        UserIds = [ userId , groupName+'-default']
        response=create_group(UserGroupId=groupName,UserIds=UserIds)
        logging.info(response)
```

要运行该程序，请输入以下命令：

```
python UserAndUserGroups.py
```

连接到 ElastiCache

以下示例使用 Redis 客户端连接到 ElastiCache。

主题

- 连接到已禁用集群模式的集群
- 连接到已启用集群模式的集群

连接到已禁用集群模式的集群

复制以下程序并粘贴到名为 ConnectClusterModeDisabled.py 的文件中。更新用于提供凭证的机制。此示例中的凭证显示为可替换凭证，并分配了未声明的项目。避免对凭证进行硬编码。

```
from redis import Redis
import logging

logging.basicConfig(level=logging.INFO)
redis = Redis(host='primary.xxx.yyyyyy.zzz1.cache.amazonaws.com', port=6379,
 decode_responses=True, ssl=True, username=example, password=EXAMPLE)

if redis.ping():
    logging.info("Connected to Redis")
```

要运行该程序，请输入以下命令：

```
python ConnectClusterModeDisabled.py
```

连接到已启用集群模式的集群

复制以下程序并粘贴到名为 ConnectClusterModeEnabled.py 的文件中。

```
from rediscluster import RedisCluster
import logging
```

```
logging.basicConfig(level=logging.INFO)
redis = RedisCluster(startup_nodes=[{"host":
 "xxx.yyy.clustercfg.zzz1.cache.amazonaws.com","port": "6379"}],
 decode_responses=True,skip_full_coverage_check=True)

if redis.ping():
    logging.info("Connected to Redis")
```

要运行该程序，请输入以下命令：

```
python ConnectClusterModeEnabled.py
```

用法示例

以下示例使用 boto3 SDK for ElastiCache 来处理 ElastiCache。

主题

- 设置和获取字符串
- 设置和获取包含多个项目的哈希
- 从 Pub/Sub 频道发布（写入）和订阅（读取）
- 从流中写入和读取

设置和获取字符串

复制以下程序并粘贴到名为 SetAndGetStrings.py 的文件中。

```
import time
import logging
logging.basicConfig(level=logging.INFO,format='%(asctime)s: %(message)s')

keyName='mykey'
currTime=time.ctime(time.time())

# Set the key 'mykey' with the current date and time as value.
# The Key will expire and removed from cache in 60 seconds.
redis.set(keyName, currTime, ex=60)

# Sleep just for better illustration of TTL (expiration) value
time.sleep(5)
```

```
# Retrieve the key value and current TTL
keyValue=redis.get(keyName)
keyTTL=redis.ttl(keyName)

logging.info("Key {} was set at {} and has {} seconds until expired".format(keyName,
 keyValue, keyTTL))
```

要运行该程序，请输入以下命令：

```
python SetAndGetStrings.py
```

设置和获取包含多个项目的哈希

复制以下程序并粘贴到名为 SetAndGetHash.py 的文件中。

```
import logging
import time

logging.basicConfig(level=logging.INFO,format='%(asctime)s: %(message)s')

keyName='mykey'
keyValues={'datetime': time.ctime(time.time()), 'epochtime': time.time()}

# Set the hash 'mykey' with the current date and time in human readable format
 (datetime field) and epoch number (epochtime field).
redis.hset(keyName, mapping=keyValues)

# Set the key to expire and removed from cache in 60 seconds.
redis.expire(keyName, 60)

# Sleep just for better illustration of TTL (expiration) value
time.sleep(5)

# Retrieves all the fields and current TTL
keyValues=redis.hgetall(keyName)
keyTTL=redis.ttl(keyName)

logging.info("Key {} was set at {} and has {} seconds until expired".format(keyName,
 keyValues, keyTTL))
```

要运行该程序，请输入以下命令：

```
python SetAndGetHash.py
```

从 Pub/Sub 频道发布（写入）和订阅（读取）

复制以下程序并粘贴到名为 PubAndSub.py 的文件中。

```
import logging
import time

def handlerFunction(message):
    """Prints message got from PubSub channel to the log output

    Return None
    :param message: message to log
    """
    logging.info(message)

logging.basicConfig(level=logging.INFO)
redis = Redis(host="redis202104053.tihewd.ng.0001.use1.cache.amazonaws.com", port=6379,
 decode_responses=True)


# Creates the subscriber connection on "mychannel"
subscriber = redis.pubsub()
subscriber.subscribe(**{'mychannel': handlerFunction})

# Creates a new thread to watch for messages while the main process continues with its
 routines
thread = subscriber.run_in_thread(sleep_time=0.01)

# Creates publisher connection on "mychannel"
redis.publish('mychannel', 'My message')

# Publishes several messages. Subscriber thread will read and print on log.
while True:
    redis.publish('mychannel',time.ctime(time.time()))
    time.sleep(1)
```

要运行该程序，请输入以下命令：

```
python PubAndSub.py
```

从流中写入和读取

复制以下程序并粘贴到名为 ReadWriteStream.py 的文件中。

```
from redis import Redis
import redis.exceptions as exceptions
import logging
import time
import threading

logging.basicConfig(level=logging.INFO)

def writeMessage(streamName):
    """Starts a loop writting the current time and thread name to 'streamName'

    :param streamName: Stream (key) name to write messages.
    """
    fieldsDict={'writerId':threading.currentThread().getName(),'myvalue':None}
    while True:
        fieldsDict['myvalue'] = time.ctime(time.time())
        redis.xadd(streamName,fieldsDict)
        time.sleep(1)

def readMessage(groupName=None,streamName=None):
    """Starts a loop reading from 'streamName'
    Multiple threads will read from the same stream consumer group. Consumer group is
 used to coordinate data distribution.
    Once a thread acknowleges the message, it won't be provided again. If message
 wasn't acknowledged, it can be served to another thread.

    :param groupName: stream group were multiple threads will read.
    :param streamName: Stream (key) name where messages will be read.
    """

    readerID=threading.currentThread().getName()
    while True:
        try:
            # Check if the stream has any message
            if redis.xlen(streamName)>0:
                # Check if if the messages are new (not acknowledged) or not (already
 processed)
                streamData=redis.xreadgroup(groupName,readerID,
{streamName:'>'},count=1)
                if len(streamData) > 0:
                    msgId,message = streamData[0][1][0]
                    logging.info("{}: Got {} from ID
 {}".format(readerID,message,msgId))
```

```
                    #Do some processing here. If the message has been processed
 sucessfuly, acknowledge it and (optional) delete the message.
                    redis.xack(streamName,groupName,msgId)
                    logging.info("Stream message ID {} read and processed successfuly
 by {}".format(msgId,readerID))
                    redis.xdel(streamName,msgId)
            else:
                pass
        except:
            raise

        time.sleep(0.5)

# Creates the stream 'mystream' and consumer group 'myworkergroup' where multiple
 threads will write/read.
try:
    redis.xgroup_create('mystream','myworkergroup',mkstream=True)
except exceptions.ResponseError as e:
    logging.info("Consumer group already exists. Will continue despite the error:
 {}".format(e))
except:
    raise

# Starts 5 writer threads.
for writer_no in range(5):
    writerThread = threading.Thread(target=writeMessage, name='writer-'+str(writer_no),
 args=('mystream',),daemon=True)
    writerThread.start()

# Starts 10 reader threads
for reader_no in range(10):
    readerThread = threading.Thread(target=readMessage, name='reader-'+str(reader_no),
 args=('myworkergroup','mystream',),daemon=True)
    readerThread.daemon = True
    readerThread.start()

# Keep the code running for 30 seconds
time.sleep(30)
```

要运行该程序，请输入以下命令：

```
python ReadWriteStream.py
```

教程：配置 Lambda 函数以在亚马逊 VPC ElastiCache 中访问亚马逊

在本教程中，您可以学习如何创建 ElastiCache 无服务器缓存，创建 Lambda 函数，然后测试 Lambda 函数，然后选择在之后进行清理。

主题

步骤 1：创建无服务器缓存

要创建无服务器缓存，请按照以下步骤操作。

主题

步骤 1.1：创建无服务器缓存

在此步骤中，您将使用 (AWS Command Line Interface CLI) 在您账户的 us-east-1 区域的默认 Amazon VPC 中创建无服务器缓存。有关使用 ElastiCache 控制台或 API 创建无服务器缓存的信息，请参阅第 1 步：创建缓存。

```
aws elasticache create-serverless-cache \
  --serverless-cache-name cache-01  \
--description "ElastiCache IAM auth application" \
--engine redis
```

请注意，“状态”字段的值设置为 CREATING。可能需要一分钟 ElastiCache 才能完成缓存的创建。

步骤 1.2：复制无服务器缓存端点

使用`describe-serverless-caches`命令验证 ElastiCache For Redis 是否已完成缓存的创建。

```
aws elasticache describe-serverless-caches \
--serverless-cache-name cache-01
```

复制输出中显示的端点地址。在为 Lambda 函数创建部署包时，您将需要此地址。

步骤 1.3：创建 IAM 角色

1. 为您的角色创建 IAM 信任政策文档，如下所示，允许您的账户承担新角色。将策略保存到名为 trust-policy.json 的文件中。

   ```
   {
   "Version": "2012-10-17",
       "Statement": [{
   "Effect": "Allow",
           "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
           "Action": "sts:AssumeRole"
       },
       {
         "Effect": "Allow",
         "Principal": {
           "Service": "lambda.amazonaws.com"
         },
         "Action": "sts:AssumeRole"
       }]
   }
   ```

2. 创建 IAM policy 文档，如下所示。将策略保存到名为 policy.json 的文件中。

   ```
   {
   "Version": "2012-10-17",
     "Statement": [
       {
   "Effect" : "Allow",
         "Action" : [
           "elasticache:Connect"
         ],
         "Resource" : [
           "arn:aws:elasticache:us-east-1:123456789012:serverlesscache:cache-01",
   ```

```
        "arn:aws:elasticache:us-east-1:123456789012:user:iam-user-01"
      ]
    }
  ]
}
```

3. 创建一个 IAM 角色。

```
aws iam create-role \
--role-name "elasticache-iam-auth-app" \
--assume-role-policy-document file://trust-policy.json
```

4. 创建 IAM policy。

```
aws iam create-policy \
  --policy-name "elasticache-allow-all" \
  --policy-document file://policy.json
```

5. 向角色附加 IAM policy。

```
aws iam attach-role-policy \
 --role-name "elasticache-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"
```

步骤 1.4：创建无服务器缓存

1. 创建新的默认用户。

```
aws elasticache create-user \
  --user-name default \
--user-id default-user-disabled \
--engine redis \
--authentication-mode Type=no-password-required \
--access-string "off +get ~keys*"
```

2. 创建启用 IAM 的新用户。

```
aws elasticache create-user \
  --user-name iam-user-01 \
--user-id iam-user-01 \
--authentication-mode Type=iam \
```

```
--engine redis \
--access-string "on ~* +@all"
```

3. 创建用户组并附加用户。

```
aws elasticache create-user-group \
  --user-group-id iam-user-group-01 \
--engine redis \
--user-ids default-user-disabled iam-user-01

aws elasticache modify-serverless-cache \
  --serverless-cache-name cache-01  \
--user-group-id iam-user-group-01
```

第 2 步：创建 Lambda 函数

要创建 Lambda 函数，请执行以下步骤。

主题

- 步骤 2.1：创建 Lambda 函数
- 步骤 2.2：创建 IAM 角色（执行角色）
- 步骤 2.3：上传部署包（创建 Lambda 函数）

步骤 2.1：创建 Lambda 函数

在本教程中，我们为您的 Lambda 函数提供了 Python 中的示例代码。

Python

以下示例 Python 代码读取项目并将其写入 ElastiCache 缓存。复制代码，并将其保存到名为 app.py 的文件中。请务必将代码中的elasticache_endpoint值替换为您在步骤 1.2 中复制的端点地址。

```
from typing import Tuple, Union
from urllib.parse import ParseResult, urlencode, urlunparse

import botocore.session
import redis
from botocore.model import ServiceId
from botocore.signers import RequestSigner
from cachetools import TTLCache, cached
```

```
import uuid

class ElastiCacheIAMProvider(redis.CredentialProvider):
    def __init__(self, user, cache_name, is_serverless=False, region="us-east-1"):
        self.user = user
        self.cache_name = cache_name
        self.is_serverless = is_serverless
        self.region = region

        session = botocore.session.get_session()
        self.request_signer = RequestSigner(
            ServiceId("elasticache"),
            self.region,
            "elasticache",
            "v4",
            session.get_credentials(),
            session.get_component("event_emitter"),
        )

    # Generated IAM tokens are valid for 15 minutes
    @cached(cache=TTLCache(maxsize=128, ttl=900))
    def get_credentials(self) -> Union[Tuple[str], Tuple[str, str]]:
        query_params = {"Action": "connect", "User": self.user}
        if self.is_serverless:
            query_params["ResourceType"] = "ServerlessCache"
        url = urlunparse(
            ParseResult(
                scheme="https",
                netloc=self.cache_name,
                path="/",
                query=urlencode(query_params),
                params="",
                fragment="",
            )
        )
        signed_url = self.request_signer.generate_presigned_url(
            {"method": "GET", "url": url, "body": {}, "headers": {}, "context": {}},
            operation_name="connect",
            expires_in=900,
            region_name=self.region,
        )
        # RequestSigner only seems to work if the URL has a protocol, but
        # Elasticache only accepts the URL without a protocol
        # So strip it off the signed URL before returning
```

```
        return (self.user, signed_url.removeprefix("https://"))

def lambda_handler(event, context):
    username = "iam-user-01" # replace with your user id
    cache_name = "cache-01" # replace with your cache name
    elasticache_endpoint = "cache-01-xxxxx.serverless.use1.cache.amazonaws.com" #
 replace with your cache endpoint
    creds_provider = ElastiCacheIAMProvider(user=username, cache_name=cache_name,
 is_serverless=True)
    redis_client = redis.Redis(host=elasticache_endpoint, port=6379,
 credential_provider=creds_provider, ssl=True, ssl_cert_reqs="none")

    key='uuid'
    # create a random UUID - this will be the sample element we add to the cache
    uuid_in = uuid.uuid4().hex
    redis_client.set(key, uuid_in)
    result = redis_client.get(key)
    decoded_result = result.decode("utf-8")
    # check the retrieved item matches the item added to the cache and print
    # the results
    if decoded_result == uuid_in:
        print(f"Success: Inserted {uuid_in}. Fetched {decoded_result} from Redis.")
    else:
        raise Exception(f"Bad value retrieved. Expected {uuid_in}, got
 {decoded_result}")

    return "Fetched value from Redis"
```

此代码使用 Python redis-py 库将项目放入缓存中并检索它们。此代码使用 cachetools 将生成的 IAM 身份验证令牌缓存 15 分钟。要创建包含 redis-py 和 cachetools 的部署包，请执行以下步骤。

在包含 app.py 源代码文件的项目目录中，创建一个用于将 redis-py 和 cachetools 库安装到的文件夹包。

```
mkdir package
```

使用 pip 安装 redis-py、缓存工具。

```
pip install --target ./package redis
pip install --target ./package cachetools
```

创建一个包含 redis-py 和 cachetools 库的.zip 文件。在 Linux 和 macOS 中，运行以下命令：在 Windows 中，使用你首选的 zip 实用程序创建一个.zip 文件，根目录为 redis-py 和 cachetools 库。

```
cd package
zip -r ../my_deployment_package.zip .
```

将您的函数代码添加到 .zip 文件。在 Linux 和 macOS 中，运行以下命令：在 Windows 中，使用你首选的 zip 实用程序将 app.py 添加到.zip 文件的根目录中。

```
cd ..
zip my_deployment_package.zip app.py
```

步骤 2.2：创建 IAM 角色（执行角色）

将名为的 AWS 托管策略附加AWSLambdaVPCAccessExecutionRole到该角色。

```
aws iam attach-role-policy \
 --role-name "elasticache-iam-auth-app" \
 --policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"
```

步骤 2.3：上传部署包（创建 Lambda 函数）

在此步骤中，您将使用创建函数命令创建 Lambda 函数 (AccessRedis)。 AWS CLI

在包含您的部署包.zip 文件的项目目录中，运行以下 Lambda `create-function` CLI 命令。

对于角色选项，请使用您在步骤 2.2 中创建的执行角色的 ARN。在 vpc-config 中，输入以逗号分隔的默认 VPC 子网列表和默认 VPC 的安全组 ID。您还可以在 Amazon VPC 控制台中找到这些值。要查找您的默认 VPC 子网，请选择您的 VPC，然后选择您 AWS 账户的默认 VPC。要查找此 VPC 的安全组，请转至安全并选择安全组。请确保您选择了 us-east-1 区域。

```
aws lambda create-function \
--function-name AccessRedis  \
--region us-east-1 \
--zip-file fileb://my_deployment_package.zip \
--role arn:aws:iam::123456789012:role/elasticache-iam-auth-app \
--handler app.lambda_handler \
--runtime python3.12  \
--timeout 30 \
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-
security-group-id
```

步骤 3：测试 Lambda 函数

在此步骤中，您将使用调用命令手动调用 Lambda 函数。当 Lambda 函数执行时，它会生成一个 UUID 并将其写入您在 Lambda 代码中指定的 ElastiCache 缓存中。然后，Lambda 函数将从缓存中检索项目。

1. 使用调用命令 AWS Lambda 调用 Lambda 函数 (AccessRedis)。

```
aws lambda invoke \
--function-name AccessRedis  \
--region us-east-1 \
output.txt
```

2. 按以下过程验证 Lambda 函数是否已成功执行：

   - 查看 output.txt 文件。
   - 打开 CloudWatch 控制台并选择函数的 CloudWatch 日志组 (/aws/lambda/AccessRedis)，验证日志中的结果。日志流应包含类似于以下内容的输出：

```
Success: Inserted 826e70c5f4d2478c8c18027125a3e01e. Fetched
 826e70c5f4d2478c8c18027125a3e01e from Redis.
```

   - 在 AWS Lambda 控制台中查看结果。

步骤 4：清理（可选）

要进行清理，请执行以下步骤。

主题

- 步骤 4.1：删除 Lambda 函数
- 步骤 4.2：删除无服务器缓存
- 步骤 4.3：移除 IAM 角色和策略

步骤 4.1：删除 Lambda 函数

```
aws lambda delete-function \
 --function-name AccessRedis
```

步骤 4.2：删除无服务器缓存

删除缓存。

```
aws elasticache delete-serverless-cache \
 --serverless-cache-name cache-01
```

移除用户和用户组。

```
aws elasticache delete-user \
 --user-id default-user-disabled

aws elasticache delete-user \
 --user-id iam-user-01

aws elasticache delete-user-group \
 --user-group-id iam-user-group-01
```

步骤 4.3：移除 IAM 角色和策略

```
aws iam detach-role-policy \
 --role-name "elasticache-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"

aws iam detach-role-policy \
--role-name "elasticache-iam-auth-app" \
--policy-arn "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"

aws iam delete-role \
 --role-name "elasticache-iam-auth-app"

 aws iam delete-policy \
  --policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"
```

设计和管理自己的 ElastiCache 集群

如果您需要对 ElastiCache 集群进行精细控制，则可以选择设计自己的集群。借助 ElastiCache，您可以通过为自己的集群选择节点类型、节点数和跨 AWS 可用区的节点放置，来操作基于节点的集群。由于 ElastiCache 是一项完全托管式服务，因此它可以自动管理集群的硬件预置、监控、节点更换和软件修补。

有关设置的信息，请参阅设置。有关管理、更新或删除节点或集群的详细信息，请参阅管理节点。在设计自己的 ElastiCache 集群时，如需大致了解 Amazon ElastiCache 部署的主要组件，请参阅以下重要概念。

主题

ElastiCache 适用于 Redis 组件和功能

接下来，您可以找到 Amazon ElastiCache 部署的主要组件的概述。

主题

- ElastiCache 为了 Redis 安全
- ElastiCache 子网组
- ElastiCache 用于 Redis 备份
- ElastiCache 事件

ElastiCache 节点

节点是 ElastiCache 部署的最小构建块。一个节点可独立于其他节点存在，也可与其他节点之间有某种关系。

节点是固定大小、与网络连接的安全 RAM 区块。每个节点都运行在您创建集群时选择的引擎和版本的实例。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实例类型。有关更多信息，请参阅针对 Redis ElastiCache 进行扩展 。

一个集群中的每个节点都是相同的实例类型且运行相同的缓存引擎。每个缓存节点都有自己的域名服务 (DNS) 名称和端口。支持多种缓存节点类型，每种可有不同的关联内存量。有关受支持的节点实例类型的列表，请参阅受支持的节点类型。

您可以 pay-as-you-go 按需购买节点，您只需为使用节点付费。您也可以相当优惠的小时费率购买预留节点。如果使用率高，则购买预留节点可节省资金。假设您的集群几乎始终在使用中，并且您有时会添加节点来满足使用峰值的需求。在这种情况下，您可以购买大量预留节点以在大多数情况下运行。然后，您可以根据偶尔需要添加节点的时间购买 pay-as-you-go 节点。有关预留节点的更多信息，请参阅ElastiCache 预留节点。

有关节点的更多信息，请参阅管理节点。

ElastiCache 用于 Redis 碎片

Redis 分区（在 API 和 CLI 中称为节点组）是 1 到 6 个相关节点的分组。Redis（已禁用集群模式）集群始终至少有一个分片。

分片是一种数据库分区方法，它将大型数据库分成更小、更快、更易于管理的部分，称为数据分片。这可以通过将操作分配到多个单独的部分来提高数据库效率。使用分片可以带来许多好处，包括提高性能、可扩展性和成本效益。

Redis（已启用集群模式）集群最多可以拥有 500 个分区，并且跨分区对您的数据进行分区。如果 Redis 引擎版本为 5.0.6 或更高版本，可将每个集群的节点或分片限制提高到最大值 500。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。有关更多信息，请参阅 创建子网组。对于低于 5.0.6 的版本，每个集群的限制为 250。

若要请求提高限制，请参阅 AWS Service Limits 并选择限制类型 Nodes per cluster per instance type（每个实例类型的每个集群的节点数）。

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 使用复制组时的高可用性。

有关分片的更多信息，请参阅使用分片。

ElastiCache 适用于 Redis 集群

Redis 集群 是一个或多个ElastiCache 用于 Redis 碎片的逻辑分组。数据在 Redis（启用集群模式）集群中的分区上进行分区。

许多 ElastiCache 操作都针对集群：

- 创建集群
- 修改集群
- 为集群拍摄快照（所有版本的 Redis）
- 删除集群
- 查看集群中的元素
- 在集群中添加和删除成本分配标签

有关更多详细信息，请参阅以下相关主题：

- 管理集群 和 管理节点

  有关集群、节点和相关操作的信息。
- AWS 服务限制：Amazon ElastiCache

  有关 ElastiCache 限制的信息，例如节点或集群的最大数量。要超出这些限制中的某些限制，您可以使用 Amazon ElastiCache 缓存节点申请表提出请求。
- 缓解故障

  有关增强集群和复制组的容错能力的信息。

典型集群配置

以下是典型的集群配置。

Redis 集群

Redis（已禁用集群模式）集群始终只包含一个分区（在 API 和 CLI 中，为一个节点组）。一个 Redis 分区包含 1 到 6 个节点。如果分区中有多个节点，则该分区支持复制。在这种情况下，一个节点是读/写主节点，其他为只读副本节点。

为了提高容错能力，我们建议在 Redis 集群中包含至少两个节点，并启用多可用区。有关更多信息，请参阅 缓解故障。

随着对 Redis（已禁用集群模式）集群的需求发生变化，您可以纵向扩展或缩减。为此，请将您的集群移到其他节点实例类型。如果应用程序是读取操作密集型，建议您添加只读副本 Redis（已禁用集群模式）集群。通过执行此操作，您可以将读取分布到更多数量的节点上。

您还可以使用数据分层功能。将访问频率更高的数据存储在内存中，而将访问频率较低的数据存储在磁盘上。使用数据分层的优点是可以减少内存需求。有关更多信息，请参阅 数据分层。

ElastiCache 支持动态将 Redis（已禁用集群模式）集群的节点类型更改为更大的节点类型。有关纵向扩展或缩减的信息，请参阅扩展单节点 Redis（已禁用集群模式）集群或扩展具有副本节点的 Redis（已禁用集群模式）集群。

ElastiCache 用于 Redis 复制

通过在一个分区（在 API 和 CLI 中，称为节点组）中对 2 到 6 个节点进行分组来实现复制。在这些节点中，有一个是读取/写入主节点。所有其他节点均为只读副本节点。

每个副本节点保留一个主节点中的数据的副本。复制节点使用异步复制机制来与主节点保持同步。应用程序可从集群中的任何节点进行读取，但只能对主节点进行写入。只读副本通过跨多个终端节点分布读取来增强可扩展性。只读副本还通过维护数据的多个副本来增强容错能力。在多个可用区内定位只读副本可进一步增强容错能力。有关容错能力的更多信息，请参阅缓解故障。

Redis（已禁用集群模式）集群支持一个分区（在 API 和 CLI 中，被称为节点组）。

从 API 和 CLI 的视角来看，复制使用了不同的术语来维护与之前版本的兼容性，但结果是相同的。下表显示了用于实现复制的 API 和 CLI 术语。

比较复制：Redis（已禁用集群模式）和 Redis（已启用集群模式）

下表所示为 Redis（已禁用集群模式）与 Redis（已启用集群模式）复制组的功能对比。

	Redis（已禁用集群模式）	Redis（已启用集群模式）
分区（节点组）	1	1-500
每个分片的副本数 (节点组)	0-5	0-5
数据分区	不支持	支持
添加/删除副本	支持	支持
添加/删除组节点	不支持	支持
支持扩展	支持	支持
支持引擎升级	支持	支持
将副本提升为主副本	支持	自动
多可用区	可选	必需
备份/还原	支持	支持
备注：		
如果任何主副本没有副本，则在主副本失败时，您将失去该主副本的所有数据。		
您可以使用备份和还原功能来迁移到 Redis（已启用集群模式）。		
您可以使用备份和恢复功能来调整您的 Redis（已启用集群模式）集群的大小。		

所有分区（在 API 和 CLI 中称为节点组）和节点必须位于同一 AWS 区域内。但是，您可以 AWS 在该区域内的多个可用区中配置单个节点。

只读副本可防止潜在的数据丢失，因为数据是在两个或多个节点（主节点和一个或多个只读副本）上复制的。为获得更高可靠性和更快地恢复，建议您在不同可用区内创建一个或多个只读副本。

您还可以利用全局数据存储。通过使用适用于 Redis 的全球数据存储功能，您可以跨 AWS 区域进行完全托管、快速、可靠和安全的复制。使用此功能，您可以为 Redis 创建跨区域只读副本集群，以实现跨区域的低延迟读取和灾难恢复。 ElastiCache AWS 有关更多信息，请参阅使用全球数据存储跨 AWS 区域复制。

复制：限制和局限

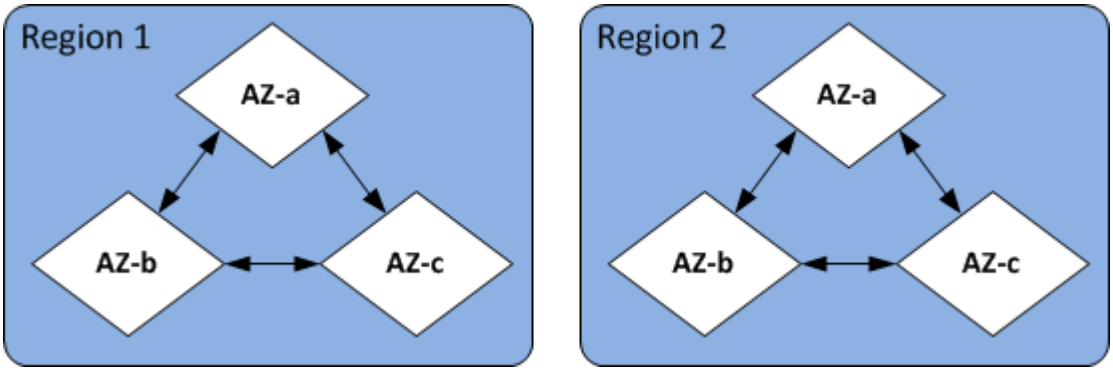
- 节点类型 T1 不支持多可用区。

AWS 区域和可用区

ElastiCache Amazon 在全球多个 AWS 地区均有销售。因此，您可以在满足业务需求的位置启动 ElastiCache集群。例如，您可以在离客户最近的 AWS 地区或满足某些法律要求的地区开店。

默认情况下， AWS 软件开发工具包 AWS CLI、 ElastiCache API 和 ElastiCache 控制台引用美国西部（俄勒冈）区域。随着新 AWS 区域的可用性 ElastiCache 扩展，这些 AWS 区域的新终端节点也可用。您可以在 HTTP 请求、 AWS 软件开发工具包和 ElastiCache 控制台中使用它们。 AWS CLI

每个 AWS 区域都设计为与其他 AWS 区域完全隔离。每个区域内有多个可用区。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 AWS 区域和可用区的更多信息，请参阅选择区域和可用区。在下图中，您可以看到 AWS 区域和可用区如何运作的高级视图。



有关支持的 AWS 区域 ElastiCache 及其终端节点的信息，请参阅支持的区域和端点。

ElastiCache 适用于 Redis 端点

终端节点是您的应用程序用于连接到 ElastiCache 节点或集群的唯一地址。

Redis（已禁用集群模式）的单节点端点

单节点 Redis 集群的终端节点用于连接到用于读取和写入的集群。

Redis（已禁用集群模式）的多节点端点

多节点 Redis（已禁用集群模式）集群包含两种类型的端点。主终端节点始终连接到集群中的主节点，即使主角色中的特定节点发生更改也是如此。使用主终端节点执行对集群的所有写入操作。

使用读取器端点 将在所有只读副本之间均匀地分配指向端点的传入连接。使用单独的节点端点 进行读取操作（在 API/CLI 中，它们被称作读取端点）。

Redis（已启用集群模式）端点

Redis（已启用集群模式）集群有一个单配置端点。通过连接到配置端点，您的应用程序可以查找集群中每个分片的主端点和读取端点。

有关更多信息，请参阅 查找连接端点。

ElastiCache 参数组

缓存参数组是为受支持的引擎软件管理运行时设置的简单方法。参数用于控制内存使用率、移出策略、项目大小等。 ElastiCache 参数组是可以应用于集群的引擎特定参数的命名集合。通过这样做，您可以确保该集群中的所有节点都以完全相同的方式进行配置。

有关受支持的参数、其默认值以及其中可以修改的参数的列表，请参阅 DescribeEngineDefaultParameters（CLI：describe-engine-default-parameters）。

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

ElastiCache 为了 Redis 安全

为了增强安全性， ElastiCache 对于 Redis 节点，仅允许在您允许的 Amazon EC2 实例上运行的应用程序进行访问。您可以使用安全组控制可访问集群的 Amazon EC2 实例。

默认情况下，所有新 ElastiCache 的 Redis 集群都是在亚马逊虚拟私有云（亚马逊 VPC）环境中启动的。可以使用子网组授予从在特定子网上运行的 Amazon EC2 实例进行集群访问的权限。

除了限制节点访问外，for Redis 还支持 TLS 和 ElastiCache 对运行 for Redis 指定版本的 ElastiCache 节点进行就地加密。有关更多信息，请参阅下列内容：

- Amazon ElastiCache 中的数据安全性
- 使用 Redis AUTH 命令进行身份验证

ElastiCache 子网组

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

如果您在 Amazon VPC 中创建集群，则必须指定缓存子网组。ElastiCache 使用该缓存子网组选择子网和该子网内的 IP 地址以与您的缓存节点相关联。

有关 Amazon VPC 环境中缓存子网组使用情况的更多信息，请参阅以下内容：

- Amazon VPC 和 ElastiCache 安全性
- 步骤 3：授予对集群的访问权限
- 子网和子网组

ElastiCache 用于 Redis 备份

备份是 Redis 集群的 point-in-time 副本。备份可用于还原现有集群或为新集群做种。备份包含集群中的所有数据和某些元数据。

根据您集群上运行的 Redis 版本，备份过程需要不同的预留内存量才能成功。有关更多信息，请参阅下列内容：

- 快照和还原
- 如何实施同步和备份
- 备份自行设计的集群所产生的性能影响
- 确保具有用于创建 Redis 快照的足够内存

ElastiCache 事件

当缓存集群上发生重要事件时， ElastiCache 会向特定的 Amazon SNS 主题发送通知。这些事件可能包括诸如添加节点失败、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，并且在许多情况下，您都可用采取相应的纠正措施。

有关 ElastiCache 事件的更多信息，请参阅Amazon SNS 监控 ElastiCache 事件。

ElastiCache for Redis 术语

2016 年 10 月，Amazon ElastiCache 增加了对 Redis 3.2 的支持。当时，我们增加了对最多 500 个分区（在 ElastiCache API 和 AWS CLI 中称为节点组）的数据进行分区的支持。为了与旧版本保持兼容，我们扩展了 API 版本 2015-02-02 的操作，以涵盖新的 Redis 功能。

同时，我们开始在 ElastiCache 控制台中采用该新功能中使用的术语以及业界常用的术语。这些更改意味着在某些时候，API 和 CLI 中使用的术语可能与控制台中使用的术语不同。以下列表指出了在 API、CLI 与控制台中可能存在差异的术语。

缓存集群或节点与节点

在没有副本节点的情况下，节点与缓存集群之间为一对一关系。因此，ElastiCache 控制台通常互换着使用这些术语。控制台现将统一使用术语节点。唯一的例外是 Create Cluster (创建集群) 按钮，该按钮用于启动一个流程来创建包含或不包含副本节点的集群。

ElastiCache API 和 AWS CLI 继续使用与以前相同的术语。

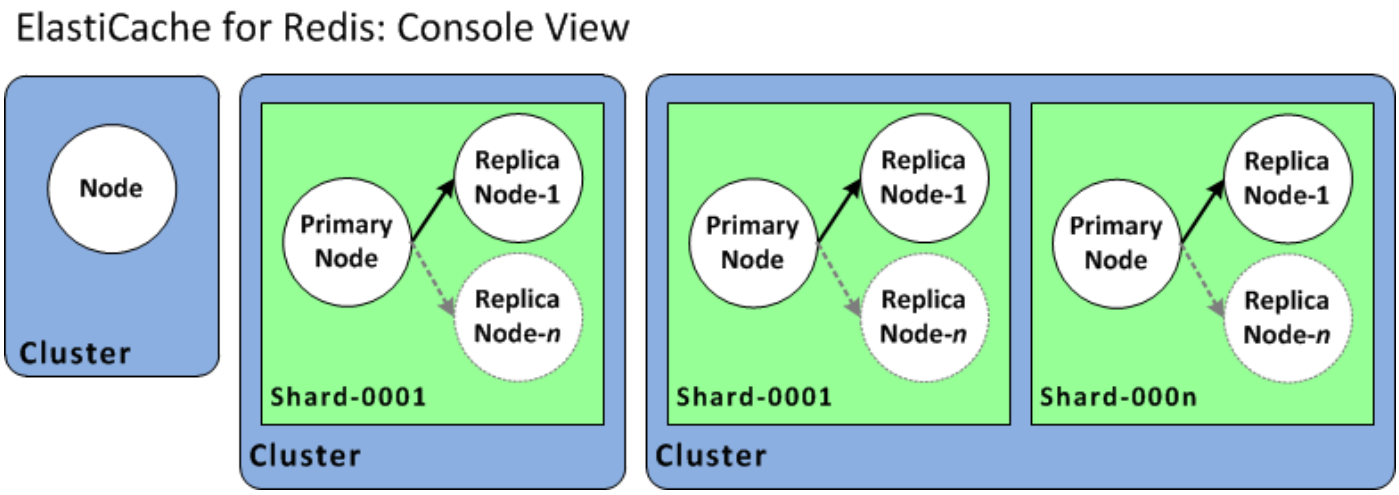
集群与复制组

对于所有 ElastiCache for Redis 集群，控制台现使用术语集群。控制台在以下所有情况中使用术语“集群”：

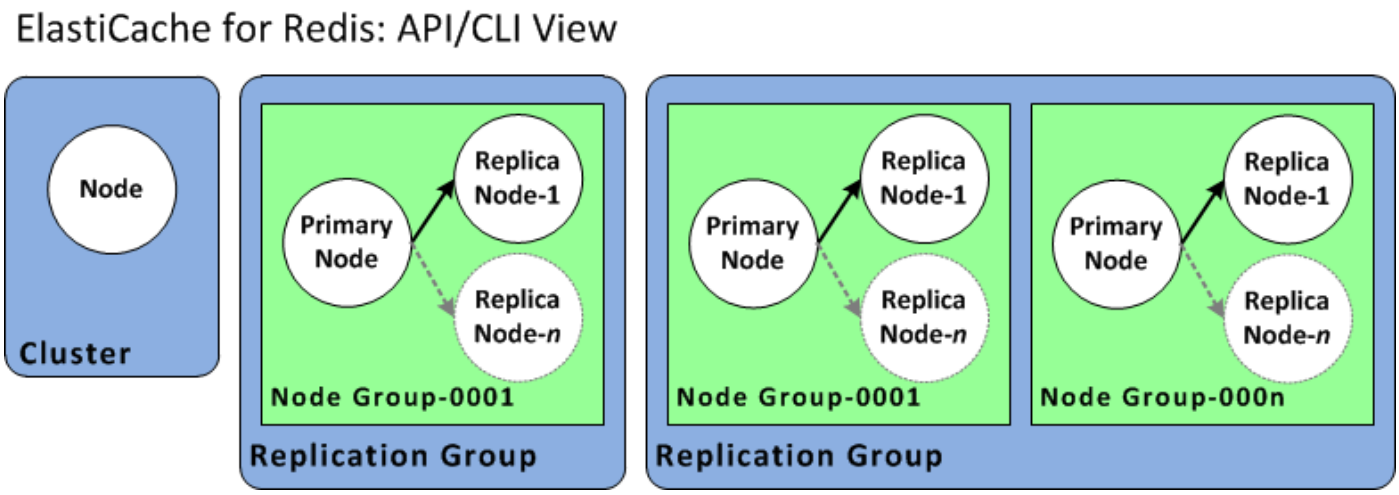
- 集群是单节点 Redis 集群。
- 当集群是支持单分区（在 API 和 CLI 中称为节点组）中复制的 Redis（已禁用集群模式）集群时。
- 当集群是支持 1-90 个分区或最多 500 个分区（须提出限制提高请求）中复制的 Redis（已启用群集模式）集群时。若要请求提高限制，请参阅 AWS Service Limits 并选择限制类型 Nodes per cluster per instance type（每个实例类型的每个集群的节点数）。

有关复制组的更多信息，请参阅使用复制组时的高可用性。

下图从控制台的角度说明了各种 ElastiCache fo Redis 集群拓扑。



ElastiCache API 和 AWS CLI 操作将继续区分单节点 ElastiCache for Redis 集群与多节点复制组。下图从 ElastiCache API 和 AWS CLI 的角度说明了各种 ElastiCache for Redis 拓扑。



复制组与全局数据存储

全局数据存储是可跨区域相互复制的一个或多个集群的集合，而复制组可复制具有多个分区的已启用集群模式集群中的数据。全局数据存储包含以下项：

- 主（主动）集群 – 主集群接受复制到全局数据存储中的所有集群的写入。主集群也接受读取请求。
- 辅助（被动）集群 – 辅助集群仅接受读取请求并从主集群复制数据更新。辅助集群需要与主集群位于不同的 AWS 区域中。

有关全局数据存储的信息，请参阅 使用全球数据存储跨 AWS 区域复制。

设计自己的集群

以下是开始设计 ElastiCache 集群时必须采取的一次性操作。

主题

设置

在您创建集群之前，请先创建子网组。缓存子网组 是您要为 VPC 中的缓存群集指定的子网集合。当您启动 VPC 中的某个缓存群集时，您需要选择一个缓存子网组。然后，ElastiCache 使用这个缓存子网组为集群中的每个缓存节点分配子网范围内的 IP 地址。

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。

有关设置 ElastiCache 的更多信息，请参阅设置。

步骤 1：创建子网组

以下过程演示如何创建名为 `mysubnetgroup` 的子网组（控制台）和 AWS CLI。

创建子网组（控制台）

以下过程介绍如何创建子网组（控制台）。

创建子网组（控制台）

1. 登录 AWS 管理控制台并打开 ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。

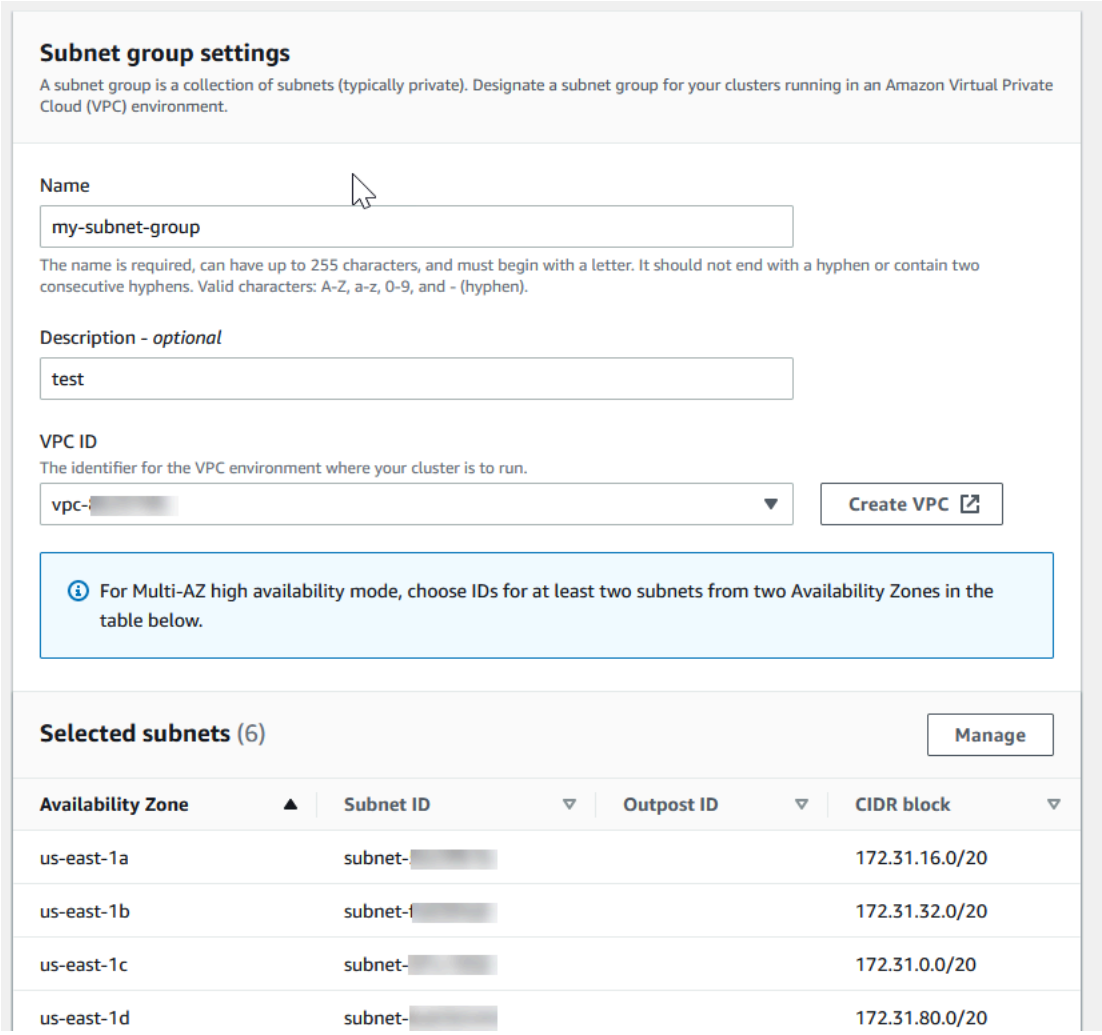
2. 在导航列表中，选择 Subnet Groups。
3. 选择 Create Subnet Group。
4. 在 Create Subnet Group（创建子网组）向导中，执行以下操作。根据需要完成所有设置后，选择 Yes, Create。

   a. 在 Name 框中，为子网组键入名称。

   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。

   d. 在 Availability Zone（可用区）和 Subnet ID（子网 ID）列表中，选择可用区或 Local Zone（本地区域）和您的私有子网 ID，然后选择 Add（添加）。



5. 在出现的确认信息中，选择 Close。

您的新子网组会显示在 ElastiCache 控制台的 Subnet Groups（子网组）列表中。您可以在窗口底部选择子网组以查看详细信息，例如与此组关联的所有子网。

创建子网组 (AWS CLI)

在命令提示符处，使用命令 `create-cache-subnet-group` 创建子网组。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup \
    --cache-subnet-group-description "Testing" \
    --subnet-ids subnet-53df9c3a
```

对于 Windows：

```
aws elasticache create-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup ^
    --cache-subnet-group-description "Testing" ^
    --subnet-ids subnet-53df9c3a
```

该命令应该生成类似于下述信息的输出：

```
{
    "CacheSubnetGroup": {
        "VpcId": "vpc-37c3cd17",
        "CacheSubnetGroupDescription": "Testing",
        "Subnets": [
            {
                "SubnetIdentifier": "subnet-53df9c3a",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                }
            }
        ],
        "CacheSubnetGroupName": "mysubnetgroup"
    }
}
```

有关更多信息，请参阅 AWS CLI 主题 create-cache-subnet-group。

步骤 2：创建集群

在创建用于生产使用的集群之前，您显然需要考虑如何配置集群以满足您的业务需求。这些问题在 准备集群 部分中解决。就本入门练习而言，您将创建一个禁用集群模式的集群，并且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。在您删除实例之前，您需要为其支付标准 ElastiCache 使用费。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 ElastiCache 使用费率的更多信息，请参阅 Amazon ElastiCache。

在虚拟私有云（VPC）中基于 Amazon VPC 服务启动集群。

创建 Redis（已禁用集群模式）集群（控制台）

使用控制台创建 Redis (已禁用集群模式) 集群 ElastiCache

1. 登录 AWS Management Console 并打开亚马逊 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 从右上角的列表中，选择要在其中启动此集群的 AWS 区域。
3. 从导航窗格中，选择 Get started（入门）。
4. 选择 Create VPC（创建 VPC）并按照创建虚拟私有云（VPC）中的步骤操作。
5. 在 ElastiCache 控制面板页面上，选择 Redis 缓存，然后选择创建 Redis 缓存。
6. 在 Cluster settings（集群设置）下，执行以下操作：

    a. 选择 Configure and create a new cluster（配置和创建新集群）。

    b. 对于 Cluster mode（集群模式），选择 Disabled（已禁用）。

    c. 对于 Cluster info（集群信息），为 Name（名称）输入一个值。

    d. （可选）为 Description（描述）输入一个值。
7. 在 Location（位置）下：

    AWS Cloud

    1. 对于 AWS Cloud，我们建议您接受 Multi-AZ（多可用区）和 Auto-failover（自动失效转移）的默认设置。有关更多信息，请参阅使用多可用区最大限度地缩短 Redis ElastiCache 的停机时间。
    2. 在 Cluster settings（集群设置）下

a. 对于 Engine version（引擎版本），选择一个可用的引擎版本。

b. 对于 Port（端口），使用默认端口 6379。如果您出于某个原因需要使用其他端口，请输入相应的端口号。

c. 对于参数组，选择一个参数组或创建一个新参数组。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 和创建参数组。

> **Note**
>
> 当您选择要设置引擎配置值的参数组时，该参数组将应用于全局数据存储中的所有集群。在 Parameter Groups（参数组）页面上，是/否 Global（全局）属性指示参数组是否属于全局数据存储。

d. 对于 Node type（节点类型），请选择向下箭头（ ）。

在 Change node type（更改节点类型）对话框中，为所需节点类型选择 Instance family（实例系列）值。接着选择要用于此集群的节点类型，然后选择保存。

有关更多信息，请参阅选择节点大小。

如果您选择 r6gd 节点类型，则系统会自动启用数据分层。有关更多信息，请参阅数据分层。

e. 对于 Number of replicas（副本数），选择所需的只读副本数。如果您启用多可用区，则该数字必须介于 1-5 之间。

3. 在 Connectivity（连接）下

a. 对于 Network type（网络类型），选择此集群将支持的 IP 版本。

b. 对于子网组，请选择要应用于此集群的子网。 ElastiCache 使用该子网组选择子网和该子网内的 IP 地址以与您的节点关联。 ElastiCache 群集需要一个同时分配 IPv4 和 IPv6 地址的双栈子网才能在双堆栈模式下运行，并且需要一个仅限 IPv6 的子网才能作为仅限 IPv6 运行。

创建新的子网组时，输入其所属的 VPC ID。

有关更多信息，请参阅：

- 选择网络类型.
- 在您的 VPC 中创建子网。

如果您是 将 Local Zones 与 ElastiCache 结合使用 ，则必须创建或选择位于本地区域中的子网。

有关更多信息，请参阅子网和子网组。

4. 对于 Availability zone placements（可用区位置），您有两种选择：

   - 无偏好 — ElastiCache 选择可用区。
   - Specify availability zones（指定可用区）– 您为各集群指定可用区。

     如果您选择指定可用区，则需从列表中为各分片中的每个集群选择可用区。

   有关更多信息，请参阅选择区域和可用区。

5. 选择 Next（下一步）
6. 在 Advanced Redis settings（高级 Redis 设置）下：

   - 对于 Security（安全）：

     i. 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**
       >
       > 您可以选择提供不同的加密密钥，方法是选择 “客户托管 AWS KMS 密钥” 并选择密钥。有关更多信息，请参阅使用 AWS KMS 客户自主管理型密钥。

     - Encryption in-transit（传输中加密）– 对传输中数据启用加密。有关更多信息，请参阅传输中加密。对于 Redis 6.0 及以上的引擎版本，如果启用了传输中加密，则系统会提示您指定以下 Access Control（访问控制）选项中的一个：

       - No Access Control（无访问控制）– 此选项为默认设置。这表示对用户访问集群的权限没有任何限制。

- User Group Access Control List（用户组访问控制列表）– 选择具有集群访问权限的已定义用户组。有关更多信息，请参阅使用控制台和 CLI 管理用户组。
- Redis AUTH Default User（Redis AUTH 默认用户）– Redis 服务器的身份验证机制。有关更多信息，请参阅 Redis AUTH。

- Redis AUTH – Redis 服务器的身份验证机制。有关更多信息，请参阅 Redis AUTH。

> **Note**
>
> 对于 Redis 3.2.6 以上的版本（版本 3.2.10 除外），只能选择 Redis AUTH。

ii. 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。

7. 如果需要定期计划自动备份，请选择启用自动备份，然后输入每个自动备份在被自动删除前保留的天数。如果您不希望定期计划自动备份，请清除 Enable automatic backups 复选框。不论是哪种情况，您始终可以选择创建手动备份。

   有关 Redis 备份和还原的更多信息，请参阅快照和还原。

8. （可选）指定维护时段。维护时段 是每周中 ElastiCache 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 ElastiCache 选择维护时段的日期和时间（No preference (无首选项)），或者自行选择日期、时间和持续时间（Specify maintenance window (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。

9. （可选）对于 Logs（日志）：

   - 在 Log format（日志格式）下，选择 Text（文本）或 JSON。
   - 在 “目标类型” 下，选择 “CloudWatch 日志” 或 “Kinesis Fire hose”。

- 在“日志目标”下，选择“新建”并输入您的 CloudWatch 日志日志组名称或 Firehose 直播名称，或者选择“选择现有”，然后选择您的 CloudWatch 日志日志组名称或 Firehose 直播名称，

10. 对于标签，为了帮助您管理集群和其他 ElastiCache 资源，您可以以标签的形式为每个资源分配自己的元数据。有关更多信息，请参阅 标记 ElastiCache 资源。
11. 选择 Next（下一步）。
12. 查看您的所有输入和选择，然后进行任意所需的更正。当您准备好后，选择 Create（创建）。

On premises

1. 对于 On premises（本地），我们建议您保留 Auto-failover（自动失效转移）为启用状态。有关更多信息，请参阅使用多可用区最大限度地缩短 Redis ElastiCache 的停机时间
2. 要完成集群创建，请按照使用 Outposts 中的步骤操作。

当您的集群状态为 available（可用）时，您可向其授予 Amazon EC2 访问权限，连接到集群并开始使用它。有关更多信息，请参阅 步骤 3：授予对集群的访问权限 和 步骤 4：连接到集群节点。

> **Important**
>
> 您的集群变为可用状态后，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 删除集群。

创建 Redis（已禁用集群模式）集群（AWS CLI）

Example

下面的 CLI 代码创建一个无副本的 Redis（已禁用集群模式）缓存群集。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-cache-cluster \
--cache-cluster-id my-cluster \
--cache-node-type cache.r4.large \
--engine redis \
--num-cache-nodes 1 \
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

对于 Windows：

```
aws elasticache create-cache-cluster ^
--cache-cluster-id my-cluster ^
--cache-node-type cache.r4.large ^
--engine redis ^
--num-cache-nodes 1 ^
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

要在启用集群模式的情况下工作，请参阅以下主题：

- 要使用控制台，请参阅创建 Redis（已启用集群模式）集群（控制台）。
- 要使用 AWS CLI，请参阅创建 Redis（已启用集群模式）集群（AWS CLI）。

步骤 3：授予对集群的访问权限

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

所有 ElastiCache 集群旨在通过 Amazon EC2 实例进行访问。最常见的情况是从同一 Amazon Virtual Private Cloud (Amazon VPC) 中的 Amazon EC2 实例访问 ElastiCache 集群，这将是本练习的情况。

原定设置情况下，对您的集群的网络访问仅限于用于创建集群的账户。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。所需步骤取决于是否将集群启动到 EC2-VPC 或 EC2-Classic 中。

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实例的访问权限。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。

   Redis 集群和复制组的默认端口为 6379。

   > **Important**
   >
   > Amazon ElastiCache 安全组仅适用于未运行于 Amazon Virtual Private Cloud 环境 (VPC) 中的集群。如果您正在 Amazon Virtual Private Cloud 中运行，安全组将在控制台导航窗格中将不可用。
   > 如果您在 Amazon VPC 中运行 ElastiCache 节点，则可使用 Amazon VPC 安全组（与 ElastiCache 安全组不同）控制对集群的访问。有关在 Amazon VPC 中使用 ElastiCache 的更多信息，请参阅 Amazon VPC 和 ElastiCache 安全性

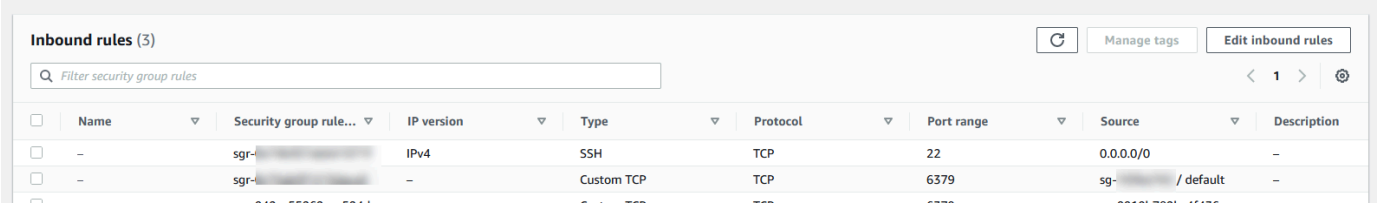
2. 为 EC2 实例 (Web 和应用程序服务器) 创建 VPC 安全组。如果需要，此安全组可允许通过 VPC 的路由表从 Internet 访问 EC2 实例。例如，您可设置此安全组的规则以允许通过端口 22 对 EC2 实例进行 TCP 访问。
3. 在集群的安全组中创建自定义规则，允许从为 EC2 实例创建的安全组连接。这将允许安全组的任何成员均可访问集群。

> **Note**
>
> 如果您计划使用 Local Zones，请确保已将其启用。当您在该本地区域中创建子网组时，您的 VPC 也会扩展到该本地区域，并且您的 VPC 会将该子网视为任何其他可用区中的任何子网。所有相关网关和路由表都将自动调整。

在 VPC 安全组中创建允许从另一安全组连接的规则

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc。
2. 在导航窗格中，选择 Security Groups（安全组）。
3. 选择或创建一个要用于集群实例的安全组。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。

      Redis 集群和复制组的默认端口为 6379。
   b. 在 Source 框中，开始键入安全组的 ID。从列表中选择要用于 Amazon EC2 实例的安全组。
5. 完成后选择 Save。



启用访问后，您现在就可以连接到节点，如下一部分中所述。

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

- 访问 Amazon VPC 中 ElastiCache 缓存的访问模式
- 从 AWS 外部访问 ElastiCache 资源

步骤 4：连接到集群节点

在继续之前，请完成步骤 3：授予对集群的访问权限。

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

仅当您进行授权后，Amazon EC2 实例才能连接到集群节点。

查找您的节点端点

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

查找 Redis（已禁用集群模式）集群的端点（控制台）

如果 Redis（已禁用集群模式）集群只有一个节点，则使用该节点的端点进行读取和写入操作。如果该集群具有多个节点，则有三种类型的端点，即主端点、读取器端点和节点端点。

主端点是一个 DNS 名称，始终解析为集群中的主节点。主端点不受集群更改的影响，如将只读副本提升为主角色。对于写入活动，我们建议您的应用程序连接到主端点。

读取器端点将在 ElastiCache for Redis 集群中的所有只读副本之间均匀地分配指向端点的传入连接。应用程序何时创建连接或应用程序如何（重复）使用连接等附加因素将决定流量分配。读取器端点会在添加或删除副本时实时跟踪集群更改。您可以将 ElastiCache for Redis 集群的多个只读副本置于不同的 AWS 可用区（AZ）中以确保读取器端点的高可用性。

> **Note**
>
> 读取器端点不是负载均衡器。它是一个 DNS 记录，将以循环方式解析为副本节点之一的 IP 地址。

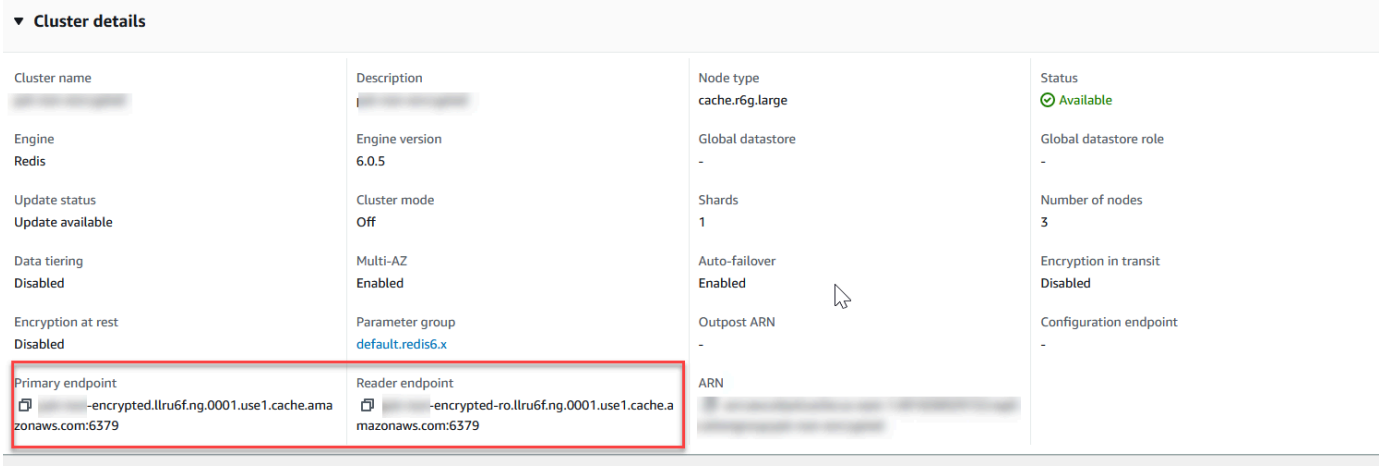
对于读取活动，应用程序还可以连接到集群中的任何节点。与主端点不同，节点端点会解析为特定端点。如果您在您的集群中进行更改（例如添加或删除副本），则必须在您的应用程序中更新节点端点。

查找 Redis（已禁用集群模式）集群的端点

1. 登录 AWS Management Console 并打开 ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 从导航窗格中，选择 Redis 缓存。

集群屏幕将显示一个列表，其中包含任何现有 Redis 无服务器缓存、Redis（已禁用集群模式）和 Redis（已启用集群模式）集群。选择在 创建 Redis（已禁用集群模式）集群（控制台） 部分中创建的集群。

3. 要查找集群的主端点和/或读取器端点，请选中集群的名称（不是单选按钮）。

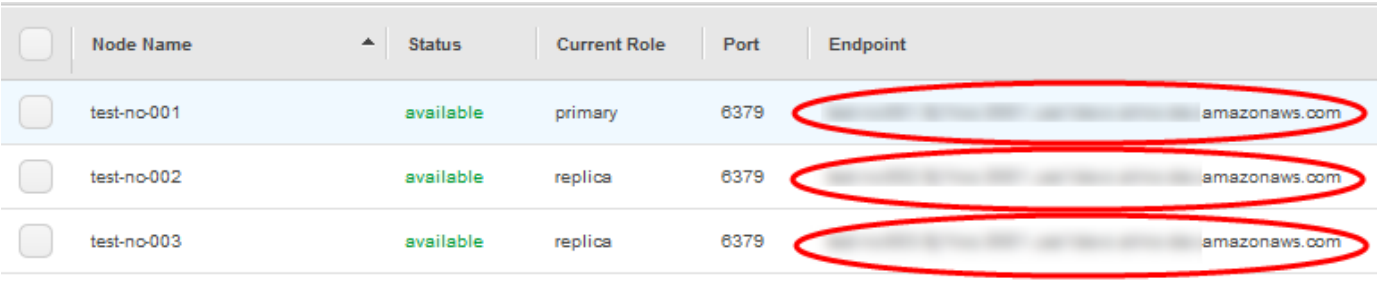


Redis（已禁用集群模式）集群的主端点和读取器端点

如果该集群只有一个节点，则没有主端点，您可以继续下一步。

4. 如果 Redis（已禁用集群模式）集群有副本节点，您可以通过选择此集群的名称、然后选择 Nodes（节点）选项卡来找到集群副本的节点端点。

此时会显示节点屏幕，其中列出了集群中的每个节点（主节点和副本节点）及其端点。



Redis（已禁用集群模式）集群的节点端点

5. 将端点复制到剪贴板：

   a. 逐一找到要复制的端点。

   b. 直接选择端点前面的复制图标。

端点现已复制到剪贴板。有关使用端点连接到节点的信息，请参阅 连接到节点。

Redis（已禁用集群模式）主端点类似以下内容。根据是否已启用传输中加密而有所不同。

未启用传输中加密

```
clusterName.xxxxxx.nodeId.regionAndAz.cache.amazonaws.com:port

redis-01.7abc2d.0001.usw2.cache.amazonaws.com:6379
```

已启用传输中加密

```
master.clusterName.xxxxxx.regionAndAz.cache.amazonaws.com:port

master.ncit.ameaqx.use1.cache.amazonaws.com:6379
```

为进一步了解如何查找您的端点，请参阅您正在运行的引擎和集群类型的相关主题。

- 查找连接端点
- 查找 Redis（已启用集群模式）集群的端点（控制台） – 您需要集群的配置端点。
- 查找端点（AWS CLI）
- 查找端点（ElastiCache API）

连接到 Redis 集群或复制组（Linux）

现在您有了所需的端点，便可以登录 EC2 实例并连接到集群或复制组。在以下示例中，您使用 redis-cli 实用工具连接到集群。最新版本的 redis-cli 还支持 SSL/TLS 用于连接启用加密/身份验证的集群。

以下示例使用运行 Amazon Linux 和 Amazon Linux 2 的 Amazon EC2 实例。有关使用其他 Linux 发行版安装和编译 redis-cli 的详细信息，请参阅特定操作系统的文档。

> **Note**
>
> 此过程包括使用仅供计划外使用的 redis-cli 实用工具测试连接。有关受支持 Redis 客户端的列表，请参阅 Redis 文档。有关通过 ElastiCache 使用 AWS 开发工具包的示例，请参阅 ElastiCache 和 AWS 开发工具包入门。

连接到已禁用集群模式的未加密的集群

1. 运行以下命令以连接到集群，并将 *primary-endpoint* 和 *port number* 替换为您的集群端点和您的端口号。（Redis 的默认端口为 6379。）

```
src/redis-cli -h primary-endpoint -p port number
```

Redis 命令提示符的结果类似于以下内容：

```
primary-endpoint:port number
```

2. 现在您就可以运行 Redis 命令了。

```
set x Hello
OK

get x
"Hello"
```

连接到已启用集群模式的未加密集群

1. 运行以下命令以连接到集群，并将 *configuration-endpoint* 和 *port number* 替换为您的集群的端点和您的端口号。（Redis 的默认端口为 6379。）

```
src/redis-cli -h configuration-endpoint -c -p port number
```

> **Note**
>
> 在上述命令中，选项 -c 可遵循 -ASK 和 -MOVED 重新导向启用集群模式。

Redis 命令提示符的结果类似于以下内容：

```
configuration-endpoint:port number
```

2. 现在您就可以运行 Redis 命令了。请注意，重新导向发生是因为您使用 -c 选项启用了它。如果未启用重新导向，则命令将返回 MOVED 错误。有关 MOVED 错误的更多信息，请参阅 Redis 集群规范。

```
set x Hi
-> Redirected to slot [16287] located at 172.31.28.122:6379
OK
set y Hello
OK
get y
"Hello"
set z Bye
-> Redirected to slot [8157] located at 172.31.9.201:6379
OK
get z
"Bye"
get x
-> Redirected to slot [16287] located at 172.31.28.122:6379
"Hi"
```

连接到启用加密/身份验证的集群

默认情况下，redis-cli 在连接到 Redis 时使用未加密的 TCP 连接。选项 BUILD_TLS=yes 在 redis-cli 编译时启用 SSL/TLS，如上述 下载并设置 redis-cli 部分所示。启用 AUTH 是可选的。但是，您必须启用传输过程中的加密才能启用 AUTH。有关 ElastiCache 加密和身份验证的更多详细信息，请参阅 ElastiCache 传输中加密 (TLS)。

> **Note**
>
> 您可以通过 redis-cli 使用选项 `--tls` 连接到已启用和已禁用集群模式的加密集群。如果集群设置了 AUTH 令牌，则可以使用选项 `-a` 以提供 AUTH 密码。

在以下示例中，确保将 *cluster-endpoint######* 和 *port number#####* 替换为您的集群端点和您的端口号。（Redis 的默认端口为 6379。）

连接到已禁用集群模式的加密群集

以下示例连接到已启用加密和身份验证的集群：

```
src/redis-cli -h cluster-endpoint --tls -a your-password -p port number
```

以下示例连接到仅启用加密的集群：

```
src/redis-cli -h cluster-endpoint --tls -p port number
```

连接到已启用集群模式的加密群集

以下示例连接到已启用加密和身份验证的集群：

```
src/redis-cli -c -h cluster-endpoint --tls -a your-password -p port number
```

以下示例连接到仅启用加密的集群：

```
src/redis-cli -c -h cluster-endpoint --tls -p port number
```

连接到集群后，您可以为未加密集群运行上述示例中的 Redis 命令。

Redis-cli 替代方案

如果集群未启用集群模式，并且您需要与集群建立连接以进行短期测试（但不经过 redis-cli 编译），则可以使用 telnet 或 openssl。在以下示例命令中，确保将 *cluster-endpoint######* 和 *port number#####* 替换为您的集群端点和您的端口号。（Redis 的默认端口为 6379。）

以下示例连接到已启用加密和/或身份验证且已禁用集群模式的集群：

```
openssl s_client -connect cluster-endpoint:port number
```

如果集群已设置密码，请先连接到集群。连接后，使用以下命令对集群进行身份验证，然后按 Enter 键。在以下示例中，将 *your-password* 替换为您的集群密码。

```
Auth your-password
```

以下示例连接到未启用加密或身份验证的已禁用集群模式的集群：

```
telnet cluster-endpoint port number
```

连接到 Redis 集群或复制组（Windows）

要使用 Redis CLI 从 EC2 Windows 实例连接到 Redis 集群，您必须下载 redis-cli 软件包并使用 redis-cli.exe 从 EC2 Windows 实例连接到 Redis 集群。

在以下示例中，您使用 redis-cli 实用工具连接到未启用加密的运行 Redis 的集群。有关 Redis 以及可用 Redis 命令的更多信息，请参阅 Redis 网站上的 Redis 命令。

使用 redis-cli 连接到未启用加密的 Redis 集群

1. 使用您选择的连接实用工具连接到 Amazon EC2 实例。有关如何连接到 Amazon EC2 实例的说明，请参阅 Amazon EC2 入门指南。
2. 复制链接 https://github.com/microsoftarchive/redis/releases/download/win-3.0.504/Redis-x64-3.0.504.zip 并将其粘贴到互联网浏览器中，以便从 GitHub https://github.com/microsoftarchive/redis/releases/tag/win-3.0.504 提供的可用版本中下载 Redis 客户端的 zip 格式文件

   将 zip 文件提取到您指定的文件夹/路径。

   打开命令提示符并更改为 Redis 目录，然后运行命令 `c:\Redis>redis-cli -h Redis_Cluster_Endpoint -p 6379`。

   例如：

   ```
   c:\Redis>redis-cli -h cmd.xxxxxxx.ng.0001.usw2.cache.amazonaws.com -p 6379
   ```

3. 运行 Redis 命令。

   您现已连接至集群并且可以按以下方式运行 Redis 命令。

   ```
   set a "hello"           // Set key "a" with a string value and no expiration
   OK
   get a                   // Get value for key "a"
   "hello"
   get b                   // Get value for key "b" results in miss
   (nil)
   set b "Good-bye" EX 5   // Set key "b" with a string value and a 5 second expiration
   "Good-bye"
   get b                   // Get value for key "b"
   "Good-bye"
                           // wait >= 5 seconds
   get b
   (nil)                   // key has expired, nothing returned
   quit                    // Exit from redis-cli
   ```

步骤 5：删除集群

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

> **Warning**
>
> 当您删除 ElastiCache for Redis 集群时，您的手动快照会保留。您也可以在删除集群之前创建最终快照。自动缓存快照不会保留。有关更多信息，请参阅快照和还原。

使用 AWS 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 AWS Management Console 并打开 Amazon ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在 ElastiCache 控制台控制面板中，选择 Redis。

   此时会显示一个列表，其中包含所有运行 Redis 的缓存。
3. 要选择要删除的集群，请从集群列表中选择该集群的名称。在这种情况下，是您在 步骤 2：创建集群 创建的集群的名称。

   > **Important**
   >
   > 从 ElastiCache 控制台一次只能删除一个集群。选择多个集群会禁用删除操作。
4. 对于操作，选择删除。
5. 在删除集群确认屏幕中，键入集群名称并选择最终备份。然后选择删除以删除集群，或选择取消以保留集群。

   如果选择了 Delete，集群的状态将变为正在删除。

只要您的集群不再在集群列表中列出，您就无需为该集群付费。

使用 AWS CLI

下面的代码删除缓存群集 `my-cluster`。在这种情况下，将 `my-cluster` 替换为您在 步骤 2：创建集群 中创建的集群的名称。

```
aws elasticache delete-cache-cluster --cache-cluster-id my-cluster
```

`delete-cache-cluster` CLI 操作仅删除一个缓存群集。要删除多个缓存群集，请对要删除的每个缓存群集调用 `delete-cache-cluster`。在删除一个缓存群集之前，您无需等待删除另一个集群的完成。

对于 Linux、macOS 或 Unix：

```
aws elasticache delete-cache-cluster \
    --cache-cluster-id my-cluster \
    --region us-east-2
```

对于 Windows：

```
aws elasticache delete-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --region us-east-2
```

有关更多信息，请参阅 AWS CLI for ElastiCache 主题 `delete-cache-cluster`。

ElastiCache 教程和视频

以下教程着重介绍了 Amazon ElastiCache 用户关注的任务。

- ElastiCache 视频
- 教程：配置 Lambda 函数，以便在 Amazon VPC 中访问 Amazon ElastiCache

ElastiCache 视频

在接下来的部分中，您可以找到视频来帮助您学习 Amazon ElastiCache 的基本概念和高级概念。有关 AWS 培训的信息，请参阅 AWS 培训和认证。

主题

- 宣传视频
- 高级视频

宣传视频

以下视频将向您介绍 Amazon ElastiCache。

主题

- AWS re:Invent 2020：Amazon ElastiCache 中的新增功能
- AWS re:Invent 2019：Amazon ElastiCache 中的新增功能
- AWS re:Invent 2017：Amazon ElastiCache 中的新增功能
- DAT204 – 在 AWS NoSQL 服务上构建可扩展应用程序 (re:Invent 2015)
- DAT207 - 利用 Amazon ElastiCache 提升应用程序性能 (AWS re:Invent 2013)

AWS re:Invent 2020：Amazon ElastiCache 中的新增功能

AWS re:Invent 2020：Amazon ElastiCache 中的新增功能

AWS re:Invent 2019：Amazon ElastiCache 中的新增功能

AWS re:Invent 2019：Amazon ElastiCache 中的新增功能

AWS re:Invent 2017：Amazon ElastiCache 中的新增功能

AWS re:Invent 2017：Amazon ElastiCache 中的新增功能

DAT204 – 在 AWS NoSQL 服务上构建可扩展应用程序 (re:Invent 2015)

在本演讲中，我们将介绍 NoSQL 数据库的优势并浏览 AWS 提供的主要 NoSQL 服务，即 Amazon DynamoDB 和 Amazon ElastiCache。然后，我们听听两家占据领先地位的客户 Expedia 和 Mapbox 讲述他们的使用案例和面临的架构挑战，以及他们如何使用 AWS NoSQL 服务（包括设计模式和最佳实践）来解决这些难题。完成本培训之后，您将更好地了解 NoSQL 及其强大的功能，满怀信心地准备好解决所面临的数据库挑战。

DAT204 – 在 AWS NoSQL 服务上构建可扩展应用程序 (re:Invent 2015)

DAT207 - 利用 Amazon ElastiCache 提升应用程序性能 (AWS re:Invent 2013)

本视频介绍了如何使用 Amazon ElastiCache 轻松部署内存中的缓存系统来提升应用程序性能。我们将向您介绍如何使用 Amazon ElastiCache 来改善应用程序延迟并减少数据库服务器上的负载。我们还将向您说明如何构建可轻松管理并随应用程序不断增多而扩展的缓存层。在此演讲中，我们将复习可通过启用缓存获益的各种方案和使用案例，并讨论 Amazon ElastiCache 提供的各种功能。

DAT207 - 利用 Amazon ElastiCache 提升应用程序性能 (re:Invent 2013)

高级视频

以下视频包括更高级的 Amazon ElastiCache 主题。

主题

- 利用 Amazon ElastiCache 最佳实践进行设计以取得成功 (re:Invent 2020)
- 使用 Amazon ElastiCache 增强您的实时应用程序 (re:Invent 2019)
- 最佳实践：将 Redis 集群从 Amazon EC2 迁移到 ElastiCache (re:Invent 2019:)
- 使用 Amazon ElastiCache 和 Amazon Aurora STP11 扩展 Fantasy Sports 平台 (re:Invent 2018)
- 利用 Amazon ElastiCache 在云中实现可靠且可扩展的 Redis (re:Invent 2018)
- ElastiCache 深入探究：内存数据存储的设计模式 (re:Invent 2018)
- DAT305 - 深入探究 Amazon ElastiCache (re:Invent 2017)
- DAT306 - 深入探究 Amazon ElastiCache (re:Invent 2016)
- DAT317 - IFTTT 如何使用 ElastiCache for Redis 来预测事件 (re:Invent 2016)
- DAT407 - 深入探究 Amazon ElastiCache (re:Invent 2015)
- SDD402 - 深入探究 Amazon ElastiCache (re:Invent 2014)
- DAT307 - 深入探究 Amazon ElastiCache 架构和设计模式 (re:Invent 2013)

利用 Amazon ElastiCache 最佳实践进行设计以取得成功 (re:Invent 2020)

随着基于 Redis 构建的业务关键型实时应用程序的爆炸式增长，可用性、可扩展性和安全性已成为首要考虑因素。了解通过线上扩展、跨多可用区部署的高可用性和安全配置设置 Amazon ElastiCache 的最佳实践，以取得成功。

利用 Amazon ElastiCache 最佳实践进行设计以取得成功 (re:Invent 2020)

使用 Amazon ElastiCache 增强您的实时应用程序 (re:Invent 2019)

随着云采用的快速发展及其支持的新场景，应用程序需要微秒级的延迟和高吞吐量来支持每秒数百万次的请求。开发人员传统上依靠专门的硬件和解决方法（例如将基于磁盘的数据库与数据缩减技术相结合）来管理实时应用程序的数据。这些方法可能很昂贵，而且无法扩展。了解如何通过使用完全托管式的内存中 Amazon ElastiCache 来提高实时应用程序的性能以获得极限性能、高可扩展性、可用性和安全性。

使用 Amazon ElastiCache 增强您的实时应用程序 (re:Invent 2019:)

最佳实践：将 Redis 集群从 Amazon EC2 迁移到 ElastiCache (re:Invent 2019:)

自己管理 Redis 集群可能很困难。您必须持续预置硬件、修补软件、备份数据和监控工作负载。通过新发布的 Amazon ElastiCache 联机迁移功能，您现在可以在禁用集群模式的情况下轻松地将数据从 Amazon EC2 上的自托管式 Redis 移动到完全托管式的 Amazon ElastiCache。在本次演讲中，您将了解新的联机迁移工具，观看演示，更重要的是学习亲自动手的最佳实践，以便顺利迁移到 Amazon ElastiCache。

最佳实践：将 Redis 集群从 Amazon EC2 迁移到 ElastiCache (re:Invent 2019)

使用 Amazon ElastiCache 和 Amazon Aurora STP11 扩展 Fantasy Sports 平台 (re:Invent 2018)

Dream11 是一家印度的领先体育科技初创公司。该公司拥有超过 4000 万用户参加了多种运动，包括梦幻板球、足球和篮球。目前该公司为 100 万并发用户提供服务，这些用户每分钟可产生 300 万个请求，响应时间仅为 50 毫秒。在本次演讲中，Dream11 CTO Amit Sharma 将介绍该公司如何使用 Amazon Aurora 和 Amazon ElastiCache 来处理闪存流量，这些流量在 30 秒的响应窗口内可能增加三倍。Sharma 还将探讨在不锁定的情况下扩展事务，并分享处理闪存流量的步骤，从而为 500 万日活跃用户提供服务。完整标题：AWS re:Invent 2018：使用 Amazon ElastiCache 和 Amazon Aurora STP11 扩展 Fantasy Sports 平台

使用 Amazon ElastiCache 和 Amazon Aurora STP11 扩展 Fantasy Sports 平台 (re:Invent 2018)

利用 Amazon ElastiCache 在云中实现可靠且可扩展的 Redis (re:Invent 2018)

在本次演讲中，将介绍我们与 Redis 兼容的服务 (Amazon ElastiCache for Redis) 中的功能和增强功能。演讲内容涵盖其主要功能，如 Redis 5、可扩展性和性能提升、安全性和合规性等。我们还将探讨即将推出的功能和客户案例研究。

利用 Amazon ElastiCache 在云中实现可靠且可扩展的 Redis (re:Invent 2018)

ElastiCache 深入探究：内存数据存储的设计模式 (re:Invent 2018)

在本次演讲中，我们将深入了解 Amazon ElastiCache 的设计和架构内幕。了解我们的 Redis 和 Memcached 产品的常见设计模式，以及客户如何利用这些产品执行内存中数据处理以减小延迟和增加应用程序吞吐量。我们还将回顾 ElastiCache 最佳实践、设计模式以及不合理的模式。

ElastiCache 深入探究：内存数据存储的设计模式 (re:Invent 2018)

DAT305 - 深入探究 Amazon ElastiCache (re:Invent 2017)

深入探究 Amazon ElastiCache 的设计和架构内幕。了解我们的 Memcached 和 Redis 产品的常见设计模式，以及客户如何利用这些产品执行内存中操作以减小延迟和增加应用程序吞吐量。在此视频中，我们将回顾 ElastiCache 最佳实践、设计模式以及不合理的模式。

此视频推出了以下内容：

- ElastiCache for Redis 线上重新分片
- ElastiCache 安全性和加密
- ElastiCache for Redis 版本 3.2.10

DAT305 - 深入探究 Amazon ElastiCache (re:Invent 2017)

DAT306 - 深入探究 Amazon ElastiCache (re:Invent 2016)

深入探究 Amazon ElastiCache 的设计和架构内幕。了解我们的 Memcached 和 Redis 产品的常见设计模式，以及客户如何利用这些产品执行内存中操作以减小延迟和增加应用程序吞吐量。在此演讲中，我们将回顾与 ElastiCache 相关的最佳实践、设计模式以及不合理的模式。

DAT306 - 深入探究 Amazon ElastiCache (re:Invent 2016)

DAT317 - IFTTT 如何使用 ElastiCache for Redis 来预测事件 (re:Invent 2016)

IFTTT 是一项免费服务，能够使人们借助所喜爱的服务完成更多的任务，从实现简单任务自动化到转换人们相互交往以及控制家居设备的方式。IFTTT 使用 ElastiCache for Redis 来存储事务处理运行历史记录和时间表预测以及 Amazon S3 上日志文档的索引。观看本次会议，了解人们如何利用 LUA 的脚本功能和 Redis 的数据类型来完成一些在其他地方不可能完成的事情。

DAT317 - IFTTT 如何使用 ElastiCache for Redis 来预测事件 (re:Invent 2016)

DAT407 - 深入探究 Amazon ElastiCache (re:Invent 2015)

深入了解 Amazon ElastiCache 的设计和架构内幕。您将了解我们的 Memcached 和 Redis 产品的常见设计模式，以及客户如何利用这些产品执行内存中操作并缩短延迟时间，同时提高应用程序的吞吐量。在此演讲中，我们将回顾与 Amazon ElastiCache 相关的最佳实践、设计模式以及不合理的模式。

DAT407 - 深入探究 Amazon ElastiCache (re:Invent 2015)

SDD402 - 深入探究 Amazon ElastiCache (re:Invent 2014)

在此视频中，我们将检查常见缓存使用案例、Memcached 和 Redis 引擎、帮助您确定哪种引擎更能满足您需求的模式、一致性哈希以及更多快速构建方法和可扩展的应用程序。Adobe 的首席科学家 Frank Wiebe 将详细介绍 Adobe 如何使用 Amazon ElastiCache 改善客户体验和扩展业务。

DAT402 - 深入探究 Amazon ElastiCache (re:Invent 2014)

DAT307 - 深入探究 Amazon ElastiCache 架构和设计模式 (re:Invent 2013)

在本视频中，我们将检查缓存、缓存策略、向外扩展和监控。我们也将比较 Memcached 和 Redis 引擎。在此演讲中，我们还将回顾与 Amazon ElastiCache 相关的最佳实践和设计模式。

DAT307 - 深入探究 Amazon ElastiCache 架构和设计模式 (AWS re:Invent 2013)。

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 ElastiCache 和可用工具的更多信息：

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

完成入门练习之后，您可以阅读以下部分了解有关 ElastiCache 管理的更多信息：

- 选择节点大小

  您希望缓存足够大以容纳要进行缓存的所有数据。同时，您不希望为所需缓存之外的缓存付费。使用本主题可帮助您选择最佳节点大小。
- ElastiCache 最佳实践和缓存策略

  确定并解决会影响集群效率的问题。

管理节点

节点是 Amazon ElastiCache 部署中的最小构建数据块。它是固定大小、与网络连接的安全 RAM 块。每个节点都运行创建或最后一次修改集群或复制组时选择的引擎。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 ElastiCache 节点，每种类型的节点具有不同的关联内存量和计算能力。

一般而言，由于支持分区，Redis（已启用集群模式）部署具有许多更小的节点。相比之下，Redis（已禁用集群模式）部署在集群中的节点数量更少、规模更大。有关要使用的节点大小的更详细讨论，请参阅选择节点大小。

主题

涉及节点的一些重要操作如下：

查看 ElastiCache 节点状态

使用ElastiCache 控制台，您可以快速访问 ElastiCache 节点的状态。 ElastiCache 节点的状态表示该节点的运行状况。您可以使用以下过程在 Amazon ElastiCache 控制台、 AWS CLI 命令或 API 操作中查看 ElastiCache 节点状态。

下表列出了 ElastiCache 节点可能的状态值。此表还显示您是否需要为该 ElastiCache 节点付费。

类型	已计费	描述
available	已计费	该 ElastiCache 节点运行状况良好，可用。
creating	不计费	正在创建 ElastiCache 节点。节点正在创建，无法访问。
deleting	不计费	正在删除该 ElastiCache 节点。
modifying	已计费	由于客户要求修改该 ElastiCache 节点，因此正在修改该节点。
updating	已计费	更新状态表明 Amazon ElastiCache 节点存在以下一项或多项情况： • 作为服务更新的一部分，正在对该 ElastiCache 节点进行修补。有关服务更新的更多信息，请参阅 Amazon ElastiCache 托管维护和服务更新帮助页面。 • 正在更新 ElastiCache 集群的 VPC 安全组。 • 集 ElastiCache 群正在扩容或缩小规模。 • 正在修改 ElastiCache 集群的日志传输配置。 • 该 ElastiCache 节点的删除操作正在等待中。 • 正在 ElastiCache 使用更新/轮换 For Redis 的密码。AWS Secrets Manager

类型	已计费	描述
`rebooting cache cluster nodes`	已计费	由于客户请求或 Amazon ElastiCache 流程要求重启节点，该 ElastiCache 节点正在重启。
`incompatible_parameters`	不计费	Amazon ElastiCache 无法启动该节点，因为节点的参数组中指定的参数与该节点不兼容。请恢复参数更改或使这些更改与数据库节点相兼容，以便重新访问节点。有关不兼容参数的更多信息，请查看该 ElastiCache 节点的事件列表。
`incompatible_network`	不计费	网络不兼容状态表明 Amazon 节点存在以下一项或多项情况： ElastiCache • 启动 ElastiCache 节点的子网中没有可用的 IP 地址。 • 子网组中提到的 ElastiCache 子网已不存在于亚马逊虚拟私有云 (Amazon VPC) 中。

类型	已计费	描述
restore_failed	不计费	恢复失败状态表明 Amazon ElastiCache 节点存在以下情况之一： • 由于持续存在实例容量不足情况，节点替换失败。运行上一代节点时，通常会发生这种情况 end-of-life。但是，当当前一代节点 AWS 没有足够的按需容量来满足您在指定可用区域中的请求时，也可能发生这种情况。有关修复或移除这些节点的更多信息，请参阅迁移上一代节点。 • 无法恢复指定的 RDB 快照。 • 该 ElastiCache 集群的 AWS 账户已被暂停。 • 节点出现故障，无法恢复。
snapshotting	已计费	ElastiCache 正在创建 for Redis 节点 ElastiCache 的快照。

使用控制台查看 ElastiCache 节点状态

要使用控制台查看 ElastiCache 节点的状态，请执行以下操作：

1. 登录 AWS Management Console 并打开亚马逊 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格中，选择 Redis 集群或内存缓存集群。将出现 “缓存” 页面，其中包含 ElastiCache 节点列表。还显示每个节点的状态值。
3. 然后，您可以导航到缓存的 “服务更新” 选项卡，以显示适用于缓存的服务更新列表。

使用查看 ElastiCache 节点状态 AWS CLI

要使用查看 ElastiCache 节点及其状态信息 AWS CLI，请使用`describe-cache-cluster`命令。例如，以下 AWS CLI 命令显示每个 ElastiCache 节点。

```
aws elasticache describe-cache-clusters
```

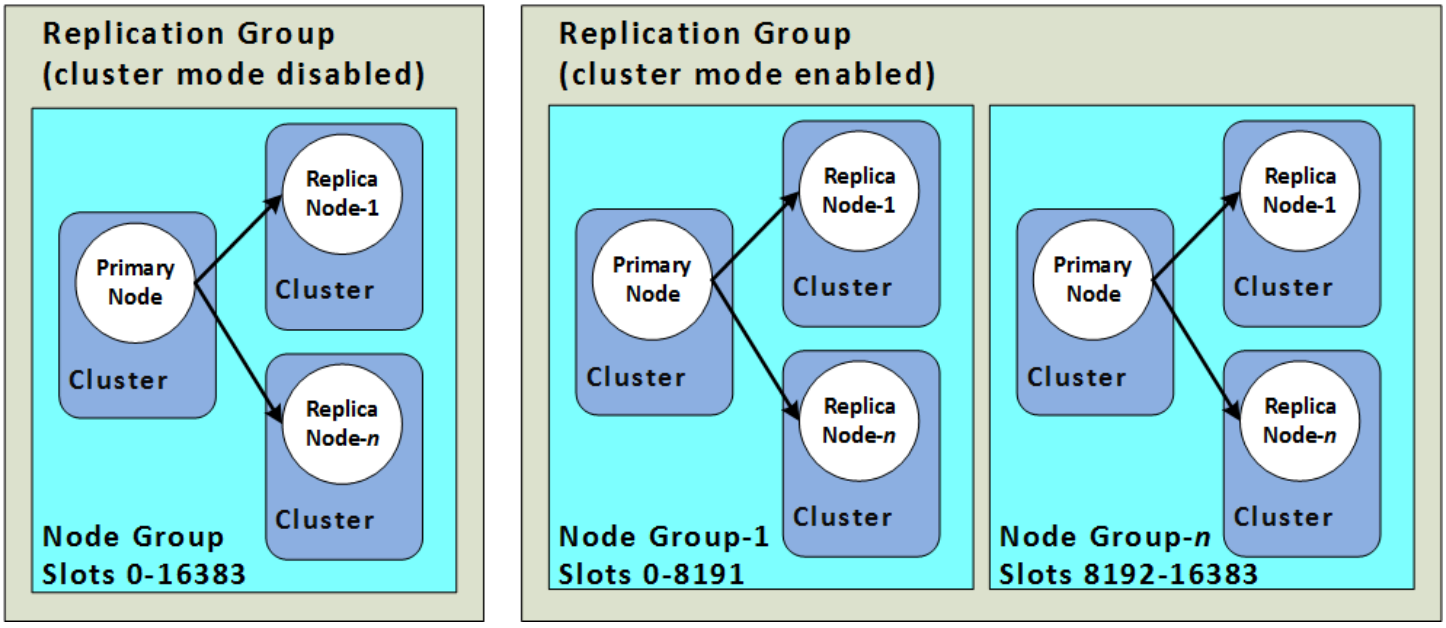
通过 API 查看 ElastiCache 节点状态

要使用 Amazon ElastiCache API 查看 ElastiCache 节点的状态，请调用`DescribeCacheClusteroperation`带`ShowCacheNodeInfo`标志的，以检索有关各个缓存节点的信息。

Redis 节点和分区

分片（在 API 和 CLI 中，为节点组）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。Redis 3.2 版及更高版本支持集群（在 API 和 CLI 中，为复制组）中的多个分片。此支持允许在 Redis（已启用集群模式）集群中对数据进行分区。

下图描述了 Redis（已禁用集群模式）集群与 Redis（已启用集群模式）集群的区别。



Redis（已启用集群模式）集群支持通过分片进行复制。API 操作 DescribeReplicationGroups（CLI：describe-replication-groups）可以列出带有成员节点的节点组、节点在节点组中的角色以及其他信息。

创建 Redis 集群时，请指定是否要创建已启用集群功能的集群。Redis（已禁用集群模式）集群永远只有一个分区，通过添加（总计最多 5 个）或删除只读副本节点可以对其进行横向扩展。有关更多信息，请参阅 使用复制组时的高可用性、向 Redis（已禁用集群模式）复制组添加只读副本 或 删除 Redis（已禁用集群模式）复制组的只读副本 。Redis（已禁用集群模式）集群也可以通过更改节点类型纵向扩展。有关更多信息，请参阅扩展具有副本节点的 Redis（已禁用集群模式）集群。

如果 Redis 引擎版本为 5.0.6 或更高版本，可将每个集群的节点或分片限制提高到最大值 500。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。有关更多信息，请参阅创建子网组。

对于低于 5.0.6 的版本，每个集群的限制为 250。

若要请求提高限制，请参阅 AWS Service Limits 并选择限制类型 Nodes per cluster per instance type（每个实例类型的每个集群的节点数）。

在创建 Redis（已启用集群模式）集群后，可以对其进行修改（横向扩展或缩减）。有关更多信息，请参阅 针对 Redis ElastiCache 进行扩展 和 替换节点。

创建新集群时，可以使用旧集群中的数据为其设定种子，以免从头开始创建。仅当集群组具有与旧集群相同数量的分片时，此方法才有效。如果您需要更改节点类型或引擎版本，这样做会很有用。有关更多信息，请参阅 进行手动备份 和 从备份还原到新缓存：

连接到节点

在尝试与 Redis 集群中的节点连接之前，您必须拥有适用于这些节点的终端节点。要找到终端节点，请参阅：

- 查找 Redis（已禁用集群模式）集群的端点（控制台）
- 查找 Redis（已启用集群模式）集群的端点（控制台）
- 查找端点（AWS CLI）
- 查找端点（ElastiCache API）

在以下示例中，您使用 redis-cli 实用工具连接到运行 Redis 的集群。

> **Note**
>
> 有关 Redis 以及可用 Redis 命令的更多信息，请参阅 http://redis.io/commands 网页。

使用 redis-cli 连接到 Redis 集群

1. 使用您选择的连接实用工具连接到 Amazon EC2 实例。

   > **Note**
   >
   > 有关如何连接到 Amazon EC2 实例的说明，请参阅 Amazon EC2 入门指南。

2. 要生成 `redis-cli`，请下载并安装 GNU Compiler Collection (`gcc`)。在 EC2 实例的命令提示符下输入下面的命令，然后在确认提示符下输入 `y`。

   ```
   sudo yum install gcc
   ```

   此时会显示类似以下内容的输出。

   ```
   Loaded plugins: priorities, security, update-motd, upgrade-helper
   Setting up Install Process
   Resolving Dependencies
   --> Running transaction check

   ...(output omitted)...
   ```

```
Total download size: 27 M
Installed size: 53 M
Is this ok [y/N]: y
Downloading Packages:
(1/11): binutils-2.22.52.0.1-10.36.amzn1.x86_64.rpm      | 5.2 MB     00:00
(2/11): cpp46-4.6.3-2.67.amzn1.x86_64.rpm                | 4.8 MB     00:00
(3/11): gcc-4.6.3-3.10.amzn1.noarch.rpm                  | 2.8 kB     00:00

...(output omitted)...

Complete!
```

3. 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。在 EC2 实例的命令提示符处，键入以下命令：

> **Note**
>
> 对于 Ubuntu 系统，在运行 make 之前，先运行 make distclean。

```
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
make distclean      # ubuntu systems only
make
```

4. 在 EC2 实例的命令提示符处，键入以下命令。

```
src/redis-cli -c -h mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com -p 6379
```

此时会显示类似于以下内容的 Redis 命令提示符。

```
redis mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com 6379>
```

5. 运行 Redis 命令测试连接。

您现已连接至集群并且可以运行 Redis 命令。以下是一些示例命令及其 Redis 响应。

```
set a "hello"          // Set key "a" with a string value and no expiration
```

```
OK
get a                   // Get value for key "a"
"hello"
get b                   // Get value for key "b" results in miss
(nil)
set b "Good-bye" EX 5  // Set key "b" with a string value and a 5 second expiration
get b
"Good-bye"
                    // wait 5 seconds
get b
(nil)                   // key has expired, nothing returned
quit                    // Exit from redis-cli
```

要连接到具有安全套接字层 (SSL) 加密（启用了传输中加密）的节点或集群，请参阅ElastiCache 传输中加密 (TLS)。

受支持的节点类型

ElastiCache 支持以下节点类型。一般而言，与其上一代类型对应项相比，最新一代类型以更低的成本提供了更多内存和计算能力。

有关各节点类型性能详细信息，请参阅 Amazon EC2 实例类型。

有关要使用的节点大小的信息，请参阅 选择节点大小。

最新一代

有关上一代的更多信息，请参阅上一代节点。

> **Note**
>
> 具备可突增网络性能的实例类型利用网络 I/O 积分机制，尽可能将其基准带宽突增到基准以上。

一般性问题

实例类型	支持的最低 Redis 版本	增强型 I/O（Redis 5.0.6+）	TLS 分载（Redis 6.2.5+）	增强型 I/O 多路复用（Redis 7.0.4+）	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.m7g.large	6.2	否	否	否	0.937	12.5
cache.m7g.xlarge	6.2	Y	Y	Y	1.876	12.5

实例类型	支持的最低 Redis 版本	增强型 I/O（Redis 5.0.6+）	TLS 分载（Redis 6.2.5+）	增强型 I/O 多路复用（Redis 7.0.4+）	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.m7g.2xlarge	6.2	Y	Y	Y	3.75	15
cache.m7g.4xlarge	6.2	Y	Y	Y	7.5	15
cache.m7g.8xlarge	6.2	Y	Y	Y	15	不适用
cache.m7g.12xlarge	6.2	Y	Y	Y	22.5	不适用
cache.m7g.16xlarge	6.2	Y	Y	Y	30	不适用
cache.m6g.large	5.0.6	否	否	否	0.75	10.0
cache.m6g.xlarge	5.0.6	Y	Y	Y	1.25	10.0
cache.m6g.2xlarge	5.0.6	Y	Y	Y	2.5	10.0

实例类型	支持的最低 Redis 版本	增强型 I/O（Redis 5.0.6+）	TLS 分载（Redis 6.2.5+）	增强型 I/O 多路复用（Redis 7.0.4+）	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.m6g.4xlarge	5.0.6	Y	Y	Y	5.0	10.0
cache.m6g.8xlarge	5.0.6	Y	Y	Y	12	不适用
cache.m6g.12xlarge	5.0.6	Y	Y	Y	20	不适用
cache.m6g.16xlarge	5.0.6	Y	Y	Y	25	不适用
cache.m5.large	3.2.4	否	否	否	0.75	10.0
cache.m5.xlarge	3.2.4	Y	否	否	1.25	10.0
cache.m5.2xlarge	3.2.4	Y	Y	Y	2.5	10.0
cache.m5.4xlarge	3.2.4	Y	Y	Y	5.0	10.0

实例类型	支持的最低 Redis 版本	增强型 I/O（Redis 5.0.6+）	TLS 分载（Redis 6.2.5+）	增强型 I/O 多路复用（Redis 7.0.4+）	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.m5.12xlarge	3.2.4	Y	Y	Y	12	不适用
cache.m5.24xlarge	3.2.4	Y	Y	Y	25	不适用
cache.m4.large	3.2.4	否	否	否	0.45	1.2
cache.m4.xlarge	3.2.4	Y	否	否	0.75	2.8
cache.m4.2xlarge	3.2.4	Y	Y	Y	1.0	10.0
cache.m4.4xlarge	3.2.4	Y	Y	Y	2.0	10.0
cache.m4.10xlarge	3.2.4	Y	Y	Y	5.0	10.0
cache.t4g.micro	3.2.4	否	否	否	0.064	5.0
cache.t4g.small	5.0.6	否	否	否	0.128	5.0
cache.t4g.medium	5.0.6	否	否	否	0.256	5.0

实例类型	支持的最低 Redis 版本	增强型 I/O（Redis 5.0.6+）	TLS 分载（Redis 6.2.5+）	增强型 I/O 多路复用（Redis 7.0.4+）	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.t3.micro	3.2.4	否	否	否	0.064	5.0
cache.t3.small	3.2.4	否	否	否	0.128	5.0
cache.t3.medium	3.2.4	否	否	否	0.256	5.0
cache.t2.micro	3.2.4	否	否	否	0.064	1.024
cache.t2.small	3.2.4	否	否	否	0.128	1.024
cache.t2.medium	3.2.4	否	否	否	0.256	1.024

内存优化

实例类型	支持的最低 Redis 版本	增强型 I/O（Redis 5.0.6+）	TLS 分载（Redis 6.2.5+）	增强型 I/O 多路复用（Redis 7.0.4+）	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.r7g.large	6.2	否	否	否	0.937	12.5
cache.r7g.xlarge	6.2	Y	Y	Y	1.876	12.5
cache.r7g.2xlarge	6.2	Y	Y	Y	3.75	15
cache.r7g.4xlarge	6.2	Y	Y	Y	7.5	15
cache.r7g.8xlarge	6.2	Y	Y	Y	15	不适用
cache.r7g.12xlarge	6.2	Y	Y	Y	22.5	不适用
cache.r7g.16xlarge	6.2	Y	Y	Y	30	不适用
cache.r6g.large	5.0.6	否	否	否	0.75	10.0
cache.r6g.xlarge	5.0.6	Y	Y	Y	1.25	10.0

实例类型	支持的最低 Redis 版本	增强型 I/O（Redis 5.0.6+）	TLS 分载（Redis 6.2.5+）	增强型 I/O 多路复用（Redis 7.0.4+）	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.r6g.2xlarge	5.0.6	Y	Y	Y	2.5	10.0
cache.r6g.4xlarge	5.0.6	Y	Y	Y	5.0	10.0
cache.r6g.8xlarge	5.0.6	Y	Y	Y	12	不适用
cache.r6g.12xlarge	5.0.6	Y	Y	Y	20	不适用
cache.r6g.16xlarge	5.0.6	Y	Y	Y	25	不适用
cache.r5.large	3.2.4	否	否	否	0.75	10.0
cache.r5.xlarge	3.2.4	Y	否	否	1.25	10.0
cache.r5.2xlarge	3.2.4	Y	Y	Y	2.5	10.0
cache.r5.4xlarge	3.2.4	Y	Y	Y	5.0	10.0

实例类型	支持的最低 Redis 版本	增强型 I/O（Redis 5.0.6+）	TLS 分载（Redis 6.2.5+）	增强型 I/O 多路复用（Redis 7.0.4+）	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.r5.12xlarge	3.2.4	Y	Y	Y	12	不适用
cache.r5.24xlarge	3.2.4	Y	Y	Y	25	不适用
cache.r4.large	3.2.4	否	否	否	0.75	10.0
cache.r4.xlarge	3.2.4	Y	否	否	1.25	10.0
cache.r4.2xlarge	3.2.4	Y	Y	Y	2.5	10.0
cache.r4.4xlarge	3.2.4	Y	Y	Y	5.0	10.0
cache.r4.8xlarge	3.2.4	Y	Y	Y	12	不适用
cache.r4.16xlarge	3.2.4	Y	Y	Y	25	不适用

利用数据分层功能优化内存

实例类型	支持的最低 Redis 版本	增强型 I/O（Redis 5.0.6+）	TLS 分载（Redis 6.2.5+）	增强型 I/O 多路复用（Redis 7.0.4+）	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.r6gd.xlarge	6.2.0	Y	否	否	1.25	10
cache.r6gd.2xlarge	6.2.0	Y	Y	Y	2.5	10
cache.r6gd.4xlarge	6.2.0	Y	Y	Y	5.0	10
cache.r6gd.8xlarge	6.2.0	Y	Y	Y	12	不适用
cache.r6gd.12xlarge	6.2.0	Y	Y	Y	20	不适用
cache.r6gd.16xlarge	6.2.0	Y	Y	Y	25	不适用

网络优化

实例类型	支持的最低 Redis 版本	增强型 I/O（Redis 5.0.6+）	TLS 分载（Redis 6.2.5+）	增强型 I/O 多路复用（Redis 7.0.4+）	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.c7gn.large	6.2	否	否	否	6.25	30
cache.c7gn.xlarge	6.2	Y	Y	Y	12.5	40
cache.c7gn.2xlarge	6.2	Y	Y	Y	25	50
cache.c7gn.4xlarge	6.2	Y	Y	Y	50	不适用
cache.c7gn.8xlarge	6.2	Y	Y	Y	100	不适用
cache.c7gn.12xlarge	6.2	Y	Y	Y	150	不适用
cache.c7gn.16xlarge	6.2	Y	Y	Y	200	不适用

AWS 区域支持的节点类型

支持的节点类型可能因 AWS 区域而异。有关更多详细信息，请参阅 Amazon ElastiCache 定价。

可突增性能实例

您可以在 Amazon ElastiCache 中启动通用型具爆发能力的 T4g、T3-Standard 和 T2-Standard 缓存节点。这些节点提供基准水平的 CPU 性能，并能随时突增 CPU 使用量，直至累积的积分耗尽。一个 CPU 积分提供一个完整 CPU 核心在一分钟内的性能。

Amazon ElastiCache 的 T4g、T3 和 T2 节点配置为标准节点，适用于平均 CPU 利用率始终低于实例基准性能的工作负载。为了突增到基准以上，节点会花费在其 CPU 积分余额中累积的积分。如果节点用完了累积的积分，则性能会逐步减低至基准性能水平。这种逐步减低的过程可确保在耗尽了累积的积分余额时，节点不会出现突然的性能下跌。有关更多信息，请参阅 Amazon EC2 用户指南中的具爆发能力的实例的 CPU 积分和基准性能。

下表列出了可突增性能节点类型以及每小时赚取 CPU 积分的速率。它还显示了一个节点可以累积所赚取的 CPU 积分最大数以及每个节点的 vCPU 数量。此外，它以一个完整核心百分比的形式提供基准性能水平（使用单个 vCPU）。

每小时获得的 CPU 积分	可累积获得的最大积分数*	vCPU	每个 vCPU 的基准性能	内存（GiB）	网络性能
12	288	2	10%	0.5	高达 5Gb
24	576	2	20%	1.37	高达 5Gb
24	576	2	20%	3.09	高达 5Gb
12	288	2	10%	0.5	高达 5Gb
24	576	2	20%	1.37	高达 5Gb
24	576	2	20%	3.09	高达 5Gb
6	144	1	10%	0.5	低到中

每小时获得的 CPU 积分	可累积获得的最大积分数*	vCPU	每个 vCPU 的基准性能	内存（GiB）	网络性能
12	288	1	20%	1.55	低到中
24	576	2	20%	3.22	低到中

* 可累积的积分数等于可在 24 小时周期内获得的积分数。

** 下表列出了每个 vCPU 的基准性能。一些节点大小具有多个 vCPU。对于这些情况，将 vCPU 百分比乘以 vCPU 数来计算节点的基准 CPU 使用率。

以下 CPU 积分指标对 T3 和 T4 可突增性能实例可用：

> **Note**
>
> 这些指标对具爆发能力的 T2 实例不可用。

- `CPUCreditUsage`
- `CPUCreditBalance`

有关这些指标的更多信息，请参阅 CPU 积分指标。

此外，请注意以下细节：

- 默认情况下，所有最新一代节点类型在基于 Amazon VPC 的虚拟私有云（VPC）中创建。
- T2 实例不支持 Redis 仅附加文件（AOF）。Redis 版本 2.8.22 及更高版本不支持 Redis 配置变量 `appendonly` 和 `appendfsync`。

相关信息

- Amazon ElastiCache 产品功能与详细信息
- Redis 特定的参数
- 传输中加密（TLS）

重启节点（仅限已禁用集群模式）

一些更改需要重启集群节点才能应用。例如，对于某些参数，对参数组中参数值的更改仅在重启后才会应用。

对于 Redis（已禁用集群模式）集群，这些参数为：

- activerehashing
- databases

只有使用 ElastiCache 控制台才能重启节点。一次只能重启单个节点。要重启多个节点，您必须为每个节点重复此过程。

> **Redis（已启用集群模式）参数更改**
>
> 如果要更改 Redis（已启用集群模式）集群上的以下参数，请按照随后的步骤操作。
>
> - activerehashing
> - databases
>
> 1. 创建集群的手动备份。请参阅进行手动备份。
> 2. 删除 Redis（已启用集群模式）集群。请参阅删除集群。
> 3. 使用修改的参数组和备份还原集群，以便为新集群创建种子。请参阅从备份还原到新缓存。
>
> 更改其他参数不需要执行此操作。

使用 AWS Management Console

可使用 ElastiCache 控制台重启节点。

重启节点（控制台）

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 从右上角的列表中，选择适用的 AWS 区域。

3. 在左侧导航窗格中，选择 Redis。

   此时会显示一个正在运行 Redis 的集群的列表。

4. 选择 Cluster Name（集群名称）下的集群。
5. 在 Node name（节点名称）下，选择要重启的节点旁边的单选按钮。
6. 选择 Actions（操作），然后选择 Reboot node（重启节点）。

要重启多个节点，请对要重启的每个节点重复步骤 2 到步骤 5。在重启另一个节点之前，您无需等待一个节点完成重启。

替换节点

Amazon ElastiCache for Redis 频繁升级其机群，将补丁和升级无缝地应用于实例。但是，我们需要经常重启您的 ElastiCache for Redis 节点，以将必需的操作系统更新应用于底层主机。我们需要进行升级来增强安全性、可靠性和操作性能，而应用这些升级就需要进行替换。

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

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

以下列表标识了在 ElastiCache 计划替换一个 Redis 节点时可执行的操作。要加快查找您的场景所需的信息，请从以下菜单中进行选择。

- Do nothing – 让 Amazon ElastiCache 按计划替换节点。
- Change your maintenance window – 将您的维护时段更改为更适合的时间。
- Redis（已启用集群模式）配置
  - Replace the only node in any Redis cluster – 使用备份和还原替换 Redis 集群中的节点的过程。
  - Replace a replica node in any Redis cluster – 在无需集群停机的情况下通过增加和减少只读副本数量来替换任何 Redis 集群中的只读副本的过程。
  - Replace any node in a Redis (cluster mode enabled) shard – 在无需集群停机的情况下通过横向扩展和横向缩减来替换 Redis（已启用集群模式）集群中的节点的动态过程。
- Redis（已禁用集群模式）配置
  - Replace the only node in any Redis cluster – 使用备份和还原替换 Redis 集群中的任何节点的过程。
  - Replace a replica node in any Redis cluster – 在无需集群停机的情况下通过增加和减少只读副本数量来替换任何 Redis 集群中的只读副本的过程。
  - Replace a node in a Redis (cluster mode disabled) cluster – 使用复制替换 Redis（已禁用集群模式）集群中的节点的过程。
  - Replace a Redis (cluster mode disabled) read-replica – 手动替换 Redis（已禁用集群模式）复制组中的只读副本的过程。

- Replace a Redis (cluster mode disabled) primary node – 手动替换 Redis（已禁用集群模式）复制组中的主节点的过程。

Redis 节点替换选项

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

  对于启用自动故障转移的非集群配置，Redis 5.0.6 及以上版本的集群将在集群继续保持在线并处理传入写入请求时完成替换。对于 4.0.10 或更低版本上启用了自动失效转移的集群，您可能会注意到与 DNS 更新相关的短暂写入中断（长达几秒钟）。

  如果节点是已启用自动故障转移的集群的成员，则 ElastiCache for Redis 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。

  对于设置为使用 ElastiCache for Redis 集群客户端的 ElastiCache for Redis 集群配置，在集群处理传入写请求时，立即完成替换。

  对于启用自动故障转移的非集群配置，Redis 5.0.6 及以上版本的集群将在集群继续保持在线并处理传入写入请求时完成替换。对于 4.0.10 或更低版本上启用了自动失效转移的集群，您可能会注意到与 DNS 更新相关的短暂写入中断（长达几秒钟）。

  如果节点是独立节点，则 Amazon ElastiCache 会首先启动替换节点，然后从现有节点同步。在这段时间内，现有节点不可用于处理服务请求。同步完成后，现有节点将会终止，新节点将取代它。ElastiCache 会尽最大努力在此操作期间保留您的数据。

- 更改维护时段 – 对于计划的维护事件，您将收到 ElastiCache 发送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，请参阅下列内容：
  - 修改集 ElastiCache 群
  - 修改复制组

> **Note**
>
> 仅当 ElastiCache 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。

- 替换任何 Redis 集群中仅有的节点 – 如果集群没有任何只读副本，您可以使用以下过程来替换节点。

  使用备份和还原替换仅有的节点

  1. 创建节点的集群的快照。有关说明，请参阅 进行手动备份。
  2. 以快照做种创建新集群。有关说明，请参阅 从备份还原到新缓存。
  3. 删除具有计划替换的节点的集群。有关说明，请参阅 删除集群。
  4. 在您的应用程序中，将旧节点的终端节点替换为新节点的终端节点。

- 替换任何 Redis 集群中的副本节点 – 要替换副本集群，请增加副本计数。为此，请添加副本，然后通过删除要替换的副本来减少副本计数。此过程是动态的，不会有任何集群停机。

> **Note**
>
> 如果您的分区或复制组已有五个副本，请反向操作步骤 1 和 2。

替换任何 Redis 集群中的副本

1. 通过将副本添加到分片或复制组来增加副本数量。有关更多信息，请参阅增加分区中的副本数量。
2. 删除要替换的副本。有关更多信息，请参阅减少分区中的副本数量。
3. 更新应用程序中的终端节点。

- 替换 Redis（已启用集群模式）分区中的任何节点 – 要替换集群中的节点而不停机，请使用线上重新分片。首先通过横向扩展来添加分片，然后通过缩减来删除具有要替换的节点的分片。

  替换 Redis（已启用集群模式）集群中的任何节点

  1. 扩展：添加其配置与具有要替换的节点的现有分片的配置相同的其他分片。有关更多信息，请参阅通过在线重新分片功能添加分片。
  2. 缩减：删除具有要替换的节点的分区。有关更多信息，请参阅通过在线重新分片功能删除分片。
  3. 更新应用程序中的终端节点。

- 替换 Redis（已禁用集群模式）集群中的节点 – 如果集群是没有任何只读副本的 Redis（已禁用集群模式）集群，请使用以下过程来替换节点。

  使用复制替换节点（仅限已禁用集群模式）

  1. 使用计划替换的节点作为主节点向集群添加复制。不要在此集群上启用多可用区。有关说明，请参阅 向没有分片的 Redis 集群添加复制。
  2. 将一个只读副本添加到该集群。有关说明，请参阅 向集群添加节点（控制台）。
  3. 将新创建的只读副本提升为主副本。有关说明，请参阅 将 Redis（已禁用集群模式）复制组的只读副本提升为主节点。
  4. 删除计划替换的节点。有关说明，请参阅 从集群中移除节点。
  5. 在您的应用程序中，将旧节点的终端节点替换为新节点的终端节点。

- 替换 Redis（已禁用集群模式）只读副本 – 如果节点是复制组中的只读副本，则替换该节点。

如果您的集群只具有一个副本节点，并且启用了多可用区，则必须禁用多可用区才能删除副本。有关说明，请参阅 修改复制组。

替换 Redis（已禁用集群模式）只读副本

1. 删除计划替换的副本。有关说明，请参阅：
   - 减少分区中的副本数量
   - 从集群中移除节点
2. 添加一个新副本来替换计划替换的副本。如果您使用的名称与刚删除的副本相同，可以跳过第 3 步。有关说明，请参阅：
   - 增加分区中的副本数量
   - 向 Redis（已禁用集群模式）复制组添加只读副本
3. 在您的应用程序中，将旧副本的终端节点替换为新副本的终端节点。
4. 如果您在开始时禁用了多可用区，现在请重新启用。有关说明，请参阅 启用多可用区 。

- 替换 Redis（已禁用集群模式）主节点 – 如果节点是主节点，请首先将一个只读副本提升为主节点。然后删除以前是主节点的副本。

  如果您的集群只具有一个副本，并且启用了多可用区，则必须禁用多可用区才能在步骤 2 中删除副本。有关说明，请参阅 修改复制组。

  替换 Redis（已禁用集群模式）主节点

  1. 将只读副本提升为主集群。有关说明，请参阅 将 Redis（已禁用集群模式）复制组的只读副本提升为主节点。
  2. 删除计划替换的节点（旧的主节点）。有关说明，请参阅 从集群中移除节点。
  3. 添加一个新副本来替换计划替换的副本。如果您使用的名称与刚删除的节点相同，可以跳过在应用程序中更改终端节点。

     有关说明，请参阅 向 Redis（已禁用集群模式）复制组添加只读副本。
  4. 在您的应用程序中，将旧节点的终端节点替换为新节点的终端节点。
  5. 如果您在开始时禁用了多可用区，现在请重新启用。有关说明，请参阅启用多可用区 。

ElastiCache 预留节点

预留一个或多个节点可能是一种降低成本的方法。预留节点需支付预付费用，此费用取决于节点类型和预留时间长短（一年或三年）。

要查看预留节点是否为您的使用案例节省了成本，请首先确定节点大小和所需节点数。然后估计节点的使用情况，并比较使用按需节点的总成本与使用预留节点的总成本。您可以在集群中混合搭配使用预留和按需节点。有关定价信息，请参阅 Amazon ElastiCache 定价。

> **Note**
>
> 预留节点不灵活；它们只适用于您预留的确切实例类型。

使用预留节点管理成本

预留一个或多个节点是一种降低成本的方法。预留节点需支付预付费用，此费用取决于节点类型和预留时间长短（一年或三年）。此费用远低于按需节点产生的每小时使用费。

要查看预留节点是否为您的使用案例节省了成本，请首先确定节点大小和所需节点数。然后估计节点的使用情况，并比较使用按需节点的总成本与使用预留节点的总成本。您可以在集群中混合搭配使用预留和按需节点。有关定价信息，请参阅 Amazon ElastiCache 定价。

AWS 区域、节点类型和有效期长度必须在购买时选择，以后不能更改。

您可以使用 AWS Management Console、AWS CLI 或 ElastiCache API 列出和购买可用的预留节点产品。

有关预留节点的更多信息，请参阅 Amazon ElastiCache 预留节点。

主题

标准预留节点产品

购买 Amazon ElastiCache 中的标准预留节点实例（RI），即表示您购买了在预留节点实例的持续时间内对某个特定节点实例类型和 AWS 区域享受折扣费率的承诺。要使用 Amazon ElastiCache 预留节点实例，您需要创建新的 ElastiCache 节点实例，就像您为按需实例创建该实例一样。

创建的新节点实例必须与预留节点实例的规格完全匹配。如果新节点实例的规格与您的账户的现有预留节点实例匹配，则会按照为预留节点实例提供的折扣费率向您收费。否则，将以按需费率对节点实例进行收费。这些标准 RI 可从 R5 和 M5 实例系列开始提供。

> **Note**
>
> 接下来讨论的所有三种产品类型均以一年和三年的期限提供。

产品类型

无预付 RI 提供对预留 ElastiCache 实例的访问，无需预付款。无论使用情况如何，您的无预付 预留 ElastiCache 实例都将按照期限内的小时数，采用打折小时费率进行计费。

部分预付：RI 需要预付部分预留 ElasticCache 实例费用。期限内剩余的小时数无论使用情况如何，都将按照打折小时费率计费。此选项替换了以前的高使用率 选项（将在下一部分中说明）。

预付全部费用 – RI 要求在 RI 有效期开始时支付全额费用。无论使用了多少小时数，剩余有效期内不会再产生其他任何费用。

旧式预留节点产品

旧式节点预留有三个级别：高利用率、中等利用率和低利用率。节点可在任意利用率级别预留一或三年。节点类型、利用率级别和预留期限将影响总成本。在购买预留节点之前，请通过比较各种模型确认预留节点是否可节省您的业务成本。

在一个利用率级别或期限购买的节点无法转换为不同的利用率级别或期限。

利用率级别

高利用率预留节点 可支持具备一致基准容量的工作负载或者运行稳定的工作负载。高利用率预留节点需要高预付款，但是如果您计划在 79% 以上的预留节点期内运行，则可最大程度地节省成本 (最多可达按需价格的 70%)。要使用高利用率预留节点，首先需要支付一次性费用。然后，不论节点是否运行，在有效期内以较低的小时费用计费。

如果计划使用预留节点的时间较多，但是希望支付较低的一次性费用或希望在关闭节点时停止计费，中等利用率预留节点是最佳选择。如果计划在 40% 以上的预留节点期限内运行，中等利用率预留节点是更加符合成本效益的选择。此选项能节省按需定价 64% 以上的费用。使用中等利用率预留节点，支付的一次性费用比低利用率预留节点略高，在节点运行时适用较低的小时使用费率。

低利用率预留节点 适合每天只运行几小时或者每周只运行几天的周期性工作负载。要使用低利用率预留节点，首先需要支付一次性费用，在节点运行时以小时使用费用折扣价计费。只要节点运行时间超过预留节点有效期的 17%，就可以节省成本。在预留节点的整个有效期内，您可以节省成本，节省成本最多可达按需费用的 56%。

旧式预留节点产品

提供物	预支费用	使用费	优势
高利用率	最高	最低小时费用。适用于整个期限，无论是否使用预留节点。	如果计划在 3 年期的 79% 以上的时间内运行预留节点，可最大程度降低总体成本。
中等利用率	中	对每个节点运行小时适用的小时使用费用。节点未运行时，不产生小时费用。	适合于弹性工作负载或者当您预期为中度使用（即超过 3 年期的 40%）。
低利用率	最低	对每个节点运行小时适用的小时使用费用。节点未运行时，不产生小时费用。所有产品类型的最高小时费用，但是该费用仅在预留节点运行时适用。	如果您计划一直运行节点，则总体成本最高。但是，如果您计划不经常（3 年期的 15% 以上的时间）使用预留节点，则总体成本最低。
按需使用（无预留节点）	无	最高小时费用。只要节点运行即适用。	最高小时成本。

有关更多信息，请参阅 Amazon ElastiCache 定价。

获取有关预留节点产品的信息

在购买预留节点之前，可了解有关可用的预留节点产品的信息。

以下示例演示如何使用 AWS Management Console、AWS CLI 和 ElastiCache API 获取有关可用的预留节点产品的定价和信息。

主题

获取有关预留节点产品的信息（控制台）

要使用 AWS Management Console获取有关可用预留集群产品的定价和其他信息，请使用以下过程。

获取有关可用预留节点产品的信息

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在导航窗格中，选择 Reserved Nodes。
3. 选择 Purchase Reserved Node（购买预留节点）。
4. 对于 Engine（引擎），选择 Redis。
5. 要确定可用的产品，请选择以下选项：

   - 节点类型
   - 期限
   - 产品类型

   在您做出这些选择后，每节点的费用和您的选择的总成本将显示在 Reservation details（预留详细信息）下。
6. 选择 Cancel 可避免购买这些节点和产生费用。

获取有关预留节点产品的信息 (AWS CLI)

若要获取有关可用预留节点产品的定价和其他信息，请在命令提示符处键入以下命令：

```
aws elasticache describe-reserved-cache-nodes-offerings
```

此操作将生成类似于以下内容的输出（JSON 格式）：

```
{
          "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
          "CacheNodeType": "cache.xxx.large",
          "Duration": 94608000,
          "FixedPrice": XXXX.X,
          "UsagePrice": X.X,
          "ProductDescription": "redis",
          "OfferingType": "All Upfront",
          "RecurringCharges": [
              {
                  "RecurringChargeAmount": X.X,
                  "RecurringChargeFrequency": "Hourly"
              }
          ]
 },
 {
          "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
          "CacheNodeType": "cache.xxx.xlarge",
          "Duration": 94608000,
          "FixedPrice": XXXX.X,
          "UsagePrice": X.X,
          "ProductDescription": "redis",
          "OfferingType": "Partial Upfront",
          "RecurringCharges": [
              {
                  "RecurringChargeAmount": X.XXX,
                  "RecurringChargeFrequency": "Hourly"
              }
          ]
 },
 {
          "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
          "CacheNodeType": "cache.xxx.large",
          "Duration": 31536000,
          "FixedPrice": X.X,
          "UsagePrice": X.X,
          "ProductDescription": "redis",
          "OfferingType": "No Upfront",
          "RecurringCharges": [
```

```
                {
                    "RecurringChargeAmount": X.XXX,
                    "RecurringChargeFrequency": "Hourly"
                }
            ]
}
```

有关更多信息，请参阅《AWS CLI 参考》中的 describe-reserved-cache-nodes-offerings。

获取有关预留节点产品的信息 (ElastiCache API)

若要获取有关可用预留节点产品的定价和信息，请调用 `DescribeReservedCacheNodesOfferings` 操作。

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReservedCacheNodesOfferings
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

有关更多信息，请参阅《ElastiCache API 参考》中的 DescribeReservedCacheNodesOfferings。

购买预留节点

以下示例演示如何使用 AWS Management Console、AWS CLI 和 ElastiCache API 购买预留节点产品。

> **Important**
>
> 按照本部分中的示例演示操作会在您的 AWS 账户中产生不可取消的费用。

主题

购买预留节点（控制台）

此示例演示如何购买预留节点 ID 为 myreservationID 的特定预留节点产品 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f。

以下过程使用 AWS Management Console通过提供 ID 来购买预留节点产品。

购买预留节点

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在导航列表中，选择 Reserved Nodes（预留节点）链接。
3. 选择 Purchase reserved nodes（购买预留节点）按钮。
4. 对于 Engine（引擎），选择 Redis。
5. 要确定可用的产品，请选择以下选项：
   - 节点类型
   - 期限
   - 产品类型
   - 一个可选的 Reserved node ID（预留节点 ID）

在您做出这些选择后，每节点的费用和您的选择的总成本将显示在 Reservation details（预留详细信息）下。

6. 选择 Purchase (购买)。

购买预留节点 (AWS CLI)

以下示例演示如何购买预留节点 ID 为 myreservationID 的特定预留集群产品 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f。

在命令提示符处输入以下命令：

对于 Linux、macOS 或 Unix：

```
aws elasticache purchase-reserved-cache-nodes-offering \
    --reserved-cache-nodes-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \
    --reserved-cache-node-id myreservationID
```

对于 Windows：

```
aws elasticache purchase-reserved-cache-nodes-offering ^
    --reserved-cache-nodes-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f ^
    --reserved-cache-node-id myreservationID
```

该命令返回的输出类似于下方内容：

```
RESERVATION  ReservationId      Class           Start Time                Duration
 Fixed Price  Usage Price  Count  State            Description      Offering Type
RESERVATION  myreservationid    cache.xx.small  2013-12-19T00:30:23.247Z  1y
 XXX.XX USD   X.XXX USD    1      payment-pending  memcached        Medium Utilization
```

有关更多信息，请参阅《AWS CLI 参考》中的 purchase-reserved-cache-nodes-offering。

购买预留节点 (ElastiCache API)

以下示例演示如何购买预留集群 ID 为 myreservationID 的特定预留节点产品 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f。

按照以下参数调用 `PurchaseReservedCacheNodesOffering` 操作：

- ReservedCacheNodesOfferingId = 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f
- ReservedCacheNodeID = myreservationID
- CacheNodeCount = 1

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=PurchaseReservedCacheNodesOffering
    &ReservedCacheNodesOfferingId=649fd0c8-cf6d-47a0-bfa6-060f8e75e95f
    &ReservedCacheNodeID=myreservationID
    &CacheNodeCount=1
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

有关更多信息，请参阅《ElastiCache API 参考》中的 PurchaseReservedCacheNodesOffering。

获取有关预留节点的信息

您可以使用 AWS Management Console、AWS CLI 和 ElastiCache API 获取有关已购买的预留节点的信息。

主题

获取有关预留节点的信息（控制台）

以下过程介绍如何使用 AWS Management Console获取有关您购买的预留节点的信息。

获取有关已购买的预留节点的信息

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在导航列表中，选择 Reserved nodes（预留节点）链接。

   您的账户的预留节点显示在 Reserved nodes（预留节点）列表中。您可选择列表中的任何预留节点，在控制台底部的详细信息窗格中查看有关该预留节点的详细信息。

获取有关预留节点的信息 (AWS CLI)

若要获取有关您的 AWS 账户的预留节点的信息，请在命令提示符处键入以下命令：

```
aws elasticache describe-reserved-cache-nodes
```

此操作将生成类似于以下内容的输出（JSON 格式）：

```
{
    "ReservedCacheNodeId": "myreservationid",
    "ReservedCacheNodesOfferingId": "649fd0c8-cf6d-47a0-bfa6-060f8e75e95f",
    "CacheNodeType": "cache.xx.small",
    "DataTiering": "disabled",
    "Duration": "31536000",
    "ProductDescription": "memcached",
    "OfferingType": "Medium Utilization",
    "MaxRecords": 0
```

```
}
```

有关更多信息，请参阅《AWS CLI 参考》中的 describe--reserved-cache-nodes。

获取有关预留节点的信息 (ElastiCache API)

若要获取有关您的 AWS 账户的预留节点的信息，请调用 DescribeReservedCacheNodes 操作。

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReservedCacheNodes
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

有关更多信息，请参阅《ElastiCache API 参考》中的 DescribeReservedCacheNodes。

迁移上一代节点

上一代节点是正在逐步停用的节点类型。如果您的现有集群未使用上一代节点类型，则 ElastiCache 不支持创建具有该节点类型的新集群。

由于上一代节点类型的数量有限，当某一节点在集群中运行状况不佳时，我们无法保证成功替换该节点。在这种情况下，您的集群可用性可能会受到负面影响。

我们建议您将集群迁移到新的节点类型，以获得更好的可用性和性能。有关要迁移到的建议节点类型，请参阅升级途径。有关 ElastiCache 中支持的节点类型和上一代节点类型的完整列表，请参阅 受支持的节点类型。

迁移 Redis 集群上的节点

以下过程介绍如何使用 ElastiCache 控制台迁移 Redis 集群节点类型。在此过程中，Redis 集群将继续处理请求，且停机时间降至最短。根据您的集群配置，您可能会遇到以下停机时间。以下是估计值，可能因您的具体配置而有所不同：

- 已禁用集群模式（单节点）的停机时间可能大约为 60 秒，主要原因是 DNS 传播。
- 对于运行 Redis 5.0.6 及以上版本的集群，已禁用集群模式（具有副本节点）的停机时间可能大约为 1 秒钟。所有较低版本的停机时间大约为 10 秒钟。
- 启用集群模式的停机时间可能大约为 1 秒钟。

要使用控制台修改 Redis 集群节点类型：

1. 登录到控制台并打开 ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 从导航窗格中，选择 Redis clusters（Redis 集群）。
3. 从集群列表中，选择要迁移的集群。
4. 选择 Actions（操作），然后选择 Modify（修改）。
5. 从节点类型列表中选择新的节点类型。
6. 如果您要立即执行迁移过程，请选择 Apply immediately（立即应用）。如果 Apply immediately（立即应用）处于未选中状态，则在此集群的下一维护时段内执行迁移过程。
7. 选择 Modify(修改)。如果您在上一步选择了 Apply immediately，则集群的状态将变为 modifying。当状态变为 available 时，即表示修改完成，您可以开始使用新集群。

要使用 AWS CLI 修改 Redis 集群节点类型：

使用 modify-replication-group API，如下所示：

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group /
 --replication-group-id my-replication-group /
 --cache-node-type new-node-type /
 --apply-immediately
```

对于 Windows：

```
aws elasticache modify-replication-group ^
 --replication-group-id my-replication-group ^
 --cache-node-type new-node-type ^
 --apply-immediately
```

在这种情况下，*new-node-type* 的值是您要迁移到的节点类型。通过传递 `--apply-immediately` 参数，当复制组从 modifying（正在修改）变为 available（可用）状态时，将立即应用更新。如果 Apply immediately（立即应用）处于未选中状态，则在此集群的下一维护时段内执行迁移过程。

> **Note**
>
> 如果您无法修改带有 `InvalidCacheClusterState` 错误的集群，则需要先删除还原失败的节点。

修复或移除 restore-failed-node

以下过程介绍如何从 Redis 集群中移除还原失败的节点。要详细了解 ElastiCache 节点如何进入还原失败状态，请参阅查看 ElastiCache 节点状态。我们建议先移除所有处于还原失败状态的节点，然后将 ElastiCache 集群中剩余的上一代节点迁移到新一代节点类型，最后再重新添加所需数量的节点。

要移除还原失败的节点（控制台）：

1. 登录到控制台并打开 ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 从导航窗格中，选择 Redis clusters（Redis 集群）。
3. 从集群列表中，选择要从中移除节点的集群。
4. 从分区列表中，选择要从中移除节点的分区。如果集群的集群模式已禁用，则跳过此步骤。
5. 从节点列表中，选择状态为 `restore-failed` 的节点。
6. 选择 Actions（操作），然后选择 Delete node（删除节点）。

从 ElastiCache 集群中移除还原失败的节点后，您现在可以迁移到新一代类型。有关更多信息，请参阅有关迁移 Redis 集群上的节点的上述内容。

要重新向 ElastiCache 集群添加节点，请参阅向集群添加节点。

管理集群

集群 是一个或多个缓存节点的集合，其中所有节点都运行 Redis 缓存引擎软件的实例。创建集群时，您需要指定所有节点将使用的引擎和版本。

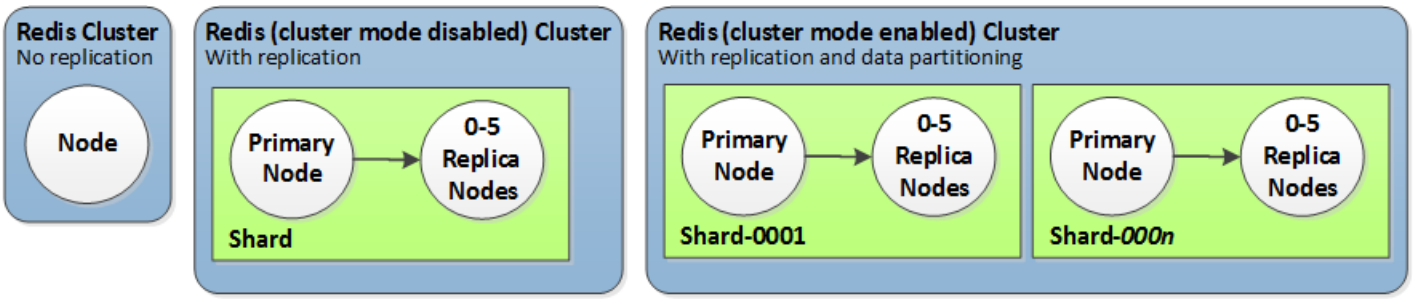
下图阐释了典型的 Redis 集群。Redis 集群可在分区（API/CLI：节点组）内包含单个节点或最多 6 个节点，单节点 Redis（已禁用集群模式）集群没有分区，多节点 Redis（已禁用集群模式）集群有一个

分区。Redis（已启用集群模式）集群最多可以拥有 500 个分区，并且跨分区对您的数据进行分区。如果 Redis 引擎版本为 5.0.6 或更高版本，可将每个集群的节点或分片限制提高到最大值 500。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。有关更多信息，请参阅创建子网组。对于低于 5.0.6 的版本，每个集群的限制为 250。

若要请求提高限制，请参阅 AWS Service Limits 并选择限制类型 Nodes per cluster per instance type（每个实例类型的每个集群的节点数）。

在一个分片中包含多个节点后，某个节点将作为读取/写入主节点。分片中的所有其他节点均为只读副本。

典型的 Redis 集群如下所示。



大多数 ElastiCache 操作都是在集群级别执行的。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。在与 ElastiCache API 和 AWS CLI 命令交互时，此标识符指定特定的集群。该客户在一个 AWS 区域中的集群标识符必须是唯一的。

ElastiCache 支持多个引擎版本。除非您有特定原因，否则我们建议您使用最新版本。

ElastiCache 集群专为使用 Amazon EC2 实例进行访问而设计。如果您根据 Amazon VPC 服务在 Virtual Private Cloud (VPC) 中启动集群，可从 AWS 外部进行访问。有关更多信息，请参阅从 AWS 外部访问 ElastiCache 资源。

有关受支持的 Redis 版本的列表，请参阅支持的 ElastiCache for Redis 版本。

选择网络类型

ElastiCache 支持互联网协议版本 4 和 6（IPv4 和 IPv6），允许您将集群配置为接受：

- 只有 IPv4 连接，
- 只有 IPv6 连接，
- 将解为 IPv4 和 IPv6 连接（双堆栈堆栈将解为）

在 Nitro 系统上构建的所有实例上使用 Redis 引擎版本 6.2 及更高版本的工作负载均支持 IPv6。通过 IPv6 访问 ElastiCache 不额外收费。

> **Note**
>
> 不支持迁移在 IPV6 /双栈可用之前创建的集群。也不支持在新创建的集群上切换网络类型。

为网络类型配置子网

如果您在 Amazon VPC 中创建集群，则必须指定一个子网组。ElastiCache 使用该子网组选择与节点关联的子网和子网中的 IP 地址。ElastiCache 集群需要一个双堆栈子网，同时分配 IPv4 和 IPv6 地址才能在双堆栈模式下运行，并需要仅 IPv6 子网才能作为仅 IPv6 运行。

使用双堆栈将解为

在启用集群模式下使用 ElastiCache for Redis 时，从应用程序的角度来看，通过配置端点连接到所有集群节点与直接连接到单个缓存节点没有什么不同。要实现此目的，集群感知客户端必须参与群集发现过程并请求所有节点的配置信息。Redis 的发现协议仅支持每个节点一个 IP。

为了保持与所有现有客户端的向后兼容性，引入了 IP 发现，它允许您在发现协议中选择要通告的 IP 类型（即 IPv4 或 IPv6）。虽然这将auto 发现限制为仅限于一种 IP 类型，但双栈仍然有利于支持集群模式的工作负载，因为它允许在不停机的情况下从 IPv4 迁移（或回滚）到 IPv6 发现 IP 类型。

启用 TLS 的双堆栈 ElastiCache 集群

为 ElastiCache 集群启用 TLS 时，集群发现函数（`cluster slots`、`cluster shards` 和 `cluster nodes`）将返回主机名而不是 IP。然后使用主机名代替 IP 来连接到 ElastiCache 集群并执行 TLS 握手。这意味着客户端不会受到 IP 发现参数的影响。对于启用 TLS 的集群，IP 发现参数对首选 IP 协议没有影响。相反，使用的 IP 协议将取决于客户端在解析 DNS 主机名时首选的 IP 协议。

有关在解析 DNS 主机名时如何配置 IP 协议首选项的示例，请参阅启用 TLS 的双堆栈 ElastiCache 集群。

使用 AWS Management Console

使用创建集群时AWS Management Console，在 “连接” 下选择一种网络类型，即 IPv4、IPv 6 或双堆栈。如果您要创建 Redis（已启用集群模式）集群并选择双堆栈，则必须选择发现 IP 类型，即 IPv6 或 IPv4。

有关更多信息，请参阅 创建 Redis（已启用集群模式）集群（控制台）或 创建 Redis（已禁用集群模式）（控制台）。

使用创建复制组时AWS Management Console，请选择网络类型，即 IPv4、IPv 6 或双堆栈。如果您选择双堆栈，则必须选择发现 IP 类型，即 IPv6 或 IPv4。

有关更多信息，请参阅 从头创建 Redis（已禁用集群模式）复制组或 从头开始在 Redis（已启用集群模式）中创建复制组。

使用 CLI

使用 CLI 创建缓存集群时，您可以使用 c reate-cache-cluster 命令并指定`NetworkType`和`IPDiscovery`参数：

对于 Linux、macOS 或 Unix：

```
aws elasticache create-cache-cluster \
    --cache-cluster-id "cluster-test" \
    --engine redis \
    --cache-node-type cache.m5.large \
    --num-cache-nodes 1 \
    --network-type dual_stack \
    --ip-discovery ipv4
```

对于 Windows：

```
aws elasticache create-cache-cluster ^
    --cache-cluster-id "cluster-test" ^
    --engine redis ^
    --cache-node-type cache.m5.large ^
    --num-cache-nodes 1 ^
    --network-type dual_stack ^
```

```
    --ip-discovery ipv4
```

使用 CLI 创建禁用集群模式的复制组时，您可以使用 cre ate-replication-group 命令并指定`NetworkType`和`IPDiscovery`参数：

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
   --replication-group-id sample-repl-group \
   --replication-group-description "demo cluster with replicas" \
   --num-cache-clusters 3 \
   --primary-cluster-id redis01 \
   --network-type dual_stack \
   --ip-discovery ipv4
```

对于 Windows：

```
aws elasticache create-replication-group ^
   --replication-group-id sample-repl-group ^
   --replication-group-description "demo cluster with replicas" ^
   --num-cache-clusters 3 ^
   --primary-cluster-id redis01 ^
   --network-type dual_stack ^
   --ip-discovery ipv4
```

在使用 CLI 创建启用集群模式的复制组并使用 IPv4 进行 IP 发现时，您可以使用 cre ate-replication-group 命令并指定 `NetworkType` 和 `IPDiscovery` 参数：

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
  --replication-group-id demo-cluster \
  --replication-group-description "demo cluster" \
  --cache-node-type cache.m5.large \
  --num-node-groups 2 \
  --engine redis \
  --cache-subnet-group-name xyz \
  --network-type dual_stack \
  --ip-discovery ipv4 \
  --region us-east-1
```

对于 Windows：

```
aws elasticache create-replication-group ^
  --replication-group-id demo-cluster ^
  --replication-group-description "demo cluster" ^
  --cache-node-type cache.m5.large ^
  --num-node-groups 2 ^
  --engine redis ^
  --cache-subnet-group-name xyz ^
  --network-type dual_stack ^
  --ip-discovery ipv4 ^
  --region us-east-1
```

在使用 CLI 创建启用集群模式的复制组并使用 IPv6 进行 IP 发现时，您可以使用 cre ate-replication-group 命令并指定 `NetworkType` 和 `IPDiscovery` 参数：

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
  --replication-group-id demo-cluster \
  --replication-group-description "demo cluster" \
  --cache-node-type cache.m5.large \
  --num-node-groups 2 \
  --engine redis \
  --cache-subnet-group-name xyz \
  --network-type dual_stack \
  --ip-discovery ipv6 \
  --region us-east-1
```

对于 Windows：

```
aws elasticache create-replication-group ^
  --replication-group-id demo-cluster ^
  --replication-group-description "demo cluster" ^
  --cache-node-type cache.m5.large ^
  --num-node-groups 2 ^
  --engine redis ^
  --cache-subnet-group-name xyz ^
  --network-type dual_stack ^
  --ip-discovery ipv6 ^
  --region us-east-1
```

数据分层

组成一个复制组并使用 r6gd 系列节点类型的集群将在内存和本地 SSD（固态硬盘）存储之间进行数据分层。借助数据分层功能，除可在内存中存储数据外，还可以在每个集群节点中使用成本更低的固态硬盘 (SSD)，从而为 Redis 工作负载提供新的高性价比选择。它非常适合经常访问的数据不超过总体数据集的 20% 的工作负载，以及能够容忍访问 SSD 中数据时所出现的额外延迟的应用程序。

在具有数据分层的集群上， ElastiCache 监控其存储的每个项目的最后访问时间。当可用内存 (DRAM) 被完全消耗时， ElastiCache使用最近最少使用的 (LRU) 算法将不经常访问的内容自动从内存移动到固态硬盘。随后访问固态硬盘上的数据时， ElastiCache 会在处理请求之前自动异步将其移回内存。如果您的工作负载只会经常访问部分数据，则数据分层将是经济高效地扩缩容量的极佳方法。

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

数据分层旨在将对应用程序工作负载的性能影响降至最低。例如，假设 500 字节的字符串值，与请求存储在内存中的数据相比，请求存储在 SSD 上的数据预计平均会增加 300 微秒的延迟。

如果使用最大型号的数据分层节点 (cache.r6gd.16xlarge)，您可以在单个 500 节点集群中存储最高 1PB 的数据（使用 1 个只读副本时 500TB）。数据分层与中支持的所有 Redis 命令和数据结构兼容。ElastiCache使用此功能无需任何客户端更改。

主题

最佳实践

我们建议您遵循以下最佳实践：

- 数据分层非常适合经常访问的数据不超过总体数据集的 20% 的工作负载，以及能够容忍访问 SSD 中数据时所出现的额外延迟的应用程序。
- 在数据分层节点上使用可用的 SSD 容量时，我们建议值大小大于键。在 DRAM 和 SSD 之间移动项目时，键将始终保留在内存中，并且只有值会移动到 SSD 层。

限制

数据分层功能存在以下限制：

- 您只能在复制组中的集群上使用数据分层。
- 您使用的节点类型必须属于 r6gd 系列，目前可在以下区域使用：us-east-2、us-east-1、us-west-2、us-west-1、eu-west-1、eu-central-1、eu-north-1、eu-west-3、ap-northeast-1、ap-southeast-1、ap-southeast-2、ap-south-1、ca-central-1 和 sa-east-1。
- 您必须使用 Redis 6.2 或更高版本的引擎。
- 除非两个集群都为 r6gd 集群，否则不能将 r6gd 集群的备份还原到其他集群。
- 不能将使用数据分层功能的集群备份导出到 Amazon S3。
- 在 r6gd 节点类型上运行的集群不支持在线迁移。
- 不支持将使用数据分层功能的集群（例如，使用 r6gd 节点类型的集群）扩缩至不使用数据分层功能的集群（例如，使用 r6g 节点类型的集群）。有关更多信息，请参阅 针对 Redis ElastiCache 进行扩展 。
- 对于 Redis 版本 7.0.7 及更高版本，使用数据分层的集群支持自动扩缩。有关更多信息，请参阅 ElastiCache 适用于 Redis 集群的 Auto Scaling
- 数据分层仅支持 volatile-lru、allkeys-lru、volatile-lfu、allkeys-lfu 和 noeviction maxmemory 策略。
- Redis 版本 7.0.7 及更高版本支持无分支保存。有关更多信息，请参阅 如何实施同步和备份。
- 大于 128MiB 的项目不会移动到 SSD。

定价

与 R6g 节点（仅内存）相比，R6gd 节点的总存储容量（内存 + SSD）提高了 4.8 倍，以最大利用率运行时可帮助实现超过 60% 的节省。有关更多信息，请参阅ElastiCache 定价。

监控

ElastiCache for Redis 提供了专为监控使用数据分层的性能集群而设计的指标。要监控 DRAM 中的项目与 SSD 的比例，可以使用 Redis 的指标中的 CurrItems 指标。您可以按以下方式计算百分比：（CurrItems 使用维度：层 = 内存 x 100）/（CurrItems 没有维度筛选器）。

如果配置的驱逐策略允许，那么 ElastiCache 当内存中的项目百分比降至5%以下时，Redis将开始驱逐项目。在配置了 noeviction 策略的节点上，写入操作将收到内存不足错误。

当内存中的项目百分比降至低于 5% 时，仍建议您考虑向上扩展已启用集群模式的集群，或者考虑向上扩展已禁用集群模式的集群。有关扩展的更多信息，请参阅扩展 Redis（启用集群模式）集群。有关使用数据分层的 Redis 集群指标的更多信息，请参阅。Redis 的指标

数据分层功能的使用

使用数据分层使用 AWS Management Console

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

有关创建集群的更多信息，请参阅创建集群。

使用启用数据分层 AWS CLI

使用创建复制组时 AWS CLI，您可以通过从 r6gd 系列中选择节点类型（例如 c ache.r6gd.xlar ge）并设置参数来使用数据分层。`--data-tiering-enabled`

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

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
   --replication-group-id redis-dt-cluster \
   --replication-group-description "Redis cluster with data tiering" \
   --num-node-groups 1 \
   --replicas-per-node-group 1 \
   --cache-node-type cache.r6gd.xlarge \
   --engine redis \
   --cache-subnet-group-name default \
   --automatic-failover-enabled \
   --data-tiering-enabled
```

对于 Windows：

```
aws elasticache create-replication-group ^
   --replication-group-id redis-dt-cluster ^
   --replication-group-description "Redis cluster with data tiering" ^
   --num-node-groups 1 ^
   --replicas-per-node-group 1 ^
   --cache-node-type cache.r6gd.xlarge ^
```

```
    --engine redis ^
    --cache-subnet-group-name default ^
    --automatic-failover-enabled ^
    --data-tiering-enabled
```

运行此操作后，您将会看到一条与以下类似的响应：

```
{
    "ReplicationGroup": {
        "ReplicationGroupId": "redis-dt-cluster",
        "Description": "Redis cluster with data tiering",
        "Status": "creating",
        "PendingModifiedValues": {},
        "MemberClusters": [
            "redis-dt-cluster"
        ],
        "AutomaticFailover": "enabled",
        "DataTiering": "enabled",
        "SnapshotRetentionLimit": 0,
        "SnapshotWindow": "06:00-07:00",
        "ClusterEnabled": false,
        "CacheNodeType": "cache.r6gd.xlarge",
        "TransitEncryptionEnabled": false,
        "AtRestEncryptionEnabled": false
    }
}
```

将数据从备份还原到启用数据分层的集群

您可以使用（控制台）、（ ）或（ElastiCache API）将备份还原到启用了数据分层的新集群。AWS CLI当您使用 r6gd 系列的节点类型创建集群时，系统会启用数据分层。

将数据从备份还原到启用数据分层的集群（控制台）

从备份还原到启用数据分层的集群（控制台）

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 从导航窗格中，选择 Backups（备份）。
3. 在备份列表中，选择您要从中进行还原的备份名称左侧的复选框。
4. 选择 Restore（还原）。

5. 完成 Restore Cluster（还原集群）对话框。务必要填写所有 Required（必填）字段以及您希望更改原定设置的任何其他字段。

   1. Cluster ID（集群 ID）– 必填。新集群的名称。
   2. Cluster mode enabled（scale out）[已启用集群模式（横向扩展）] – 对 Redis（已启用集群模式）集群选择此项。
   3. Node Type（节点类型）– 选择 cache.r6gd.xlarge 或 r6gd 系列中的任何其他节点类型。
   4. Number of Shards（分片数量）– 选择您希望新集群拥有的分片（API/CLI：节点组）数量。
   5. Replicas per Shard（每个分区的副本数）– 选择您希望各分区拥有的只读副本节点数量。
   6. Slots and keyspaces（槽和键空间）– 选择您希望如何在分区之间分布键。如果您选择指定键分配，请完成为各分片指定键范围的表。
   7. Availability zone(s)（可用区）– 指定您希望如何选择集群的可用区。
   8. Port（端口）– 仅当您希望新集群使用不同端口时才更改此项。
   9. Choose a VPC（选择 VPC）– 选择要在其中创建此集群的 VPC。
   10. Parameter Group（参数组）– 选择为所选节点类型预留了足够 Redis 内存开销的参数组。

6. 根据需要进行设置后，选择 Create（创建）。

有关创建集群的更多信息，请参阅创建集群。

将数据从备份还原到启用数据分层的集群 (AWS CLI)

使用创建复制组时，默认情况下 AWS CLI，通过从 r6gd 系列中选择节点类型（例如 c ache.r6gd.xlarge）并设置参数来使用数据分层。`--data-tiering-enabled`

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

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
   --replication-group-id redis-dt-cluster \
   --replication-group-description "Redis cluster with data tiering" \
   --num-node-groups 1 \
   --replicas-per-node-group 1 \
   --cache-node-type cache.r6gd.xlarge \
   --engine redis \
   --cache-subnet-group-name default \
   --automatic-failover-enabled \
```

```
    --data-tiering-enabled \
    --snapshot-name my-snapshot
```

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group ^
   --replication-group-id redis-dt-cluster ^
   --replication-group-description "Redis cluster with data tiering" ^
   --num-node-groups 1 ^
   --replicas-per-node-group 1 ^
   --cache-node-type cache.r6gd.xlarge ^
   --engine redis ^
   --cache-subnet-group-name default ^
   --automatic-failover-enabled ^
   --data-tiering-enabled ^
   --snapshot-name my-snapshot
```

运行此操作后，您将会看到一条与以下类似的响应：

```
{
    "ReplicationGroup": {
        "ReplicationGroupId": "redis-dt-cluster",
        "Description": "Redis cluster with data tiering",
        "Status": "creating",
        "PendingModifiedValues": {},
        "MemberClusters": [
            "redis-dt-cluster"
        ],
        "AutomaticFailover": "enabled",
        "DataTiering": "enabled",
        "SnapshotRetentionLimit": 0,
        "SnapshotWindow": "06:00-07:00",
        "ClusterEnabled": false,
        "CacheNodeType": "cache.r6gd.xlarge",
        "TransitEncryptionEnabled": false,
        "AtRestEncryptionEnabled": false
    }
}
```

准备集群

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

您还可以使用 AWS CloudFormation 创建 ElastiCache 集群。有关更多信息，请参阅《AWS Cloud Formation 用户指南》中的 AWS::ElastiCache::CacheCluster，其中包括关于如何实施这一方法的指导意见。

每当创建集群或复制组时，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

了解以下问题的答案有助于使集群的创建更加流畅：

- 您需要哪种节点实例类型？

  有关选择实例节点类型的指导信息，请参阅选择节点大小。
- 您是否会在基于 Amazon VPC 的 Virtual Private Cloud (VPC) 中启动集群？

  > **Important**
  >
  > 如果您打算在 VPC 中启动集群，则需要先在相同 VPC 中创建子网组，然后再开始创建集群。有关更多信息，请参阅子网和子网组。
  > ElastiCache 专为 AWS 使用 Amazon EC2 从内部进行访问而设计。但是，如果根据 Amazon VPC 在 VPC 中启动且集群位于 VPC 中，则可以提供从 AWS 外部进行访问的权限。有关更多信息，请参阅从 AWS 外部访问 ElastiCache 资源。
- 您是否需要自定义任何参数值？

  如果这样做，请创建自定义参数组。有关更多信息，请参阅创建参数组。

  如果您正在运行 Redis，请考虑设置 `reserved-memory` 或 `reserved-memory-percent`。有关更多信息，请参阅管理预留内存。
- 您是否需要创建自己的 VPC 安全组？

  有关更多信息，请参阅您的 VPC 的安全性。
- 您想如何实现容错？

有关更多信息，请参阅缓解故障。

主题

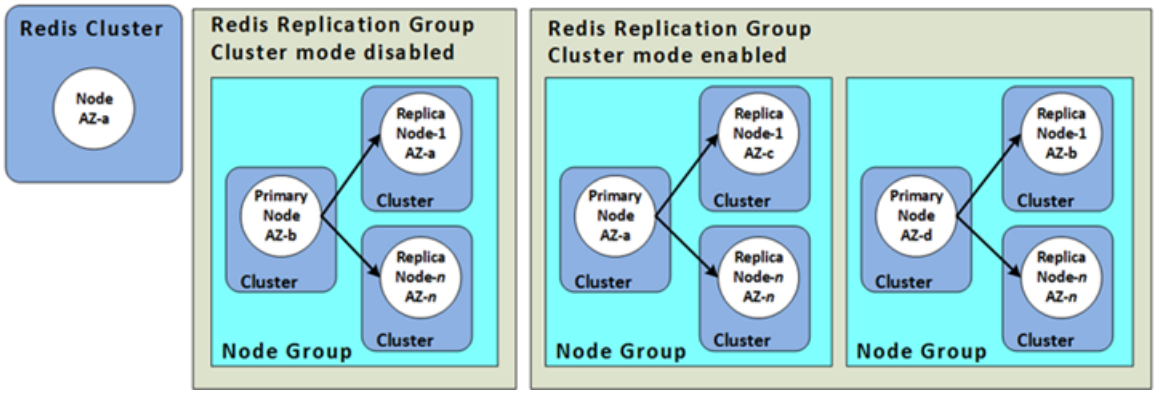
- 内存和处理器要求
- Redis 集群配置
- 扩展要求
- 访问要求
- 区域、可用区和 Local Zone 要求

内存和处理器要求

Amazon 的基本构建块 ElastiCache 是节点。配置单个节点，或成组配置节点以形成集群。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。

Redis 集群配置

ElastiCache 对于 Redis 的集群由 0 到 500 个分片（也称为节点组）组成。Redis 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 Redis 集群连接。Redis 分片中的节点执行这两个角色之一：一个读取/写入主节点以及所有其他辅助只读节点 (也称为只读副本)。除了节点终端节点外，Redis 集群本身还具有一个称为配置终端节点的终端节点。您的应用程序可以使用此终端节点对集群进行读取或写入，将从哪个节点读取或写入的决定权留给 Redis。 ElastiCache



有关更多信息，请参阅管理集群。

扩展要求

通过创建具有更大的新节点类型的新集群，可以对所有集群进行扩展。扩展 Redis 集群时，可以从备份中为其添加种子，并避免新集群开始为空。

有关更多信息，请参阅本指南中的针对 Redis ElastiCache 进行扩展 。

访问要求

根据设计，亚马逊 ElastiCache 集群是通过亚马逊 EC2 实例访问的。对 ElastiCache 集群的网络访问仅限于创建该集群的账户。因此，必须先授权 Amazon EC2 实例访问集群，然后您才能从 Amazon EC2 实例访问集群。执行此操作的步骤会有所变化，具体取决于启动到 EC2-VPC 还是 EC2-Classic。

如果您已将集群启动到 EC2-VPC，则需向集群授予网络入口。如果您在 EC2-Classic 中启动集群，则需要向与该实例关联的亚马逊弹性计算云安全组授予访问您的 ElastiCache 安全组的权限。有关详细说明，请参阅本指南中的 步骤 3：授予对集群的访问权限。

区域、可用区和 Local Zone 要求

Amazon ElastiCache 支持所有 AWS 区域。通过将 ElastiCache 集群放置在靠近应用程序的 AWS 区域，可以减少延迟。如果集群有多个节点，将节点放置在不同的可用区或 Local Zones 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
- 将 Local Zones 与 ElastiCache 结合使用
- 缓解故障

选择节点大小

为集群选择的节点大小会影响成本、性能和容错能力。

选择节点大小

有关 Graviton 处理器的优势的信息，请参阅 AWS Graviton 处理器。

回答以下问题可帮助您决定实施 Redis 所需的最小节点类型：

- 是否预期会出现采用多个客户端连接但吞吐量受限的工作负载？

  如果是这种情况，并且您运行的是 Redis 版本 5.0.6 或更高版本，则可以为 Redis 引擎使用增强型 I/O 功能来获得更好的吞吐量和延迟，此时可用的 CPU 用于分载客户端连接。如果您运行的是 Redis 版本 7.0.4 或更高版本，则除了增强型 I/O 之外，您还将通过增强型 I/O 多路复用来获得额外的加速，此时，每个专用网络 IO 线程利用 Redis 高效地批量处理命令的能力，将来自多个客户端的命令通过管道传送到 Redis 引擎。在 ElastiCache for Redis v7.1 及更高版本中，我们扩展了增强型 I/O

线程功能，使其还可以处理表示层逻辑。对于表示层，这是指增强型 I/O 线程现在不仅可以读取客户端输入，还可以将输入解析为 Redis 二进制命令格式，然后将其转发到主线程以便执行，从而提高性能。有关更多详细信息，请参阅博客文章和支持的版本页面。

- 您是否有仅会经常访问少部分数据的工作负载？

  如果是这种情况，并且您运行的是 Redis 6.2 版或更高版本的引擎，则可以通过选择 r6gd 节点类型来使用数据分层功能。使用数据分层功能时，最近极少使用的数据将存储到 SSD 中。在检索这些数据时，虽然延迟会轻微增加，但是可以节省成本。有关更多信息，请参阅数据分层。

  有关更多信息，请参阅受支持的节点类型。

- 您的数据共需要多少内存量？

  要获得一般估计值，请取要缓存的项目的大小。将此大小乘以同时要保留在缓存中的项目数。要获得项目大小的合理估计值，请先序列化您的缓存项目，再计算字符数。然后将该值除以集群中的分区数。

  有关更多信息，请参阅受支持的节点类型。

- 您运行 Redis 的哪个版本？

  对于 2.8.22 版之前的 Redis，您需要预留更多内存以用于故障转移、快照、同步和将副本提升为主节点的操作。之所以有此要求，是因为您必须具有足够的内存来执行过程中的所有写入。

  Redis 2.8.22 版和更高版本使用无分支保存过程，相比之前的过程，所需可用内存更少。

  有关更多信息，请参阅下列内容：

  - 如何实施同步和备份
  - 确保具有用于创建 Redis 快照的足够内存

- 您的应用程序的写操作有多密集？

  在拍摄快照或进行故障转移时，写操作密集的应用程序需要多得多的可用内存（ 数据未使用的内存）。无论何时执行 BGSAVE 进程，必须有足够的、数据未使用的内存，以容纳在 BGSAVE 过程中执行的所有写入。例如，拍摄快照时、将主集群与集群中的副本同步时以及启用仅附加文件 (AOF) 功能时。另一个示例是将副本提升为主节点时（如果您启用了多可用区）。最糟糕的情况是在此过程中重写所有数据。在这种情况下，您需要的节点实例大小是数据所需的内存量的两倍。

  有关更多详细信息，请参阅 确保具有用于创建 Redis 快照的足够内存。

- 您的实施是一个独立的 Redis（已禁用集群模式）集群，还是具有多个分区的 Redis（已启用集群模式）集群？

Redis（已禁用集群模式）集群

如果您实施的是 Redis（已禁用集群模式）集群，则节点类型必须能够容纳所有数据及必要的开销，如上一要点中所述。

例如，假设您估计所有项目的总大小为 12GB。在此情况下，您可以使用具有 13.3GB 内存的 cache.m3.xlarge 节点 或具有 13.5GB 内存的 cache.r3.large 节点。但是，您可能需要更多内存才能执行 BGSAVE 操作。如果您的应用程序写入操作繁重，请将内存要求翻倍至至少 24GB。因此，使用具有 27.9GB 内存的 cache.m3.2xlarge 或具有 30.5GB 内存的 cache.r3.xlarge。

具有多个分区的 Redis（已启用集群模式）

如果您实施的是具有多个分区的 Redis （已启用集群模式）集群，则节点类型必须能够容纳 bytes-for-data-and-overhead / number-of-shards 字节的数据。

例如，假设您估计所有项目的总大小为 12GB 且您具有两个分区。在此情况下，您可以使用具有 6.05GB 内存的 cache.m3.large 节点（12GB/2）。但是，您可能需要更多内存才能执行 BGSAVE 操作。如果您的应用程序写入操作繁重，请将内存要求翻倍至至少每个分区 12GB。因此，使用具有 13.3GB 内存的 cache.m3.xlarge 或具有 13.5GB 内存的 cache.r3.large。

- 您是否正在使用 Local Zones？

  Local Zones 使您能够将 ElastiCache 集群等资源放置在靠近用户的多个位置。但是，当您选择节点大小时，请注意，无论容量要求如何，本次可用节点大小目前仅限于以下内容：

  - 最新一代：

    M5 节点类型：cache.m5.large、cache.m5.xlarge、cache.m5.2xlarge、cache.m5.4xlarge、cache.

    R5 节点类型：cache.r5.large、cache.r5.xlarge、cache.r5.2xlarge、cache.r5.4xlarge、cache.

    T3 节点类型：cache.t3.micro、cache.t3.small、cache.t3.medium

您的集群运行期间，您可以监控发布到 CloudWatch 的内存使用率、处理器利用率、缓存命中数和缓存未命中数指标。您可能会注意到您的集群没有您想要的命中率，或者密钥被移出的频率过于频繁。在这些情况下，您可以选择具有较高 CPU 和内存规格的不同节点大小。

监控 CPU 使用情况时，请记住 Redis 是单线程的。因此，将报告的 CPU 使用率乘以 CPU 核心数来获得实际使用量。例如，报告的使用率为 20% 的四核 CPU 实际上相当于一个使用率为 80% 的单核 Redis。

创建集群

以下示例展示了如何使用 AWS Management Console、 AWS CLI 和 ElastiCache API 创建 Redis 集群。

创建 Redis（已禁用集群模式）（控制台）

ElastiCache 在使用 Redis 引擎时支持复制。要监控数据写入 Redis 读/写主集群与传播到只读辅助集群之间的延迟，请向集群 ElastiCache 添加一个特殊密钥。`ElastiCacheMasterReplicationTimestamp`此键为当前世界时（UTC）时间。因为 Redis 集群可能会在以后添加到复制组中，所以此键包含在所有 Redis 集群中，即使它们最初不是复制组的成员也会如此。有关复制组的更多信息，请参阅使用复制组时的高可用性。

要创建 Redis（已禁用集群模式）集群，请按照 创建 Redis（已禁用集群模式）集群（控制台） 中的步骤操作。

当您的集群状态为 available（可用）时，您可向其授予 Amazon EC2 访问权限，连接到集群并开始使用它。有关更多信息，请参阅 步骤 3：授予对集群的访问权限 和 步骤 4：连接到集群节点。

> **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 删除集群。

创建 Redis（已启用集群模式）集群（控制台）

如果运行的是 Redis 3.2.4 或更高版本，您可以创建 Redis（已启用集群模式）集群。Redis（已启用集群模式）集群支持将您的数据分配到 1 到 500 个分片（API/CLI：节点组）上，但存在一些限制。有关 Redis（已禁用集群模式）和 Redis（已启用集群模式）的对比，请参阅 支持的 ElastiCache for Redis 版本。

使用控制台创建 Redis (已启用集群模式) 集群 ElastiCache

1. 登录 AWS Management Console 并打开亚马逊 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 从右上角的列表中，选择要在其中启动此集群的 AWS 区域。
3. 从导航窗格中，选择 Get started（入门）。
4. 选择 Create VPC（创建 VPC）并按照创建虚拟私有云（VPC）中的步骤操作。

5. 在 ElastiCache 控制面板页面上，选择创建集群，然后选择创建 Redis 集群。
6. 在 Cluster settings（集群设置）下，执行以下操作：
   a. 选择 Configure and create a new cluster（配置和创建新集群）。
   b. 对于 Cluster mode（集群模式），选择 Enabled（已启用）。
   c. 对于 Cluster info（集群信息），为 Name（名称）输入一个值。
   d. （可选）为 Description（描述）输入一个值。
7. 在 Location（位置）下：

   AWS Cloud

   1. 对于 AWS Cloud，我们建议您接受 Multi-AZ（多可用区）和 Auto-failover（自动失效转移）的默认设置。有关更多信息，请参阅使用多可用区最大限度地缩短 Redis ElastiCache 的停机时间。
   2. 在 Cluster settings（集群设置）下
      a. 对于 Engine version（引擎版本），选择一个可用的引擎版本。
      b. 对于 Port（端口），使用默认端口 6379。如果您出于某个原因需要使用其他端口，请输入相应的端口号。
      c. 对于参数组，选择一个参数组或创建一个新参数组。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 和创建参数组。

         > **Note**
         > 当您选择要设置引擎配置值的参数组时，该参数组将应用于全局数据存储中的所有集群。在 Parameter Groups（参数组）页面上，是/否 Global（全局）属性指示参数组是否属于全局数据存储。

      d. 对于 Node type（节点类型），请选择向下箭头（ ）。
      在 Change node type（更改节点类型）对话框中，为所需节点类型选择 Instance family（实例系列）值。接着选择要用于此集群的节点类型，然后选择保存。

      有关更多信息，请参阅选择节点大小。

      如果您选择 r6gd 节点类型，则系统会自动启用数据分层。有关更多信息，请参阅数据分层。

e. 对于 Number of shards（分片数），选择要用于此 Redis （已启用集群模式）集群的分片（分区/节点组）数。

对于某些版本的 Redis（已启用集群模式），您可以动态更改集群中的分片数量：

- Redis 3.2.10 及更高版本– 如果您的集群运行 Redis 3.2.10 或更高版本，则可以动态更改集群中的分片数量。有关更多信息，请参阅扩展 Redis（启用集群模式）集群。
- 其他 Redis 版本 – 如果您的集群正在运行 3.2.10 版之前的 Redis 版本，则还有另一种方法。在这种情况下，要更改集群中的分片数量，请使用新分片数量创建一个新集群。有关更多信息，请参阅从备份还原到新缓存。

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。

Redis（已启用集群模式）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。
- 每个分片的只读副本数固定，无法更改。如果您需要增加或减少各分片（API/CLI：节点组）的副本数，您必须使用新的副本数量创建一个新集群。有关更多信息，请参阅使用外部创建的备份为新的自行设计的集群制作种子。

3. 在 Connectivity（连接）下

a. 对于 Network type（网络类型），选择此集群将支持的 IP 版本。

b. 对于子网组，请选择要应用于此集群的子网。 ElastiCache 使用该子网组选择子网和该子网内的 IP 地址以与您的节点关联。 ElastiCache 群集需要一个同时分配 IPv4 和 IPv6 地址的双栈子网才能在双堆栈模式下运行，并且需要一个仅限 IPv6 的子网才能作为仅限 IPv6 运行。

创建新的子网组时，输入其所属的 VPC ID。

选择 Discovery IP type（发现 IP 类型）。仅返回所选协议的 IP 地址。

有关更多信息，请参阅：

- 选择网络类型.
- 在您的 VPC 中创建子网。

如果您是 将 Local Zones 与 ElastiCache 结合使用 ，则必须创建或选择位于本地区域中的子网。

有关更多信息，请参阅子网和子网组。

4. 对于 Availability zone placements（可用区位置），您有两种选择：

   - 无偏好 — ElastiCache 选择可用区。
   - Specify availability zones（指定可用区）– 您为各集群指定可用区。

     如果您选择指定可用区，则需从列表中为各分片中的每个集群选择可用区。

   有关更多信息，请参阅选择区域和可用区。

5. 选择 Next（下一步）

6. 在 Advanced Redis settings（高级 Redis 设置）下：

   - 对于 Security（安全）：

     i. 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**
       >
       > 您可以选择提供不同的加密密钥，方法是选择 “客户托管 AWS KMS 密钥” 并选择密钥。有关更多信息，请参阅使用 AWS KMS 客户自主管理型密钥。

     - Encryption in-transit（传输中加密）– 对传输中数据启用加密。有关更多信息，请参阅传输中加密。对于 Redis 6.0 及以上的引擎版本，如果启用了传输中加密，则系统会提示您指定以下 Access Control（访问控制）选项中的一个：

       - No Access Control（无访问控制）– 此选项为默认设置。这表示对用户访问集群的权限没有任何限制。

- User Group Access Control List（用户组访问控制列表）– 选择具有集群访问权限的已定义用户组。有关更多信息，请参阅使用控制台和 CLI 管理用户组。
- Redis AUTH Default User（Redis AUTH 默认用户）– Redis 服务器的身份验证机制。有关更多信息，请参阅 Redis AUTH。

- Redis AUTH – Redis 服务器的身份验证机制。有关更多信息，请参阅 Redis AUTH。

> **Note**
>
> 对于 Redis 3.2.6 以上的版本（版本 3.2.10 除外），只能选择 Redis AUTH。

ii. 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。

7. 如果需要定期计划自动备份，请选择启用自动备份，然后输入每个自动备份在被自动删除前保留的天数。如果您不希望定期计划自动备份，请清除 Enable automatic backups 复选框。不论是哪种情况，您始终可以选择创建手动备份。

   有关 Redis 备份和还原的更多信息，请参阅快照和还原。

8. （可选）指定维护时段。维护时段 是每周中 ElastiCache 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 ElastiCache 选择维护时段的日期和时间（No preference (无首选项)），或者自行选择日期、时间和持续时间（Specify maintenance window (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。

9. （可选）对于 Logs（日志）：

   - 在 Log format（日志格式）下，选择 Text（文本）或 JSON。
   - 在目标类型下，选择CloudWatch 日志或 Kinesis Fire hose。

- 在“日志目标”下，选择“新建”并输入您的 CloudWatch 日志日志组名称或 Firehose 直播名称，或者选择“选择现有”，然后选择您的 CloudWatch 日志日志组名称或 Firehose 直播名称，

10. 对于标签，为了帮助您管理集群和其他 ElastiCache 资源，您可以以标签的形式为每个资源分配自己的元数据。有关更多信息，请参阅 标记 ElastiCache 资源。
11. 选择 Next（下一步）。
12. 查看您的所有输入和选择，然后进行任意所需的更正。当您准备好后，选择 Create（创建）。

On premises

1. 对于 On premises（本地），我们建议您保留 Auto-failover（自动失效转移）为启用状态。有关更多信息，请参阅使用多可用区最大限度地缩短 Redis ElastiCache 的停机时间
2. 按照使用 Outposts 中的步骤操作。

要使用 ElastiCache API 或 AWS CLI 代替 ElastiCache 控制台创建等效项，请参阅以下内容：

- API: CreateReplicationGroup
- CLI：create-replication-group

当您的集群状态为 available 时，您可向其授予 EC2 访问权限，连接到集群并开始使用它。有关更多信息，请参阅 步骤 3：授予对集群的访问权限 和 步骤 4：连接到集群节点。

> **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 删除集群。

创建集群（AWS CLI）

要使用创建集群 AWS CLI，请使用`create-cache-cluster`命令。

> **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 删除集群。

创建 Redis（已禁用集群模式）集群（CLI）

Example – 一个无只读副本的 Redis（已禁用集群模式）集群

下面的 CLI 代码创建一个无副本的 Redis（已禁用集群模式）缓存群集。

> **Note**
>
> 使用 r6gd 系列的节点类型创建集群时，必须传递 `data-tiering-enabled` 参数。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-cache-cluster \
--cache-cluster-id my-cluster \
--cache-node-type cache.r4.large \
--engine redis \
--num-cache-nodes 1 \
--cache-parameter-group default.redis6.x \
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

对于 Windows：

```
aws elasticache create-cache-cluster ^
--cache-cluster-id my-cluster ^
--cache-node-type cache.r4.large ^
--engine redis ^
--num-cache-nodes 1 ^
--cache-parameter-group default.redis6.x ^
--snapshot-arns arn:aws:s3:::my_bucket/snapshot.rdb
```

创建 Redis（已启用集群模式）集群（AWS CLI）

Redis（已启用集群模式）集群（API/CLI：复制组）不能使用 `create-cache-cluster` 操作创建。要创建 Redis（已启用集群模式）集群（API/CLI：复制组），请参阅 从头开始创建 Redis（已启用集群模式）复制组（AWS CLI）。

有关更多信息，AWS CLI 请参阅 ElastiCache 参考主题`create-replication-group`。

创建集群 (ElastiCache API)

要使用 ElastiCache API 创建集群，请使用`CreateCacheCluster`操作。

> **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 删除集群。

主题

- 创建 Redis (已禁用集群模式) 缓存集群 (ElastiCache API)
- 在 Redis 中创建缓存集群（已启用集群模式）(ElastiCache API)

创建 Redis (已禁用集群模式) 缓存集群 (ElastiCache API)

以下代码创建 Redis（已禁用集群模式）缓存集群 (ElastiCache API)。

添加换行符以便于阅读。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=CreateCacheCluster
    &CacheClusterId=my-cluster
    &CacheNodeType=cache.r4.large
    &CacheParameterGroup=default.redis3.2
    &Engine=redis
    &EngineVersion=3.2.4
    &NumCacheNodes=1
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &SnapshotArns.member.1=arn%3Aaws%3As3%3A%3A%3AmyS3Bucket%2Fdump.rdb
    &Timestamp=20150508T220302Z
    &Version=2015-02-02
```

```
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20150508T220302Z
    &X-Amz-Expires=20150508T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Signature=<signature>
```

在 Redis 中创建缓存集群（已启用集群模式）(ElastiCache API)

Redis（已启用集群模式）集群（API/CLI：复制组）不能使用 `CreateCacheCluster` 操作创建。要创建 Redis（已启用集群模式）集群（API/CLI：复制组），请参阅 从 Redis（已启用集群模式）中从头开始创建复制组 (ElastiCache API)。

有关更多信息，请参阅 ElastiCache API 参考主题`CreateReplicationGroup`。

查看集群的详细信息

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

查看 Redis（已禁用集群模式）集群（控制台）的详细信息

您可以使用 ElastiCache 控制台、AWS CLI for ElastiCache，或 ElastiCache API 查看 Redis（已禁用集群模式）集群的详细信息。

以下过程详细说明了如何使用 ElastiCache 控制台查看 Redis（已禁用集群模式）集群的详细信息。

查看 Redis（已禁用集群模式）集群的详细信息

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 在 ElastiCache 控制台控制面板中，选择 Redis 可显示运行任意 Redis 版本的所有集群的列表。
3. 要查看集群的详细信息，请选择集群名称左侧的复选框。确保所选的集群运行 Redis 引擎，而不是 Clustered Redis 引擎。执行此操作将显示该集群的详细信息，包括集群的主端点。
4. 查看节点信息：
   a. 选择集群的名称。
   b. 选择 Shards and nodes（分片和节点）选项卡。执行此操作将显示每个节点的详细信息，包括节点中用于从集群进行读取的端点。
5. 要查看指标，请选择 Metrics（指标）选项卡，该选项卡显示集群中所有节点的相关指标。有关更多信息，请参阅使用 CloudWatch 指标监控使用情况。
6. 要查看日志，请选择 Logs（日志）选项卡，该选项卡指示集群使用的是慢日志还是引擎日志，并提供相关详细信息。有关更多信息，请参阅日志传输。
7. 选择 Network and security（网络和安全）选项卡，可查看有关集群的网络连接和子网组配置的详细信息。有关更多信息，请参阅子网和子网组。
8. 选择 Maintenance（维护）选项卡，可查看有关集群维护设置的详细信息。有关更多信息，请参阅管理维护。
9. 选择 Service updates（服务更新）选项卡，可查看所有可用服务更新的详细信息及其建议的应用截止日期。有关更多信息，请参阅中的服务更新 ElastiCache。
10. 选择 Tags（标签）选项卡，可查看应用于集群资源的任何标签的详细信息。有关更多信息，请参阅标记 ElastiCache 资源。

查看 Redis（已启用集群模式）集群的详细信息（控制台）

您可以使用 ElastiCache 控制台、AWS CLI for ElastiCache，或 ElastiCache API 查看 Redis（已启用集群模式）集群的详细信息。

以下过程详细说明了如何使用 ElastiCache 控制台查看 Redis（已启用集群模式）集群的详细信息。

查看 Redis（已启用集群模式）集群的详细信息

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 从右上角的列表中，选择您感兴趣的 AWS 区域。
3. 在 ElastiCache 控制台控制面板中，选择 Redis 可显示运行任意 Redis 版本的所有集群的列表。
4. 要查看 Redis（已启用集群模式）集群的详细信息，请选择集群名称左侧的复选框。确保选择的是运行 Clustered Redis 引擎而不是 Redis 引擎的集群。

   该集群下方的屏幕会扩展，显示有关该集群的详细信息，包括该集群的配置端点。
5. 要查看集群分片的列表及每个分片的节点数量，请选择 Shards and nodes（分片和节点）选项卡。
6. 查看有关节点的特定信息：

   - 选择分片的 ID。

     这将显示有关每个节点的信息，包括用于从集群中读取数据的每个节点的端点。
7. 要查看指标，请选择 Metrics（指标）选项卡，该选项卡显示集群中所有节点的相关指标。有关更多信息，请参阅使用 CloudWatch 指标监控使用情况。
8. 要查看日志，请选择 Logs（日志）选项卡，该选项卡指示集群使用的是慢日志还是引擎日志，并提供相关详细信息。有关更多信息，请参阅日志传输。
9. 选择 Network and security（网络和安全）选项卡，可查看有关集群的网络连接和子网组配置、VPC 安全组以及集群上启用的加密方法（如有）的详细信息。有关更多信息，请参阅 子网和子网组 和 Amazon ElastiCache 中的数据安全性：
10. 选择 Maintenance（维护）选项卡，可查看有关集群维护设置的详细信息。有关更多信息，请参阅管理维护。
11. 选择 Service updates（服务更新）选项卡，可查看所有可用服务更新的详细信息及其建议的应用截止日期。有关更多信息，请参阅中的服务更新 ElastiCache。
12. 选择 Tags（标签）选项卡，可查看应用于集群资源的任何标签的详细信息。有关更多信息，请参阅标记 ElastiCache 资源。

查看集群的详细信息（AWS CLI）

以下代码列出 *my-cluster* 的详细信息：

```
aws elasticache describe-cache-clusters --cache-cluster-id my-cluster
```

如果使用 create-cache-cluster 命令创建具有 1 个缓存节点和 0 个分片的集群，请用集群的名称替换 *my-cluster*。

```
{
    "CacheClusters": [
        {
            "CacheClusterStatus": "available",
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "wed:12:00-wed:13:00",
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "08:30-09:30",
            "TransitEncryptionEnabled": false,
            "AtRestEncryptionEnabled": false,
            "CacheClusterId": "my-cluster1",
            "CacheClusterCreateTime": "2018-02-26T21:06:43.420Z",
            "PreferredAvailabilityZone": "us-west-2c",
            "AuthTokenEnabled": false,
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
           "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis3.2"
            },
            "SnapshotRetentionLimit": 0,
            "AutoMinorVersionUpgrade": true,
            "EngineVersion": "3.2.10",
            "CacheSecurityGroups": [],
```

```
            "NumCacheNodes": 1
        }
```

```
{
    "CacheClusters": [
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": false,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:13:24.250Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": false,
            "PreferredAvailabilityZone": "us-west-2a",
            "TransitEncryptionEnabled": false,
            "ReplicationGroupId": "my-cluster2",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "sun:08:30-sun:09:30",
            "CacheClusterId": "my-cluster2-001",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
            "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
```

```
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": false,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:13:24.250Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": false,
            "PreferredAvailabilityZone": "us-west-2b",
            "TransitEncryptionEnabled": false,
            "ReplicationGroupId": "my-cluster2",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "sun:08:30-sun:09:30",
            "CacheClusterId": "my-cluster2-002",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
            "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": false,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
```

```
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:13:24.250Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": false,
            "PreferredAvailabilityZone": "us-west-2c",
            "TransitEncryptionEnabled": false,
            "ReplicationGroupId": "my-cluster2",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "sun:08:30-sun:09:30",
            "CacheClusterId": "my-cluster2-003",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
            "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis3.2"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "3.2.10",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        }
```

```
{
    "CacheClusters": [
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2a",
```

```
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0001-001",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
            "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2b",
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0001-002",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
```

```
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis3.2.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "3.2.6",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2c",
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0001-003",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
```

```
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2b",
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0002-001",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
```

```
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2c",
            "TransitEncryptionEnabled": true,
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0002-002",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis3.2.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "3.2.6",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        },
        {
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "AuthTokenEnabled": true,
            "CacheSubnetGroupName": "default",
            "SnapshotWindow": "12:30-13:30",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterCreateTime": "2018-02-26T21:17:01.439Z",
            "CacheClusterStatus": "available",
            "AtRestEncryptionEnabled": true,
            "PreferredAvailabilityZone": "us-west-2a",
            "TransitEncryptionEnabled": true,
```

```
            "ReplicationGroupId": "my-cluster3",
            "Engine": "redis",
            "PreferredMaintenanceWindow": "thu:11:00-thu:12:00",
            "CacheClusterId": "my-cluster3-0002-003",
            "PendingModifiedValues": {},
            "CacheNodeType": "cache.r4.large",
             "DataTiering": "disabled",
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "ParameterApplyStatus": "in-sync",
                "CacheParameterGroupName": "default.redis6.x.cluster.on"
            },
            "SnapshotRetentionLimit": 0,
            "EngineVersion": "6.0",
            "CacheSecurityGroups": [],
            "NumCacheNodes": 1
        }
    ]
}
```

如果使用 AWS Management Console创建集群（启用或禁用集群节点，具有一个或多个分片），请使用以下命令描述集群的详细信息 [将 *my-cluster* 替换为复制组的名称（集群的名称）]：

```
aws elasticache describe-replication-groups --replication-group-id my-cluster
```

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

查看集群的详细信息（ElastiCache API）

您可以使用 ElastiCache API `DescribeCacheClusters` 操作查看集群的详细信息。如果包含 `CacheClusterId` 参数，则将返回指定的集群的详细信息。如果省略 `CacheClusterId` 参数，则会返回最多 `MaxRecords` 个（默认 100 个）集群的详细信息。`MaxRecords` 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://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheClusters
   &CacheClusterId=my-cluster
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
```

```
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

以下代码列出了最多 25 个集群的详细信息。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheClusters
   &MaxRecords=25
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 ElastiCache API 参考主题 DescribeCacheClusters。

修改集 ElastiCache 群

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

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

在更改集群的参数时，所做的更改将立即或在重新启动集群后应用于集群。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。要确定何时应用特定的参数更改，请参阅 Redis 特定的参数 的表中详细信息列的更改生效部分。

使用 AWS Management Console

修改集群

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 从右上角的列表中，选择要修改的集群所在的 AWS 区域。
3. 在导航窗格中，选择在您要修改的集群上运行的引擎。

   此时会显示选定引擎的集群列表。
4. 在集群列表中，对于要修改的集群，选择其名称。
5. 选择 Actions（操作），然后选择 Modify（修改）。

   Modify Cluster（修改集群）窗口随即出现。
6. 在修改集群窗口中，根据需要做出修改。选项包括：

   - 描述
   - 集群模式 - 要将集群模式从已禁用修改为已启用，必须先将集群模式设置为兼容。

     兼容模式允许您的 Redis 客户端使用“已启用集群模式”和“已禁用集群模式”进行连接。在将所有 Redis 客户端迁移为使用“已启用集群模式”后，您可以完成集群模式配置并将集群模式设置为已启用。
   - 引擎版本兼容性

> **Important**
>
> 您可以升级到较新的引擎版本。升级主要引擎版本（例如从 5.0.6 升级到 6.0）时，您需要选择一个与新引擎版本兼容的参数组系列。有关执行此操作的更多信息，请参阅 引擎版本和升级 。不过，您不能降级到较早的引擎版本，除非删除现有集群并重新创建它。

- VPC 安全组
- 参数组
- 节点类型

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

- 多可用区
- 自动故障转移（仅限已禁用集群模式）
- 启用自动备份
- 备份节点 ID
- 备份保留期
- 备份时段
- SNS 主题通知

Apply Immediately（立即应用）框仅适用于引擎版本修改。要立即应用更改，请选中 Apply Immediately（立即应用）复选框。如果未选中此框，则将在下一维护时段内应用节点类型和引擎版本修改。诸如更改维护时段这样的其他修改是立即应用的。

7. 选择 Modify(修改)。

启用/禁用日志传输

1. 从集群列表中，选择要修改的集群。选择 Cluster name（集群名称）而不是旁边的复选框。
2. 在 Cluster details（集群详细信息）页面上，选择 Logs（日志）选项卡，
3. 要启用/禁用慢日志，请选择 Enable（启用）或 Disable（禁用）。

如果您选择启用：

a. 在 Log format（日志格式）下，选择 JSON 或 Text（文本）。
b. 在日志目标类型下，选择CloudWatch 日志或 Kinesis Fire hose。
c. 在 “日志目标” 下，选择 “新建”，然后输入您的 CloudWatchLogs 日志组名称或 Kinesis Data Firehose 直播名称。或者选择 “选择现有”，然后选择您的 CloudWatchLogs 日志组名称或 Kinesis Data Firehose 直播名称。
d. 请选择 启用。

更改配置：

1. 选择 Modify（修改）
2. 在 Log format（日志格式）下，选择 JSON 或 Text（文本）。
3. 在目标类型下，选择CloudWatch 日志或 Kinesis Fire hose。
4. 在日志目标下，选择新建并输入您的 CloudWatchLogs 日志组名称或您的 Kinesis Data Firehose 直播名称。或者选择 “选择现有”，然后选择您的 CloudWatchLogs 日志组名称或 Kinesis Data Firehose 直播名称。

使用 AWS CLI

您可以使用 AWS CLI `modify-cache-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。

> **Important**
>
> 您可以升级到较新的引擎版本。升级主要引擎版本（例如从 5.0.6 升级到 6.0）时，您需要选择一个与新引擎版本兼容的参数组系列。有关执行此操作的更多信息，请参阅 引擎版本和升级。不过，您不能降级到较早的引擎版本，除非删除现有集群并重新创建它。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --preferred-maintenance-window sun:23:00-mon:02:00
```

对于 Windows：

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --preferred-maintenance-window sun:23:00-mon:02:00
```

--apply-immediately 参数仅适用于节点类型、引擎版本的修改，并更改 集群中的节点数。如果您希望立即应用任意这些更改，请使用 --apply-immediately 参数。如果您希望将这些更改推迟到下一维护时段，请使用 --no-apply-immediately 参数。诸如更改维护时段这样的其他修改是立即应用的。

有关更多信息，请参阅 f AWS CLI or ElastiCache 主题modify-cache-cluster。

使用 ElastiCache API

您可以使用 ElastiCache API ModifyCacheCluster 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。

> **Important**
>
> 您可以升级到较新的引擎版本。升级主要引擎版本（例如从 5.0.6 升级到 6.0）时，您需要选择一个与新引擎版本兼容的参数组系列。有关执行此操作的更多信息，请参阅 引擎版本和升级。不过，您不能降级到较早的引擎版本，除非删除现有集群并重新创建它。

添加换行符以便于阅读。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &CacheClusterId=my-cluster
    &PreferredMaintenanceWindow=sun:23:00-mon:02:00
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150901T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20150202T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20150901T220302Z
    &X-Amz-Credential=<credential>
```

```
    &X-Amz-Signature=<signature>
```

ApplyImmediately 参数仅适用于节点类型、引擎版本的修改，并更改 集群中的节点数。如果您希望立即应用任意这些更改，请将 ApplyImmediately 参数设置为 true。如果您希望将这些更改推迟到下一维护时段，请将 ApplyImmediately 参数设置为 false。诸如更改维护时段这样的其他修改是立即应用的。

有关更多信息，请参阅 ElastiCache API 参考主题ModifyCacheCluster。

向集群添加节点

要重新配置 Redis（已启用集群模式）集群，请参阅 扩展 Redis（启用集群模式）集群

您可以使用 ElastiCache 管理控制台、AWS CLI 或 ElastiCache API 向集群添加节点。

使用 AWS Management Console

将节点添加到单节点 Redis（已禁用集群模式）集群（未启用复制的集群）是包含两个步骤的流程：首先添加复制，然后添加副本节点。

主题

下面的流程将向未启用复制的单节点 Redis 添加复制。在添加复制时，现有的节点将成为启用复制的集群中的主节点。添加复制后，您可以向该集群添加最多 5 个副本节点。

向没有分片的 Redis 集群添加复制

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 从导航窗格中，选择 Redis clusters（Redis 集群）。

   此时会显示运行 Redis 引擎的集群列表。
3. 选择要将节点添加到的集群的名称，而不是集群名称左侧的框。

   未启用复制的 Redis 集群满足以下条件：

   - 它运行的是 Redis，而不是 Clustered Redis。
   - 它有零个分片。

   如果该集群有任何分片、已启用复制，则您可以在向集群添加节点（控制台）处继续。
4. 选择 Add replication。
5. 在添加复制中，为该启用复制的集群输入说明。
6. 选择 Add（添加）。

   一旦该集群的状态恢复为 available，您就可以在下一个流程中继续，并向该集群中添加副本。

向集群添加节点（控制台）

以下步骤可用于将节点添加到集群。

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在导航窗格中，选择在您要添加节点的集群上运行的引擎。

   此时会显示运行所选引擎的集群的列表。
3. 从集群列表中，对于要向其添加节点的集群，选择其名称。

   如果您的集群是 Redis（已启用集群模式）集群，请参阅 扩展 Redis（启用集群模式）集群。

   如果您的集群是具有零个分区的 Redis（已禁用集群模式）集群，请先完成 向没有分片的 Redis 集群添加复制 中的步骤。
4. 选择 Add node。
5. 在 Add Node (添加节点) 对话框中，填写请求的信息。
6. 选择 Apply Immediately - Yes (立即应用 - 是) 按钮立即添加此节点，或选择 No (否) 在集群的下一个维护时段添加此节点。

   新添加和删除请求对待处理请求的影响

场景	待处理的操作	新建请求	结果
方案 1	删除	删除	新的删除请求（待处理或立即）将替换待处理的删除请求。 例如，如果节点 0001、0003 和 0007 处于等待删除状态，同时发出了删除节点 0002 和 0004 的新请求，则只删除节点 0002 和 0004。节点 0001、0003 和 0007 不会被删除。
方案 2	删除	创建	新的创建请求（待处理或立即）将替换待处理的删除请求。 例如，如果节点 0001、0003 和 0007 处于等待删除状态，同时发出了创建节点的新请求，则会创建一个新节点，节点 0001、0003 和 0007 不会被删除。

场景	待处理的操作	新建请求	结果
方案三	创建	删除	新的删除请求（待处理或立即）将替换待处理的创建请求。 例如，如果存在创建两个节点的待处理请求，同时发出了删除节点 0003 的新请求，则不会创建新节点，节点 0003 会被删除。
方案 4	创建	创建	新创建请求将添加到待处理创建请求中。 例如，如果存在创建两个节点的待处理请求，同时发出了创建三个节点的新请求，则新请求将添加到待处理请求中，并将创建五个节点。 **Important** 如果新创建请求设置为 Apply Immediately - Yes（立即应用 – 是），则立即执行所有创建请求。如果新创建请求设置为 Apply Immediately - No（立即应用 – 否），则所有创建请求为待处理。

要确定哪些操作处于待处理状态，请选择 Description（描述）选项卡，然后查看显示了多少待处理的创建或删除操作。您不能同时拥有待处理的创建操作和待处理的删除操作。

7. 选择 Add 按钮。

   片刻之后，新的节点应会出现在节点列表中，其状态为 creating。若非如此，请刷新浏览器页面。当节点的状态更改为 available 时，便可以使用新节点。

使用 AWS CLI

如果要向未启用复制的现有 Redis（已禁用集群模式）集群添加节点，则必须先创建将该现有集群指定为主集群的复制组。有关更多信息，请参阅使用可用 Redis 缓存集群创建复制组 (AWS CLI)。待该复制组为 available（可用）后，您可以继续下面的流程。

要使用 AWS CLI 向集群添加节点，请使用带以下参数的 AWS CLI 操作 `increase-replica-count`：

- `--replication-group-id` 要添加节点的复制组的 ID。
- `--new-replica-count` 指定应用修改后此复制组中应有的节点的数量。要向此集群添加节点，`--new-replica-count` 必须大于此集群中的当前节点数。
- `--apply-immediately` 或 `--no-apply-immediately`，用于指定是立即添加这些节点还是在下一维护时段添加这些节点。

对于 Linux、macOS 或 Unix：

```
aws elasticache increase-replica-count \
    --replication-group-id my-replication-group \
    --new-replica-count 4 \
    --apply-immediately
```

对于 Windows：

```
aws elasticache increase-replica-count ^
    --replication-group-id my-replication-group ^
    --new-replica-count 4 ^
    --apply-immediately
```

此操作将生成类似于以下内容的输出（JSON 格式）：

```
{
    "ReplicationGroup": {
        "ReplicationGroupId": "node-test",
        "Description": "node-test",
        "Status": "modifying",
        "PendingModifiedValues": {},
        "MemberClusters": [
            "node-test-001",
            "node-test-002",
            "node-test-003",
            "node-test-004",
            "node-test-005"
        ],
        "NodeGroups": [
            {
```

```
                "NodeGroupId": "0001",
                "Status": "modifying",
                "PrimaryEndpoint": {
                    "Address": "node-test.zzzzzz.ng.0001.usw2.cache.amazonaws.com",
                    "Port": 6379
                },
                "ReaderEndpoint": {
                    "Address": "node-test.zzzzzz.ng.0001.usw2.cache.amazonaws.com",
                    "Port": 6379
                },
                "NodeGroupMembers": [
                    {
                        "CacheClusterId": "node-test-001",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-001.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2a",
                        "CurrentRole": "primary"
                    },
                    {
                        "CacheClusterId": "node-test-002",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-002.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2c",
                        "CurrentRole": "replica"
                    },
                    {
                        "CacheClusterId": "node-test-003",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-003.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CurrentRole": "replica"
                    }
```

```
                ]
            }
        ],
        "SnapshottingClusterId": "node-test-002",
        "AutomaticFailover": "enabled",
        "MultiAZ": "enabled",
        "SnapshotRetentionLimit": 1,
        "SnapshotWindow": "07:30-08:30",
        "ClusterEnabled": false,
        "CacheNodeType": "cache.r5.large",
         "DataTiering": "disabled",
        "TransitEncryptionEnabled": false,
        "AtRestEncryptionEnabled": false,
        "ARN": "arn:aws:elasticache:us-west-2:123456789012:replicationgroup:node-test"
    }
}
```

有关更多信息，请参阅 AWS CLI 主题 increase-replica-count。

使用 ElastiCache API

如果要向未启用复制的现有 Redis（已禁用集群模式）集群添加节点，则必须先创建将该现有集群指定为主集群的复制组。有关更多信息，请参阅向独立 Redis（已禁用集群模式）集群添加副本 (ElastiCache API)。待该复制组为 available（可用）后，您可以继续下面的流程。

向集群添加节点 (ElastiCache API)

- 按照以下参数调用 IncreaseReplicaCount API 操作：
  - ReplicationGroupId 要将节点添加到的集群的 ID。
  - NewReplicaCount NewReplicaCount 参数指定应用修改后此集群中应有的节点的数量。要向此集群添加节点，NewReplicaCount 必须大于此集群中的当前节点数。如果此值小于当前节点数，请使用 DecreaseReplicaCount API 以及要从集群中移除的节点数量。
  - ApplyImmediately 指定是立即添加这些节点还是在下一维护时段添加这些节点。
  - Region 指定要添加节点的集群的 AWS 区域。

  以下示例演示向集群添加节点的调用。

  Example

```
https://elasticache.us-west-2.amazonaws.com/
```

```
    ?Action=IncreaseReplicaCount
    &ApplyImmediately=true
    &NumCacheNodes=4
    &ReplicationGroupId=my-replication-group
    &Region=us-east-2
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

有关更多信息，请参阅 ElastiCache API 主题 IncreaseReplicaCount。

从集群中移除节点

您可以使用 AWS Management Console、AWS CLI 或 ElastiCache API 从集群中删除节点。

使用 AWS Management Console

从集群中移除节点（控制台）

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 从右上角的列表中，选择要从中删除节点的集群所在的 AWS 区域。
3. 在导航窗格中，选择在您要删除节点的集群上运行的引擎。

   此时会显示运行所选引擎的集群的列表。
4. 从集群列表中，选择要从中删除节点的集群的名称。

   此时会显示集群节点的列表。
5. 选择要删除的节点的节点 ID 左侧的复选框。使用 ElastiCache 控制台时，一次只能删除一个节点，因此选择多个节点表明您无法使用 Delete node（删除节点）按钮。

   此时将显示删除节点页面。
6. 要删除节点，请完成删除节点页面，然后选择删除节点。要保留节点，请选择取消。

> **Important**
>
> 如果删除节点导致集群不再符合多可用区标准，请确保首先清除多可用区复选框，然后删除该节点。如果清除多可用区复选框，则可以选择启用自动故障转移。

新添加和删除请求对待处理请求的影响

场景	待处理的操作	新建请求	结果
方案 1	删除	删除	新的删除请求（待处理或立即）将替换待处理的删除请求。 例如，如果节点 0001、0003 和 0007 处于等待删除状态，同时发出了删除节点 0002 和 0004 的新请求，则

场景	待处理的操作	新建请求	结果
			只删除节点 0002 和 0004。节点 0001、0003 和 0007 不会被删除。
方案 2	删除	创建	新的创建请求（待处理或立即）将替换待处理的删除请求。 例如，如果节点 0001、0003 和 0007 处于等待删除状态，同时发出了创建节点的新请求，则会创建一个新节点，节点 0001、0003 和 0007 不会被删除。
方案三	创建	删除	新的删除请求（待处理或立即）将替换待处理的创建请求。 例如，如果存在创建两个节点的待处理请求，同时发出了删除节点 0003 的新请求，则不会创建新节点，节点 0003 会被删除。
方案 4	创建	创建	新创建请求将添加到待处理创建请求中。 例如，如果存在创建两个节点的待处理请求，同时发出了创建三个节点的新请求，则新请求将添加到待处理请求中，并将创建五个节点。 **Important** 如果新创建请求设置为 Apply Immediately - Yes（立即应用 – 是），则立即执行所有创建请求。如果新创建请求设置为 Apply Immediately - No（立即应用 – 否），则所有创建请求为待处理。

要确定哪些操作处于待处理状态，请选择 Description（描述）选项卡，然后查看显示了多少待处理的创建或删除操作。您不能同时拥有待处理的创建操作和待处理的删除操作。

使用 AWS CLI

1. 确定要删除的节点的 ID。有关更多信息，请参阅查看集群的详细信息。
2. 将 `decrease-replica-count` 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 `decrease-replica-count`：

   - `--replication-group-id` 要从其中删除节点的复制组的 ID。
   - `--new-replica-count --new-replica-count` 参数指定应用修改后此集群中应有的节点的数量。
   - `--replicas-to-remove` 要从此集群中删除的节点 ID 的列表。
   - `--apply-immediately` 或 `--no-apply-immediately` 指定是立即移除这些节点还是在下一维护时段移除这些节点。
   - `--region` 指定要从其中删除节点的集群的 AWS 区域。

   > **Note**
   >
   > 您只能在调用此操作时传递 `--replicas-to-remove` 或者 `--new-replica-count` 参数。

   对于 Linux、macOS 或 Unix：

   ```
   aws elasticache decrease-replica-count \
       --replication-group-id my-replication-group \
       --new-replica-count 2 \
       --region us-east-2 \
       --apply-immediately
   ```

   对于 Windows：

   ```
   aws elasticache decrease-replica-count ^
       --replication-group-id my-replication-group ^
       --new-replica-count 3 ^
       --region us-east-2 ^
       --apply-immediately
   ```

此操作将生成类似于以下内容的输出（JSON 格式）：

```
{
    "ReplicationGroup": {
        "ReplicationGroupId": "node-test",
        "Description": "node-test"
       },
        "Status": "modifying",
        "PendingModifiedValues": {},
        "MemberClusters": [
            "node-test-001",
            "node-test-002",
            "node-test-003",
            "node-test-004",
            "node-test-005",
            "node-test-006"
        ],
        "NodeGroups": [
            {
                "NodeGroupId": "0001",
                "Status": "modifying",
                "PrimaryEndpoint": {
                    "Address": "node-test.zzzzzz.ng.0001.usw2.cache.amazonaws.com",
                    "Port": 6379
                },
                "ReaderEndpoint": {
                    "Address": "node-test-
ro.zzzzzz.ng.0001.usw2.cache.amazonaws.com",
                    "Port": 6379
                },
                "NodeGroupMembers": [
                    {
                        "CacheClusterId": "node-test-001",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-001.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2a",
                        "CurrentRole": "primary"
                    },
                    {
```

```
                        "CacheClusterId": "node-test-002",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-002.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2c",
                        "CurrentRole": "replica"
                    },
                    {
                        "CacheClusterId": "node-test-003",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-003.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CurrentRole": "replica"
                    },
                    {
                        "CacheClusterId": "node-test-004",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-004.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2c",
                        "CurrentRole": "replica"
                    },
                    {
                        "CacheClusterId": "node-test-005",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-005.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CurrentRole": "replica"
                    },
                    {
```

```
                        "CacheClusterId": "node-test-006",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Address": "node-
test-006.zzzzzz.0001.usw2.cache.amazonaws.com",
                            "Port": 6379
                        },
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CurrentRole": "replica"
                    }
                ]
            }
        ],
        "SnapshottingClusterId": "node-test-002",
        "AutomaticFailover": "enabled",
        "MultiAZ": "enabled",
        "SnapshotRetentionLimit": 1,
        "SnapshotWindow": "07:30-08:30",
        "ClusterEnabled": false,
        "CacheNodeType": "cache.r5.large",
         "DataTiering": "disabled",
        "TransitEncryptionEnabled": false,
        "AtRestEncryptionEnabled": false,
        "ARN": "arn:aws:elasticache:us-west-2:123456789012:replicationgroup:node-
test"
    }
}
```

或者，您也可以调用 `decrease-replica-count`，而不必传递 `--new-replica-count` 参数，您可以传递 `--replicas-to-remove` 参数，如下所示：

对于 Linux、macOS 或 Unix：

```
aws elasticache decrease-replica-count \
    --replication-group-id my-replication-group \
    --replicas-to-remove node-test-003 \
    --region us-east-2 \
    --apply-immediately
```

对于 Windows：

```
aws elasticache decrease-replica-count ^
```

```
    --replication-group-id my-replication-group ^
    --replicas-to-remove node-test-003 ^
    --region us-east-2 ^
    --apply-immediately
```

有关更多信息，请参阅 AWS CLI 主题 decrease-replica-count。

使用 ElastiCache API

要使用 ElastiCache API 删除节点，请使用复制组 ID 和要删除节点的列表调用 DecreaseReplicaCount API 操作，如下所示：

- ReplicationGroupId 要从其中删除节点的复制组的 ID。
- ReplicasToRemove ReplicasToRemove 参数指定应用修改后此集群中应有的节点的数量。
- ApplyImmediately 指定是立即移除这些节点还是在下一维护时段移除这些节点。
- Region 指定要从其中删除节点的集群的 AWS 区域。

以下示例从集群 my-cluster 中立即删除节点 0004 和 0005。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DecreaseReplicaCount
    &ReplicationGroupId=my-replication-group
    &ApplyImmediately=true
    &ReplicasToRemove=node-test-003
    &Region us-east-2
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

有关更多信息，请参阅 ElastiCache API 主题 DecreaseReplicaCount。

取消待处理的添加或删除节点操作

如果您选择不立即应用更改，则操作将一直保持 pending (等待) 状态，直到在您的下一个维护时段执行。您可以取消任意等待操作。

取消等待操作

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 从右上角的列表中，选择您要取消其中待处理的添加或删除节点操作的 AWS 区域。
3. 在导航窗格中，选择在包含要取消的待处理操作的集群上运行的引擎。此时会显示运行所选引擎的集群的列表。
4. 在集群列表中，选择具有要取消等待的操作的集群名称，而不是集群名称左侧的框。
5. 要确定哪些操作处于待处理状态，请选择 Description（描述）选项卡，然后查看显示了多少待处理的创建或删除操作。您不能同时拥有待处理的创建操作和待处理的删除操作。
6. 选择 Nodes（节点）选项卡。
7. 要取消所有待处理的操作，请单击 Cancel Pending。此时会显示 Cancel Pending (取消等待) 对话框。
8. 选择 Cancel Pending 按钮确认取消所有等待操作，或选择 Cancel 保留这些操作。

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 ElastiCache for Redis 集群时，您的手动快照会保留。您也可以在删除集群之前创建最终快照。自动缓存快照不会保留。

使用 AWS 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 AWS Management Console 并打开 Amazon ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在 ElastiCache 控制台控制面板中，选择您要删除的集群当前所运行的引擎。

   此时会显示运行该引擎的所有集群的列表。
3. 要选择要删除的集群，请从集群列表中选择该集群的名称。

   > **Important**
   >
   > 从 ElastiCache 控制台一次只能删除一个集群。选择多个集群会禁用删除操作。
4. 对于 Actions (操作)，选择 Delete (删除)。
5. 在删除集群确认屏幕上，选择删除可删除集群，选择取消可保留集群。

   如果选择了 Delete，集群的状态将变为正在删除。

只要您的集群不再在集群列表中列出，您就无需为该集群付费。

使用 AWS CLI

下面的代码删除缓存集群 `my-cluster`。

```
aws elasticache delete-cache-cluster --cache-cluster-id my-cluster
```

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

对于 Linux、macOS 或 Unix：

```
aws elasticache delete-cache-cluster \
    --cache-cluster-id my-cluster \
    --region us-east-2
```

对于 Windows：

```
aws elasticache delete-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --region us-east-2
```

有关更多信息，请参阅 AWS CLI for ElastiCache 主题 delete-cache-cluster。

使用 ElastiCache API

以下代码删除集群 my-cluster。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DeleteCacheCluster
    &CacheClusterId=my-cluster
    &Region us-east-2
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20150202T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20150202T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

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

有关更多信息，请参阅 ElastiCache API 参考主题 DeleteCacheCluster。

访问您的集群或复制组

您的 Amazon ElastiCache 实例旨在通过 Amazon EC2 实例进行访问。

如果您在 Amazon Virtual Private Cloud (Amazon VPC) 中启动了您的 ElastiCache 实例，您可以从同一 Amazon VPC 中的 Amazon EC2 实例访问您的 ElastiCache 实例。或者，通过使用 VPC 对等连接，您可以从不同 Amazon VPC 中的 Amazon EC2 访问您的 ElastiCache 实例。

如果您已在 EC2 Classic 中启动 ElastiCache 实例，则可以通过向与该实例关联的 Amazon EC2 安全组授予对您的缓存安全组的访问权限来允许 EC2 实例访问您的集群。默认情况下，仅启动了集群的账户能够访问集群。

主题

授予访问您的集群或复制组的权限

您将集群启动到 EC2-VPC 中

如果您将集群启动到 Amazon Virtual Private Cloud (Amazon VPC) 中，则只能从在同一 Amazon VPC 中运行的 Amazon EC2 实例连接到您的 ElastiCache 集群。在此情况下，您需要向集群授予网络进入。

> **Note**
>
> 如果您正在使用 Local Zones，请确保已启用它。有关更多信息，请参阅启用 Local Zones。通过这样做，您的 VPC 将扩展到该 Local Zone，您的 VPC 会将子网视为任何其他可用区中的任何子网，并且相关网关、路由表和其他安全组注意事项将自动调整。

授予从 Amazon VPC 安全组到集群的网络入口

1. 登录到 AWS Management Console 并打开 Amazon EC2 控制台（https://console.aws.amazon.com/ec2/）。
2. 在导航窗格中的 Network & Security 下，选择 Security Groups。
3. 从安全组列表中，为 Amazon VPC 选择安全组。除非创建安全组供 ElastiCache 使用，否则此安全组将命名为 default。
4. 选择 Inbound 选项卡，然后执行以下操作：

a. 选择 Edit（编辑）。

b. 选择添加规则。

c. 在 Type 列中，选择 Custom TCP rule。

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

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

> **Important**
>
> 向 0.0.0.0/0 公开 ElastiCache 集群时，不会在互联网上公开集群，因为它没有公有 IP 地址，因此无法从 VPC 外部访问。但是，默认安全组可以应用到客户账户中的其他 Amazon EC2 实例，这些实例可能具有公有 IP 地址。如果这些实例碰巧在默认端口上运行某些内容，则该服务可能会意外暴露。因此，我们建议创建将由 ElastiCache 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）。

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

从 AWS 外部访问 ElastiCache 资源

Amazon ElastiCache 是提供云端内存中键值存储的 AWS 服务。该服务设计为只能从 AWS 中访问。但是，如果 ElastiCache 集群托管在 VPC 中，您可以使用网络地址转换（NAT）实例来提供外部访问。

要求

您必须满足以下要求才能从 AWS 外部访问您的 ElastiCache 资源：

- 集群必须驻留在 VPC 中，并且可以通过网络地址转换 (NAT) 实例访问。此要求不存在例外情况。
- NAT 实例必须在与集群相同的 VPC 中启动。
- NAT 实例必须在与集群分开的公有子网中启动。
- 弹性 IP 地址 (EIP) 必须与 NAT 实例关联。iptables 的端口转发功能用于将 NAT 实例上的端口转发到 VPC 中的缓存节点端口。

注意事项

从 ElastiCache 外部访问您的 ElastiCache 资源时，必须牢记以下注意事项。

- 客户端连接到 NAT 实例的 EIP 和缓存端口。NAT 实例上的端口转发功能会将流量转发到相应的缓存群集端口。
- 如果添加或替换集群节点，则需要更新 iptables 规则以反映此更改。

限制

此方法应仅应用于测试和开发用途。由于以下限制，不建议将其用于生产用途：

- NAT 实例用作客户端与多个集群之间的代理。添加代理会影响到缓存群集的性能。这种影响会随着您通过 NAT 实例访问的缓存群集数量增加而增大。
- 从客户端到 NAT 实例的流量未加密。因此，您应当避免通过 NAT 实例发送敏感数据。
- NAT 实例增加了维护另一实例的开销。
- NAT 实例会成为单点故障。有关如何在 VPC 上设置高可用性 NAT 的信息，请参阅 Amazon VPC NAT 实例的高可用性：示例。

从 AWS 外部访问 ElastiCache 资源的方法

以下过程演示了如何使用 NAT 实例连接到您的 ElastiCache 资源。

这些步骤假定以下各项：

- `iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6380 -j DNAT --to 10.0.1.231:6379`
- `iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6381 -j DNAT --to 10.0.1.232:6379`

接下来，您需要相反方向的 NAT：

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 10.0.0.55
```

您还需要启用 IP 转发，该功能默认情况下处于禁用状态：

```
sudo sed -i 's/net.ipv4.ip_forward=0/net.ipv4.ip_forward=1/g' /etc/
sysctl.conf sudo sysctl --system
```

- 您可以使用以下项访问 Redis 集群：
  - IP 地址 – 10.0.1.230
  - 默认 Redis 端口 – 6379
  - 安全组 – sg-bd56b7da
  - AWS 实例 IP 地址：sg-bd56b7da
- 您的可信客户端使用 IP 地址 198.51.100.27。
- 您的 NAT 实例具有弹性 IP 地址 203.0.113.73。
- 您的 NAT 实例具有安全组 sg-ce56b7a9。

使用 NAT 实例连接您的 ElastiCache 资源

1. 在与缓存群集相同的 VPC 中创建 NAT 实例，但位于公有子网上。

   默认情况下，VPC 向导将启动 cache.m1.small 节点类型。您应该根据需求选择节点大小。您必须使用 EC2 NAT AMI 才能从 AWS 外部访问 ElastiCache。

   有关创建 NAT 实例的信息，请参阅《AWS VPC 用户指南》中的 NAT 实例。
2. 为缓存群集和 NAT 实例创建安全组规则。

   NAT 实例安全组和集群实例应具有以下规则：

- 两个入站规则
  - 一个用于允许从可信客户端到每个缓存端口的 TCP 连接，这些缓存端口是从 NAT 实例 (6379 - 6381) 转发的。
  - 第二个用于允许通过 SSH 访问可信客户端。

  NAT 实例安全组 – 入站规则

类型	协议	端口范围	来源
自定义 TCP 规则	TCP	6379-6380	198.51.100.27/32
SSH	TCP	22	203.0.113.73/32

- 一个允许与缓存端口 (6379) 建立 TCP 连接的出站规则。

  NAT 实例安全组 – 出站规则

类型	协议	端口范围	目标位置
自定义 TCP 规则	TCP	6379	sg-ce56b7a9（集群实例安全组）

- 集群安全组的入站规则，允许从 NAT 实例到缓存端口 (6379) 的 TCP 连接。

  集群实例安全组 – 入站规则

类型	协议	端口范围	来源
自定义 TCP 规则	TCP	6379	sg-bd56b7da（集群安全组）

3. 验证规则。
   - 确认可信客户端可以通过 SSH 连接到 NAT 实例。
   - 确认可信客户端能够从 NAT 实例连接到集群。
4. 将 iptables 规则添加到 NAT 实例。

   必须为集群中的每个节点将 iptables 规则添加到 NAT 表，以便将缓存端口从 NAT 实例转发到集群节点。示例如下所示：

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6379 -j DNAT --to
 10.0.1.230:6379
```

集群中每个节点的端口号必须唯一。例如，如果是三个节点的 Redis 集群，使用端口 6379 - 6381，则规则将如下所示：

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6379 -j DNAT --to
 10.0.1.230:6379
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6380 -j DNAT --to
 10.0.1.231:6379
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 6381 -j DNAT --to
 10.0.1.232:6379
```

5. 确认可信客户端能够连接到集群。

   可信客户端应连接到与 NAT 实例关联的 EIP 以及对应于相应集群节点的集群端口。例如，PHP 的连接字符串如下所示：

   ```
   redis->connect( '203.0.113.73', 6379 );
   redis->connect( '203.0.113.73', 6380 );
   redis->connect( '203.0.113.73', 6381 );
   ```

   也可以使用 telnet 客户端来验证连接。例如：

   ```
   telnet 203.0.113.73 6379
   telnet 203.0.113.73 6380
   telnet 203.0.113.73 6381
   ```

6. 保存 iptables 配置。

   在您测试和验证规则之后保存规则。如果您使用基于 Redhat 的 Linux 分发（例如 Amazon Linux），请运行以下命令：

   ```
   service iptables save
   ```

相关 主题

下列主题可能会有用处。

- 访问 Amazon VPC 中 ElastiCache 缓存的访问模式
- 从客户数据中心运行的应用程序访问 ElastiCache 缓存
- NAT 实例
- 配置 ElastiCache 客户端
- Amazon VPC NAT 实例的高可用性：示例

查找连接端点

您的应用程序使用端点连接到集群。端点是节点或集群的唯一的地址。

如果您不使用 Automatic Discovery，则必须对客户端进行配置，以便针对读取和写入使用不同的节点端点。此外，在添加或删除节点时，您还必须跟踪它们的情况。

该使用哪些端点

- Redis 独立节点：使用该节点的端点进行读取和写入操作。

- Redis（已禁用集群模式）集群，使用主端点执行所有写入操作。使用读取器端点 将在所有只读副本之间均匀地分配指向端点的传入连接。使用单独的节点端点 进行读取操作（在 API/CLI 中，它们被称作读取端点）。

- Redis（已启用集群模式）集群，使用集群的配置端点执行所有支持已启用集群模式命令的操作。您必须使用支持 Redis 集群的客户端（Redis 3.2）。您仍可以从独立的节点端点进行读取（在 API/CLI 中，它们被称作读取端点）。

以下部分将引导您发现正在运行的引擎所需的端点。

查找 Redis（已禁用集群模式）集群的端点（控制台）

如果 Redis（已禁用集群模式）集群只有一个节点，则使用该节点的端点进行读取和写入操作。如果 Redis（已禁用集群模式）集群具有多个节点，则有三种类型的端点，即主端点、读取器端点和节点端点。

主端点是一个 DNS 名称，始终解析为集群中的主节点。主端点不受集群更改的影响，如将只读副本提升为主角色。对于写入活动，我们建议您的应用程序连接到主端点。

读取器端点将在 ElastiCache for Redis 集群中的所有只读副本之间均匀地分配指向端点的传入连接。应用程序何时创建连接或应用程序如何（重复）使用连接等附加因素将决定流量分配。读取器端点会在添加或删除副本时实时跟踪集群更改。您可以将 ElastiCache for Redis 集群的多个只读副本置于不同的 AWS 可用区（AZ）中以确保读取器端点的高可用性。

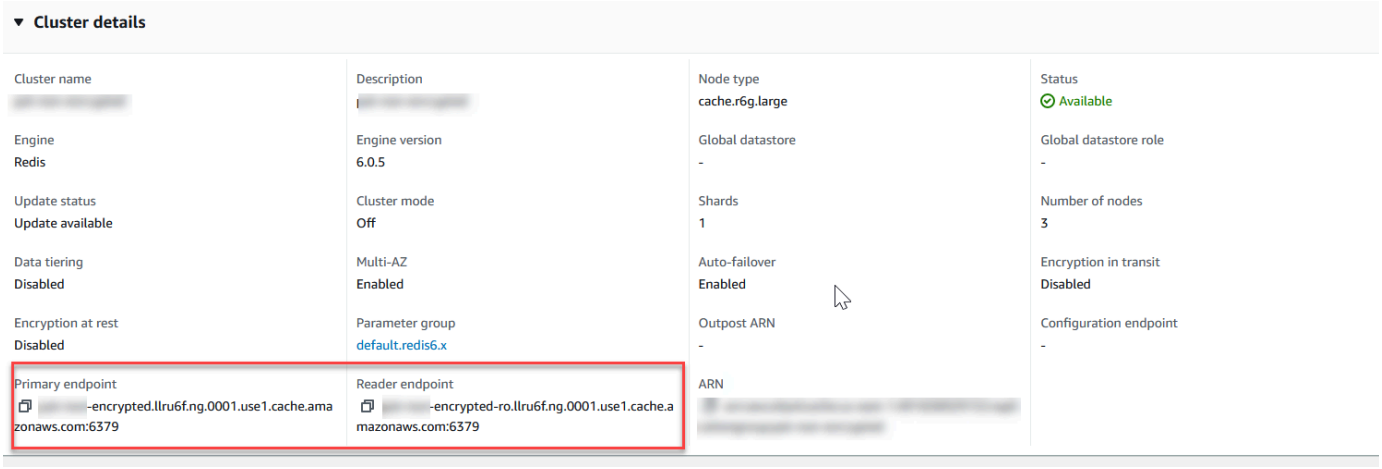
> **Note**
>
> 读取器端点不是负载均衡器。它是一个 DNS 记录，将以循环方式解析为副本节点之一的 IP 地址。

对于读取活动，应用程序还可以连接到集群中的任何节点。与主端点不同，节点端点会解析为特定端点。如果您在您的集群中进行更改（例如添加或删除副本），则必须在您的应用程序中更新节点端点。

查找 Redis（已禁用集群模式）集群的端点

1. 登录 AWS Management Console 并打开 ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 从导航窗格中，选择 Redis clusters（Redis 集群）。

   集群屏幕将显示 Redis（已禁用集群模式）和 Redis（已启用集群模式）集群的列表。
3. 要查找集群的主端点和/或读取器端点，请选中集群的名称（而不是其左侧的按钮）。

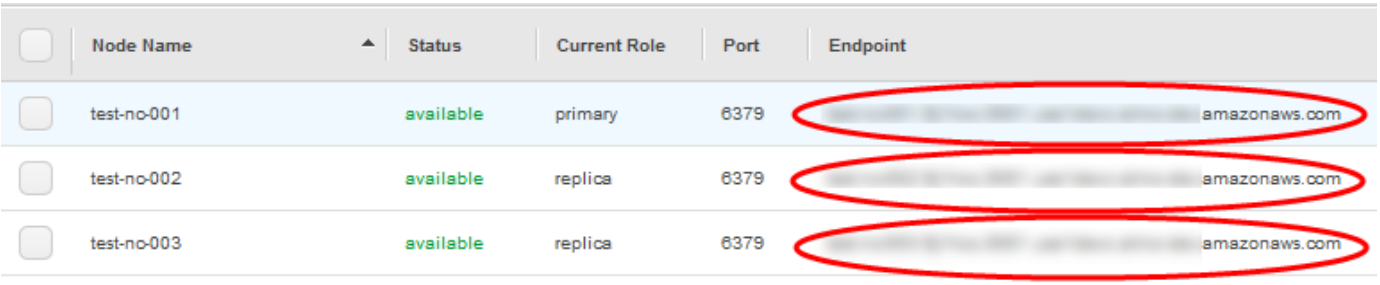


Redis（已禁用集群模式）集群的主端点和读取器端点

如果该集群只有一个节点，则没有主端点，您可以继续下一步。

4. 如果 Redis（已禁用集群模式）集群有副本节点，您可以通过选择此集群的名称、然后选择 Nodes（节点）选项卡来找到集群副本的节点端点。

   此时会显示节点屏幕，其中列出了集群中的每个节点（主节点和副本节点）及其端点。



   Redis（已禁用集群模式）集群的节点端点

5. 将端点复制到剪贴板：

   a. 逐一找到要复制的端点。

   b. 直接选择端点前面的复制图标。

   端点现已复制到剪贴板。有关使用端点连接到节点的信息，请参阅 连接到节点。

Redis（已禁用集群模式）主端点类似以下内容。根据是否已启用传输中加密而有所不同。

未启用传输中加密

```
clusterName.xxxxxx.nodeId.regionAndAz.cache.amazonaws.com:port

redis-01.7abc2d.0001.usw2.cache.amazonaws.com:6379
```

已启用传输中加密

```
master.clusterName.xxxxxx.regionAndAz.cache.amazonaws.com:port

master.ncit.ameaqx.use1.cache.amazonaws.com:6379
```

查找 Redis（已启用集群模式）集群的端点（控制台）

Redis（已启用集群模式）集群有一个单配置端点。通过连接到配置端点，您的应用程序可以查找集群中每个分片的主端点和读取端点。

查找 Redis（已启用集群模式）集群的端点

1. 登录 AWS Management Console 并打开 ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 从导航窗格中，选择 Redis clusters（Redis 集群）。

   集群屏幕将显示 Redis（已禁用集群模式）和 Redis（已启用集群模式）集群的列表。选择要连接的 Redis（已启用集群模式）集群。
3. 要查找集群的配置端点，请选择集群的名称（不是单选按钮）。
4. Cluster details（集群详细信息）下将显示 Configuration endpoint（配置端点）。要复制它，请选择位于端点左侧的 copy（复制）图标。

查找端点（AWS CLI）

您可以使用 AWS CLI for Amazon ElastiCache 来搜索节点、集群和复制组的端点。

主题

查找节点和集群的端点（AWS CLI）

您可以使用 AWS CLI，通过 `describe-cache-clusters` 命令查找集群及其节点的端点。对于 Redis 集群，此命令将返回集群端点。如果包含可选参数 `--show-cache-node-info`，则此命令还将返回集群中的单个节点的端点。

Example

以下命令会检索单节点 Redis（已禁用集群模式）集群 mycluster 的集群信息。

> **Important**
>
> 参数 `--cache-cluster-id` 可与 Redis 复制组中的单节点 Redis（已禁用集群模式）集群 ID 或特定节点 ID 配合使用。Redis 复制组的 `--cache-cluster-id` 是一个四位值，例如 `0001`。如果 `--cache-cluster-id` 是 Redis 复制组中集群（节点）的 ID，输出中会包含 `replication-group-id`。

对于 Linux、macOS 或 Unix：

```
aws elasticache describe-cache-clusters \
    --cache-cluster-id redis-cluster \
    --show-cache-node-info
```

对于 Windows：

```
aws elasticache describe-cache-clusters ^
    --cache-cluster-id redis-cluster ^
    --show-cache-node-info
```

上面的操作输出类似以下的内容（JSON 格式）。

```
{
    "CacheClusters": [
        {
            "CacheClusterStatus": "available",
            "SecurityGroups": [
                {
                    "SecurityGroupId": "sg-77186e0d",
                    "Status": "active"
                }
            ],
            "CacheNodes": [
                {
                    "CustomerAvailabilityZone": "us-east-1b",
                    "CacheNodeCreateTime": "2018-04-25T18:19:28.241Z",
                    "CacheNodeStatus": "available",
                    "CacheNodeId": "0001",
                    "Endpoint": {
                        "Address": "redis-cluster.amazonaws.com",
                        "Port": 6379
                    },
                    "ParameterGroupStatus": "in-sync"
                }
            ],
            "AtRestEncryptionEnabled": false,
            "CacheClusterId": "redis-cluster",
            "TransitEncryptionEnabled": false,
            "CacheParameterGroup": {
                "ParameterApplyStatus": "in-sync",
                "CacheNodeIdsToReboot": [],
                "CacheParameterGroupName": "default.redis3.2"
            },
            "NumCacheNodes": 1,
            "PreferredAvailabilityZone": "us-east-1b",
            "AutoMinorVersionUpgrade": true,
            "Engine": "redis",
            "AuthTokenEnabled": false,
            "PendingModifiedValues": {},
            "PreferredMaintenanceWindow": "tue:08:30-tue:09:30",
            "CacheSecurityGroups": [],
            "CacheSubnetGroupName": "default",
            "CacheNodeType": "cache.t2.small",
             "DataTiering": "disabled"
            "EngineVersion": "3.2.10",
```

```
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "CacheClusterCreateTime": "2018-04-25T18:19:28.241Z"
        }
    ]
}
```

有关更多信息，请参阅主题 describe-cache-clusters。

查找复制组的端点（AWS CLI）

您可以使用 AWS CLI，通过 `describe-replication-groups` 命令查找复制组及其集群的端点。此命令将返回复制组的主端点、复制组中所有集群（节点）及其端点的列表以及读取器端点。

以下操作检索复制组 `myreplgroup` 的主端点和读取器端点。将主端点用于所有写入操作。

```
aws elasticache describe-replication-groups \
    --replication-group-id myreplgroup
```

对于 Windows：

```
aws elasticache describe-replication-groups ^
    --replication-group-id myreplgroup
```

该操作输出类似以下的内容（JSON 格式）。

```
{
   "ReplicationGroups": [
     {
       "Status": "available",
       "Description": "test",
       "NodeGroups": [
         {
            "Status": "available",
               "NodeGroupMembers": [
                  {
                     "CurrentRole": "primary",
                     "PreferredAvailabilityZone": "us-west-2a",
                     "CacheNodeId": "0001",
                     "ReadEndpoint": {
                        "Port": 6379,
                        "Address": "myreplgroup-001.amazonaws.com"
                     },
```

```
                    "CacheClusterId": "myreplgroup-001"
                },
                {
                    "CurrentRole": "replica",
                    "PreferredAvailabilityZone": "us-west-2b",
                    "CacheNodeId": "0001",
                    "ReadEndpoint": {
                        "Port": 6379,
                        "Address": "myreplgroup-002.amazonaws.com"
                    },
                    "CacheClusterId": "myreplgroup-002"
                },
                {
                    "CurrentRole": "replica",
                    "PreferredAvailabilityZone": "us-west-2c",
                    "CacheNodeId": "0001",
                    "ReadEndpoint": {
                        "Port": 6379,
                        "Address": "myreplgroup-003.amazonaws.com"
                    },
                    "CacheClusterId": "myreplgroup-003"
                }
            ],
            "NodeGroupId": "0001",
            "PrimaryEndpoint": {
                "Port": 6379,
                "Address": "myreplgroup.amazonaws.com"
            },
            "ReaderEndpoint": {
                "Port": 6379,
                "Address": "myreplgroup-ro.amazonaws.com"
            }
        }
    ],
    "ReplicationGroupId": "myreplgroup",
    "AutomaticFailover": "enabled",
    "SnapshottingClusterId": "myreplgroup-002",
    "MemberClusters": [
        "myreplgroup-001",
        "myreplgroup-002",
        "myreplgroup-003"
    ],
    "PendingModifiedValues": {}
}
```

```
    ]
}
```

有关更多信息，请参阅 AWS CLI 命令参考中的 describe-replication-groups。

查找端点（ElastiCache API）

您可以使用 Amazon ElastiCache API 来搜索节点、集群和复制组的端点。

主题

- 查找节点和集群的端点（ElastiCache API）
- 查找复制组的端点（ElastiCache API）

查找节点和集群的端点（ElastiCache API）

您可以使用 ElastiCache API，通过 `DescribeCacheClusters` 操作查找集群及其节点的端点。对于 Redis 集群，此命令将返回集群端点。如果包含可选参数 `ShowCacheNodeInfo`，则此操作还将返回集群中的各个节点的端点。

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterId=mycluster
    &ShowCacheNodeInfo=true
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &Version=2015-02-02
    &X-Amz-Credential=<credential>
```

查找复制组的端点（ElastiCache API）

您可以使用 ElastiCache API，以通过 `DescribeReplicationGroups` 操作查找复制组及其集群的端点。此操作将返回复制组的主端点、复制组中所有集群及其端点的列表以及读取器端点。

以下操作检索复制组 `myreplgroup` 的主端点（PrimaryEndpoint）、读取器端点（ReaderEndpoint）和各个节点端点（ReadEndpoint）。将主端点用于所有写入操作。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReplicationGroups
    &ReplicationGroupId=myreplgroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
```

```
    &Version=2015-02-02
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 DescribeReplicationGroups。

使用分片

分区（API/CLI：节点组）是 1 到 6 个 Redis 节点的集合。Redis（已禁用集群模式）集群永远不会有多个分区。使用分片，您可以将大型数据库分成更小、更快、更易于管理的部分，称为数据分片。这可以通过将操作分配到多个单独的部分来提高数据库效率。使用分片可以带来许多好处，包括提高性能、可扩展性和成本效益。

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 90 个节点。此集群配置的范围可以从 90 个分片和 0 个副本到 15 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。

如果 Redis 引擎版本为 5.0.6 或更高版本，可将每个集群的节点或分片限制提高到最大值 500。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。有关更多信息，请参阅创建子网组。

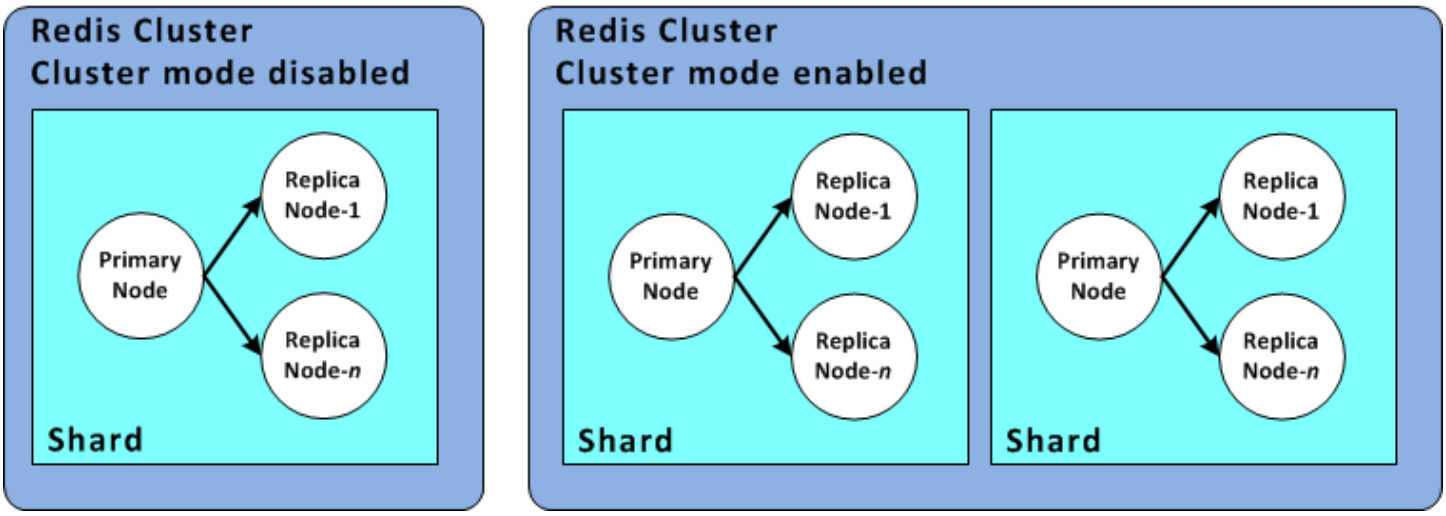
对于低于 5.0.6 的版本，每个集群的限制为 250。

若要请求提高限制，请参阅 AWS Service Limits 并选择限制类型 Nodes per cluster per instance type（每个实例类型的每个集群的节点数）。

使用 ElastiCache 控制台创建 Redis（已启用集群模式）集群时，需要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 Redis（已启用集群模式）集群（控制台）。如果您使用 ElastiCache API 或 AWS CLI 创建集群（在 API/CLI 中称为复制组），则可以独立配置分片（API/CLI：节点组）中的节点数量。有关更多信息，请参阅下列内容：

- API: CreateReplicationGroup
- CLI：create-replication-group

分片中每个节点的计算、存储和内存规格均相同。该 ElastiCache API 允许您控制分片范围的属性，例如节点数量、安全设置和系统维护时段。



Redis 分片配置

有关更多信息，请参阅 Redis（已启用集群模式）的离线重新分区和分区重新平衡 和 Redis（已启用集群模式）的在线重新分片和分片重新平衡。

查找分区的 ID

您可以使用 AWS Management Console、 AWS CLI 或 ElastiCache API 查找分片的 ID。

使用 AWS Management Console

主题

- 对于 Redis（已禁用集群模式）
- 对于 Redis（已启用集群模式）

对于 Redis（已禁用集群模式）

Redis（已禁用集群模式）复制组分区 ID 始终为 `0001`。

对于 Redis（已启用集群模式）

以下过程使用查找 Redis（已启用集群模式）的复制组的分片 ID。 AWS Management Console

在 Redis（已启用集群模式）复制组中查找分区 ID

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。

2. 在导航窗格上，选择 Redis，然后选择要为其查找分区 ID 的 Redis（已启用集群模式）复制组的名称。
3. 在 Shard Name（分区名称）列中，分区 ID 是分区名称的最后四位数字。

使用 AWS CLI

要查找 Redis（已禁用集群模式）或 Redis（已启用集群模式）复制组的分片（节点组）ID，请使用`describe-replication-groups`带有以下可 AWS CLI 选参数的操作。

- **`--replication-group-id`** – 用来将输出限制为指定复制组的详细信息的可选参数。如果忽略此参数，将返回最多 100 个复制组的详细信息。

Example

此命令将返回 `sample-repl-group` 的详细信息。

对于 Linux、macOS 或 Unix：

```
aws elasticache describe-replication-groups \
    --replication-group-id sample-repl-group
```

对于 Windows：

```
aws elasticache describe-replication-groups ^
    --replication-group-id sample-repl-group
```

该命令的输出类似于此处所示。分片（节点组）ID 在此处`####`，以便更容易找到它们。

```
{
    "ReplicationGroups": [
        {
            "Status": "available",
            "Description": "2 shards, 2 nodes (1 + 1 replica)",
            "NodeGroups": [
                {
                    "Status": "available",
                    "Slots": "0-8191",
                    "NodeGroupId": "0001",
                    "NodeGroupMembers": [
                        {
```

```
                            "PreferredAvailabilityZone": "us-west-2c",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "sample-repl-group-0001-001"
                        },
                        {
                            "PreferredAvailabilityZone": "us-west-2a",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "sample-repl-group-0001-002"
                        }
                    ]
                },
                {
                    "Status": "available",
                    "Slots": "8192-16383",
                    "NodeGroupId": "0002",
                    "NodeGroupMembers": [
                        {
                            "PreferredAvailabilityZone": "us-west-2b",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "sample-repl-group-0002-001"
                        },
                        {
                            "PreferredAvailabilityZone": "us-west-2a",
                            "CacheNodeId": "0001",
                            "CacheClusterId": "sample-repl-group-0002-002"
                        }
                    ]
                }
            ],
            "ConfigurationEndpoint": {
                "Port": 6379,
                "Address": "sample-repl-
group.9dcv5r.clustercfg.usw2.cache.amazonaws.com"
            },
            "ClusterEnabled": true,
            "ReplicationGroupId": "sample-repl-group",
            "SnapshotRetentionLimit": 1,
            "AutomaticFailover": "enabled",
            "SnapshotWindow": "13:00-14:00",
            "MemberClusters": [
                "sample-repl-group-0001-001",
                "sample-repl-group-0001-002",
                "sample-repl-group-0002-001",
                "sample-repl-group-0002-002"
```

```
            ],
            "CacheNodeType": "cache.m3.medium",
            "DataTiering": "disabled",
            "PendingModifiedValues": {}
        }
    ]
}
```

使用 ElastiCache API

要查找 Redis（已禁用集群模式）或 Redis（已启用集群模式）复制组的分片（节点组）ID，请使用`describe-replication-groups`带有以下可 AWS CLI 选参数的操作。

- **`ReplicationGroupId`** – 用来将输出限制为指定复制组的详细信息的可选参数。如果忽略此参数，将返回最多 *`xxx`* 个复制组的详细信息。

Example

此命令将返回 `sample-repl-group` 的详细信息。

对于 Linux、macOS 或 Unix：

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeReplicationGroup
   &ReplicationGroupId=sample-repl-group
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

比较 Memcached 和 Redis 自行设计缓存

亚马逊 ElastiCache 支持 Memcached 和 Redis 缓存引擎。每种引擎都有自己的优点。使用本主题中的信息有助于您选择出最能满足您的要求的引擎和版本。

> **Important**
>
> 创建缓存、自行设计的集群或复制组后，您可以升级到较新的引擎版本，但不能降级到较旧的引擎版本。如果要使用较旧的引擎版本，则必须删除现有缓存、自行设计的集群或复制组，然后使用较早的引擎版本重新创建它。

从表面上看，这两个引擎十分类似。其中的每个引擎都是一个内存中键/值存储。不过，这两者实际上有很大差异。

如果您存在以下情况，请选择 Memcached：

- 您需要使模型尽可能简单。
- 您需要运行具有多个核心或线程的大型节点。
- 您需要具备缩放能力，随着系统需求的增加和减少来添加和移除节点。
- 您需要缓存对象。

如果您符合以下条件，请选择版本 ElastiCache 为 Redis 的 Redis：

- ElastiCache 适用于 Redis 版本 7.0（增强版）

  您想要使用 Redis 函数、分片发布/订阅 或 Redis ACL 改进。有关更多信息，请参阅 Redis 版本 7.0（加强版）。
- ElastiCache 适用于 Redis 版本 6.2（增强版）

  您希望能够使用 r6gd 节点类型在内存和 SSD 之间进行数据分层。有关更多信息，请参阅数据分层。
- ElastiCache 适用于 Redis 版本 6.0（增强版）

  您希望使用基于角色的访问控制对用户进行身份验证。

  有关更多信息，请参阅 Redis 版本 6.0（加强版）。
- ElastiCache 适用于 Redis 版本 5.0.0（增强版）

  您需要使用 Redis 流，它是一个日志数据结构，允许生成者实时附加新项，并允许使用者以阻塞或非阻塞方式使用消息。

  有关更多信息，请参阅 Redis 5.0.0 版（加强版）。

- ElastiCache 适用于 Redis 版本 4.0.10（增强版）

  支持加密以及从您的 Redis（已启用集群模式）集群动态添加或移除分区。

  有关更多信息，请参阅 Redis 4.0.10 版（加强版）。

以下版本已弃用、已达到使用寿命或即将达到使用寿命。

- ElastiCache 适用于 Redis 版本 3.2.10（增强版）

  支持从您的 Redis（已启用集群模式）集群动态添加或移除分区的功能。

  > **Important**
  >
  > 目前 ElastiCache ，Redis 3.2.10 不支持加密。

  有关更多信息，请参阅下列内容：

  - Redis 3.2.10 版（加强版）
  - Redis 的在线重新分片最佳实践，有关更多信息，请参阅：
    - 最佳实践：在线重新分片
    - Redis（已启用集群模式）的在线重新分片和分区重新平衡
  - 有关扩展 Redis 集群的更多信息，请参阅扩展。

- ElastiCache 适用于 Redis 版本 3.2.6（增强版）

  如果您需要早期 Redis 版本的功能以及以下功能，请选择 Redi ElastiCache s 3.2.6：

  - 传输中加密。有关更多信息，请参阅 Amazon f ElastiCache or Redis 传输中加密。
  - 静态加密。有关更多信息，请参阅 Amazon for Redi ElastiCache s At-Rest 加密。
- ElastiCache 适用于 Redis（已启用集群模式）版本 3.2.4

  如果需要 Redis 2.8.x 的功能外加以下功能，请选择 Redis 3.2.4 (集群模式)：

  - 您需要在 2 到 500 个节点组（仅限集群模式）之间对数据分区。
  - 您需要地理空间索引（集群模式或非集群模式）。
  - 您不需要支持多个数据库。
- ElastiCache 适用于 Redis（非集群模式）2.8.x 和 3.2.4（增强版）

如果您存在以下情况，请选择 Redis 2.8.x 或 Redis 3.2.4 (非集群模式)：

- 您需要复杂数据类型，如字符串、哈希、列表、集、排序集和位图。
- 您需要对内存数据集进行排序或排名。
- 您需要持久保留密钥库。
- 您需要为读取操作密集型应用程序将主集群中的数据复制到一个或多个只读副本。
- 您需要在主节点出现故障的情况下执行自动故障转移。
- 您需要发布和订阅 (pub/sub) 功能 – 向客户端通知服务器上发生的事件。
- 您需要为自己设计的集群以及无服务器缓存提供备份和还原功能。
- 您需要支持多个数据库。

Memcached、Redis（已禁用集群模式）和 Redis（已启用集群模式）的比较摘要

	Memcached	Redis（已禁用集群模式）	Redis（已启用集群模式）
引擎版本+	1.4.5 及更高版本	4.0.10 及后续版本	4.0.10 及后续版本
数据类型	简便	2.8.x - 复杂 * 复杂	3.2.x 及更高版本 - 复杂
数据分区	是	否	是
集群是可修改的	是	是	3.2.10 及更高版本 - 有限
在线重新分片	否	否	3.2.10 和后续版本
加密	在途 1.6.12 及更高版本	4.0.10 及后续版本	4.0.10 及后续版本
数据分层	否	6.2 及更高版本	6.2 及更高版本
合规性认证			
合规性认证			
FedRAMP	是 - 1.6.12 及更高版本	4.0.10 及后续版本	4.0.10 及后续版本
HIPAA	是 - 1.6.12 及更高版本	4.0.10 及后续版本	4.0.10 及后续版本
PCI DSS	是	4.0.10 及后续版本	4.0.10 及后续版本
多线程	是	否	否
节点类型升级	否	是	是
引擎升级	是	是	是
高可用性 (复制)	否	是	是
自动失效转移	否	可选	必需

	Memcached	Redis（已禁用集群模式）	Redis（已启用集群模式）
发布/订阅功能	否	是	是
排序集	否	是	是
备份与还原	仅适用于无服务器 Memcached，不适用于自行设计的 Memcached 集群	是	是
地理空间索引	否	4.0.10 及后续版本	是

备注：

字符串、对象（如数据库）

* 字符串、集、排序集、列表、哈希、位图、hyperloglog

字符串、集、排序集、列表、哈希、位图、hyperloglog、地理空间索引

+ 不包括已弃用、已达到或即将到期的版本。

为集群选择引擎后，建议您使用该引擎的最新版本。有关更多信息，请参阅 Memcached 版本支持或支持 Redi ElastiCache s 版本。 ElastiCache

联机迁移到 ElastiCache

使用在线迁移，您可以将数据从 Amazon EC2 上的自托管开源 Redis 迁移到 Amazon ElastiCache。

> **Note**
>
> 在 r6gd 节点类型上运行的 ElastiCache 无服务器缓存或集群不支持在线迁移。

概述

要将数据从 Amazon EC2 上运行的开源 Redis 迁移到 Amazon ElastiCache，需要使用现有或新创建的 Amazon ElastiCache 部署。该部署必须具有准备好进行迁移的配置。它还应该符合所需的配置，包括实例类型、分片数量和副本数量等属性。

在线迁移设计用于从 Amazon EC2 上自托管的开源 Redis 到 ElastiCache for Redis 的数据迁移，而不是 ElastiCache for Redis 集群之间的数据迁移。

> **Important**
>
> 我们强烈建议您完整阅读以下部分，然后再开始在线迁移过程。

在调用 `StartMigration` API 操作或 AWS CLI 命令时，将开始迁移。对于已禁用 Redis 集群模式，通过迁移过程，可以使 ElastiCache for Redis 集群的主节点成为源 Redis 主节点的副本。对于已启用 Redis 集群模式，迁移过程使每个 ElastiCache 分片的主节点成为源集群的拥有相同槽的对应分片的副本。

在客户端更改准备就绪后，调用 `CompleteMigration` API 操作。此 API 操作将您的 ElastiCache 部署提升到具有主节点和副本节点（在适用时）的主 Redis 部署。现在，您可以重新导向客户端应用程序以开始将数据写入到 ElastiCache。在整个迁移过程中，您可以通过在 Redis 节点和 ElastiCache 主节点上运行 redis-cli INFO 命令来查看复制状态。

迁移步骤

以下主题简要说明了迁移数据的过程：

- 准备源和目标 Redis 节点以进行迁移
- 测试数据迁移
- 开始迁移
- 验证数据迁移进度
- 完成数据迁移

准备源和目标 Redis 节点以进行迁移

您必须确保已经满足了下面提到的所有四个先决条件，才能开始从 ElastiCache 控制台、API 或 AWS CLI 迁移数据。

准备源和目标 Redis 节点以进行迁移

1. 确定目标 ElastiCache 部署，并确保您可以将数据迁移到该部署。

   现有或新创建的 ElastiCache 部署应满足以下要求才能进行迁移：

   - 它使用的是 Redis 引擎 5.0.6 或更高版本。
   - 它没有启用传输中加密或静态加密。
   - 它已启用多可用区。
   - 它具有足够的可用内存以容纳 Redis 集群中的数据。要配置正确的保留内存设置，请参阅管理预留内存。
   - 对于已禁用集群模式，如果您使用的是 CLI 或 Redis 版本 5.0.6 及以上版本（使用 CLI 或控制台），则可以直接从 Redis 版本 2.8.21 及以上版本迁移到 Redis 版本 5.0.6 及以上版本。对于已启用集群模式，如果您使用的是 CLI 或 Redis 版本 5.0.6 及以上版本（使用 CLI 或控制台），您可以直接从任何已启用集群模式的 Redis 版本迁移到 Redis 版本 5.0.6 及以上版本。
   - 源和目标中的分片数量相匹配。
   - 它不是全局数据存储的一部分。
   - 它已禁用数据分层。

2. 确保开源 Redis 与 ElastiCache for Redis 部署的配置兼容。

   至少，目标 ElastiCache 部署中的所有以下内容应与 Redis 配置兼容才能进行 Redis 复制：

   - Redis 集群不应启用 Redis AUTH。
   - Redis 配置 `protected-mode` 应设置为 `no`。
   - 如果在 Redis 配置中具有 `bind` 配置，应对其进行更新以允许来自 ElastiCache 节点的请求。
   - ElastiCache 节点和 Redis 集群上的逻辑数据库数量应该相同。该值是在 Redis 配置中使用 `databases` 设置的。
   - 不应重命名执行数据修改的 Redis 命令来使数据复制成功完成。例如 `sync`、`psync`、`info`、`config`、`command` 和 `cluster`。
   - 要将数据从 Redis 集群复制到 ElastiCache，请确保具有足够的 CPU 和内存以处理该额外的负载。该负载来自于 Redis 集群创建并通过网络传输到 ElastiCache 节点的 RDB 文件。
   - 源集群上的所有 Redis 实例都应在同一个端口上运行。

3. 执行以下操作，以确保实例可以连接到 ElastiCache：

   - 确保每个实例的 IP 地址是私有的。

- 在与实例上的 Redis 相同的虚拟私有云（VPC）中分配或创建 ElastiCache 部署（建议）。
- 如果 VPC 不同，请设置 VPC 对等以允许集群之间的访问。有关 VPC 对等的更多信息，请参阅访问 Amazon VPC 中 ElastiCache 缓存的访问模式。
- 附加到 Redis 实例的安全组应允许来自 ElastiCache 节点的入站流量。

4. 在数据迁移完成后，确保应用程序可以将流量传送到 ElastiCache 节点。有关更多信息，请参阅访问 Amazon VPC 中 ElastiCache 缓存的访问模式。

测试数据迁移

在满足所有先决条件后，您可以使用 AWS Management Console、ElastiCache API 或 AWS CLI 验证迁移设置。下面显示了使用 CLI 的示例。

使用以下参数调用 `test-migration` 命令以测试迁移：

- `--replication-group-id` – 数据要迁移到的复制组的 ID。
- `--customer-node-endpoint-list` – 应从中迁移数据的端点的列表。列表应该只有一个元素。

以下是一个使用 CLI 的示例。

```
aws elasticache test-migration --replication-group-id test-cluster --customer-node-endpoint-list "Address='10.0.0.241',Port=6379"
```

ElastiCache 将在不进行任何实际数据迁移的情况下验证迁移设置。

开始迁移

在满足所有先决条件后，您可以使用 AWS Management Console、ElastiCache API 或 AWS CLI 开始迁移数据。如果启用集群模式，当插槽迁移有所不同时，将在实时迁移之前执行重新分片。下面显示了使用 CLI 的示例。

> **Note**
>
> 我们建议使用 `TestMigration` API 来验证迁移设置。但这完全是可选的。

可以使用以下参数调用 `start-migration` 命令以开始进行迁移：

- `--replication-group-id` – 目标 ElastiCache 复制组的标识符

- `--customer-node-endpoint-list` – 具有 DNS 或 IP 地址以及运行源 Redis 集群的端口的端点列表。对于禁用集群模式和启用集群模式，列表只能使用一个元素。如果启用了链式复制，则端点可能指向副本，而不是 Redis 集群中的主节点。

以下是一个使用 CLI 的示例。

```
aws elasticache start-migration --replication-group-id test-cluster --customer-node-endpoint-list "Address='10.0.0.241',Port=6379"
```

当您运行此命令时，ElastiCache 主节点（在每个分片中）会将自己配置成为您的 Redis 实例的副本（在启用集群的 redis 中拥有相同插槽的相应分片中）。ElastiCache 集群的状态将变为正在迁移，并开始将数据从 Redis 实例迁移到 ElastiCache 主节点。根据 Redis 实例上的数据大小和负载，迁移可能需要一段时间才能完成。您可以在 Redis 实例和 ElastiCache 主节点上运行 redis-cli INFO 命令来检查迁移进程。

在成功复制后，写入到 Redis 实例的所有数据都将传播到 ElastiCache 集群。您可以使用 ElastiCache 节点执行读取操作。不过，您无法写入到 ElastiCache 集群。如果 ElastiCache 主节点连接了其他副本节点，这些副本节点将继续从 ElastiCache 主节点中进行复制。这样，Redis 集群中的所有数据将复制到 ElastiCache 集群中的所有节点。

如果 ElastiCache 主节点无法成为 Redis 实例的副本，则它会重试几次，然后最终将自身重新提升回为主节点。ElastiCache 集群的状态将变为 available（可用），并发送有关启动迁移失败的复制组事件。要解决此类故障，请检查以下内容：

- 查看复制组事件。使用事件中的任何特定信息修复迁移失败。
- 如果事件未提供任何特定信息，请确保您遵循准备源和目标 Redis 节点以进行迁移中的准则。
- 确保 VPC 和子网的路由配置允许 ElastiCache 节点和 Redis 实例之间的流量。
- 确保附加到 Redis 实例的安全组允许来自 ElastiCache 节点的入站流量。
- 检查 Redis 实例的 Redis 日志，以获取有关特定于复制的失败的更多信息。

验证数据迁移进度

在数据迁移开始后，您可以执行以下操作以跟踪其进度：

- 在 ElastiCache 主节点上验证 `INFO` 命令中的 Redis `master_link_status` 是否为 up。您还可以在 ElastiCache 控制台中找到该信息。选择集群，然后在 CloudWatch metrics（CloudWatch 指标）下观察 Primary Link Health Status（主链接运行状况）。在该值达到 1 后，数据就会同步。

- 您可以在 Redis 实例上运行 `INFO` 命令来查看 ElastiCache 副本是否具有在线状态。这样做还会提供有关复制滞后的信息。
- 对 Redis 实例使用 CLIENT LIST Redis 命令以验证低客户端输出缓冲区。

在数据迁移完成后，数据与进入 Redis 集群的主节点的任何新的写入保持同步。

完成数据迁移

在准备好切换到 ElastiCache 集群时，请使用具有以下参数的 `complete-migration` CLI 命令：

- `--replication-group-id` – 复制组的标识符。
- `--force` – 此值强制停止迁移，而不确保数据保持同步。

以下是示例。

```
aws elasticache complete-migration --replication-group-id test-cluster
```

在运行该命令时，ElastiCache 主节点（在每个分片中）将停止从 Redis 实例进行复制，并将其提升为主实例。该提升通常在几分钟内完成。要确认提升为主节点，请检查事件 `Complete Migration successful for test-cluster`。此时，您可以将应用程序定向到 ElastiCache 写入和读取。ElastiCache 集群状态应从 migrating（正在迁移）变为 available（可用）。

如果提升为主实例失败，ElastiCache 主节点将继续从 Redis 实例中进行复制。ElastiCache 集群继续处于 migrating（正在迁移）状态，并发送有关该失败的复制组事件消息。要解决该故障，请查看以下内容：

- 检查复制组事件。使用事件中的特定信息修复故障。
- 您可能会收到有关数据不同步的事件消息。如果收到该消息，请确保 ElastiCache 主节点可以从 Redis 实例中复制，并且两者保持同步。如果仍要停止迁移，您可以使用 `–force` 选项运行前面的命令。
- 如果正在替换其中一个 ElastiCache 节点，您可能会收到事件消息。在完成替换后，您可以重新尝试完成迁移步骤。

使用控制台执行联机数据迁移

您可以使用 AWS Management Console将数据从您的集群迁移到 Redis 集群。

使用控制台执行联机数据迁移

1. 登录到控制台并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 创建新的 Redis 集群，或者选择现有的集群。确保集群满足以下要求：
   - Redis 引擎版本应为 5.0.6 或更高版本。
   - Redis 集群不应启用 Redis AUTH。
   - Redis 配置 `protected-mode` 应设置为 `no`。
   - 如果在 Redis 配置中具有 `bind` 配置，应对其进行更新以允许来自 ElastiCache 节点的请求。
   - ElastiCache 节点和您的 Redis 集群的数据库数量应该相同。该值是在 Redis 配置中使用 `databases` 设置的。
   - 不应重命名执行数据修改的 Redis 命令，以使数据复制成功完成。
   - 要将数据从 Redis 集群复制到 ElastiCache，请确保具有足够的 CPU 和内存以处理该额外的负载。该负载来自于 Redis 集群创建并通过网络传输到 ElastiCache 节点的 RDB 文件。
   - 集群处于 available (可用) 状态。
3. 选择您的集群后，在 Actions（操作）中选择 Migrate Data from Endpoint（从端点迁移数据）。
4. 在从端点迁移数据对话框中，输入 Redis 集群可用的 IP 地址和端口。

   > **Important**
   >
   > IP 地址必须准确。如果未正确输入地址，迁移将失败。

5. 选择 Start Migration (开始迁移)。

   在集群开始迁移时，它将变为 Modifying (正在修改) 状态，然后变为 Migrating (正在迁移) 状态。
6. 在导航窗格上选择 Events (事件) 以监视迁移进度。

在迁移过程中的任何时候，您都可以停止迁移。为此，选择您的集群，然后在 Actions（操作）中选择 Stop Data Migration（停止数据迁移）。然后，集群将变为 Available (可用) 状态。

如果迁移成功，集群将变为 Available (可用) 状态，并且事件日志显示以下内容：

`Migration operation succeeded for replication group` *`ElastiCacheClusterName`*`.`

如果迁移失败，集群将变为 Available (可用) 状态，并且事件日志显示以下内容：

`Migration operation failed for replication group` *`ElastiCacheClusterName`*`.`

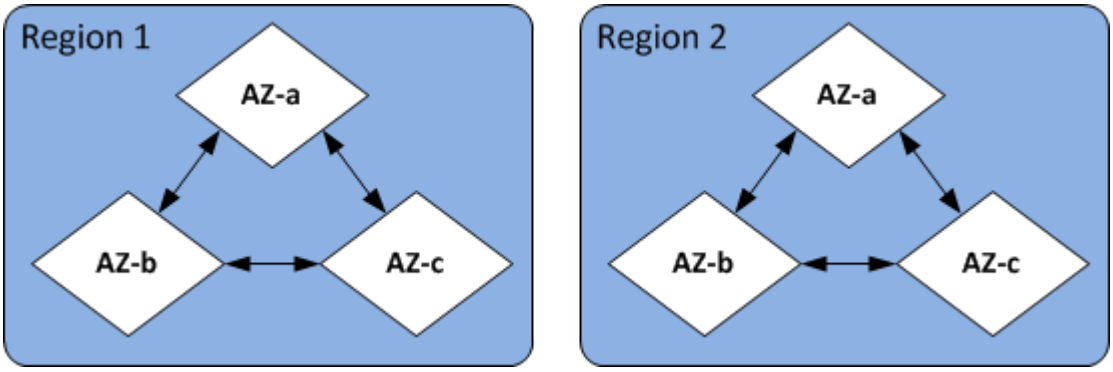
选择区域和可用区

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

AWS 区域很大，而且广泛分散在不同的地理位置。可用区是一个 AWS 区域内的不同位置，旨在隔绝其他可用区域的故障。它们为同一 AWS 区域中的其他可用区提供低成本、低延迟的网络连接。

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

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



区域和可用区

主题

- 找到您的节点
- 支持的区域和端点
- 将 Local Zones 与 ElastiCache 结合使用
- 使用 Outposts

找到您的节点

Amazon ElastiCache 支持将集群的所有节点定位在单个或多个可用区 (AZ) 中。此外，如果您选择将节点放置在多个可用区中（推荐），则 ElastiCache 可以为每个节点选择可用区，也可以ElastiCache 允许您选择这些可用区。

通过在不同的可用区内放置节点，可排除某个可用区内的故障（如停电）导致整个系统失败的可能性。测试表明，将所有节点放置在同一个可用区与跨多个可用区放置节点相比并不存在显著延迟。

您可以在创建集群时为每个节点指定可用区，或在修改现有集群时通过添加节点来指定可用区。有关更多信息，请参阅下列内容：

- 创建集群
- 修改集 ElastiCache 群
- 向集群添加节点

支持的区域和端点

ElastiCache Amazon 在多个 AWS 地区可用。这意味着您可以在满足您要求的位置启动 ElastiCache 集群。例如，您可以在离客户最近的 AWS 地区推出，或者在满足特定法律要求的特定 AWS 地区开店。

从设计而言，每个 区域都与其他 区域完全隔离。在每个区域中有多个可用区（AZ）。ElastiCache 无服务器缓存会自动跨多个可用区复制数据（在两个可用区中复制数据除外）us-west-1，以实现高可用性。在设计自己的 ElastiCache 集群时，您可以选择在不同的可用区中启动节点以实现容错。有关区域和可用区的更多信息，请参阅此主题顶部的选择区域和可用区。

支持 ElastiCache 的地区

区域名称/区域	终端节点	协议	
美国东部（俄亥俄州）区域 us-east-2	elasticache.us-east-2.amazonaws.com	HTTPS	
美国东部（弗吉尼亚州北部）区域	elasticache.us-east-1.amazonaws.com	HTTPS	

区域名称/区域	终端节点	协议	
us-east-1			
美国西部（北加利福尼亚）区域 us-west-1	elasticache.us-west-1.amazonaws.com	HTTPS	
美国西部（俄勒冈州）区域 us-west-2	elasticache.us-west-2.amazonaws.com	HTTPS	
加拿大（中部）区域 ca-central-1	elasticache.ca-central-1.amazonaws.com	HTTPS	
加拿大（西部）区域 ca-west-1	elasticache.ca-west-1.amazonaws.com	HTTPS	
亚太地区（雅加达） ap-southeast-3	elasticache.ap-southeast-3.amazonaws.com	HTTPS	
亚太地区（孟买）区域 ap-south-1	elasticache.ap-south-1.amazonaws.com	HTTPS	
亚太地区（海得拉巴）区域 ap-south-2	elasticache.ap-south-2.amazonaws.com	HTTPS	
亚太地区（东京）区域 ap-northeast-1	elasticache.ap-northeast-1.amazonaws.com	HTTPS	

区域名称/区域	终端节点	协议	
亚太地区（首尔）区域 ap-northeast-2	elasticache.ap-northeast-2.amazonaws.com	HTTPS	
亚太地区（大阪）区域 ap-northeast-3	elasticache.ap-northeast-3.amazonaws.com	HTTPS	
亚太地区（新加坡）区域 ap-southeast-1	elasticache.ap-southeast-1.amazonaws.com	HTTPS	
亚太地区（悉尼）区域 ap-southeast-2	elasticache.ap-southeast-2.amazonaws.com	HTTPS	
欧洲地区（法兰克福）区域 eu-central-1	elasticache.eu-central-1.amazonaws.com	HTTPS	
欧洲地区（苏黎世）地区 eu-central-2	elasticache.eu-central-2.amazonaws.com	HTTPS	
欧洲地区（斯德哥尔摩）区域 eu-north-1	elasticache.eu-north-1.amazonaws.com	HTTPS	
中东（巴林）区域 me-south-1	elasticache.me-south-1.amazonaws.com	HTTPS	

区域名称/区域	终端节点	协议	
中东（阿联酋）区域 me-central-1	elasticache.me-central-1.amazonaws.com	HTTPS	
欧洲地区（爱尔兰）区域 eu-west-1	elasticache.eu-west-1.amazonaws.com	HTTPS	
欧洲地区（伦敦）区域 eu-west-2	elasticache.eu-west-2.amazonaws.com	HTTPS	
欧洲地区（巴黎）区域 eu-west-3	elasticache.eu-west-3.amazonaws.com	HTTPS	
欧洲地区（米兰）区域 eu-south-1	elasticache.eu-south-1.amazonaws.com	HTTPS	
欧洲地区（西班牙）区域 eu-south-2	elasticache.eu-south-2.amazonaws.com	HTTPS	
南美洲（圣保罗）区域 sa-east-1	elasticache.sa-east-1.amazonaws.com	HTTPS	
中国（北京）区域 cn-north-1	elasticache.cn-north-1.amazonaws.com.cn	HTTPS	

区域名称/区域	终端节点	协议	
中国（宁夏）区域 cn-northwest-1	elasticache.cn-northwest-1.amazonaws.com.cn	HTTPS	
亚太地区（香港）区域 ap-east-1	elasticache.ap-east-1.amazonaws.com	HTTPS	
非洲（开普敦）区域 af-south-1	elasticache.af-south-1.amazonaws.com	HTTPS	
以色列（特拉维夫）区域 il-central-1	elasticache.il-central-1.amazonaws.com	HTTPS	
AWS GovCloud （美国西部） us-gov-west-1	elasticache.us-gov-west-1.amazonaws.com	HTTPS	
AWS GovCloud （美国东部） us-gov-east-1	elasticache.us-gov-east-1.amazonaws.com	HTTPS	
有关将 AWS GovCloud （美国）与配合使用的信息 ElastiCache，请参阅 AWS GovCloud （美国）地区的服务： ElastiCache。			

某些区域只支持部分节点类型。有关按 AWS 地区划分的支持的节点类型的表，请参阅AWS 区域支持的节点类型。

如需按地区列出的 AWS 产品和服务表，请参阅按地区划分的产品和服务。

将 Local Zones 与 ElastiCache 结合使用

本地扩展区 是在地理位置上靠近用户的 AWS 区域的扩展。您可以通过创建新子网并将其分配到 Local Zones，将任何 Virtual Private Cloud (VPC) 从 AWS 父区域扩展到 Local Zones。当您在本地扩展区中创建子网时，VPC 也会扩展到该本地扩展区。本地扩展区中的子网与 VPC 中其他子网的运行相同。

通过使用 Local Zones，您可以将 ElastiCache 集群等资源放置在靠近用户的多个位置。

创建 ElastiCache 集群时，您可以选择 Local Zones 中的子网。Local Zones 有自己的 Internet 连接并支持 AWS Direct Connect。因此，在本地扩展区中创建的资源可以通过非常低延迟的通信为本地用户提供服务。有关更多信息，请参阅 AWSLocal Zones。

本地扩展区由 AWS 区域代码后跟一个指示位置的标识符表示，例如 `us-west-2-lax-1a`。

目前，可用的 Local Zones 是 `us-west-2-lax-1a` 和 `us-west-2-lax-1b`。

以下限制适用于 Local Zones 的 ElastiCache：

- 不支持全局数据存储。
- 不支持在线迁移。
- Local Zones 目前支持以下节点类型：
  - 最新一代：

    M5 节点类型：`cache.m5.large`、`cache.m5.xlarge`、`cache.m5.2xlarge`、`cache.m5.4xlarge`、`cache.`

    R5 节点类型：`cache.r5.large`、`cache.r5.xlarge`、`cache.r5.2xlarge`、`cache.r5.4xlarge`、`cache.`

    T3 节点类型：`cache.t3.micro`、`cache.t3.small`、`cache.t3.medium`

启用本地区域

1. 在 Amazon EC2 控制台中启用本地扩展区。

   有关更多信息，请参阅 Amazon EC2 用户指南（适用于 Linux 实例）中的启用 Local Zones。
2. 在本地扩展区中创建子网。

   有关更多信息，请参阅 Amazon VPC 用户指南 中的在 VPC 中创建子网。
3. 在 Local Zones 中创建 ElastiCache 子网组。

创建 ElastiCache 子网组时，请为 Local Zones 选择可用区组。

有关更多信息，请参阅 ElastiCache 用户指南中的创建子网组。

4. 创建一个使用 Local Zones 中的 ElastiCache 子网的 ElastiCache for Redis 集群。有关更多信息，请参阅下列主题之一：

   - 创建 Redis（已禁用集群模式）集群（控制台）
   - 创建 Redis（已启用集群模式）集群（控制台）

使用 Outposts

AWS Outposts 是一项完全托管的服务，可将 AWS 基础架构、服务、API 和工具扩展到客户驻地。通过提供对 AWS 托管基础设施的本地访问权限， AWS Outposts使客户能够使用与 AWS 区域相同的编程接口在本地构建和运行应用程序，同时使用本地计算和存储资源来降低延迟和满足本地数据处理需求。Outpost 是部署在客户现场的 AWS 计算和存储容量池。 AWS 将此容量作为 AWS 区域的一部分进行运营、监控和管理。您可以在 Outpost 上创建子网，并在创建集群等 AWS ElastiCache 资源时指定子网。

> **Note**
>
> 在此版本中，以下限制适用：
>
> - ElastiCache for Outposts 仅支持 M5 和 R5 节点系列。
> - 多可用区（不支持跨站点复制）。
> - 不支持实时迁移。
> - 不支持本地快照。
> - 无法启用引擎日志和慢速日志。
> - ElastiCache on Outposts 不支持 CoIP。
> - ElastiCache 以下区域不支持 for Outposts：cn-northeast-1、cn-northeast-1 和 ap-northeast-3。

将 Outposts 与 Redis 控制台一起使用

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。

2. 在导航窗格上，选择 Redis 缓存。
3. 选择创建 Redis 缓存。
4. 在集群设置下，选择设计自己的缓存和集群缓存。将 “集群模式” 设置为 “已禁用”。然后为缓存创建名称和可选描述。
5. 要查看位置，请选择本地部署。
6. 在 “本地” 部分中，您将看到 “前哨基地 ID” 字段。输入集群运行位置的 ID。

   “集群设置” 下的所有其他设置都可以保持默认状态。
7. 在连接中，选择创建新的子网组并输入 VPC ID。将其余部分保留为默认值，然后选择 “下一步”。

配置本地选项

您可以选择一个可用的 Outpost 来添加缓存集群，或者，如果没有可用的 Outposts，请使用以下步骤创建一个新的缓存集群：

在 On-Premises options（本地选项）下：

1. 在 Redis settings（Redis 设置）下：
   a. Name（名称）：为 Redis 集群输入名称。
   b. Description（描述）：输入 Redis 集群的描述。
   c. 引擎版本兼容性：引擎版本基于 Outpost 区域 AWS
   d. Port（端口），接受默认端口 6379。如果您出于某个原因需要使用其他端口，请键入相应的端口号。
   e. Parameter group（参数组）：使用下拉菜单选择默认或自定义参数组。
   f. Node Type（节点类型）：可用实例基于 Outposts 可用性。适用于 Outposts 的 Porting Assistant for .NET 仅支持 M5 和 R5 节点系列。从下拉列表中，选择 Outposts，然后选择要用于此集群的可用节点类型。然后选择 Save（保存）。
   g. Number of Replicas（副本数）：输入要为此复制组创建的只读副本数。您必须至少有一个只读副本，但不超过五个。默认值是 2。

      只读副本的自动生成的名称与主群集名称的模式相同，末尾添加一个破折号和连续三位数字，并以 `-002` 开头。例如，如果您的复制组名为 `MyGroup`，辅助集群的名称将为 `MyGroup-002`、`MyGroup-003`、`MyGroup-004`、`MyGroup-005`、`MyGroup-006`。
2. 在 “连接” 下：

a. Subnet Group（子网组）：从列表中选择 Create new（创建新子网组）。

- Name（名称）：输入子网组的名称
- Description（描述）：输入子网组的描述
- VPC ID：VPC ID 应与 Outpost VPC 一致。如果您选择的 VPC 在 Outposts 上没有子网 ID，则列表将返回为空。
- Availability Zone or Outpost（可用区或 Outpost）：选择您正在使用的 Outpost。
- Subnet ID（子网 ID）：选择可用于 Outpost 的子网 ID。如果没有可用的子网 ID，则需要创建它们。有关更多信息，请参阅创建子网。

b. 选择创建。

查看 Outpost 集群详细信息

在 Redis 列表页面上，选择属于 AWS Outpost 的集群，并在查看集群详细信息时注意以下事项：

- 可用区域：这将使用 ARN（Amazon 资源名称）和 AWS 资源编号表示前哨基地。
- 前哨基地名称：前 AWS 哨基地的名称。

在 CLI 中使用 Outposts AWS

您可以使用 AWS Command Line Interface (AWS CLI) 从命令行控制多项 AWS 服务，并通过脚本自动执行这些服务。您可以使用 AWS CLI 进行临时（一次性）操作。

正在下载和配置 AWS CLI

它们可以在 Windows、macOS 或 Linux 上 AWS CLI 运行。按照以下步骤下载和并对其进行配置。

下载、安装和配置 CLI

1. 在AWS 命令行界面网页上下载 AWS CLI。
2. 按照《AWS Command Line Interface 用户指南》中有关安装 AWS CLI 和配置 AWS CLI 的说明进行操作。

在 Outposts 中使用 AWS CLI

使用以下 CLI 操作创建使用 Outposts 的缓存集群：

- create-cache-cluster— 使用此操作，`outpost-mode`参数接受一个值，该值指定缓存集群中的节点是在单个 Outpost 中创建的，还是在多个 Outposts 中创建的。

> **Note**
>
> 目前仅支持 `single-outpost` 模式。

```
aws elasticache create-cache-cluster \
   --cache-cluster-id cache cluster id \
   --outpost-mode single-outpost \
```

与 ElastiCache

在本节中，您可以找到有关如何管理 ElastiCache 实施中各个组件的详细信息。

主题

快照和还原

运行 Redis Serverless 可以通过创建快照来备份其数据。您可以使用备份将缓存或种子数据还原到新缓存。备份包含缓存的元数据以及缓存中的所有数据。所有备份都会写入 Amazon Simple Storage Service (Amazon S3)，该服务提供持久存储。您可以随时通过创建新的 Redis 并在其中填充备份中的数据来恢复数据。使用 ElastiCache，您可以使用 AWS Management Console、 AWS Command Line Interface (AWS CLI) 和 ElastiCache API 管理备份。

如果您计划删除缓存并且保留数据很重要，则可以采取额外的预防措施。为此，请先创建手动备份，验证其状态是否为可用，然后删除缓存。这样做可确保如果备份失败，您仍然可以使用缓存数据。您可以按照前面概述的最佳实践重新尝试创建备份。

主题

- 创建最终备份
- 描述备份
- 复制备份
- 导出备份
- 从备份还原到新缓存
- 删除备份
- 标记备份
- 使用外部创建的备份为新的自行设计的集群制作种子

备份约束

在计划或创建备份时考虑以下约束：

- 只有在 Redis 或 Serverless Memcached 上运行的缓存才支持备份和恢复。
- Redis（已禁用集群模式）集群的 `cache.t1.micro` 节点上不支持备份和还原。支持所有其他缓存节点类型。
- Redis（已启用集群模式）集群的所有节点类型均支持备份和还原。
- 在任何连续的 24 小时内，每个无服务器缓存只能创建不超过 24 个手动备份。对于 Redis 自行设计的集群，您可以在集群中为每个节点创建不超过 20 个手动备份。
- Redis（已启用集群模式）仅支持在集群级别（对于 API 或 CLI，为复制组级别）进行备份。Redis（已启用集群模式）不支持在分区级别（对于 API 或 CLI，为节点组级别）进行备份。
- 在备份过程中，您无法在无服务器缓存上运行任何其他 API 或 CLI 操作。备份期间，您可以在自行设计的集群上运行 API 或 CLI 操作。
- 如果使用带数据分层的缓存，则无法将备份导出到 Amazon S3。
- 您只能将使用 r6gd 节点类型的集群备份还原到使用 r6gd 节点类型的集群。

备份自行设计的集群所产生的性能影响

无服务器缓存上的备份对于应用程序是透明的，并且不会影响性能。但是，在为自行设计的集群创建备份时，可能会产生一些性能影响，具体取决于可用的预留内存。自行设计的集群不适用于 ElastiCache 和 Memcached，但可与 ElastiCache 和 Redis 一起使用。

以下是提高自行设计的集群的备份性能的准则。

- 设置 `reserved-memory-percent` 参数 – 为了缓解过多分页问题，我们建议设置 reserved-memory-percent 参数。此参数可防止 Redis 使用节点的所有可用内存，有助于减少分页量。只需使用更大的节点，您也可以获得性能改进。有关 reserved-memory 和 reserved-memory-percent 参数的更多信息，请参阅 管理预留内存。

- 从只读副本创建备份 – 如果您正在具有多个节点的节点组中运行 Redis，则可以从主节点或一个只读副本进行备份。由于在 BGSAVE 期间需要系统资源，因此我们建议您从一个只读副本上创建备份。从副本创建备份时，主节点不受 BGSAVE 资源要求的影响。主节点可以继续处理请求，而不会降低速度。

  若要执行此操作，请参阅 创建手动备份（控制台），并在 Create Backup（创建备份）窗口中的 Cluster Name（集群名称）字段中，选择副本而不是默认主节点。

如果您删除复制组并请求最终备份，则 ElastiCache 始终从主节点获取备份。这可确保您在删除复制组之前捕获最新的 Redis 数据。

计划自动备份

您可以为任何 Redis 无服务器缓存或自行设计的集群启用自动备份。启用自动备份后，ElastiCache 将每天创建缓存的备份。自动备份不会对缓存产生任何影响，而且更改是即时发生的。自动备份可以帮助防止数据丢失。出现故障时，您可以创建新的缓存，并从最新的备份中恢复数据。这样可得到热启动的缓存，其中预先加载了您的数据，已准备好可供使用。有关更多信息，请参阅 从备份还原到新缓存。

当您计划自动备份时，应规划以下设置：

- 备份开始时间 — 一天中 ElastiCache 开始创建备份的时间。您可以将备份时段设置为任何方便的时间。如果您未指定备份窗口，则会自动 ElastiCache 分配一个备份窗口。

- 备份保留期限 – 备份在 Amazon S3 中保留的天数。例如，如果您将保留期限设置为 5，则当天进行的备份将保留 5 天。保留期限过期时，会自动删除备份。

  最大备份保留期限为 35 天。如果备份保留期限设置为 0，则会为缓存禁用自动备份。

在创建新缓存或更新现有 Redis 缓存时，您可以使用 ElastiCache 控制台、或 ElastiCache API 启用或禁用自动备份。 AWS CLI这是通过选中 “高级 Redis 设置” 部分中的 “启用自动备份” 复选框来完成的。

进行手动备份

除了自动备份以外，您还可以随时创建手动 备份。与在指定保留期之后自动删除的自动备份不同，手动备份并没有在超过之后就会自动删除的保留期。即使您删除缓存，也会保留该缓存中的任何手动备份。如果您不再需要保留某个手动备份，您必须自行显式删除它。

除了直接创建手动备份外，您还可以通过下列方法之一创建手动备份：

- 复制备份。源备份是自动还是手动创建并不重要。
- 创建最终备份。创建备份，然后立即删除集群或节点。

您可以使用 AWS Management Console、或 ElastiCache API 创建缓存的 AWS CLI手动备份。

创建手动备份（控制台）

创建缓存的备份（控制台）

1. 登录 AWS Management Console 并打开亚马逊 EC2 控制台，网址为 https://console.aws.amazon.com/ec2/。
2. 从导航窗格中，选择 Redis 缓存。
3. 选择要备份的缓存名称左侧的复选框。
4. 选择 Backup。
5. 在 Create Backup 对话框的 Backup Name 框中键入备份的名称。建议该名称指明所备份的集群以及进行备份的日期和时间。

   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. 选择 Create Backup。

   集群的状态将变为快照。

创建手动备份 (AWS CLI)

使用手动备份无服务器缓存 AWS CLI

要使用创建缓存的手动备份 AWS CLI，请使用带有以下参数的`create-serverless-snapshot` AWS CLI 操作：

- `--serverless-cache-name`：要备份的无服务器缓存的名称。
- `--serverless-cache-snapshot-name` – 要创建的快照的名称。

对于 Linux、macOS 或 Unix：

- ```
  aws elasticache create-serverless-snapshot \
                          --serverless-cache-name CacheName \
                          --serverless-cache-snapshot-name bkup-20231127
  ```

对于 Windows：

- ```
  aws elasticache create-serverless-snapshot ^
      --serverless-cache-name CacheName ^
      --serverless-cache-snapshot-name bkup-20231127
  ```

使用手动备份自己设计的集群 AWS CLI

要使用为自己设计的集群创建手动备份 AWS CLI，请使用带有以下参数的`create-snapshot` AWS CLI 操作：

- `--cache-cluster-id`
  - 如果您正备份的集群没有副本节点，则 `--cache-cluster-id` 是您正备份的集群的名称，例如，*`mycluster`*。
  - 如果您正备份的集群有一个或多个副本节点，则 `--cache-cluster-id` 是您要用于备份的集群中节点的名称。例如，名称可能为 *`mycluster-002`*。

  仅在备份 Redis（已禁用集群模式）集群时使用此参数。

- `--replication-group-id` – 用作备份源的 Redis（已启用集群模式）集群的名称（CLI/API：复制组）。在备份 Redis（已启用集群模式）集群时使用此参数。

- `--snapshot-name` – 要创建的快照的名称。

  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。

示例 1：备份无副本节点的 Redis（已禁用集群模式）集群

以下 AWS CLI 操作`bkup-20150515`从没有只读副本的 Redis (已禁用集群模式) 集群`myNonClusteredRedis`创建备份。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-snapshot \
    --cache-cluster-id myNonClusteredRedis \
    --snapshot-name bkup-20150515
```

对于 Windows：

```
aws elasticache create-snapshot ^
    --cache-cluster-id myNonClusteredRedis ^
    --snapshot-name bkup-20150515
```

示例 2：备份有副本节点的 Redis（已禁用集群模式）集群

以下 AWS CLI 操作`bkup-20150515`从 Redis (已禁用集群模式) 集群创建备份。`myNonClusteredRedis`此备份具有一个或多个只读副本。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-snapshot \
    --cache-cluster-id myNonClusteredRedis-001 \
```

```
    --snapshot-name bkup-20150515
```

对于 Windows：

```
aws elasticache create-snapshot ^
    --cache-cluster-id myNonClusteredRedis-001 ^
    --snapshot-name bkup-20150515
```

示例输出：备份有副本节点的 Redis（已禁用集群模式）集群

此操作的输出将类似于下文。

```
{
    "Snapshot": {
        "Engine": "redis",
        "CacheParameterGroupName": "default.redis6.x",
        "VpcId": "vpc-91280df6",
        "CacheClusterId": "myNonClusteredRedis-001",
        "SnapshotRetentionLimit": 0,
        "NumCacheNodes": 1,
        "SnapshotName": "bkup-20150515",
        "CacheClusterCreateTime": "2017-01-12T18:59:48.048Z",
        "AutoMinorVersionUpgrade": true,
        "PreferredAvailabilityZone": "us-east-1c",
        "SnapshotStatus": "creating",
        "SnapshotSource": "manual",
        "SnapshotWindow": "08:30-09:30",
        "EngineVersion": "6.0",
        "NodeSnapshots": [
            {
                "CacheSize": "",
                "CacheNodeId": "0001",
                "CacheNodeCreateTime": "2017-01-12T18:59:48.048Z"
            }
        ],
        "CacheSubnetGroupName": "default",
        "Port": 6379,
        "PreferredMaintenanceWindow": "wed:07:30-wed:08:30",
        "CacheNodeType": "cache.m3.2xlarge",
        "DataTiering": "disabled"
    }
}
```

示例 3：备份 Redis（启用集群模式）的集群

以下 AWS CLI 操作bkup-20150515从 Redis (已启用集群模式) 集群创建备份。myClusteredRedis请注意使用 --replication-group-id 而非 --cache-cluster-id 来标识源。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-snapshot \
    --replication-group-id myClusteredRedis \
    --snapshot-name bkup-20150515
```

对于 Windows：

```
aws elasticache create-snapshot ^
    --replication-group-id myClusteredRedis ^
    --snapshot-name bkup-20150515
```

示例输出：备份 Redis（已启用集群模式）集群

此操作的输出将类似于下文。

```
{
    "Snapshot": {
        "Engine": "redis",
        "CacheParameterGroupName": "default.redis6.x.cluster.on",
        "VpcId": "vpc-91280df6",
        "NodeSnapshots": [
            {
                "CacheSize": "",
                "NodeGroupId": "0001"
            },
            {
                "CacheSize": "",
                "NodeGroupId": "0002"
            }
        ],
        "NumNodeGroups": 2,
        "SnapshotName": "bkup-20150515",
        "ReplicationGroupId": "myClusteredRedis",
        "AutoMinorVersionUpgrade": true,
        "SnapshotRetentionLimit": 1,
```

```
        "AutomaticFailover": "enabled",
        "SnapshotStatus": "creating",
        "SnapshotSource": "manual",
        "SnapshotWindow": "10:00-11:00",
        "EngineVersion": "6.0",
        "CacheSubnetGroupName": "default",
        "ReplicationGroupDescription": "2 shards 2 nodes each",
        "Port": 6379,
        "PreferredMaintenanceWindow": "sat:03:30-sat:04:30",
        "CacheNodeType": "cache.r3.large",
        "DataTiering": "disabled"
    }
}
```

相关 主题

有关更多信息，请参阅 AWS CLI 命令参考中的 create-snapshot。

创建最终备份

您可以使用 ElastiCache 控制台 AWS CLI、或 ElastiCache API 创建最终备份。

创建最终备份（控制台）

当您使用控制台删除 Redis 无服务器缓存或自行设计的集群时，您可以创建最终备份。 ElastiCache

要在删除缓存时创建最终备份，请在删除对话框中在 “创建备份” 下选择 “是”，然后为备份命名。

相关 主题

- 使用 AWS Management Console
- 删除复制组（控制台）

创建最终备份 (AWS CLI)

删除缓存时，您可以使用创建最终备份 AWS CLI。

主题

- 删除无服务器缓存时
- 删除没有只读副本的 Redis 自行设计的集群时
- 删除有只读副本的 Redis 集群时

删除无服务器缓存时

要创建最终备份，请使用带有以下参数的`delete-serverless-cache` AWS CLI 操作。

- `--serverless-cache-name`：要删除的缓存的名称。
- `--final-snapshot-name` – 备份的名称。

以下代码在删除缓存 `myserverlesscache` 时创建最终备份 `bkup-20231127-final`。

对于 Linux、macOS 或 Unix：

```
aws elasticache delete-serverless-cache \
        --serverless-cache-name myserverlesscache \
        --final-snapshot-name bkup-20231127-final
```

对于 Windows：

```
aws elasticache delete-serverless-cache ^
        --serverless-cache-name myserverlesscache ^
        --final-snapshot-name bkup-20231127-final
```

有关更多信息，请参阅《AWS CLI Command Reference》中的 delete-serverless-cache。

删除没有只读副本的 Redis 自行设计的集群时

要为自己设计的没有只读副本的集群创建最终备份，请使用带有以下参数的`delete-cache-cluster` AWS CLI 操作。

- `--cache-cluster-id` – 要删除的集群的名称。
- `--final-snapshot-identifier` – 备份的名称。

以下代码在删除集群 `myRedisCluster` 时创建最终备份 `bkup-20150515-final`。

对于 Linux、macOS 或 Unix：

```
aws elasticache delete-cache-cluster \
        --cache-cluster-id myRedisCluster \
        --final-snapshot-identifier bkup-20150515-final
```

对于 Windows：

```
aws elasticache delete-cache-cluster ^
        --cache-cluster-id myRedisCluster ^
        --final-snapshot-identifier bkup-20150515-final
```

有关更多信息，请参阅 AWS CLI 命令参考中的 delete-cache-cluster。

删除有只读副本的 Redis 集群时

要在删除复制组时创建最终备份，请使用带有以下参数的`delete-replication-group` AWS CLI 操作：

- `--replication-group-id` – 要删除的复制组的名称。
- `--final-snapshot-identifier` – 最终备份的名称。

以下代码在删除复制组 myReplGroup 时创建最终备份 bkup-20150515-final。

对于 Linux、macOS 或 Unix：

```
aws elasticache delete-replication-group \
        --replication-group-id myReplGroup \
        --final-snapshot-identifier bkup-20150515-final
```

对于 Windows：

```
aws elasticache delete-replication-group ^
        --replication-group-id myReplGroup ^
        --final-snapshot-identifier bkup-20150515-final
```

有关更多信息，请参阅 AWS CLI 命令参考中的 delete-replication-group。

描述备份

以下过程演示如何显示备份列表。如果需要，您还可以查看特定备份的详细信息。

描述备份（控制台）

要显示备份，请使用 AWS Management Console

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 从导航窗格中，选择 Backups（备份）。
3. 要查看特定备份的详细信息，请选择备份名称左侧的复选框。

描述无服务器备份（AWS CLI）

要显示无服务器备份列表以及（可选）特定备份的详细信息，请使用 `describe-serverless-cache-snapshots` CLI 操作。

示例

以下操作使用参数 `--max-records` 列出与您的账户关联的最多 20 个备份。忽略参数 `--max-records` 最多可列出 50 个备份。

```
aws elasticache describe-serverless-cache-snapshots --max-records 20
```

以下操作使用参数 `--serverless-cache-name`，以仅列出与缓存 `my-cache` 关联的备份。

```
aws elasticache describe-serverless-cache-snapshots --serverless-cache-name my-cache
```

以下操作使用参数 `--serverless-cache-snapshot-name` 显示备份 `my-backup` 的详细信息。

```
aws elasticache describe-serverless-cache-snapshots --serverless-cache-snapshot-
name my-backup
```

有关更多信息，请参阅《命令参考》中的 desc ribe-serverless-cache-snaps hots。 AWS CLI

描述自行设计集群的备份（AWS CLI）

要显示自行设计集群的备份列表以及（可选）特定备份的详细信息，请使用 `describe-snapshots` CLI 操作。

示例

以下操作使用参数 `--max-records` 列出与您的账户关联的最多 20 个备份。忽略参数 `--max-records` 最多可列出 50 个备份。

```
aws elasticache describe-snapshots --max-records 20
```

以下操作使用参数 `--cache-cluster-id` 仅列出与集群 `my-cluster` 关联的备份。

```
aws elasticache describe-snapshots --cache-cluster-id my-cluster
```

以下操作使用参数 `--snapshot-name` 显示备份 `my-backup` 的详细信息。

```
aws elasticache describe-snapshots --snapshot-name my-backup
```

有关更多信息，请参阅《命令参考》中的 d escribe-snapsh ots AWS CLI 。

复制备份

您可以创建任意备份的副本，无论它是自动还是手动创建的。您也可以导出备份，这样您就可以从外部访问它 ElastiCache。有关导出备份的指导，请参阅导出备份。

以下步骤演示如何复制备份。

复制备份（控制台）

复制备份（控制台）

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 要查看您的备份列表，请从左侧导航窗格中，选择 Backups。
3. 从备份列表中，选择要复制的备份名称左侧的复选框。
4. 选择操作和复制。
5. 在 New backup name 框中键入新备份的名称。
6. 选择复制。

复制无服务器备份（AWS CLI）

要复制无服务器缓存的备份，请使用 `copy-serverless-cache-snapshot` 操作。

参数

- `--source-serverless-cache-snapshot-name` – 要复制的备份的名称。
- `--target-serverless-cache-snapshot-name` – 备份副本的名称。

以下示例复制自动备份。

对于 Linux、macOS 或 Unix：

```
aws elasticache copy-serverless-cache-snapshot \
    --source-serverless-cache-snapshot-name automatic.my-cache-2023-11-27-03-15 \
    --target-serverless-cache-snapshot-name my-backup-copy
```

对于 Windows：

```
aws elasticache copy-serverless-cache-snapshot ^
    --source-serverless-cache-snapshot-name automatic.my-cache-2023-11-27-03-15 ^
    --target-serverless-cache-snapshot-name my-backup-copy
```

有关更多信息，请参阅AWS CLI中的 copy-serverless-cache-snapshot。

复制自行设计集群的备份（AWS CLI）

要复制自行设计集群的备份，请使用 copy-snapshot 操作。

参数

- --source-snapshot-name – 要复制的备份的名称。
- --target-snapshot-name – 备份副本的名称。
- --target-bucket – 为导出备份预留。在复制备份时，请勿使用此参数。仅适用于 Redis 无服务器缓存和 Redis 自行设计的集群。有关更多信息，请参阅 导出备份。

以下示例复制自动备份。

对于 Linux、macOS 或 Unix：

```
aws elasticache copy-snapshot  \
    --source-snapshot-name automatic.my-redis-primary-2014-03-27-03-15 \
    --target-snapshot-name my-backup-copy
```

对于 Windows：

```
aws elasticache copy-snapshot ^
    --source-snapshot-name automatic.my-redis-primary-2014-03-27-03-15 ^
    --target-snapshot-name my-backup-copy
```

有关更多信息，请参阅AWS CLI中的 copy-snapshot。

导出备份

亚马逊 ElastiCache 支持将您的 Fo ElastiCache r Redis 备份导出到亚马逊简单存储服务 (Amazon S3) 存储桶，这样您就可以从外部访问该存储桶。 ElastiCache您可以使用 ElastiCache 控制台 AWS CLI、或 ElastiCache API 导出备份。

如果您需要在其他 AWS 区域启动集群，则导出备份会很有帮助。您可以将数据导出到一个 AWS 区域，将.rdb 文件复制到新 AWS 区域，然后使用该.rdb 文件为新缓存做种子，而不必等待新集群通过使用进行填充。有关为新集群做种的信息，请参阅 使用外部创建的备份为新的自行设计的集群制作种子。您可能想要导出缓存数据的另一个原因是使用.rdb 文件进行离线处理。

> **Important**
>
> - ElastiCache 备份和您要将其复制到的 Amazon S3 存储桶必须位于同一 AWS 区域。
>
> 尽管复制到 Amazon S3 存储桶的备份已加密，但我们强烈建议您不要将要存储备份的 Amazon S3 存储桶的访问权限授予他人。
> - 使用数据分层功能的集群不支持将备份导出到 Amazon S3。有关更多信息，请参阅 数据分层。
> - 导出备份适用于 Redis 自己设计的集群、无服务器 Redis 和无服务器 Memcached。对于自行设计的 Memcached 集群，无法导出备份。

在将备份导出到 Amazon S3 存储桶之前，您必须将 Amazon S3 存储桶与备份位于同一 AWS 区域。授予对存储桶的 ElastiCache 访问权限。前两个步骤向您演示了如何执行此操作。

步骤 1：创建 Amazon S3 存储桶

以下步骤使用 Amazon S3 控制台创建用于导出和存储 ElastiCache 备份的 Amazon S3 存储桶。

创建 Amazon S3 存储桶

1. 登录 AWS Management Console 并打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
2. 选择创建存储桶。
3. 在 Create a Bucket - Select a Bucket Name and Region 中，执行以下操作：

    a. 在 Bucket Name（存储桶名称）中键入 Amazon S3 存储桶的名称。

Amazon S3 存储桶的名称必须符合 DNS 标准。否则，将 ElastiCache 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）。

b. 从区域列表中，为您的 Amazon S3 存储桶选择一个 AWS 区域。此 AWS 区域必须与您要导出的 ElastiCache 备份位于同一 AWS 区域。

c. 选择创建。

有关创建 Amazon S3 存储桶的更多信息，请参阅 Amazon Simple Storage Service 用户指南中的创建存储桶。

第 2 步：授予对您的 Amazon S3 存储桶的 ElastiCache 访问权限

ElastiCache 为了能够将快照复制到 Amazon S3 存储桶，您必须更新存储桶策略以授予对存储桶的 ElastiCache 访问权限。

> **Warning**
>
> 虽然复制到 Amazon S3 存储桶的备份已加密，但是对您的 Amazon S3 存储桶拥有访问权限的任何人都可以访问您的数据。因此，我们强烈建议您设置 IAM 策略来防止未经授权访问此 Amazon S3 存储桶。有关更多信息，请参阅《Amazon S3 用户指南》中的管理访问权限。

要创建 Amazon S3 存储桶的适当权限，请执行以下步骤。

授予 ElastiCache 对 S3 存储桶的访问权限

1. 登录 AWS Management Console 并打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
2. 选择要将备份复制到其中的 Amazon S3 存储桶的名称。这应该是您在步骤 1：创建 Amazon S3 存储桶中创建的 S3 存储桶。

3. 选择 Permissions（权限）选项卡，然后在 Permissions（权限）下，选择 Access control list（ACL）[访问控制列表（ACL）]，再选择 Edit（编辑）。
4. 使用以下选项为被授权者添加规范 ID `540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353`：
   - Objects: List, Write（对象：列出、写入）
   - Bucket ACL: Read, Write（存储桶 ACL：读取、写入）

   > **Note**
   > - 对于太平洋夏令时 GovCloud 区域，规范 ID 为。`40fa568277ad703bd160f66ae4f83fc9dfdfd06c2f1b5060ca22442ac3ef8be6`
   > - 对于俄勒冈州立大学 GovCloud 区域，规范 ID 为。`c54286759d2a83da9c480405349819c993557275cf37d820d514b42da6893f5c`

5. 选择保存。

步骤 3：导出备 ElastiCache 份

现在，您已经创建了 S3 存储桶并授予了访问该存储桶的 ElastiCache 权限。接下来，您可以使用 ElastiCache 控制台、 AWS CLI 或 ElastiCache API 将快照导出到控制台。以下示例假设调用方的 IAM 身份拥有以下附加的 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:::*"
 }]
}
```

对于选择加入型区域，以下是 S3 存储桶的更新后策略具体形式的示例。[此示例使用亚太地区（香港）区域。]

```
{
    "Version": "2012-10-17",
    "Id": "Policy15397346",
    "Statement": [
        {
            "Sid": "Stmt15399483",
            "Effect": "Allow",
            "Principal": {
                "Service": "elasticache.amazonaws.com"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::hkg-elasticache-backup",
                "arn:aws:s3:::hkg-elasticache-backup/*"
            ]
        },
        {
            "Sid": "Stmt15399484",
            "Effect": "Allow",

            "Principal": {
                "Service": "ap-east-1.elasticache-snapshot.amazonaws.com"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::hkg-elasticache-backup",
                "arn:aws:s3:::hkg-elasticache-backup/*"
            ]
        }
    ]
}
```

导出备 ElastiCache 份（控制台）

以下步骤使用 ElastiCache 控制台将备份导出到 Amazon S3 存储桶，以便您可以从外部访问该存储桶 ElastiCache。Amazon S3 存储桶必须与 ElastiCache 备份位于同一 AWS 区域。

将 ElastiCache 备份导出到 Amazon S3 存储桶

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 要查看您的备份列表，请从左侧导航窗格中，选择 Backups。
3. 从备份列表中，选择要导出的备份名称左侧的复选框。
4. 选择复制。
5. 在 Create a Copy of the Backup?（创建备份副本?）中，执行以下操作：

   a. 在 New backup name 框中键入新备份的名称。

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

      ElastiCache 将实例标识符和.rdb添加到您在此处输入的值中。例如，如果您输入 my-exported-backup，则 ElastiCache 创建 my-exported-backup-0001.rdb。

   b. 从 Target S3 Location（目标 S3 位置）列表中，选择要将备份复制到其中的 Amazon S3 存储桶（您在 步骤 1：创建 Amazon S3 存储桶 中创建的存储桶）的名称。

      目标 S3 位置必须是备份 AWS 区域中具有以下权限的 Amazon S3 存储桶，导出过程才能成功。

      - 对象访问 – Read（读取）和 Write（写入）。
      - 权限访问 – Read（读取）。

      有关更多信息，请参阅 第 2 步：授予对您的 Amazon S3 存储桶的 ElastiCache 访问权限。

   c. 选择复制。

> **Note**
>
> 如果您的 S3 存储桶没有 ElastiCache 向其导出备份所需的权限，则会收到以下错误消息之一。返回到第 2 步：授予对您的 Amazon S3 存储桶的 ElastiCache 访问权限，添加指定权限并重试导出备份的操作。
>
> - ElastiCache 尚未被授予 S3 存储桶上的 %s 读取权限。
>
>   解决方案：在存储桶上添加 Read 权限。
>
> - ElastiCache 尚未被授予 S3 存储桶上的 %s 写入权限。

解决方案：在存储桶上添加 Write 权限。

- ElastiCache 尚未被授予 S3 存储桶上的 READ_ACP 权限 %s。

  解决方案：为存储桶的权限访问添加 Read。

如果您想将备份复制到其他 AWS 区域，请使用 Amazon S3 将其复制。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的复制对象。

导出 ElastiCache 无服务器备份 ()AWS CLI

导出无服务器缓存的备份

使用带有以下参数的 `export-serverless-cache-snapshot` CLI 操作将备份导出到 Amazon S3 存储桶：

参数

- `--serverless-cache-snapshot-name` – 要复制的备份的名称。
- `--s3-bucket-name` – 您要将备份导出到其中的 Amazon S3 存储桶的名称。在指定存储桶中生成备份的副本。

  `--s3-bucket-name`必须是备份 AWS 区域中具有以下权限的 Amazon S3 存储桶，导出过程才能成功。
  - 对象访问 – Read（读取）和 Write（写入）。
  - 权限访问 – Read（读取）。

以下操作将备份复制到 my-s3-bucket。

对于 Linux、macOS 或 Unix：

```
aws elasticache export-serverless-cache-snapshot \
    --serverless-cache-snapshot-name automatic.my-redis-2023-11-27 \
    --s3-bucket-name my-s3-bucket
```

对于 Windows：

```
aws elasticache export-serverless-cache-snapshot ^
```

```
    --serverless-cache-snapshot-name automatic.my-redis-2023-11-27 ^
    --s3-bucket-name my-s3-bucket
```

导出自己设计的 ElastiCache 集群备份 ()AWS CLI

导出自行设计的集群的备份

使用带有以下参数的 copy-snapshot CLI 操作将备份导出到 Amazon S3 存储桶：

参数

- --source-snapshot-name – 要复制的备份的名称。
- --target-snapshot-name – 备份副本的名称。

  名称必须在 1 到 1000 个字符之间，并能够以 UTF-8 编码。

  ElastiCache 将实例标识符和.rdb添加到您在此处输入的值中。例如，如果您输入 my-exported-backup，则 ElastiCache 创建 my-exported-backup-0001.rdb。
- --target-bucket – 您要将备份导出到其中的 Amazon S3 存储桶的名称。在指定存储桶中生成备份的副本。

  --target-bucket必须是备份 AWS 区域中具有以下权限的 Amazon S3 存储桶，导出过程才能成功。
  - 对象访问 – Read（读取）和 Write（写入）。
  - 权限访问 – Read（读取）。

  有关更多信息，请参阅 第 2 步：授予对您的 Amazon S3 存储桶的 ElastiCache 访问权限。

以下操作将备份复制到 my-s3-bucket。

对于 Linux、macOS 或 Unix：

```
aws elasticache copy-snapshot \
    --source-snapshot-name automatic.my-redis-primary-2016-06-27-03-15 \
    --target-snapshot-name my-exported-backup \
    --target-bucket my-s3-bucket
```

对于 Windows：

```
aws elasticache copy-snapshot ^
```

```
    --source-snapshot-name automatic.my-redis-primary-2016-06-27-03-15 ^
    --target-snapshot-name my-exported-backup ^
    --target-bucket my-s3-bucket
```

从备份还原到新缓存

您可以将现有备份还原到新的无服务器缓存或自行设计的集群中。

将备份还原到无服务器缓存（控制台）

> **Note**
>
> ElastiCache Serverless 支持与 Redis 版本兼容的 RDB 文件，介于 5.0 和最新可用版本之间。

将备份还原到无服务器缓存（控制台）

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 从导航窗格中，选择 Backups（备份）。
3. 在备份列表中，选中要还原的备份名称左侧的框。
4. 选择操作，然后选择还原。
5. 输入新无服务器缓存的名称和可选描述。
6. 单击创建以创建新的缓存并从备份中导入数据。

将备份还原到自行设计的集群（控制台）

将备份还原到自行设计的集群（控制台）

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 从导航窗格中，选择 Backups（备份）。
3. 在备份列表中，选择您要从中进行还原的备份名称左侧的复选框。
4. 选择操作，然后选择还原。
5. 选择设计自己的缓存并自定义集群设置，例如节点类型、大小、分片数量、副本、可用区放置和安全设置。
6. 单击创建以创建新的自行设计的缓存，并从备份中导入数据。

将备份还原到无服务器缓存（AWS CLI）

> **Note**
>
> ElastiCache Serverless 支持与 Redis 版本兼容的 RDB 文件，介于 5.0 和最新可用版本之间。

将备份还原到新的无服务器缓存（AWS CLI）

以下 AWS CLI 示例使用备份创建新缓存`create-serverless-cache`并从备份中导入数据。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-serverless-cache \
    --serverless-cache-name CacheName \
    --engine redis
    --snapshot-arns-to-restore Snapshot-ARN
```

对于 Windows：

```
aws elasticache create-serverless-cache ^
    --serverless-cache-name CacheName ^
    --engine redis ^
    --snapshot-arns-to-restore Snapshot-ARN
```

对于 Windows：

将备份还原到自行设计的集群（AWS CLI）

将备份还原到自行设计的集群（AWS CLI）

您可以恢复 Redis 无服务器缓存备份，也可以恢复 Redis 自行设计的集群。

您可以通过两种方式恢复 Redis 无服务器缓存备份。

- 您可以使用操作恢复到单节点 Redis（已禁用集群模式）集群。 AWS CLI create-cache-cluster
- 您可以还原到具有只读副本的 Redis 集群（复制组）。为此，您可以使用 Redis（已禁用集群模式）或 Redis（已启用集群模式）进行操作。 AWS CLI create-replication-group在这种情况下，使用 Redis .rdb 文件为还原设定种子。有关为自行设计的集群制作种子的更多信息，请参阅 使用外部创建的备份为新的自行设计的集群制作种子。

您可以通过两种方式还原 Redis（已禁用集群模式）备份。

- 您可以使用操作恢复到单节点 Redis（已禁用集群模式）集群。 AWS CLI create-cache-cluster
- 您可以还原到具有只读副本的 Redis 集群（复制组）。为此，您可以使用 Redis（已禁用集群模式）或 Redis（已启用集群模式）进行操作。 AWS CLI create-replication-group在这种情况下，使用 Redis .rdb 文件为还原设定种子。有关为自行设计的集群制作种子的更多信息，请参阅 使用外部创建的备份为新的自行设计的集群制作种子。

使用 `create-cache-cluster` 或 `create-replication-group` 操作时，请确保包括参数 `--snapshot-name` 或 `--snapshot-arn`，以使用来自备份的数据为新集群或复制组制作种子。

删除备份

自动备份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备份。如果您删除某个复制组，则也会删除该组中集群的所有自动备份。

ElastiCache 提供了删除 API 操作，允许您随时删除备份，无论备份是自动创建还是手动创建。由于手动备份没有保留期限，所以手动删除是移除备份的唯一方法。

您可以使用 ElastiCache 控制台 AWS CLI、或 ElastiCache API 删除备份。

删除备份（控制台）

以下过程使用 ElastiCache 控制台删除备份。

删除备份

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格中，选择备份。

   此时会显示“Backups”屏幕，其中包含您的备份的列表。
3. 选择要删除的备份名称左侧的复选框。
4. 选择删除。
5. 如果要删除此备份，在 Delete Backup 确认屏幕中选择 Delete。状态将变为正在删除。

删除无服务器备份（AWS CLI）

使用带有以下参数的删除快照 AWS CLI 操作来删除无服务器备份。

- `--serverless-cache-snapshot-name` – 要删除的备份的名称。

以下代码删除备份 myBackup。

```
aws elasticache delete-serverless-cache-snapshot --serverless-cache-snapshot-
name myBackup
```

有关更多信息，请参阅《AWS CLI Command Reference》中的 delete-serverless-cache-snapshot。

删除自行设计集群的备份（AWS CLI）

使用带有以下参数的删除快照 AWS CLI 操作来删除自己设计的集群备份。

- `--snapshot-name` – 要删除的备份的名称。

以下代码删除备份 myBackup。

```
aws elasticache delete-snapshot --snapshot-name myBackup
```

有关更多信息，请参阅 AWS CLI 命令参考中的 delete-snapshot。

标记备份

您可以标签形式将自己的元数据分配给各个备份。标签可让您按各种标准（例如用途、拥有者或环境）对备份进行分类。这在您具有相同类型的很多资源时会很有用 – 您可以根据分配给特定资源的标签快速识别该资源。有关更多信息，请参阅 您可以为之添加标签的资源。

成本分配标签是一种通过标签值对发票上的费用进行分组来跟踪多项 AWS 服务成本的方法。要了解有关成本分配标签的更多信息，请参阅使用成本分配标签。

使用 ElastiCache 控制台 AWS CLI、或 ElastiCache API，您可以在备份上添加、列出、修改、移除或复制成本分配标签。有关更多信息，请参阅 使用成本分配标签监控成本。

使用外部创建的备份为新的自行设计的集群制作种子

创建新的 Redis 自行设计的集群时，可以使用 Redis .rdb 备份文件中的数据为它制作种子。如果您当前正在管理外部的 Redis 实例， ElastiCache 并希望使用现有的 Redis 数据填充新的 Redis 自主设计的集群，则 ElastiCache 为集群播种很有用。

要从在 Ama ElastiCache zon 中创建的 Redis 备份中为自己设计的新 Redis 集群播种，请参阅。从备份还原到新缓存

使用 Redis .rdb 文件为新的 Redis 自行设计集群制作种子时，您可以执行以下操作：

- 从未分区的集群升级到运行 Redis 版本 3.2.4 的 Redis（已启用集群模式）自行设计的集群。
- 指定新的自行设计的集群中的分片（在 API 和 CLI 中称为节点组）数量。此数量可以与用于创建备份文件的自行设计的集群中的分片数量不同。
- 为新的自行设计的集群指定不同的节点类型 – 大于或小于创建备份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。有关更多信息，请参阅 确保具有用于创建 Redis 快照的足够内存。
- 以不同于创建备份文件时所用集群中的方法将您的键分发到新 Redis（已启用集群模式）集群的槽中。

> **Note**
>
> 无法从 Redis（已启用集群模式）集群中创建的 .rdb 文件为 Redis（已禁用集群模式）集群设定种子。

> **Important**
>
> - 您必须确保 Redis 备份数据不超过节点的资源容量。例如，您无法将具有 5 GB Redis 数据的 .rdb 文件上传到具有 2.9 GB 内存的 cache.m3.medium 节点。
>
> 如果备份太大，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。
>
> 有关节点类型和规格的完整列表，请参阅特定于 Redis 节点类型的参数和 Amazon ElastiCache 产品功能和详情。

- 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。

接下来，您可以找到一些主题，这些主题将引导您完成将 Redis 集群从外部迁移到 ElastiCache ElastiCache 适用于 Redis 的 Redis 集群。

迁移到 f ElastiCache or Redis

- 步骤 1：创建 Redis 备份
- 步骤 2：创建 Amazon S3 存储桶和文件夹
- 步骤 3：将备份上传到 Amazon S3
- 步骤 4：授予对.rdb 文件的 ElastiCache 读取权限

步骤 1：创建 Redis 备份

创建 Redis 备份来 ElastiCache 为你的 Redis 实例做种子

1. 连接到您的现有 Redis 实例。
2. 运行 Redis BGSAVE 或 SAVE 操作以创建备份。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。

有关创建备份的其他信息，请参阅 Redis 网站上的 Redis 持久化。

步骤 2：创建 Amazon S3 存储桶和文件夹

创建备份文件后，您需要将其上传到 Amazon S3 存储桶中的文件夹。要执行该操作，您必须先拥有 Amazon S3 存储桶以及该存储桶中的文件夹。如果您已有 Amazon S3 存储桶和文件夹并具备相应权限，则可以跳到 步骤 3：将备份上传到 Amazon S3。

创建 Amazon S3 存储桶

1. 登录 AWS Management Console 并打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。

2. 按照 Amazon Simple Storage Service 用户指南中的创建存储桶的说明，创建 Amazon S3 存储桶。

   Amazon S3 存储桶的名称必须符合 DNS 标准。否则，将 ElastiCache 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）。

   您必须在与新 ElastiCache 的 Redis 集群相同的 AWS 区域中创建 Amazon S3 存储桶。这种方法可确保从 Amazon S3 ElastiCache 读取您的.rdb 文件时达到最高的数据传输速度。

   > **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 Redis 集群设定种子。

向 Amazon S3 存储桶添加文件夹

1. 登录 AWS Management Console 并打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
2. 选择将 .rdb 文件上传到的存储桶的名称。
3. 请选择 Create folder（创建文件夹）。
4. 输入新文件夹的名称。
5. 选择保存。

   记录存储桶名称和文件夹名称。

步骤 3：将备份上传到 Amazon S3

现在，上传您在步骤 1：创建 Redis 备份中创建的 .rdb 文件。将其上传到您在 步骤 2：创建 Amazon S3 存储桶和文件夹 中创建的 Amazon S3 存储桶和文件夹。有关此任务的更多信息，请参阅将对象添加到存储桶。在步骤 2 和 3 之间，选择您创建的文件夹的名称。

将 .rdb 文件上传到 Amazon S3 文件夹

1. 登录 AWS Management Console 并打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
2. 选择您在步骤 2 中创建的 Amazon S3 存储桶的名称。
3. 选择您在步骤 2 中创建的文件夹的名称。
4. 选择上传。
5. 选择 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 Upload 对话框中列出了正确的文件，然后选择 Upload。

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`。使用此备份中的数据为新集群做种时需要此路径。

有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的存储桶限制。

步骤 4：授予对.rdb 文件的 ElastiCache 读取权限

现在，授予对.rdb 备份文件的 ElastiCache 读取权限。您可以通过不同的方式授予对备份文件的 ElastiCache 访问权限，具体取决于您的存储桶位于默认 AWS 区域还是选择加入 AWS 区域。

AWS 2019 年 3 月 20 日之前推出的区域默认处于启用状态。您可以立即开始在这些 AWS 地区工作。2019 年 3 月 20 日之后推出的区域默认处于禁用状态，如亚太地区（香港）和中东（巴林）。您必须按照 AWS 一般参考中的管理 AWS 区域所述，先启用或选择加入这些区域，然后才能使用它们。

根据您所在的 AWS 地区选择您的方法：

- 对于默认区域，请使用 授予对默认区域中.rdb 文件的 ElastiCache 读取权限 中的过程。
- 对于选择加入的区域，请使用 在可 ElastiCache 选区域中授予对.rdb 文件的读取权限 中的过程。

授予对默认区域中.rdb 文件的 ElastiCache 读取权限

AWS 2019 年 3 月 20 日之前推出的区域默认处于启用状态。您可以立即开始在这些 AWS 地区工作。2019 年 3 月 20 日之后推出的区域默认处于禁用状态，如亚太地区（香港）和中东（巴林）。您必须按照 AWS 一般参考中的管理 AWS 区域所述，先启用或选择加入这些区域，然后才能使用它们。

授予对默认启用的 AWS 区域中备份文件的 ElastiCache 读取权限

1. 登录 AWS Management Console 并打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
2. 选择包含您 .rdb 文件的 S3 存储桶的名称。
3. 选择包含 .rdb 文件的文件夹的名称。
4. 选择 .rdb 备份文件的名称。所选文件的名称将显示在页面顶部的选项卡上方。
5. 选择权限。
6. 如果 aws-scs-s3-readonly 或以下列表中的规范 ID 之一未作为用户列出，请执行以下操作：

   a. 在 “其他 AWS 账户的访问权限” 下，选择添加被授权者。

   b. 在框中，添加该 AWS 地区的规范 ID，如下所示：

      - AWS GovCloud （美国西部）区域：

        ```
        40fa568277ad703bd160f66ae4f83fc9dfdfd06c2f1b5060ca22442ac3ef8be6
        ```

        > **Important**
        >
        > 备份必须位于中的 S3 存储桶中 AWS GovCloud (US) ，您才能将其下载到中的 AWS GovCloud (US) Redis 集群。

      - AWS 默认情况下启用的区域：

        ```
        540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353
        ```

   c. 通过为以下对象选择 Yes（是）对存储桶设置权限：

      - List/write object（列出/写入对象）
      - Read/write object ACL permissions（读/写对象 ACL 权限）

   d. 选择保存。

7. 选择 Overview（概述），然后选择 Download（下载）。

在可 ElastiCache 选区域中授予对.rdb 文件的读取权限

AWS 2019 年 3 月 20 日之前推出的区域默认处于启用状态。您可以立即开始在这些 AWS 地区工作。2019 年 3 月 20 日之后推出的区域默认处于禁用状态，如亚太地区（香港）和中东（巴林）。您必须按照 AWS 一般参考中的管理 AWS 区域所述，先启用或选择加入这些区域，然后才能使用它们。

现在，授予对.rdb 备份文件的 ElastiCache 读取权限。

授予对备份文件的 ElastiCache 读取权限

1. 登录 AWS Management Console 并打开 Amazon S3 控制台，网址为 https://console.aws.amazon.com/s3/。
2. 选择包含您 .rdb 文件的 S3 存储桶的名称。
3. 选择包含 .rdb 文件的文件夹的名称。
4. 选择 .rdb 备份文件的名称。所选文件的名称将显示在页面顶部的选项卡上方。
5. 选择权限选项卡。
6. 在 Permissions（权限）下，选择 Bucket policy（存储桶策略），然后选择 Edit（编辑）。
7. 更新策略以授予执行操作 ElastiCache 所需的权限：

   - 将 `[ "Service" : "region-full-name.elasticache-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": "ap-east-1.elasticache-snapshot.amazonaws.com"
            },
            "Action": [
                "s3:GetObject",
                "s3:ListBucket",
                "s3:GetBucketAcl"
            ],
            "Resource": [
                "arn:aws:s3:::example-bucket",
                "arn:aws:s3:::example-bucket/backup1.rdb",
                "arn:aws:s3:::example-bucket/backup2.rdb"
            ]
        }
    ]
}
```

8. 选择保存更改。

步骤 5：使用.rdb 文件数据为 ElastiCache 集群播种

现在，您可以创建 ElastiCache 集群并使用.rdb 文件中的数据为其做种子了。要创建集群，请按照创建集群或从头创建 Redis 复制组中的说明操作。请确保选择 Redis 作为集群引擎。

您用来判断 ElastiCache 在哪里可以找到您上传到 Amazon S3 的 Redis 备份的方法取决于您创建集群时使用的方法：

使用.rd ElastiCache b 文件数据为 Redis 集群或复制组做种子

- 使用控制 ElastiCache 台

  选择集群设置时，请选择 Restore from backups（从备份中恢复）作为集群创建方法，然后在 Backup source（备份源）部分中选择 Other backups（其他备份）作为您的 Source（源）。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 AWS CLI

  如果您使用 `create-cache-cluster` 或 `create-replication-group` 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 中的备份文件。

- 使用 ElastiCache API

  如果您使用CreateCacheCluster或 CreateReplicationGroup ElastiCache API 操作，请使用参数SnapshotArns为每个.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 中的备份文件。

> **Important**
>
> 在为 Redis（已启用集群模式）集群设定种子时，您必须在新集群或复制组中配置每个节点组（分片）。为此，请使用参数 --node-group-configuration（API：NodeGroupConfiguration）。有关更多信息，请参阅下列内容：
>
> - CLI：参考资料中的创建复制组 AWS CLI
> - API：AP ElastiCache I 参考中的CreateReplication群组

在创建集群的过程中，Redis 备份中的数据将写入集群。您可以通过查看 ElastiCache 事件消息来监控进度。为此，请访问 ElastiCache 控制台并选择 “缓存事件”。您也可以使用 AWS ElastiCache 命令行界面或 ElastiCache API 来获取事件消息。有关更多信息，请参阅 查看 ElastiCache 事件。

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

引擎版本和升级

ElastiCache for Redis 版本由包含 MAJOR 和 MINOR 组件的语义版本标识。例如，在 Redis 6.2 中，主要版本为 6，次要版本为 2。在操作自行设计的集群时，ElastiCache for Redis 还会公开 PATCH 组件，例如 Redis 6.2.1 的补丁版本为 1。

MAJOR 主要版本针对 API 不兼容的更改，而 MINOR 版本针对以向后兼容的方式添加的新功能。PATCH 版本针对向后兼容的错误修复和非功能性更改。

ElastiCache 无服务器的版本管理

ElastiCache 无服务器会自动将最新的 MINOR 和 PATCH 软件版本应用到您的缓存，而不会对您的应用程序造成任何影响或导致停机。在您的末端不需要执行任何操作。

当新的 MAJOR 版本可用时，ElastiCache 无服务器将在控制台中向您发送通知，在 EventBridge 中向您发送事件。您可以选择使用控制台、CLI 或 API 修改缓存并选择最新的引擎版本，将缓存升级到最新的主要版本。

自行设计的 ElastiCache 集群的版本管理

使用自行设计的 ElastiCache 集群时，您可以控制何时将缓存群集上所用的软件升级到 ElastiCache 支持的新版本。您可以控制何时将缓存升级到最新的 MAJOR、MINOR 和 PATCH 版本。可以通过修改集群或复制组并指定新的引擎版本，对您的集群或复制组启动引擎版本升级。

您可以控制是否及何时将支持缓存群集、符合协议标准的软件升级到 ElastiCache 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。

因为版本升级可能会涉及到某些兼容性风险，因此版本升级不会自动发生。您必须启动它们。

可以通过修改集群或复制组并指定新的引擎版本，对您的集群或复制组启动引擎版本升级。有关更多信息，请参阅下列内容：

- 修改集群
- 修改复制组

使用自行设计集群时的升级注意事项

> **Note**
>
> 以下注意事项仅在升级自行设计集群时适用。这些事项不适用于 ElastiCache 无服务器。

在升级自行设计集群时，请注意以下事项

- 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，如果发生系统或缓存软件中存在严重安全漏洞这种不太可能发生的情况，ElastiCache 保留代表您修补集群的权利。
- 从 Redis 6.0 开始，ElastiCache for Redis 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。
- 从 Redis 引擎版本 5.0.6 开始，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。
- 您还可以使用 5.0.6 以下的版本升级您的 ElastiCache 集群。所涉及的过程相同，但在 DNS 传播期间可能会导致更长的故障转移时间（30 秒 - 1 分钟）。
- 从 Redis 7 开始，ElastiCache for Redis 支持在 Redis（已禁用集群模式）和 Redis（已启用集群模式）之间切换。
- Amazon ElastiCache for Redis 引擎升级旨在尽最大努力保留您的现有数据，并需要 Redis 复制成功。
- 升级引擎时，ElastiCache for Redis 将终止现有的客户端连接。为了最大限度地减少引擎升级期间的停机时间，我们建议您实施关于 Redis 客户端的最佳实践，包括错误重试和指数回退，以及关于维护期间最大限度减少停机时间的最佳实践。
- 升级引擎时，无法从 Redis（已禁用集群模式）直接升级到 Redis（已启用集群模式）。以下过程演示如何从 Redis（已禁用集群模式）升级到 Redis（已启用集群模式）。

  从 Redis（已禁用集群模式）升级到 Redis（已启用集群模式）引擎版本

  1. 备份 Redis（已禁用集群模式）集群或复制组。有关更多信息，请参阅进行手动备份。
  2. 使用备份创建具有一个分区（节点组）的 Redis（已启用集群模式）集群并为其设定种子。在创建集群或复制组时，指定新的引擎版本并启用集群模式。有关更多信息，请参阅使用外部创建的备份为新的自行设计的集群制作种子。
  3. 删除旧 Redis（已禁用集群模式）集群或复制组。有关更多信息，请参阅 删除集群或 删除复制组。

4. 将新的 Redis（已启用集群模式）集群或复制组扩展到所需的分区（节点组）数。有关更多信息，请参阅 扩展 Redis（启用集群模式）集群

- 升级主要引擎版本（例如从 5.0.6 升级到 6.0）时，还需要选择一个与新引擎版本兼容的新参数组。
- 对于单个 Redis 集群以及禁用了多可用区的集群，建议有足够的内存可供 Redis 使用，如确保具有用于创建 Redis 快照的足够内存中所述。在这些情况下，主项在升级过程中不可用于处理服务请求。
- 对于启用了多可用区的 Redis 集群，还建议在传入的写流量较低期间安排引擎升级。升级到 Redis 5.0.6 或更高版本时，主集群在升级过程中仍可用于服务请求。

  将处理和修补带多个分片的集群和复制组，如下所示：

  - 将并行处理所有分片。在任何时候，仅在分片上执行一次升级操作。
  - 在每个分片中，在处理主副本之前，会先处理所有其他副本。如果一个分片中的副本较少，则可能会在处理完其他分片中的副本之前处理该分片中的主副本。
  - 在所有分片中，主节点都是按顺序处理的。一次只升级一个主节点。
- 如果已对当前集群或复制组启用了加密，则无法升级到不支持加密的引擎版本，例如，从 3.2.6 升级到 3.2.10。

如何升级引擎版本

使用 ElastiCache 控制台、AWS CLI 或 ElastiCache API 修改集群或复制组并指定较新的引擎版本，从而启动集群或复制组的版本升级。有关更多信息，请参阅以下主题。

如何修改集群和复制组	
集群	**复制组**
使用 AWS Management Console	使用 AWS Management Console
使用 AWS CLI	使用 AWS CLI
使用 ElastiCache API	使用 ElastiCache API

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
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

解决阻止的 Redis 引擎升级

- 请执行以下操作之一：
  - 通过清除 Apply immediately (立即应用) 复选框，将 Redis 引擎升级操作安排在下一维护时段内。

    对于 CLI，请使用 `--no-apply-immediately`。对于 API，请使用 `ApplyImmediately=false`。
  - 等到下一维护时段（或之后）再执行 Redis 引擎升级操作。
  - 将 Redis 纵向扩展操作添加到此选中了 Apply Immediately (立即应用) 复选框的集群修改中。

    对于 CLI，请使用 `--apply-immediately`。对于 API，请使用 `ApplyImmediately=true`。

    此方法将立即执行引擎升级，从而有效地在下一维护时段内取消该操作。

支持的 ElastiCache for Redis 版本

ElastiCache 无服务器缓存支持以下 Redis 版本：

- ElastiCache for Redis 版本 7.1（加强版）

自行设计的 ElastiCache 集群支持以下 Redis 版本：

- ElastiCache for Redis 版本 7.1（加强版）
- ElastiCache for Redis 版本 7.0（加强版）
- ElastiCache for Redis 版本 6.2（加强版）
- ElastiCache for Redis 版本 6.0（加强版）
- ElastiCache for Redis 版本 5.0.6（加强版）
- ElastiCache for Redis 版本 5.0.5（已弃用，使用 5.0.6 版本）
- ElastiCache for Redis 版本 5.0.4（已弃用，使用 5.0.6 版本）
- ElastiCache for Redis 版本 5.0.3（已弃用，使用 5.0.6 版本）
- ElastiCache for Redis 版本 5.0.0（已弃用，使用 5.0.6 版本）
- ElastiCache for Redis 版本 4.0.10（加强版）
- 已经终止生命期周期（EOL）版本（3.x）
- 已经终止生命期周期（EOL）版本（2.x）

ElastiCache for Redis 版本 7.1（加强版）

此版本包含性能改进，使工作负载能够实现更高的吞吐量和更低的操作延迟。ElastiCache 7.1 引入了两项主要增强功能：

我们扩展了增强型 I/O 线程功能，使其还可以处理表示层逻辑。对于表示层，这是指增强型 I/O 线程现在不仅可以读取客户端输入，还可以将输入解析为 Redis 二进制命令格式。然后将其转发到主线程用于执行，从而提高性能。改进了 Redis 内存访问模式。许多数据结构操作的执行步骤是交错的，以确保并行内存访问并减少内存访问延迟。在采用 Graviton3 的 `R7g.4xlarge` 或更大实例上运行 ElastiCache 时，客户的每个节点每秒可以实现超过 100 万个请求。借助 ElastiCache for Redis v7.1 的性能改进，与 ElastiCache for Redis v7.0 相比，客户可以将吞吐量提高多达 100%，P99 延迟降低 50%。这些增强功能适用于具有至少 8 个物理内核的节点大小（采用 Graviton 时为 `2xlarge`，采用 x86 时为 `4xlarge`），不受 CPU 类型的限制，并且无需更改客户端。

> **Note**
>
> ElastiCache v7.1 与 OSS Redis v7.0 兼容。

ElastiCache for Redis 版本 7.0（加强版）

ElastiCache for Redis 7.0 增加了多项改进和对新功能的支持：

- Redis Functions：ElastiCache for Redis 7 增加了对 Redis Functions 的支持，并提供了托管体验，使开发人员能够使用存储在 ElastiCache 集群上的应用程序逻辑执行 LUA 脚本，而无需客户端在每次连接时都将脚本重新发送到服务器。
- ACL 改进：ElastiCache for Redis 7 增加了对下一版本的 Redis 访问控制列表（ACL）的支持。借助 ElastiCache for Redis 7，客户端现在可以为 Redis 中的特定键或键空间指定多组权限。
- 分片发布/订阅：ElastiCache for Redis 7 增加了对在启用集群模式（CME）下运行 ElastiCache 时以分片方式运行 Redis 发布/订阅功能的支持。Redis 发布/订阅功能使发布者能够向频道上任意数量的订阅者发布消息。借助 Amazon ElastiCache for Redis 7，频道可绑定到 ElastiCache 集群中的分片，无需在分片之间传播频道信息，从而提高了可扩展性。
- 增强型 I/O 多路复用：ElastiCache for Redis 版本 7 引入了增强型 I/O 多路复用，此功能为与 ElastiCache 集群有着许多并发客户端连接的高吞吐量工作负载提供了更高的吞吐量和更短的延迟。例如，与 ElastiCache for Redis 版本 6 相比，当使用由 r6g.xlarge 节点组成的集群并运行 5200 个并发客户端时，吞吐量（每秒读写操作数）可以提高多达 72%，P99 延迟可减少多达 71%。

有关 Redis 7.0 版本的更多信息，请在 GitHub 上参阅 Redis 的 Redis 7.0 发布说明。

ElastiCache for Redis 版本 6.2（加强版）

ElastiCache for Redis 6.2 包含多项针对启用 TLS 的集群的性能改进，包括具有 8 个或以上 vCPU 的使用 x86 节点类型的集群，以及具有 4 个或以上 vCPU 的使用 Graviton2 节点类型的集群。这些增强功能通过将加密操作转移到其他 vCPU 来提高吞吐量并缩短建立客户端连接所需的时间。Redis 6.2 还支持使用访问控制列表（ACL）规则来管理对发布/订阅频道的访问。

在此版本中，我们还推出支持在包含本地挂载 NVMe SSD 的集群节点上使用数据分层功能。有关更多信息，请参阅数据分层。

Redis 引擎版本 6.2.6 还引入了对原生 JavaScript 对象表示法（JSON）格式的支持，这是在 Redis 集群中对复杂数据集进行编码的一种简单的无 Schema 的方法。借助 JSON 支持，您可以帮助基于 JSON 运行的应用程序利用性能和 Redis API。有关更多信息，请参阅 JSON 入门。还包括与 JSON

相关的指标 JsonBasedCmds 和 JsonBasedCmdsLatency，它们被合并到 CloudWatch 中以监控此数据类型的使用情况。有关更多信息，请参阅Redis 的指标。

您可以使用 6.2 来指定引擎版本。ElastiCache for Redis 会自动调用可用的 Redis 6.2 首选补丁版本。例如，您可在创建/修改缓存群集时将 --engine-version 参数设置为 6.2。则在集群创建/修改时，系统将会使用当前可用的 Redis 6.2 首选补丁版本启动集群。如果您在 API 中指定引擎版本 6.x，系统将会使用 Redis 6 的最新次要版本。

对于现有的 6.0 集群，您可以在 CreateCacheCluster、ModifyCacheCluster、CreateReplicationGroup 或 ModifyReplicationGroup API 中将 AutoMinorVersionUpgrade 参数设置为 yes，从而选择加入下一次自动次要版本升级。ElastiCache for Redis 会使用自助更新功能将现有 6.0 集群的次要版本升级到 6.2。有关更多信息，请参阅 Amazon ElastiCache 中的自助更新。

在调用 DescribeCacheEngineVersions API 时，EngineVersion 参数值将会设置为 6.2，并且将会在 CacheEngineVersionDescription 字段中返回实际引擎版本以及补丁版本。

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

ElastiCache for Redis 版本 6.0（加强版）

Amazon ElastiCache for Redis 推出新版本的 Redis 引擎，其中包括使用基于角色的访问控制对用户进行身份验证、客户端缓存和重要的操作改进。

从 Redis 6.0 开始，ElastiCache for Redis 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。ElastiCache for Redis 还将自动管理正在运行状态中的缓存群集的补丁版本，确保更好的性能和安全性。

您还可以通过将 AutoMinorVersionUpgrade 参数设置为 yes 来选择加入下一次自动次要版本升级，然后 ElastiCache for Redis 将使用自助更新功能来管理次要版本升级。有关更多信息，请参阅中的服务更新 ElastiCache。

您可以使用 6.0 指定引擎版本。ElastiCache for Redis 会自动调用可用的 Redis 6.0 首选补丁版本。例如，您可在创建/修改缓存群集时将 --engine-version 参数设置为 6.0。则在集群创建/修改时，系统将会使用当前可用的 Redis 6.0 首选补丁版本启动集群。任何包含特定补丁版本值的请求都将被拒绝，同时引发异常且进程会失败。

在调用 DescribeCacheEngineVersions API 时，EngineVersion 参数值将会设置为 6.0，并且将会在 CacheEngineVersionDescription 字段中返回实际引擎版本以及补丁版本。

有关 Redis 6.0 版本的更多信息，请在 GitHub 上参阅 Redis 的 Redis 6.0 发布说明。

ElastiCache for Redis 版本 5.0.6（加强版）

Amazon ElastiCache for Redis 推出新版本的 Redis 引擎，其中包含多项错误修复及以下累积更新：

- 特殊情况下的引擎稳定性保证。
- 改进的 Hyperloglog 错误处理。
- 增强的握手命令以进行可靠的复制。
- 通过 XCLAIM 命令进行一致的消息交付跟踪。
- 对象中改进的 LFU 字段管理。
- 使用 ZPOP 时增强的事务管理。
- 能够重命名命令：一个名为 rename-commands 的参数，允许您重命名可能导致意外数据丢失的潜在危险或成本高昂的 Redis 命令，例如 FLUSHALL 或 FLUSHDB。这与开源 Redis 中的重命名命令配置类似。但是，ElastiCache 通过提供完全托管的工作流程改善了体验。命令名称更改将立即应用，并自动在集群中包含命令列表的所有节点上传播。您无需干预，例如重新启动节点。

  以下示例演示了如何修改现有参数组。它们包括 rename-commands 参数，该参数是要重命名的以空格分隔的命令列表：

  ```
  aws elasticache modify-cache-parameter-group --cache-parameter-group-
  name custom_param_group
  --parameter-name-values "ParameterName=rename-commands,  ParameterValue='flushall
   restrictedflushall'" --region region
  ```

  在本示例中，rename-commands 参数用于将 flushall 命令重命名为 restrictedflushall。

  要重命名多个命令，请使用以下操作：

  ```
  aws elasticache modify-cache-parameter-group --cache-parameter-group-
  name custom_param_group
  --parameter-name-values "ParameterName=rename-commands,  ParameterValue='flushall
   restrictedflushall flushdb restrictedflushdb'" --region region
  ```

  要还原任何更改，请重新运行该命令并从要保留的 ParameterValue 列表中排除任何重命名的值，如下所示：

  ```
  aws elasticache modify-cache-parameter-group --cache-parameter-group-
  name custom_param_group
  ```

```
--parameter-name-values "ParameterName=rename-commands,  ParameterValue='flushall
 restrictedflushall'" --region region
```

在本示例中，flushall 命令将重命名为 restrictedflushall，而任何其他重命名的命令将恢复为其原始命令名称。

> **Note**
>
> 在重命名命令时，您将受到以下限制：
>
> - 所有重命名的命令都应该是字母数字。
> - 新命令名称的最大长度为 20 个字母数字字符。
> - 重命名命令时，请确保更新与集群关联的参数组。
> - 要完全阻止命令的使用，请使用关键字 blocked，如下所示：
>
> ```
> aws elasticache modify-cache-parameter-group --cache-parameter-group-
> name custom_param_group
> --parameter-name-values "ParameterName=rename-commands,
>  ParameterValue='flushall blocked'" --region region
> ```

有关参数更改以及有资格重命名的命令列表的更多信息，请参阅Redis 5.0.3 参数更改。

- Redis Streams：这模拟日志数据结构，允许创建者实时追加新项目。它还允许使用者以阻塞或非阻塞方式使用消息。Streams 还允许代表一组客户端的使用者组以合作方式使用同一消息流的不同部分，类似于 Apache Kafka。有关更多信息，请参阅 Redis 流简介。
- 支持一系列流命令，例如 XADD、XRANGE 和 XREAD。有关更多信息，请参阅 Redis 流命令。
- 大量新参数和重命名的参数。有关更多信息，请参阅Redis 5.0.0 参数更改。
- 一个新的 Redis 指标 StreamBasedCmds。
- Redis 节点的快照时间略快一些。

> **Important**
>
> Amazon ElastiCache for Redis 从开源版本 5.0.1 中反向移植了两个关键错误修复。下面列出了它们：
>
> - 在一些密钥过期后，RESTORE 将不匹配回复。

- XCLAIM 命令可能会返回错误的条目或使协议不同步。

这两个错误修复都包含在 ElastiCache for Redis 对 Redis 引擎版本 5.0.0 的支持中，并且将在未来版本更新中使用。

有关更多信息，请参阅 GitHub 上关于 Redis 的 Redis 5.0.6 发行说明。

ElastiCache for Redis 版本 5.0.5（已弃用，使用 5.0.6 版本）

Amazon ElastiCache for Redis 推出新版本的 Redis 引擎。该版本中包括在所有计划的操作期间针对自动故障转移集群的 ElastiCache for Redis 的在线配置更改。现在您可以扩展集群，升级 Redis 引擎版本，并应用补丁和维护更新，同时集群可保持在线并继续处理传入请求。它还包括错误修复。

有关更多信息，请参阅 GitHub 上关于 Redis 的 Redis 5.0.5 发行说明。

ElastiCache for Redis 版本 5.0.4（已弃用，使用 5.0.6 版本）

Amazon ElastiCache for Redis 推出了受 Amazon ElastiCache 支持的下一版 Redis 引擎。其中包含以下增强功能：

- 特殊情况下的引擎稳定性保证。
- 改进的 Hyperloglog 错误处理。
- 增强的握手命令以进行可靠的复制。
- 通过 XCLAIM 命令进行一致的消息交付跟踪。
- 对象中改进的 LFU 字段管理。
- 使用 ZPOP 时增强的事务管理。

有关更多信息，请参阅 GitHub 上关于 Redis 的 Redis 5.0.4 发布说明。

ElastiCache for Redis 版本 5.0.3（已弃用，使用 5.0.6 版本）

Amazon ElastiCache for Redis 推出了受 Amazon ElastiCache 支持的新版本 Redis 引擎，该版本中包括多项错误修复。

ElastiCache for Redis 版本 5.0.0（已弃用，使用 5.0.6 版本）

Amazon ElastiCache for Redis 推出了受 Amazon ElastiCache 支持的 Redis 引擎的下一个主要版本。ElastiCache for Redis 5.0.0 包含对以下改进的支持：

- Redis Streams：这模拟日志数据结构，允许创建者实时追加新项目。它还允许使用者以阻塞或非阻塞方式使用消息。Streams 还允许代表一组客户端的使用者组以合作方式使用同一消息流的不同部分，类似于 Apache Kafka。有关更多信息，请参阅 Redis 流简介。
- 支持一系列流命令，例如 XADD、XRANGE 和 XREAD。有关更多信息，请参阅 Redis 流命令。
- 大量新参数和重命名的参数。有关更多信息，请参阅Redis 5.0.0 参数更改。
- 一个新的 Redis 指标 StreamBasedCmds。
- Redis 节点的快照时间略快一些。

ElastiCache for Redis 版本 4.0.10（加强版）

Amazon ElastiCache for Redis 推出了受 Amazon ElastiCache 支持的 Redis 引擎的下一个主要版本。ElastiCache for Redis 4.0.10 包含对以下改进的支持：

- 单个 ElastiCache for Redis 版本中同时具有在线集群大小调整和加密功能。有关更多信息，请参阅下列内容：
  - 扩展 Redis（启用集群模式）集群
  - Redis（已启用集群模式）的在线重新分片和分片重新平衡
  - Amazon ElastiCache 中的数据安全性
- 许多新参数。有关更多信息，请参阅Redis 4.0.10 参数更改。
- 支持内存命令系列，如 MEMORY。有关更多信息，请参阅 Redis 命令（在 MEMO 上搜索）。
- 支持在线内存碎片整理，从而可实现更高效的内存使用率并可为您的数据提供更多内存。
- 支持异步刷新和删除。ElastiCache for Redis 支持 UNLINK、FLUSHDB 和 FLUSHALL 等命令在与主线程不同的线程中运行。这样做可以异步释放内存，从而有助于提高应用程序的性能和响应速度。
- 一个新的 Redis 指标 ActiveDefragHits。有关更多信息，请参阅 Redis 的指标。

运行 Redis 3.2.10 版的 Redis 用户（集群模式已禁用）可以使用控制台通过在线升级来升级自己的集群。

ElastiCache for Redis 集群大小调整和加密支持对比

功能	3.2.6	3.2.10	4.0.10 及后续版本
在线集群大小调整*	否	是	是

功能	3.2.6	3.2.10	4.0.10 及后续版本
传输中加密 **	是	否	是
静态加密**	是	否	是
* 添加、删除和重新平衡分片。			
** 对于符合 FedRAMP、HIPAA 和 PCI DSS 标准的应用程序是必需的。有关更多信息，请参阅Amazon 合规性验证 ElastiCache。			

已经终止生命期周期（EOL）版本（3.x）

ElastiCache for Redis 版本 3.2.10（加强版）

Amazon ElastiCache for Redis 推出了受 Amazon ElastiCache 支持的 Redis 引擎的下一个主要版本。ElastiCache for Redis 3.2.10 推出了在线调整集群大小的功能，以便在集群中添加或删除分区时可以继续为传入输入/输出请求提供服务。ElastiCache for Redis 3.2.10 用户可使用早期 Redis 版本的所有功能，但数据加密功能除外。此功能目前仅在版本 3.2.6 中可用。

ElastiCache for Redis 版本 3.2.6 和 3.2.10 对比

功能	3.2.6	3.2.10
在线集群大小调整*	否	是
传输中加密 **	是	否
静态加密**	是	否
* 添加、删除和重新平衡分片。		
** 对于符合 FedRAMP、HIPAA 和 PCI DSS 标准的应用程序是必需的。有关更多信息，请参阅Amazon 合规性验证 ElastiCache。		

有关更多信息，请参阅下列内容：

- Redis（已启用集群模式）的在线重新分片和分片重新平衡

- 在线集群大小调整

ElastiCache for Redis 版本 3.2.6（加强版）

Amazon ElastiCache for Redis 推出了受 Amazon ElastiCache 支持的 Redis 引擎的下一个主要版本。ElastiCache for Redis 3.2.6 用户可使用早期 Redis 版本的所有功能以及加密数据的选项。有关更多信息，请参阅下列内容：

- ElastiCache 传输中加密 (TLS)
- ElastiCache 中的静态加密
- Amazon 合规性验证 ElastiCache

ElastiCache for Redis 版本 3.2.4（加强版）

Amazon ElastiCache for Redis 版本 3.2.4 推出了受 Amazon ElastiCache 支持的 Redis 引擎的下一个主要版本。ElastiCache for Redis 3.2.4 用户可使用早期版本的 Redis 中的所有功能，并且能够在集群模式或非集群模式下运行。下表进行了汇总。

Redis 3.2.4 非集群模式与集群模式对比

功能	非集群模式	集群模式
数据分区	否	是
地理空间索引	是	是
更改节点类型	是	是*
副本扩展	是	是*
扩展	否	是*
数据库支持	多个	单列排序
参数组	`default.redis3.2` **	`default.redis3.2.cluster.on` **
*请参阅 从备份还原到新缓存		

功能	非集群模式	集群模式
**或从其派生的某个参数组。		

注意:

- 分区 – 将数据拆分到 2 到 500 个节点组（分片），为每个节点组提供复制支持的能力。
- 地理空间索引 – Redis 3.2.4 推出了对通过 6 条 GEO 命令执行地理空间索引的支持。有关更多信息，请参阅 Redis Commands 页面（针对 GEO 进行筛选）上的 Redis GEO* 命令文档 Redis 命令：GEO。

有关其他的 Redis 3 功能的信息，请参阅 Redis 3.2 发布说明和 Redis 3.0 发布说明。

当前，ElastiCache 托管式 Redis（集群模式已禁用）不支持以下 Redis 3.2 功能：

- 副本迁移
- 集群重新平衡
- Lua 调试程序

ElastiCache 禁用了以下 Redis 3.2 管理命令：

- `cluster meet`
- `cluster replicate`
- `cluster flushslots`
- `cluster addslots`
- `cluster delslots`
- `cluster setslot`
- `cluster saveconfig`
- `cluster forget`
- `cluster failover`
- `cluster bumpepoch`
- `cluster set-config-epoch`
- `cluster reset`

有关 Redis 3.2.4 参数的信息，请参阅Redis 3.2.4 参数更改。

已经终止生命期周期（EOL）版本（2.x）

ElastiCache for Redis 版本 2.8.24（加强版）

自版本 2.8.23 起增加的 Redis 改进功能包括错误修复和针对错误内存访问地址的记录功能。有关更多信息，请参阅 Redis 2.8 发布说明。

ElastiCache for Redis 版本 2.8.23（加强版）

自版本 2.8.22 起增加的 Redis 改进功能包括 Bug 修复。有关更多信息，请参阅 Redis 2.8 发布说明。此版本还包括对新参数 `close-on-slave-write` 的支持；如果启用该参数，尝试写入只读副本的客户端将会断开连接。

有关 Redis 2.8.23 参数的更多信息，请参见 ElastiCache 用户指南中的 Redis 2.8.23（加强版）增加的参数。

ElastiCache for Redis 版本 2.8.22（增强版）

自版本 2.8.21 起增加的 Redis 改进功能包括：

- 支持无分支备份和同步，使您能够为备份开销分配更少内存并为应用程序分配更多内存。有关更多信息，请参阅如何实施同步和备份。此无分支过程会影响延迟和吞吐量。当存在高写入吞吐量时，如果副本重新同步，则在整个同步过程中将无法访问副本。
- 如果存在故障转移，由于副本将尽可能执行与主集群的部分同步而不是执行完整同步，因此复制组现在能够更快地恢复。此外，主集群和副本在同步期间不再使用磁盘，并将进一步加快速度。
- 支持两个新的 CloudWatch 指标。
  - `ReplicationBytes` – 复制组的主集群发送到只读副本的字节数。
  - `SaveInProgress` – 一个指示是否有后台保存进程正在运行的二进制值。

  有关更多信息，请参阅使用 CloudWatch 指标监控使用情况。
- 修复了复制 PSYNC 行为中的许多关键 Bug。有关更多信息，请参阅 Redis 2.8 发布说明。
- 为保持多可用区复制组中增强的复制性能并提高集群的稳定性，将不再支持非 ElastiCache 副本。
- 为了提高主集群与复制组中的副本之间的数据一致性，该副本不再移出独立于主集群的密钥。
- Redis 版本 2.8.22 及更高版本不支持 Redis 配置变量 `appendonly` 和 `appendfsync`。
- 在内存不足的情况下，具有较大的输出缓冲区的客户端可能会与副本集群断开。如果已断开连接，则客户端需要重新连接。此类情况最有可能在 PUBSUB 客户端上出现。

ElastiCache for Redis 版本 2.8.21

自版本 2.8.19 起增加的 Redis 改进功能包括大量 Bug 修复。有关更多信息，请参阅 Redis 2.8 发布说明。

ElastiCache for Redis 版本 2.8.19

自版本 2.8.6 起增加的 Redis 改进功能包括：

- 支持 HyperLogLog。有关更多信息，请参阅 Redis new data structure: HyperLogLog。
- 现在，经过排序的集数据类型通过新命令 ZRANGEBYLEX、ZLEXCOUNT 和 ZREMRANGEBYLEX 支持字典顺序范围查询。
- 为了防止主节点向副本节点发送陈旧数据，如果后台保存 (bgsave) 子进程中止，则主同步会失败。
- 对 HyperLogLogBasedCommands CloudWatch 指标的支持。有关更多信息，请参阅Redis 的指标。

ElastiCache for Redis 版本 2.8.6

自版本 2.6.13 起增加的 Redis 改进功能包括：

- 提高了只读副本的弹性和容错性能。
- 支持部分重新同步。
- 支持必须始终可用的只读副本的用户定义最小数目。
- 完全支持发布/订阅 – 就服务器上发生的事件通知客户端。
- 自动检测主节点故障并将主节点故障转移至辅助节点。

ElastiCache for Redis 版本 2.6.13

Amazon ElastiCache for Redis 最开始支持的 Redis 版本是 Redis 版本 2.6.13。Redis 2.6.13 不支持多可用区。

Redis 版本生命周期终止计划

本部分定义了较早的主要版本在宣布时的生命周期终止（EOL）日期。这样有助于您为将来做出版本和升级决策。

> **Note**
>
> 从 5.0.0 到 5.0.5 的 ElastiCache for Redis 补丁版本已弃用。使用版本 5.0.6 或更高版本。

下表汇总了每个版本及其宣布的生命周期终止日期，以及推荐的升级目标版本。

已经终止生命周期

源次要版本	建议升级目标	生命周期终止日期
3.2.4、3.2.6 和 3.2.10	版本 6.2 或更高版本 **Note** 对于 US-ISO-EAST-1、US-ISO-WEST-1 和 US-ISOB-EAST-1 区域，我们建议使用 5.0.6 或更高版本。	2023 年 7 月 31 日
2.8.24、2.8.23、2.8.22、2.8.21、2.8.19、2.8.12、2.8.6、2.6.13	版本 6.2 或更高版本 **Note** 对于 US-ISO-EAST-1、US-ISO-WEST-1 和 US-ISOB-EAST-1 区域，我们建议使用 5.0.6 或更高版本。	2023 年 1 月 13 日

如何升级引擎版本

使用 ElastiCache 控制台、AWS CLI 或 ElastiCache API 修改集群或复制组并指定较新的引擎版本，从而启动集群或复制组的版本升级。有关更多信息，请参阅以下主题。

如何修改集群和复制组	
集群	复制组
使用 AWS Management Console	使用 AWS Management Console
使用 AWS CLI	使用 AWS CLI
使用 ElastiCache API	使用 ElastiCache API

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
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

解决阻止的 Redis 引擎升级

- 请执行以下操作之一：
  - 通过清除 Apply immediately (立即应用) 复选框，将 Redis 引擎升级操作安排在下一维护时段内。

    对于 CLI，请使用 `--no-apply-immediately`。对于 API，请使用 `ApplyImmediately=false`。

- 等到下一维护时段（或之后）再执行 Redis 引擎升级操作。
- 将 Redis 纵向扩展操作添加到此选中了 Apply Immediately (立即应用) 复选框的集群修改中。

  对于 CLI，请使用 `--apply-immediately`。对于 API，请使用 `ApplyImmediately=true`。

  此方法将立即执行引擎升级，从而有效地在下一维护时段内取消该操作。

主要版本行为和兼容性差异

> **Important**
>
> 以下页面的结构设计为指出版本之间的所有不兼容性差异，并告知您在升级到较新版本时应考虑的任何注意事项。此列表包含升级时可能遇到的任何版本不兼容问题。
> 您可以直接从当前 Redis 版本升级到可用的最新 Redis 版本，无需连续升级。例如，您可以直接从 Redis 版本 3.0 升级到版本 7.0。

Redis 版本由包含 MAJOR、MINOR 和 PATCH 组件的语义版本标识。例如，在 Redis 4.0.10 中，主要版本为 4，次要版本为 0，补丁版本为 10。这些值通常根据以下约定递增：

- MAJOR 版本针对 API 不兼容的更改
- MINOR 版本针对以向后兼容的方式添加的新功能
- PATCH 版本针对向后兼容的错误修复和非功能性更改

我们建议始终使用给定 MAJOR.MINOR 版本中的最新补丁版本，以获得最新的性能和稳定性改进。从 Redis 6.0 开始，ElastiCache for Redis 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。ElastiCache for Redis 还将自动管理正在运行状态中的缓存集群的补丁版本，确保更好的性能和安全性。

我们还建议定期升级到最新的主要版本，因为大多数主要改进都不会向后移植到旧版本。随着 ElastiCache 将可用性扩展到新的 AWS 区域，ElastiCache for Redis 支持将当时最新的两个 MAJOR.MINOR 版本用于此新区域。例如，如果一个新 AWS 区域推出，并且最新的 MAJOR.MINOR ElastiCache for Redis 版本为 7.0 和 6.2，则 ElastiCache for Redis 将在新的 AWS 区域中支持版本 7.0 和 6.2。随着 ElastiCache for Redis 的更高 MAJOR.MINOR 版本发布，ElastiCache 将继续增加对新发布的 ElastiCache for Redis 版本的支持。要详细了解如何为 ElastiCache 选择区域，请参阅选择区域和可用区。

在进行跨主版本或次版本的升级时，请考虑以下列表，其中包括随着时间的推移随 Redis 发布的行为和向后不兼容更改。

Redis 7.0 行为和向后不兼容更改

有关更改的完整列表，请参阅 Redis 7.0 发布说明。

- SCRIPT LOAD 和 SCRIPT FLUSH 不再传播到副本。如果您需要脚本具有一定的持久性，我们建议您考虑使用 Redis 函数。
- 现在，对于新的 ACL 用户，发布订阅通道默认处于屏蔽状态。
- STRALGO 命令已替换为 LCS 命令。
- ACL GETUSER 的格式已更改，因此所有字段都显示标准访问字符串模式。如果您使用 ACL GETUSER 实现了自动化，则应验证它是否可以处理任何一种格式。
- SELECT、WAIT、ROLE、LASTSAVE、READONLY、READWRITE 和 ASKING 的 ACL 类别已更改。
- INFO 命令现在显示每个子命令的命令统计信息，而不是在顶级容器命令中显示命令统计信息。
- 在某些边缘情况下，LPOP、RPOP、ZPOPMIN 和 ZPOPMAX 命令的返回值已更改。如果您使用这些命令，则应查看发布说明并评估是否受到影响。
- SORT 和 SORT_RO 命令现在需要访问整个键空间才能使用 GET 和 BY 参数。

Redis 6.2 行为和向后不兼容更改

有关更改的完整列表，请参阅 Redis 6.2 发布说明。

- TIME、ECHO、ROLE 和 LASTSAVE 命令的 ACL 标志已更改。这可能会导致先前允许的命令被拒绝，反之亦然。

  > **Note**
  >
  > 这些命令不会修改或授予对数据的访问权限。

- 从 Redis 6.0 升级时，从映射响应返回到 lua 脚本的键/值对的顺序发生了变化。如果您的脚本使用 `redis.setresp()` 或返回映射（Redis 6.0 中的新功能），请考虑脚本可能会在升级时发生中断所带来的影响。

Redis 6.0 行为和向后不兼容更改

有关更改的完整列表，请参阅 Redis 6.0 发布说明。

- 允许的最大数据库数量已从 120 万减少到 1 万个。默认值为 16，我们不鼓励使用比这大许多的值，因为我们发现这样会带来性能和内存问题。
- 将 AutoMinorVersionUpgrade 参数设置为“yes”（是），ElastiCache for Redis 将通过自助更新管理次版本升级。这将通过自助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 ElastiCache 中的自助更新。

Redis 5.0 行为和向后不兼容更改

有关更改的完整列表，请参阅 Redis 5.0 发布说明。

- 脚本由效果复制，而不是在副本上重新执行脚本。这通常可以提高性能，但可能会增加主副本和副本之间复制的数据量。有一个选项可以恢复到之前的行为，该选项仅在 ElastiCache for Redis 5.0 中可用。
- 如果您是从 Redis 4.0 升级，LUA 脚本中的某些命令将以与早期版本不同的顺序返回参数。在 Redis 4.0 中，Redis 会按字典顺序对某些响应进行排序，以使响应具有确定性，当脚本通过效果复制时，不应用此排序。
- 在 Redis 5.0.3 及更高版本中，ElastiCache for Redis 会将一些 IO 工作转移到具有 4 个 vCPU 以上的实例类型的后台内核。这可能会改变 Redis 的性能特性并更改某些指标的值。有关更多信息，请参阅 应监控哪些指标？，以了解您是否需要更改要关注的指标。

Redis 4.0 行为和向后不兼容更改

有关更改的完整列表，请参阅 Redis 4.0 发布说明。

- 慢日志现在记录两个额外参数，即客户端名称和地址。除非您明确依赖包含 3 个值的每个慢日志条目，否则此更改应向后兼容。
- CLUSTER NODES 命令现在返回的格式略有不同，不向后兼容。我们建议客户端不要使用此命令来了解集群中存在的节点，而应使用 CLUSTER SLOTS。

已经终止生命周期

Redis 3.2 行为和向后不兼容更改

有关更改的完整列表，请参阅 Redis 3.2 发布说明。

- 此版本没有兼容性更改需要调用。

有关更多信息，请参阅Redis 版本生命周期终止计划。

Redis 2.8 行为和向后不兼容更改

有关更改的完整列表，请参阅 Redis 2.8 发布说明。

- 从 Redis 2.8.22 开始，ElastiCache for Redis 不再支持 Redis AOF。当数据需要持久保存时，我们建议使用 MemoryDB。
- 从 Redis 2.8.22 开始，ElastiCache for Redis 不再支持将副本附加到 ElastiCache 中托管的主节点。升级时，外部副本将断开连接，且无法重新连接。我们建议使用 Redis 6.0 中提供的客户端缓存作为外部副本的替代方案。
- 如果键不存在，TTL 和 PTTL 命令现在返回 -2；如果键存在但没有关联的过期时间，则返回 -1。Redis 2.6 和以前的版本曾经在这两种情况下都返回 -1。
- 如果未使用任何 STORE 选项，SORT 和 ALPHA 现在将根据本地排序规则语言环境进行排序。

有关更多信息，请参阅Redis 版本生命周期终止计划。

ElastiCache 最佳实践和缓存策略

您可以在下面找到推荐的 Amazon 最佳实践 ElastiCache。遵循这些最佳实践可提高您的缓存的性能和可靠性。

主题

- 使用 Redis
- 有关 Redis 客户端的最佳实践
- 管理预留内存
- 使用自行设计的集群时的最佳实践
- Redis 最佳实践
- 缓存策略

使用 Redis

在下文中，您将了解有关 ElastiCache 中 Redis 接口的信息。

主题

- 支持和限制使用的 Redis 命令
- Redis 配置和限制

支持和限制使用的 Redis 命令

支持的 Redis 命令

支持的 Redis 命令

无服务器缓存支持以下 Redis 命令。除了这些命令外，还支持这些 支持的 Redis JSON 命令 命令。

位图命令

- BITCOUNT

  计算字符串中的设置位数（群体计数）。

  了解更多
- BITFIELD

  对字符串执行任意位字段整数运算。

  了解更多
- BITFIELD_RO

  对字符串执行任意只读位字段整数运算。

  了解更多
- BITOP

  对多个字符串执行按位运算，并存储结果。

  了解更多
- BITPOS

  查找字符串中的第一个设置（1）或清除（0）位。

  了解更多
- GETBIT

按偏移量返回位值。

了解更多

- SETBIT

  设置或清除字符串值中偏移量处的位。创建键（如果它不存在）。

  了解更多

集群管理命令

- CLUSTER COUNTKEYSINSLOT

  返回哈希槽中的键数。

  了解更多

- CLUSTER GETKEYSINSLOT

  返回哈希槽中的键名称。

  了解更多

- CLUSTER INFO

  返回有关节点状态的信息。在无服务器缓存中，返回有关显示给客户端的单个虚拟“分片”的状态。

  了解更多

- CLUSTER KEYSLOT

  返回键的哈希槽。

  了解更多

- CLUSTER MYID

  返回节点的 ID。在无服务器缓存中，返回有关显示给客户端的单个虚拟“分片”的状态。

  了解更多

- CLUSTER NODES

  返回节点的集群配置。在无服务器缓存中，返回有关显示给客户端的单个虚拟“分片”的状态。

了解更多

- CLUSTER REPLICAS

  列出主节点的副本节点。在无服务器缓存中，返回有关显示给客户端的单个虚拟“分片”的状态。

  了解更多

- CLUSTER SHARDS

  返回集群槽与分片的映射。在无服务器缓存中，返回有关显示给客户端的单个虚拟“分片”的状态。

  了解更多

- CLUSTER SLOTS

  返回集群槽与节点的映射。在无服务器缓存中，返回有关显示给客户端的单个虚拟“分片”的状态。

  了解更多

- READONLY

  为与 Redis 集群副本节点的连接启用只读查询。

  了解更多

- READWRITE

  为与 Redis 集群副本节点的连接启用读写查询。

  了解更多

连接管理命令

- AUTH

  对连接进行身份验证。

  了解更多

- CLIENT GETNAME

  返回连接的名称。

  了解更多

- CLIENT REPLY

  指示服务器是否回复命令。

  了解更多

- CLIENT SETNAME

  设置连接名称。

  了解更多

- ECHO

  返回给定字符串。

  了解更多

- HELLO

  与 Redis 服务器握手。

  了解更多

- PING

  返回服务器的活跃度响应。

  了解更多

- QUIT

  关闭连接。

  了解更多

- RESET

  重置连接。

  了解更多

- SELECT

  更改所选数据库。

  了解更多

通用命令

- COPY

  将键的值复制到新键中。

  了解更多

- DEL

  删除一个或多个键。

  了解更多

- DUMP

  返回存储在某个键上的值的序列化表示形式。

  了解更多

- EXISTS

  确定是否存在一个或多个键。

  了解更多

- EXPIRE

  以秒为单位设置键的过期时间。

  了解更多

- EXPIREAT

  以 Unix 时间戳的格式设置键的过期时间。

  了解更多

- EXPIRETIME

  以 Unix 时间戳的格式返回键的过期时间。

  了解更多

- PERSIST

  移除键的过期时间。

了解更多

- PEXPIRE

  以毫秒为单位设置键的过期时间。

  了解更多

- PEXPIREAT

  以 Unix 毫秒时间戳的格式设置键的过期时间。

  了解更多

- PEXPIRETIME

  以 Unix 毫秒时间戳的格式返回键的过期时间。

  了解更多

- PTTL

  以毫秒为单位返回键的过期时间。

  了解更多

- RANDOMKEY

  从数据库中返回随机键名称。

  了解更多

- RENAME

  重命名键并覆盖目标。

  了解更多

- RENAMENX

  仅在目标键名称不存在时重命名键。

  了解更多

- RESTORE

  根据值的序列化表示形式创建键。

了解更多

- SCAN

  遍历数据库中的键名。

  了解更多

- SORT

  对列表、集或排序集中的元素进行排序，可以选择将结果存储起来。

  了解更多

- SORT_RO

  返回列表、集或排序集的排序元素。

  了解更多

- TOUCH

  返回在更新最后一次访问键的时间后，指定键中现有键的数量。

  了解更多

- TTL

  以秒为单位返回键的过期时间。

  了解更多

- TYPE

  确定存储在键中的值的类型。

  了解更多

- UNLINK

  异步删除一个或多个键。

  了解更多

地理空间命令

- GEOADD

  将一个或多个成员添加到地理空间索引。如果键不存在，则创建它。

  了解更多

- GEODIST

  返回地理空间索引的两个成员之间的距离。

  了解更多

- GEOHASH

  以地理哈希字符串的格式返回地理空间索引中的成员。

  了解更多

- GEOPOS

  从地理空间索引返回成员的经度和纬度。

  了解更多

- GEORADIUS

  查询与坐标相隔一定距离内的成员的地理空间索引，可以选择将结果存储起来。

  了解更多

- GEORADIUS_RO

  从地理空间索引中返回与坐标相隔一定距离内的成员。

  了解更多

- GEORADIUSBYMEMBER

  查询与某个成员相隔一定距离内的成员的地理空间索引，可以选择将结果存储起来。

  了解更多

- GEORADIUSBYMEMBER_RO

  从地理空间索引中返回与成员相隔一定距离内的成员。

  了解更多

- GEOSEARCH

  查询方框或圆圈区域内成员的地理空间索引。

  了解更多

- GEOSEARCHSTORE

  查询方框或圆圈区域内成员的地理空间索引，可以选择将结果存储起来。

  了解更多

哈希命令

- HDEL

  从哈希中删除一个或多个字段及其值。如果没有字段剩余，则删除哈希。

  了解更多

- HEXISTS

  确定哈希中是否存在某个字段。

  了解更多

- HGET

  返回哈希中某个字段的值。

  了解更多

- HGETALL

  返回哈希中的所有字段和值。

  了解更多

- HINCRBY

  将哈希中某个字段的整数值增加一个数字。如果该字段不存在，则使用 0 作为初始值。

  了解更多

- HINCRBYFLOAT

将某个字段的浮点值增加一个数字。如果该字段不存在，则使用 0 作为初始值。

了解更多

- HKEYS

  返回哈希中的所有字段。

  了解更多

- HLEN

  返回哈希中的字段数。

  了解更多

- HMGET

  返回哈希中的所有字段的值。

  了解更多

- HMSET

  设置多个字段的值。

  了解更多

- HRANDFIELD

  从哈希返回一个或多个随机字段。

  了解更多

- HSCAN

  遍历哈希的字段和值。

  了解更多

- HSET

  在哈希中创建或修改某个字段的值。

  了解更多

- HSETNX

仅当哈希中不存在某个字段时才设置该字段的值。

了解更多

- HSTRLEN

  返回某个字段的值的长度。

  了解更多

- HVALS

  返回哈希中的所有值。

  了解更多

HyperLogLog 命令

- PFADD

  向 HyperLogLog 键添加元素。创建键（如果它不存在）。

  了解更多

- PFCOUNT

  返回 HyperLogLog 键观察到的集的近似基数。

  了解更多

- PFMERGE

  将一个或多个 HyperLogLog 值合并到单个键中。

  了解更多

列出命令

- BLMOVE

  从列表中弹出一个元素，将其推送到另一个列表并返回。否则将阻止，直到元素可用。如果最后一个元素已移除，则删除列表。

了解更多

- BLMPOP

  从多个列表之一弹出第一个元素。否则将阻止，直到元素可用。如果最后一个元素已弹出，则删除列表。

  了解更多

- BLPOP

  删除并返回列表中的第一个元素。否则将阻止，直到元素可用。如果最后一个元素已弹出，则删除列表。

  了解更多

- BRPOP

  删除并返回列表中的最后一个元素。否则将阻止，直到元素可用。如果最后一个元素已弹出，则删除列表。

  了解更多

- BRPOPLPUSH

  从列表中弹出一个元素，将其推送到另一个列表并返回。否则将阻止，直到元素可用。如果最后一个元素已弹出，则删除列表。

  了解更多

- LINDEX

  从列表中按元素索引返回该元素。

  了解更多

- LINSERT

  在列表中的另一个元素之前或之后插入一个元素。

  了解更多

- LLEN

  返回列表的长度。

了解更多

- LMOVE

  从一个列表中弹出一个元素并将其推送到另一个列表后，返回该元素。如果最后一个元素已移除，则删除列表。

  了解更多

- LMPOP

  删除列表中的多个元素后，返回这些元素。如果最后一个元素已弹出，则删除列表。

  了解更多

- LPOP

  删除列表中的第一个元素之后返回该元素。如果最后一个元素已弹出，则删除列表。

  了解更多

- LPOS

  返回列表中匹配元素的索引。

  了解更多

- LPUSH

  在列表前面追加一个或多个元素。创建键（如果它不存在）。

  了解更多

- LPUSHX

  仅当列表存在时，在列表前面追加一个或多个元素。

  了解更多

- LRANGE

  返回列表中元素的范围。

  了解更多

- LREM

从列表中删除元素。如果最后一个元素已删除，则删除列表。

了解更多

- LSET

  在列表中按元素索引设置元素的值。

  了解更多

- LTRIM

  从列表的两端删除元素。如果所有元素都已去除，则删除该列表。

  了解更多

- RPOP

  返回并删除列表中的最后一个元素。如果最后一个元素已弹出，则删除列表。

  了解更多

- RPOPLPUSH

  在删除列表的最后一个元素并将其推送到另一个列表后，返回该元素。如果最后一个元素已弹出，则删除列表。

  了解更多

- RPUSH

  在列表中附加一个或多个元素。创建键（如果它不存在）。

  了解更多

- RPUSHX

  仅当列表存在时将元素附加到列表中。

  了解更多

Pub/Sub 命令

> **Note**
>
> PUBSUB 命令在内部使用分片 PUBSUB，因此频道名称会混合。

- PUBLISH

  将消息发布到频道。

  了解更多

- PUBSUB CHANNELS

  返回活跃频道。

  了解更多

- PUBSUB NUMSUB

  返回频道的订阅用户数量。

  了解更多

- PUBSUB SHARDCHANNELS

  返回活跃的分片频道。

  PUBSUB-SHARDCHANNELS

- PUBSUB SHARDNUMSUB

  返回分片频道的订阅用户数量。

  PUBSUB-SHARDNUMSUB

- SPUBLISH

  向分片频道发布消息

  了解更多

- SSUBSCRIBE

  侦听发布到分片频道的消息。

  了解更多

- SUBSCRIBE

  侦听发布到频道的消息。

  了解更多

- SUNSUBSCRIBE

  停止侦听发布到分片频道的消息。

  了解更多

- UNSUBSCRIBE

  停止侦听发布到频道的消息。

  了解更多

脚本命令

- EVAL

  执行服务器端 Lua 脚本。

  了解更多

- EVAL_RO

  执行只读服务器端 Lua 脚本。

  了解更多

- EVALSHA

  按 SHA1 摘要执行服务器端 Lua 脚本。

  了解更多

- EVALSHA_RO

  按 SHA1 摘要执行只读服务器端 Lua 脚本。

  了解更多

- SCRIPT EXISTS

确定脚本缓存中是否存在服务器端 Lua 脚本。

了解更多

- SCRIPT FLUSH

  目前，无操作脚本缓存由该服务管理。

  了解更多

- SCRIPT LOAD

  将服务器端 Lua 脚本加载到脚本缓存中。

  了解更多

服务器管理命令

- ACL CAT

  列出 ACL 类别或类别内的命令。

  了解更多

- ACL GENPASS

  生成可用于识别 ACL 用户的伪随机安全密码。

  了解更多

- ACL GETUSER

  列出用户的 ACL 规则。

  了解更多

- ACL LIST

  以 ACL 文件格式转储有效规则。

  了解更多

- ACL USERS

  列出所有 ACL 用户。

了解更多

- ACL WHOAMI

  返回当前连接的经过身份验证的用户名。

  了解更多

- DBSIZE

  返回当前所选数据库中的键数量。不保证此操作在所有槽中都是原子形式的。

  了解更多

- COMMAND

  返回有关所有命令的详细信息。

  了解更多

- COMMAND COUNT

  返回命令的计数。

  了解更多

- COMMAND DOCS

  返回有关一个、多个或所有命令的已记录信息。

  了解更多

- COMMAND GETKEYS

  从任意命令中提取键名称。

  了解更多

- COMMAND GETKEYSANDFLAGS

  提取任意命令的键名称和访问标志。

  了解更多

- COMMAND INFO

  返回有关一个、多个或所有命令的信息。

了解更多

- COMMAND LIST

  返回命令名称的列表。

  了解更多

- FLUSHALL

  从所有数据库中删除所有键。不保证此操作在所有槽中都是原子形式的。

  了解更多

- FLUSHDB

  从当前数据库中删除所有键。不保证此操作在所有槽中都是原子形式的。

  了解更多

- INFO

  返回有关服务器的信息和统计信息。

  了解更多

- LOLWUT

  显示 Redis 版本和对应的计算机图像。

  了解更多

- ROLE

  返回复制角色。

  了解更多

- TIME

  返回服务器时间。

  了解更多

集命令

- SADD

  将一个或多个成员添加到集中。创建键（如果它不存在）。

  了解更多

- SCARDT

  返回集中的成员数。

  了解更多

- SDIFF

  返回多个集的差值。

  了解更多

- SDIFFSTORE

  将多个集的差值存储在一个键中。

  了解更多

- SINTER

  返回多个集的交集。

  了解更多

- SINTERCARD

  返回多个集的交集的成员数。

  了解更多

- SINTERSTORE

  将多个集的交集存储在一个键中。

  了解更多

- SISMEMBER

  确定成员是否属于一个集。

  了解更多

- SMEMBERS

  返回集的所有成员。

  了解更多

- SMISMEMBER

  确定多个成员是否属于一个集。

  了解更多

- SMOVE

  将成员从一个集移动到另一个集。

  了解更多

- SPOP

  删除集中的一个或多个随机成员后，返回这些成员。如果最后一个成员已弹出，则删除集。

  了解更多

- SRANDMEMBER

  从集中获取一个或多个随机成员

  了解更多

- SREM

  从集中删除一个或多个成员。如果最后一个成员已删除，则删除集。

  了解更多

- SSCAN

  遍历集的成员。

  了解更多

- SUNION

  返回多个集的并集。

  了解更多

- SUNIONSTORE

  将多个集的并集存储在一个密钥中。

  了解更多

排序集命令

- BZMPOP

  从一个或多个排序集中按分数移除并返回成员。否则将阻止，直到成员可用。如果最后一个元素已弹出，则删除排序集。

  了解更多

- BZPOPMAX

  从一个或多个排序集中移除并返回分数最高的成员。否则将阻止，直到成员可用。如果最后一个元素已弹出，则删除排序集。

  了解更多

- BZPOPMIN

  从一个或多个排序集中移除并返回分数最低的成员。否则将阻止，直到成员可用。如果最后一个元素已弹出，则删除排序集。

  了解更多

- ZADD

  将一个或多个成员添加到排序集中，或更新其分数。创建键（如果它不存在）。

  了解更多

- ZCARD

  返回排序集中的成员数。

  了解更多

- ZCOUNT

  返回排序集中分数在某个范围内的成员数。

了解更多

- ZDIFF

  返回多个排序集的差值。

  了解更多

- ZDIFFSTORE

  将多个排序集的差值存储在一个键中。

  了解更多

- ZINCRBY

  递增排序集中成员的分数。

  了解更多

- ZINTER

  返回多个排序集的交集。

  了解更多

- ZINTERCARD

  返回多个排序集的交集的成员数。

  了解更多

- ZINTERSTORE

  将多个排序集的交集存储在一个键中。

  了解更多

- ZLEXCOUNT

  返回排序集中某个字母表范围内的成员数。

  了解更多

- ZMPOP

删除一个或多个排序集中分数最高或最低的成员后，返回这些成员。如果最后一个元素已弹出，则删除排序集。

了解更多

- ZMSCORE

  返回排序集中一个或多个成员的分数。

  了解更多

- ZPOPMAX

  删除排序集中分数最高的成员后，返回这些成员。如果最后一个元素已弹出，则删除排序集。

  了解更多

- ZPOPMIN

  删除排序集中分数最低的成员后，返回这些成员。如果最后一个元素已弹出，则删除排序集。

  了解更多

- ZRANDMEMBER

  返回排序集中的一个或多个随机成员。

  了解更多

- ZRANGE

  返回排序集中某个索引范围内的成员。

  了解更多

- ZRANGEBYLEX

  返回排序集中某个字母表范围内的成员。

  了解更多

- ZRANGEBYSCORE

  返回排序集中某个分数范围内的成员。

  了解更多

- ZRANGESTORE

  将排序集中的一系列成员存储在一个键中。

  了解更多

- ZRANK

  返回排序集中按分数升序排序的成员的索引。

  了解更多

- ZREM

  从排序集中删除一个或多个成员。如果所有成员已删除，则删除排序集。

  了解更多

- ZREMRANGEBYLEX

  删除排序集中某个字母表范围内的成员。如果所有成员已删除，则删除排序集。

  了解更多

- ZREMRANGEBYRANK

  删除排序集中某个索引范围内的成员。如果所有成员已删除，则删除排序集。

  了解更多

- ZREMRANGEBYSCORE

  删除排序集中某个分数范围内的成员。如果所有成员已删除，则删除排序集。

  了解更多

- ZREVRANGE

  以相反的顺序返回某个索引范围内排序集中的成员。

  了解更多

- ZREVRANGEBYLEX

  以相反的顺序返回排序集中某个字母表范围内的成员。

  了解更多

- ZREVRANGEBYSCORE

  以相反的顺序返回排序集中某个分数范围内的成员。

  了解更多

- ZREVRANK

  按分数降序排序，返回排序集中某个成员的索引。

  了解更多

- ZSCAN

  遍历排序集的成员和分数。

  了解更多

- ZSCORE

  返回排序集中某个成员的分数。

  了解更多

- ZUNION

  返回多个排序集的并集。

  了解更多

- ZUNIONSTORE

  将多个排序集的并集存储在一个键中。

  了解更多

流命令

- XACK

  返回流的使用者组成员已成功确认的消息数量。

  了解更多

- XADD

在流中追加一条新消息。创建键（如果它不存在）。

了解更多

- XAUTOCLAIM

  更改或获取使用者组中某条消息的所有权，就像消息是以使用者组成员的身份传递一样。

  了解更多

- XCLAIM

  更改或获取使用者组中某条消息的所有权，就像消息是传递给使用者组成员一样。

  了解更多

- XDEL

  从流中删除消息后，返回删除的消息数。

  了解更多

- XGROUP CREATE

  创建使用者组。

  了解更多

- XGROUP CREATECONSUMER

  在使用者组中创建使用者。

  了解更多

- XGROUP DELCONSUMER

  从使用者组中删除使用者。

  了解更多

- XGROUP DESTROY

  销毁使用者组。

  了解更多

- XGROUP SETID

设置使用者组的上次传输 ID。

了解更多

- XINFO CONSUMERS

  返回使用者组中的使用者列表。

  了解更多

- XINFO GROUPS

  返回流的使用者组列表。

  了解更多

- XINFO STREAM

  返回有关流的信息。

  了解更多

- XLEN

  返回流中的消息数。

  了解更多

- XPENDING

  返回流使用者组的待处理条目列表中的信息和条目。

  了解更多

- XRANGE

  返回流中某个 ID 范围内的消息。

  了解更多

- XREAD

  返回多个流中其 ID 大于所请求 ID 的消息。否则将阻止，直到消息可用。

  了解更多

- XREADGROUP

为组中的某个使用者返回流中的新消息或历史消息。否则将阻止，直到消息可用。

了解更多

- XREVRANGE

  以相反的顺序返回流中某个 ID 范围内的消息。

  了解更多

- XTRIM

  从流的开头删除消息。

  了解更多

字符串命令

- APPEND

  在键值后面附加一个字符串。创建键（如果它不存在）。

  了解更多

- DECR

  将键的整数值减去 1。如果该键不存在，则使用 0 作为初始值。

  了解更多

- DECRBY

  从键的整数值减去一个数字。如果该键不存在，则使用 0 作为初始值。

  了解更多

- GET

  返回键的字符串值。

  了解更多

- GETDEL

  删除键后，返回键的字符串值。

了解更多

- GETEX

  在设置键的过期时间后，返回其字符串值。

  了解更多

- GETRANGE

  返回存储在键中的字符串的子字符串。

  了解更多

- GETSET

  将键设置为新值后，返回键的上一个字符串值。

  了解更多

- INCR

  将键的整数值增加 1。如果该键不存在，则使用 0 作为初始值。

  了解更多

- INCRBY

  将键的整数值增加一个数字。如果该键不存在，则使用 0 作为初始值。

  了解更多

- INCRBYFLOAT

  将键的浮点值增加一个数字。如果该键不存在，则使用 0 作为初始值。

  了解更多

- LCS

  查找最长的公共子字符串。

  了解更多

- MGET

  以原子方式返回一个或多个键的字符串值。

了解更多

- MSET

  以原子方式创建或修改一个或多个键的字符串值。

  了解更多

- MSETNX

  仅当所有键不存在时，以原子方式修改一个或多个键的字符串值。

  了解更多

- PSETEX

  设置键的字符串值和过期时间，以毫秒为单位。如果键不存在，则创建它。

  了解更多

- SET

  设置键的字符串值，忽略其类型。如果键不存在，则创建它。

  了解更多

- SETEX

  设置键的字符串值和过期时间。创建键（如果它不存在）。

  了解更多

- SETNX

  仅在某个键不存在时才设置该键的字符串值。

  了解更多

- SETRANGE

  按偏移量用字符串值的一部分覆盖另一个部分。创建键（如果它不存在）。

  了解更多

- STRLEN

  返回字符串值的长度。

了解更多

- SUBSTR

  返回字符串值中的子字符串。

  了解更多

事务命令

- DISCARD

  丢弃某个事务。

  了解更多

- EXEC

  执行某个事务中的所有命令。

  了解更多

- MULTI

  开始事务。

  了解更多

受限 Redis 命令

为了提供托管式服务体验，ElastiCache 限制了对某些需要高级特权的特定于缓存引擎的命令的访问。对于运行 Redis 的缓存，以下命令不可用：

- acl setuser
- acl load
- acl save
- acl deluser
- bgrewriteaof
- bgsave
- cluster addslot

- cluster addslotsrange
- cluster bumpepoch
- cluster delslot
- cluster delslotsrange
- cluster failover
- cluster flushslots
- cluster forget
- cluster links
- cluster meet
- cluster setslot
- config
- debug
- migrate
- psync
- replicaof
- save
- slaveof
- shutdown
- sync

此外，以下命令不可用于无服务器缓存：

- acl log
- client caching
- client getredir
- client id
- client info
- client kill
- client list
- client no-evict

- client pause
- client tracking
- client trackinginfo
- client unblock
- client unpause
- cluster count-failure-reports
- fcall
- fcall_ro
- function
- function delete
- function dump
- function flush
- function help
- function kill
- function list
- function load
- function restore
- function stats
- keys
- lastsave
- latency
- latency doctor
- latency graph
- latency help
- latency histogram
- latency history
- latency latest
- latency reset
- memory

- memory doctor
- memory help
- memory malloc-stats
- memory purge
- memory stats
- memory usage
- monitor
- move
- object
- object encoding
- object freq
- object help
- object idletime
- object refcount
- pfdebug
- pfselftest
- psubscribe
- pubsub numpat
- punsubscribe
- script kill
- slowlog
- slowlog get
- slowlog help
- slowlog len
- slowlog reset
- swapdb
- unwatch
- wait
- watch

Redis 配置和限制

Redis 引擎提供了许多配置参数，其中一些参数可以在 ElastiCache for Redis 中修改，另一些则不可修改，以提供稳定的性能和可靠性。

无服务器缓存

无服务器缓存不使用参数组，而且所有 Redis 配置都不可修改。其中包括以下 Redis 参数：

名称	详细信息	描述
acl-pubsub-default	allchannels	此缓存上的 ACL 用户默认拥有 pubsub 频道权限。
client-output-buffer-limit	normal 0 0 0 pubsub 32mb 8mb 60	普通客户端没有缓冲区限制。如果 PUB/SUB 客户端的积压超过 32MiB，或者在 60 秒内积压超过 8MiB，则它们将断开连接。
client-query-buffer-limit	1 GiB	单个客户端查询缓冲区的最大大小。此外，客户端不能发布参数超过 4000 个的请求。
cluster-allow-pubsubshard-when-down	yes	这样在缓存部分关闭时，缓存仍能为 pubsub 流量提供服务。
cluster-allow-reads-when-down	yes	这样在缓存部分关闭时，缓存仍能为读取流量提供服务。
cluster-enabled	yes	所有无服务器缓存都启用了集群模式，这使得它们可以在多个后端分片之间透明地将数据分区。所有槽都对客户端显示为属于单个虚拟节点。
cluster-require-fu	no	当键空间部分关闭时（即，至少有一个哈希槽无法访问），缓存将继续接受对仍在覆盖范围的部

名称	详细信息	描述
ll-coverage		分键空间的查询。在 cluster slots 中，整个键空间将始终由单个虚拟节点“覆盖”。
lua-time-limit	5000	Lua 脚本在 ElastiCache 执行操作以停止脚本之前的最大执行时间（毫秒）。 如果超过 lua-time-limit ，则所有 Redis 命令都可能返回形式为 ____-BUSY 的错误。因为此状态可能会导致干扰许多必要 Redis 操作，所以 ElastiCache 会首先发送 SCRIPT KILL 命令。如果这样做不成功，则 ElastiCache 强制重新启动 Redis。
maxclients	65000	缓存上可以一次连接的最大客户端连接数。超过此数量后，连接不一定能成功。
maxmemory-policy	volatile-lru	在达到缓存的内存限制时，将根据最近使用最少（LRU）原则估算值，对设置了 TTL 的项目进行驱逐。
notify-keyspace-events	（空字符串）	无服务器缓存目前不支持键空间事件。
port	主端口：6379 读取端口：6380	无服务器缓存向两个端口传播相同的主机名。主端口允许写入和读取，而读取端口则允许使用命令 READONLY 实现更低延迟的最终一致读数。
proto-max-bulk-len	512 MiB	单元素请求的最大大小。
timeout	0	客户端不会在达到特定空闲时间后被强制断开连接，但在稳定状态期间，它们可能会出于负载均衡目的而断开连接。

此外还有以下限制：

名称	详细信息	描述
键名称长度	4 KiB	单个 Redis 键或频道名称的最大大小。引用大于此值的键的客户端将收到错误消息。
Lua 脚本大小	4 MiB	单个 Redis Lua 脚本的最大大小。尝试加载大于此值的 Lua 脚本将收到错误。
槽大小	32 GiB	单个 Redis 哈希槽的最大大小。客户端如果尝试在单个 Redis 槽上设置超过此值的更多数据，就会触发驱逐策略，如果没有键可供驱逐，则会收到内存不足（OOM）错误。

自行设计的集群

对于自行设计的集群，请参阅 Redis 特定的参数 以了解配置参数的默认值，以及哪些参数可供配置。通常建议使用默认值，除非您的特定使用案例要求覆盖这些值。

有关 Redis 客户端的最佳实践

了解适用于常见场景的最佳实践，并遵循一些最常用的开源 Redis 客户端库（redis-py、PHPRedis 和 Lettuce）的代码示例。

主题

- 大量连接
- Redis 集群客户端发现和指数回退
- 配置客户端超时
- 配置服务器端空闲超时
- Redis Lua 脚本
- 存储大型复合项目
- Lettuce 客户端配置
- IPv6 客户端示例

大量连接

无服务器缓存和单个 ElastiCache for Redis 节点支持多达 65000 个并发客户端连接。但为了优化性能，我们建议客户端应用程序不要一直在该级别的连接上运行。Redis 是一个基于事件循环的单线程进程，其中将按顺序处理传入客户端请求。这意味着随着已连接客户端的增多，给定客户端的响应时间会变长。

您可以执行下面的一组操作，避免在 Redis 服务器上遇到连接瓶颈：

- 从只读副本执行读取操作。这可以通过使用 ElastiCache 读取器端点（在已禁用集群模式的情况下）或使用只读副本（在已启用集群模式的情况下，包括无服务器缓存）来实现。
- 在多个主节点之间分配写入流量。您可以通过两种方式执行此操作。您可以将多分片 Redis 集群与支持 Redis 集群模式的客户端结合使用。您还可以在已禁用集群模式的情况下通过客户端分片对多个主节点进行写入。此过程在无服务器缓存中自动完成。
- 如果您的客户端库中有连接池，请使用它。

通常，与典型的 Redis 命令相比，创建 TCP 连接是一项计算成本很高的操作。例如，在重复使用现有连接时，处理 SET/GET 请求的速度快一个数量级。使用大小有限的客户端连接池可减少连接管理的开销。它还将限制来自客户端应用程序的并发传入连接数。

以下 PHPRedis 代码示例说明为每个新用户请求创建一个新连接：

```
$redis = new Redis();
if ($redis->connect($HOST, $PORT) != TRUE) {
 //ERROR: connection failed
 return;
}
$redis->set($key, $value);
unset($redis);
$redis = NULL;
```

我们在连接到 Graviton2（m6g.2xlarge）ElastiCache for Redis 节点的 Amazon Elastic Compute Cloud（Amazon EC2）实例上的循环中对此代码进行了基准测试。我们将客户端和服务器置于同一可用区中。整个操作的平均延迟为 2.82 毫秒。

在更新代码并使用持久连接和连接池时，整个操作的平均延迟为 0.21 毫秒：

```
$redis = new Redis();
if ($redis->pconnect($HOST, $PORT) != TRUE) {
```

```
 // ERROR: connection failed
 return;
}
$redis->set($key, $value);
unset($redis);
$redis = NULL;
```

所需的 redis.ini 配置：

- `redis.pconnect.pooling_enabled=1`
- `redis.pconnect.connection_limit=10`

以下代码是 Redis-py 连接池的示例：

```
conn = Redis(connection_pool=redis.BlockingConnectionPool(host=HOST,
 max_connections=10))
conn.set(key, value)
```

以下代码是 Lettuce 连接池的示例：

```
RedisClient client = RedisClient.create(RedisURI.create(HOST, PORT));
GenericObjectPool<StatefulRedisConnection> pool =
 ConnectionPoolSupport.createGenericObjectPool(() -> client.connect(), new
 GenericObjectPoolConfig());
pool.setMaxTotal(10); // Configure max connections to 10
try (StatefulRedisConnection connection = pool.borrowObject()) {
 RedisCommands syncCommands = connection.sync();
 syncCommands.set(key, value);
}
```

Redis 集群客户端发现和指数回退

在已启用集群模式的情况下连接到 ElastiCache for Redis 集群时，相应的 Redis 客户端库必须能够感知集群。客户端必须获取哈希槽与集群中相应节点的映射，才能将请求发送到正确的节点，并避免处理集群重定向时产生的性能开销。因此，在两种不同的情况下，客户端必须发现槽和映射节点的完整列表：

- 客户端将初始化，并且必须填充初始槽配置
- 从服务器接收 MOVED 重定向，例如在前主节点提供的所有槽都由副本接管时进行失效转移的情况下，或者在槽从源主节点移动到目标主节点时进行重新分片的情况下

通常，通过向 Redis 服务器发出 CLUSTER SLOT 或 CLUSTER NODE 命令来完成客户端发现。我们建议使用 CLUSTER SLOT 方法，因为它会将一组槽范围以及关联的主节点和副本节点发送回客户端。这不需要从客户端进行额外分析，并且效率更高。

根据集群拓扑，CLUSTER SLOT 命令的响应大小可能会因集群大小而异。带多个节点的集群越大，响应越大。因此，请务必确保执行集群拓扑发现的客户端的数量不会无限增长。例如，在客户端应用程序启动或丢失与服务器的连接且必须执行集群发现时，通常会出现的一个错误是，客户端应用程序会在重试时未添加指数回退的情况下触发多个重新连接和发现请求。这可能导致 Redis 服务器长时间无响应，并且 CPU 利用率达到 100%。如果每条 CLUSTER SLOT 命令均必须处理集群总线中的大量节点，则中断时间会延长。过去，我们在包括 Python（redis-py-cluster）和 Java（Lettuce 和 Redisson）在内的许多不同语言中观察到，此行为导致多次发生客户端中断。

在无服务器缓存中，由于公布的集群拓扑是静态的，并且包含两个条目（写入端点和读取端点），因此许多问题会自动得到缓解。在使用缓存端点时，集群发现还会自动将负载分布到多个节点。但以下建议仍然有用。

为了减少突然涌入的连接和发现请求所造成的影响，我们建议采取以下措施：

- 实施一个大小有限的客户端连接池，以限制来自客户端应用程序的并发传入连接数。
- 当客户端因超时而断开与服务器的连接时，请使用带抖动的指数回退进行重试。这有助于避免多个客户端同时给服务器带来压力而导致其不堪重负。
- 使用查找连接端点中的指南查找用于执行集群发现的集群端点。这样一来，您便能将发现负载分布到集群中的所有节点（最多 90 个）上，而不是分布到集群中的几个硬编码的种子节点上。

以下是 redis-py、PHPRedis 和 Lettuce 中指数回退重试逻辑的一些代码示例。

回退逻辑示例 1：redis-py

redis-py 具有一个内置的重试机制，可在失败后立即重试一次。可以通过在创建 Redis 对象时提供的 `retry_on_timeout` 参数启用此机制。在这里，我们演示了一种带指数回退和抖动的自定义重试机制。我们提交了一个拉取请求，以便在 redis-py (#1494) 中本机实施指数回退。将来，可能无需手动实施它。

```
def run_with_backoff(function, retries=5):
base_backoff = 0.1 # base 100ms backoff
max_backoff = 10 # sleep for maximum 10 seconds
tries = 0
while True:
try:
```

```
  return function()
except (ConnectionError, TimeoutError):
  if tries >= retries:
 raise
  backoff = min(max_backoff, base_backoff * (pow(2, tries) + random.random()))
  print(f"sleeping for {backoff:.2f}s")
  sleep(backoff)
  tries += 1
```

之后，您可以使用以下代码来设置值：

```
client = redis.Redis(connection_pool=redis.BlockingConnectionPool(host=HOST,
 max_connections=10))
res = run_with_backoff(lambda: client.set("key", "value"))
print(res)
```

根据您的工作负载，您可能需要针对延迟敏感型工作负载将基本回退值从 1 秒更改为几十或几百毫秒。

回退逻辑示例 2：PHPRedis

PHPRedis 具有一个内置的重试机制，允许最多重试 10 次（不可配置）。可以配置两次尝试之间的延迟（从第二次重试开始会有抖动）。有关更多信息，请参阅以下示例代码。我们提交了一个拉取请求，以便在 PHPredis (#1986) 中本机实施已合并且记录的指数回退。对于最新版本的 PHPRedis 中的指数回退，无需手动实施，但我们在此处提供了早期版本中指数回退的参考。目前，以下是配置重试机制延迟的代码示例：

```
$timeout = 0.1; // 100 millisecond connection timeout
$retry_interval = 100; // 100 millisecond retry interval
$client = new Redis();
if($client->pconnect($HOST, $PORT, $timeout, NULL, $retry_interval) != TRUE) {
 return; // ERROR: connection failed
}
$client->set($key, $value);
```

回退逻辑示例 3：Lettuce

Lettuce 具有基于指数回退和抖动文章中描述的指数回退策略的内置重试机制。以下是显示完整抖动方法的代码摘录：

```
public static void main(String[] args)
```

```
{
 ClientResources resources = null;
 RedisClient client = null;

 try {
  resources = DefaultClientResources.builder()
    .reconnectDelay(Delay.fullJitter(
   Duration.ofMillis(100),     // minimum 100 millisecond delay
   Duration.ofSeconds(5),      // maximum 5 second delay
   100, TimeUnit.MILLISECONDS) // 100 millisecond base
  ).build();

  client = RedisClient.create(resources, RedisURI.create(HOST, PORT));
  client.setOptions(ClientOptions.builder()
 .socketOptions(SocketOptions.builder().connectTimeout(Duration.ofMillis(100)).build()) //
 100 millisecond connection timeout
 .timeoutOptions(TimeoutOptions.builder().fixedTimeout(Duration.ofSeconds(5)).build()) //
 5 second command timeout
 .build());

     // use the connection pool from above example
 } finally {
  if (connection != null) {
   connection.close();
  }

  if (client != null){
   client.shutdown();
  }

  if (resources != null){
   resources.shutdown();
  }

 }
}
```

配置客户端超时

适当地配置客户端超时，使服务器有足够的时间来处理请求并生成响应。如果无法建立与服务器的连接，这也将允许它快速失效。某些 Redis 命令的计算成本高于其他命令。例如，包含多条必须以原子方式运行的命令的 Lua 脚本或 MULTI/EXEC 事务。通常，建议使用更大的客户端超时，以避免客户端在收到来自服务器的响应之前超时，包括：

- 在多个键上运行命令
- 运行由多条 Redis 命令组成的 MULTI/EXEC 事务或 Lua 脚本
- 读取较大的值
- 执行诸如 BLPOP 之类的阻止操作

对于诸如 BLPOP 之类的阻止操作，最佳实践是将命令超时设置为小于套接字超时的数字。

以下是在 redis-py、PHPRedis 和 Lettuce 中实施客户端超时的代码示例。

超时配置示例 1：redis-py

以下是 redis-py 的代码示例：

```
# connect to Redis server with a 100 millisecond timeout
# give every Redis command a 2 second timeout
client = redis.Redis(connection_pool=redis.BlockingConnectionPool(host=HOST,
 max_connections=10,socket_connect_timeout=0.1,socket_timeout=2))

res = client.set("key", "value") # will timeout after 2 seconds
print(res)                       # if there is a connection error

res = client.blpop("list", timeout=1) # will timeout after 1 second
                                      # less than the 2 second socket timeout
print(res)
```

超时配置示例 2：PHPRedis

以下是 PHPRedis 的代码示例：

```
// connect to Redis server with a 100ms timeout
// give every Redis command a 2s timeout
$client = new Redis();
$timeout = 0.1; // 100 millisecond connection timeout
$retry_interval = 100; // 100 millisecond retry interval
$client = new Redis();
if($client->pconnect($HOST, $PORT, 0.1, NULL, 100, $read_timeout=2) != TRUE){
 return; // ERROR: connection failed
}
$client->set($key, $value);
```

```
$res = $client->set("key", "value"); // will timeout after 2 seconds
print "$res\n";                      // if there is a connection error

$res = $client->blpop("list", 1); // will timeout after 1 second
print "$res\n";                   // less than the 2 second socket timeout
```

超时配置示例 3：Lettuce

以下是 Lettuce 的代码示例：

```
// connect to Redis server and give every command a 2 second timeout
public static void main(String[] args)
{
 RedisClient client = null;
 StatefulRedisConnection<String, String> connection = null;
 try {
  client = RedisClient.create(RedisURI.create(HOST, PORT));
  client.setOptions(ClientOptions.builder()
 .socketOptions(SocketOptions.builder().connectTimeout(Duration.ofMillis(100)).build()) //
 100 millisecond connection timeout
 .timeoutOptions(TimeoutOptions.builder().fixedTimeout(Duration.ofSeconds(2)).build()) //
 2 second command timeout
 .build());

  // use the connection pool from above example

  commands.set("key", "value"); // will timeout after 2 seconds
  commands.blpop(1, "list"); // BLPOP with 1 second timeout
 } finally {
  if (connection != null) {
   connection.close();
  }

  if (client != null){
   client.shutdown();
  }
 }
}
```

配置服务器端空闲超时

我们观察到客户的应用程序连接了大量空闲客户端，但未主动发送命令的情况。在此类情况下，您可能会用尽所有 65000 个连接，并有大量空闲客户端。为了避免出现此类情况，可通过 Redis 特定的参数

在服务器上适当配置超时设置。这将确保服务器主动断开空闲客户端的连接，以避免连接增多。此设置不适用于无服务器缓存。

Redis Lua 脚本

Redis 支持 200 多条命令，包括用于运行 Lua 脚本的命令。不过，对于 Lua 脚本，有几个缺陷可能会影响 Redis 的内存和可用性。

非参数化 Lua 脚本

每个 Lua 脚本在运行之前都会在 Redis 服务器上进行缓存。非参数化 Lua 脚本是独有的，这可能会导致 Redis 服务器存储大量 Lua 脚本并占用更多内存。为了减轻此情况，请确保所有 Lua 脚本都已参数化，并在需要时定期执行 SCRIPT FLUSH 来清除缓存的 Lua 脚本。

以下示例说明如何定义和使用参数化脚本。首先，我们提供了一个非参数化方法的示例，它会生成三个不同的缓存 Lua 脚本，建议不使用此方法：

```
eval "return redis.call('set','key1','1')" 0
eval "return redis.call('set','key2','2')" 0
eval "return redis.call('set','key3','3')" 0
```

相反，请使用以下模式来创建能够接受传递的参数的单个脚本：

```
eval "return redis.call('set',KEYS[1],ARGV[1])" 1 key1 1
eval "return redis.call('set',KEYS[1],ARGV[1])" 1 key2 2
eval "return redis.call('set',KEYS[1],ARGV[1])" 1 key3 3
```

长时间运行的 Lua 脚本

Lua 脚本可以原子方式运行多条命令，因此它们的完成时间长于常用 Redis 命令的完成时间。如果 Lua 脚本仅运行只读操作，则可中途将其停止。不过，一旦 Lua 脚本执行写入操作，就无法将其终止，必须运行直至完成。如果长时间运行的 Lua 脚本发生突变，则会导致 Redis 服务器长时间无法响应。要缓解此问题，请避免长时间运行的 Lua 脚本，并在预生产环境中测试脚本。

带 Stealth 写入的 Lua 脚本

即使 Redis 超出 `maxmemory`，Lua 脚本也可通过以下几种方式继续向 Redis 写入新数据：

- 该脚本在 Redis 服务器低于 `maxmemory` 时启动，并包含多个写入操作
- 脚本的第一条写入命令不占用内存（例如 DEL），后跟的多个写入操作会占用内存

- 可以通过在 `noeviction` 之外的 Redis 服务器中配置适当的驱逐策略来缓解此问题。这将允许 Redis 在 Lua 脚本之间驱逐项目并释放内存。

存储大型复合项目

在某些情况下，应用程序可能会在 Redis 中存储大型复合项目（例如多 GB 哈希数据集）。建议不要这样做，因为这经常会导致 Redis 中出现性能问题。例如，客户端可以执行 HGETALL 命令来检索整个多 GB 哈希集合。这可能会给在客户端输出缓冲区中缓冲大型项目的 Redis 服务器带来巨大的内存压力。此外，对于集群模式下的槽迁移，ElastiCache 不迁移包含序列化大小超过 256 MB 的项目的槽。

为了解决大型项目问题，我们建议：

- 将大型复合项目分解成多个小型项目。例如，将一个大型哈希集合分解成多个单独的键值字段，其键名架构恰当地反映了该集合，例如在键名中使用公共前缀来标识项目集合。如果您必须以原子方式访问同一集合中的多个字段，则可以使用 MGET 命令在同一个命令中检索多个键值。
- 如果您评估了所有选项，但仍无法分解大型数据集，请尝试使用对集合中的部分数据而不是整个集合运行的命令。避免出现要求您以原子方式在同一命令中检索整个多 GB 集合的使用案例。一个示例是，在哈希集合上使用 HGET 或 HMGET 命令而不是使用 HGETALL 命令。

Lettuce 客户端配置

本节介绍建议的 Java 和 Lettuce 配置选项，以及它们如何应用于 ElastiCache 集群。

本节中的建议已在 Lettuce 版本 6.2.2 中进行测试。

主题

- 示例：启用集群模式和 TLS 的 Lettuce 配置
- 示例：禁用集群模式并启用 TLS 的 Lettuce 配置

Java DNS 缓存 TTL

Java 虚拟机 (JVM) 缓存 DNS 名称查找。当 JVM 将主机名解析为 IP 地址时，它会在指定时间段内 (称为生存时间 (TTL)) 缓存 IP 地址。

选择 TTL 值就是在延迟和对变化的响应能力之间进行权衡。TTL 越短，DNS 解析器就能越快注意到集群 DNS 中的更新。这样，您的应用程序就能更快地响应集群所经历的替换或其他工作流。但是，如

果 TTL 过低，将会增加查询量，从而增加应用程序的延迟。虽然没有正确的 TTL 值，但在设置 TTL 值时，值得考虑您可以用来等待更改生效的时间长度。

由于 ElastiCache 节点使用可能会变更的 DNS 名称条目，因此建议您为 JVM 配置一个 5 到 10 秒的低 TTL 值。这可确保在节点的 IP 地址发生更改时，您的应用程序将能够通过重新查询 DNS 条目来接收和使用资源的新 IP 地址。

对于一些 Java 配置，将设置 JVM 默认 TTL，以便在重新启动 JVM 之前绝不刷新 DNS 条目。

有关如何设置 JVM TTL 的详细信息，请参阅如何设置 JVM TTL。

Lettuce 版本

我们建议使用 Lettuce 版本 6.2.2 或更高版本。

端点

当您使用启用集群模式的集群时，将 `redisUri` 设置为集群配置终端节点。此 URI 的 DNS 查询将返回集群中所有可用节点的列表，并在集群初始化期间随机解析为其中一个节点。有关拓扑刷新工作原理的更多详细信息，请参阅本主题后面的 dynamicRefreshResources。

SocketOption

启用 KeepAlive。启用此选项可减少在命令运行时期间处理失败连接的需求。

确保根据应用程序要求和工作负载设置连接超时。有关更多信息，请参阅本主题后面的“超时”部分。

ClusterClientOption：启用集群模式的客户端选项

连接丢失时启用 AutoReconnect。

设置 CommandTimeout。有关更多详细信息，请参阅本主题后面的“超时”部分。

设置 nodeFilter 以从拓扑中筛选掉故障节点。Lettuce 将“集群节点”输出中找到的所有节点（包括处于 PFAIL/FAIL 状态的节点）保存在客户端的“分区”（也称为分片）中。在创建群集拓扑的过程中，它会尝试连接到所有分区节点。当节点因任何原因被替换时，Lettuce 这种添加故障节点的行为可能会导致连接错误（或警告）。

例如，在故障转移完成且集群启动恢复过程后，当刷新 clusterTopology 时，集群总线节点映射会在短时间内将故障节点列为 FAIL 节点，然后才会将其从拓扑中完全删除。在此期间，Lettuce Redis 客户端会将其视为正常的节点并不断与其连接。这会导致在重试用尽后出现故障。

例如：

```
final ClusterClientOptions clusterClientOptions =
    ClusterClientOptions.builder()
    ... // other options
    .nodeFilter(it ->
        ! (it.is(RedisClusterNode.NodeFlag.FAIL)
        || it.is(RedisClusterNode.NodeFlag.EVENTUAL_FAIL)
        || it.is(RedisClusterNode.NodeFlag.HANDSHAKE)
        || it.is(RedisClusterNode.NodeFlag.NOADDR)))
    .validateClusterNodeMembership(false)
    .build();
redisClusterClient.setOptions(clusterClientOptions);
```

> **Note**
>
> 节点筛选最好是在将 DynamicRefreshSources 设置为 true 时使用。否则，如果拓扑视图取自单个问题种子节点（认为某个分片的主节点出现故障），则它将会筛选掉该主节点，从而导致插槽未被覆盖。拥有多个种子节点（当 DynamicRefreshSources 为 true 时）可以减小出现此问题的可能性，因为在使用新提升的主节点进行故障转移后，至少某些种子节点会有更新的拓扑视图。

ClusterTopologyRefreshOptions：用于控制启用集群模式的客户端刷新集群拓扑的选项

> **Note**
>
> 已禁用集群模式的集群不支持集群发现命令，并且与所有客户端的动态拓扑发现功能不兼容。使用 ElastiCache 禁用的集群模式与 Lettuce 的 `MasterSlaveTopologyRefresh` 不兼容。相反，如果禁用了集群模式，则可以配置 `StaticMasterReplicaTopologyProvider` 并提供集群读取和写入端点。
>
> 有关连接到已禁用集群模式的集群的更多信息，请参阅查找 Redis（已禁用集群模式）集群的端点（控制台）。
>
> 如果您想使用 Lettuce 的动态拓扑发现功能，则可以使用与现有集群相同的分片配置创建启用集群模式的集群。但是，对于启用集群模式的集群，我们建议至少配置 3 个分片以及至少一个副本，以支持快速失效转移。

启用 enablePeriodicRefresh。这将启用定期集群拓扑更新，以便客户端按照 refreshPeriod 的间隔（默认为 60 秒）更新集群拓扑。如果禁用，则只有在客户端尝试对集群运行命令时出现错误的情况下，才会更新集群拓扑。

启用此选项后，您可以通过将此作业添加到后台任务来减少与刷新集群拓扑相关的延迟。虽然拓扑刷新是在后台作业中执行的，但对于具有多个节点的集群来说，拓扑刷新可能会有些慢。这是因为将会查询所有节点的视图以获取最新的集群视图。如果您运行大型集群，则可能需要延长间隔。

启用 enableAllAdaptiveRefreshTriggers。这将启用使用所有触发器自适应拓扑刷新：MMOVED_REDIRECT、ASK_REDIRECT、PERSISTENT_RECONNECTS、UNCOVERED_SLOT、UNKNO自适应刷新触发器根据 Redis 集群操作期间发生的事件启动拓扑视图更新。当发生上述触发器之一时，启用此选项会导致立即刷新拓扑。自适应触发刷新使用超时限制速率，因为事件可能会大规模发生（更新之间的默认超时时间为 30）。

启用 closeStaleConnections。这可以在刷新集群拓扑时关闭过时的连接。只有在 ClusterTopologyRefreshOptions.isPeriodicRefreshEnabled() 为 true 时，它才会生效。启用此选项后，客户端可以关闭过时的连接并在后台创建新连接。这减少了在命令运行时期间处理失败连接的需求。

启用 dynamicRefreshResources。我们建议为小型集群启用 dynamicRefreshResources，为大型集群禁用 dynamicRefreshResources。dynamicRefreshResources 支持从提供的种子节点（例如，群集配置终端节点）发现集群节点。它使用所有发现的节点作为刷新集群拓扑的源。

使用动态刷新查询所有已发现的集群拓扑节点，并尝试选择最准确的集群视图。如果将其设置为 false，则仅使用初始种子节点作为拓扑发现的源，并且仅获取初始种子节点的客户端数量。禁用后，如果将集群配置终端节点解析为故障节点，则尝试刷新集群视图会失败并导致异常。之所以会发生这种情况，是因为从集群配置端点中删除故障节点的条目需要一些时间。因此，仍然会在短时间内将配置终端节点随机解析为故障节点。

但是，启用后，我们会使用从集群视图接收到的所有集群节点来查询其当前视图。因为我们从该视图中筛选掉了故障的节点，所以拓扑刷新将会成功。但是，当 dynamicRefreshSources 为 true 时，Lettuce 会查询所有节点来获取集群视图，然后比较结果。因此，对于拥有大量节点的集群来说，这可能会很昂贵。我们建议您为具有多个节点的集群关闭此功能。

```
final ClusterTopologyRefreshOptions topologyOptions =
    ClusterTopologyRefreshOptions.builder()
    .enableAllAdaptiveRefreshTriggers()
    .enablePeriodicRefresh()
    .dynamicRefreshSources(true)
    .build();
```

ClientResources

配置 DnsResolver 与 DirContextDnsResolver。DNS 解析器基于 Java 的 com.sun.jndi.dns.DnsContextFactory。

为 reconnectDelay 配置指数回退和完全抖动。Lettuce 具有基于指数回退策略的内置重试机制。有关详细信息，请参阅 AWS 架构博客上的指数回退和抖动。有关具备重试回退策略的重要性的更多信息，请参阅 AWS 数据库博客上最佳实践博客文章的回退逻辑部分。

```
ClientResources clientResources = DefaultClientResources.builder()
   .dnsResolver(new DirContextDnsResolver())
    .reconnectDelay(
        Delay.fullJitter(
            Duration.ofMillis(100),     // minimum 100 millisecond delay
            Duration.ofSeconds(10),      // maximum 10 second delay
            100, TimeUnit.MILLISECONDS)) // 100 millisecond base
    .build();
```

超时

使用比您的命令超时更低的连接超时值。Lettuce 使用延迟连接建立。因此，在连接超时高于命令超时的情况下，如果 Lettuce 尝试连接到不正常的节点并且总是超过命令超时，则拓扑刷新后可能会持续失败一段时间。

对不同的命令使用动态命令超时。建议您根据命令预期时长设置命令超时。例如，对遍历多个键的命令（例如 FLUSHDB、FLUSHALL、KEYS、SMEMBERS 或 Lua 脚本）使用较长的超时。对单键命令（例如 SET、GET 和 HSET）使用较短的超时。

> **Note**
>
> 以下示例中配置的超时适用于运行键和值长度最多为 20 字节的 SET/GET 命令的测试。当命令较复杂或键和值较大时，处理时间可能会更长。您应该根据应用程序的用例设置超时。

```
private static final Duration META_COMMAND_TIMEOUT = Duration.ofMillis(1000);
private static final Duration DEFAULT_COMMAND_TIMEOUT = Duration.ofMillis(250);
// Socket connect timeout should be lower than command timeout for Lettuce
private static final Duration CONNECT_TIMEOUT = Duration.ofMillis(100);

SocketOptions socketOptions = SocketOptions.builder()
    .connectTimeout(CONNECT_TIMEOUT)
    .build();
```

```
class DynamicClusterTimeout extends TimeoutSource {
     private static final Set<ProtocolKeyword> META_COMMAND_TYPES =
 ImmutableSet.<ProtocolKeyword>builder()
          .add(CommandType.FLUSHDB)
          .add(CommandType.FLUSHALL)
          .add(CommandType.CLUSTER)
          .add(CommandType.INFO)
          .add(CommandType.KEYS)
          .build();

    private final Duration defaultCommandTimeout;
    private final Duration metaCommandTimeout;

    DynamicClusterTimeout(Duration defaultTimeout, Duration metaTimeout)
    {
        defaultCommandTimeout = defaultTimeout;
        metaCommandTimeout = metaTimeout;
    }

    @Override
    public long getTimeout(RedisCommand<?, ?, ?> command) {
        if (META_COMMAND_TYPES.contains(command.getType())) {
            return metaCommandTimeout.toMillis();
        }
        return defaultCommandTimeout.toMillis();
    }
}

// Use a dynamic timeout for commands, to avoid timeouts during
// cluster management and slow operations.
TimeoutOptions timeoutOptions = TimeoutOptions.builder()
.timeoutSource(
    new DynamicClusterTimeout(DEFAULT_COMMAND_TIMEOUT, META_COMMAND_TIMEOUT))
.build();
```

示例：启用集群模式和 TLS 的 Lettuce 配置

> **Note**
>
> 以下示例中的超时适用于运行键和值长度最多 20 字节的 SET/GET 命令的测试。当命令较复杂或键和值较大时，处理时间可能会更长。您应该根据应用程序的用例设置超时。

```
// Set DNS cache TTL
public void setJVMProperties() {
    java.security.Security.setProperty("networkaddress.cache.ttl", "10");
}

private static final Duration META_COMMAND_TIMEOUT = Duration.ofMillis(1000);
private static final Duration DEFAULT_COMMAND_TIMEOUT = Duration.ofMillis(250);
// Socket connect timeout should be lower than command timeout for Lettuce
private static final Duration CONNECT_TIMEOUT = Duration.ofMillis(100);

// Create RedisURI from the cluster configuration endpoint
clusterConfigurationEndpoint = <cluster-configuration-endpoint> // TODO: add your
 cluster configuration endpoint
final RedisURI redisUriCluster =
    RedisURI.Builder.redis(clusterConfigurationEndpoint)
        .withPort(6379)
        .withSsl(true)
        .build();

// Configure the client's resources
ClientResources clientResources = DefaultClientResources.builder()
    .reconnectDelay(
        Delay.fullJitter(
            Duration.ofMillis(100),     // minimum 100 millisecond delay
            Duration.ofSeconds(10),      // maximum 10 second delay
            100, TimeUnit.MILLISECONDS)) // 100 millisecond base
    .dnsResolver(new DirContextDnsResolver())
    .build();

// Create a cluster client instance with the URI and resources
RedisClusterClient redisClusterClient =
    RedisClusterClient.create(clientResources, redisUriCluster);

// Use a dynamic timeout for commands, to avoid timeouts during
// cluster management and slow operations.
class DynamicClusterTimeout extends TimeoutSource {
     private static final Set<ProtocolKeyword> META_COMMAND_TYPES =
 ImmutableSet.<ProtocolKeyword>builder()
          .add(CommandType.FLUSHDB)
          .add(CommandType.FLUSHALL)
          .add(CommandType.CLUSTER)
          .add(CommandType.INFO)
          .add(CommandType.KEYS)
```

```
            .build();

    private final Duration metaCommandTimeout;
    private final Duration defaultCommandTimeout;

    DynamicClusterTimeout(Duration defaultTimeout, Duration metaTimeout)
    {
        defaultCommandTimeout = defaultTimeout;
        metaCommandTimeout = metaTimeout;
    }

    @Override
    public long getTimeout(RedisCommand<?, ?, ?> command) {
        if (META_COMMAND_TYPES.contains(command.getType())) {
            return metaCommandTimeout.toMillis();
        }
        return defaultCommandTimeout.toMillis();
    }
}

TimeoutOptions timeoutOptions = TimeoutOptions.builder()
    .timeoutSource(new DynamicClusterTimeout(DEFAULT_COMMAND_TIMEOUT,
 META_COMMAND_TIMEOUT))
     .build();

// Configure the topology refreshment optionts
final ClusterTopologyRefreshOptions topologyOptions =
    ClusterTopologyRefreshOptions.builder()
    .enableAllAdaptiveRefreshTriggers()
    .enablePeriodicRefresh()
    .dynamicRefreshSources(true)
    .build();

// Configure the socket options
final SocketOptions socketOptions =
    SocketOptions.builder()
    .connectTimeout(CONNECT_TIMEOUT)
    .keepAlive(true)
    .build();

// Configure the client's options
final ClusterClientOptions clusterClientOptions =
    ClusterClientOptions.builder()
    .topologyRefreshOptions(topologyOptions)
```

```
    .socketOptions(socketOptions)
    .autoReconnect(true)
    .timeoutOptions(timeoutOptions)
    .nodeFilter(it ->
        ! (it.is(RedisClusterNode.NodeFlag.FAIL)
        || it.is(RedisClusterNode.NodeFlag.EVENTUAL_FAIL)
        || it.is(RedisClusterNode.NodeFlag.NOADDR)))
    .validateClusterNodeMembership(false)
    .build();

redisClusterClient.setOptions(clusterClientOptions);

// Get a connection
final StatefulRedisClusterConnection<String, String> connection =
    redisClusterClient.connect();

// Get cluster sync/async commands
RedisAdvancedClusterCommands<String, String> sync = connection.sync();
RedisAdvancedClusterAsyncCommands<String, String> async = connection.async();
```

示例：禁用集群模式并启用 TLS 的 Lettuce 配置

> **Note**
>
> 以下示例中的超时适用于运行键和值长度最多 20 字节的 SET/GET 命令的测试。当命令较复杂或键和值较大时，处理时间可能会更长。您应该根据应用程序的用例设置超时。

```
// Set DNS cache TTL
public void setJVMProperties() {
    java.security.Security.setProperty("networkaddress.cache.ttl", "10");
}

private static final Duration META_COMMAND_TIMEOUT = Duration.ofMillis(1000);
private static final Duration DEFAULT_COMMAND_TIMEOUT = Duration.ofMillis(250);
// Socket connect timeout should be lower than command timeout for Lettuce
private static final Duration CONNECT_TIMEOUT = Duration.ofMillis(100);

// Create RedisURI from the primary/reader endpoint
clusterEndpoint = <primary/reader-endpoint> // TODO: add your node endpoint
RedisURI redisUriStandalone =
```

```
 RedisURI.Builder.redis(clusterEndpoint).withPort(6379).withSsl(true).withDatabase(0).build();

ClientResources clientResources =
    DefaultClientResources.builder()
        .dnsResolver(new DirContextDnsResolver())
        .reconnectDelay(
            Delay.fullJitter(
                Duration.ofMillis(100), // minimum 100 millisecond delay
                Duration.ofSeconds(10), // maximum 10 second delay
                100,
                TimeUnit.MILLISECONDS)) // 100 millisecond base
        .build();

// Use a dynamic timeout for commands, to avoid timeouts during
// slow operations.
class DynamicTimeout extends TimeoutSource {
     private static final Set<ProtocolKeyword> META_COMMAND_TYPES =
 ImmutableSet.<ProtocolKeyword>builder()
          .add(CommandType.FLUSHDB)
          .add(CommandType.FLUSHALL)
          .add(CommandType.INFO)
          .add(CommandType.KEYS)
          .build();

    private final Duration metaCommandTimeout;
    private final Duration defaultCommandTimeout;

    DynamicTimeout(Duration defaultTimeout, Duration metaTimeout)
    {
        defaultCommandTimeout = defaultTimeout;
        metaCommandTimeout = metaTimeout;
    }

    @Override
    public long getTimeout(RedisCommand<?, ?, ?> command) {
        if (META_COMMAND_TYPES.contains(command.getType())) {
            return metaCommandTimeout.toMillis();
        }
        return defaultCommandTimeout.toMillis();
    }
}

TimeoutOptions timeoutOptions = TimeoutOptions.builder()
```

```
    .timeoutSource(new DynamicTimeout(DEFAULT_COMMAND_TIMEOUT, META_COMMAND_TIMEOUT))
     .build();

final SocketOptions socketOptions =
    SocketOptions.builder().connectTimeout(CONNECT_TIMEOUT).keepAlive(true).build();

ClientOptions clientOptions =

 ClientOptions.builder().timeoutOptions(timeoutOptions).socketOptions(socketOptions).build();

RedisClient redisClient = RedisClient.create(clientResources, redisUriStandalone);
redisClient.setOptions(clientOptions);
```

IPv6 客户端示例

以下是使用常用的开源客户端库与支持 IPv6 的 ElastiCache 资源进行交互的最佳实践。您可以查看与交互的现有最佳实践， ElastiCache以获取有关为 ElastiCache 资源配置客户端的建议。但是，在与启用 IPv6 的资源进行交互时，有一些注意事项值得注意。

经过验证的客户端

ElastiCache 与开源 Redis 兼容。这意味着支持 IPv6 连接的开源 Redis 客户端应该能够连接到为 Redis 集群启用 ElastiCache 的 IPv6。此外，一些最受欢迎的 Python 和 Java 客户端已经过专门的测试和验证，可以与所有支持的网络类型配置（仅 IPv4、仅 IPv6 和双堆栈）配合使用

经过验证的客户端：

- Redis Py () – 4.1.2
- Lettuce – 版本：6.1.6.RELEASE
- Jedis – 版本：3.6.0

为双堆栈集群配置首选协议

对于启用了集群模式的 Redis 集群，您可以使用 IP 发现参数控制客户端将用于连接到集群中的节点的协议。IP 发现参数可以设置为 IPv4 或 IPv6。

对于 Redis 集群，IP 发现参数可设置 cluster slots ()、cluster shards () 和 cluster nodes () 输出中使用的 IP 协议。客户端使用这些命令来发现集群拓扑。客户端使用这些命令中的 IP 连接到集群中的其他节点。

更改 IP 发现不会导致连接的客户端出现任何停机。但是，更改需要一些时间才能传播。要确定 Redis 集群的更改何时完全传播，请监控 `cluster slots` 的输出。一旦 cluster slots 命令返回的所有节点报告了使用新协议的 IP，就表明更改完成了传播。

使用 Redis-Py 的示例：

```
cluster = RedisCluster(host="xxxx", port=6379)
target_type = IPv6Address # Or IPv4Address if changing to IPv4

nodes = set()
while len(nodes) == 0 or not all((type(ip_address(host)) is target_type) for host in
 nodes):
    nodes = set()

   # This refreshes the cluster topology and will discovery any node updates.
   # Under the hood it calls cluster slots
    cluster.nodes_manager.initialize()
    for node in cluster.get_nodes():
        nodes.add(node.host)
    self.logger.info(nodes)

    time.sleep(1)
```

使用 Lettuce 的示例：

```
RedisClusterClient clusterClient = RedisClusterClient.create(RedisURI.create("xxxx",
 6379));

Class targetProtocolType = Inet6Address.class; // Or Inet4Address.class if you're
 switching to IPv4

Set<String> nodes;

do {
   // Check for any changes in the cluster topology.
   // Under the hood this calls cluster slots
    clusterClient.refreshPartitions();
    Set<String> nodes = new HashSet<>();

    for (RedisClusterNode node : clusterClient.getPartitions().getPartitions()) {
        nodes.add(node.getUri().getHost());
    }
```

```
    Thread.sleep(1000);
} while (!nodes.stream().allMatch(node -> {
            try {
                return finalTargetProtocolType.isInstance(InetAddress.getByName(node));
            } catch (UnknownHostException ignored) {}
            return false;
}));
```

启用 TLS 的双堆栈 ElastiCache 集群

为 ElastiCache 集群启用 TLS 后，集群发现功能（`cluster slots`、`cluster shards`和`cluster nodes`）将返回主机名而不是 IP。然后使用主机名而不是 IP 来连接到 ElastiCache 集群并执行 TLS 握手。这意味着客户端不会受到 IP 发现参数的影响。对于启用 TLS 的集群，IP 发现参数对首选 IP 协议没有影响。相反，使用的 IP 协议将取决于客户端在解析 DNS 主机名时首选的 IP 协议。

Java 客户端

从同时支持 IPv4 和 IPv6 的 Java 环境进行连接时，为了实现向后兼容，Java 默认情况下会优先使用 IPv4 而不是 IPv6。但是，IP 协议首选项可通过 JVM 参数进行配置。要首选 IPv4，JVM 会接受 `-Djava.net.preferIPv4Stack=true` 并首选 IPv6 集 `-Djava.net.preferIPv6Stack=true`。设置 `-Djava.net.preferIPv4Stack=true` 意味着 JVM 将不再建立任何 IPv6 连接。将这些包括到其他非 Redis 应用程序。

主机级别首选项

通常，如果客户端或客户端运行时系统不提供用于设置 IP 协议首选项的配置选项，则在执行 DNS 解析时，IP 协议将取决于主机的配置。默认情况下，大多数主机更喜欢 IPv6 而不是 IPv4，但可以在主机级别配置此首选项。这将影响来自该主机的所有 DNS 请求，而不仅仅是对 ElastiCache 群集的请求。

Linux 主

对于 Linux，可以通过修改`gai.conf`文件来配置 IP 协议首选项。可以在下方找到该`gai.conf`文件`/etc/gai.conf`。如果未`gai.conf`指定，则应提供一个示例`/usr/share/doc/glibc-common-x.xx/gai.conf`，可以在其下复制到，`/etc/gai.conf`然后应取消注释默认配置。要将配置更新为在连接到集群时首选 IPv4，请将包含 ElastiCache 集群 IP 的 CIDR 范围的优先级更新为高于默认 IPv6 连接的优先级。默认情况下，IPv6 连接的优先级为 40。例如，假设集群位于 CIDR 172.31.0.0:0/16 的子网中，则以下配置将导致客户端优先使用该集群的 IPv4 连接。

```
label ::1/128       0
```

```
label ::/0          1
label 2002::/16     2
label ::/96         3
label ::ffff:0:0/96 4
label fec0::/10     5
label fc00::/7      6
label 2001:0::/32   7
label ::ffff:172.31.0.0/112 8
#
#    This default differs from the tables given in RFC 3484 by handling
#    (now obsolete) site-local IPv6 addresses and Unique Local Addresses.
#    The reason for this difference is that these addresses are never
#    NATed while IPv4 site-local addresses most probably are.  Given
#    the precedence of IPv6 over IPv4 (see below) on machines having only
#    site-local IPv4 and IPv6 addresses a lookup for a global address would
#    see the IPv6 be preferred.  The result is a long delay because the
#    site-local IPv6 addresses cannot be used while the IPv4 address is
#    (at least for the foreseeable future) NATed.  We also treat Teredo
#    tunnels special.
#
# precedence  <mask>   <value>
#    Add another rule to the RFC 3484 precedence table.  See section 2.1
#    and 10.3 in RFC 3484.  The default is:
#
precedence  ::1/128       50
precedence  ::/0          40
precedence  2002::/16     30
precedence ::/96          20
precedence ::ffff:0:0/96  10
precedence ::ffff:172.31.0.0/112 100
```

有关更多详细信息可gai.conf在 Linux 主页上找到

微软主机

Windows 主机的过程与此类似。对于 Windows 主机，你可以运行netsh interface ipv6 set prefix CIDR_CONTAINING_CLUSTER_IPS PRECEDENCE LABEL。这与修改 Linux 主机上的gai.conf文件效果相同。

这将更新优先级策略，使指定的 CIDR 范围优先于 IPv4 连接而不是 IPv6 连接。例如，假设群集位于子网中，执行 172.31.0.0:0/16 CIDRnetsh interface ipv6 set prefix ::ffff:172.31.0.0:0/112 100 15 将生成以下优先级表，这将导致客户端在连接到集群时首选 IPv4。

```
C:\Users\Administrator>netsh interface ipv6 show prefixpolicies
Querying active state...

Precedence Label Prefix
---------- ----- --------------------------------
100 15 ::ffff:172.31.0.0:0/112
20 4 ::ffff:0:0/96
50 0 ::1/128
40 1 ::/0
30 2 2002::/16
5 5 2001::/32
3 13 fc00::/7
1 11 fec0::/10
1 12 3ffe::/16
1 3 ::/96
```

管理预留内存

预留内存是为非数据使用情况留出的内存。执行备份或故障转移时，Redis 使用可用的内存来记录将集群数据写入 .rdb 文件时对集群执行的写入操作。如果您没有足够的内存可供所有写入使用，则进程失败。接下来，您可以找到有关管理 Redis 预留内存 ElastiCache 的选项以及如何应用这些选项的信息。

主题

您需要预留多少内存？

由于 ElastiCache 实现备份和复制过程的方式不同，经验法则是使用`reserved-memory-percent`参数保留节点类型`maxmemory`值的 25%。这是默认值，建议在大多数情况下使用。

当突发型微型和小型实例类型的运行接近`maxmemory`极限时，它们可能会遇到交换使用情况。为了提高备份、复制和高流量期间这些实例类型的运行可靠性，我们建议将小型实例类型的`reserved-memory-percent`参数值提高到 30%，对于微型实例类型，将该参数的值提高到 50%。

对于具有数据分层的 ElastiCache 集群上的写入密集型工作负载，我们建议`reserved-memory-percent`将节点可用内存最多增加到 50%。

有关更多信息，请参阅下列内容：

- 确保具有用于创建 Redis 快照的足够内存
- 如何实施同步和备份
- 数据分层

用于管理预留内存的参数

自 2017 年 3 月 16 日起，Amazon f ElastiCache or Redis 提供了两个相互排斥的参数来管理你的 Redis 内存，`reserved-memory`以及。`reserved-memory-percent`Redis 发行版中不包含这两个参数。

根据您成为 ElastiCache 客户的时间，这些参数中的一个或另一个是默认的内存管理参数。在您创建新的 Redis 集群或复制组并使用默认参数组时，此参数适用。

- 对于在 2017 年 3 月 16 日之前开始使用 ElastiCache 的客户 – 当您使用默认参数组创建 Redis 集群或复制组时，内存管理参数为 `reserved-memory`。在此情况下，将预留零 (0) 字节内存。
- 对于在 2017 年 3 月 16 日或之后开始使用 ElastiCache 的客户 – 当您使用默认参数组创建 Redis 集群或复制组时，内存管理参数为 `reserved-memory-percent`。在此情况下，将预留 25% 的节点 `maxmemory` 值用于非数据目的。

在了解了两个 Redis 内存管理参数后，您可能首选使用不是默认值或具有非默认值的该内存管理参数。如若如此，您可以更改为其他预留内存管理参数。

要更改该参数的值，您可以创建一个自定义参数组并对其进行修改以使用首选内存管理参数和值。然后，每当您创建新的 Redis 集群或复制组时，都可以使用自定义参数组。对于现有集群或复制组，您可以修改它们以使用自定义参数组。

有关更多信息，请参阅下列内容：

- 指定您的预留内存管理参数
- 创建参数组
- 修改参数组
- 修改集 ElastiCache 群
- 修改复制组

reserved-memory 参数

在 2017 年 3 月 16 日之前，所有 ElastiCache 的 Redis 预留内存管理都是使用参数`reserved-memory`完成的。`reserved-memory` 的默认值为 0。此默认值不为 Redis 开销预留内存，并允许 Redis 将所有节点内存用于数据。

您需要创建自定义参数组，才可更改 `reserved-memory` 以使您有足够的内存可用于备份和故障转移。在此自定义参数组中，您可将 `reserved-memory` 设置为适用于您的集群和集群节点类型上所运行的 Redis 版本的值。有关更多信息，请参阅 您需要预留多少内存？

f ElastiCache or Redis 参数特定`reserved-memory` ElastiCache于 Redis，不是 Redis 发行版的一部分。

以下过程显示如何使用 `reserved-memory` 来管理 Redis 集群上的内存。

使用 reserved-memory 预留内存

1. 创建一个自定义参数组，指定与正在运行的引擎版本匹配的参数组系列，例如，指定 redis2.8 参数组系列。有关更多信息，请参阅 创建参数组。

```
aws elasticache create-cache-parameter-group \
   --cache-parameter-group-name redis6x-m3xl \
   --description "Redis 2.8.x for m3.xlarge node type" \
   --cache-parameter-group-family redis6.x
```

2. 计算要为 Redis 开销预留的内存大小。在特定于 Redis 节点类型的参数中可找到适合您节点类型的 maxmemory 值。

3. 修改自定义参数组，使得参数 reserved-memory 为您在上一步中计算得到的字节数。以下 AWS CLI 示例假设您运行的是 2.8.22 之前的 Redis 版本，并且需要保留一半的节点。maxmemory有关更多信息，请参阅 修改参数组。

```
aws elasticache modify-cache-parameter-group \
   --cache-parameter-group-name redis28-m3xl \
   --parameter-name-values "ParameterName=reserved-memory,
 ParameterValue=7130316800"
```

   您使用的每个节点类型需要一个单独的自定义参数组，因为每个节点类型的 maxmemory 值不同。因此，每个节点类型需要不同的 reserved-memory 值。

4. 修改您的 Redis 集群或复制组以使用自定义参数组。

   以下 CLI 示例修改集群 my-redis-cluster 以立即开始使用自定义参数组 redis28-m3xl。有关更多信息，请参阅 修改集 ElastiCache 群。

```
aws elasticache modify-cache-cluster \
   --cache-cluster-id my-redis-cluster \
   --cache-parameter-group-name redis28-m3xl \
   --apply-immediately
```

   以下 CLI 示例修改复制组 my-redis-repl-grp 以立即开始使用自定义参数组 redis28-m3xl。有关更多信息，请参阅 修改复制组。

```
aws elasticache modify-replication-group \
   --replication-group-id my-redis-repl-grp \
   --cache-parameter-group-name redis28-m3xl \
```

```
   --apply-immediately
```

该 reserved-memory-percent 参数

2017 年 3 月 16 日，亚马逊 ElastiCache 推出了该参数，reserved-memory-percent并在所有版本的 Redis 上推出了该参数。 ElastiCache reserved-memory-percent 的用途是简化所有集群上的预留内存管理。这是因为它让您可以对各参数组系列 (例如 redis2.8) 使用单个参数组来管理集群的预留内存，而不管节点类型如何。reserved-memory-percent 的默认值是 25 (25%)。

f ElastiCache or Redis 参数特定reserved-memory-percent ElastiCache 于 Redis，不是 Redis 发行版的一部分。

如果您的集群使用 r6gd 系列的节点类型，且内存利用率达到 75%，则会自动触发数据分层。有关更多信息，请参阅 数据分层。

要预留内存，请使用 reserved-memory-percent

reserved-memory-percent要使用管理 for Redi ElastiCache s 集群上的内存，请执行以下操作之一：

- 如果您运行的是 Redis 2.8.22 或更高版本，请向集群分配默认参数组。默认值 25% 应已足够。否则，请执行以下所述步骤更改该值。
- 如果您运行的是 Redis 2.8.22 以前的版本，则可能需要预留比 reserved-memory-percent 默认值的 25% 更高的内存。为此，请使用以下过程。

要更改的百分比值 reserved-memory-percent

1. 创建一个自定义参数组，指定与正在运行的引擎版本匹配的参数组系列，例如，指定 redis2.8 参数组系列。由于您无法修改默认参数组，所以需要自定义参数组。有关更多信息，请参阅 创建参数组。

   ```
   aws elasticache create-cache-parameter-group \
      --cache-parameter-group-name redis28-50 \
      --description "Redis 2.8.x 50% reserved" \
      --cache-parameter-group-family redis2.8
   ```

   由于 reserved-memory-percent 以节点 maxmemory 的百分比来预留内存，因此您无需为每个节点类型设置一个自定义参数组。

2. 修改自定义参数组，使得 reserved-memory-percent 为 50 (50%)。有关更多信息，请参阅 修改参数组。

```
aws elasticache modify-cache-parameter-group \
   --cache-parameter-group-name redis28-50 \
   --parameter-name-values "ParameterName=reserved-memory-percent,
 ParameterValue=50"
```

3. 为任何运行的 Redis 版本早于 2.8.22 的 Redis 集群或复制组使用此自定义参数组。

   以下 CLI 示例修改 Redis 集群 my-redis-cluster 以立即开始使用自定义参数组 redis28-50。有关更多信息，请参阅 修改集 ElastiCache 群。

```
aws elasticache modify-cache-cluster \
   --cache-cluster-id my-redis-cluster \
   --cache-parameter-group-name redis28-50 \
   --apply-immediately
```

   以下 CLI 示例修改 Redis 复制组 my-redis-repl-grp 以立即开始使用自定义参数组 redis28-50。有关更多信息，请参阅 修改复制组。

```
aws elasticache modify-replication-group \
   --replication-group-id my-redis-repl-grp \
   --cache-parameter-group-name redis28-50 \
   --apply-immediately
```

指定您的预留内存管理参数

如果您是 2017 年 3 月 16 日的现有 ElastiCache 客户，则您的默认预留内存管理参数为reserved-memory零 (0) 字节的预留内存。如果您在 2017 年 3 月 16 日之后成为 ElastiCache 客户，则您的默认预留内存管理参数为reserved-memory-percent预留节点 25% 的内存。无论您何时创建 ElastiCache 适用于 Redis 的集群还是复制组，都是如此。但是，您可以使用 AWS CLI 或 ElastiCache API 更改您的预留内存管理参数。

参数 reserved-memory 和 reserved-memory-percent 互斥。参数组始终有这两个参数之一，但不能同时有它们两者。您可以通过修改参数组，更改参数组用于管理预留内存的参数。由于您无法修改默认参数组，因此参数组必须是自定义参数组。有关更多信息，请参阅 创建参数组。

要指定 reserved-memory-percent

要将 reserved-memory-percent 用作预留内存管理参数，请使用 modify-cache-parameter-group 命令修改自定义参数组。使用 parameter-name-values 参数指定 reserved-memory-percent 及其值。

以下 CLI 示例修改自定义参数组 redis32-cluster-on，以便使用 reserved-memory-percent 管理预留内存。必须为参数组的 ParameterValue 分配一个值，才能将 ParameterName 参数用于预留内存管理。有关更多信息，请参阅 修改参数组。

```
aws elasticache modify-cache-parameter-group \
   --cache-parameter-group-name redis32-cluster-on \
   --parameter-name-values "ParameterName=reserved-memory-percent, ParameterValue=25"
```

指定 reserved-memory

要将 reserved-memory 用作预留内存管理参数，请使用 modify-cache-parameter-group 命令修改自定义参数组。使用 parameter-name-values 参数指定 reserved-memory 及其值。

以下 CLI 示例修改自定义参数组 redis32-m3xl，以便使用 reserved-memory 管理预留内存。必须为参数组的 ParameterValue 分配一个值，才能将 ParameterName 参数用于预留内存管理。因为引擎版本比 2.8.22 新，所以我们将该值设置为 3565158400，它是 cache.m3.xlarge 的 maxmemory 的 25%。有关更多信息，请参阅 修改参数组。

```
aws elasticache modify-cache-parameter-group \
   --cache-parameter-group-name redis32-m3xl \
   --parameter-name-values "ParameterName=reserved-memory, ParameterValue=3565158400"
```

使用自行设计的集群时的最佳实践

此部分仅在您选择设计自己的 Redis 集群时适用。建议您查看并遵循这些最佳实践。

主题

- 利用多可用区最大限度减少停机时间
- 确保具有用于创建 Redis 快照的足够内存
- 在线集群大小调整
- 最大程度减少维护期间的停机时间

利用多可用区最大限度减少停机时间

要详细了解多可用区并最大限度地减少停机时间，请参阅使用多可用区的 Redis 最大限度地减少停机时间。 ElastiCache

确保具有用于创建 Redis 快照的足够内存

2.8.22 版本及更高版本中的 Redis 快照和同步

Redis 2.8.22 引入了无分支的保存过程，使您能够在同步和保存期间，将更多内存分配给应用程序使用而不会增加交换分区使用率。有关更多信息，请参阅 如何实施同步和备份。

2.8.22 版本之前的版本中的 Redis 快照和同步

当你使用 Redis 时 ElastiCache，Redis 会在许多情况下调用后台写入命令：

- 为备份创建快照时。
- 将副本与复制组中的主副本同步时。
- 为 Redis 启用仅附加文件功能 (AOF) 时。
- 将副本提升为主快照时（这会导致主集群/副本同步）。

每当 Redis 执行后台写入进程时，您都必须有足够的可用内存来处理进程开销。内存不足会导致该进程失败。因此，重要的是在创建 Redis 集群时应选择有足够内存的节点实例类型。

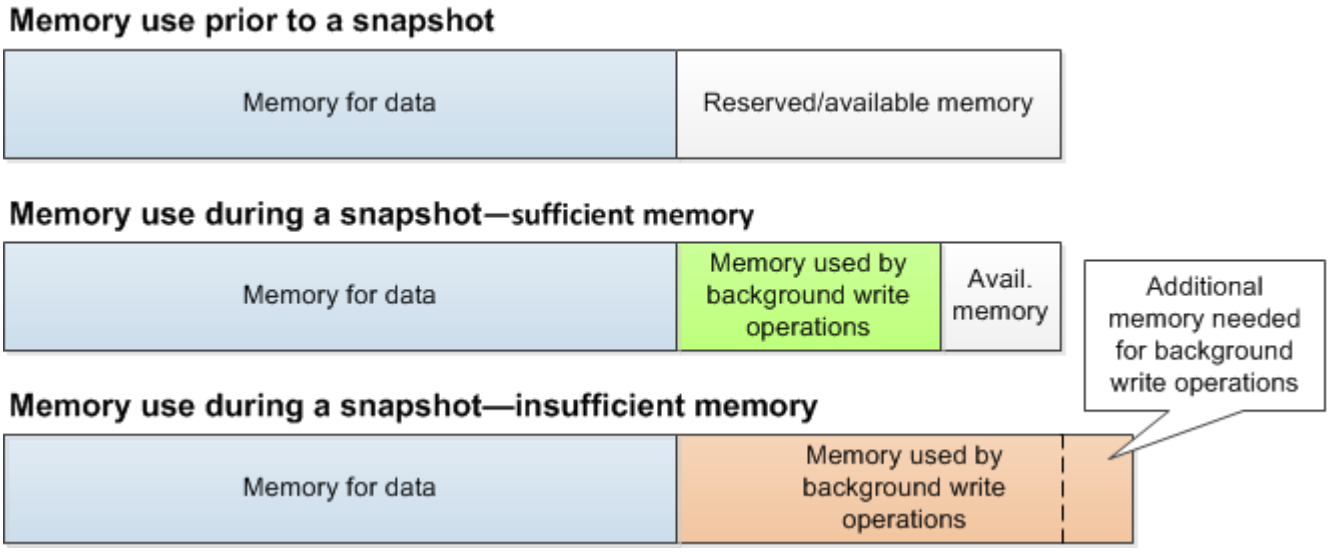
后台写入进程和内存使用情况

每当调用后台写入进程时，Redis 都会创建进程分叉（请记住，Redis 是单线程的）。一个分叉将您的数据保存到磁盘上的 Redis .rdb 快照文件中，另一个分叉为所有读取和写入操作提供服务。为确保您的快照是 point-in-time 快照，所有数据更新和添加都将写入与数据区域分开的可用内存区域。

只要您在数据保存到磁盘期间拥有足够的可用内存来记录所有写入操作，便不会遇到内存不足的问题。如果出现以下任何情况，您便可能会遇到内存不足的问题：

- 您的应用程序执行很多写入操作，因此需要大量可用内存来接收新数据或更新的数据。
- 可用于写入新数据或更新的数据的内存非常少。
- 您的数据集很大，需要很长时间才能保存到磁盘，因而需要大量写入操作。

下图说明了执行后台写入进程时的内存使用情况。



有关执行备份对性能的影响的信息，请参阅备份自行设计的集群所产生的性能影响。

有关 Redis 如何执行快照的更多信息，请参阅 http://redis.io。

有关区域和可用区的更多信息，请参阅选择区域和可用区。

避免在执行后台写入时耗尽内存

每当调用后台写入进程（例如 BGSAVE 或 BGREWRITEAOF）时，为防止该进程失败，您拥有的可用内存必须多于写入操作在该进程执行期间所消耗的内存量。最糟糕的情况是，在后台写入操作期间，每个 Redis 记录都进行更新，并且有一些新的记录增加到缓存。因此，对于 2.8.22 版之前的 Redis，建议您将 `reserved-memory-percent` 设置为 50 (50%)；对于 2.8.22 版及更高版本的 Redis，建议您将此参数设置为 25 (25%)。

`maxmemory` 值指示可供您用于数据和操作开销的内存。因为您无法修改默认参数组中的 `reserved-memory` 参数，所以必须为集群创建自定义参数组。`reserved-memory` 的默认值是 0，这允许 Redis 为数据消耗所有 maxmemory，可能不会为其他用途（例如后台写入进程）留下内存。有关各种节点实例类型的 `maxmemory` 值，请参阅 特定于 Redis 节点类型的参数。

您还可以使用 `reserved-memory` 参数来减少 Redis 在框架上使用的内存量。

有关中特定于 Redis 的参数的更多信息，请参阅。 ElastiCache Redis 特定的参数

有关创建和修改参数组的信息，请参阅创建参数组和修改参数组。

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片期间测试应用程序行为。
- 获取扩展问题的提前通知 – 重新分片是一项需使用大量计算资源的操作。因此，我们建议在重新分片期间，多核心实例的情况下 CPU 保持 80% 以下的利用率，单核心实例的情况下 CPU 保持 50% 以下的利用率。在应用程序开始监测扩展问题前监控 ElastiCache for Redis 指标并启动重新分片。跟踪的有用指标为 `CPUUtilization`、`NetworkBytesIn`、`NetworkBytesOut`、`CurrConnections`、`NewConnections` 和 `BytesUsedForCacheItems`。
- 横向缩减前请确保有足够的空余内存可用 – 如果要进行横向缩减，请确保要保留的分片上的可用空余内存至少是您计划删除的分片上已用内存的 1.5 倍。
- 在非高峰时间启动重新分片 – 此做法有助于减少重新分片操作期间对客户端的延迟和吞吐量的影响。同样有助于更快完成重新分片，因为有更多资源可用于槽重新分配。
- 审核客户端超时行为 – 部分客户端可能会在联机集群调整大小期间出现更高的延迟。为客户端库配置更高的超时会有所帮助，即便服务器处于更高的负载条件下，系统也有时间进行连接。在某些情况下，您可能会打开与服务器的大量连接。在这些情况下，请考虑增加指数回退以便重新连接逻辑。这样做可防止突增的新连接同时连接服务器。
- 在每个分片上加载函数 - 横向扩展集群时，ElastiCache 会自动将其中一个现有节点（随机选择）中加载的函数复制到新节点。如果您的集群有 Redis 7.0 或更高版本，并且您的应用程序使用 Redis Functions，我们建议您在横向扩展之前将所有函数加载到所有分片，这样您的集群就不会在不同的分片上有不同的函数。

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。目标分片上的数据不会被删除。
- 带有大型项目的槽不会迁移。特别是带有超过 256 MB 后序列化的槽不会迁移。
- 在重新分片操作期间，Lua 脚本中不支持 `FLUSHALL` 和 `FLUSHDB` 命令。在 Redis 6 之前，不支持在要迁移的槽上运行 `BRPOPLPUSH` 命令。

最大程度减少维护期间的停机时间

集群模式配置在托管或非托管操作期间具有最高可用性。我们建议您使用支持集群模式的客户端，此客户端会连接到集群发现终端节点。对于禁用的集群模式，我们建议您将主终端节点用于所有写入操作。

对于读取活动，应用程序还可以连接到集群中的任何节点。与主端点不同，节点端点会解析为特定端点。如果您在您的集群中进行更改（例如添加或删除副本），则必须在您的应用程序中更新节点端点。这就是我们建议您在已禁用集群模式的情况下使用读取器端点进行读取活动的原因。

如果在集群中启用了自动失效转移，则主节点可能发生变化。因此，应用程序应确认节点的角色并更新所有读取终端节点。这样做有助于确保您不会在主节点上引发重大负载。禁用自动失效转移后，节点的角色不会发生变化。但是，与启用自动失效转移的集群相比，托管式或非托管式操作的停机时间更长。

避免将读取请求定向到单个只读副本节点，因为它的不可用性可能会导致读取中断。要么回退以从主节点读取，要么确保至少有两个只读副本，以避免在维护期间出现任何读取中断。

Redis 最佳实践

以下是使用 Redis 提高性能和可靠性时的最佳实践：

- 使用已启用集群模式配置 – 已启用集群模式允许缓存水平扩展，从而实现比已禁用集群模式配置更大的存储和吞吐量。ElastiCache 无服务器仅在已启用集群模式配置中可用。
- 使用长时间生存的连接 – 创建新连接的成本很高，而且需要花费时间和缓存中的 CPU 资源。尽可能重复使用连接（例如，使用连接池），以将此成本分摊到多条命令上。
- 从副本读取 – 如果您使用的是 ElastiCache 无服务器或已配置只读副本（自行设计的集群），请直接读取副本来提高可扩展性和/或减少延迟。从副本中读取的数据最终与主节点是一致的。

  在自行设计的集群中，避免将读取请求定向到单个只读副本，因为在节点出现故障时，可能暂时无法读取。将您的客户端配置为将读取请求定向到至少两个只读副本，或将读取定向到单个副本和主节点。

  在 ElastiCache 无服务器中，从副本端口（6380）进行读取会尽可能将读取定向到客户端的本地可用区，从而减少检索延迟。在故障期间，它将自动回退到其他节点。
- 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。
- 使用分片 pub/sub – 在使用 Redis 支持具有高吞吐量的 pub/sub 工作负载时，建议您使用分片 pub/sub（在 Redis 7 或更高版本中可用）。已启用集群模式的集群中的传统 pub/sub 会向集群中的所有节点广播消息，这可能会导致较高的 EngineCPUUtilization。请注意，在 ElastiCache 无服务器中，传统的 pub/sub 命令会在内部使用分片 pub/sub 命令。

缓存策略

在以下主题中，您可以找到填充和维护缓存的策略。

为填充并维护缓存而执行的策略取决于要缓存的数据以及针对这些数据的访问模式。例如，您可能不想对游戏站点上排名前 10 排行榜和趋势新闻报道执行相同的策略。在本节的剩余内容中，我们将讨论常见缓存维护策略及其优点和缺点。

主题

延迟加载

顾名思义，延迟加载是一种仅在需要时将数据加载到缓存中的缓存策略。它的工作原理如下。

Amazon ElastiCache 是一种内存键值存储，位于您的应用程序和其访问的数据存储（数据库）之间。当应用程序请求数据时，它会先向 ElastiCache 缓存发出请求。如果数据在缓存中且最新，则 ElastiCache 会将数据返回到应用程序。如果数据不在缓存中或已过期，则应用程序会请求数据存储中的数据。然后，数据存储将数据返回到应用程序。之后，应用程序会将从存储接收的数据写入缓存。这样就可以在下次请求时更快地检索这些数据。

当数据位于缓存中且未过期时，就会发生缓存命中：

1. 应用程序请求缓存中的数据。
2. 缓存将数据返回给应用程序。

当数据不在缓存中或已过期时，就会发生缓存未命中：

1. 应用程序请求缓存中的数据。
2. 缓存不具有所请求的数据，因此返回了 `null`。
3. 应用程序请求数据库中的数据并收到数据。
4. 应用程序使用新数据更新缓存。

延迟加载的优点和缺点

延迟加载的优点如下：

- 仅对请求的数据进行缓存。

  由于大部分数据从未被请求过，因此延迟加载避免了向缓存中填入未请求的数据。
- 节点故障对应用程序来说并不致命。

  当某个节点发生故障并由新的空节点替换时，应用程序会继续运行，但延迟会增加。向新节点发出请求时，每次缓存未命中都会导致在数据库中进行查询。同时会将数据副本添加到缓存中，以便后续请求从缓存中进行检索。

延迟加载的缺点如下：

- 缓存未命中会导致性能损失。每次缓存未命中都会导致 3 次往返：
  1. 初次从缓存中请求数据
  2. 查询数据库中的数据
  3. 将数据写入缓存

  这些未命中会导致在数据到达应用程序时出现显著延迟。
- 过时数据。

  如果仅在缓存未命中时将数据写入缓存，则缓存中的数据会过时。出现此结果的原因在于，在数据库中更改数据时未更新缓存。要解决此问题，您可以使用 直写 和 添加 TTL 策略。

延迟加载伪代码示例

以下代码是延迟加载逻辑的伪代码示例。

```
// ****************************************
// function that returns a customer's record.
// Attempts to retrieve the record from the cache.
// If it is retrieved, the record is returned to the application.
// If the record is not retrieved from the cache, it is
//    retrieved from the database,
//    added to the cache, and
//    returned to the application
// ****************************************
```

```
get_customer(customer_id)

    customer_record = cache.get(customer_id)
    if (customer_record == null)

        customer_record = db.query("SELECT * FROM Customers WHERE id = {0}",
 customer_id)
        cache.set(customer_id, customer_record)

    return customer_record
```

对于此示例，获取数据的应用程序代码如下。

```
customer_record = get_customer(12345)
```

直写

直写策略会在将数据写入数据库时在缓存中添加或更新数据。

直写的优点和缺点

直写的优点如下：

- 缓存中的数据永不过时。

  由于每次将缓存中的数据写入数据库时都会更新这些数据，因此缓存中的数据始终为最新数据。
- 直写性能损失与读取性能损失比较。

  每次写入都涉及两次往返：

  1. 对缓存进行写入
  2. 对数据库进行写入

  这将增加流程的延迟。即便如此，与检索数据时的延迟相比，最终用户通常更能容忍更新数据时的延迟。有一个内在的意义，即更新的工作量更大，因而花费的时间会更长。

直写的缺点如下：

- 缺失的数据。

  如果启动新节点（无论是由于节点故障还是横向扩展），都会出现数据缺失。此数据将持续保持丢失状态直到将其添加或更新到数据库。您可以通过实现延迟加载和使用直写来最大限度减少此情况。

- 缓存扰动。

  大多数数据从不会被读取，这是一种资源浪费。通过添加存活时间 (TTL) 值，可以最大程度地减少空间浪费。

直写伪代码示例

以下代码是直写逻辑的伪代码示例。

```
// ****************************************
// function that saves a customer's record.
// ****************************************
save_customer(customer_id, values)

    customer_record = db.query("UPDATE Customers WHERE id = {0}", customer_id, values)
    cache.set(customer_id, customer_record)
    return success
```

对于此示例，获取数据的应用程序代码如下。

```
save_customer(12345,{"address":"123 Main"})
```

添加 TTL

延迟加载允许过时数据，但不会失败并产生空节点。直写可确保数据始终为最新数据，但直写可能会失败并产生空节点，还可能向缓存填充过多数据。您可对每次写入添加存活时间 (TTL) 值，充分利用每种策略的优势。同时，您可以并在很大程度上避免多余数据混淆缓存。

存活时间 (TTL) 是一个整数值，此值指定密钥过期之前的秒数。Redis 可以指定此值的秒数或毫秒数。当应用程序尝试读取过期密钥时，其处理方式是当做未找到该密钥。应用程序会在数据库中查询该密钥并更新缓存。这种方法不能保证值不会过时。不过，其可以防止数据过时太久，并要求不时从数据库中刷新缓存中的值。

有关更多信息，请参阅 Redis set 命令 。

TTTL 伪代码示例

以下代码为具有 TTL 的直写逻辑伪代码示例。

```
// ****************************************
```

```
// function that saves a customer's record.
// The TTL value of 300 means that the record expires
//    300 seconds (5 minutes) after the set command
//    and future reads will have to query the database.
// ***************************************
save_customer(customer_id, values)

    customer_record = db.query("UPDATE Customers WHERE id = {0}", customer_id, values)
    cache.set(customer_id, customer_record, 300)

    return success
```

以下代码为具有 TTL 的延迟加载逻辑伪代码示例。

```
// ***************************************
// function that returns a customer's record.
// Attempts to retrieve the record from the cache.
// If it is retrieved, the record is returned to the application.
// If the record is not retrieved from the cache, it is
//    retrieved from the database,
//    added to the cache, and
//    returned to the application.
// The TTL value of 300 means that the record expires
//    300 seconds (5 minutes) after the set command
//    and subsequent reads will have to query the database.
// ***************************************
get_customer(customer_id)

    customer_record = cache.get(customer_id)

    if (customer_record != null)
        if (customer_record.TTL < 300)
            return customer_record        // return the record and exit function

    // do this only if the record did not exist in the cache OR
    //    the TTL was >= 300, i.e., the record in the cache had expired.
    customer_record = db.query("SELECT * FROM Customers WHERE id = {0}", customer_id)
    cache.set(customer_id, customer_record, 300)  // update the cache
    return customer_record                // return the newly retrieved record and exit
 function
```

对于此示例，获取数据的应用程序代码如下。

```
save_customer(12345,{"address":"123 Main"})
```

```
customer_record = get_customer(12345)
```

相关 主题

- 内存中的数据存储
- 选择引擎和版本
- 针对 Redis ElastiCache 进行扩展

管理自行设计的集群

此部分中的主题有助于您管理自行设计的集群。

> **Note**
>
> 这些主题不适用于 ElastiCache 无服务器。

主题

- ElastiCache 适用于 Redis 集群的 Auto Scaling
- 修改集群模式
- 使用全球数据存储跨 AWS 区域复制
- 使用复制组时的高可用性
- 管理维护
- 使用参数组配置引擎参数

ElastiCache 适用于 Redis 集群的 Auto Scaling

先决条件

ElastiCache 适用于 Redis 的 Auto Scaling 仅限于以下内容：

- 运行 Redis 6.0 及以上引擎版本的 Redis（已启用集群模式）集群

- 运行 Redis 引擎版本 7.0.7 及更高版本的数据分层（已启用集群模式）集群
- 实例大小 – 大型、XLarge、2XLarge
- 实例类型系列 - R7g、R6g、R6gd、R5、M7g、M6g、M5、C7gn
- 在全球数据存储、 ElastiCache Outposts 或本地区域中运行的集群不支持 Redis 的 Auto Scaling。

使用 Redis Auto S ElastiCache caling 自动管理容量

ElastiCache 对于 Redis 来说，自动缩放是指能够自动增加或减少 for Redis 服务中所需的分片或副本。 ElastiCache ElastiCache for Redis 利用 Application Auto Scaling 服务来提供此功能。有关更多信息，请参阅 Application Auto Scaling。要使用自动扩展，您需要定义并应用使用您分配的 CloudWatch 指标和目标值的扩展策略。 ElastiCache for Redis auto Scaling 使用该策略来增加或减少实例数量以响应实际工作负载。

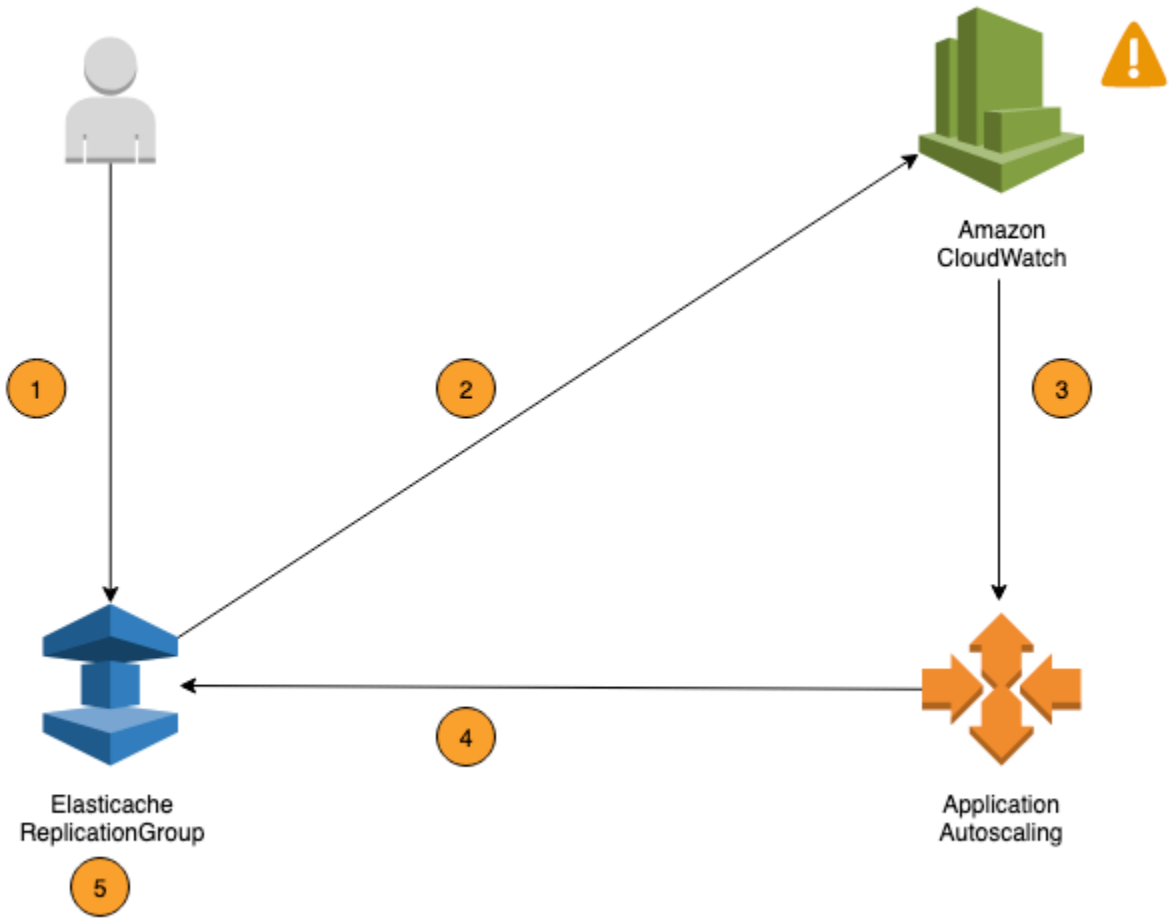
您可以使用 AWS Management Console 来应用基于预定义指标的扩展策略。枚举中对 `predefined metric` 进行了定义，因此您可在代码中按名称指定或在 AWS Management Console中使用它。自定义指标不可用于使用 AWS Management Console的选择。或者，您可以使用 AWS CLI 或 Application Auto Scaling API 来应用基于预定义或自定义指标的扩展策略。

ElastiCache for Redis 支持缩放以下维度：

- 分片 – 自动添加/删除集群中的分片，类似于手动在线重新分片。在这种情况下， ElastiCache 对于 Redis，自动缩放会代表您触发缩放。
- 副本 –自动向集群添加/删除其中的副本，类似于手动增加/减少副本操作。 ElastiCache for Redis auto scaling 可在集群中的所有分片上统一添加/删除副本。

ElastiCache for Redis 支持以下类型的自动扩展策略：

- 目标跟踪扩缩策略 – 根据特定指标的目标值，增加或减少服务所运行的分片/副本数量。这与恒温器保持家里温度的方式类似。您选择一个温度，恒温器将完成所有其他工作。
- App@@ lication ElastiCache for Redis 的计划扩展 auto scaling — 根据日期和时间增加或减少服务运行的分片/副本数量。



以下步骤总结了 Redis ElastiCache 的自动缩放流程，如上图所示：

1. 您可以 ElastiCache 为适用于 Redis 的复制组创建适用于 Redis ElastiCache 的自动扩展策略。
2. ElastiCache for Redis auto scaling 会代表你创建一对 CloudWatch 警报。每对告警代表指标的上限和下限。当集群的实际利用率持续偏离您的目标利用率时，就会触发这些 CloudWatch 警报。您现在可以在控制台中查看告警。
3. 如果配置的指标值在特定时间段内超过您的目标利用率（或低于目标），则 CloudWatch 会触发警报，调用 Redis ElastiCache auto Scaling 来评估您的扩展策略。
4. ElastiCache for Redis auto scaling 会发出修改请求以调整您的集群容量。
5. ElastiCache for Redis 会处理修改请求，动态增加（或减少）集群分片/副本容量，使其接近您的目标利用率。

要了解 Redi ElastiCache s Auto Scaling 的工作原理，假设你有一个名为`UsersCluster`的集群。通过监控的 CloudWatch 指标`UsersCluster`，您可以确定流量达到峰值时集群所需的最大分片数以及流量处于最低点时所需的最小分片。此外，还针对 `UsersCluster` 集群确定 CPU 利用率目标值。ElastiCache for Redis auto scaling 使用其目标跟踪算法来确保根据需要调整的`UsersCluster`预配置分片，以便利用率保持在或接近目标值。

> **Note**
>
> 扩展可能需要很长时间，并且需要额外的群集资源才能使分片重新平衡。 ElastiCache for Redis Auto Scaling 只有在实际工作负载持续几分钟内保持升高（或低迷）状态时才会修改资源设置。f ElastiCache or Redis auto scaling 目标跟踪算法旨在长期将目标利用率保持在或接近您选择的值。

弹性伸缩策略

扩展策略包含以下组件：

- 目标指标 – ElastiCache for Redis 弹性伸缩用于确定何时扩展以及扩展程度的 CloudWatch 指标。
- 最小和最大容量 – 可扩展的最小和最大分区或副本数。

> **Important**
>
> 创建弹性伸缩策略时，如果当前容量高于配置的最大容量，我们会在策略创建过程中横向缩减为最大容量。同样，如果当前容量低于配置的最小容量，我们将横向扩展到最小容量。

- 冷却时间 – 在完成一个横向缩减或横向扩展活动后开始另一个横向扩展活动之前等待的时间（秒）。
- 服务相关角色 – 与特定 AWS 服务关联的 AWS Identity and Access Management (IAM) 角色。服务相关角色包含服务代表您调用其他 AWS 服务所需的一切权限。ElastiCache for Redis 弹性伸缩会自动为您生成此角色 `AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG`。
- 启用或禁用横向缩减活动 – 能够为策略启用或禁用横向缩减活动。

主题

- 启用或禁用缩减活动

弹性伸缩目标指标

在这种类型的策略中，预定义或自定义指标以及指标目标值是在目标跟踪扩展策略配置中指定的。ElastiCache for Redis 弹性伸缩创建和管理触发扩缩策略的 CloudWatch 告警，并根据指标和目标值计算扩展调整。扩展策略根据需要添加或删除分区/副本，以便将指标保持在指定的目标值或该值附近。除了将指标保持在目标值附近以外，目标跟踪扩展策略还会根据由于工作负载变化而造成的指标波动进行调整。这种策略还会最大限度减少集群的可用分区/副本数的快速波动。

例如，考虑使用具有预定义的平均 `ElastiCachePrimaryEngineCPUUtilization` 指标的扩展策略。这种策略可以将 CPU 使用率保持在指定的使用率百分比（如 70%）或该值附近。

> **Note**
>
> 对于每个集群，您只能针对每个目标指标创建一个弹性伸缩策略。

最小和最大容量

分片

您可以指定 ElastiCache for Redis 弹性伸缩可扩展的最大分区数量。此值必须小于或等于 250 且最小为 1。您还可以指定由 ElastiCache for Redis 弹性伸缩管理的最小分区数。此值必须至少为 1，且等于或小于为最大分区数 (250) 指定的值。

副本

您还可以指定由 ElastiCache for Redis 弹性伸缩管理的最大副本数。此值必须小于或等于 5。您还可以指定由 ElastiCache for Redis 弹性伸缩管理的最小副本数。此值必须至少为 1，且等于或小于为最大副本数 (5) 指定的值。

要确定典型流量所需的最小和最大分区/副本数，请使用模型的预期通信速率测试弹性伸缩配置。

> **Note**
>
> ElastiCache for Redis 弹性伸缩策略会增加集群容量，直到其达到您定义的最大大小或直到服务限额适用为止。若要请求提高限制，请参阅 AWS Service Limits 并选择限制类型 Nodes per cluster per instance type（每个实例类型的每个集群的节点数）。

> **Important**
>
> 在无流量时横向缩减。如果变体的流量变为零，ElastiCache for Redis 会弹性横向缩减到指定的最小实例数。

冷却时间

您可以添加影响集群扩展的冷却时间，以优化目标跟踪扩缩策略的响应速度。冷却时间阻止后续扩展或缩减请求，直至冷却时间到期。这会减慢针对横向缩减请求在 ElastiCache for Redis 集群中删除分区/副本的速度，及针对横向扩展请求创建分区/副本的速度。您可以指定以下冷却时间：

- 横向缩减活动减少 ElastiCache for Redis 集群中的分区/副本数。缩减冷却时间指定在完成一个缩减活动后开始另一个缩减活动之前等待的时间 (秒)。
- 横向扩展活动增加 ElastiCache for Redis 集群中的分区/副本数。扩展冷却时间指定在完成一个扩展活动后开始另一个扩展活动之前等待的时间 (秒)。

如果未指定横向缩减或横向扩展冷却时间，则默认横向扩展冷却时间为 600 秒，默认横向缩减冷却时间为 900 秒。

启用或禁用缩减活动

您可以为策略启用或禁用缩减活动。启用横向缩减活动允许扩展策略删除分区/副本。在启用缩减活动时，扩展策略中的缩减冷却时间将应用于缩减活动。禁用横向缩减活动将禁止扩展策略删除分区/副本。

> **Note**
>
> 横向扩展活动始终处于启用状态，以便扩展策略可以根据需要创建 ElastiCache for Redis 分区/副本。

Redis Auto Scal ElastiCache ing 所需的 IAM 权限

ElastiCache for Redis 的 Auto Scaling 是通过 for Redis、 CloudWatch和 Appl ElastiCache ication Auto Scaling API 的组合实现的。使用适用 ElastiCache 于 Redis 创建和更新集群，使用创建警报 CloudWatch，使用Application Auto Scaling创建扩展策略。除了用于创建和更新集群的标准 IAM 权限外，访问 ElastiCache Redis Auto Scaling 设置的 IAM 用户还必须拥有支持动态扩展的服务的相应权限。IAM 用户必须具有使用以下示例策略中的操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "application-autoscaling:*",
                "elasticache:DescribeReplicationGroups",
                "elasticache:ModifyReplicationGroupShardConfiguration",
                "elasticache:IncreaseReplicaCount",
                "elasticache:DecreaseReplicaCount",
                "elasticache:DescribeCacheClusters",
                "elasticache:DescribeCacheParameters",
                "cloudwatch:DeleteAlarms",
                "cloudwatch:DescribeAlarmHistory",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:DescribeAlarmsForMetric",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics",
                "cloudwatch:PutMetricAlarm",
                "cloudwatch:DisableAlarmActions",
                "cloudwatch:EnableAlarmActions",
                "iam:CreateServiceLinkedRole",
                "sns:CreateTopic",
                "sns:Subscribe",
                "sns:Get*",
                "sns:List*"
            ],
            "Resource": "arn:aws:iam::123456789012:role/autoscaling-roles-for-cluster"
        }
    ]
}
```

服务相关角色

fo ElastiCache r Redis auto Scaling 服务还需要描述您的集群和 CloudWatch 警报的权限，以及代表您修改您 ElastiCache 的 Redis 目标容量的权限。如果您为适用于 Redis 的集群启用 Auto Scaling，它会创建一个名为的服务相关角色。 ElastiCache `AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG`此服务相关角色授予 ElastiCache Redis auto Scaling 描述策略警报、监控队列当前容量和修改队列容量的权限。服务相关角色是 Redis auto Scal ElastiCache ing 的默认角色。有关更多信息，请参阅《Application Auto Scaling 用户指南》中的 Redis 自动缩放的服务相关角色。 ElastiCache

Auto Scaling 最佳实践

在注册 Auto Scaling 功能之前，我们建议执行以下操作：

1. 仅使用一个跟踪指标 – 确定您的集群是具有 CPU 密集型工作负载还是数据密集型工作负载，并使用相应的预定义指标来定义扩展策略。
   - 引擎 CPU：`ElastiCachePrimaryEngineCPUUtilization`（分片维度）或 `ElastiCacheReplicaEngineCPUUtilization`（副本维度）
   - 数据库使用情况：`ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage` 此扩展策略在集群上将 maxmemory-policy 设置为 noeviction 时效果最佳。

   我们建议您避免在集群上使用每个维度使用多个策略。 ElastiCache for Redis 如果任何目标跟踪策略已准备好进行横向扩展，Auto Scaling 将扩展可扩展的目标，但只有在所有目标跟踪策略（启用了缩小部分）都准备好进行扩展时，才会缩小规模。如果多个策略指示可扩展目标同时横向扩展或缩减，ElastiCache for Redis 会根据为横向缩减和横向扩展提供最大容量的策略进行扩展。
2. 目标跟踪的自定义指标 – 在对目标跟踪使用自定义指标时要谨慎，因为自动扩缩最适合与为策略所选的指标变化成比例的横向扩展/缩减。如果这些指标未随用于策略创建的扩缩操作成比例变化，则可能会导致持续的横向扩展或横向缩减操作，从而可能会影响可用性或成本。

   对于数据分层集群（r6gd 系列实例类型），请避免使用基于内存的指标进行扩缩。
3. 计划扩缩 – 如果您确定工作负载是确定性的（在特定时间达到高/低），我们建议您使用计划扩缩并根据需要配置目标容量。目标跟踪最适合以通过在您需要更多资源时横向扩展，并在需要较少资源时横向缩减的方式按所需目标指标运行的非确定性工作负载和集群。
4. 禁用横向缩减 – 基于目标跟踪的 Auto Scaling 最适合工作负载逐渐增加/减少的集群，因为指标的峰值/下降可触发连续横向扩展/缩减振荡。为了避免这种振荡，您可以先禁用横向缩减，之后可以随时根据需要手动横向缩减。
5. 测试应用程序 – 我们建议您使用估计的最小/最大工作负载测试应用程序，以确定集群所需的绝对最小、最大分片/副本，同时创建扩缩策略以避免出现可用性问题。Auto Scaling 可以横向扩展至为目标配置的最大阈值，也可以横向缩减至配置的最小阈值。
6. 定义目标值-您可以分析四周内集群利用率的相应 CloudWatch 指标，以确定目标值阈值。如果您仍然不确定要选择哪个值，我们建议您从支持的最小预定义指标值开始。
7. AutoScaling on Target Tracking 最适合在分片/副本维度上均匀分配工作负载的集群。分布不均可能导致：
   - 因几个热分片/副本上工作负载峰值/下降而出现在不需要扩展时进行扩展。
   - 因整体平均值接近目标（即使具有热分片/副本）而在需要扩展时不执行扩展。

> **Note**
>
> 扩展集群时，ElastiCache 会自动将其中一个现有节点（随机选择）中加载的函数复制到新节点。如果您的集群有 Redis 7.0 或更高版本，并且您的应用程序使用 Redis Functions，我们建议您在横向扩展之前将所有函数加载到所有分片，这样您的集群就不会在不同的分片上有不同的函数。

注册后 AutoScaling，请注意以下几点：

- Auto Scaling 支持的配置存在限制，因此我们建议不要更改已注册 Auto Scaling 的复制组的配置。示例如下：
  - 手动将实例类型修改为不支持的类型。
  - 将复制组与全局数据存储关联。
  - 更改 `ReservedMemoryPercent` 参数。
  - 手动增加/减少分片/副本，使其数目超出策略创建过程中配置的最小/最大容量。

对分区使用弹性伸缩

下面提供了有关目标跟踪和计划策略以及如何使用 AWS Management Console AWS CLI 和 API 应用它们的详细信息。

目标跟踪扩缩策略

在使用目标跟踪扩展策略时，您可以选择一个指标并设置一个目标值。ElastiCache for Redis 弹性伸缩创建和管理触发扩缩策略的 CloudWatch 告警，并根据指标和目标值计算扩展调整。扩展策略根据需要添加或删除分区，以便将指标保持在指定的目标值或接近该值。除了将指标保持在目标值附近以外，目标跟踪扩展策略还会根据由于负载模式波动而造成的指标波动进行调节，并最大限度减少队列容量发生快速波动的情况。

例如，考虑使用具有已配置了目标值的预定义平均 `ElastiCachePrimaryEngineCPUUtilization` 指标的扩展策略。这种策略可以将 CPU 使用率保持在指定的目标值或接近该值。

预定义指标

预定义指标是一种结构，用于指示给定 CloudWatch 指标的特定名称、维度和统计数据（`average`）。自动扩缩策略为您的集群定义下面的预定义指标之一：

预定义指标名称	CloudWatch 指标名称	CloudWatch 指标维度	不符合条件的实例类型
ElastiCachePrimaryEngineCPUUtilization	EngineCPUUtilization	ReplicationGroupId，角色 = 主要	无
ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage	DatabaseCapacityUsageCountedForEvictPercentage	Redis 复制组指标	无
ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage	DatabaseMemoryUsageCountedForEvictPercentage	Redis 复制组指标	R6gd

数据分层实例类型不能使用 ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage，因为这些实例类型将数据同时存储在内存和 SSD 中。数据分层实例的预期使用案例是 100% 的内存使用率，并根据需要填满 SSD。

分区的弹性伸缩条件

当服务检测到您的预定义指标等于或大于目标设置时，它会自动提高分片容量。ElastiCache for Redis 按等于两个数字之间较大者的计数来横向扩展集群分区：与目标之间差异的百分比和当前分区数的 20%。对于横向缩减，ElastiCache for Redis 不会弹性横向缩减，除非整体指标值低于所定义的目标的 75%。

关于横向扩展示例，如果您的分区数为 50，以及

- 在目标超出了 30% 的情况下，ElastiCache for Redis 将横向扩展 30%，最后的结果是每个集群 65 个分区。
- 在目标超出了 10% 的情况下，ElastiCache for Redis 默认横向扩展最低 20%，最后的结果是每个集群 60 个分区。

对横向缩减示例，如果您选择的目标值是 60%，则 ElastiCache for Redis 会直到该指标小于或等于 45%（比目标 60% 低 25%）时才会弹性横向缩减。

弹性伸缩注意事项

请注意以下事项：

- 目标跟踪扩展策略假设它应该在指定指标高于目标值时执行向外扩展。因此，不能使用目标跟踪扩展策略在指定指标低于目标值时向外扩展。ElastiCache for Redis 在与集群现有分区目标数之间存在最小 20% 的偏差时横向扩展分区。
- 当指定指标数据不足时，目标跟踪扩展策略不会执行扩展。它不会执行横向缩减，因为它不会将数据不足解读为使用率低。
- 您可能会看到目标值与实际指标数据点之间存在差距。这是因为 ElastiCache for Redis 弹性伸缩在确定要添加或删除多少容量时将始终通过向上或向下舍入保守地进行操作。以免添加的容量不足或删除的容量过多。
- 为了确保应用程序可用性，服务会针对指标尽快按比例横向扩展，但横向缩减过程相对缓慢。
- 您可以为 ElastiCache for Redis 集群提供多个目标跟踪扩缩策略，前提是它们各自使用不同的指标。ElastiCache for Redis 弹性伸缩的目的是始终优先考虑可用性，因此其行为会有所不同，具体取决于目标跟踪策略是否已准备好横向扩展或横向缩减。如果任何目标跟踪策略已准备好进行扩展，它将扩展服务，但仅在所有目标跟踪策略（启用了缩减部分）准备好缩减时才执行缩减。
- 请勿编辑或删除 ElastiCache for Redis 弹性伸缩为目标跟踪扩缩策略管理的 CloudWatch 告警。在您删除扩缩策略时，ElastiCache for Redis 弹性伸缩会自动删除相应的告警。
- ElastiCache for Redis 弹性伸缩不会阻止您手动修改集群分区。这些手动调整不会影响附加到扩展策略的任何现有 CloudWatch 告警，但可能会影响可触发这些 CloudWatch 告警的指标。
- 这些由弹性伸缩管理的 CloudWatch 告警是通过集群中所有分区的 AVG 指标来定义的。因此，拥有热分区可能会导致以下任一情况：
  - 因若干热分区上负载触发 CloudWatch 告警而导致在不需要扩展时执行扩展
  - 因所有分区的影响告警的聚合 AVG 指标不违例而在需要扩展时不执行扩展。

- ElastiCache for Redis 对每个集群的节点的默认限制仍然适用。因此，当选择弹性伸缩时，如果您希望最大节点数超过默认限制，请在 AWS Service Limits 请求提高限制，并选择限制类型 Nodes per cluster per instance type（每个实例类型的集群的节点数）。
- 请确保您的 VPC 中有足够的 ENI（弹性网络接口）可用，横向扩展过程中需要弹性网络接口。有关更多信息，请参阅弹性网络接口。
- 如果 EC2 没有足够的可用容量，则 ElastiCache for Redis 弹性伸缩将不会扩展，并延迟直到容量可用。
- 在横向缩减期间，ElastiCache for Redis 弹性伸缩不会删除具有项目大小大于 256MB 后序列化的槽的分区。
- 在横向缩减期间，如果生成的分区配置中可用的内存不足，则横向缩减不会删除分区。

添加扩展策略

您可以使用 AWS Management Console 添加扩展策略。

向 ElastiCache for Redis 集群添加弹性伸缩策略

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在导航窗格中，选择 Redis。
3. 选择您要向其中添加策略的集群（选择集群名称，而不是其左侧的按钮）。
4. 选择 Auto Scaling policies（弹性伸缩策略）选项卡。
5. 选择 add dynamic scaling（添加动态扩展）。
6. 对于 Policy name（策略名称），输入一个策略名称。
7. 对于 Scalable Dimension（可扩展维度），选择 shards（分区）。
8. 对于目标指标，请选择以下选项之一：
   - Primary CPU Utilization（主 CPU 使用率），用于根据平均 CPU 使用率创建策略。
   - Memory（内存），用于根据平均数据库内存创建策略。
   - 容量，用于根据平均数据库使用量创建策略。“容量”指标包括用于数据分层实例的内存和 SSD 利用率，以及用于所有其他实例类型的内存利用率。
9. 对于目标值，请选择大于或等于 35 且小于或等于 70 的值。自动扩缩将在各 ElastiCache 分片上为选定的目标指标保持该值：
   - 主 CPU 利用率：在主节点上保持 `EngineCPUUtilization` 指标的目标值。

- 内存：保持 DatabaseMemoryUsageCountedForEvictPercentage 指标的目标值。
- 容量保持 DatabaseCapacityUsageCountedForEvictPercentage 指标的目标值。

将添加或删除集群分区以使指标接近于指定的值。

10. （可选）控制台不支持横向缩减或横向扩展冷却时间。请使用 AWS CLI 修改冷却时间值。
11. 对于 Minimum capacity（最小容量），请键入 ElastiCache for Redis 弹性伸缩策略需要保持的最小分区数。
12. 对于 Maximum capacity（最大容量），请键入 ElastiCache for Redis 弹性伸缩策略需要保持的最大分区数。此值必须小于或等于 250。
13. 选择 Create（创建）。

注册可扩展目标

您需要先对 ElastiCache for Redis 集群注册 ElastiCache for Redis 弹性伸缩，才能对集群使用弹性伸缩。这样做是为了定义应用于该集群的扩展维度和限制。ElastiCache for Redis 弹性伸缩会连同 elasticache:replication-group:NodeGroups 可扩展维度一起动态扩展 ElastiCache for Redis 集群，该可扩展维度表示集群分区的数量。

使用 AWS CLI

要注册 ElastiCache for Redis 集群，请使用包含下列参数的 register-scalable-target 命令：

- --service-namespace – 将该值设置为 elasticache
- --resource-id – ElastiCache for Redis 集群的资源标识符。对于此参数，资源类型为 ReplicationGroup，唯一标识符为 ElastiCache for Redis 集群的名称，例如 replication-group/myscalablecluster。
- --scalable-dimension – 将该值设置为 elasticache:replication-group:NodeGroups。
- --max-capacity – 由 ElastiCache for Redis 弹性伸缩管理的最大分区数。有关 --min-capacity、--max-capacity 和集群中分区数之间关系的信息，请参阅 最小和最大容量。
- --min-capacity – 由 ElastiCache for Redis 弹性伸缩管理的最小分区数。有关 --min-capacity、--max-capacity 和集群中分区数之间关系的信息，请参阅 最小和最大容量。

Example

在以下示例中，您注册一个名为 `myscalablecluster` 的 ElastiCache for Redis 集群。该注册表示应将集群动态扩展为具有 1 到 10 个分区。

对于 Linux、macOS 或 Unix：

```
aws application-autoscaling register-scalable-target \
    --service-namespace elasticache \
    --resource-id replication-group/myscalablecluster \
    --scalable-dimension elasticache:replication-group:NodeGroups \
    --min-capacity 1 \
    --max-capacity 10 \
```

对于 Windows：

```
aws application-autoscaling register-scalable-target ^
    --service-namespace elasticache ^
    --resource-id replication-group/myscalablecluster ^
    --scalable-dimension elasticache:replication-group:NodeGroups ^
    --min-capacity 1 ^
    --max-capacity 10 ^
```

使用 API

要注册您的 ElastiCache 集群，请使用包含下列参数的 register-scalable-target 命令：

- ServiceNamespace – 将此值设置为 elasticache。
- ResourceID – ElastiCache 集群的资源标识符。对于此参数，资源类型为复制组，唯一标识符为 ElastiCache for Redis 集群的名称，例如 `replication-group/myscalablecluster`。
- ScalableDimension – 将此值设置为 `elasticache:replication-group:NodeGroups`。
- MinCapacity – 由 ElastiCache for Redis 弹性伸缩管理的最小分区数。有关 --min-capacity、--max-capacity 和集群中副本数之间关系的信息，请参阅 最小和最大容量。
- MaxCapacity – 由 ElastiCache for Redis 弹性伸缩管理的最大分区数。有关 --min-capacity、--max-capacity 和集群中副本数之间关系的信息，请参阅 最小和最大容量。

Example

在以下示例中，您使用 Application Auto Scaling API 注册一个名为 `myscalablecluster` 的 ElastiCache for Redis 集群。该注册表示应将集群动态扩展为具有 1 到 5 个副本。

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:NodeGroups",
    "MinCapacity": 1,
    "MaxCapacity": 5
}
```

定义扩缩策略

目标跟踪扩展策略配置是由 JSON 块表示的，其中定义了指标和目标值。您可以在文本文件中将扩展策略配置保存为 JSON 块。您可以在调用 AWS CLI 或 Application Auto Scaling API 时使用该文本文件。有关策略配置语法的更多信息，请参阅《Application Auto Scaling API 参考》TargetTrackingScalingPolicyConfiguration中的。

您可以使用以下选项定义目标跟踪扩缩策略配置：

主题

- 使用预定义的指标
- 使用自定义指标
- 使用冷却时间
- 禁用横向缩减活动
- 应用扩缩策略

使用预定义的指标

通过使用预定义的指标，您可以为适用于 Redis 的集群快速定义目标跟踪扩展策略，该策略与 Redis Auto Scaling 中的 ElastiCache 目标跟踪配合使用。 ElastiCache

目前， ElastiCache 适用于 Redis 的 Redis NodeGroup Auto Scalin ElastiCache g 支持以下预定义指标：

- ElastiCachePrimaryEngineCPU利用率 — 适用于 Redis 集群中 CloudWatch所有主节点的EngineCPUUtilization指标 ElastiCache 的平均值。
- ElastiCacheDatabaseMemoryUsageCountedForEvictPercentage— for Redis 集群中 CloudWatch所有主节点的DatabaseMemoryUsageCountedForEvictPercentage指标 ElastiCache 的平均值。
- ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage— for Redis 集群中 CloudWatch所有主节点的ElastiCacheDatabaseCapacityUsageCountedForEvictPercentage指标 ElastiCache 的平均值。

有关 EngineCPUUtilization、DatabaseMemoryUsageCountedForEvictPercentage 和 DatabaseCapacityUsageCountedForEvictPercentage 指标的更多信息，请参阅使用 CloudWatch 指标监控使用情况。要在扩展策略中使用预定义的指标，您需要为扩展策略创建一个目标跟踪配置。此配置必须包括PredefinedMetricSpecification用于预定义指标的，以及 TargetValue 用于该指标的目标值的。

Example

以下示例介绍了 for Redis 集群的目标跟踪扩展 ElastiCache 的典型策略配置。在此配置中，ElastiCachePrimaryEngineCPUUtilization预定义的指标用于根据集群中所有主节点的平均 CPU 利用率为 40% 来调整 Redis 集群的。 ElastiCache

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
    }
}
```

使用自定义指标

通过使用自定义指标，您可以定义满足您的自定义要求的目标跟踪扩展策略。您可以根据任何与扩展成比例变化的 ElastiCache 指标来定义自定义指标。并非所有 ElastiCache 指标都适用于目标跟踪。指标必须是有效的使用率指标，它用于描述实例的繁忙程度。指标值必须随集群中分区数按比例增加或减少。要使用指标数据按比例横向扩展或缩减分区数，必须按比例进行这种增加或减少。

Example

以下示例说明了扩缩策略的目标跟踪配置。在此配置中，自定义指标会根据名 ElastiCache 为的集群中所有分片的平均 CPU 利用率为 50% 来调整 Redis 集群。`my-db-cluster`

```
{
    "TargetValue": 50,
    "CustomizedMetricSpecification":
    {
        "MetricName": "EngineCPUUtilization",
        "Namespace": "AWS/ElastiCache",
        "Dimensions": [
            {
                "Name": "RelicationGroup","Value": "my-db-cluster"
            },
            {
                "Name": "Role","Value": "PRIMARY"
            }
        ],
        "Statistic": "Average",
        "Unit": "Percent"
    }
}
```

使用冷却时间

您可以为 `ScaleOutCooldown` 指定一个值（秒）以添加横向扩展集群的冷却时间。同样，您可以为 `ScaleInCooldown` 添加一个值（秒）以添加横向缩减集群的冷却时间。有关更多信息，请参阅《Applicati TargetTrackingScalingPolicyConfigurationon Auto Scaling API 参考》中的。

以下示例说明了扩缩策略的目标跟踪配置。在此配置中，`ElastiCachePrimaryEngineCPUUtilization`预定义的指标用于根据该集群中所有主节点的平均 CPU 利用率为 40% 来调整该集群。 ElastiCache 该配置将缩减冷却时间指定为 10 分钟，并将扩展冷却时间指定为 5 分钟。

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
    },
    "ScaleInCooldown": 600,
```

```
    "ScaleOutCooldown": 300
}
```

禁用横向缩减活动

您可以通过禁用缩容活动来阻止目标跟踪扩展策略配置在 for Redis 集群中扩展。 ElastiCache 禁用横向缩减活动将禁止扩展策略删除分区，同时仍允许扩展策略根据需要创建分区。

您可以为 `DisableScaleIn` 指定一个布尔值，以便为集群启用或禁用横向缩减活动。有关更多信息，请参阅《Applicati TargetTrackingScalingPolicyConfigurationon Auto Scaling API 参考》中的。

以下示例说明了扩缩策略的目标跟踪配置。在此配置
中，`ElastiCachePrimaryEngineCPUUtilization`预定义的指标会根据该集群中所有主节点的平均 CPU 利用率为 40% 来调整 Redis 集群。 ElastiCache 该配置禁用扩展策略的缩减活动。

```
{
    "TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {
        "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
    },
    "DisableScaleIn": true
}
```

应用扩缩策略

在 Redis a ElastiCache uto Scaling 中注册集群并定义扩展策略后，您可以将扩展策略应用于已注册的集群。要将扩展策略应用于 Redis ElastiCache 的集群，您可以使用 AWS CLI 或 Application Auto Scaling API。

使用应用扩展策略 AWS CLI

要将扩展策略应用于您的 f ElastiCache or Redis 集群，请使用带有以下参数的put-scaling-policy命令：

- --policy-name – 扩展策略的名称。
- --policy-type – 将此值设置为 `TargetTrackingScaling`。
- --resource-id — Redis 的资源标识符。 ElastiCache 例如`replication-group/myscalablecluster`，对于此参数，资源类型为`ReplicationGroup`，唯一标识符是 Redi ElastiCache s 集群的名称。
- --service-namespace – 将此值设置为 `elasticache`。

- --scalable-dimension – 将此值设置为 `elasticache:replication-group:NodeGroups`。
- --target-tracking-scaling-policy-配置 — 用于 Redis 集群的目标跟踪扩展策略配置 ElastiCache 。

在以下示例中，您将名 ElastiCache 为的目标跟踪扩展策略应用于名为 `myscalablepolicy` for Redis 自动缩放的 R ElastiCache ed `myscalablecluster` is 集群。为此，请使用在名为 `config.json` 的文件中保存的策略配置。

对于 Linux、macOS 或 Unix：

```
aws application-autoscaling put-scaling-policy \
    --policy-name myscalablepolicy \
    --policy-type TargetTrackingScaling \
    --resource-id replication-group/myscalablecluster \
    --service-namespace elasticache \
    --scalable-dimension elasticache:replication-group:NodeGroups \
    --target-tracking-scaling-policy-configuration file://config.json
```

对于 Windows：

```
aws application-autoscaling put-scaling-policy ^
    --policy-name myscalablepolicy ^
    --policy-type TargetTrackingScaling ^
    --resource-id replication-group/myscalablecluster ^
    --service-namespace elasticache ^
    --scalable-dimension elasticache:replication-group:NodeGroups ^
    --target-tracking-scaling-policy-configuration file://config.json
```

使用 API 应用扩展策略

要将扩展策略应用于您的 f ElastiCache or Redis 集群，请使用带有以下参数的PutScalingPolicy AWS CLI 命令：

- --policy-name – 扩展策略的名称。
- --resource-id — Redis 的资源标识符。 ElastiCache 例如`replication-group/myscalablecluster`，对于此参数，资源类型为`ReplicationGroup`，唯一标识符是 Redi ElastiCache s 集群的名称。
- --service-namespace – 将此值设置为 `elasticache`。

- --scalable-dimension – 将此值设置为 `elasticache:replication-group:NodeGroups`。
- --target-tracking-scaling-policy-配置 — 用于 Redis 集群的目标跟踪扩展策略配置 ElastiCache 。

在以下示例中，您将名 ElastiCache 为的目标跟踪扩展策略应用于名为 `myscalablepolicy` for Redis 自动缩放的 R ElastiCache ed `myscalablecluster` is 集群。您使用的策略配置基于 `ElastiCachePrimaryEngineCPUUtilization` 预定义指标。

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.PutScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:NodeGroups",
    "PolicyType": "TargetTrackingScaling",
    "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 40.0,
        "PredefinedMetricSpecification":
        {
            "PredefinedMetricType": "ElastiCachePrimaryEngineCPUUtilization"
        }
    }
}
```

编辑扩展策略

您可以使用 AWS Management Console、AWS CLI 或 Application Auto Scaling API 编辑扩缩策略。

使用 AWS Management Console编辑扩展策略

编辑 ElastiCache for Redis 集群的弹性伸缩策略

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。

2. 在导航窗格中，选择 Redis。
3. 选择您要向其中添加策略的集群（选择集群名称，而不是其左侧的按钮）。
4. 选择 Auto Scaling policies（弹性伸缩策略）选项卡。
5. 在 Scaling policies（扩展策略）下，选择要更改的 Auto Scaling 策略左侧的按钮，然后选择 Modify（修改）。
6. 对该策略做出必要更改。
7. 选择 Modify（修改）。

使用 AWS CLI 或 API 编辑扩展策略

您可以使用 AWS CLI 或 Application Auto Scaling API 按照与应用扩缩策略相同的方式编辑扩缩策略：

- 在使用 AWS CLI 时，请在 `--policy-name` 参数中指定要编辑的策略名称。为要更改的参数指定新的值。
- 在使用 Application Auto Scaling API 时，请在 `PolicyName` 参数中指定要编辑的策略名称。为要更改的参数指定新的值。

有关更多信息，请参阅应用扩缩策略。

删除扩展策略

您可以使用AWS Management Console、AWS CLI 或 Application Auto Scaling API 删除扩缩策略。

使用 AWS Management Console删除扩展策略

删除 ElastiCache for Redis 集群的弹性伸缩策略

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在导航窗格中，选择 Redis。
3. 选择要编辑其 Auto Scaling 策略的集群（选择集群名称，而不是其左侧的按钮）。
4. 选择 Auto Scaling policies（弹性伸缩策略）选项卡。
5. 在 Scaling policies（扩展策略）下，选择 Auto Scaling 策略，然后选择 Delete（删除）。

使用 AWS CLI删除扩展策略

要删除 ElastiCache for Redis 集群的扩展策略，请使用包含下列参数的 delete-scaling-policy AWS CLI 命令：

- --policy-name – 扩展策略的名称。
- --resource-id – ElastiCache for Redis 的资源标识符。对于此参数，资源类型为 `ReplicationGroup`，唯一标识符为 ElastiCache for Redis 集群的名称，例如 `replication-group/myscalablecluster`。
- --service-namespace – 将此值设置为 `elasticache`。
- --scalable-dimension – 将此值设置为 `elasticache:replication-group:NodeGroups`。

在以下示例中，您从名为 `myscalablecluster` 的 ElastiCache for Redis 集群中删除名为 `myscalablepolicy` 的目标跟踪扩缩策略。

对于 Linux、macOS 或 Unix：

```
aws application-autoscaling delete-scaling-policy \
    --policy-name myscalablepolicy \
    --resource-id replication-group/myscalablecluster \
    --service-namespace elasticache \
    --scalable-dimension elasticache:replication-group:NodeGroups
```

对于 Windows：

```
aws application-autoscaling delete-scaling-policy ^
    --policy-name myscalablepolicy ^
    --resource-id replication-group/myscalablecluster ^
    --service-namespace elasticache ^
    --scalable-dimension elasticache:replication-group:NodeGroups
```

使用 API 删除扩展策略

要删除 ElastiCache for Redis 集群的扩展策略，请使用包含下列参数的 DeleteScalingPolicy AWS CLI 命令：

- --policy-name – 扩展策略的名称。

- --resource-id – ElastiCache for Redis 的资源标识符。对于此参数，资源类型为 ReplicationGroup，唯一标识符为 ElastiCache for Redis 集群的名称，例如 replication-group/myscalablecluster。
- --service-namespace – 将此值设置为 elasticache。
- --scalable-dimension – 将此值设置为 elasticache:replication-group:NodeGroups。

在以下示例中，您从名为 myscalablecluster 的 ElastiCache for Redis 集群中删除名为 myscalablepolicy 的目标跟踪扩缩策略。

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:NodeGroups"
}
```

将 AWS CloudFormation 用于 Auto Scaling 策略

此代码段演示如何使用 AWS::ApplicationAutoScaling::ScalableTarget 资源创建目标跟踪策略并将其应用于 AWS::ElastiCache::ReplicationGroup 资源。此示例利用 Fn::Join 和 Ref 内置函数，使用在同一模板中指定的 AWS::ElastiCache::ReplicationGroup 资源的逻辑名称来构建 ResourceId 属性。

```
ScalingTarget:
   Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
   Properties:
     MaxCapacity: 3
     MinCapacity: 1
     ResourceId: !Sub replication-group/${logicalName}
     ScalableDimension: 'elasticache:replication-group:NodeGroups'
     ServiceNamespace: elasticache
```

```
     RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"

  ScalingPolicy:
    Type: "AWS::ApplicationAutoScaling::ScalingPolicy"
    Properties:
      ScalingTargetId: !Ref ScalingTarget
      ServiceNamespace: elasticache
      PolicyName: testpolicy
      PolicyType: TargetTrackingScaling
      ScalableDimension: 'elasticache:replication-group:NodeGroups'
      TargetTrackingScalingPolicyConfiguration:
        PredefinedMetricSpecification:
          PredefinedMetricType: ElastiCachePrimaryEngineCPUUtilization
        TargetValue: 40
```

计划扩缩

按计划扩展使您可以按照可预测的需求变化来扩展应用程序。要使用计划扩展，请创建指示 ElastiCache for Redis 在特定时间执行扩缩活动的计划操作。创建计划操作时，您可以指定现有的 ElastiCache for Redis 集群、执行扩缩活动的时间、最小容量和最大容量。您可以创建仅扩展一次或按重复计划扩展的计划操作。

您只能为已存在的 ElastiCache for Redis 集群创建计划操作。您不能在创建集群的同时创建计划操作。

有关计划操作创建、管理和删除的相关术语的详细信息，请参阅计划操作创建、管理和删除的常用命令

创建定期计划：

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在导航窗格中，选择 Redis。
3. 选择要对其添加策略的集群。
4. 从 Actions（操作）下拉菜单中选择 Manage Auto Scaling policies（管理弹性伸缩策略）。
5. 选择 Auto Scaling policies（Auto Scaling 策略）选项卡。
6. Auto scaling policies（弹性伸缩策略）部分中会显示 Add Scaling policy（添加扩缩策略）对话框。选择 Scheduled scaling（计划扩展）。
7. 对于 Policy name（策略名称），请输入策略的名称。

8. 对于 Scalable Dimension（可扩展维度），选择 Shards（分区）。
9. 对于 Target Shards（目标分区），请选择值。
10. 对于 Recurrence（重复），请选择 Recurring（定期）。
11. 对于 Frequency（频率），请选择相应的值。
12. 对于 Start Date（开始日期）和 Start time（开始时间），请选择策略开始生效的时间。
13. 选择 Add Policy（添加策略）。

创建一次性计划操作：

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在导航窗格中，选择 Redis。
3. 选择要对其添加策略的集群。
4. 从 Actions（操作）下拉菜单中选择 Manage Auto Scaling policies（管理弹性伸缩策略）。
5. 选择 Auto Scaling policies（Auto Scaling 策略）选项卡。
6. Auto scaling policies（弹性伸缩策略）部分中会显示 Add Scaling policy（添加扩缩策略）对话框。选择 Scheduled scaling（计划扩展）。
7. 对于 Policy name（策略名称），请输入策略的名称。
8. 对于 Scalable Dimension（可扩展维度），选择 Shards（分区）。
9. 对于 Target Shards（目标分区），请选择值。
10. 对于 Recurrence（重复），请选择 Once（一次）。
11. 对于 Start Date（开始日期）和 Start time（开始时间），请选择策略开始生效的时间。
12. 对于 End Date（结束日期），请选择策略生效结束日期。
13. 选择 Add Policy（添加策略）。

删除计划操作

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在导航窗格中，选择 Redis。
3. 选择要对其添加策略的集群。

4. 从 Actions（操作）下拉菜单中选择 Manage Auto Scaling policies（管理弹性伸缩策略）。
5. 选择 Auto Scaling policies（Auto Scaling 策略）选项卡。
6. 在 Auto Scaling policies（弹性伸缩策略）部分，选择弹性伸缩策略，然后从 Actions（操作）对话中选择 Delete（删除）。

使用 AWS CLI 管理计划扩展

使用以下 application-autoscaling API：

- put-scheduled-action
- describe-scheduled-actions
- delete-scheduled-action

使用 AWS CloudFormation 创建计划的操作

此代码段演示如何使用 AWS::ApplicationAutoScaling::ScalableTarget 资源创建目标跟踪策略并将其应用于 AWS::ElastiCache::ReplicationGroup 资源。此示例利用 Fn::Join 和 Ref 内置函数，使用在同一模板中指定的 AWS::ElastiCache::ReplicationGroup 资源的逻辑名称来构建 ResourceId 属性。

```
ScalingTarget:
   Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
   Properties:
     MaxCapacity: 3
     MinCapacity: 1
     ResourceId: !Sub replication-group/${logicalName}
     ScalableDimension: 'elasticache:replication-group:NodeGroups'
     ServiceNamespace: elasticache
     RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"
     ScheduledActions:
       - EndTime: '2020-12-31T12:00:00.000Z'
         ScalableTargetAction:
           MaxCapacity: '5'
           MinCapacity: '2'
         ScheduledActionName: First
         Schedule: 'cron(0 18 * * ? *)'
```

将弹性伸缩与副本结合使用

下面提供了有关目标跟踪和计划策略以及如何使用 AWS Management Console AWS CLI 和 API 应用它们的详细信息。

目标跟踪扩缩策略

在使用目标跟踪扩展策略时，您可以选择一个指标并设置一个目标值。ElastiCache for Redis 弹性伸缩创建和管理触发扩展策略的 CloudWatch 告警，并根据指标和目标值计算扩展调整。扩展策略根据需要均匀地添加或删除所有分区的副本，以便将指标保持在指定的目标值或接近该值。除了将指标保持在目标值附近以外，目标跟踪扩展策略还会根据由于负载模式波动而造成的指标波动进行调节，并最大限度减少队列容量发生快速波动的情况。

副本的弹性伸缩条件

弹性伸缩策略为您的集群定义以下预定义指标：

`ElastiCacheReplicaEngineCPUUtilization`：跨所有副本聚合的 AVG 引擎 CPU 使用率阈值，ElastiCache for Redis 使用此阈值触发弹性伸缩操作。您可以将此使用率目标设置为 35% 到 70% 之间。

当服务检测到 `ElastiCacheReplicaEngineCPUUtilization` 指标等于或大于“目标”设置时，其会自动提高所有分区的副本。ElastiCache for Redis 按等于以下两个数字之间较大者的计数来横向扩展集群副本：与目标之间差异的百分比和 1 个副本。对于横向缩减，ElastiCache for Redis 不会弹性横向缩减，除非整体指标值低于所定义的目标的 75%。

关于横向扩展示例，如果您有 5 个分区，每个分区中有 1 个副本：

如果目标违例了 30%，则 ElastiCache for Redis 在所有分区上横向扩展 1 个副本（最大 0.3，默认为 1），最后的结果是有 5 个分区，每个分区包含 2 个副本，

对横向缩减示例，如果您选择的目标值是 60%，则 ElastiCache for Redis 会直到该指标小于或等于 45%（25% 低于目标 60%）时才会弹性横向缩减。

Auto Scaling 注意事项

请注意以下事项：

- 目标跟踪扩展策略假设它应该在指定指标高于目标值时执行向外扩展。因此，不能使用目标跟踪扩展策略在指定指标低于目标值时向外扩展。ElastiCache for Redis 按集群中所有分区的现有副本的最大数量（偏离目标的偏差经四舍五入的百分比，默认为 1）横向扩展副本。

- 当指定指标数据不足时，目标跟踪扩展策略不会执行扩展。它不会执行向内扩展，因为它不会将数据不足解读为使用率低。
- 您可能会看到目标值与实际指标数据点之间存在差距。这是因为 ElastiCache for Redis 弹性伸缩在确定要添加或删除多少容量时将始终通过向上或向下舍入保守地进行操作。以免添加的容量不足或删除的容量过多。
- 为了确保应用程序可用性，服务会针对指标尽快按比例横向扩展，但渐进式横向缩减，集群中所有分区的最大横向缩减副本数为 1。
- 您可以为 ElastiCache for Redis 集群提供多个目标跟踪扩缩策略，前提是它们各自使用不同的指标。ElastiCache for Redis 弹性伸缩的目的是始终优先考虑可用性，因此其行为会有所不同，具体取决于目标跟踪策略是否已准备好横向扩展或横向缩减。如果任何目标跟踪策略已准备好进行扩展，它将扩展服务，但仅在所有目标跟踪策略（启用了缩减部分）准备好缩减时才执行缩减。
- 请勿编辑或删除 ElastiCache for Redis 弹性伸缩为目标跟踪扩缩策略管理的 CloudWatch 告警。在您删除扩缩策略或删除集群时，ElastiCache for Redis 弹性伸缩会自动删除相应的告警。
- ElastiCache for Redis 弹性伸缩不会阻止您手动修改分区中的副本。这些手动调整不会影响附加到扩展策略的任何现有 CloudWatch 告警，但可能会影响可触发这些 CloudWatch 告警的指标。
- 这些由弹性伸缩管理的 CloudWatch 告警是通过集群中所有分区的 AVG 指标来定义的。因此，拥有热分区可能会导致以下任一情况：
  - 因若干热分区上负载触发 CloudWatch 告警而导致在不需要扩展时执行扩展
  - 因所有分区的影响告警的聚合 AVG 指标不违例而在需要扩展时不执行扩展。
- ElastiCache for Redis 对每个集群的节点的默认限制仍然适用。因此，当选择弹性伸缩时，如果您希望最大节点数超过默认限制，请在 AWS Service Limits 请求提高限制，并选择限制类型 Nodes per cluster per instance type（每个实例类型的集群的节点数）。
- 请确保您的 VPC 中有足够的 ENI（弹性网络接口）可用，横向扩展过程中需要弹性网络接口。有关更多信息，请参阅弹性网络接口。
- 如果 EC2 没有足够的可用容量，则 ElastiCache for Redis 弹性伸缩会直到容量可用或您手动将集群修改为具有足够容量的实例类型才会横向扩展。
- ElastiCache for Redis 弹性伸缩不支持扩展其集群 `ReservedMemoryPercent` 低于 25% 的副本。有关更多信息，请参阅管理预留内存。

添加扩展策略

您可以使用添加扩展策略 AWS Management Console。

使用添加扩展策略 AWS Management Console

向 ElastiCache 适用于 Redis 的添加自动缩放策略

1. 登录 AWS Management Console 并打开亚马逊 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格中，选择 Redis。
3. 选择您要向其中添加策略的集群（选择集群名称，而不是其左侧的按钮）。
4. 选择 Auto Scaling policies（弹性伸缩策略）选项卡。
5. 选择 add dynamic scaling（添加动态扩展）。
6. 在 Scaling policies（扩展策略）下，选择 Add dynamic scaling（添加动态扩展）。
7. 对于 Policy name（策略名称），请输入策略的名称。
8. 对于 Scalable Dimension（可扩展维度），从对话框中选择 Replicas（副本）。
9. 在目标值中，键入要在 ElastiCache 副本上保持的 CPU 使用率的平均百分比。此值必须介于 35 到 70 之间。将添加或删除集群副本以使指标接近于指定的值。
10. （可选）控制台不支持横向缩减或横向扩展冷却时间。使用 AWS CLI 修改冷却值。
11. 在最小容量中，键入适用 ElastiCache 于 Redis Auto Scaling 策略需要维护的最小副本数。
12. 在最大容量中，键入适用于 Redis Auto Scaling 策略需要维护的最大副本数。 ElastiCache 此值必须大于或等于 5。
13. 选择创建。

注册可扩展目标

您可以应用基于预定义或自定义指标的扩展策略。为此，您可以使用 AWS CLI 或 Application Auto Scaling API。第一步是注册你的 Redi ElastiCache s 复制组，以便 Redi ElastiCache s 自动扩展。

在使用 ElastiCache 适用于 Redis 的集群进行 Redis 自动缩放之前，您需要先注册集群 ElastiCache 以实现 Redis 自动缩放。 ElastiCache 您这样做是为了定义要应用于该集群的缩放维度和限制。ElastiCache for Redis auto scaling 沿着`elasticache:replication-group:Replicas`可扩展维度动态扩展 for Redis 集群，该维度表示每个分片的集群副本数。 ElastiCache

使用 CLI

要注册您的 ElastiCache 集群，请使用带有以下参数的register-scalable-target命令：

- --service-namespace – 将此值设置为 elasticache。

- --resource-id — 集群的资源标识符。 ElastiCache 例如`replication-group/myscalablecluster`，对于此参数，资源类型为 ReplicationGroup ，唯一标识符是 Redi ElastiCache s 集群的名称。
- --scalable-dimension – 将此值设置为 `elasticache:replication-group:Replicas`。
- --min-capacity — ElastiCache Redis 自动缩放要管理的最小副本数量。有关 --min-capacity、--max-capacity 和集群中副本数之间关系的信息，请参阅 最小和最大容量。
- --max-capacity — ElastiCache Redis 自动缩放要管理的最大副本数。有关 --min-capacity、--max-capacity 和集群中副本数之间关系的信息，请参阅 最小和最大容量。

Example

在以下示例中，您注册了一个名为 `myscalablecluster` Redis ElastiCache 的集群。该注册表示应将集群动态扩展为具有 1 到 5 个副本。

对于 Linux、macOS 或 Unix：

```
aws application-autoscaling register-scalable-target \
    --service-namespace elasticache \
    --resource-id replication-group/myscalablecluster \
    --scalable-dimension elasticache:replication-group:Replicas \
    --min-capacity 1 \
    --max-capacity 5 \
```

对于 Windows：

```
aws application-autoscaling register-scalable-target ^
    --service-namespace elasticache ^
    --resource-id replication-group/myscalablecluster ^
    --scalable-dimension elasticache:replication-group:Replicas ^
    --min-capacity 1 ^
    --max-capacity 5 ^
```

使用 API

要注册您的 ElastiCache 集群，请使用带有以下参数的register-scalable-target命令：

- ServiceNamespace — 将此值设置为 elasticache。
- 资源 ID-群集的资源标识符。 ElastiCache 例如`replication-group/myscalablecluster`，对于此参数，资源类型为 ReplicationGroup ，唯一标识符是 Redi ElastiCache s 集群的名称。

- ScalableDimension — 将此值设置为`elasticache:replication-group:Replicas`。
- MinCapacity — Redis auto 缩放要管理的最小副本数量。 ElastiCache 有关 --min-capacity、--max-capacity 和集群中副本数之间关系的信息，请参阅 最小和最大容量。
- MaxCapacity — Redis auto 缩放要管理的最大副本数。 ElastiCache 有关 --min-capacity、--max-capacity 和集群中副本数之间关系的信息，请参阅 最小和最大容量。

Example

在以下示例中，您将使用 Application Auto Scaling API 注册一个 ElastiCache 名`myscalablecluster`为 Redis 的集群。该注册表示应将集群动态扩展为具有 1 到 5 个副本。

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.RegisterScalableTarget
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:Replicas",
    "MinCapacity": 1,
    "MaxCapacity": 5
}
```

定义扩缩策略

目标跟踪扩展策略配置是由 JSON 块表示的，其中定义了指标和目标值。您可以在文本文件中将扩展策略配置保存为 JSON 块。在调用 AWS CLI 或 Application Auto Scaling API 时，您可以使用该文本文件。有关策略配置语法的更多信息，请参阅 Application Auto Scaling API 参考中的 TargetTrackingScalingPolicyConfiguration。

您可以使用以下选项定义目标跟踪扩缩策略配置：

主题

- 使用预定义的指标

- 编辑扩展策略
- 删除扩展策略
- 将 AWS CloudFormation 用于 Auto Scaling 策略
- 计划扩缩

使用预定义的指标

目标跟踪扩展策略配置是由 JSON 块表示的，其中定义了指标和目标值。您可以在文本文件中将扩展策略配置保存为 JSON 块。在调用 AWS CLI 或 Application Auto Scaling API 时，您可以使用该文本文件。有关策略配置语法的更多信息，请参阅 Application Auto Scaling API 参考中的 TargetTrackingScalingPolicyConfiguration。

您可以使用以下选项定义目标跟踪扩缩策略配置：

主题

- 使用预定义的指标
- 使用自定义指标
- 使用冷却时间
- 禁用横向缩减活动
- 对 ElastiCache for Redis 集群应用扩展策略

使用预定义的指标

通过使用预定义的指标，您可以快速为 ElastiCache for Redis 集群定义与 ElastiCache for Redis 弹性伸缩中的目标跟踪搭配使用的目标跟踪扩缩策略。目前，ElastiCache for Redis 支持 ElastiCache 副本自动扩缩中的以下预定义指标：

`ElastiCacheReplicaEngineCPUUtilization` – ElastiCache for Redis 集群中所有副本在 CloudWatch 中的 EngineCPUUtilization 指标的平均值。ElastiCache for Redis 集群中所有副本在 CloudWatch 中的 `EngineCPUUtilization` 指标的平均值。您可以在 CloudWatch 中在 ElastiCache for Redis `ReplicationGroupId, Role` 下找到聚合指标值，获得所需的 ReplicationGroupId 和角色副本。

要在扩展策略中使用预定义的指标，您需要为扩展策略创建一个目标跟踪配置。该配置必须包含 `PredefinedMetricSpecification` 以表示预定义的指标，并包含 `TargetValue` 以表示该指标的目标值。

使用自定义指标

通过使用自定义指标，您可以定义满足您的自定义要求的目标跟踪扩展策略。您可以根据随扩展按比例变化的任何 ElastiCache for Redis 指标来定义自定义指标。并非所有 ElastiCache for Redis 指标都适用于目标跟踪。指标必须是有效的使用率指标，它用于描述实例的繁忙程度。指标值必须随集群中的副本数按比例增加或减少。要使用指标数据按比例增加或减少副本数，必须按比例进行这种增加或减少。

Example

以下示例说明了扩缩策略的目标跟踪配置。在该配置中，一个自定义指标根据名为 `my-db-cluster` 的集群中所有副本的平均 CPU 使用率 50% 调整该 ElastiCache for Redis 集群。

```
{"TargetValue": 50,
    "CustomizedMetricSpecification":
    {"MetricName": "EngineCPUUtilization",
        "Namespace": "AWS/ElastiCache",
        "Dimensions": [
            {"Name": "RelicationGroup","Value": "my-db-cluster"},
            {"Name": "Role","Value": "REPLICA"}
        ],
        "Statistic": "Average",
        "Unit": "Percent"
    }
}
```

使用冷却时间

您可以为 `ScaleOutCooldown` 指定一个值（秒）以添加横向扩展集群的冷却时间。同样，您可以为 `ScaleInCooldown` 添加一个值（秒）以添加横向缩减集群的冷却时间。有关 `ScaleInCooldown` 和 `ScaleOutCooldown` 的更多信息，请参阅 Application Auto Scaling API 参考中的 TargetTrackingScalingPolicyConfiguration。以下示例说明了扩缩策略的目标跟踪配置。在该配置中，`ElastiCacheReplicaEngineCPUUtilization` 预定义指标用于根据集群中所有副本的平均 CPU 使用率 40% 调整该 ElastiCache for Redis 集群。该配置将缩减冷却时间指定为 10 分钟，并将扩展冷却时间指定为 5 分钟。

```
{"TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {"PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"
    },
    "ScaleInCooldown": 600,
    "ScaleOutCooldown": 300
```

```
}
```

禁用横向缩减活动

您可以禁用横向缩减活动以禁止目标跟踪扩缩策略配置横向缩减r ElastiCache for Redis 集群。禁用横向缩减活动将禁止扩展策略删除副本，同时仍允许扩展策略根据需要添加副本。

您可以为 `DisableScaleIn` 指定一个布尔值，以便为集群启用或禁用横向缩减活动。有关 `DisableScaleIn` 的更多信息，请参阅 Application Auto Scaling API 参考中的 TargetTrackingScalingPolicyConfiguration。

Example

以下示例说明了扩缩策略的目标跟踪配置。在该配置中，`ElastiCacheReplicaEngineCPUUtilization` 预定义指标根据集群中所有副本的平均 CPU 使用率 40% 调整该 ElastiCache for Redis 集群。该配置禁用扩展策略的缩减活动。

```
{"TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {"PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"
    },
    "DisableScaleIn": true
}
```

对 ElastiCache for Redis 集群应用扩展策略

在对集群注册 ElastiCache for Redis 弹性伸缩并定义扩展策略后，您可以将扩展策略应用于已注册的集群。要将扩展策略应用于 ElastiCache for Redis 集群，您可以使用 AWS CLI 或 Application Auto Scaling API。

使用 AWS CLI

要将扩展策略应用于 ElastiCache for Redis 集群，请使用具有以下参数的 put-scaling-policy 命令：

- --policy-name – 扩展策略的名称。
- --policy-type – 将此值设置为 `TargetTrackingScaling`。
- --resource-id – 该 ElastiCache for Redis 集群的资源标识符。对于此参数，资源类型为复制组，唯一标识符为 ElastiCache for Redis 集群的名称，例如 `replication-group/myscalablecluster`。
- --service-namespace – 将此值设置为 elasticache。

- --scalable-dimension – 将此值设置为 `elasticache:replication-group:Replicas`。
- --target-tracking-scaling-policy-configuration – 用于 ElastiCache for Redis 集群的目标跟踪扩缩策略配置。

Example

在以下示例中，您使用 ElastiCache for Redis 弹性伸缩对名为 `myscalablecluster` 的 ElastiCache for Redis 集群应用名为 `myscalablepolicy` 的目标跟踪扩缩策略。为此，请使用在名为 `config.json` 的文件中保存的策略配置。

对于 Linux、macOS 或 Unix：

```
aws application-autoscaling put-scaling-policy \
    --policy-name myscalablepolicy \
    --policy-type TargetTrackingScaling \
    --resource-id replication-group/myscalablecluster \
    --service-namespace elasticache \
    --scalable-dimension elasticache:replication-group:Replicas \
    --target-tracking-scaling-policy-configuration file://config.json
```

```
{"TargetValue": 40.0,
    "PredefinedMetricSpecification":
    {"PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"
    },
    "DisableScaleIn": true
}
```

对于 Windows：

```
aws application-autoscaling put-scaling-policy ^
    --policy-name myscalablepolicy ^
    --policy-type TargetTrackingScaling ^
    --resource-id replication-group/myscalablecluster ^
    --service-namespace elasticache ^
    --scalable-dimension elasticache:replication-group:Replicas ^
    --target-tracking-scaling-policy-configuration file://config.json
```

使用 API

要使用 Application Auto Scaling API 将扩缩策略应用于 ElastiCache for Redis 集群，请使用具有以下参数的 PutScalingPolicy Application Auto Scaling API 操作：

- PolicyName – 扩展策略的名称。
- PolicyType – 将此值设置为 `TargetTrackingScaling`。
- ResourceID – 该 ElastiCache for Redis 集群的资源标识符。对于此参数，资源类型为复制组，唯一标识符为 ElastiCache for Redis 集群的名称，例如 `replication-group/myscalablecluster`。
- ServiceNamespace – 将此值设置为 elasticache。
- ScalableDimension – 将此值设置为 `elasticache:replication-group:Replicas`。
- TargetTrackingScalingPolicyConfiguration – 用于 ElastiCache for Redis 集群的目标跟踪扩缩策略配置。

Example

在以下示例中，您使用 ElastiCache for Redis 弹性伸缩对名为 `myscalablecluster` 的 ElastiCache for Redis 集群应用名为 `scalablepolicy` 的目标跟踪扩缩策略。您使用的策略配置基于 `ElastiCacheReplicaEngineCPUUtilization` 预定义指标。

```
POST / HTTP/1.1
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.PutScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:Replicas",
    "PolicyType": "TargetTrackingScaling",
    "TargetTrackingScalingPolicyConfiguration": {
        "TargetValue": 40.0,
        "PredefinedMetricSpecification":
        {
```

```
            "PredefinedMetricType": "ElastiCacheReplicaEngineCPUUtilization"
        }
    }
}
```

编辑扩展策略

您可以使用 AWS Management Console、AWS CLI 或 Application Auto Scaling API 编辑扩缩策略。

使用 AWS Management Console编辑扩展策略

您只能通过 AWS Management Console 使用预定义类型指标编辑策略

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在导航窗格中，选择 Redis
3. 选择您要向其中添加策略的集群（选择集群名称，而不是其左侧的按钮）。
4. 选择 Auto Scaling policies（弹性伸缩策略）选项卡。
5. 在 Scaling policies（扩展策略）下，选择要更改的 Auto Scaling 策略左侧的按钮，然后选择 Modify（修改）。
6. 对该策略做出必要更改。
7. 选择 Modify（修改）。
8. 对该策略进行更改。
9. 选择 Modify（修改）。

使用 AWS CLI 或 Application Auto Scaling API 编辑扩缩策略

您可以使用 AWS CLI 或 Application Auto Scaling API 按照与应用扩缩策略相同的方式编辑扩缩策略：

- 在使用 Application Auto Scaling API 时，请在 `PolicyName` 参数中指定要编辑的策略名称。为要更改的参数指定新的值。

有关更多信息，请参阅对 ElastiCache for Redis 集群应用扩展策略。

删除扩展策略

你可以使用 AWS Management Console、 AWS CLI 或 Application Auto Scaling API 删除扩展策略

使用删除扩展策略 AWS Management Console

您只能通过 AWS Management Console使用预定义类型指标编辑策略

1. 登录 AWS Management Console 并打开亚马逊 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格中，选择 Redis
3. 选择要删除其弹性伸缩策略的集群。
4. 选择 Auto Scaling policies（Auto Scaling 策略）选项卡。
5. 在 Scaling policies（扩展策略）下，选择 Auto Scaling 策略，然后选择 Delete（删除）。

使用 AWS CLI 或 Application Auto Scaling API 删除扩展策略

您可以使用 AWS CLI 或 Application Auto Scaling API 从 ElastiCache 集群中删除扩展策略。

CLI

要从 for Redis 集群中删除扩展策略，请使用带有以下参数的delete-scaling-policy命令： ElastiCache

- --policy-name – 扩展策略的名称。
- --resource-id — Redis 集群的资源标识符 ElastiCache 。例如，对于此参数，资源类型为 ReplicationGroup ，唯一标识符是 ElastiCache 群集的名称`replication-group/myscalablecluster`。
- --service-namespace – 将此值设置为 elasticache。
- --scalable-dimension – 将此值设置为 `elasticache:replication-group:Replicas`。

Example

在以下示例中，您从名为 `myscalablecluster` 的 ELC 集群中删除名为 `myscalablepolicy` 的目标跟踪扩缩策略。

对于 Linux、macOS 或 Unix：

```
aws application-autoscaling delete-scaling-policy \
    --policy-name myscalablepolicy \
    --resource-id replication-group/myscalablecluster \
    --service-namespace elasticache \
    --scalable-dimension elasticache:replication-group:Replicas \
```

对于 Windows：

```
aws application-autoscaling delete-scaling-policy ^
    --policy-name myscalablepolicy ^
    --resource-id replication-group/myscalablecluster ^
    --service-namespace elasticache ^
    --scalable-dimension elasticache:replication-group:Replicas ^
```

API

要从 For Redis 集群中删除扩展策略，请使用带有以下参数的 App DeleteScalingPolicylication Auto Scaling API 操作： ElastiCache

- PolicyName — 扩展策略的名称。
- 资源 ID — Redis 集群的资源标识符 ElastiCache 。例如，对于此参数，资源类型为 ReplicationGroup ，唯一标识符是 ElastiCache 群集的名称`replication-group/myscalablecluster`。
- ServiceNamespace — 将此值设置为 elasticache。
- ScalableDimension — 将此值设置为`elasticache:replication-group:Replicas`。

在以下示例中，您将`myscalablepolicy`从以 Application Auto Scaling API 命名的适用于 Redis ElastiCache 的集群中删除名为`myscalablecluster`的目标跟踪扩展策略。

```
POST / HTTP/1.1
>>>>>>> mainline
Host: autoscaling.us-east-2.amazonaws.com
Accept-Encoding: identity
Content-Length: 219
X-Amz-Target: AnyScaleFrontendService.DeleteScalingPolicy
X-Amz-Date: 20160506T182145Z
User-Agent: aws-cli/1.10.23 Python/2.7.11 Darwin/15.4.0 botocore/1.4.8
Content-Type: application/x-amz-json-1.1
Authorization: AUTHPARAMS
{
    "PolicyName": "myscalablepolicy",
    "ServiceNamespace": "elasticache",
    "ResourceId": "replication-group/myscalablecluster",
    "ScalableDimension": "elasticache:replication-group:Replicas"
```

```
}
```

将 AWS CloudFormation 用于 Auto Scaling 策略

此代码段演示如何使用 AWS::ApplicationAutoScaling::ScalableTarget 资源创建计划操作并将其应用于 AWS::ElastiCache::ReplicationGroup 资源。此示例利用 Fn::Join 和 Ref 内置函数，使用在同一模板中指定的 AWS::ElastiCache::ReplicationGroup 资源的逻辑名称来构建 ResourceId 属性。

```
ScalingTarget:
   Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
   Properties:
     MaxCapacity: 0
     MinCapacity: 0
     ResourceId: !Sub replication-group/${logicalName}
     ScalableDimension: 'elasticache:replication-group:Replicas'
     ServiceNamespace: elasticache
     RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"

  ScalingPolicy:
    Type: "AWS::ApplicationAutoScaling::ScalingPolicy"
    Properties:
      ScalingTargetId: !Ref ScalingTarget
      ServiceNamespace: elasticache
      PolicyName: testpolicy
      PolicyType: TargetTrackingScaling
      ScalableDimension: 'elasticache:replication-group:Replicas'
      TargetTrackingScalingPolicyConfiguration:
        PredefinedMetricSpecification:
          PredefinedMetricType: ElastiCacheReplicaEngineCPUUtilization
        TargetValue: 40
```

计划扩缩

按计划扩展使您可以按照可预测的需求变化来扩展应用程序。要使用计划扩展，请创建指示 ElastiCache for Redis 在特定时间执行扩缩活动的计划操作。创建计划操作时，您可以指定现有的 ElastiCache for Redis 集群、执行扩缩活动的时间、最小容量和最大容量。您可以创建仅扩展一次或按重复计划扩展的计划操作。

您只能为已存在的 ElastiCache for Redis 集群创建计划操作。您不能在创建集群的同时创建计划操作。

有关计划操作创建、管理和删除的相关术语的详细信息，请参阅计划操作创建、管理和删除的常用命令

创建一次性计划操作：

类似于分区维度。请参阅计划扩缩 。

删除计划操作

类似于分区维度。请参阅计划扩缩 。

使用 AWS CLI 管理计划扩展

使用以下 application-autoscaling API：

- put-scheduled-action
- describe-scheduled-actions
- delete-scheduled-action

使用 AWS CloudFormation 创建弹性伸缩策略

此代码段演示如何使用 AWS::ApplicationAutoScaling::ScalableTarget 资源创建计划操作并将其应用于 AWS::ElastiCache::ReplicationGroup 资源。此示例利用 Fn::Join 和 Ref 内置函数，使用在同一模板中指定的 AWS::ElastiCache::ReplicationGroup 资源的逻辑名称来构建 ResourceId 属性。

```
ScalingTarget:
   Type: 'AWS::ApplicationAutoScaling::ScalableTarget'
   Properties:
     MaxCapacity: 0
     MinCapacity: 0
     ResourceId: !Sub replication-group/${logicalName}
     ScalableDimension: 'elasticache:replication-group:Replicas'
     ServiceNamespace: elasticache
     RoleARN: !Sub "arn:aws:iam::${AWS::AccountId}:role/aws-
service-role/elasticache.application-autoscaling.amazonaws.com/
AWSServiceRoleForApplicationAutoScaling_ElastiCacheRG"
     ScheduledActions:
       - EndTime: '2020-12-31T12:00:00.000Z'
         ScalableTargetAction:
           MaxCapacity: '5'
```

```
          MinCapacity: '2'
        ScheduledActionName: First
        Schedule: 'cron(0 18 * * ? *)'
```

修改集群模式

Redis 是一个分布式内存数据库，支持分片和复制。ElastiCache for Redis 集群是 Redis 的分布式实现，它允许在多个 Redis 节点间对数据进行分区。ElastiCache for Redis 集群有两种操作模式：“已启用集群模式”（CME）和“已禁用集群模式”（CMD）。在 CME 中，Redis 充当具有多个分片和节点的分布式数据库，而在 CMD 中，Redis 充当单个节点。

从 CMD 迁移到 CME 之前，必须满足以下条件：

> **Important**
>
> 集群模式配置只能从“已禁用集群模式”更改为“已启用集群模式”。无法还原此配置。

- 集群只能在数据库 0 中拥有密钥。
- 应用程序必须使用能够使用集群协议的 Redis 客户端，并使用配置端点。
- 必须在至少有 1 个副本的集群上启用自动失效转移。
- 迁移所需的最低 Redis 引擎版本为 7.0。

要从 CMD 迁移到 CME，必须将集群模式配置从“已禁用集群模式”更改为“已启用集群模式”。这是一个两步过程，可确保迁移过程中集群的可用性。

> **Note**
>
> 您需要为参数组提供已启用集群的配置，也就是说，cluster-enabled 参数设置为 `yes`。如果您使用的是原定设置参数组，则 ElastiCache for Redis 将自动选择具有已启用集群的配置的相应原定设置参数组。对于 CMD 集群，cluster-enabled 参数值设置为 no。当集群移至兼容模式时，作为修改操作的一部分，cluster-enabled 参数值将更新为 `yes`。
> 有关更多信息，请参阅使用参数组配置引擎参数。

1. 准备 – 创建一个测试 CME 集群，并确保您的堆栈已准备好使用它。ElastiCache for Redis 无法验证您是否准备就绪。有关更多信息，请参阅创建集群。

2. 将现有 CMD 集群配置修改为“与集群模式兼容” – 在此模式下，将部署单个分片，ElastiCache for Redis 既可以用作单个节点，也可以用作单个分片集群。兼容模式意味着客户端应用程序可以使用任一协议与集群通信。在此模式下，必须重新配置应用程序才能开始使用 Redis 集群协议和配置端点。要将 Redis 集群模式更改为“与集群模式兼容”，请执行以下步骤：

   > **Note**
   >
   > 在兼容模式下，不允许对集群执行其他修改操作，例如扩缩和引擎版本。此外，在 ModifyReplicationGroup 请求中定义集群模式参数时，无法修改参数（不包括 `cacheParameterGroupName`）。

   a. 使用 AWS Management Console时，请参阅修改复制组并将集群模式设置为兼容

   b. 使用 API，请参阅 ModifyReplicationGroup 并将 `ClusterMode` 参数更新为 `compatible`。

   c. 使用 AWS CLI，请参阅 modify-replication-group 并将 `cluster-mode` 参数更新为 `compatible`。

   将 Redis 集群模式更改为“与集群模式兼容”后，DescribeReplicationGroups API 将返回 ElastiCache for Redis 集群配置端点。集群配置端点是应用程序可以用来连接到集群的单个端点。有关更多信息，请参阅查找连接端点。

3. 将集群配置修改为“已启用集群模式” - 在将集群模式设置为“与集群模式兼容”后，第二步是将集群配置修改为“已启用集群模式”。在此模式下，单个分片正在运行，客户现在可以扩缩其集群或修改其他集群配置。

   要将集群模式更改为已启用，请执行以下步骤：

   开始之前，请确保您的 Redis 客户端已迁移到使用集群协议，并且集群的配置端点未在使用中。

   a. 使用 AWS Management Console时，请参阅修改复制组并将集群模式设置为已启用。

   b. 使用 API，请参阅 ModifyReplicationGroup 并将 `ClusterMode` 参数更新为 `enabled`。

   c. 使用 AWS CLI，请参阅 modify-replication-group 并将 `cluster-mode` 参数更新为 `enabled`。

   在将集群模式更改为已启用后，端点将按照 Redis 集群规范进行配置。DescribeReplicationGroups API 将返回集群模式参数（值为 `enabled`）和现在可供应用程序用于连接到集群的集群端点。

请注意，一旦集群模式更改为已启用，集群端点就会发生变化。请确保使用新端点更新您的应用程序。

您也可以选择从“与集群模式兼容”恢复为“已禁用集群模式”（CMD）并保留原始配置。

将集群配置从“与集群模式兼容”修改为“已禁用集群模式”

1. 使用 AWS Management Console时，请参阅修改复制组并将集群模式设置为已禁用
2. 使用 API，请参阅 ModifyReplicationGroup 并将 `ClusterMode` 参数更新为 `disabled`。
3. 使用 AWS CLI，请参阅 modify-replication-group 并将 `cluster-mode` 参数更新为 `disabled`。

将集群模式更改为已禁用后，DescribeReplicationGroups API 会将集群模式参数返回为 `disabled`。

使用全球数据存储跨 AWS 区域复制

> **Note**
>
> 全局数据存储目前仅适用于自行设计的集群。

通过使用适用于 Redis 的全球数据存储功能，您可以跨 AWS 区域进行完全托管、快速、可靠和安全的复制。使用此功能，您可以为 Redis 创建跨区域只读副本集群，以实现跨区域的低延迟读取和灾难恢复。 ElastiCache AWS

在以下部分中，您可以找到有关如何使用全局数据存储的说明。

主题

- 概述
- 先决条件和限制
- 使用全局数据存储（控制台）
- 使用全局数据存储 (CLI)

概述

每个全局数据存储 是一个或多个相互复制的集群的集合。

全局数据存储包含以下项：

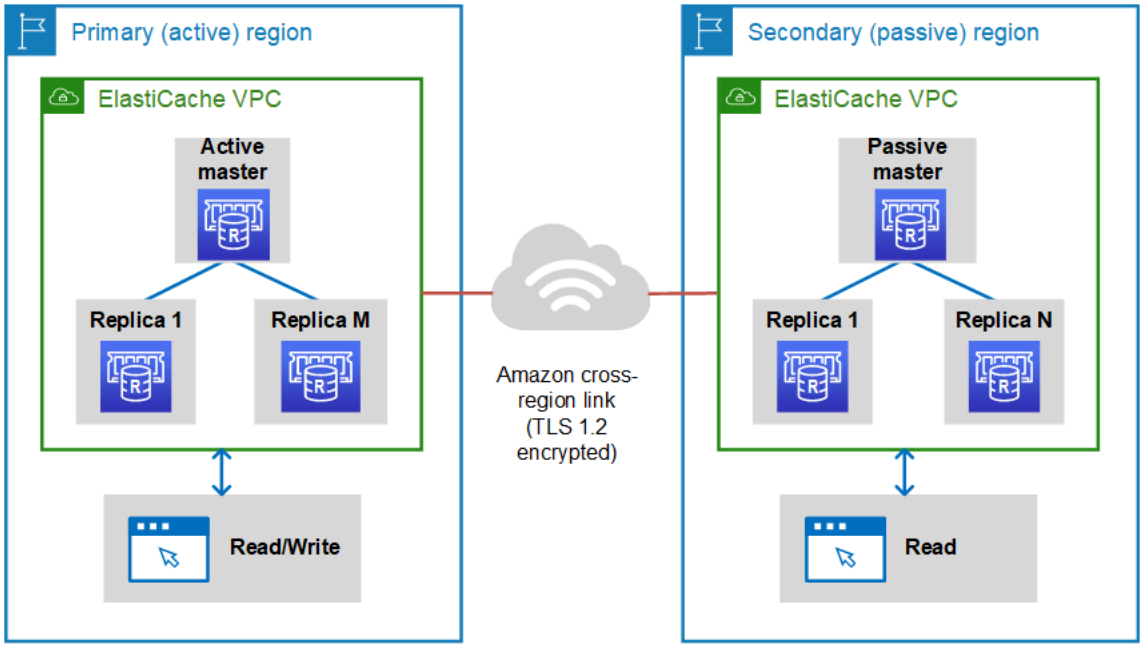
- 主（主动）集群 – 主集群接受复制到全局数据存储中的所有集群的写入。主集群也接受读取请求。
- 辅助（被动）集群 – 辅助集群仅接受读取请求并从主集群复制数据更新。辅助群集必须与主群集位于不同的 AWS 区域。

当您在中创建全局数据存储时 ElastiCache， ElastiCache for Redis 会自动将您的数据从主集群复制到辅助集群。您可以选择应复制 Redis 数据的 AWS 区域，然后在该 AWS 区域中创建辅助集群。ElastiCache 然后设置并管理两个集群之间数据的自动、异步复制。

为 Redis 使用全局数据存储具有以下优势：

- 地理本地性能-通过在其他 AWS 区域设置远程副本集群并在它们之间同步数据，可以减少该 AWS 区域的数据访问延迟。全球数据存储可以通过提供跨区域的低延迟地理本地读取来帮助提高应用程序的响应能力。 AWS
- 灾难恢复 – 如果全局数据存储中的主集群出现性能降级，则可以将辅助集群提升为新的主集群。您可以通过连接到任何包含辅助群集的 AWS 区域来实现此目的。

下图显示了全局数据存储的工作方式。



先决条件和限制

开始使用全局数据存储之前，请注意以下事项：

- 以下 AWS 区域支持全球数据存储：亚太地区（首尔、东京、新加坡、悉尼、孟买和大阪）、欧洲（法兰克福、巴黎、伦敦、爱尔兰和斯德哥尔摩）、美国东部（弗吉尼亚北部和俄亥俄州）、美国西部（加利福尼亚北部和俄勒冈州）、南美洲（圣保罗）、 AWS GovCloud （美国西部和美国东部）、加拿大（中部）地区、中国（北京和宁夏）
- 全局数据存储中的所有集群（主集群和辅助集群）应具有相同数量的主节点、节点类型、引擎版本和分区数（如果启用了集群模式）。全局数据存储中的每个集群可以有不同数量的读取副本，以适应该集群本地的读取流量。

  如果您计划使用现有的单节点集群，则必须启用复制。
- 早于 m5 或 r5 的实例不支持全局数据存储。
- 您可以为主集群设置从一个 AWS 区域到最多两个其他 AWS 区域的辅助集群的复制。

> **Note**
>
> 中国（北京）区域和中国（宁夏）区域除外，只能在这两个区域之间进行复制。

- 您只能在 VPC 集群中使用全局数据存储。有关更多信息，请参阅 访问 Amazon VPC 中 ElastiCache 缓存的访问模式。使用 EC2-Classic 时，不支持全局数据存储。有关更多信息，请参阅 Amazon EC2 用户指南中的 EC 2-Class ic。

> **Note**
>
> 目前，无法在 将 Local Zones 与 ElastiCache 结合使用 中使用全局数据存储。

- ElastiCache 不支持从一个 AWS 区域到另一个区域的自动故障转移。如果需要，您可以手动提升辅助集群。有关示例，请参阅将辅助集群提升为主集群。
- 要从现有数据引导，请使用现有集群作为主集群来创建全局数据存储。我们不支持将现有集群添加为辅助集群。将集群添加为辅助集群的过程会擦除数据，这可能会导致数据丢失。
- 当您修改属于全局数据存储的某个集群的本地参数组时，参数更新应用于所有集群。
- 您可以垂直（向上扩展和向下扩展）和水平扩展（向内扩展和向外扩展）区域集群。您可以通过修改全局数据存储来扩展集群。然后，全局数据存储中的所有区域集群都会在不中断的情况下扩展。有关更多信息，请参阅 针对 Redis ElastiCache 进行扩展 。
- 全局数据存储支持静态加密、传输中加密和 Redis AUTH。
- 全球数据存储库不支持互联网协议版本 6 (IPv6)。
- 全局数据存储支持 AWS KMS 密钥。有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的 AWS 密钥管理服务概念。

> **Note**
>
> 全局数据存储支持发布/订阅消息收发，且具有以下规定：
>
> - 如已禁用集群模式，则完全支持发布/订阅。在主 AWS 区域的主集群上发布的事件会传播到辅助 AWS 区域。
> - 如已启用集群模式，则以下情况适用：
>   - 对于不在密钥空间中的已发布事件，只有同一 AWS 地区的订阅者才能收到事件。
>   - 对于已发布的密钥空间事件，所有 AWS 地区的订阅者都会收到事件。

使用全局数据存储（控制台）

要使用控制台创建全局数据存储，请遵循以下两步过程：

1. 通过使用现有集群或创建新集群来创建主集群。引擎必须是 Redis 5.0.6 或更高版本。
2. 再次使用 Redis 5.0.6 或更高版本引擎，在不同 AWS 区域最多添加两个辅助集群。

以下过程将指导您如何使用 for Redis 控制台为 Redis 创建全局数据存储以及如何执行其他操作。ElastiCache

主题

- 使用现有集群创建全局数据存储
- 使用新主集群创建新的全局数据存储
- 查看全局数据存储详细信息
- 将区域添加到全局数据存储
- 修改全局数据存储
- 将辅助集群提升为主集群
- 从全局数据存储中删除区域
- 删除全局数据存储

使用现有集群创建全局数据存储

在这种情况下，使用现有集群作为新全局数据存储的主集群。然后，您可以在单独的 AWS 区域中创建辅助只读集群。此辅助集群接收来自主集群的自动和异步更新。

> **Important**
>
> 现有集群必须使用 Redis 5.0.6 引擎或更高版本。

使用现有集群创建全局数据存储

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格上，选择全局数据存储，然后选择创建全局数据存储。
3. 在主集群设置页面上，执行以下操作：
   - 在全局数据存储信息字段中，输入新的全局数据存储的名称。
   - （可选）输入 Description（描述）值。
4. 在区域群集下，选择使用现有区域群集。
5. 在现有集群下，选择要使用的现有集群。
6. 使以下选项保持不变。它们会预先填入以匹配主集群配置，您无法对其进行更改。
   - 引擎版本
   - 节点类型
   - 参数组

     > **Note**
     >
     > ElastiCache 根据提供的参数组的值自动生成新的参数组，并将新的参数组应用于集群。使用这个新参数组修改全局数据存储上的参数。每个自动生成的参数组与一个（且只有一个）集群关联，因此只有一个全局数据存储。

   - 分片数量
   - 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**
     >
     > 您可以通过选择客户管理的 KMS 密钥并选择密 AWS 钥来提供不同的加密密钥。有关更多信息，请参阅使用客户托管的 AWS KMS 密钥。

- Encryption in-transit（传输中加密）– 对传输中数据启用加密。有关更多信息，请参阅传输中加密。对于 Redis 6.0 及以上的引擎版本，如果启用了传输中加密，则系统会提示您指定以下 Access Control（访问控制）选项中的一个：
  - No Access Control（无访问控制）– 此选项为默认设置。表示无任何限制。
  - User Group Access Control List（用户组访问控制列表）– 选择具有对可用操作定义有用户集和权限的用户组。有关更多信息，请参阅 使用控制台和 CLI 管理用户组。
  - Redis AUTH Default User（Redis AUTH 默认用户）– Redis 服务器的身份验证机制。有关更多信息，请参阅 Redis AUTH。

7. （可选）根据需要更新剩余的辅助集群设置。这些设置会预先填入与主集群相同的值，但您可以对其进行更新，以满足该集群的特定要求。

   - 端口
   - 副本数量
   - 子网组
   - 首选可用区
   - 安全组
   - 客户管理（AWS KMS 密钥）
   - Redis AUTH 令牌
   - 启用自动备份
   - 备份保留期
   - 备份时段
   - 维护时段
   - SNS 主题通知

8. 选择创建。这样做会将全局数据存储的状态设置为 Creating（正在创建）。在主集群与全局数据存储关联且辅助集群处于 Associating（正在关联）状态后，状态将转换为 Modifying（正在修改）。

   主集群和辅助集群与全局数据存储关联后，状态会更改为 Available（可用）。此时，您有一个接受读取和写入的主集群和接受从主集群复制的读取数据的辅助集群。

   Redis 页面更新为指示集群是否属于全局数据存储，包括：

   - Global Datastore（全局数据存储）– 集群所属的全局数据存储的名称。
   - Global Datastore Role（全局数据存储角色）– 主集群或辅助集群的角色。

您最多可以在不同的 AWS 区域中再添加一个辅助群集。有关更多信息，请参阅 将区域添加到全局数据存储。

使用新主集群创建新的全局数据存储

如果选择使用新集群创建全局数据存储，请按照以下过程操作。

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格上，选择全局数据存储，然后选择创建全局数据存储。
3. 在 Primary cluster settings（主集群设置）中，执行以下操作：
   a. 对于 Cluster mode（集群模式），选择 Enabled（已启用）或 Disabled（已禁用）。
   b. 要获取全局数据存储信息，请为 “名称” 输入一个值。 ElastiCache 使用后缀为全局数据存储生成唯一的名称。您可以使用在此处指定的后缀搜索全局数据存储。
   c. （可选）为 Global Datastore Description（全局数据存储描述）输入一个值。
4. 在 Regional cluster（区域集群）中：
   a. 对于区域，选择一个可用 AWS 区域。
   b. 选择 Create new regional cluster（创建新的区域集群）或 Use existing regional cluster（使用现有的区域集群）
   c. 如果选择 Create new regional cluster（创建新的区域集群），在 Cluster info（集群信息）下，输入集群的名称和可选描述。
   d. 在 Location（位置）下，我们建议您接受 Multi-AZ（多可用区）和 Auto-failover（自动失效转移）的默认设置。
5. 在 Cluster settings（集群设置）下
   a. 对于 Engine version（引擎版本），选择 5.0.6 或更高版本。
   b. 对于 Port（端口），使用默认端口 6379。如果您出于某个原因需要使用其他端口，请输入相应的端口号。
   c. 对于参数组，选择一个参数组或创建一个新参数组。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 和创建参数组。

> **Note**
>
> 当您选择要设置引擎配置值的参数组时，该参数组将应用于全局数据存储中的所有集群。在 Parameter Groups（参数组）页面上，是/否 Global（全局）属性指示参数组是否属于全局数据存储。

d. 对于 Node type（节点类型），请选择向下箭头（ ）。
在 Change node type（更改节点类型）对话框中，为所需节点类型选择 Instance family（实例系列）值。接着选择要用于此集群的节点类型，然后选择保存。

有关更多信息，请参阅 选择节点大小。

如果您选择 r6gd 节点类型，则系统会自动启用数据分层。有关更多信息，请参阅 数据分层。

e. 如果您要创建 Redis（已禁用集群模式）集群：

对于 Number of replicas（副本数量），为此集群选择所需的副本数。

f. 如果您要创建 Redis（已启用集群模式）集群：

i. 对于 Number of shards（分片数），选择要用于此 Redis （已启用集群模式）集群的分片（分区/节点组）数。

对于某些版本的 Redis（已启用集群模式），您可以动态更改集群中的分片数量：

- Redis 3.2.10 及更高版本– 如果您的集群运行 Redis 3.2.10 或更高版本，则可以动态更改集群中的分片数量。有关更多信息，请参阅 扩展 Redis（启用集群模式）集群。
- 其他 Redis 版本 – 如果您的集群正在运行 3.2.10 版之前的 Redis 版本，则还有另一种方法。在这种情况下，要更改集群中的分片数量，请使用新分片数量创建一个新集群。有关更多信息，请参阅 从备份还原到新缓存。

ii. 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。

Redis（已启用集群模式）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。

- 每个分片的只读副本数固定，无法更改。如果您需要增加或减少各分片（API/CLI：节点组）的副本数，您必须使用新的副本数量创建一个新集群。有关更多信息，请参阅使用外部创建的备份为新的自行设计的集群制作种子。

6. 对于子网组设置，请选择要应用于此集群的子网。 ElastiCache 提供了默认 IPv4 子网组，也可以选择创建新子网组。对于 IPv6，您需要创建一个带有 IPv6 CIDR 块的子网组。如果您选择双堆栈，则必须选择发现 IP 类型，即 IPv6 或 IPv4。

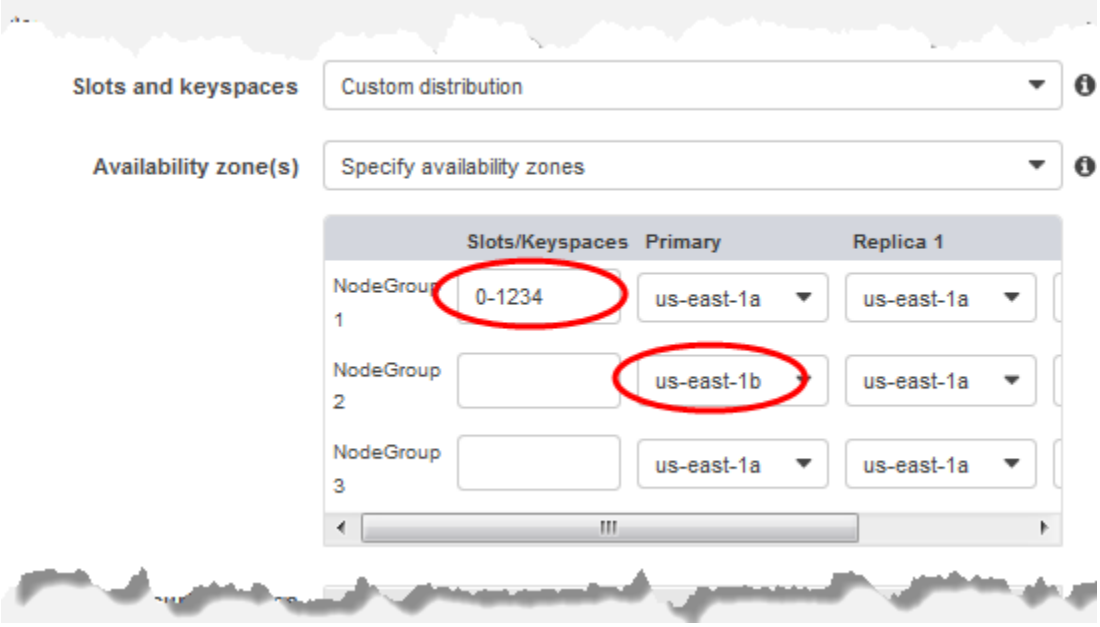
   有关更多信息，请参阅在 VPC 中创建子网。

7. 对于 Availability zone placements（可用区位置），您有两种选择：

   - 无偏好 — ElastiCache 选择可用区。
   - Specify availability zones（指定可用区）– 您为各集群指定可用区。

     如果您选择指定可用区，则需从列表中为各分片中的每个集群选择可用区。

   有关更多信息，请参阅 选择区域和可用区。



   指定键空间和可用区

8. 选择 Next（下一步）

9. 在 Advanced Redis settings（高级 Redis 设置）下：

   - 对于 Security（安全）：

i. 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**
>
> 您可以选择提供不同的加密密钥，方法是选择 “客户托管 AWS KMS 密钥” 并选择密钥。有关更多信息，请参阅使用 AWS KMS 客户自主管理型密钥。

- Encryption in-transit（传输中加密）– 对传输中数据启用加密。有关更多信息，请参阅传输中加密。对于 Redis 6.0 及以上的引擎版本，如果启用了传输中加密，则系统会提示您指定以下 Access Control（访问控制）选项中的一个：
  - No Access Control（无访问控制）– 此选项为默认设置。这表示对用户访问集群的权限没有任何限制。
  - User Group Access Control List（用户组访问控制列表）– 选择具有集群访问权限的已定义用户组。有关更多信息，请参阅 使用控制台和 CLI 管理用户组。
  - Redis AUTH Default User（Redis AUTH 默认用户）– Redis 服务器的身份验证机制。有关更多信息，请参阅 Redis AUTH。
- Redis AUTH – Redis 服务器的身份验证机制。有关更多信息，请参阅 Redis AUTH。

> **Note**
>
> 对于 Redis 3.2.6 以上的版本（版本 3.2.10 除外），只能选择 Redis AUTH。

ii. 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。

10. 如果需要定期计划自动备份，请选择启用自动备份，然后输入每个自动备份在被自动删除前保留的天数。如果您不希望定期计划自动备份，请清除 Enable automatic backups 复选框。不论是哪种情况，您始终可以选择创建手动备份。

    有关 Redis 备份和还原的更多信息，请参阅快照和还原。

11. （可选）指定维护时段。维护时段是指每周为集群 ElastiCache 安排系统维护的时间，通常为一小时。您可以 ElastiCache 允许选择维护时段的日期和时间（无偏好），也可以自己选择日期、时间

和持续时间（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。

12. （可选）对于 Logs（日志）：

    - 在 Log format（日志格式）下，选择 Text（文本）或 JSON。
    - 在目标类型下，选择CloudWatch 日志或 Kinesis Fire hose。
    - 在 “日志目标” 下，选择 “新建” 并输入您的 CloudWatch 日志组名称或 Firehose 直播名称，或者选择 “选择现有”，然后选择您的 CloudWatch 日志日志组名称或 Firehose 直播名称，

13. 对于标签，为了帮助您管理集群和其他 ElastiCache资源，您可以以标签的形式为每个资源分配自己的元数据。有关更多信息，请参阅 标记 ElastiCache 资源。
14. 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，选择 Next（下一步）。
15. 在前面步骤中配置集群后，您现在可以配置辅助集群详细信息。
16. 在区域群集下，选择集群所在的 AWS 区域。
17. 在 Cluster info（集群信息）下，输入集群的名称和可选描述。
18. 以下选项已预先填入以匹配主集群配置，且无法更改：

    - 位置
    - 引擎版本
    - 实例类型
    - 节点类型
    - 分片数量
    - 参数组

    > **Note**
    >
    > ElastiCache 根据提供的参数组的值自动生成新的参数组，并将新的参数组应用于集群。使用这个新参数组修改全局数据存储上的参数。每个自动生成的参数组与一个（且只有一个）集群关联，因此只有一个全局数据存储。

    - 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**
>
> 您可以通过选择客户管理的 KMS 密钥并选择密 AWS 钥来提供不同的加密密钥。有关更多信息，请参阅使用客户托管的 AWS KMS 密钥。

- Encryption in-transit（传输中加密）– 对传输中数据启用加密。有关更多信息，请参阅传输中加密。对于 Redis 引擎版本 6.4 及更高版本时，如果启用了传输中加密，则系统会提示您指定以下 Access Control（访问控制）选项之一：
  - No Access Control（无访问控制）– 此选项为默认设置。这表示对用户访问集群的权限没有任何限制。
  - User Group Access Control List（用户组访问控制列表）– 选择具有集群访问权限的已定义用户组。有关更多信息，请参阅 使用控制台和 CLI 管理用户组。
  - Redis AUTH Default User（Redis AUTH 默认用户）– Redis 服务器的身份验证机制。有关更多信息，请参阅 Redis AUTH。

> **Note**
>
> 对于介于 4.0.2 和 6.0.4 之间的 Redis 版本，当第一次支持传输中加密时，Redis AUTH 是唯一的选择。

剩下的辅助集群设置会预先填入与主集群相同的值，但您可以更新以下值来满足对该集群的特定要求：

- 端口
- 副本数量
- 子网组
- 首选可用区
- 安全组
- 客户管理（AWS KMS 密钥）
- Redis AUTH 令牌
- 启用自动备份
- 备份保留期
- 备份时段

- 维护时段
- SNS 主题通知

19. 选择创建。这会将全局数据存储的状态设置为 Creating（正在创建）。主集群和辅助集群与全局数据存储关联后，状态会更改为 Available（可用）。您有一个接受读取和写入的主集群和接受从主集群复制的读取数据的辅助集群。

    Redis 页面也会更新为指示集群是否属于全局数据存储，包括以下项：

    - Global Datastore（全局数据存储）– 集群所属的全局数据存储的名称。
    - Global Datastore Role（全局数据存储角色）– 主集群或辅助集群的角色。

您最多可以在不同的 AWS 区域中再添加一个辅助群集。有关更多信息，请参阅 将区域添加到全局数据存储。

查看全局数据存储详细信息

您可以查看现有全局数据存储的详细信息，也可以在全局数据存储页面上对其进行修改。

查看全局数据存储详细信息

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格上，选择全局数据存储，然后选择可用的全局数据存储。

然后，您可以检查以下全局数据存储属性：

- 全局数据存储名称：全局数据存储的名称
- 描述：全局数据存储的描述
- 状态：选项包括：
  - Creating
  - Modifying
  - 可用
  - Deleting（正在删除）
  - 仅主集群 - 此状态表示全局数据存储仅包含主集群。所有辅助集群均已删除，或者未成功创建。
- 集群模式：启用或禁用
- Redis 引擎版本：运行全局数据存储的 Redis 引擎版本

- 实例节点类型：全局数据存储所用的节点类型
- 静态加密：启用或禁用
- 传输中加密：启用或禁用
- Redis AUTH：启用或禁用

您可以对全局数据存储进行以下更改：

- 将区域添加到全局数据存储
- 从全局数据存储中删除区域
- 将辅助集群提升为主集群
- 修改全局数据存储

“Global Datastore（全局数据存储）”页面还列出组成全局数据存储的各个集群以及每个集群的以下属性：

- 区域-存储集群的 AWS 区域
- Role（角色）- 主集群或辅助集群
- Cluster name（集群名称）- 集群的名称
- Status（状态）- 选项包括：
  - Associating（正在关联）- 集群正在关联到全局数据存储
  - Associated（已关联）- 集群已与全局数据存储关联
  - Disassociating（正在解除关联）- 使用全局数据存储名称从全局数据存储中删除辅助集群的过程。此后，辅助群集不再接收来自主群集的更新，但它仍作为该 AWS 区域的独立群集保留。
  - Disassociated（已取消关联）– 辅助集群已从全局数据存储中删除，现为其 AWS 区域中的独立集群。
- 全局数据存储副本延迟-显示全球数据存储中每个辅助 AWS 区域的一个值。此为辅助区域的主节点与主区域的主节点之间的滞后。对于已启用集群模式的 Redis，滞后表示分片之间的最大延迟（以秒计）。

将区域添加到全局数据存储

您最多可以向现有的全球数据存储中添加一个额外 AWS 区域。在这种情况下，您将在单独的 AWS 区域中创建一个只读集群，该集群接收来自主集群的自动和异步更新。

向全球数据存储中添加 AWS 区域

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格上，选择全局数据存储，然后选择现有的全局数据存储。
3. 选择添加区域群集，然后选择辅助群集要驻留的 AWS 区域。
4. 在集群信息下，为集群的名称输入一个值，也可以为集群的描述输入一个值。
5. 使以下选项保持不变。它们会预先填入以匹配主集群配置，您无法对其进行更改。

   - 引擎版本
   - 实例类型
   - 节点类型
   - 分片数量
   - 参数组

   > **Note**
   >
   > ElastiCache 根据提供的参数组的值自动生成新的参数组，并将新的参数组应用于集群。使用这个新参数组修改全局数据存储上的参数。每个自动生成的参数组与一个（且只有一个）集群关联，因此只有一个全局数据存储。

   - 静态加密

   > **Note**
   >
   > 您可以通过选择客户管理的 KMS 密钥并选择密 AWS 钥来提供不同的加密密钥。

   - 传输中加密
   - Redis AUTH
6. （可选）更新剩余的辅助集群设置。这些设置会预先填入与主集群相同的值，但您可以对其进行更新，以满足该集群的特定要求：

   - 端口
   - 副本数量
   - 子网组
   - 首选可用区

- 安全组
- 客户托管 AWS KMS 密钥）
- Redis AUTH 令牌
- 启用自动备份
- 备份保留期
- 备份时段
- 维护时段
- SNS 主题通知

7. 选择添加。

修改全局数据存储

您可以修改区域集群的属性。全局数据存储上只能进行一个修改操作，但将辅助集群提升为主集群除外。有关更多信息，请参阅 将辅助集群提升为主集群。

修改全局数据存储

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格上，选择全局数据存储，然后在全局数据存储名称中选择一个全局数据存储。
3. 选择 Modify（修改）并在以下选项中进行选择：

   - Modify description（修改描述）– 更新全局数据存储的描述
   - Modify engine version（修改引擎版本）– 只有 Redis 引擎版本 5.0.6 或更高版本才可用。
   - Modify node type（修改节点类型）– 纵向（扩展和缩减）和横向（扩展和缩减）扩展区域集群。选项包括 R5 和 M5 节点系列。有关节点类型的更多信息，请参阅受支持的节点类型。
   - Modify Automatic Failover（修改自动故障转移）– 启用或禁用自动故障转移。当您启用故障转移并且区域群集中的主节点意外关闭时， ElastiCache 会故障转移到其中一个区域副本。有关更多信息，请参阅自动故障转移。

   对于启用集群模式的 Redis 集群：

   - Add shards（添加分片）– 输入要添加的分片数量，并有选择地指定一个或多个可用区。
   - 删除分片-选择每个 AWS 区域中要删除的分片。

- Rebalance shards（重新平衡分片）– 重新平衡插槽分配，以确保在集群中的现有分片之间均匀分配。

要修改全局数据存储的参数，请修改该全局数据存储的任何成员集群的参数组。 ElastiCache 将此更改自动应用于该全局数据存储中的所有集群。要修改该集群的参数组，请使用 Redis 控制台或 ModifyCacheClusterAPI 操作。有关更多信息，请参阅 修改参数组。修改全局数据存储中包含的任何集群的参数组时，该参数组应用于该全局数据存储中的所有集群。

要重置整个参数组或特定参数，请使用 ResetCacheParameterGroupAPI 操作。

将辅助集群提升为主集群

如果主群集或 AWS 区域不可用或遇到性能问题，则可以将辅助群集提升为主群集。随时可进行提升，即使正在进行其他修改时也可以。您还可以并行进行多个提升，并且全局数据存储最终会解析为一个主集群。如果您同时提升多个辅助集群， ElastiCache for Redis 并不能保证哪一个最终会解析为主集群。

将辅助集群提升为主集群

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格上，选择全局数据存储。
3. 选择全局数据存储名称以查看详细信息。
4. 选择 Secondary（辅助）集群。
5. 选择 Promote to primary（提升为主集群）。

   然后，系统会提示您确认您的决定，并显示以下警告： `Promoting a region to primary will make the cluster in this region as read/writable. Are you sure you want to promote the secondary cluster to primary?`

   `The current primary cluster in primary region will become secondary and will stop accepting writes after this operation completes. Please ensure you update your application stack to direct traffic to the new primary region.`
6. 如果您希望继续提升，请选择 Confirm（确认），如果您不希望继续提升，请选择 Cancel（取消）。

如果选择确认，则全局数据存储会转换为 Modifying（正在修改）状态，并且在提升完成之前不可用。

从全局数据存储中删除区域

您可以使用以下步骤从全球数据存储中移除 AWS 区域。

从全球数据存储中移除 AWS 区域

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格上，选择全局数据存储。
3. 选择一个全局数据存储。
4. 选择要删除的 Region（区域）。
5. 选择 Remove region（删除区域）。

   > **Note**
   >
   > 此选项仅适用于辅助集群。

   然后，系统会提示您确认您的决定，并显示以下警告： `Removing the region will remove your only available cross region replica for the primary cluster. Your primary cluster will no longer be set up for disaster recovery and improved read latency in remote region. Are you sure you want to remove the selected region from the global datastore?`
6. 如果您希望继续提升，请选择 Confirm（确认），如果您不希望继续提升，请选择 Cancel（取消）。

如果选择确认，则该 AWS 区域将被删除，辅助群集将不再收到复制更新。

删除全局数据存储

要删除全局数据存储，请先删除所有辅助集群。有关更多信息，请参阅 从全局数据存储中删除区域。这样做会使全局数据存储处于 primary-only（仅主集群）状态。

删除全局数据存储

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。

2. 在导航窗格上，选择全局数据存储。
3. 在 Global Datastore Name（全局数据存储名称）下，选择要删除的全局数据存储，然后选择 Delete（删除）。

   然后，系统会提示您确认您的决定，并显示以下警告：`Are you sure you want to delete this Global Datastore?`
4. 选择 Delete（删除）。

全局数据存储将转换为 Deleting（正在删除）状态。

使用全局数据存储 (CLI)

您可以使用 AWS Command Line Interface (AWS CLI) 从命令行管理多个 AWS 服务并通过脚本自动执行这些服务。您可以使用 AWS CLI 执行临时（一次性）操作。

下载和配置 AWS CLI

AWS CLI 在 Windows、macOS 或 Linux 上运行。按照以下步骤下载和并对其进行配置。

下载、安装和配置 CLI

1. 在 AWS Command Line Interface 网页上下载 AWS CLI。
2. 按照 AWS Command Line Interface 用户指南中的说明安装 AWS CLI 和配置 AWS CLI。

将 AWS CLI 与全局数据存储结合使用

使用以下 CLI 操作来处理全局数据存储：

- create-global-replication-group

```
aws elasticache create-global-replication-group \
   --global-replication-group-id-suffix my global datastore  \
   --primary-replication-group-id sample-repl-group  \
   --global-replication-group-description an optional description of the global
 datastore
```

Amazon ElastiCache 会在全局数据存储 ID 被创建时自动对其应用前缀。每个 AWS 区域有自己的前缀。例如，在美国西部（北加利福尼亚）区域创建的全局数据存储 ID 以“virxk”和您提供的后缀名称开头。后缀与自动生成的前缀相结合，保证了全局数据存储名称在多个区域中的唯一性。

下表列出了每个 AWS 区域及其全局数据存储 ID 前缀。

区域名称/区域	Prefix
美国东部（俄亥俄州）区域 us-east-2	fpkhr
美国东部（弗吉尼亚州北部）区域 us-east-1	ldgnf
美国西部（加利福尼亚北部）区域 us-west-1	virxk
美国西部（俄勒冈州）区域 us-west-2	sgaui
加拿大（中部）区域 ca-central-1	bxodz
亚太地区（孟买）区域 ap-south-1	erpgt
亚太地区（东京）区域 ap-northeast-1	quwsw
亚太地区（首尔）区域 ap-northeast-2	lfqnh
亚太地区（大阪）区域 ap-northeast-3	nlapn

区域名称/区域	Prefix
亚太地区（新加坡）区域 ap-southeast-1	vlqxn
亚太地区（悉尼）区域 ap-southeast-2	vbgxd
欧洲地区（法兰克福）区域 eu-central-1	iudkw
Europe (Ireland) Region eu-west-1	gxeiz
欧洲地区（伦敦）区域 eu-west-2	okuqm
欧洲（巴黎）区域 eu-west-3	fgjhi
南美洲（圣保罗）区域 sa-east-1	juxlw
中国（北京）区域 cn-north-1	emvgo
中国（宁夏）区域 cn-northwest-1	ckbem
亚太地区（香港）区域 ap-east-1	knjmp

区域名称/区域	Prefix
AWS GovCloud（美国西部） us-gov-west-1	sgwui

- create-replication-group – 使用此操作可通过向 --global-replication-group-id 参数提供全局数据存储的名称来创建全局数据存储的辅助集群。

```
aws elasticache create-replication-group \
  --replication-group-id secondary replication group name \
  --replication-group-description “Replication group description" \
  --global-replication-group-id global datastore name
```

当调用此操作并传入 --global-replication-group-id 值时，ElastiCache for Redis 会从全局复制组的主复制组中推断下列参数的值。请勿传入这些参数的值：

"PrimaryClusterId",

"AutomaticFailoverEnabled",

"NumNodeGroups",

"CacheParameterGroupName",

"CacheNodeType",

"Engine",

"EngineVersion",

"CacheSecurityGroupNames",

"EnableTransitEncryption",

"AtRestEncryptionEnabled",

"SnapshotArns",

"SnapshotName"

- describe-global-replication-groups

```
aws elasticache describe-global-replication-groups \
   --global-replication-group-id my global datastore  \
   --show-member-info an optional parameter that returns a list of the primary and
 secondary clusters that make up the global datastore
```

- modify-global-replication-group

```
aws elasticache modify-global-replication-group \
   --global-replication-group-id my global datastore  \
   --automatic-failover-enabled \
   --cache-node-type node type \
   --cache-parameter-group-name parameter group name \
   --engine-version engine version \
   --–apply-immediately \
   --global-replication-group-description description
```

- delete-global-replication-group

```
aws elasticache delete-global-replication-group \
   --global-replication-group-id my global datastore  \
   --retain-primary-replication-group defaults to true
```

- disassociate-global-replication-group

```
aws elasticache disassociate-global-replication-group \
   --global-replication-group-id my global datastore  \
   --replication-group-id my secondary cluster  \
   --replication-group-region the AWS Region in which the secondary cluster resides
```

- failover-global-replication-group

```
aws elasticache failover-replication-group \
   --global-replication-group-id my global datastore \
   --primary-region The AWS Region of the primary cluster \
   --primary-replication-group-id  The name of the global datastore, including the
 suffix.
```

- increase-node-groups-in-global-replication-group

```
aws elasticache increase-node-groups-in-global-replication-group \
   --apply-immediately yes \
```

```
   --global-replication-group-id global-replication-group-name \
   --node-group-count 3
```

- decrease-node-groups-in-global-replication-group

```
aws elasticache decrease-node-groups-in-global-replication-group \
   --apply-immediately yes \
   --global-replication-group-id global-replication-group-name \
   --node-group-count 3
```

- rebalance-shards-in-global-replication-group

```
aws elasticache rebalance-shards-in-global-replication-group \
   --apply-immediately yes \
   --global-replication-group-id global-replication-group-name
```

使用帮助列出 ElastiCache for Redis 的所有可用命令。

```
aws elasticache help
```

您还可以使用帮助来描述特定命令并了解有关其用法的详细信息：

```
aws elasticache create-global-replication-group help
```

使用复制组时的高可用性

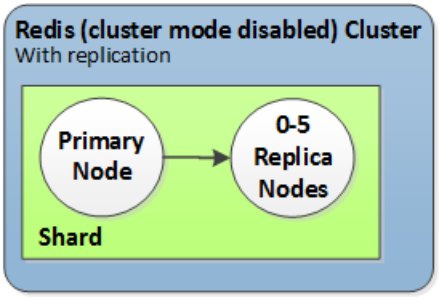
单节点 Amazon ElastiCache Redis 集群是具有有限数据保护服务 (AOF) 的内存中实体。如果您的集群出于任何原因发生故障，您将丢失集群中的所有数据。但是，如果您运行的是 Redis 引擎，则可将 2 到 6 个节点归入一个具有副本的集群，其中，1 到 5 个只读节点包含该组的单个读/写主节点的复制数据。在这种情况下，如果一个节点出于任何原因发生故障，您不会丢失所有数据，因为这些数据已在另外的一个或多个节点中复制。如果发生故障的是读/写主节点，由于复制延迟，某些数据可能会丢失。

如下图所示，复制结构包含在分区中（在 API/CLI 中称为节点组），而分区包含在 Redis 集群中。Redis（已禁用集群模式）集群始终有一个分区。Redis（已启用集群模式）集群最多可以拥有 500 个分区，而且集群的数据跨分区进行分区。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 90 个节点。此集群配置的范围可以从 90 个分片和 0 个副本到 15 个分片和 5 个副本，这是允许的最大副本数。

如果 Redis 引擎版本为 5.0.6 或更高版本，可将每个集群的节点或分片限制提高到最大值 500。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。有关更多信息，请参阅 创建子网组。

对于低于 5.0.6 的版本，每个集群的限制为 250。

若要请求提高限制，请参阅 AWS Service Limits 并选择限制类型 Nodes per cluster per instance type（每个实例类型的每个集群的节点数）。



Redis（已禁用集群模式）集群拥有一个分区和 0 到 5 个副本节点

如果为具有副本的集群启用了多可用区，则当主节点发生故障时，主节点将故障转移到某个只读副本。由于在副本节点上异步更新数据，因此可能会因更新副本节点时存在延迟而导致丢失某些数据。有关更多信息，请参阅 缓解运行 Redis 时发生的故障。

主题

- 了解 Redis 复制
- 复制：Redis（已禁用集群模式）与 Redis（已启用集群模式）对比
- 使用多可用区最大限度地 ElastiCache 缩短 Redis 的停机时间
- 如何实施同步和备份
- 创建 Redis 复制组
- 查看复制组的详细信息
- 查找复制组端点
- 修改复制组
- 删除复制组
- 更改副本数量
- 将 Redis（已禁用集群模式）复制组的只读副本提升为主节点

了解 Redis 复制

Redis 通过两种方式实现复制：

- 各个节点中的所有集群数据包含在单一分片中 – Redis（已禁用集群模式）
- 数据在最多 500 个分片中进行分区 – Redis（已启用集群模式）

复制组中的每个分片都包含一个读/写主节点和最多 5 个只读副本节点。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 90 个节点。此集群配置的范围可以从 90 个分片和 0 个副本到 15 个分片和 5 个副本，这是允许的最大副本数。

如果 Redis 引擎版本为 5.0.6 或更高版本，可将每个集群的节点或分片限制提高到最大值 500。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。有关更多信息，请参阅 创建子网组。

对于低于 5.0.6 的版本，每个集群的限制为 250。

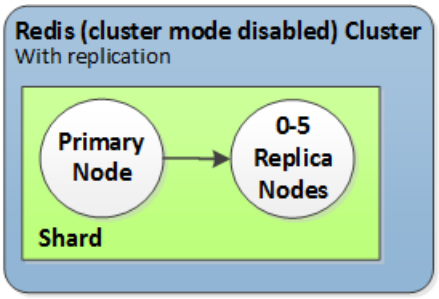
若要请求提高限制，请参阅 AWS Service Limits 并选择限制类型 Nodes per cluster per instance type（每个实例类型的每个集群的节点数）。

主题

- Redis（已禁用集群模式）
- Redis（已启用集群模式）

Redis（已禁用集群模式）

Redis（已禁用集群模式）集群拥有一个分片，其中包含一个 Redis 节点集合；一个读/写主节点和最多 5 个辅助只读副本节点。每个只读副本保留一个集群主节点数据的副本。可使用异步复制机制使只读副本与主集群同步。应用程序可以从集群中的任何节点进行读取。应用程序只能对主节点进行写入。只读副本增加了读取吞吐量，并在节点发生故障时防止数据丢失。



具有一个分片和多个副本节点的 Redis（已禁用集群模式）集群

您可以使用带有副本节点的 Redis（已禁用集群模式）集群来扩展您的 Redis 解决方案， ElastiCache 以处理读取密集型应用程序或支持同时从同一集群读取的大量客户端。

Redis（已禁用集群模式）集群中的所有节点必须位于同一区域内。

在向集群添加只读副本时，主集群中的所有数据都会复制到新节点。从此之后，只要向主集群写入数据，更改便会异步传播到所有只读副本。

要增强容错能力并减少写入停机时间，请对具有副本的 Redis（已禁用集群模式）集群启用带自动故障转移功能的多可用区。有关更多信息，请参阅 使用多可用区最大限度地 ElastiCache 缩短 Redis 的停机时间。

您可以更改 Redis（已禁用集群模式）集群中的节点的角色，在这种情况下，主集群会与一个副本交换角色。您可能会为了优化性能执行此操作。例如，对于具有大量写入活动的 Web 应用程序，您可以选择网络延迟最低的节点。有关更多信息，请参阅 将 Redis（已禁用集群模式）复制组的只读副本提升为主节点。

Redis（已启用集群模式）

Redis （已启用集群模式）集群由 1 到 500 个分片（API/CLI：节点组）组成。每个分片有一个主节点和最多 5 个只读副本节点。配置的范围可以从 90 个分片和 0 个副本到 15 个分片和 5 个副本，这是允许的最大副本数。

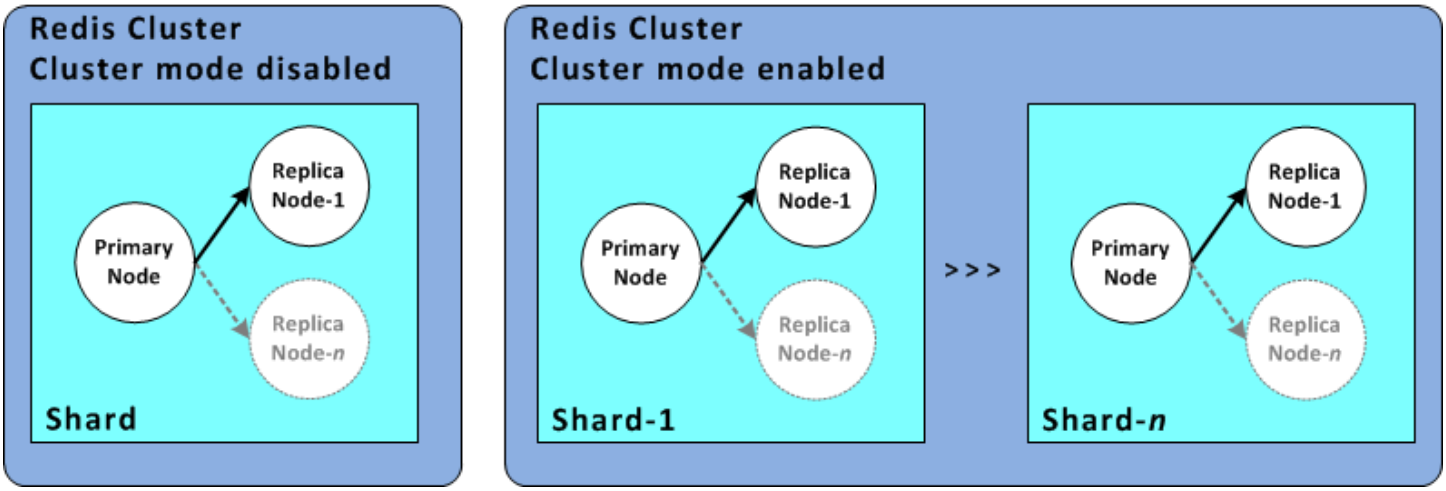
如果 Redis 引擎版本为 5.0.6 或更高版本，可将每个集群的节点或分片限制提高到最大值 500。例如，您可以选择配置一个 500 节点的集群，范围介于 83 个分片（一个主分片和 5 个副本分片）和 500 个分片（一个主分片，无副本分片）之间。确保可提供足够的 IP 地址来满足增长需求。常见的陷阱包括子网组中的子网 CIDR 范围太小，或者子网被其他集群共享和大量使用。有关更多信息，请参阅 创建子网组。

对于低于 5.0.6 的版本，每个集群的限制为 250。

若要请求提高限制，请参阅 AWS Service Limits 并选择限制类型 Nodes per cluster per instance type（每个实例类型的每个集群的节点数）。

分片中的每个只读副本均保留分片主集群中数据的一份副本。可使用异步复制机制使只读副本与主集群同步。应用程序可以从集群中的任何节点进行读取。应用程序只能对主节点进行写入。只读副本可增强读取可扩展性和防止数据丢失。数据在 Redis（启用集群模式）集群中的分片上进行分区。

应用程序使用 Redis（已启用集群模式）集群的配置端点连接集群中的节点。有关更多信息，请参阅查找连接端点。



具有多个分片和副本节点的 Redis（已启用集群模式）集群

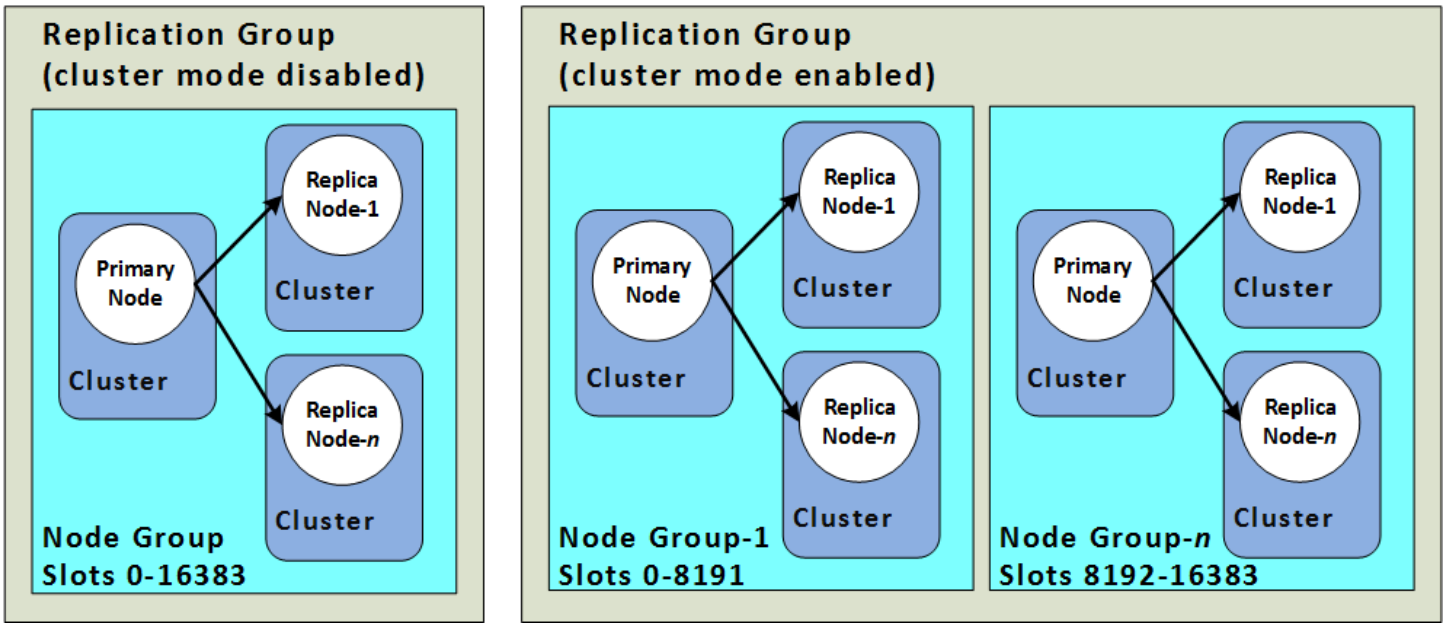
Redis（已启用集群模式）集群中的所有节点必须位于同一区域内。要增强容错能力，您可在该区域内的多个可用区中预配置主副本和只读副本。

目前，Redis（已启用集群模式）中有一些限制。

- 您无法手动将任何副本节点提升为主节点。

复制：Redis（已禁用集群模式）与 Redis（已启用集群模式）对比

从 Redis 版本 3.2 开始，您可以创建两种不同类型的 Redis 集群 (API/CLI：复制组) 之一。Redis（已禁用集群模式）集群始终具有单个分区（API/CLI：节点组），包含最多 5 个只读副本节点。Redis（已启用集群模式）集群最多拥有 500 个分区，每个分区中包含 1 到 5 个只读副本节点。



Redis（已禁用集群模式）集群与 Redis（已启用集群模式）集群比较

下表总结了 Redis（禁用集群模式）集群和 Redis（已启用集群模式）集群之间的重要差异。

比较 Redis（已禁用集群模式）集群与 Redis（已启用集群模式）集群

功能	Redis（已禁用集群模式）	Redis（已启用集群模式）
可修改	是。支持添加和删除副本节点，以及纵向扩展节点类型。	有限。有关更多信息，请参阅 引擎版本和升级 和 扩展 Redis（启用集群模式）集群。
数据分区	否	是
分片	1	1 至 500
只读副本	0 到 5 **Important** 如果您没有副本并且节点失败，就会遇到全部数据丢失的情况。	每个分区 0 至 5 个。 **Important** 如果您没有副本并且节点失败，则分片中的所有数据将丢失。
多可用区	是，至少 1 个副本。	是

功能	Redis（已禁用集群模式）	Redis（已启用集群模式）
	可选。默认情况下处于打开状态。	可选。默认情况下处于打开状态。
快照（备份）	是，创建单个 .rdb 文件。	是，为每个分片创建单个 .rdb 文件。
还原	是，使用 Redis（已禁用集群模式）集群中的单个 .rdb 文件。	是，使用 Redis（已禁用集群模式）集群或 Redis（已启用集群模式）集群中的 .rdb 文件。
支持	所有 Redis 版本	Redis 3.2 和更高版本
可升级引擎	是，但有一些限制。有关更多信息，请参阅 引擎版本和升级。	是，但有一些限制。有关更多信息，请参阅 引擎版本和升级。
加密	版本 3.2.6（计划终止生命周期，请参阅 Redis 版本生命周期终止计划）、4.0.10 及更高版本。	版本 3.2.6（计划终止生命周期，请参阅 Redis 版本生命周期终止计划）、4.0.10 及更高版本。
符合 HIPAA 要求	版本 3.2.6（计划终止生命周期，请参阅 Redis 版本生命周期终止计划）、4.0.10 及更高版本。	版本 3.2.6（计划终止生命周期，请参阅 Redis 版本生命周期终止计划）、4.0.10 及更高版本。
与 PCI DSS 兼容	版本 3.2.6（计划终止生命周期，请参阅 Redis 版本生命周期终止计划）、4.0.10 及更高版本。	版本 3.2.6（计划终止生命周期，请参阅 Redis 版本生命周期终止计划）、4.0.10 及更高版本。
在线重新分片	不适用	版本 3.2.10（计划终止生命周期，请参阅 Redis 版本生命周期终止计划）及更高版本。

我应该使用哪一种？

在 Redis（已禁用集群模式）或 Redis（已启用集群模式）之间进行选择时，请考虑以下因素：

- 扩展与分区 – 业务需求在发生变化。您需要针对峰值需求进行预置，还是随需求变化进行扩展。Redis（已禁用集群模式）支持扩展。您可以通过添加或删除副本节点来扩展读取容量，或者通过纵向扩展到更大的节点类型来扩展容量。所有这些操作都需要一些时间。有关更多信息，请参阅扩展具有副本节点的 Redis（已禁用集群模式）集群。

  Redis（已启用集群模式）支持将数据分配到最多 500 个节点组。您可以根据业务的变更需求，动态更改分片数量。分区的优势之一是您可以将负载分散到更多数量的终端节点上，从而减少峰值期间的访问瓶颈。此外，由于数据可分散到多个服务器上，您可以容纳更大的数据集。有关扩展分区的信息，请参阅扩展 Redis（启用集群模式）集群。

- 节点大小与节点数 – 由于 Redis（已禁用集群模式）集群只有一个分区，节点类型必须足够大才能容纳所有集群的数据以及涵盖所需开销。另一方面，由于您在使用 Redis（已启用集群模式）集群时可以将数据分区到多个分区上，节点类型可以较小，虽然您需要的节点的数量更多。有关更多信息，请参阅 选择节点大小。

- 读取与写入 – 如果集群上的主要负载是读取数据的应用程序，则您可以通过添加和删除只读副本来扩展 Redis（已禁用集群模式）集群。不过，最多只能有 5 个只读副本。如果集群上的负载为写入密集型负载，则您可以获益于具有多分区的 Redis（已启用集群模式）集群上的额外写入端点。

不论您选择实施什么类型的集群，请确保选择足以满足您现在和未来需求的节点类型。

使用多可用区最大限度地 ElastiCache 缩短 Redis 的停机时间

在许多情况下 ElastiCache ，For Redis 可能需要更换主节点；其中包括某些类型的计划内维护以及主节点或可用区故障这种不太可能发生的事件。

在这些情形下进行替换时，会导致集群出现停机时间，但如果启用了多可用区，则会最大限度缩短停机时间。主节点的角色会自动将故障转移到其中一个只读副本。无需创建和配置新的主节点，因为 ElastiCache 这将以透明的方式处理这个问题。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。

ElastiCache 还会传播已提升副本的域名服务 (DNS) 名称。这样做的原因是，如果您的应用程序写入到主终端节点，则无需在应用程序中进行终端节点更改。如果您从单个终端节点进行读取，请确保将提升为主节点的副本的读取终端节点更改为新副本的终端节点。

如果由于维护更新或自助服务更新而启动了计划的节点替换，请注意以下事项：

- 对 ElastiCache 于 Redis Cluster，在集群处理传入的写入请求时，计划中的节点更换已完成。
- 对于启用了多可用区并在 5.0.6 或更高版本引擎上运行的已禁用 Redis 集群模式的集群，在集群处理传入的写请求时，完成计划的节点替换。
- 对于启用了多可用区且在 4.0.10 或更早版本的引擎上运行的已禁用 Redis 集群模式的集群，您可能会注意到与 DNS 更新关联的短暂写入中断。此中断可能需要几秒钟。此过程比重新创建并预置新的主节点过程要快得多，后者是在您未启用多可用区的情况下使用的过程。

您可以使用 ElastiCache 管理控制台 AWS CLI、或 ElastiCache API 启用多可用区。

在 Redis 集群（在 API 和 CLI 中，复制组）上启用 ElastiCache 多可用区可以提高容错能力。在集群的读取/写入主集群出于任何原因变得无法连接或发生故障时，此情况尤其如此。仅在每个分片中有多个节点的 Redis 集群上支持多可用区。

主题

启用多可用区

在使用 ElastiCache 控制台或 API 创建或修改集群（API 或 CLI、复制组）时 AWS CLI，您可以启用多可用区。 ElastiCache

您只能在具有至少一个可用只读副本的 Redis（已禁用集群模式）集群上启用多可用区。没有只读副本的集群不提供高可用性或容错能力。有关创建具有复制功能的集群的更多信息，请参阅创建 Redis 复制组。有关将只读副本添加到具有复制功能的集群的信息，请参阅向 Redis（已禁用集群模式）复制组添加只读副本。

主题

- 启用多可用区（控制台）
- 启用多可用区 (AWS CLI)
- 启用多可用区 (ElastiCache API)

启用多可用区（控制台）

您可以在创建新的 Redis 集群时使用 ElastiCache 控制台启用多可用区，也可以通过复制修改现有 Redis 集群来启用多可用区。

默认情况下，Redis（已启用集群模式）集群上已启用多可用区。

> **Important**
>
> ElastiCache 仅当集群在与主分片不同的可用区中至少包含一个副本时，才会自动启用多可用区。

使用控制台创建集群时启用多可用区 ElastiCache

有关此过程的更多信息，请参阅 创建 Redis（已禁用集群模式）集群（控制台）。确保有一个或多个副本并启用多可用区。

在现有集群上启用多可用区（控制台）

有关此过程的更多信息，请参阅修改集群使用 AWS Management Console。

启用多可用区 (AWS CLI)

以下代码示例使用 AWS CLI 为复制组`redis12`启用多可用区。

> **Important**
>
> 复制组 `redis12` 必须已存在且具有至少一个可用只读副本。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
    --replication-group-id redis12 \
    --automatic-failover-enabled \
    --multi-az-enabled \
    --apply-immediately
```

对于 Windows：

```
aws elasticache modify-replication-group ^
    --replication-group-id redis12 ^
    --automatic-failover-enabled ^
    --multi-az-enabled ^
    --apply-immediately
```

该命令的 JSON 输出内容应如下所示。

```
{
    "ReplicationGroup": {
        "Status": "modifying",
        "Description": "One shard, two nodes",
        "NodeGroups": [
            {
                "Status": "modifying",
                "NodeGroupMembers": [
                    {
                        "CurrentRole": "primary",
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
 "redis12-001.v5r9dc.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "redis12-001"
                    },
```

```
                    {
                        "CurrentRole": "replica",
                        "PreferredAvailabilityZone": "us-west-2a",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
 "redis12-002.v5r9dc.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "redis12-002"
                    }
                ],
                "NodeGroupId": "0001",
                "PrimaryEndpoint": {
                    "Port": 6379,
                    "Address": "redis12.v5r9dc.ng.0001.usw2.cache.amazonaws.com"
                }
            }
        ],
        "ReplicationGroupId": "redis12",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "enabling",
        "MultiAZ": "enabled",
        "SnapshotWindow": "07:00-08:00",
        "SnapshottingClusterId": "redis12-002",
        "MemberClusters": [
            "redis12-001",
            "redis12-002"
        ],
        "PendingModifiedValues": {}
    }
}
```

有关更多信息，请参阅 AWS CLI 命令参考中的下列主题：

- create-cache-cluster
- create-replication-group
- AWS CLI 命令参考中的 modify-replication-group。

启用多可用区 (ElastiCache API)

以下代码示例使用 ElastiCache API 为复制组`redis12`启用多可用区。

> **Note**
>
> 要使用此示例，复制组 `redis12` 必须已存在且具有至少一个可用只读副本。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyReplicationGroup
    &ApplyImmediately=true
    &AutoFailover=true
    &MultiAZEnabled=true
    &ReplicationGroupId=redis12
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20140401T192317Z
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 ElastiCache API 参考中的以下主题：

- CreateCache集群
- CreateReplication群组
- ModifyReplication群组

故障情形及多可用区响应

在引入多可用区之前，通过重新创建和重新配置故障节点来 ElastiCache 检测并替换集群的故障节点。如果启用多可用区，发生故障的主节点将故障转移至复制滞后最小的副本。选定副本会自动提升为主节点，这比创建并重新预配置新的主节点快得多。提升过程通常只需几秒钟的时间，然后您可以再次对集群进行写入。

启用多可用区后， ElastiCache 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

如果只有主节点出现故障，则复制滞后最小的只读副本将提升为主节点。然后，将在与发生故障的主节点相同的可用区域中创建和预置替换只读副本。

当只有主节点出现故障时， ElastiCache 多可用区会执行以下操作：

1. 发生故障的主节点脱机。
2. 复制滞后最小的只读副本将提升为主节点。

   一旦提升过程完成（通常只需几秒钟的时间），写入操作就会恢复。如果您的应用程序正在写入主终端节点，则无需更改用于写入或读取的终端节点。 ElastiCache传播已提升副本的 DNS 名称。
3. 启动和预配置替代只读副本。

   将在可用区 (发生故障的主节点的位置) 启动替换只读副本，以便维护节点的分配。
4. 该副本将与新的主节点同步。

新的副本可用后，请注意以下影响：

- 主端点 – 由于新主节点的 DNS 名称会传播到主端点，因此您不需要对应用程序进行任何更改。
- 读取端点 – 读取器终端节点会自动更新为指向新的副本节点。

有关查找集群的终端节点的信息，请参阅以下主题：

- 查找 Redis（已禁用集群模式）集群的端点（控制台）
- 查找复制组的端点（AWS CLI）
- 查找复制组的端点（ElastiCache API）

当主节点和一些只读副本发生故障时的故障情形

如果主节点和至少一个只读副本发生故障，则具有最低复制滞后的可用副本将提升到主集群，并在与故障节点以及提升为主节点的副本相同的可用区中创建新只读副本。

当主节点和某些只读副本出现故障时， ElastiCache 多可用区会执行以下操作：

1. 发生故障的主节点和发生故障的只读副本脱机。
2. 复制滞后最小的可用副本将提升为主节点。

   一旦提升过程完成（通常只需几秒钟的时间），写入操作就会恢复。如果您的应用程序正在写入主终端节点，则无需更改用于写入的终端节点。 ElastiCache 传播已提升副本的 DNS 名称。
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. 所有集群将与新的主节点同步。

在新节点可用后，对应用程序进行以下更改：

- 主端点 – 不要对应用程序进行任何更改。新主节点的 DNS 名称将传播到主终端节点。
- 读取端点 – 读取端点会自动更新为指向新的副本节点。

有关查找复制组的终端节点的信息，请参阅以下主题：

- 查找 Redis（已禁用集群模式）集群的端点（控制台）
- 查找复制组的端点（AWS CLI）
- 查找复制组的端点（ElastiCache API）

整个集群出现故障时的故障情形

如果整个集群全部发生故障，则在与原始节点相同的可用区中重新创建所有节点并预配置。

在此情况下，由于集群中的每个节点均发生故障，因此集群中的所有数据将丢失。这种情况很少出现。

当整个集群出现故障时， ElastiCache 多可用区会执行以下操作：

1. 发生故障的主节点和只读副本脱机。
2. 创建和预配置替换主节点。
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，以便维护节点的分配。

   由于整个集群发生故障，因此数据将丢失，并且所有新节点将冷启动。

由于每个替换节点具有与其要替换的节点相同的终端节点，因此不需要在应用程序中对任何终端节点进行更改。

有关查找复制组的终端节点的信息，请参阅以下主题：

- 查找 Redis（已禁用集群模式）集群的端点（控制台）
- 查找复制组的端点（AWS CLI）
- 查找复制组的端点（ElastiCache API）

建议您在不同的可用区内创建主节点和只读副本以提高容错能力水平。

测试自动故障转移

启用自动故障转移后，您可以使用 ElastiCache 控制台 AWS CLI、和 ElastiCache API 对其进行测试。

在测试时，请注意以下内容：

- 在任何 24 小时内，您可以使用此操作在最多 15 个分片（在 ElastiCache API 中称为节点组和 AWS CLI）上测试自动故障转移。
- 如果在不同集群的分片 (在 API 和 CLI 中称为复制组) 上调用此操作，您可以让调用同时进行。
- 在某些情况下，您可能会在同一 Redis（已启用集群模式）复制组中的不同分区上多次调用此操作。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。
- 要确定节点更换是否已完成，请使用 Amazon ElastiCache 控制台 AWS CLI、或 ElastiCache API 检查事件。查找下列与自动故障转移相关的事件，此处按事件的可能发生顺序列出：

  1. 复制组消息：`Test Failover API called for node group <node-group-id>`
  2. 缓存集群消息：`Failover from primary node <primary-node-id> to replica node <node-id> completed`
  3. 复制组消息：`Failover from primary node <primary-node-id> to replica node <node-id> completed`
  4. 缓存集群消息：`Recovering cache nodes <node-id>`
  5. 缓存集群消息：`Finished recovery for cache nodes <node-id>`

  有关更多信息，请参阅下列内容：

  - 《ElastiCache 用户指南》中的 查看 ElastiCache 事件
  - 《ElastiCache API 参考》中的 DescribeEvents
  - AWS CLI 命令参考中的 describe-events。
- 此 API 专为测试应用程序在 ElastiCache 故障转移时的行为而设计。它不是用于启动故障转移以解决集群问题的操作工具。此外，在某些情况下，例如大规模运营事件， AWS 可能会阻止此 API。

主题

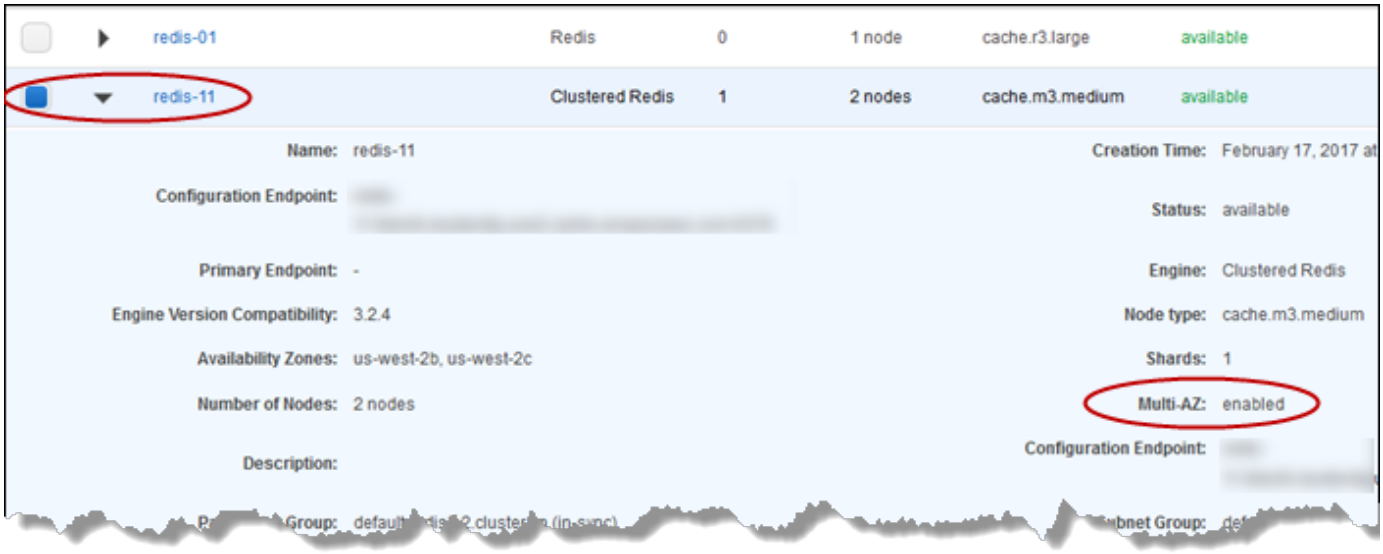
- 使用测试自动故障转移 AWS Management Console
- 使用测试自动故障转移 AWS CLI
- 使用 ElastiCache API 测试自动故障转移

使用测试自动故障转移 AWS Management Console

使用以下过程测试通过控制台进行自动故障转移。

测试自动故障转移

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格中，选择 Redis。
3. 从 Redis 集群列表选择要测试的集群名称左侧的复选框。此集群必须至少有一个只读副本节点。
4. 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 使用 AWS Management Console。



5. 对于 Redis（已禁用集群模式），请选择集群的名称。

   对于 Redis（已启用集群模式），请执行以下操作：

   a. 选择集群的名称。

   b. 在 Shards 页面上，对于要测试故障转移的分片 (在 API 和 CLI 中称为节点组)，选择分片的名称。

6. 在“Nodes”页面上，选择 Failover Primary。
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（`Test Failover API called`）和已完成（`Recovery completed`）的事件。

使用测试自动故障转移 AWS CLI

您可以使用该 AWS CLI 操作`test-failover`在任何启用多可用区的集群上测试自动故障转移。

参数

- `--replication-group-id` – 必需。要测试的复制组 (在控制台上为集群)。
- `--node-group-id` – 必需。要在其上测试自动故障转移的节点组的名称。在连续的 24 小时内，您最多可以测试 15 个节点组。

以下示例使用测试 Redis（已启用集群模式）群`redis00`集`redis00-0003`中的节点组上的自动故障转移。 AWS CLI

Example 测试自动故障转移

对于 Linux、macOS 或 Unix：

```
aws elasticache test-failover \
   --replication-group-id redis00 \
   --node-group-id redis00-0003
```

对于 Windows：

```
aws elasticache test-failover ^
   --replication-group-id redis00 ^
   --node-group-id redis00-0003
```

上面命令的输出类似于下面所示。

```
{
    "ReplicationGroup": {
        "Status": "available",
        "Description": "1 shard, 3 nodes (1 + 2 replicas)",
        "NodeGroups": [
            {
                "Status": "available",
                "NodeGroupMembers": [
                    {
                        "CurrentRole": "primary",
                        "PreferredAvailabilityZone": "us-west-2c",
```

```
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
"redis1x3-001.7ekv3t.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "redis1x3-001"
                    },
                    {
                        "CurrentRole": "replica",
                        "PreferredAvailabilityZone": "us-west-2a",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
"redis1x3-002.7ekv3t.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "redis1x3-002"
                    },
                    {
                        "CurrentRole": "replica",
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                            "Port": 6379,
                            "Address":
"redis1x3-003.7ekv3t.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "redis1x3-003"
                    }
                ],
                "NodeGroupId": "0001",
                "PrimaryEndpoint": {
                    "Port": 6379,
                    "Address": "redis1x3.7ekv3t.ng.0001.usw2.cache.amazonaws.com"
                }
            }
        ],
        "ClusterEnabled": false,
        "ReplicationGroupId": "redis1x3",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "enabled",
        "MultiAZ": "enabled",
        "SnapshotWindow": "11:30-12:30",
```

```
        "SnapshottingClusterId": "redis1x3-002",
        "MemberClusters": [
            "redis1x3-001",
            "redis1x3-002",
            "redis1x3-003"
        ],
        "CacheNodeType": "cache.m3.medium",
        "DataTiering": "disabled",
        "PendingModifiedValues": {}
    }
}
```

要跟踪故障转移的进度，请使用 AWS CLI `describe-events`操作。

有关更多信息，请参阅下列内容：

- AWS CLI 命令参考中的 test-failover。
- AWS CLI 命令参考中的 describe-events。

使用 ElastiCache API 测试自动故障转移

您可以使用 ElastiCache API 操作`TestFailover`在任何启用了多可用区的集群上测试自动故障转移。

参数

- `ReplicationGroupId` – 必需。要测试的复制组（在控制台上为集群）。
- `NodeGroupId` – 必需。要在其上测试自动故障转移的节点组的名称。在连续的 24 小时内，您最多可以测试 15 个节点组。

以下示例在复制组 (在控制台上为集群) `redis00` 中的节点组 `redis00-0003` 上测试自动故障转移。

Example 测试自动失效转移

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=TestFailover
    &NodeGroupId=redis00-0003
    &ReplicationGroupId=redis00
    &Version=2015-02-02
```

```
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20140401T192317Z
    &X-Amz-Credential=<credential>
```

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

有关更多信息，请参阅下列内容：

- TestFailover在 ElastiCache API 参考中
- DescribeEvents在 ElastiCache API 参考中

Redis 多可用区的限制

请注意 Redis 多可用区的以下限制：

- Redis 版本 2.8.6 和更高版本支持多可用区。
- T1 节点类型不支持 Redis 多可用区。
- Redis 复制是异步的。因此，当主节点故障转移到副本时，可能因复制滞后导致丢失少量数据。

  在选择要升级为主副本时， ElastiCache 对于 Redis，请选择复制延迟最小的副本。换句话说，它选择最新的副本。这样做有助于最大程度地减少丢失的数据量。复制滞后最小的副本可以位于同一发生故障的主节点中，也可以位于不同于发生故障的主节点的可用区中。
- 在 Redis（已禁用集群模式）上手动将只读副本提升为主副本时，只有在多可用区和自动故障转移功能已禁用时才能执行此操作。要将只读副本提升为主副本，请执行以下步骤：
  1. 禁用集群上的多可用区。
  2. 禁用集群上的自动故障转移。您可以在 Redis 控制台中取消勾选复制组的 Auto failover（自动故障转移）复选框来执行此操作。您可以 AWS CLI 通过在调用`ModifyReplicationGroup`操作`false`时将`AutomaticFailoverEnabled`属性设置为，使用来实现此目的。
  3. 将只读副本提升为主集群。
  4. 重新启用多可用区。
- ElastiCache 适用于 Redis 的多可用区和仅限附加的文件 (AOF) 是相互排斥的。启用了一个则不能启用另一个。
- 节点的故障可能是因极少出现的整个可用区故障造成的。在此情况下，仅在备份可用区时才会创建替换故障主副本的副本。例如，假设一个复制组的主节点在 AZ-a 中且副本在 AZ-b 和 AZ-c 中。如果

主节点出现故障，则复制滞后最小的副本将提升为主集群。然后，仅当 AZ-a 已备份且可用时，才在 AZ-a（出现故障的主节点所在的位置）中 ElastiCache 创建新的副本。

- 客户发起的主节点重启不会触发自动故障转移。其他重启和故障会触发自动故障转移。
- 在重启主节点后，将在其重新联机时清除其数据。当只读副本查看清除的主集群时，它们将清除其数据的副本，这会导致数据丢失。
- 在提升只读副本后，另一个副本将与新的主节点同步。初始同步后，副本的内容将被删除，它们会同步来自新主节点的数据。此同步过程会导致短暂的中断，在此期间无法访问复制副本。此同步过程还导致在与副本同步时增加主节点上的临时负载。这种行为是 Redis 原生的，并不是 ElastiCache多可用区所独有的。有关此 Redis 行为的详细信息，请参阅 Redis 网站上的复制。

> **Important**
>
> 对于 Redis 版本 2.8.22 及更高版本，您无法创建外部副本。
> 对于 2.8.22 之前的 Redis 版本，我们建议您不要将外部 Redis 副本连接到支持多可用区的 ElastiCache 适用于 Redis 的集群。这种不支持的配置可能会导致无法正常执行故障转移和恢复的问题。 ElastiCache 要将外部 Redis 副本连接到 ElastiCache集群，请确保在建立连接之前未启用多可用区。

如何实施同步和备份

所有支持的 Redis 版本均支持在主节点与副本节点之间备份和同步。但是，实施备份和同步的方式因 Redis 版本而异。

Redis 版本 2.8.22 及更高版本

版本 2.8.22 及更高版本的 Redis 复制可在两种方法之间选择。有关更多信息，请参阅 2.8.22 版之前的 Redis 和 快照和还原。

在无分支过程中，如果写入负载较重，则对集群的写入将延迟，以确保您不会累积太多更改并因而阻止成功快照。

2.8.22 版之前的 Redis

2.8.22 版之前的 Redis 备份和同步过程分为三步。

1. 分支以及后台进程会将集群数据序列化到磁盘。这将创建时间点快照。
2. 在前台中，在客户端输出缓冲区 中累积更改日志。

> **Important**
>
> 如果更改日志超出了客户端输出缓冲区 大小，则备份或同步将失败。有关更多信息，请参阅确保具有用于创建 Redis 快照的足够内存。

3. 最后，依次将缓存数据和更改日志传输到副本节点。

创建 Redis 复制组

在创建具有副本节点的集群时，您有以下选择。当您已有一个可用 Redis（已禁用集群模式）集群且此集群未与具有要用作主节点的副本的任何集群关联时，其中一个选项适用。当您需要使用集群和只读副本创建主节点时，会应用另一个选择。目前，Redis（已启用集群模式）集群必须从头开始创建。

选项 1：使用可用 Redis（已禁用集群模式）集群创建复制组

使用此选项可使用现有单节点 Redis（已禁用集群模式）集群。您可以指定此现有节点作为新集群中的主节点，然后单独向集群中添加 1 到 5 个只读副本。如果现有集群处于活动状态，则只读副本在创建时都将与该集群同步。请参阅 使用可用 Redis（已禁用集群模式）集群创建复制组。

> **Important**
>
> 无法使用现有集群创建 Redis（已启用集群模式）集群。要使用 ElastiCache 控制台创建 Redis（已启用集群模式）集群（API/CLI：复制组），请参阅。创建 Redis（已启用集群模式）集群（控制台）

选项 2：从头创建 Redis 复制组

如果您还没有任何可用的 Redis（禁用集群模式）集群以用作集群的主节点，或者您希望创建 Redis（启用集群模式）集群，请使用此选项。请参阅 从头创建 Redis 复制组。

使用可用 Redis（已禁用集群模式）集群创建复制组

可用的集群是现有的单节点 Redis 集群。当前，Redis（已启用集群模式）不支持使用可用的单节点集群创建具有副本的集群。如果您要创建 Redis（已启用集群模式）集群，请参阅 创建 Redis（已启用集群模式）集群（控制台）。

下面的程序只适用于您具有单节点 Redis（已禁用集群模式）集群的情况。该集群的节点将成为新集群中的主节点。如果您没有可用作新集群的主集群的 Redis（已禁用集群模式）集群，请参阅 从头创建 Redis 复制组。

使用可用 Redis 集群创建复制组（控制台）

请参阅主题使用 AWS Management Console。

使用可用 Redis 缓存集群创建复制组 (AWS CLI)

使用 AWS CLI时，在将可用 Redis 缓存集群用作主集群时，可使用只读副本通过两个步骤创建复制组。

使用时， AWS CLI 您可以创建一个复制组，将可用的独立节点指定为集群的主节点，`--primary-cluster-id`并使用 CLI 命令指定集群中要包含的节点数量`create-replication-group`。包括以下参数。

--replication-group-id

正在创建的复制组的名称。此参数的值用作所添加节点的名称的基础，方法是向 `--replication-group-id` 末尾添加一个连续的 3 位数。例如，`sample-repl-group-001`。

Redis（已禁用集群模式）复制组命名约束如下：

- 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。

--replication-group-description

复制组的描述。

--num-node-groups

您希望此集群中包含的节点数。此值包含主节点。此参数的最大值为 6。

--primary-cluster-id

要用作此复制组中主节点的可用 Redis（已禁用集群模式）集群的节点的名称。

以下命令通过将可用 Redis（已禁用集群模式）集群 `redis01` 用作复制组的主节点来创建复制组 `sample-repl-group`。它将创建 2 个为只读副本的新节点。`redis01` 的设置（即参数组、安全组、节点类型、引擎版本等）将应用于复制组中的所有节点。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
   --replication-group-id sample-repl-group \
   --replication-group-description "demo cluster with replicas" \
   --num-cache-clusters 3 \
   --primary-cluster-id redis01
```

对于 Windows：

```
aws elasticache create-replication-group ^
   --replication-group-id sample-repl-group ^
   --replication-group-description "demo cluster with replicas" ^
   --num-cache-clusters 3 ^
   --primary-cluster-id redis01
```

有关您可能要使用的其他信息和参数，请参阅 AWS CLI 主题create-replication-group。

接下来，向复制组添加只读副本

创建复制组后，使用 `create-cache-cluster` 命令向复制组添加 1 到 5 个只读副本，并确保包含以下参数。

--cache-cluster-id

正在向复制组添加的集群的名称。

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。

--replication-group-id

正在将此缓存集群添加到的复制组的名称。

对要添加到复制组的每个只读副本重复此命令，并仅更改 `--cache-cluster-id` 参数的值。

> **Note**
>
> 请记住，一个复制组最多只能有五个只读副本。尝试向已有五个只读副本的复制组添加只读副本会导致操作失败。

以下代码将只读副本 `my-replica01` 添加到复制组 `sample-repl-group`。主集群的设置（参数组、安全组、节点类型等）将在节点添加到复制组时应用到节点。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-cache-cluster \
   --cache-cluster-id my-replica01 \
   --replication-group-id sample-repl-group
```

对于 Windows：

```
aws elasticache create-cache-cluster ^
   --cache-cluster-id my-replica01 ^
   --replication-group-id sample-repl-group
```

此命令的输出如下所示。

```
{
    "ReplicationGroup": {
        "Status": "creating",
        "Description": "demo cluster with replicas",
        "ClusterEnabled": false,
        "ReplicationGroupId": "sample-repl-group",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "disabled",
        "SnapshotWindow": "00:00-01:00",
```

```
        "SnapshottingClusterId": "redis01",
        "MemberClusters": [
            "sample-repl-group-001",
            "sample-repl-group-002",
            "redis01"
        ],
        "CacheNodeType": "cache.m4.large",
        "DataTiering": "disabled",
        "PendingModifiedValues": {}
    }
}
```

有关更多信息，请参阅以下 AWS CLI 主题：

- create-replication-group
- modify-replication-group

向独立 Redis（已禁用集群模式）集群添加副本 (ElastiCache API)

使用 ElastiCache API 时，您可以创建一个复制组，将可用的独立节点指定为集群的主节点，`PrimaryClusterId`并使用 CLI 命令指定集群中要包含的节点数量`CreateReplicationGroup`。包括以下参数。

ReplicationGroup我是

正在创建的复制组的名称。此参数的值用作所添加节点的名称的基础，方法是向 `ReplicationGroupId` 末尾添加一个连续的 3 位数。例如，`sample-repl-group-001`。

Redis（已禁用集群模式）复制组命名约束如下：

- 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。

ReplicationGroup描述

有副本的集群的描述。

NumCache集群

您希望此集群中包含的节点数。此值包含主节点。此参数的最大值为 6。

PrimaryCluster我是

要用作此集群中主节点的可用 Redis（已禁用集群模式）集群的名称。

以下命令通过将可用 Redis（已禁用集群模式）集群 `redis01` 用作复制组的主节点来创建具有副本的集群 `sample-repl-group`。它将创建 2 个为只读副本的新节点。`redis01` 的设置（即参数组、安全组、节点类型、引擎版本等）将应用于复制组中的所有节点。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateReplicationGroup
   &Engine=redis
   &EngineVersion=6.0
   &ReplicationGroupDescription=Demo%20cluster%20with%20replicas
   &ReplicationGroupId=sample-repl-group
   &PrimaryClusterId=redis01
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 ElastiCache APL 主题：

- CreateReplication群组
- ModifyReplication群组

接下来，向复制组添加只读副本

创建复制组后，使用 `CreateCacheCluster` 操作向复制组添加 1 到 5 个只读副本，并确保包含以下参数。

CacheCluster我是

正在向复制组添加的集群的名称。

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。

ReplicationGroup我是

正在将此缓存集群添加到的复制组的名称。

对要添加到复制组的每个只读副本重复此操作，并仅更改 CacheClusterId 参数的值。

以下代码将只读副本 myReplica01 添加到复制组 myReplGroup。主集群的设置（参数组、安全组、节点类型等）将在节点添加到复制组时应用到节点。

```
https://elasticache.us-west-2.amazonaws.com/
 ?Action=CreateCacheCluster
 &CacheClusterId=myReplica01
 &ReplicationGroupId=myReplGroup
 &SignatureMethod=HmacSHA256
 &SignatureVersion=4
 &Version=2015-02-02
 &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
 &X-Amz-Credential=[your-access-key-id]/20150202/us-west-2/elasticache/aws4_request
 &X-Amz-Date=20150202T170651Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
 &X-Amz-Signature=[signature-value]
```

有关您可能要使用的其他信息和参数，请参阅 ElastiCache API 主题CreateCacheCluster。

从头创建 Redis 复制组

接下来，您可以找到如何在不将现有 Redis 集群用作主集群的情况下创建 Redis 复制组。您可以使用 ElastiCache 控制台、AWS CLI 或 ElastiCache API 从头开始创建 Redis（已禁用集群模式）或 Redis（已启用集群模式）复制组。

在继续之前，请确定您希望创建 Redis（已禁用集群模式）复制组还是 Redis（已启用集群模式）复制组。有关如何做出决定的指南，请参阅复制：Redis（已禁用集群模式）与 Redis（已启用集群模式）对比。

主题

- 从头创建 Redis（已禁用集群模式）复制组
- 从头开始在 Redis（已启用集群模式）中创建复制组

从头创建 Redis（已禁用集群模式）复制组

您可以使用 ElastiCache控制台、或 ElastiCache API 从头开始创建 Redis（已禁用集群模式）复制组。 AWS CLI Redis（已禁用集群模式）复制组始终有一个节点组、一个主集群和最多 5 个只读副本。Redis（已禁用集群模式）复制组不支持对数据分区。

> **Note**
>
> 每个集群的节点/分片限制最高可提高到 500。若要请求提高限制，请参阅 AWS Service Limits 并在请求中包含实例类型。

要从头开始创建 Redis（已禁用集群模式）复制组，请采用以下方法之一：

从头开始创建 Redis（已禁用集群模式）复制组（AWS CLI）

以下过程使用 AWS CLI创建 Redis（已禁用集群模式）复制组。

当您从头开始创建 Redis（已禁用集群模式）复制组时，只需调用一次 AWS CLI `create-replication-group`命令即可创建该复制组及其所有节点。包括以下参数。

--replication-group-id

正在创建的复制组的名称。

Redis（已禁用集群模式）复制组命名约束如下：

- 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。

--replication-group-description

复制组的描述。

--num-cache-clusters

要使用此复制组、主集群和只读副本创建的节点的数目。

如果您启用多可用区 (`--automatic-failover-enabled`)，则 `--num-cache-clusters` 值必须至少为 2。

--cache-node-type

复制组中的每个节点的节点类型。

ElastiCache 支持以下节点类型。一般而言，与其上一代类型对应项相比，最新一代类型以更低的成本提供了更多内存和计算能力。

有关各节点类型性能详细信息，请参阅 Amazon EC2 实例类型。

--data-tiering-enabled

如果您使用的是 r6gd 节点类型，请设置此参数。如果您不想使用数据分层功能，则设置 `--no-data-tiering-enabled`。有关更多信息，请参阅 数据分层。

--cache-parameter-group

指定与您的引擎版本对应的参数组。如果您运行的是 Redis 3.2.4 或更高版本，请指定 `default.redis3.2` 参数组或者从 `default.redis3.2` 派生的参数组来创建 Redis（已禁用集群模式）复制组。有关更多信息，请参阅 Redis 特定的参数。

--network-type

`ipv4`、`ipv6` 或 `dual-stack`。如果选择双堆栈，则必须将 `--IpDiscovery` 参数设置为 `ipv4` 或 `ipv6`。

--engine

redis

--engine-version

要拥有最丰富的功能，请选择最新的引擎版本。

节点的名称会通过将 `-00#` 添加到复制组名称的后面，从复制组名称得出。例如，通过使用复制组名称 `myReplGroup`，主集群的名称将为 `myReplGroup-001` 以及只读副本的名称将为 `myReplGroup-002` 到 `myReplGroup-006`。

如果要在复制组上启用传输中加密或静态加密，请添加 `--transit-encryption-enabled` 和/或 `--at-rest-encryption-enabled` 参数并满足以下条件。

- 您的复制组必须运行 Redis 版本 3.2.6 或 4.0.10。
- 复制组必须在 Amazon VPC 中创建。
- 还必须包含参数 `--cache-subnet-group`。

- 还必须提供 `--auth-token` 参数以及客户为对此集群执行操作所需的 AUTH 令牌指定的字符串值（密码）。

以下操作使用三个节点（一个主节点和两个副本节点）创建 Redis（已禁用集群模式）复制组 `sample-repl-group`。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
   --replication-group-id sample-repl-group \
   --replication-group-description "Demo cluster with replicas" \
   --num-cache-clusters 3 \
   --cache-node-type cache.m4.large \
   --engine redis
```

对于 Windows：

```
aws elasticache create-replication-group ^
   --replication-group-id sample-repl-group ^
   --replication-group-description "Demo cluster with replicas" ^
   --num-cache-clusters 3 ^
   --cache-node-type cache.m4.large ^
   --engine redis
```

此命令的输出如下所示。

```
{
    "ReplicationGroup": {
        "Status": "creating",
        "Description": "Demo cluster with replicas",
        "ClusterEnabled": false,
        "ReplicationGroupId": "sample-repl-group",
        "SnapshotRetentionLimit": 0,
        "AutomaticFailover": "disabled",
        "SnapshotWindow": "01:30-02:30",
        "MemberClusters": [
            "sample-repl-group-001",
            "sample-repl-group-002",
            "sample-repl-group-003"
        ],
        "CacheNodeType": "cache.m4.large",
```

```
        "DataTiering": "disabled",
        "PendingModifiedValues": {}
    }
}
```

有关您可能要使用的其他信息和参数，请参阅 AWS CLI 主题创建复制组。

从头开始创建 Redis（已禁用集群模式）复制组 (ElastiCache API)

以下过程使用 ElastiCache API 创建 Redis（已禁用集群模式）复制组。

当您从头开始创建 Redis（已禁用集群模式）复制组时，只需调用 ElastiCache API `CreateReplicationGroup` 操作即可创建该复制组及其所有节点。包括以下参数。

ReplicationGroup我是

正在创建的复制组的名称。

Redis（已启用集群模式）复制组命名约束如下：

- 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。

ReplicationGroup描述

您对复制组的描述。

NumCache集群

要使用此复制组、主集群和只读副本创建的节点的总数。

如果您启用多可用区 (`AutomaticFailoverEnabled=true`)，则 `NumCacheClusters` 值必须至少为 2。

CacheNode类型

复制组中的每个节点的节点类型。

ElastiCache 支持以下节点类型。一般而言，与其上一代类型对应项相比，最新一代类型以更低的成本提供了更多内存和计算能力。

有关各节点类型性能详细信息，请参阅 Amazon EC2 实例类型。

--data-tiering-enabled

如果您使用的是 r6gd 节点类型，请设置此参数。如果您不想使用数据分层功能，则设置 --no-data-tiering-enabled。有关更多信息，请参阅 数据分层。

CacheParameter组

指定与您的引擎版本对应的参数组。如果您运行的是 Redis 3.2.4 或更高版本，请指定 default.redis3.2 参数组或者从 default.redis3.2 派生的参数组来创建 Redis（已禁用集群模式）复制组。有关更多信息，请参阅 Redis 特定的参数。

--network-type

ipv4、ipv 或 dual-stack。如果选择双堆栈，则必须将 --IpDiscovery 参数设置为 ipv4 或 ipv6。

引擎

redis

EngineVersion

6.0

节点的名称会通过将 -00# 添加到复制组名称的后面，从复制组名称得出。例如，通过使用复制组名称 myReplGroup，主集群的名称将为 myReplGroup-001 以及只读副本的名称将为 myReplGroup-002 到 myReplGroup-006。

如果要在复制组上启用传输中加密或静态加密，请添加 TransitEncryptionEnabled=true 和/或 AtRestEncryptionEnabled=true 参数并满足以下条件。

- 您的复制组必须运行 Redis 版本 3.2.6 或 4.0.10。
- 复制组必须在 Amazon VPC 中创建。
- 还必须包含参数 CacheSubnetGroup。
- 还必须提供 AuthToken 参数以及客户为对此集群执行操作所需的 AUTH 令牌指定的字符串值（密码）。

以下操作使用创建具有三个节点（一个主节点和两个副本节点）的 Redis（已禁用集群模式）复制组 myReplGroup。

```
https://elasticache.us-west-2.amazonaws.com/
```

```
    ?Action=CreateReplicationGroup
    &CacheNodeType=cache.m4.large
    &CacheParameterGroup=default.redis6.x
    &Engine=redis
    &EngineVersion=6.0
    &NumCacheClusters=3
    &ReplicationGroupDescription=test%20group
    &ReplicationGroupId=myReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

有关您可能要使用的其他信息和参数，请参阅 ElastiCache API 主题CreateReplicationGroup。

从头开始在 Redis（已启用集群模式）中创建复制组

您可以使用 ElastiCache 控制台、或 API 创建 Redis（已启用集群模式）集群（API/CLI：复制组）。AWS CLI ElastiCache Redis（已启用集群模式）复制组可以有 1 到 500 个分片（API/CLI：节点组）；每个分片中可以有一个主节点，以及最多 5 个只读副本。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 90 个节点。此集群配置的范围可以从 90 个分片和 0 个副本到 15 个分片和 5 个副本，这是允许的最大副本数。

如果 Redis 引擎版本为 5.0.6 或更高版本，可将每个集群的节点或分片限制提高到最大值 500。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。有关更多信息，请参阅 创建子网组。

对于低于 5.0.6 的版本，每个集群的限制为 250。

若要请求提高限制，请参阅 AWS Service Limits 并选择限制类型 Nodes per cluster per instance type（每个实例类型的每个集群的节点数）。

在 Redis（已启用集群模式）中创建集群

- 创建 Redis（已启用集群模式）集群（控制台）
- 从头开始创建 Redis（已启用集群模式）复制组（AWS CLI）
- 从 Redis（已启用集群模式）中从头开始创建复制组 (ElastiCache API)

创建 Redis（已启用集群模式）集群（控制台）

若要创建 Redis（已启用集群模式）集群，请参阅 创建 Redis（已启用集群模式）集群（控制台）。请确保启用集群模式 Cluster Mode enabled (Scale Out)（启用集群模式（横向扩展）），并在每个模式中指定至少两个分片和一个副本节点。

从头开始创建 Redis（已启用集群模式）复制组（AWS CLI）

以下过程为使用 AWS CLI创建 Redis（已启用集群模式）复制组。

当您从头开始创建 Redis（已启用集群模式）复制组时，只需调用一次 AWS CLI `create-replication-group`命令即可创建复制组及其所有节点。包括以下参数。

--replication-group-id

正在创建的复制组的名称。

Redis（已启用集群模式）复制组命名约束如下：

- 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。

--replication-group-description

复制组的描述。

--cache-node-type

复制组中的每个节点的节点类型。

ElastiCache 支持以下节点类型。一般而言，与其上一代类型对应项相比，最新一代类型以更低的成本提供了更多内存和计算能力。

有关各节点类型性能详细信息，请参阅 Amazon EC2 实例类型。

--data-tiering-enabled

如果您使用的是 r6gd 节点类型，请设置此参数。如果您不想使用数据分层功能，则设置 `--no-data-tiering-enabled`。有关更多信息，请参阅 数据分层。

--cache-parameter-group

指定 `default.redis6.x.cluster.on` 参数组或者派生自 `default.redis6.x.cluster.on` 的参数组以创建 Redis（已启用集群模式）复制组。有关更多信息，请参阅 Redis 6.x 参数更改。

--engine

redis

--engine-version

3.2.4

--num-node-groups

此复制组中的节点组数量。有效值为 1 到 500。

> **Note**
>
> 每个集群的节点/分片限制最高可提高到 500。若要请求提高限制，请参阅 AWS Service Limits 并选择限制类型“Nodes per cluster per instance type（每个实例类型的每个集群的节点数）”。

--replicas-per-node-group

各节点组中的副本节点数量。有效值为 0 到 5。

--network-type

`ipv4`、`ipv` 或 `dual-stack`。如果选择双堆栈，则必须将 `--IpDiscovery` 参数设置为 `ipv4` 或 `ipv6`。

如果要在复制组上启用传输中加密或静态加密，请添加 `--transit-encryption-enabled` 和/或 `--at-rest-encryption-enabled` 参数并满足以下条件。

- 您的复制组必须运行 Redis 版本 3.2.6 或 4.0.10。
- 复制组必须在 Amazon VPC 中创建。
- 还必须包含参数 `--cache-subnet-group`。
- 还必须提供 `--auth-token` 参数以及客户为对此集群执行操作所需的 AUTH 令牌指定的字符串值（密码）。

以下操作创建具有三个节点组/分片 (--num-node-groups) 的 Redis（已启用集群模式）复制组 `sample-repl-group`，每个节点组/分片具有三个节点，即一个主节点和两个只读副本 (--replicas-per-node-group)。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
   --replication-group-id sample-repl-group \
   --replication-group-description "Demo cluster with replicas" \
   --num-node-groups 3 \
   --replicas-per-node-group 2 \
   --cache-node-type cache.m4.large \
   --engine redis \
   --security-group-ids SECURITY_GROUP_ID \
```

```
   --cache-subnet-group-name SUBNET_GROUP_NAME>
```

对于 Windows：

```
aws elasticache create-replication-group ^
   --replication-group-id sample-repl-group ^
   --replication-group-description "Demo cluster with replicas" ^
   --num-node-groups 3 ^
   --replicas-per-node-group 2 ^
   --cache-node-type cache.m4.large ^
   --engine redis ^
   --security-group-ids SECURITY_GROUP_ID ^
   --cache-subnet-group-name SUBNET_GROUP_NAME>
```

之前的命令生成以下输出。

```
{
    "ReplicationGroup": {
        "Status": "creating",
        "Description": "Demo cluster with replicas",
        "ReplicationGroupId": "sample-repl-group",
        "SnapshotRetentionLimit": 0,
        "AutomaticFailover": "enabled",
        "SnapshotWindow": "05:30-06:30",
        "MemberClusters": [
            "sample-repl-group-0001-001",
            "sample-repl-group-0001-002",
            "sample-repl-group-0001-003",
            "sample-repl-group-0002-001",
            "sample-repl-group-0002-002",
            "sample-repl-group-0002-003",
            "sample-repl-group-0003-001",
            "sample-repl-group-0003-002",
            "sample-repl-group-0003-003"
        ],
        "PendingModifiedValues": {}
    }
}
```

在您从头开始创建 Redis（已启用集群模式）复制组时，您可以使用 `--node-group-configuration` 参数配置集群中的每个分片，如下例中所示，其中配置了两个节点组（控制台：分片）。第一个分片有两个节点：一个主节点和一个只读副本节点。第二个分片有三个节点：一个主节点和两个只读副本节点。

--node-group-configuration

各节点组的配置。`--node-group-configuration` 参数包括以下字段。

- `PrimaryAvailabilityZone` – 此节点组的主节点所在的可用区。如果省略此参数，则为主节点 ElastiCache 选择可用区。

  示例： us-west-2a。

- `ReplicaAvailabilityZones` – 只读副本所在可用区的列表，以逗号分隔。此列表中的可用区数量必须与 `ReplicaCount` 的值匹配。如果省略此参数，则为副本节点 ElastiCache 选择可用区。

  示例："us-west-2a,us-west-2b,us-west-2c"

- `ReplicaCount` – 此节点组中的副本节点数量。
- `Slots` – 指定节点组的键空间的字符串。字符串的格式为 `startKey-endKey`。如果省略此参数，则在节点组之间平均 ElastiCache 分配密钥。

  示例： "0-4999"

以下操作创建具有两个节点组/分片（`--num-node-groups`）的 Redis（已启用集群模式）复制组 `new-group`。与前例不同，各节点组配置为彼此不同 (`--node-group-configuration`)。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
  --replication-group-id new-group \
  --replication-group-description "Sharded replication group" \
  --engine redis \
  --snapshot-retention-limit 8 \
  --cache-node-type cache.m4.medium \
  --num-node-groups 2 \
  --node-group-configuration \
      "ReplicaCount=1,Slots=0-8999,PrimaryAvailabilityZone='us-
east-1c',ReplicaAvailabilityZones='us-east-1b'" \
```

```
      "ReplicaCount=2,Slots=9000-16383,PrimaryAvailabilityZone='us-
east-1a',ReplicaAvailabilityZones='us-east-1a','us-east-1c'"
```

对于 Windows：

```
aws elasticache create-replication-group ^
  --replication-group-id new-group ^
  --replication-group-description "Sharded replication group" ^
  --engine redis ^
  --snapshot-retention-limit 8 ^
  --cache-node-type cache.m4.medium ^
  --num-node-groups 2 ^
  --node-group-configuration \
      "ReplicaCount=1,Slots=0-8999,PrimaryAvailabilityZone='us-
east-1c',ReplicaAvailabilityZones='us-east-1b'" \
      "ReplicaCount=2,Slots=9000-16383,PrimaryAvailabilityZone='us-
east-1a',ReplicaAvailabilityZones='us-east-1a','us-east-1c'"
```

之前的操作生成以下输出。

```
{
    "ReplicationGroup": {
        "Status": "creating",
        "Description": "Sharded replication group",
        "ReplicationGroupId": "rc-rg",
        "SnapshotRetentionLimit": 8,
        "AutomaticFailover": "enabled",
        "SnapshotWindow": "10:00-11:00",
        "MemberClusters": [
            "rc-rg-0001-001",
            "rc-rg-0001-002",
            "rc-rg-0002-001",
            "rc-rg-0002-002",
            "rc-rg-0002-003"
        ],
        "PendingModifiedValues": {}
    }
}
```

有关您可能要使用的其他信息和参数，请参阅 AWS CLI 主题create-replication-group。

从 Redis（已启用集群模式）中从头开始创建复制组 (ElastiCache API)

以下过程使用 ElastiCache API 创建 Redis（已启用集群模式）复制组。

当您从头开始创建 Redis（已启用集群模式）复制组时，只需调用 ElastiCache API `CreateReplicationGroup` 操作即可创建该复制组及其所有节点。包括以下参数。

ReplicationGroup我是

正在创建的复制组的名称。

Redis（已启用集群模式）复制组命名约束如下：

- 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。

ReplicationGroup描述

复制组的描述。

NumNode群组

您希望对此复制组创建的节点组数量。有效值为 1 到 500。

ReplicasPerNodeGroup

各节点组中的副本节点数量。有效值为 0 到 5。

NodeGroup配置

各节点组的配置。`NodeGroupConfiguration` 参数包括以下字段。

- `PrimaryAvailabilityZone` – 此节点组的主节点所在的可用区。如果省略此参数，则为主节点 ElastiCache 选择可用区。

  示例： us-west-2a。

- `ReplicaAvailabilityZones` – 只读副本所在可用区的列表。此列表中的可用区数量必须与 `ReplicaCount` 的值匹配。如果省略此参数，则为副本节点 ElastiCache 选择可用区。
- `ReplicaCount` – 此节点组中的副本节点数量。
- `Slots` – 指定节点组的键空间的字符串。字符串的格式为 `startKey-endKey`。如果省略此参数，则在节点组之间平均 ElastiCache 分配密钥。

示例：“0-4999”

CacheNode类型

复制组中的每个节点的节点类型。

ElastiCache 支持以下节点类型。一般而言，与其上一代类型对应项相比，最新一代类型以更低的成本提供了更多内存和计算能力。

有关各节点类型性能详细信息，请参阅 Amazon EC2 实例类型。

--data-tiering-enabled

如果您使用的是 r6gd 节点类型，请设置此参数。如果您不想使用数据分层功能，则设置 `--no-data-tiering-enabled`。有关更多信息，请参阅 数据分层。

CacheParameter组

指定 `default.redis6.x.cluster.on` 参数组或者派生自 `default.redis6.x.cluster.on` 的参数组以创建 Redis（已启用集群模式）复制组。有关更多信息，请参阅 Redis 6.x 参数更改。

--network-type

`ipv4`、`ipv` 或 `dual-stack`。如果选择双堆栈，则必须将 `--IpDiscovery` 参数设置为 `ipv4` 或 `ipv6`。

引擎

redis

EngineVersion

6.0

如果要在复制组上启用传输中加密或静态加密，请添加 `TransitEncryptionEnabled=true` 和/或 `AtRestEncryptionEnabled=true` 参数并满足以下条件。

- 您的复制组必须运行 Redis 版本 3.2.6 或 4.0.10。
- 复制组必须在 Amazon VPC 中创建。
- 还必须包含参数 `CacheSubnetGroup`。

- 还必须提供 AuthToken 参数以及客户为对此集群执行操作所需的 AUTH 令牌指定的字符串值（密码）。

添加换行符以便于阅读。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateReplicationGroup
   &CacheNodeType=cache.m4.large
   &CacheParemeterGroup=default.redis6.xcluster.on
   &Engine=redis
   &EngineVersion=6.0
   &NumNodeGroups=3
   &ReplicasPerNodeGroup=2
   &ReplicationGroupDescription=test%20group
   &ReplicationGroupId=myReplGroup
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

有关您可能要使用的其他信息和参数，请参阅 ElastiCache API 主题CreateReplicationGroup。

查看复制组的详细信息

有时候您可能希望查看复制组的详细信息。您可以使用 ElastiCache 控制台、AWS CLI for ElastiCache 或 ElastiCache API。查看 Redis（已禁用集群模式）与查看 Redis（已启用集群模式）的控制台操作流程有多不同。

查看复制组的详细信息

- 查看具有副本的 Redis（已禁用集群模式）的详细信息
  - 查看 Redis（已禁用集群模式）复制组的详细信息（控制台）
  - 查看 Redis（已禁用集群模式）复制组的详细信息 (AWS CLI)
  - 查看 Redis（已禁用集群模式）复制组 (ElastiCache API) 的详细信息
- 查看复制组的详细信息：Redis（已启用集群模式）
  - 查看 Redis（已启用集群模式）集群的详细信息（控制台）
  - 查看 Redis（已启用集群模式）集群的详细信息 (AWS CLI)
  - 查看 Redis (已启用集群模式) 集群 (ElastiCache API) 的详细信息

- 查看复制组的详细信息 (AWS CLI)
- 查看复制组的详细信息 (ElastiCache API)

查看具有副本的 Redis（已禁用集群模式）的详细信息

您可以使用 ElastiCache 控制台、for 或 API 查看带有副本（API/CLI：复制组）的 Redis（已禁用集群模式）集群 AWS CLI 的 ElastiCache详细信息。 ElastiCache

查看 Redis（已禁用集群模式）集群的详细信息

- 查看 Redis（已禁用集群模式）复制组的详细信息（控制台）
- 查看 Redis（已禁用集群模式）复制组的详细信息 (AWS CLI)
- 查看 Redis（已禁用集群模式）复制组 (ElastiCache API) 的详细信息

查看 Redis（已禁用集群模式）复制组的详细信息（控制台）

要使用 ElastiCache 控制台查看带有副本的 Redis (已禁用集群模式) 集群的详细信息，请参阅主题。查看 Redis（已禁用集群模式）集群（控制台）的详细信息

查看 Redis（已禁用集群模式）复制组的详细信息 (AWS CLI)

有关显示 Redis（已禁用集群模式）复制组详细信息的 AWS CLI 示例，请参阅查看复制组的详细信息 (AWS CLI)。

查看 Redis（已禁用集群模式）复制组 (ElastiCache API) 的详细信息

有关显示 Redis（已禁用集群模式）复制组详细信息的 ElastiCache API 示例，请参阅查看复制组的详细信息 (ElastiCache API)。

查看复制组的详细信息：Redis（已启用集群模式）

查看 Redis（已启用集群模式）集群的详细信息（控制台）

要使用 ElastiCache 控制台查看 Redis (已启用集群模式) 集群的详细信息，请参阅查看 Redis（已启用集群模式）集群的详细信息（控制台）。

查看 Redis（已启用集群模式）集群的详细信息 (AWS CLI)

有关显示 Redis（已启用集群模式）复制组详细信息的 ElastiCache CLI 示例，请参阅查看复制组的详细信息 (AWS CLI)。

查看 Redis (已启用集群模式) 集群 (ElastiCache API) 的详细信息

有关显示 Redis（已启用集群模式）复制组详细信息的 ElastiCache API 示例，请参阅查看复制组的详细信息 (ElastiCache API)。

查看复制组的详细信息 (AWS CLI)

您可以使用 AWS CLI `describe-replication-groups` 命令查看复制组的详细信息。使用以下可选参数来细化列表。忽略该参数将返回最多 100 个复制组的详细信息。

可选参数

- `--replication-group-id` – 使用此参数列出特定复制组的详细信息。如果指定的复制组有多个节点组，则将按照节点组分组返回的结果。
- `--max-items` – 使用此参数限制列出的复制组数量。`--max-items` 的值不能低于 20 或超过 100。

Example

以下代码列出了最多 100 个复制组的详细信息。

```
aws elasticache describe-replication-groups
```

以下代码列出了 `sample-repl-group` 的详细信息。

```
aws elasticache describe-replication-groups --replication-group-id sample-repl-group
```

以下代码列出了 `sample-repl-group` 的详细信息。

```
aws elasticache describe-replication-groups --replication-group-id sample-repl-group
```

以下代码列出了最多 25 个复制组的详细信息。

```
aws elasticache describe-replication-groups --max-items 25
```

该操作输出类似以下的内容（JSON 格式）。

```
{
   "ReplicationGroups": [
     {
```

```
        "Status": "available",
        "Description": "test",
        "NodeGroups": [
          {
             "Status": "available",
                "NodeGroupMembers": [
                   {
                      "CurrentRole": "primary",
                      "PreferredAvailabilityZone": "us-west-2a",
                      "CacheNodeId": "0001",
                      "ReadEndpoint": {
                         "Port": 6379,
                         "Address": "rg-name-001.1abc4d.0001.usw2.cache.amazonaws.com"
                      },
                      "CacheClusterId": "rg-name-001"
                   },
                   {
                      "CurrentRole": "replica",
                      "PreferredAvailabilityZone": "us-west-2b",
                      "CacheNodeId": "0001",
                      "ReadEndpoint": {
                         "Port": 6379,
                         "Address": "rg-name-002.1abc4d.0001.usw2.cache.amazonaws.com"
                      },
                      "CacheClusterId": "rg-name-002"
                   },
                   {
                      "CurrentRole": "replica",
                      "PreferredAvailabilityZone": "us-west-2c",
                      "CacheNodeId": "0001",
                      "ReadEndpoint": {
                         "Port": 6379,
                         "Address": "rg-name-003.1abc4d.0001.usw2.cache.amazonaws.com"
                      },
                      "CacheClusterId": "rg-name-003"
                   }
                ],
                "NodeGroupId": "0001",
                "PrimaryEndpoint": {
                   "Port": 6379,
                   "Address": "rg-name.1abc4d.ng.0001.usw2.cache.amazonaws.com"
                }
             }
          ],
```

```
         "ReplicationGroupId": "rg-name",
         "AutomaticFailover": "enabled",
         "SnapshottingClusterId": "rg-name-002",
         "MemberClusters": [
            "rg-name-001",
            "rg-name-002",
            "rg-name-003"
         ],
         "PendingModifiedValues": {}
      },
      {
      ... some output omitted for brevity
      }
   ]
}
```

有关更多信息，请参阅 AWS CLI for ElastiCache 主题 describe-replication-groups。

查看复制组的详细信息 (ElastiCache API)

您可以使用 AWS CLI `DescribeReplicationGroups` 操作查看复制的详细信息。使用以下可选参数来细化列表。忽略该参数将返回最多 100 个复制组的详细信息。

可选参数

- `ReplicationGroupId` – 使用此参数列出特定复制组的详细信息。如果指定的复制组有多个节点组，则将按照节点组分组返回的结果。
- `MaxRecords` – 使用此参数限制列出的复制组数量。`MaxRecords` 的值不能低于 20 或超过 100。默认值为 100。

Example

以下代码列出了最多 100 个复制组的详细信息。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeReplicationGroups
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

以下代码列出了 myReplGroup 的详细信息。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeReplicationGroups
   &ReplicationGroupId=myReplGroup
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

以下代码列出了最多 25 个集群的详细信息。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeReplicationGroups
   &MaxRecords=25
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 ElastiCache API 参考主题 DescribeReplicationGroups。

查找复制组端点

应用程序可以连接到复制组中的任何节点，前提是它具有该节点的 DNS 终端节点和端口号。根据您运行的是 Redis（已禁用集群模式）还是 Redis（已启用集群模式）复制组，您可能会关注不同的端点。

Redis（已禁用集群模式）

具有副本的 Redis（已禁用集群模式）集群有三种类型的端点：主端点、读取器终端节点和节点端点。主端点是一个 DNS 名称，始终解析为集群中的主节点。主端点不受集群更改的影响，如将只读副本提升为主角色。对于写入活动，我们建议您的应用程序连接到主端点。

读取器终端节点将在 for Redis 集群中的所有只读副本之间平均分配到该终端节点 ElastiCache 的传入连接。应用程序何时创建连接或应用程序如何（重复）使用连接等附加因素将决定流量分配。读取器端点会在添加或删除副本时实时跟踪集群更改。您可以将 for Redis 集群的多个只读副本放在不同的 AWS 可用区 (AZ) 中，以确保读取器终端节点的高可用性。 ElastiCache

> **Note**
>
> 读取器端点不是负载均衡器。它是一个 DNS 记录，将以循环方式解析为副本节点之一的 IP 地址。

对于读取活动，应用程序还可以连接到集群中的任何节点。与主端点不同，节点端点会解析为特定端点。如果您在您的集群中进行更改（例如添加或删除副本），则必须在您的应用程序中更新节点端点。

Redis（已启用集群模式）

具有副本的 Redis（已启用集群模式）集群 [由于具有多个分区（API/CLI：节点组），它们还具有多个主节点] 与 Redis（已禁用集群模式）集群的端点结构有所不同。Redis（已启用集群模式）具有一个配置端点，其“知道”集群中的所有主端点和节点端点。您的应用程序连接到配置终端节点。只要您的应用程序对集群的配置终端节点进行写入或读取，Redis 在后台确定密钥所属的分片以及分片所使用的终端节点。这对于您的应用程序是完全透明的。

您可以使用 ElastiCache 控制台、或 ElastiCache API 查找集群的 AWS CLI终端节点。

查找复制组的终端节点

要查找复制组的终端节点，请参阅以下主题之一：

- 查找 Redis（已禁用集群模式）集群的端点（控制台）

- 查找 Redis（已启用集群模式）集群的端点（控制台）
- 查找复制组的端点（AWS CLI）
- 查找复制组的端点（ElastiCache API）

修改复制组

⚠ 重要约束

- 目前，ElastiCache 支持对 Redis（已启用集群模式）复制组进行有限的修改，例如使用 API 操作 `ModifyReplicationGroup` (CLI:`modify-replication-group`) 更改引擎版本。您可以使用 API 操作 `ModifyReplicationGroupShardConfiguration`（CLI：`modify-replication-group-shard-configuration`）修改 Redis（已启用集群模式）集群中的分区（节点组）数量。有关更多信息，请参阅 扩展 Redis（启用集群模式）集群。

  要对 Redis（已启用集群模式）集群进行其他修改，您需要使用集成了更改的新集群重新创建新集群。
- 您可以将 Redis（已禁用集群模式）和 Redis（已启用集群模式）集群和复制组升级到较新的引擎版本。不过，您不能降级到较早的引擎版本，除非删除现有集群或复制组并重新创建它。有关更多信息，请参阅 引擎版本和升级 。
- 您可以使用控制台、群ModifyReplication组 API 或 modify-re plication-group CLI 命令将禁用集群模式的现 ElastiCache 有 Redis 集群升级为启用集群模式，如下例所示。也可以按照修改集群模式中的步骤进行操作。

您可以使用 ElastiCache 控制台、或 ElastiCache API 修改 Redis（已禁用集群模式）集群的设置。AWS CLI目前，ElastiCache 支持对 Redis（已启用集群模式）复制组进行有限数量的修改。其他修改要求您创建当前复制组的备份，然后使用此备份为新 Redis（已启用集群模式）复制组设定种子的方式进行修改。

主题

- 使用 AWS Management Console
- 使用 AWS CLI
- 使用 ElastiCache API

使用 AWS Management Console

若要修改 Redis（已禁用集群模式）集群，请参阅 修改集 ElastiCache 群。

使用 AWS CLI

以下是该modify-replication-group命令的 AWS CLI 示例。您可以使用同样的命令对复制组进行其他修改。

在现有 Redis 复制组上启用多可用区：

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --multi-az-enabled = true
```

对于 Windows：

```
aws elasticache modify-replication-group ^
   --replication-group-id myReplGroup ^
   --multi-az-enabled
```

将集群模式从已禁用修改为已启用：

要将集群模式从已禁用修改为已启用，必须先将集群模式设置为兼容。兼容模式允许您的 Redis 客户端使用“已启用集群模式”和“已禁用集群模式”进行连接。在将所有 Redis 客户端迁移为使用“已启用集群模式”后，您可以完成集群模式配置并将集群模式设置为已启用。

对于 Linux、macOS 或 Unix：

将集群模式设置为兼容。

```
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --cache-parameter-group-name myParameterGroupName \
   --cluster-mode compatible
```

将集群模式设置为已启用。

```
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --cluster-mode enabled
```

对于 Windows：

将集群模式设置为兼容。

```
aws elasticache modify-replication-group ^
   --replication-group-id myReplGroup ^
   --cache-parameter-group-name myParameterGroupName ^
   --cluster-mode compatible
```

将集群模式设置为已启用。

```
aws elasticache modify-replication-group ^
   --replication-group-id myReplGroup ^
   --cluster-mode enabled
```

有关该 AWS CLI `modify-replication-group`命令的更多信息，请参阅 for Redis 用户指南中的modify-replication-groupElastiCache 或修改集群模式。

使用 ElastiCache API

以下 ElastiCache API 操作在现有 Redis 复制组上启用多可用区。您可以使用同样的操作对复制组进行其他修改。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyReplicationGroup
   &AutomaticFailoverEnabled=true
   &Mutli-AZEnabled=true
   &ReplicationGroupId=myReplGroup
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20141201T220302Z
   &Version=2014-12-01
   &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
   &X-Amz-Date=20141201T220302Z
   &X-Amz-SignedHeaders=Host
   &X-Amz-Expires=20141201T220302Z
   &X-Amz-Credential=<credential>
   &X-Amz-Signature=<signature>
```

有关 ElastiCache API `ModifyReplicationGroup` 操作的更多信息，请参阅ModifyReplicationGroup。

删除复制组

如果您不再需要某个具有副本的集群（在 API/CLI 中称作复制组），可将其删除。在删除复制组时，ElastiCache 会删除该组中的所有节点。

此操作开始执行后，就无法中断或取消。

> **Warning**
>
> 当您删除 ElastiCache for Redis 集群时，您的手动快照会保留。您还可以选择在删除集群前创建最后一个快照。自动缓存快照不会保留。

删除复制组（控制台）

要删除具有副本的集群，请参阅删除集群。

删除复制组 (AWS CLI)

使用命令 delete-replication-group 删除复制组。

```
aws elasticache delete-replication-group --replication-group-id my-repgroup
```

系统会提示您确认您的决定。输入 y（是）立即开始操作。此过程一经启动便无法撤销。

```
  After you begin deleting this replication group, all of its nodes will be deleted as
well.
  Are you sure you want to delete this replication group? [Ny]y

REPLICATIONGROUP  my-repgroup  My replication group  deleting
```

删除复制组 (ElastiCache API)

调用带 `ReplicationGroup` 参数的 DeleteReplicationGroup。

Example

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DeleteReplicationGroup
   &ReplicationGroupId=my-repgroup
```

```
&Version=2014-12-01
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20141201T220302Z
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256
&X-Amz-Date=20141201T220302Z
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20141201T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

Note

如果您将 `RetainPrimaryCluster` 参数设置为 `true`，则会删除所有只读副本，但是将保留主集群。

更改副本数量

您可以使用、或 API 动态增加或减少 Redis 复制组中只读副本的 AWS CLI数量。 AWS Management Console ElastiCache 如果您的复制组为 Redis（已启用集群模式）复制组，则可以选择要在其中增加或减少副本数量的分区（节点组）。

要动态更改您的 Redis 复制组中的副本数量，请从下表中选择符合您的情况的操作。

要执行的操作	对于 Redis（已启用集群模式）	对于 Redis（已禁用集群模式）
添加副本	增加分区中的副本数量	增加分区中的副本数量 向 Redis（已禁用集群模式）复制组添加只读副本
删除副本	减少分区中的副本数量	减少分区中的副本数量 删除 Redis（已禁用集群模式）复制组的只读副本

增加分区中的副本数量

您可以将 Redis（已启用集群模式）分区或 Redis（已禁用集群模式）复制组中的副本数量最多增加到 5 个。您可以使用 AWS Management Console、AWS CLI 或 ElastiCache API 执行此操作。

主题

- 使用 AWS Management Console
- 使用 AWS CLI
- 使用 ElastiCache API

使用 AWS Management Console

以下过程使用控制台增加 Redis（已启用集群模式）复制组中的副本数量。

增加 Redis 分片中的副本数量

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在导航窗格中，选择 Redis，然后选择要将副本添加到的复制组的名称。
3. 选中要将副本添加到的每个分片对应的框。
4. 选择 Add replicas (添加副本)。
5. 完成 Add Replicas to Shards (将副本添加到分片) 页面：

   - 对于 New number of replicas/shard (新副本/分片数量)，输入您希望所有选定的分片应具有的副本数量。此值必须大于或造型 Current Number of Replicas per shard (每个分片的当前副本数量) 且小于或等于 5。我们建议使用至少两个副本作为有效的最小值。
   - 对于 Availability Zones（可用区），选择 No preference（无首选项）以让 ElastiCache 为每个新副本选择可用区，或者选择 Specify Availability Zones（指定可用区）以为每个新副本选择可用区。

     如果选择 Specify Availability Zones (指定可用区)，对于每个新副本，请使用列表指定可用区。
6. 选择 Add (添加) 以添加副本，或选择 Cancel (取消) 以取消该操作。

使用 AWS CLI

要增加 Redis 分片中的副本数量，请使用带有以下参数的 `increase-replica-count` 命令：

- `--replication-group-id` – 必需。确定要在其中增加副本数量的复制组。
- `--apply-immediately` 或 `--no-apply-immediately` – 必需。指定是立即增加副本数量 (`--apply-immediately`) 还是在下一维护时段增加副本数量 (`--no-apply-immediately`)。当前不支持 `--no-apply-immediately`。
- `--new-replica-count` – 可选。指定完成时所希望的副本节点数量（最多 5 个）。对其中仅有一个节点组的 Redis（已禁用集群模式）复制组或您希望其中的所有节点组均有相同副本数量的 Redis（已启用集群模式）复制组使用此参数。如果此值小于或等于节点组中的当前副本数量，则调用失败并返回异常。
- `--replica-configuration` – 可选。允许您单独地为每个节点组设置副本和可用区的数量。对您希望单独配置其中每个节点组的 Redis（已启用集群模式）组使用此参数。

  `--replica-configuration` 具有三位可选成员：

  - `NodeGroupId` – 您要配置的节点组的四位数 ID。对于 Redis（已禁用集群模式）复制组，分区 ID 始终为 `0001`。若要查找 Redis（已启用集群模式）节点组（分区）ID，请参阅 查找分区的 ID。
  - `NewReplicaCount` – 您希望在此操作结束时此节点组中所具有的副本数量。此值必须大于当前副本数量，最多为 5 个。如果此值小于或等于节点组中的当前副本数量，则调用失败并返回异常。
  - `PreferredAvailabilityZones` – `PreferredAvailabilityZone` 字符串的列表，指定复制组的节点即将位于的可用区。`PreferredAvailabilityZone` 值的数字必须等于 `NewReplicaCount` 的值再加上 1 以形成主节点。如果忽略此 `--replica-configuration` 的成员，ElastiCache for Redis 会为每个新副本选择可用区。

> **Important**
>
> 您必须在调用中包含 `--new-replica-count` 或 `--replica-configuration` 参数，但不能同时包含这两项。

Example

以下示例将复制组 `sample-repl-group` 中的副本数量增加到 3 个。在完成此示例后，每个节点组中将有 3 个副本。无论是具有单个节点组的 Redis（已禁用集群模式）复制组，还是具多个节点组的 Redis（已禁用集群模式）复制组，此数字都适用。

对于 Linux、macOS 或 Unix：

```
aws elasticache increase-replica-count \
    --replication-group-id sample-repl-group \
    --new-replica-count 3 \
    --apply-immediately
```

对于 Windows：

```
aws elasticache increase-replica-count ^
    --replication-group-id sample-repl-group ^
    --new-replica-count 3 ^
    --apply-immediately
```

以下示例将复制组 sample-repl-group 中的副本数量增加到两个指定节点组指定的值。假定存在多个节点组，则这是一个 Redis（已启用集群模式）复制组。指定可选 PreferredAvailabilityZones 时，所列可用区的数量必须等于 NewReplicaCount 的值再加上 1。此方法适用于由 NodeGroupId 标识的组的主节点。

对于 Linux、macOS 或 Unix：

```
aws elasticache increase-replica-count \
    --replication-group-id sample-repl-group \
    --replica-configuration \
        NodeGroupId=0001,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-east-1c,us-east-1b \
        NodeGroupId=0003,NewReplicaCount=3,PreferredAvailabilityZones=us-east-1a,us-east-1b,us-east-1c,us-east-1c \
    --apply-immediately
```

对于 Windows：

```
aws elasticache increase-replica-count ^
    --replication-group-id sample-repl-group ^
    --replica-configuration ^
        NodeGroupId=0001,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-east-1c,us-east-1b ^
        NodeGroupId=0003,NewReplicaCount=3,PreferredAvailabilityZones=us-east-1a,us-east-1b,us-east-1c,us-east-1c \
    --apply-immediately
```

有关使用 CLI 增加副本数量的更多信息，请参阅 Amazon ElastiCache 命令行参考 中的 increase-replica-count。

使用 ElastiCache API

要增加 Redis 分片中的副本数量，请使用带有以下参数的 IncreaseReplicaCount 操作：

- ReplicationGroupId – 必需。确定要在其中增加副本数量的复制组。
- ApplyImmediately – 必需。指定是立即增加副本数量 (ApplyImmediately=True) 还是在下一维护时段增加副本数量 (ApplyImmediately=False)。当前不支持 ApplyImmediately=False。
- NewReplicaCount – 可选。指定完成时所希望的副本节点数量（最多 5 个）。对其中仅有一个节点组的 Redis（已禁用集群模式）复制组或您希望其中的所有节点组均有相同副本数量的 Redis（已启用集群模式）复制组使用此参数。如果此值小于或等于节点组中的当前副本数量，则调用失败并返回异常。
- ReplicaConfiguration – 可选。允许您单独地为每个节点组设置副本和可用区的数量。对您希望单独配置其中每个节点组的 Redis（已启用集群模式）组使用此参数。

  ReplicaConfiguraion 具有三位可选成员：

  - NodeGroupId – 您要配置的节点组的四位数 ID。对于 Redis（已禁用集群模式）复制组，节点组（分区）ID 始终为 0001。若要查找 Redis（已启用集群模式）节点组（分区）ID，请参阅 查找分区的 ID。
  - NewReplicaCount – 您希望在此操作结束时此节点组中所具有的副本数量。此值必须大于当前副本数量，且最多为 5 个。如果此值小于或等于节点组中的当前副本数量，则调用失败并返回异常。
  - PreferredAvailabilityZones – PreferredAvailabilityZone 字符串的列表，指定复制组的节点即将位于的可用区。PreferredAvailabilityZone 值的数字必须等于 NewReplicaCount 的值再加上 1 以形成主节点。如果忽略此 ReplicaConfiguration 的成员，ElastiCache for Redis 会为每个新副本选择可用区。

> **Important**
>
> 您必须在调用中包含 NewReplicaCount 或 ReplicaConfiguration 参数，但不能同时包含这两项。

Example

以下示例将复制组 sample-repl-group 中的副本数量增加到 3 个。在完成此示例后，每个节点组中将有 3 个副本。无论是具有单个节点组的 Redis（已禁用集群模式）复制组，还是具多个节点组的 Redis（已禁用集群模式）复制组，此数字都适用。

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=IncreaseReplicaCount
      &ApplyImmediately=True
      &NewReplicaCount=3
      &ReplicationGroupId=sample-repl-group
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
      &X-Amz-Credential=<credential>
```

以下示例将复制组 sample-repl-group 中的副本数量增加到两个指定节点组指定的值。假定存在多个节点组，则这是一个 Redis（已启用集群模式）复制组。指定可选 PreferredAvailabilityZones 时，所列可用区的数量必须等于 NewReplicaCount 的值再加上 1。此方法适用于由 NodeGroupId 标识的组的主节点。

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=IncreaseReplicaCount
      &ApplyImmediately=True
      &ReplicaConfiguration.ConfigureShard.1.NodeGroupId=0001
      &ReplicaConfiguration.ConfigureShard.1.NewReplicaCount=2

 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.1=
east-1a

 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.2=
east-1c

 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.3=
east-1b
      &ReplicaConfiguration.ConfigureShard.2.NodeGroupId=0003
      &ReplicaConfiguration.ConfigureShard.2.NewReplicaCount=3

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.1=
east-1a
```

```
 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.2=
east-1b

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.3=
east-1c

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.4=
east-1c
      &ReplicationGroupId=sample-repl-group
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
      &X-Amz-Credential=<credential>
```

有关使用 API 增加副本数量的更多信息，请参阅 Amazon ElastiCache API 参考中的 IncreaseReplicaCount。

减少分区中的副本数量

您可以减少 Redis（已启用集群模式）的分区中的副本数量，或 Redis（已禁用集群模式）复制组中的副本数量：

- 对于 Redis（已禁用集群模式），如果启用了多可用区，则可以将副本数量减少到 1 个；如果未启用，则可以将副本数量减少到 0。
- 对于 Redis（已启用集群模式），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 AWS Management Console、 AWS CLI 或 ElastiCache API 来减少节点组（分片）或复制组中的副本数量。

主题

- 使用 AWS Management Console
- 使用 AWS CLI
- 使用 ElastiCache API

使用 AWS Management Console

以下过程使用控制台减少 Redis（已启用集群模式）复制组中的副本数量。

减少 Redis 分片中的副本数量

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格中，选择 Redis，然后选择要从其中删除副本的复制组的名称。
3. 选中要从其中删除副本的每个分片对应的框。
4. 选择 Delete replicas (删除副本)。
5. 完成 Delete Replicas from Shards (从分片中删除副本) 页面：
   a. 对于 New number of replicas/shard (新副本/分片数量)，输入您希望选定的分片应具有的副本数量。此数字必须大于或等于 1。我们建议每个分片使用至少两个副本作为有效的最小值。
   b. 选择 Delete (删除) 以删除副本，或选择 Cancel (取消) 以取消该操作。

> **Important**
>
> - 如果您未指定要删除的副本节点，for Redi ElastiCache s 会自动选择要删除的副本节点。在这样做的同时，f ElastiCache or Redis 会尝试为您的复制组保留多可用区架构，然后保留与主副本的复制延迟最小的副本。
> - 无法删除复制组中的主节点。如果指定主节点进行删除，此操作会失败并显示指示已选中主节点进行删除的错误事件。

使用 AWS CLI

要减少 Redis 分片中的副本数量，请使用带有以下参数的 `decrease-replica-count` 命令：

- `--replication-group-id` – 必需。确定要在其中减少副本数量的复制组。
- `--apply-immediately` 或 `--no-apply-immediately` – 必需。指定是立即减少副本数量 (`--apply-immediately`) 还是在下一维护时段减少副本数量 (`--no-apply-immediately`)。当前不支持 `--no-apply-immediately`。
- `--new-replica-count` – 可选。指定希望的副本节点数。`--new-replica-count` 的值必须为小于节点组中的当前副本数量的有效值。有关允许的最小值，请参阅减少分区中的副本数量。如果 `--new-replica-count` 的值不满足此要求，则调用失败。
- `--replicas-to-remove` – 可选。包含指定要删除的副本节点的节点 ID 的列表。
- `--replica-configuration` – 可选。允许您单独地为每个节点组设置副本和可用区的数量。对您希望单独配置其中每个节点组的 Redis（已启用集群模式）组使用此参数。

  `--replica-configuration` 具有三位可选成员：

  - NodeGroupId – 您要配置的节点组的四位数 ID。对于 Redis（已禁用集群模式）复制组，分区 ID 始终为 `0001`。若要查找 Redis（已启用集群模式）节点组（分区）ID，请参阅 查找分区的 ID。
  - NewReplicaCount – 指定希望的副本节点数的可选参数。NewReplicaCount 的值必须为小于节点组中的当前副本数量的有效值。有关允许的最小值，请参阅减少分区中的副本数量。如果 NewReplicaCount 的值不满足此要求，则调用失败。
  - PreferredAvailabilityZones – PreferredAvailabilityZone 字符串的列表，指定复制组的节点所在的可用区。PreferredAvailabilityZone 值的数字必须等于 NewReplicaCount 的值再加上 1 以形成主节点。如果省略了`--replica-configuration`该成员，ElastiCache 则 Redis 会为每个新副本选择可用区。

> **Important**
>
> 您必须包含且只能包含 --new-replica-count、--replicas-to-remove 或 --replica-configuration 参数之一。

Example

以下示例使用 --new-replica-count 将复制组 sample-repl-group 中的副本数量减少为 1 个。在完成此示例后，每个节点组中将有一个副本。无论是具有单个节点组的 Redis（已禁用集群模式）复制组，还是具多个节点组的 Redis（已禁用集群模式）复制组，此数字都适用。

对于 Linux、macOS 或 Unix：

```
aws elasticache decrease-replica-count
    --replication-group-id sample-repl-group \
    --new-replica-count 1 \
    --apply-immediately
```

对于 Windows：

```
aws elasticache decrease-replica-count ^
    --replication-group-id sample-repl-group ^
    --new-replica-count 1 ^
    --apply-immediately
```

以下示例通过从节点组中删除两个指定的副本（0001 和 0003）来减少复制 sample-repl-group 中的副本数量。

对于 Linux、macOS 或 Unix：

```
aws elasticache decrease-replica-count \
    --replication-group-id sample-repl-group \
    --replicas-to-remove 0001,0003 \
    --apply-immediately
```

对于 Windows：

```
aws elasticache decrease-replica-count ^
    --replication-group-id sample-repl-group ^
```

```
    --replicas-to-remove 0001,0003 \
    --apply-immediately
```

以下示例使用 --replica-configuration 将复制组 sample-repl-group 中的副本数量减少为两个指定节点组指定的值。假定存在多个节点组，则这是一个 Redis（已启用集群模式）复制组。指定可选 PreferredAvailabilityZones 时，所列可用区的数量必须等于 NewReplicaCount 的值再加上 1。此方法适用于由 NodeGroupId 标识的组的主节点。

对于 Linux、macOS 或 Unix：

```
aws elasticache decrease-replica-count \
    --replication-group-id sample-repl-group \
    --replica-configuration \
        NodeGroupId=0001,NewReplicaCount=1,PreferredAvailabilityZones=us-east-1a,us-
east-1c \
        NodeGroupId=0003,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-
east-1b,us-east-1c \
    --apply-immediately
```

对于 Windows：

```
aws elasticache decrease-replica-count ^
    --replication-group-id sample-repl-group ^
    --replica-configuration ^
        NodeGroupId=0001,NewReplicaCount=2,PreferredAvailabilityZones=us-east-1a,us-
east-1c ^
        NodeGroupId=0003,NewReplicaCount=3,PreferredAvailabilityZones=us-east-1a,us-
east-1b,us-east-1c \
    --apply-immediately
```

有关使用 CLI 减少副本数量的更多信息，请参阅《Amazon ElastiCache 命令行参考》中的 d ecrease-replica-count。

使用 ElastiCache API

要减少 Redis 分片中的副本数量，请使用带有以下参数的 DecreaseReplicaCount 操作：

- ReplicationGroupId – 必需。确定要在其中减少副本数量的复制组。
- ApplyImmediately – 必需。指定是立即减少副本数量 (ApplyImmediately=True) 还是在下一维护时段减少副本数量 (ApplyImmediately=False)。当前不支持 ApplyImmediately=False。

- NewReplicaCount – 可选。指定希望的副本节点数。NewReplicaCount 的值必须为小于节点组中的当前副本数量的有效值。有关允许的最小值，请参阅减少分区中的副本数量。如果 --new-replica-count 的值不满足此要求，则调用失败。
- ReplicasToRemove – 可选。包含指定要删除的副本节点的节点 ID 的列表。
- ReplicaConfiguration – 可选。包含节点组列表，这些节点组允许您单独地为每个节点组设置副本和可用区的数量。对您希望单独配置其中每个节点组的 Redis（已启用集群模式）组使用此参数。

  ReplicaConfiguraion 具有三位可选成员：

  - NodeGroupId – 您要配置的节点组的四位数 ID。对于 Redis（已禁用集群模式）复制组，节点组 ID 始终为 0001。若要查找 Redis（已启用集群模式）节点组（分区）ID，请参阅 查找分区的 ID。
  - NewReplicaCount – 您希望在此操作结束时此节点组中所具有的副本数量。如果启用了多可用区，则此值必须小于当前副本数量（最少为 1 个）；如果未启用具有自动故障转移功能的多可用区，则此值为 0。如果此值大于或等于节点组中的当前副本数量，则调用失败并返回异常。
  - PreferredAvailabilityZones – PreferredAvailabilityZone 字符串的列表，指定复制组的节点所在的可用区。PreferredAvailabilityZone 值的数字必须等于 NewReplicaCount 的值再加上 1 以形成主节点。如果省略了ReplicaConfiguration该成员， ElastiCache 则 Redis 会为每个新副本选择可用区。

> **Important**
>
> 您必须包含且只能包含 NewReplicaCount、ReplicasToRemove 或 ReplicaConfiguration 参数之一。

Example

以下示例使用 NewReplicaCount 将复制组 sample-repl-group 中的副本数量减少为 1 个。在完成此示例后，每个节点组中将有一个副本。无论是具有单个节点组的 Redis（已禁用集群模式）复制组，还是具多个节点组的 Redis（已禁用集群模式）复制组，此数字都适用。

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=DecreaseReplicaCount
      &ApplyImmediately=True
      &NewReplicaCount=1
      &ReplicationGroupId=sample-repl-group
```

```
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
      &X-Amz-Credential=<credential>
```

以下示例通过从节点组中删除两个指定的副本（0001 和 0003）来减少复制 sample-repl-group 中的副本数量。

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=DecreaseReplicaCount
      &ApplyImmediately=True
      &ReplicasToRemove.ReplicaToRemove.1=0001
      &ReplicasToRemove.ReplicaToRemove.2=0003
      &ReplicationGroupId=sample-repl-group
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
      &X-Amz-Credential=<credential>
```

以下示例使用 ReplicaConfiguration 将复制组 sample-repl-group 中的副本数量减少为两个指定节点组指定的值。假定存在多个节点组，则这是一个 Redis（已启用集群模式）复制组。指定可选 PreferredAvailabilityZones 时，所列可用区的数量必须等于 NewReplicaCount 的值再加上 1。此方法适用于由 NodeGroupId 标识的组的主节点。

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=DecreaseReplicaCount
      &ApplyImmediately=True
      &ReplicaConfiguration.ConfigureShard.1.NodeGroupId=0001
      &ReplicaConfiguration.ConfigureShard.1.NewReplicaCount=1

 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.1=
east-1a

 &ReplicaConfiguration.ConfigureShard.1.PreferredAvailabilityZones.PreferredAvailabilityZone.2=
east-1c
      &ReplicaConfiguration.ConfigureShard.2.NodeGroupId=0003
      &ReplicaConfiguration.ConfigureShard.2.NewReplicaCount=2

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.1=
east-1a
```

```
 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.2=
east-1b

 &ReplicaConfiguration.ConfigureShard.2.PreferredAvailabilityZones.PreferredAvailabilityZone.4=
east-1c
      &ReplicationGroupId=sample-repl-group
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
      &X-Amz-Credential=<credential>
```

有关使用 API 减少副本数量的更多信息，请参阅 Amazon ElastiCache API 参考中的DecreaseReplica计数。

向 Redis（已禁用集群模式）复制组添加只读副本

以下主题中的信息仅适用于 Redis（已禁用集群模式）复制组。

随着您的读取流量的增加，您可能需要跨多个节点分布这些读取操作，并且减少任一节点上的读取压力。在本主题中，您可以了解如何向 Redis（已禁用集群模式）集群添加只读副本。

Redis（已禁用集群模式）复制组最多可以有五个只读副本。如果您尝试向已有 5 个只读副本的复制组添加只读副本，则此操作将失败。

有关向 Redis（已启用集群模式）复制组添加副本的信息，请参阅以下内容：

- 扩展 Redis（启用集群模式）集群
- 增加分区中的副本数量

您可以使用 ElastiCache 控制台、AWS CLI 或 ElastiCache API 向 Redis（已禁用集群模式）集群添加只读副本。

相关主题

- 向集群添加节点
- 向复制组添加只读副本 (AWS CLI)
- 使用 API 向复制组添加只读副本

向复制组添加只读副本 (AWS CLI)

要向 Redis（已禁用集群模式）复制组添加只读副本，请使用 AWS CLI `create-cache-cluster` 命令，其中参数 `--replication-group-id` 指定要向其添加集群（节点）的复制组。

以下示例创建集群 `my-read replica` 并将其添加到复制组 `my-replication-group`。该只读副本的节点类型、参数组、安全组、维护时段及其他设置与 `my-replication-group` 中的其他节点的相同。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-cache-cluster \
      --cache-cluster-id my-read-replica \
      --replication-group-id my-replication-group
```

对于 Windows：

```
aws elasticache create-cache-cluster ^
      --cache-cluster-id my-read-replica ^
      --replication-group-id my-replication-group
```

有关使用 CLI 添加只读副本的更多信息，请参阅 Amazon ElastiCache 命令行参考中的 create-cache-cluster。

使用 API 向复制组添加只读副本

要向 Redis（已禁用集群模式）复制组添加只读副本，请使用 ElastiCache `CreateCacheCluster` 操作，其中参数 `ReplicationGroupId` 指定要向其添加集群（节点）的复制组。

以下示例创建集群 `myReadReplica` 并将其添加到复制组 `myReplicationGroup`。该只读副本的节点类型、参数组、安全组、维护时段及其他设置与 `myReplicationGroup` 中的其他节点的相同。

```
https://elasticache.us-west-2.amazonaws.com/
      ?Action=CreateCacheCluster
      &CacheClusterId=myReadReplica
      &ReplicationGroupId=myReplicationGroup
      &Version=2015-02-02
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20150202T192317Z
      &X-Amz-Credential=<credential>
```

有关使用 API 添加只读副本的更多信息，请参阅 Amazon ElastiCache API 参考中的 CreateCacheCluster。

删除 Redis（已禁用集群模式）复制组的只读副本

以下主题中的信息仅适用于 Redis（已禁用集群模式）复制组。

由于 Redis 复制组上的读取流量是不断变化的，您可能需要添加或移除只读副本。删除 Redis（已禁用集群模式）复制组中的节点与删除集群的过程相同，但存在一些限制：

- 您无法从复制组中移除主集群。如果要删除主集群，请执行以下操作：
  1. 将只读副本提升为主集群。有关将只读副本提升为主集群的更多信息，请参阅将 Redis（已禁用集群模式）复制组的只读副本提升为主节点。
  2. 删除旧的主集群。有关此方法的限制，请参阅下一要点。
- 如果在复制组上启用了多可用区，则无法从复制组中移除上一个只读副本。在此情况下，请执行以下操作：
  1. 通过禁用多可用区来修改复制组。有关更多信息，请参阅修改复制组。
  2. 删除只读副本。

您可以使用 ElastiCache 控制台、AWS CLI 或 ElastiCache API 删除 Redis（已禁用集群模式）复制组中的只读副本。

有关从 Redis 复制组中删除集群的说明，请参阅以下内容：

- 使用 AWS Management Console
- 使用 AWS CLI
- 使用 ElastiCache API
- 扩展 Redis（启用集群模式）集群
- 减少分区中的副本数量

将 Redis（已禁用集群模式）复制组的只读副本提升为主节点

以下主题中的信息仅适用于 Redis（已禁用集群模式）复制组。

您可以使用 AWS Management Console、或 ElastiCache API 将 Redis（已禁用集群模式）只读副本提升为 AWS CLI主副本。当 Redis（已禁用集群模式）复制组上启用了多可用区（具有自动故障转移功能）时，无法将只读副本提升为主节点。若要在启用了多可用区的复制组上将 Redis（已禁用集群模式）副本提升为主节点，请执行以下操作：

1. 修改复制组以禁用多可用区（执行此操作不要求所有集群都位于同一个可用区）。有关更多信息，请参阅 修改复制组。
2. 将只读副本提升为主集群。
3. 修改复制组以重新启用多可用区。

在运行 Redis 2.6.13 或更早版本的复制组上，多可用区不可用。

使用 AWS Management Console

以下过程使用控制台将副本节点提升为主集群。

将只读副本提升为主节点（控制台）

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 如果要提升的副本是启用了多可用区的 Redis（已禁用集群模式）复制组中的成员，请先修改复制组以禁用多可用区，然后再继续。有关更多信息，请参阅 修改复制组。
3. 选择 Redis，然后从集群列表中选择要修改的复制组。该复制组必须运行“Redis”引擎，而不是“集群化 Redis”引擎，而且必须具有 2 个或更多个节点。
4. 从节点列表中，选择要提升为主集群的副本节点，然后对于 Actions (操作)，选择 Promote (提升)。
5. 在 Promote Read Replica (提升只读副本) 对话框中，执行以下操作：
   a. 对于 Apply Immediately (立即应用)，选择 Yes (是) 立即提升只读副本，或者选择 No (否) 在集群的下一维护时段提升它。
   b. 选择 Promote 提升只读副本，或选择 Cancel 取消该操作。
6. 如果在开始提升过程之前集群已启用多可用区，请等待直到复制组的状态为 available (可用)，然后修改集群以重新启用多可用区。有关更多信息，请参阅 修改复制组。

使用 AWS CLI

当复制组启用多可用区时，您无法将只读副本提升为主集群。在某些情况下，要提升的副本可能是启用了多可用区的复制组的成员。在这些情况下，您必须先修改复制组以禁用多可用区，然后再继续。执行此操作不要求所有集群都位于同一个可用区。有关修改复制组的更多信息，请参阅修改复制组。

以下 AWS CLI 命令修改复制组`sample-repl-group`，使只读副本成为复制组`my-replica-1`中的主副本。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
   --replication-group-id sample-repl-group \
   --primary-cluster-id my-replica-1
```

对于 Windows：

```
aws elasticache modify-replication-group ^
   --replication-group-id sample-repl-group ^
   --primary-cluster-id my-replica-1
```

有关修改复制组的更多信息，请参阅《Amazon ElastiCache 命令行参考》modify-replication-group中的。

使用 ElastiCache API

当复制组启用多可用区时，您无法将只读副本提升为主集群。在某些情况下，要提升的副本可能是启用了多可用区的复制组的成员。在这些情况下，您必须先修改复制组以禁用多可用区，然后再继续。执行此操作不要求所有集群都位于同一个可用区。有关修改复制组的更多信息，请参阅修改复制组。

以下 ElastiCache API 操作修改复制组`myReplGroup`，使只读副本成为复制组`myReplica-1`中的主副本。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyReplicationGroup
   &ReplicationGroupId=myReplGroup
   &PrimaryClusterId=myReplica-1
   &Version=2014-12-01
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20141201T220302Z
```

```
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256
&X-Amz-Date=20141201T220302Z
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20141201T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

有关修改复制组的更多信息，请参阅 Amazon ElastiCache API 参考ModifyReplicationGroup中的。

管理维护

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

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

区域代码	区域名称	区域维护时段
ap-northeast-1	亚太（东京）区域	13:00–21:00 UTC
ap-northeast-2	亚太地区（首尔）区域	12:00–20:00 UTC
ap-northeast-3	亚太地区（大阪）区域	12:00–20:00 UTC
ap-southeast-3	亚太地区（雅加达）区域	14:00–22:00 UTC
ap-south-1	亚太地区（孟买）区域	17:30–1:30 UTC
ap-southeast-1	亚太（新加坡）区域	14:00–22:00 UTC
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
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
eu-west-3	欧洲（巴黎）区域	23:59–07:29 UTC

区域代码	区域名称	区域维护时段
af-south-1	非洲（开普敦）区域	13:00–21:00 UTC
eu-central-1	欧洲地区（法兰克福）区域	23:00–07:00 UTC
eu-west-1	欧洲地区（爱尔兰）区域	22:00–06:00 UTC
eu-west-2	欧洲地区（伦敦）区域	23:00–07:00 UTC
me-south-1	中东（巴林）区域	13:00–21:00 UTC
me-central-1	中东（阿联酋）区域	13:00–21:00 UTC
eu-south-1	欧洲地区（米兰）	21:00–05:00 UTC
sa-east-1	南美洲（圣保罗）区域	01:00–09:00 UTC
us-east-1	美国东部（弗吉尼亚州北部）区域	03:00–11:00 UTC
us-east-2	美国东部（俄亥俄州）区域	04:00–12:00 UTC
us-gov-west-1	AWS GovCloud (US) 区域	06:00–14:00 UTC
us-west-1	美国西部（北加利福尼亚）区域	06:00–14:00 UTC
us-west-2	美国西部（俄勒冈）区域	06:00–14:00 UTC

更改您的集群或复制组的维护时段

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或复制组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。

> **Note**
>
> 如果要使用 AWS Management Console立即应用节点类型修改和/或引擎升级，请选择 Apply Immediately（立即应用）框。否则，将在下一计划的维护时段中应用这些修改。要使用 API，请参阅 modify-replication-group 或 modify-cache-cluster。

更多信息

有关维护时段和节点替换的信息，请参阅：

- ElastiCache 维护 – 有关维护和节点替换的常见问题
- 替换节点 – 管理节点替换
- 修改复制组 – 更改复制组的维护时段

使用参数组配置引擎参数

Amazon ElastiCache 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。

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 Redis 节点类型的参数。

主题

- 参数管理
- 缓存参数组层
- 创建参数组
- 按名称列出参数组
- 列出参数组的值
- 修改参数组
- 删除参数组
- Memcached 特定的参数
- Redis 特定的参数

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 3.2.10，您只能使用 Redis 3.2 系列中的参数组 (默认或自定义)。
- 如果更改某个集群的参数组，则任何可以按照条件修改的参数的值在当前参数组和新参数组中必须相同。
- 当您更改集群的参数时，所做的更改将立即或在集群节点重启后应用于集群，但以下说明的例外情况除外。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。要确定何时应用特定参数更改，请参阅表格中的更改生效列以了解 Redis 特定的参数。

  有关更多信息，请参阅重启节点。

  > **Redis（已启用集群模式）参数更改**
  >
  > 如果要更改 Redis（已启用集群模式）集群上的以下参数，请按照随后的步骤操作。
  >
  > - activerehashing
  > - 数据库
  >
  > 1. 创建集群的手动备份。请参阅进行手动备份。
  > 2. 删除 Redis（已启用集群模式）集群。请参阅删除集群。
  > 3. 使用修改的参数组和备份还原集群，以便为新集群创建种子。请参阅从备份还原到新缓存。
  >
  > 对其他参数的更改不需要执行此操作。

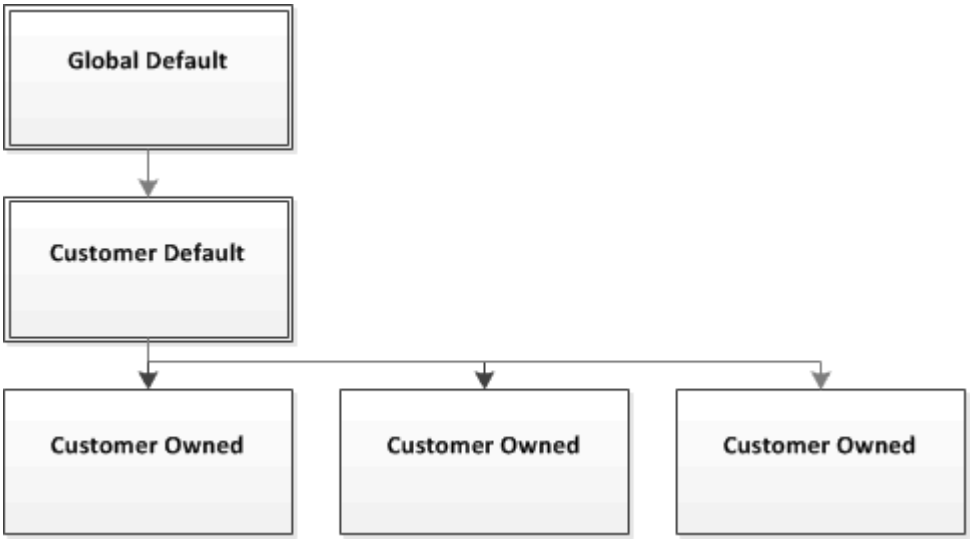
- 您可以将参数组与 Redis 全局数据存储关联。全局数据存储是跨 AWS 区域的一个或多个集群的集合。在这种情况下，参数组由组成全局数据存储的所有集群共享。对主集群的参数组作出的任何修改都会复制到全局数据存储中的所有剩余集群。有关更多信息，请参阅使用全球数据存储跨 AWS 区域复制。

您可以通过查看以下位置来检查参数组是否属于全局数据存储：

- ElastiCache 控制台的 Parameter Groups（参数组）页面上的是/否 Global（全局）属性
- CacheParameterGroup API 操作的是/否 `IsGlobal` 属性

缓存参数组层

Amazon ElastiCache 具有三层缓存参数组，如下所示。



Amazon ElastiCache 参数组层

全局默认值

区域中所有 Amazon ElastiCache 客户的顶级根参数组。

全局默认缓存参数组：

- 预留供 ElastiCache 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缓存参数组的副本。

客户默认缓存参数组：

- 由 ElastiCache 创建和所有。
- 可供客户用作缓存参数组，用于运行此缓存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

客户默认缓存参数组的副本。客户拥有的缓存参数组在客户创建缓存参数组时创建。

客户拥有的缓存参数组：

- 由客户创建并拥有。
- 可以分配给任意客户兼容的集群。
- 可由客户修改用于创建自定义缓存参数组。

  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。

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 ElastiCache 控制台、AWS CLI 或 ElastiCache API 创建参数组。

创建参数组（控制台）

以下过程介绍了如何使用 ElastiCache 控制台创建参数组。

使用 ElastiCache 控制台创建参数组

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 要查看所有可用的参数组列表，请在导航窗格左侧选择Parameter Groups。
3. 要创建参数组，请选择 Create Parameter Group。

   Create Parameter Group（创建参数组）屏幕随即出现。
4. 从 Family 列表中，选择将作为参数组的模板的参数组系列。

   参数组系列（例如 redis3.2）定义了参数组中的实际参数及其初始值。参数组系列必须与集群的引擎和版本一致。
5. 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。

6. 在 Description 框中，键入参数组的说明。
7. 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。
8. 创建参数组后，它将具有系列的默认值。要更改默认值，您必须修改参数组。有关更多信息，请参阅修改参数组。

创建参数组 (AWS CLI)

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

- `--cache-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。
- `--cache-parameter-group-family` – 参数组的引擎和版本系列。
- `--description` – 用户提供的参数组描述。

Example

以下示例使用 redis2.8 系列作为模板来创建名为 myRed28 的参数组。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-cache-parameter-group \
    --cache-parameter-group-name myRed28  \
    --cache-parameter-group-family redis2.8 \
    --description "My first parameter group"
```

对于 Windows：

```
aws elasticache create-cache-parameter-group ^
    --cache-parameter-group-name myRed28  ^
    --cache-parameter-group-family redis2.8 ^
    --description "My first parameter group"
```

该命令的输出内容应类似如下所示。

```
{
    "CacheParameterGroup": {
        "CacheParameterGroupName": "myRed28",
        "CacheParameterGroupFamily": "redis2.8",
        "Description": "My first parameter group"
    }
}
```

创建参数组后，它将具有系列的默认值。要更改默认值，您必须修改参数组。有关更多信息，请参阅修改参数组。

有关更多信息，请参阅create-cache-parameter-group。

创建参数组 (ElastiCache API)

要使用 ElastiCache API 创建参数组，请使用带以下参数的 CreateCacheParameterGroup 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。
- CacheParameterGroupFamily – 参数组的引擎和版本系列。例如，redis2.8。
- Description – 用户提供的参数组描述。

Example

以下示例使用 redis2.8 系列作为模板来创建名为 myRed28 的参数组。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateCacheParameterGroup
   &CacheParameterGroupFamily=redis2.8
   &CacheParameterGroupName=myRed28
   &Description=My%20first%20parameter%20group
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

来自此操作的响应应类似如下所示。

```
<CreateCacheParameterGroupResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <CreateCacheParameterGroupResult>
    <CacheParameterGroup>
      <CacheParameterGroupName>myRed28</CacheParameterGroupName>
      <CacheParameterGroupFamily>redis2.8</CacheParameterGroupFamily>
      <Description>My first parameter group</Description>
    </CacheParameterGroup>
  </CreateCacheParameterGroupResult>
  <ResponseMetadata>
    <RequestId>d8465952-af48-11e0-8d36-859edca6f4b8</RequestId>
  </ResponseMetadata>
</CreateCacheParameterGroupResponse>
```

创建参数组后，它将具有系列的默认值。要更改默认值，您必须修改参数组。有关更多信息，请参阅修改参数组。

有关更多信息，请参阅CreateCacheParameterGroup。

按名称列出参数组

您可以使用 ElastiCache 控制台、AWS CLI 或 ElastiCache API 列出参数组。

按名称列出参数组（控制台）

以下过程介绍了如何使用 ElastiCache 控制台查看参数组列表。

使用 ElastiCache 控制台列出参数组

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
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 (AWS CLI)

要使用 AWS CLI 生成参数组的列表，请使用命令 `describe-cache-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-records` 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 myRed28。

对于 Linux、macOS 或 Unix：

```
aws elasticache describe-cache-parameter-groups \
    --cache-parameter-group-name myRed28
```

对于 Windows：

```
aws elasticache describe-cache-parameter-groups ^
    --cache-parameter-group-name myRed28
```

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。

```
{
    "CacheParameterGroups": [
     {
         "CacheParameterGroupName": "myRed28",
         "CacheParameterGroupFamily": "redis2.8",
         "Description": "My first parameter group"
```

```
     }
    ]
}
```

Example

以下示例代码列出了在 Redis 引擎 5.0.6 和更高版本上运行的参数组 MyRed56。如果参数组是 使用全球数据存储跨 AWS 区域复制 的一部分，则在输出中返回的 IsGlobal 属性值将为 Yes。

对于 Linux、macOS 或 Unix：

```
aws elasticache describe-cache-parameter-groups \
    --cache-parameter-group-name myRed56
```

对于 Windows：

```
aws elasticache describe-cache-parameter-groups ^
    --cache-parameter-group-name myRed56
```

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，以及是否属于全局数据存储。

```
{
    "CacheParameterGroups": [
     {
         "CacheParameterGroupName": "myRed56",
         "CacheParameterGroupFamily": "redis5.0",
         "Description": "My first parameter group",
         "IsGlobal": "yes"
     }
    ]
}
```

Example

以下示例代码列出最多 10 个参数组。

```
aws elasticache describe-cache-parameter-groups --max-records 10
```

该命令的 JSON 输出将类似如下所示，列出每个参数组的名称、系列和描述，如果是 redis5.6，还会列出该参数组是否属于全局数据存储 (isGlobal)。

```
{
```

```
    "CacheParameterGroups": [
        {
            "CacheParameterGroupName": "custom-redis32",
            "CacheParameterGroupFamily": "redis3.2",
            "Description": "custom parameter group with reserved-memory > 0"
        },
        {
            "CacheParameterGroupName": "default.memcached1.4",
            "CacheParameterGroupFamily": "memcached1.4",
            "Description": "Default parameter group for memcached1.4"
        },
        {
            "CacheParameterGroupName": "default.redis2.6",
            "CacheParameterGroupFamily": "redis2.6",
            "Description": "Default parameter group for redis2.6"
        },
        {
            "CacheParameterGroupName": "default.redis2.8",
            "CacheParameterGroupFamily": "redis2.8",
            "Description": "Default parameter group for redis2.8"
        },
        {
            "CacheParameterGroupName": "default.redis3.2",
            "CacheParameterGroupFamily": "redis3.2",
            "Description": "Default parameter group for redis3.2"
        },
        {
            "CacheParameterGroupName": "default.redis3.2.cluster.on",
            "CacheParameterGroupFamily": "redis3.2",
            "Description": "Customized default parameter group for redis3.2 with
 cluster mode on"
        },
        {
            "CacheParameterGroupName": "default.redis5.6.cluster.on",
            "CacheParameterGroupFamily": "redis5.0",
            "Description": "Customized default parameter group for redis5.6 with
 cluster mode on",
            "isGlobal": "yes"
        },
    ]
}
```

有关更多信息，请参阅[describe-cache-parameter-groups](#)。

按名称列出参数组 (ElastiCache API)

要使用 ElastiCache API 生成参数组的列表，请使用 `DescribeCacheParameterGroups` 操作。如果提供了参数组的名称，将只会列出该参数组。如果未提供参数组的名称，将列出最多 `MaxRecords` 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 10 个参数组。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameterGroups
   &MaxRecords=10
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &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，如果是 redis5.6，还会列出该参数组是否属于全局数据存储 (isGlobal)。

```
<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
        <CacheParameterGroupName>myRedis28</CacheParameterGroupName>
        <CacheParameterGroupFamily>redis2.8</CacheParameterGroupFamily>
        <Description>My custom Redis 2.8 parameter group</Description>
      </CacheParameterGroup>
      <CacheParameterGroup>
        <CacheParameterGroupName>myMem14</CacheParameterGroupName>
        <CacheParameterGroupFamily>memcached1.4</CacheParameterGroupFamily>
        <Description>My custom Memcached 1.4 parameter group</Description>
      </CacheParameterGroup>
       <CacheParameterGroup>
        <CacheParameterGroupName>myRedis56</CacheParameterGroupName>
        <CacheParameterGroupFamily>redis5.0</CacheParameterGroupFamily>
        <Description>My custom redis 5.6 parameter group</Description>
        <isGlobal>yes</isGlobal>
      </CacheParameterGroup>
    </CacheParameterGroups>
```

```
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>
```

Example

以下示例代码列出了参数组 myRed28。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameterGroups
   &CacheParameterGroupName=myRed28
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &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。

```
<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
        <CacheParameterGroupName>myRed28</CacheParameterGroupName>
        <CacheParameterGroupFamily>redis2.8</CacheParameterGroupFamily>
        <Description>My custom Redis 2.8 parameter group</Description>
      </CacheParameterGroup>
    </CacheParameterGroups>
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>
```

Example

以下示例代码列出了参数组 myRed56。

```
https://elasticache.us-west-2.amazonaws.com/
```

```
    ?Action=DescribeCacheParameterGroups
    &CacheParameterGroupName=myRed56
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &Version=2015-02-02
    &X-Amz-Credential=<credential>
```

此操作所得到的响应将类似如下所示，列出名称、系列、描述，以及该参数组是否属于全局数据存储 (isGlobal)。

```
<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
        <CacheParameterGroupName>myRed56</CacheParameterGroupName>
        <CacheParameterGroupFamily>redis5.0</CacheParameterGroupFamily>
        <Description>My custom Redis 5.6 parameter group</Description>
        <isGlobal>yes</isGlobal>
      </CacheParameterGroup>
    </CacheParameterGroups>
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>
```

有关更多信息，请参阅DescribeCacheParameterGroups。

列出参数组的值

您可以使用 ElastiCache 控制台、AWS CLI 或 ElastiCache API 列出参数组的参数及其值。

列出参数组的值（控制台）

以下过程介绍了如何使用 ElastiCache 控制台列出参数组的参数及其值。

使用 ElastiCache 控制台列出参数组的参数及其值

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
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框来选择要列出其中包含的参数及其值的参数组。

   屏幕底部将列出这些参数及其值。由于参数的数量，您可能需要上下滚动来查找所需的参数。

列出参数组的值 (AWS CLI)

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

Example

以下示例代码列出了参数组 myRedis28 的所有参数及其值。

对于 Linux、macOS 或 Unix：

```
aws elasticache describe-cache-parameters \
    --cache-parameter-group-name myRedis28
```

对于 Windows：

```
aws elasticache describe-cache-parameters ^
    --cache-parameter-group-name myRed28
```

有关更多信息，请参阅`describe-cache-parameters`。

列出参数组的值 (ElastiCache API)

要使用 ElastiCache API 列出参数组的参数及其值，请使用 `DescribeCacheParameters` 操作。

Example

以下示例代码列出了参数组 myRed28 的所有参数。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameters
   &CacheParameterGroupName=myRed28
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

来自此操作的响应将类似如下所示。此响应已被截断。

```
<DescribeCacheParametersResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParametersResult>
    <CacheClusterClassSpecificParameters>
      <CacheNodeTypeSpecificParameter>
        <DataType>integer</DataType>
        <Source>system</Source>
        <IsModifiable>false</IsModifiable>
        <Description>The maximum configurable amount of memory to use to store items,
 in megabytes.</Description>
        <CacheNodeTypeSpecificValues>
          <CacheNodeTypeSpecificValue>
            <Value>1000</Value>
            <CacheClusterClass>cache.c1.medium</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>6000</Value>
            <CacheClusterClass>cache.c1.xlarge</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>7100</Value>
            <CacheClusterClass>cache.m1.large</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>1300</Value>
            <CacheClusterClass>cache.m1.small</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
```

```
...output omitted...

    </CacheClusterClassSpecificParameters>
  </DescribeCacheParametersResult>
  <ResponseMetadata>
    <RequestId>6d355589-af49-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParametersResponse>
```

有关更多信息，请参阅DescribeCacheParameters。

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。

修改参数组（控制台）

以下过程介绍了如何使用 ElastiCache 控制台更改 `cluster-enabled` 参数的值。您可以使用相同的过程来更改任意参数的值。

使用 ElastiCache 控制台更改参数的值

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
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框来选择要修改的参数组。

   屏幕底部将列出参数组的参数。您可能需要浏览列表才能查看所有参数。
4. 要修改一个或多个参数，请选择 Edit Parameters。
5. 选择 Save Changes。
6. 要查找您更改的参数名称，请参阅Redis 特定的参数。如果您的集群为 Redis（已禁用集群模式）集群并对以下参数进行了更改，则您必须重启集群中的节点：

- activerehashing
- 数据库

有关更多信息，请参阅重启节点。

> **Redis（已启用集群模式）参数更改**
>
> 如果要更改 Redis（已启用集群模式）集群上的以下参数，请按照随后的步骤操作。
>
> - activerehashing
> - 数据库
>
> 1. 创建集群的手动备份。请参阅进行手动备份。
> 2. 删除 Redis（已启用集群模式）集群。请参阅删除集群。
> 3. 使用修改的参数组和备份还原集群，以便为新集群创建种子。请参阅从备份还原到新缓存。
>
> 对其他参数的更改不需要执行此操作。

修改参数组 (AWS CLI)

要使用 AWS CLI 更改参数值，请使用命令 `modify-cache-parameter-group`。

Example

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

以下示例代码演示设置两个参数的值：参数组 `myredis32-on-30` 的 reserved-memory-percent 和 cluster-enabled。我们将 reserved-memory-percent 设置为 `30` (30%) 并将 cluster-enabled 设置为 `yes`，以便参数组可与 Redis（已启用集群模式）集群（复制组）搭配使用。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-cache-parameter-group \
    --cache-parameter-group-name myredis32-on-30 \
```

```
    --parameter-name-values \
        ParameterName=reserved-memory-percent,ParameterValue=30 \
        ParameterName=cluster-enabled,ParameterValue=yes
```

对于 Windows：

```
aws elasticache modify-cache-parameter-group ^
    --cache-parameter-group-name myredis32-on-30 ^
    --parameter-name-values ^
        ParameterName=reserved-memory-percent,ParameterValue=30 ^
        ParameterName=cluster-enabled,ParameterValue=yes
```

此命令的输出如下所示。

```
{
    "CacheParameterGroupName": "my-redis32-on-30"
}
```

有关更多信息，请参阅modify-cache-parameter-group。

要查找您更改的参数名称，请参阅Redis 特定的参数。

如果您的集群为 Redis（已禁用集群模式）集群并对以下参数进行了更改，则您必须重启集群中的节点：

- activerehashing
- 数据库

有关更多信息，请参阅重启节点。

> **Redis（已启用集群模式）参数更改**
>
> 如果要更改 Redis（已启用集群模式）集群上的以下参数，请按照随后的步骤操作。
>
> - activerehashing
> - 数据库
>
> 1. 创建集群的手动备份。请参阅进行手动备份。
> 2. 删除 Redis（已启用集群模式）集群。请参阅删除集群。

3. 使用修改的参数组和备份还原集群，以便为新集群创建种子。请参阅从备份还原到新缓存。

对其他参数的更改不需要执行此操作。

修改参数组 (ElastiCache API)

要使用 ElastiCache API 更改参数组的参数值，请使用 `ModifyCacheParameterGroup` 操作。

Example

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

以下示例代码演示设置两个参数的值：参数组 `myredis32-on-30` 的 reserved-memory-percent 和 cluster-enabled。我们将 reserved-memory-percent 设置为 `30` (30%) 并将 cluster-enabled 设置为 `yes`，以便参数组可与 Redis（已启用集群模式）集群（复制组）搭配使用。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyCacheParameterGroup
   &CacheParameterGroupName=myredis32-on-30
   &ParameterNameValues.member.1.ParameterName=reserved-memory-percent
   &ParameterNameValues.member.1.ParameterValue=30
   &ParameterNameValues.member.2.ParameterName=cluster-enabled
   &ParameterNameValues.member.2.ParameterValue=yes
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

有关更多信息，请参阅ModifyCacheParameterGroup。

如果您的集群为 Redis（已禁用集群模式）集群并对以下参数进行了更改，则您必须重启集群中的节点：

- activerehashing
- 数据库

有关更多信息，请参阅重启节点。

Redis（已启用集群模式）参数更改

如果要更改 Redis（已启用集群模式）集群上的以下参数，请按照随后的步骤操作。

- activerehashing
- 数据库

1. 创建集群的手动备份。请参阅进行手动备份。
2. 删除 Redis（已启用集群模式）集群。请参阅删除集群。
3. 使用修改的参数组和备份还原集群，以便为新集群创建种子。请参阅从备份还原到新缓存。

对其他参数的更改不需要执行此操作。

删除参数组

您可以使用 ElastiCache 控制台、AWS CLI 或 ElastiCache API 删除自定义参数组。

如果参数组与任何集群关联，则无法将其删除。也无法删除任一默认参数组。

删除参数组（控制台）

以下过程介绍了如何使用 ElastiCache 控制台删除参数组。

使用 ElastiCache 控制台删除参数组

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
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框来选择要删除的参数组。

   Delete 按钮将变为活动状态。
4. 选择 Delete (删除)。

   Delete Parameter Groups 确认屏幕随即出现。
5. 要删除参数组，请在 Delete Parameter Groups 确认屏幕上选择 Delete。

   要保留参数组，请选择 Cancel。

删除参数组 (AWS CLI)

要使用 AWS CLI 删除参数组，请使用命令 `delete-cache-parameter-group`。对于要删除的参数组，由 `--cache-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 myMem14 参数组。

Example

对于 Linux、macOS 或 Unix：

```
aws elasticache delete-cache-parameter-group \
    --cache-parameter-group-name myRed28
```

对于 Windows：

```
aws elasticache delete-cache-parameter-group ^
    --cache-parameter-group-name myRed28
```

有关更多信息，请参阅delete-cache-parameter-group。

删除参数组 (ElastiCache API)

要使用 ElastiCache API 删除参数组，请使用 DeleteCacheParameterGroup 操作。对于要删除的参数组，由 CacheParameterGroupName 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 myRed28 参数组。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DeleteCacheParameterGroup
   &CacheParameterGroupName=myRed28
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

有关更多信息，请参阅DeleteCacheParameterGroup。

Memcached 特定的参数

如果您没有为 Memcached 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。有关更多信息，请参阅创建参数组。

主题

Memcached 1.6.17 更改

从 Memcached 1.6.17 开始，我们不再支持以下管理命令：`lru_crawler`、`lru` 和 `slabs`。鉴于这些更改，您将无法在运行时系统中通过命令启用/禁用 `lru_crawler`。请通过修改您的自定义参数组来启用/禁用 `lru_crawler`。

Memcached 1.6.6 增加的参数

对于 Memcached 1.6.6，不支持任何附加参数。

参数组系列：memcached1.6

Memcached 1.5.10 参数更改

对于 Memcached 1.5.10，支持以下附加参数。

参数组系列：memcached1.5

名称	详细信息	描述
no_modern	默认值：1 类型：布尔值 可修改：是 允许的值：0,1 更改生效：启动时	用于禁用 slab_reassign 、slab_automove 、lru_crawler 、lru_maintainer 、maxconns_fast 命令的别名。No modern 还将 hash_algorithm 设置为 jenkins，并允许内联 ASCII VALUE。适用于 memcached 1.5 版和更高版本。要还原到 modern，您必须禁用此参数然后重新启动，这将自动启用 slab_reassign 、slab_automove 、lru_crawler 、lru_maintainer 和 maxconns_fast 。 **Note** 自 2021 年 8 月 20 日起，此参数的原定设置配置值已从 0 改为 1。2021 年 8 月 20 日之后，每个区域的新 ElastiCache 用户将自动获取更新的原定设置值。2021 年 8 月 20 日之前，各区域的现有 ElastiCache 用户需要手动修改其自定义参数组才能应用此新更改。
inline_ascii_resp	默认值：0 类型：布尔值	存储项中的 VALUE 响应的数字，最多使用 24 个字节。ASCII get 和 faster 集的速度较慢。

名称	详细信息	描述
	可修改：是 允许的值：0,1 更改生效：启动时	

对于 Memcached 1.5.10，删除了以下参数。

名称	详细信息	描述
expirezero_does_not_evict	默认值：0 类型：布尔值 可修改：是 允许的值：0,1 更改生效：启动时	在此版本中不再受支持。
modern	默认值：1 类型：布尔值 可修改：是（如果设置为 no_modern ，则需要重新启动） 允许的值：0,1 更改生效：启动时	在此版本中不再受支持。从此版本开始，默认情况下，每次启动或重新启动时都会启用 no-modern 。

Memcached 1.4.34 增加的参数

对于 Memcached 1.4.34，不支持任何附加参数。

参数组系列： memcached1.4

Memcached 1.4.33 增加的参数

对于 Memcached 1.4.33，支持以下附加参数。

参数组系列： memcached1.4

名称	详细信息	描述
modern	默认值：启用 类型：布尔值 可修改：是 更改生效：启动时	访问多项功能的别名。启用 modern 等同于启用以下命令并使用 murmur3 哈希算法：slab_reassign 、slab_automove 、lru_crawler 、lru_maintainer 、maxconns_fast 和 hash_algorithm=murmur3 。
watch	默认值：启用 类型：布尔值 可修改：是 更改生效：立即 在用户达到其 watcher_logbuf_size 和 worker_logbuf_size 限制时可以删除日志。	日志提取、移出或更改。例如，在用户启用 watch 时，出现 get、set、delete 或 update 的情况下可以查看日志。
idle_timeout	默认值：0（禁用） 类型：整数 可修改：是	在要求关闭客户端之前，允许客户端保持空闲的最短秒数。取值范围：0 到 86400。

名称	详细信息	描述
	更改生效：启动时	
track_sizes	默认值：禁用 类型：布尔值 可修改：是 更改生效：启动时	显示每个 slab 组已用的大小。 启用 track_sizes 可让您运行 stats sizes 而无需运行 stats sizes_enable ,
watcher_logbuf_size	默认值：256 (KB) 类型：整数 可修改：是 更改生效：启动时	watch 命令启用 Memcached 的流日志记录。但是，在移出、更改或提取速率足够高而导致了日志记录缓冲区满填满时，watch 可以删除日志。在这些情况下，用户可以增加缓冲区大小以减少日志丢失的可能性。
worker_logbuf_size	默认值：64 (KB) 类型：整数 可修改：是 更改生效：启动时	watch 命令启用 Memcached 的流日志记录。但是，在移出、更改或提取速率足够高而导致了日志记录缓冲区满填满时，watch 可以删除日志。在这些情况下，用户可以增加缓冲区大小以减少日志丢失的可能性。
slab_chunk_max	默认值：524288 (字节) 类型：整数 可修改：是 更改生效：启动时	指定 slab 的最大大小。设置较小的 slab 大小可以更有效地使用内存。大于 slab_chunk_max 的项目将拆分为多个 slab。

名称	详细信息	描述
`lru_crawler metadump [all|1|2|3]`	默认值：禁用 类型：布尔值 可修改：是 更改生效：立即	如果启用了 lru_crawler，则此命令会转储所有键。 `all|1|2|3` - 所有 slab，或者指定特定 slab 编号

Memcached 1.4.24 增加的参数

对于 Memcached 1.4.24，支持以下附加参数。

参数组系列： memcached1.4

名称	详细信息	描述
`disable_flush_all`	默认值：0（禁用） 类型：布尔值 可修改：是 更改生效：启动时	添加参数 (-F) 以禁用 flush_all。如果您再也不想在生产实例上运行完全刷新，则这样做会很有用。 值：0，1（当值为 0 时，用户可以执行 `flush_all` ）。
`hash_algorithm`	默认值：jenkins 类型：字符串 可修改：是 更改生效：启动时	要使用的哈希算法。允许的值：murmur3 和 jenkins。
`lru_crawler`	默认值：0（禁用） 类型：布尔值 可修改：是	清除已过期的项目的 Slab 类。此过程在后台运行，并且产生的影响很小。目前要求使用手动命令来启用网络爬取。

名称	详细信息	描述
	更改生效：重新启动后 **Note** 您可在运行时通过命令行临时启用 `lru_crawler` 。有关更多信息，请参阅“描述”列。	要临时启用网络爬取，请在命令行处运行 `lru_crawler enable`。 `lru_crawler 1,3,5` 对 Slab 类 1、3 和 5 进行网络爬取，以查找要添加到空闲列表的过期项目。 值：0，1 **Note** 在命令行处启用 `lru_crawler` 将启用爬网程序，直到在命令行处或下次重启时将其禁用。要永久性启用爬网程序，您必须修改参数值。有关更多信息，请参阅修改参数组。
`lru_maintainer`	默认值：0（禁用） 类型：布尔值 可修改：是 更改生效：启动时	当达到容量时对 LRU 之间的项目进行随机处理的后台线程。值：0，1。

名称	详细信息	描述
expirezero_does_not_evict	默认值：0（禁用） 类型：布尔值 可修改：是 更改生效：启动时	在与 lru_maintainer 一起使用时，使过期时间为 0 的项目不可收回。 **Warning** 这可以挤出内存以供其他可收回项目使用。 可以设置为忽略 lru_maintainer 。

Memcached 1.4.14 增加的参数

对于 Memcached 1.4.14，支持以下附加参数。

参数组系列： memcached1.4

Memcached 1.4.14 中添加的参数

名称	描述
config_max	ElastiCache 配置条目的最大数量。

名称	描述
config_size_max	配置条目的最大大小（单位：字节）。

名称	描述
hashpower_init	ElastiCache 哈希表的初始大小（以二次幂表示）。默认值为 16 (2^16) 或 65536 长度的密钥。

名称	描述
maxconns_fast	在达到最大连接限制时，请更改处理新连接请求的方式。如果将此参数设为 0（即零），新连接将被添加至缓冲区队列，并将等待直到其他连接已关闭。如果将参数设为 1，ElastiCache 将会发送一个错误到客户端，并且立即关闭连接。

名称	描述
slab_automove	调整 Slab 自动移动算法：如果将此参数设为 0（即零），自动移动算法将禁用。如果将参数设为 1，ElastiCache 会采用一种缓慢而保守的方法来自动移动 Slab。如果将参数设为 2，ElastiCache 会在出现移出情况时积极地移动 Slab。（建议不要使用此模式，测试用途除外）。

名称	描述
slab_reassign	启用或禁用 Slab 重新分配。如果将此参数设为 1，您可以使用“Slab 重新分配”命令来手动重新分配内存。

Memcached 1.4.5 支持的参数

参数组系列： memcached1.4

对于 Memcached 1.4.5，支持以下参数。

Memcached 1.4.5 中添加的参数

名称	详细信息	描述
backlog_queue_limit	默认值：1024 类型：整数 可修改：否	积压队列限制。
binding_protocol	默认值：auto 类型：字符串 可修改：是 更改生效：重新启动后	绑定协议。 允许的值为：ascii 和 auto。 有关修改 binding_protocol 的值的指南，请参阅修改参数组。
cas_disabled	默认值：0 (false) 类型：布尔值 可修改：是 更改生效：重新启动后	如果为 1 (true)，则检查和设置 (CAS) 操作将禁用，存储的项目消耗的字节将比启用 CAS 时消耗的字节少 8 字节。
chunk_size	默认值：48 类型：整数 可修改：是 更改生效：重新启动后	为最小项目的密钥、值和标志分配的最小空间量（以字节为单位）。
chunk_size_growth_factor	默认值：1.25 类型：浮点数 可修改：是 更改生效：重新启动后	控制各个连续 Memcached 区块的大小的增长系数；每个区块将比前一个区块大 chunk_size_growth_factor 倍。

名称	详细信息	描述
error_on_memory_exhausted	默认值：0 (false) 类型：布尔值 可修改：是 更改生效：重新启动后	如果 1（为真），当没有更多的内存用于存储项目时，Memcached 将返回一个错误，而非移出项目。
large_memory_pages	默认值：0 (false) 类型：布尔值 可修改：否	如果 1（为真），ElastiCache 会试图使用大内存页。
lock_down_paged_memory	默认值：0 (false) 类型：布尔值 可修改：否	如果 1（为真），ElastiCache 会锁定所有分页内存。
max_item_size	默认值：1048576 类型：整数 可修改：是 更改生效：重新启动后	可以存储在集群中的最大项目的大小（单位：字节）。
max_simultaneous_connections	默认值：65000 类型：整数 可修改：否	最大同时连接数。
maximize_core_file_limit	默认值：0 (false) 类型：布尔值 可修改: 更改生效：重新启动后	如果 1（为真），ElastiCache 会最大限度地提高核心文件限制。

名称	详细信息	描述
memcached_connections_overhead	默认值：100 类型：整数 可修改：是 更改生效：重新启动后	为 Memcached 连接和其他杂项开支预留的内存量。有关此参数的信息，请参阅Memcached 连接开销。
requests_per_event	默认值：20 类型：整数 可修改：否	每个事件请求获取给定连接的最大数量。此限制需要防止资源匮乏。

Memcached 连接开销

每个节点上可用于存储项目的内存计算方式为：此节点上的总可用内存（存储于 max_cache_memory 参数中）减去连接和其他开支所占用的内存（存储于 memcached_connections_overhead 参数中）。例如，cache.m1.small 类型的节点的 max_cache_memory 为 1300MB。memcached_connections_overhead 的默认值为 100MB，Memcached 进程可用于存储项目的内存则为 1200MB。

memcached_connections_overhead 参数的默认值满足大多数用例；然而，分配给连接开支的必需量会因多种因素（包括请求率、有效负载大小和连接数）而异。

您可以更改 memcached_connections_overhead 的值，以更好地满足您的应用程序的需求。例如，增大 memcached_connections_overhead 参数的值将减少用于存储项目的内存量，并为连接开销提供更大的缓冲区。减小 memcached_connections_overhead 参数的值将为您提供更多的内存来存储项目，但可能会增加使用交换分区和性能下降的风险。如果您发现交换分区使用情况和性能降低，请尝试增加 memcached_connections_overhead 参数的值。

> **Important**
>
> 对于 cache.t1.micro 节点类型，memcached_connections_overhead 的值是通过以下方式决定：
>
> - 如果您的集群使用的是默认参数组，那么 ElastiCache 会将 memcached_connections_overhead 的值设置为 13MB。

- 如果您的集群使用的是您自己创建的参数组，那么您可以将 memcached_connections_overhead 的值设置为您选定的值。

特定于 Memcached 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。下表显示了每种节点类型的 max_cache_memory 和 num_threads 参数的默认值。无法修改这些参数的值。

节点类型	max_cache_memory（单位：MB）	num_threads
cache.t1.micro	213	1
cache.t2.micro	555	1
cache.t2.small	1588	1
cache.t2.medium	3301	2
cache.t3.micro	512	2
cache.t3.small	1402	2
cache.t3.medium	3364	2
cache.t4g.micro	512	2
cache.t4g.small	1402	2
cache.t4g.medium	3164	2
cache.m1.small	1301	1
cache.m1.medium	3350	1
cache.m1.large	7100	2
cache.m1.xlarge	14600	4
cache.m2.xlarge	33800	2

节点类型	max_cache_memory（单位：MB）	num_threads
cache.m2.2xlarge	30412	4
cache.m2.4xlarge	68000	16
cache.m3.medium	2850	1
cache.m3.large	6200	2
cache.m3.xlarge	13600	4
cache.m3.2xlarge	28600	8
cache.m4.large	6573	2
cache.m4.xlarge	11496	4
cache.m4.2xlarge	30412	8
cache.m4.4xlarge	62234	16
cache.m4.10xlarge	158355	40
cache.m5.large	6537	2
cache.m5.xlarge	13248	4
cache.m5.2xlarge	26671	8
cache.m5.4xlarge	53516	16
cache.m5.12xlarge	160900	48
cache.m5.24xlarge	321865	96
cache.m6g.large	6537	2
cache.m6g.xlarge	13248	4
cache.m6g.2xlarge	26671	8

节点类型	max_cache_memory（单位：MB）	num_threads
cache.m6g.4xlarge	53516	16
cache.m6g.8xlarge	107000	32
cache.m6g.12xlarge	160900	48
cache.m6g.16xlarge	214577	64
cache.c1.xlarge	6600	8
cache.r3.large	13800	2
cache.r3.xlarge	29100	4
cache.r3.2xlarge	59600	8
cache.r3.4xlarge	120600	16
cache.r3.8xlarge	120600	32
cache.r4.large	12590	2
cache.r4.xlarge	25652	4
cache.r4.2xlarge	51686	8
cache.r4.4xlarge	103815	16
cache.r4.8xlarge	208144	32
cache.r4.16xlarge	416776	64
cache.r5.large	13387	2
cache.r5.xlarge	26953	4
cache.r5.2xlarge	54084	8
cache.r5.4xlarge	108347	16

节点类型	max_cache_memory（单位：MB）	num_threads
cache.r5.12xlarge	325400	48
cache.r5.24xlarge	650869	96
cache.r6g.large	13387	2
cache.r6g.xlarge	26953	4
cache.r6g.2xlarge	54084	8
cache.r6g.4xlarge	108347	16
cache.r6g.8xlarge	214577	32
cache.r6g.12xlarge	325400	48
cache.r6g.16xlarge	429154	64
cache.c7gn.large	3164	2
cache.c7gn.xlarge	6537	4
cache.c7gn.2xlarge	13248	8
cache.c7gn.4xlarge	26671	16
cache.c7gn.8xlarge	53516	32
cache.c7gn.12xlarge	325400	48
cache.c7gn.16xlarge	108347	64

Note

所有 T2 实例都是在 Amazon Virtual Private Cloud (Amazon VPC) 中创建的。

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。

主题

Redis 7 参数更改

参数组系列：redis7

Redis 7 默认参数组如下所示：

- `default.redis7` – 对 Redis（已禁用集群模式）集群和复制组使用此参数组或从中派生的参数组。
- `default.redis7.cluster.on` – 对 Redis（已启用集群模式）集群和复制组使用此参数组或从中派生的参数组。

Redis 7 中增加的参数如下所示。

名称	详细信息	描述
cluster-allow-pubsubshard-when-down	允许的值：yes、no 默认：yes 类型：字符串 可修改：是 更改生效：立即跨集群中的所有节点生效。	如果设置为默认值“yes”（是），则允许节点在集群处于停机状态时提供发布订阅分片流量，只要它认为自己拥有插槽即可。
cluster-preferred-endpoint-type	允许的值：ip、tls-dynamic 默认：tls-dynamic 类型：字符串 可修改：是 更改生效：立即跨集群中的所有节点生效。	此值控制 MOVED/ASKING 请求返回的端点以及 CLUSTER SLOTS 和 CLUSTER SHARDS 的端点字段。当该值设置为 ip 时，节点将传播其 IP 地址。当该值设置为 tls-dynamic 时，节点将在启用时通告主机名 encryption-in-transit ，否则通告一个 IP 地址。
latency-tracking	允许的值：yes、no 默认：no 类型：字符串 可修改：是 更改生效：立即跨集群中的所有节点生效。	设置为“yes”（是）时，将跟踪每个命令的延迟，并允许通过 INFO 延迟统计命令导出百分位数分布，并通过 LATENCY 命令导出累积延迟分布（直方图）。

名称	详细信息	描述
hash-max-listpack-entries	允许的值：0+ 默认：512 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	压缩数据集所需的最大哈希条目数。
hash-max-listpack-value	允许的值：0+ 默认：64 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	压缩数据集所需的最大哈希条目数的阈值。
zset-max-listpack-entries	允许的值：0+ 默认：128 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	压缩数据集所需的最大已排序集合条目数。

名称	详细信息	描述
zset-max-listpack-value	允许的值：0+ 默认：64 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	压缩数据集所需的最大已排序集合条目数的阈值。

Redis 7 中更改的参数如下所示。

名称	详细信息	描述
activerehashing	可修改：no。在 Redis 7 中，原定设置情况下，此参数处于隐藏状态并处于已启用状态。为了禁用此参数，您需要创建一个支持案例。	可修改：是。

Redis 7 中删除的参数如下所示。

名称	详细信息	描述
hash-max-ziplist-entries	允许的值：0+ 默认：512 类型：整数 可修改：是	使用 listpack 而非 ziplist 来表示小哈希编码

名称	详细信息	描述
	更改生效：立即跨集群中的所有节点生效。	
hash-max-ziplist-value	允许的值：0+ 默认：64 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	使用 listpack 而非 ziplist 来表示小哈希编码
zset-max-ziplist-entries	允许的值：0+ 默认：128 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	使用 listpack 而非 ziplist 来表示小哈希编码。
zset-max-ziplist-value	允许的值：0+ 默认：64 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	使用 listpack 而非 ziplist 来表示小哈希编码。

名称	详细信息	描述
list-max-ziplist-size	允许的值： 默认：-2 类型：整数 可修改：是 更改生效：立即跨集群中的所有节点生效。	每个内部列表节点允许的条目数。

Redis 6.x 参数更改

参数组系列：redis6.x

Redis 6.x 默认参数组如下所示：

- `default.redis6.x` – 对 Redis（已禁用集群模式）集群和复制组使用此参数组或从中派生的参数组。
- `default.redis6.x.cluster.on` – 对 Redis（已启用集群模式）集群和复制组使用此参数组或从中派生的参数组。

> **Note**
>
> 在 Redis 6.2 引擎版本中，在推出 r6gd 节点系列以支持数据分层功能时，r6gd 节点类型仅支持 noeviction、volatile-lru 和 allkeys-lru max-memory 策略。

有关更多信息，请参阅 ElastiCache for Redis 版本 6.2（加强版）和 ElastiCache for Redis 版本 6.0（加强版）。

Redis 6.x 中增加的参数如下所示。

名称	详细信息	描述
acl-pubsub-default (added in 6.2)	允许的值：resetchannels 、allchannels 默认：allchannels 类型：字符串 可修改：是 更改的生效范围：与集群关联的现有 Redis 用户将继续拥有现有的权限。需要更新用户或者重新启动集群以更新现有的 Redis 用户。	部署到此集群的 ACL 用户将默认拥有发布订阅频道权限。
cluster-allow-reads-when-down (added in 6.0)	默认值：no 类型：字符串 可修改：是 更改生效：立即跨集群中的所有节点生效	当设置为“yes（是）”时，Redis（已启用集群模式）复制组将继续处理读取命令，即使节点无法达到主节点的法定数量。 当设置为默认值“no（不）”时，复制组将拒绝所有命令。如果您使用的集群的节点组少于三个，或者您的应用程序可以安全地处理陈旧读取，我们建议将此值设置为“yes（是）”。
tracking-table-max-keys (added in 6.0)	默认值：1000000 类型：数字 可修改：是 更改生效：立即跨集群中的所有节点生效	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。超出此参数值后，将随机向客户端发送失效。应该调整此值以限制内存使用，同时仍对足够的密钥进行跟踪。在内存不足的情况下，密钥也会失效。
acllog-max-len	默认值：128	此值对应于 ACL 日志中的最大条目数。

名称	详细信息	描述
(added in 6.0)	类型：数字 可修改：是 更改生效：立即跨集群中的所有节点生效	
active-expire-effort (added in 6.0)	默认值：1 类型：数字 可修改：是 更改生效：立即跨集群中的所有节点生效	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 (added in 6.0)	默认值：no 类型：字符串 可修改：是 更改生效：立即跨集群中的所有节点生效	当该值设置为“yes（是）”时，DEL 命令的行为与 UNLINK 相同。

Redis 6.x 中删除的参数如下所示。

名称	详细信息	描述
lua-replicate-commands	允许的值：yes/no 默认值：yes 类型：布尔值 可修改：是 更改生效：立即	是否始终在 Lua 脚本中启用 Lua 效果复制

Redis 5.0.3 参数更改

参数组系列：redis5.0

Redis 5.0 默认参数组

- default.redis5.0 – 对 Redis（已禁用集群模式）集群和复制组使用此参数组或从中派生的参数组。
- default.redis5.0.cluster.on – 对 Redis（已启用集群模式）集群和复制组使用此参数组或从中派生的参数组。

Redis 5.0.3 中增加的参数

名称	详细信息	描述
rename-commands	默认：无 类型：字符串 可修改：是 更改生效：立即跨集群中的所有节点生效	重命名的 Redis 命令列表，以空格分隔。以下是可用于重命名的命令的限制列表： APPEND AUTH BITCOUNT BITFIELD BITOP BITPOS BLPOP BRPOP BRPOPLPUSH BZPOPMIN BZPOPMAX CLIENT CLUSTER COMMAND DBSIZE DECR DECRBY DEL DISCARD DUMP ECHO EVAL EVALSHA EXEC EXISTS EXPIRE EXPIREAT FLUSHALL FLUSHDB GEOADD GEOHASH GEOPOS GEODIST GEORADIUS

名称	详细信息	描述
		GEORADIUSBYMEMBER GET GETBIT GETRANGE GETSET HDEL HEXISTS HGET HGETALL HINCRBY HINCRBYFL OAT HKEYS HLEN HMGET HMSET HSET HSETNX HSTRLEN HVALS INCR INCRBY INCRBYFLOAT INFO KEYS LASTSAVE LINDEX LINSERT LLEN LPOP LPU SH LPUSHX LRANGE LREM LSET LTRIM MEMORY MGET MONITOR MOVE MSET MSETNX MULTI OBJECT PERSIST PEXPIRE PEXPIREAT PFADD PFCOUNT PFMERGE PING PSETEX PSUBSCRIBE PUBSUB PTTL PUBLISH PUNSUBSCRIBE RANDOMKEY READONLY READWRITE RENAME RENAMENX RESTORE ROLE RPOP RPOPLPUSH RPUSH RPUSHX SADD SCARD SCRIPT SDIFF SDIFFSTORE SELECT SET SETBIT SETEX SETNX SETRANGE SINTER SINTERSTORE SISMEMBER SLOWLOG SMEMBERS SMOVE SORT SPOP SRANDMEMBER SREM STRLEN SUBSCRIBE SUNION SUNIONSTORE SWAPDB TIME TOUCH TTL TYPE UNSUBSCRIBE UNLINK UNWATCH WAIT WATCH ZADD ZCARD ZCOUNT ZINCRBY ZINTERSTO RE ZLEXCOUNT ZPOPMAX ZPOPMIN ZRANGE ZRANGEBYLEX ZREVRANGE BYLEX ZRANGEBYSCORE ZRANK ZREM ZREMRANGEBYLEX ZREMRANGEBYRANK ZREMRANGEBYSCORE ZREVRANGE ZREVRANGEBYSCORE ZREVRANK ZSCORE ZUNIONSTORE SCAN SSCAN HSCAN ZSCAN XINFO XADD XTRIM XDEL XRA NGE XREVRANGE XLEN XREAD XGROUP

名称	详细信息	描述
		XREADGROUP XACK XCLAIM XPENDING GEORADIUS_RO GEORADIUSBYMEMBER_RO LOLWUT XSETID SUBSTR

有关更多信息，请参阅 ElastiCache for Redis 版本 5.0.6（加强版）。

Redis 5.0.0 参数更改

参数组系列：redis5.0

Redis 5.0 默认参数组

- default.redis5.0 – 对 Redis（已禁用集群模式）集群和复制组使用此参数组或从中派生的参数组。
- default.redis5.0.cluster.on – 对 Redis（已启用集群模式）集群和复制组使用此参数组或从中派生的参数组。

Redis 5.0 中增加的参数

名称	详细信息	描述
stream-node-max-bytes	允许的值：0+ 默认值：4096 类型：整数 可修改：是 更改生效：立即	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	允许的值：0+ 默认值：100 类型：整数	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

名称	详细信息	描述
	可修改：是 更改生效：立即	
active-defrag-max-scan-fields	允许的值：1 到 1000000 默认值：1000 类型：整数 可修改：是 更改生效：立即	将从主字典扫描中处理的最大 set/hash/zset/list 字段数
lua-replicate-commands	允许的值：yes/no 默认值：yes 类型：布尔值 可修改：是 更改生效：立即	是否始终在 Lua 脚本中启用 Lua 效果复制
replica-ignore-maxmemory	默认值：yes 类型：布尔值 可修改：否	确定副本是否通过不移出独立于主节点的项来忽略 maxmemory 设置

Redis 已在引擎版本 5.0 中重命名几个参数以响应社区反馈。有关更多信息，请参阅 Redis 5 中的新增功能。下表列出了新名称以及它们映射到早期版本的方式。

Redis 5.0 中已重命名的参数

名称	详细信息	描述
replica-lazy-flush	默认值：yes 类型：布尔值 可修改：否 以前的名字： slave-lazy-flush	在副本同步期间执行异步 flushDB。
client-output-buffer-limit-replica-hard-limit	默认值：有关值的信息，请参阅 特定于 Redis 节点类型的参数 类型：整数 可修改：否 以前的名称：client-output-buffer-limit-slave-hard-limit	对于 Redis 只读副本：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-replica-soft-limit	默认值：有关值的信息，请参阅 特定于 Redis 节点类型的参数 类型：整数 可修改：否 以前的名称：client-output-buffer-limit-slave-soft-limit	对于 Redis 只读副本：如果客户端的输出缓冲区达到指定字节数，仅当此条件保持 client-output-buffer-limit-replica-soft-seconds 时间时，客户端将断开连接。
client-output-buffer-limit-replica-soft-seconds	默认值：60 类型：整数 可修改：否	对于 Redis 只读副本：如果客户端的输出缓冲区保持 client-output-buffer-limit-replica-soft-limit 字节的时间长于此秒数，则客户端将断开连接。

名称	详细信息	描述
	以前的名称：client-output-buffer-limit-slave-soft-seconds	
replica-allow-chaining	默认值：no 类型：字符串 可修改：否 以前的名字： slave-allow-chaining	确定 Redis 中的只读副本是否可以具有自己的只读副本。
min-replicas-to-write	默认：0 类型：整数 可修改：是 以前的名字： min-slaves-to-write 更改生效：立即	使主节点可以从客户端接受写入所必需的可用只读副本的最小数目。如果可用副本数下降到低于此数字，则主节点不再接受写入请求。 如果此参数或 min-replicas-max-lag 为 0，则即使没有副本可用，主节点也将始终接受写入请求。
min-replicas-max-lag	默认值：10 类型：整数 可修改：是 以前的名字： min-slaves-max-lag 更改生效：立即	主节点必须从只读副本收到 Ping 请求的秒数。如果此时间量已过，但主节点未收到 Ping，则不再将副本视为可用。如果可用副本的数量降至以下 min-replicas-to-write，则主副本将在此时停止接受写入。 如果此参数或 min-replicas-to-write 为 0，则即使没有副本可用，主节点也将始终接受写入请求。

名称	详细信息	描述
close-on-replica-write	默认值：yes 类型：布尔值 可修改：是 以前的名字： close-on-slave-write 更改生效：立即	如果启用，尝试写入只读副本的客户端将会断开连接。

Redis 5.0 中已删除的参数

名称	详细信息	描述
repl-timeout	默认值：60 可修改：否	此版本中不提供参数。

Redis 4.0.10 参数更改

参数组系列： redis4.0

Redis 4.0.x 默认参数组

- default.redis4.0 – 对 Redis（已禁用集群模式）集群和复制组使用此参数组或从中派生的参数组。
- default.redis4.0.cluster.on – 对 Redis（已启用集群模式）集群和复制组使用此参数组或从中派生的参数组。

Redis 4.0.10 中已更改的参数

名称	详细信息	描述
maxmemory-policy		在 2.6.13 版中增加了 maxmemory-policy 。在版本 4.0.10 中增加了两个新允许的

名称	详细信息	描述
	允许的值：allkeys-lru 、volatile-lru 、**allkeys-lfu** 、**volatile-lfu** 、allkeys-random 、volatile-random 、volatile-ttl 、noeviction 默认值：volatile-lru 类型：字符串 可修改：是 发生更改：立即	值：allkeys-lfu （将使用近似的 LFU 移出任何键）；和 volatile-lfu （将使用近似的 LFU 移出具有过期设置的键）。在 6.2 版中，在推出 r6gd 节点系列以支持数据分层功能时，r6gd 节点类型仅支持 noeviction 、volatile-lru 和 allkeys-lru max-memory 策略

Redis 4.0.10 中增加的参数

名称	详细信息	描述
异步删除参数		
lazyfree-lazy-eviction	允许的值：yes/no 默认值：no 类型：布尔值 可修改：是 发生更改：立即	对移出执行异步删除。
lazyfree-lazy-expire	允许的值：yes/no 默认值：no	对已过期密钥执行异步删除。

名称	详细信息	描述
	类型：布尔值 可修改：是 发生更改：立即	
lazyfree-lazy-server-del	允许的值：yes/no 默认值：no 类型：布尔值 可修改：是 发生更改：立即	对更新值的命令执行异步删除。
slave-lazy-flush	允许的值：N/A 默认值：no 类型：布尔值 可修改：否 发生更改：N/A	在从属同步期间执行异步 flushDB。
LFU 参数		
lfu-log-factor	允许的值：任意整数 > 0 默认值：10 类型：整数 可修改：是 发生更改：立即	设置日志因素，这确定键命中数以使键计数器饱和。

名称	详细信息	描述
lfu-decay-time	允许的值：任意整数 默认值：1 类型：整数 可修改：是 发生更改：立即	减少键计数器的时间，以分钟为单位。
有效的碎片整理参数		
activedefrag	允许的值：yes/no 默认值：no 类型：布尔值 可修改：是 发生更改：立即	已启用有效的碎片整理。
active-defrag-ignore-bytes	允许的值：10485760 – 104857600 默认值：104857600 类型：整数 可修改：是 发生更改：立即	启动有效碎片整理的碎片垃圾最低量。

名称	详细信息	描述
active-defrag-threshold-lower	允许的值：1-100 默认值：10 类型：整数 可修改：是 发生更改：立即	启动有效碎片整理的碎片最低百分比。
active-defrag-threshold-upper	允许的值：1-100 默认值：100 类型：整数 可修改：是 发生更改：立即	我们使用最大精力的碎片最高百分比。
active-defrag-cycle-min	允许的值：1-75 默认值：25 类型：整数 可修改：是 发生更改：立即	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	描述
active-defrag-cycle-max	允许的值：1-75 默认值：75 类型：整数 可修改：是 发生更改：立即	用于碎片整理的最多精力，以 CPU 百分比为单位。
客户端输出缓冲区参数		
client-query-buffer-limit	允许的值：1048576 – 1073741824 默认值：1073741824 类型：整数 可修改：是 发生更改：立即	单个客户端查询缓冲区的最大大小。
proto-max-bulk-len	允许的值：1048576 – 536870912 默认值：536870912 类型：整数 可修改：是 发生更改：立即	单个元素请求的最大大小。

Redis 3.2.10 参数更改

参数组系列： redis3.2

ElastiCache 对于 Redis 3.2.10，不支持其他参数。

Redis 3.2.6 参数更改

参数组系列： redis3.2

对于 Redis 3.2.6，不支持附加参数。

Redis 3.2.4 参数更改

参数组系列： redis3.2

从 Redis 3.2.4 开始，提供两个默认参数组。

- `default.redis3.2` – 在运行 Redis 3.2.4 时，如果您需要创建 Redis（已禁用集群模式）复制组并仍使用 Redis 3.2.4 的附加功能，请指定此参数组或从中派生的参数组。
- `default.redis3.2.cluster.on` – 当您需要创建 Redis（已启用集群模式）复制组时，请指定此参数组或从中派生的参数组。

主题

- Redis 3.2.4 的新参数
- Redis 3.2.4 中已更改的参数（加强版）

Redis 3.2.4 的新参数

参数组系列： redis3.2

对于 Redis 3.2.4，支持以下附加参数。

名称	详细信息	描述
`list-max-ziplist-size`	默认值：-2 类型：整数 可修改：否	将采用特殊方式对列表进行编码以节省空间。可将每个内部列表节点允许的条目数指定为固定的最大大小或最大元素数。对于固定最大大小，请使用数字 -5 到 -1，含义如下： • -5：最大大小：64 Kb - 对于一般工作负载，不推荐使用

名称	详细信息	描述
		• -4：最大大小：32 Kb - 不推荐 • -3：最大大小：16 Kb - 不推荐 • -2：最大大小：8 Kb - 推荐 • -1：最大大小：4 Kb - 推荐 • 正数表示每节点存储最多该数目的元素。
list-compress-depth	默认：0 类型：整数 可修改：是 更改生效：立即	也可以压缩列表。压缩深度是要从压缩中排除的列表各端的 quicklist ziplist 节点的数目。始终不会压缩列表的首尾以便执行快速推送和弹出操作。设置为： • 0：禁止所有压缩。 • 1：从首尾开始将第一个节点压缩到列表中。 [head]->node->node->...->node->[tail] 压缩除 [head] 和 [tail] 以外的所有节点。 • 2：从首尾开始将第二个节点压缩到列表中。 [head]->[next]->node->node->...->node->[prev]->[tail] [head]、[next]、[prev]、[tail] 不压缩。压缩所有其他节点。 • 等等

名称	详细信息	描述
cluster-enabled	默认值：否/是 * 类型：字符串 可修改：否	指示这是处于集群模式 [“yes（是）”] 的 Redis（已启用集群模式）复制组，还是处于非集群模式 [“no（否）”] 的 Redis（已启用集群模式）复制组。集群模式下的 Redis（已启用集群模式）复制组可跨最多 500 个节点组对数据进行分区。 * Redis 3.2.x 具有两个默认参数组。 • default.redis3.2 – 默认值 no。 • default.redis3.2.cluster.on – 默认值 yes。 .
cluster-require-full-coverage	默认值：no 类型：布尔值 可修改：是 更改生效：立即	如果设为 yes，集群模式下的 Redis（已启用集群模式）节点会在检测到至少一个未覆盖的哈希槽（没有节点为其提供服务）时停止接受查询。因此，如果集群部分出现故障，则整个集群将不可用。只要所有槽全都覆盖到，它就会自动恢复可用。 但有时，您需要让集群的子集能够继续接受仍覆盖到的键空间部分的查询。为此，请将 cluster-require-full-coverage 选项设为 no。

名称	详细信息	描述
hll-sparse-max-bytes	默认值：3000 类型：整数 可修改：是 更改生效：立即	HyperLogLog 稀疏表示字节限制。限制包括 16 个字节的标头。当 HyperLogLog 使用稀疏表示超过此限制时，它将转换为密集表示。 不建议使用超过 16000 的值，因为此时密集表现形式具有更高的内存效率。 我们建议使用 3000 左右的值来获得空间效率较高的编码，同时不会过多地降低 PFADD 效率，即稀疏编码的复杂度为 O(N)。如果不考虑CPU，而是空间，并且数据集由许多 HyperLogLogs 基数介于 0-15000 范围内的数据集组成，则可以将该值提高到大约 10000。
reserved-memory-percent	默认值：25 类型：整数 可修改：是 更改生效：立即	为非数据使用预留的节点内存的百分比。默认情况下，Redis 数据占用会一直增长，直至消耗掉节点的所有内存。如果发生这种情况，可能会因内存分页过多而影响节点性能。通过预留内存，您可以为非 Redis 用途留出一些可用内存，以帮助减少分页量。 此参数特定于标准 Redis 发行版 ElastiCache，不属于标准 Redis 发行版的一部分。 有关更多信息，请参阅 reserved-memory 和管理预留内存。

Redis 3.2.4 中已更改的参数（加强版）

参数组系列： redis3.2

对于 Redis 3.2.4，以下参数已更改。

名称	详细信息	更改
activerehashing	可修改：如果参数组未与任何缓存群集关联，则为“是”。否则不是必需。	可修改：否。
databases	可修改：如果参数组未与任何缓存群集关联，则为“是”。否则不是必需。	可修改：否。
appendonly	默认：关闭 可修改：否	如果您需要从早期版本的 Redis 升级，则必须先禁用 appendonly 。
appendfsync	默认：关闭 可修改：否	如果您需要从早期版本的 Redis 升级，则必须先禁用 appendfsync 。
repl-timeout	默认值：60 可修改：否	目前无法修改，默认值为 60。
tcp-keepalive	默认：300	默认值为 0。
list-max-ziplist-entries		参数不再可用。
list-max-ziplist-value		参数不再可用。

Redis 2.8.24（加强版）增加的参数

参数组系列： redis2.8

对于 Redis 2.8.24，不支持附加参数。

Redis 2.8.23（加强版）增加的参数

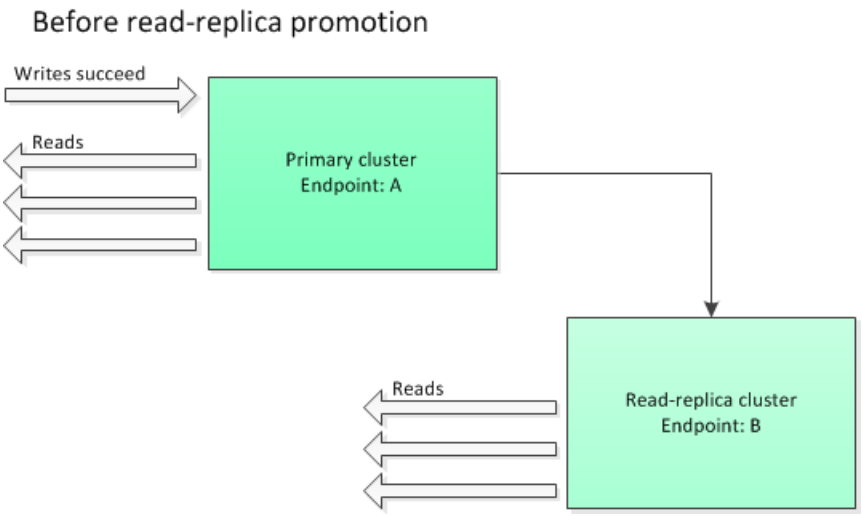
参数组系列： redis2.8

对于 Redis 2.8.23，支持以下附加参数。

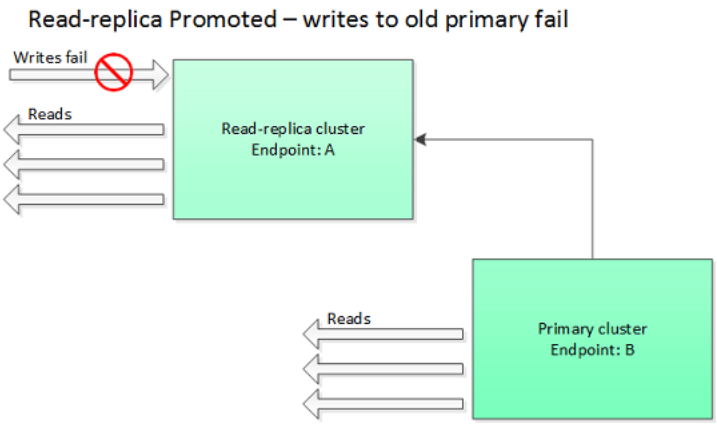
名称	详细信息	描述
`close-on-slave-write`	默认值：yes 类型：字符串（yes/no） 可修改：是 更改生效：立即	如果启用，尝试写入只读副本的客户端将会断开连接。

如何 close-on-slave-write 运作

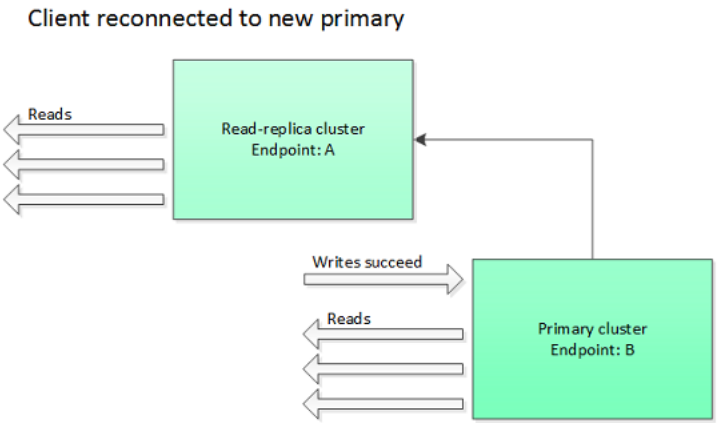
该`close-on-slave-write`参数由 Amazon ElastiCache 引入，可让您更好地控制集群在主节点和只读副本节点因将只读副本提升为主节点而交换角色时群集的响应方式。



如果出于支持多可用区的复制组发生故障转移之外的任何其他原因，只读副本集群提升为主集群，则客户端将继续尝试写入端点 A。由于端点现在是只读副本的端点，这些写入将会失败。这是 Redis 在 ElastiCache 引入之前的行为，`close-on-replica-write`以及禁`close-on-replica-write`用后的行为。



在启用 close-on-replica-write 的情况下，只要客户端尝试写入只读副本，客户端与集群的连接就会被关闭。您的应用程序逻辑应检测断开连接情况，检查 DNS 表，然后重新连接到主端点（现在是端点 B）。



何时可以禁用 close-on-replica-write

如果禁用 close-on-replica-write 会导致写入集群的操作失败，那么为什么禁用 close-on-replica-write？

如前所述，在启用 close-on-replica-write 的情况下，只要客户端尝试写入只读副本，客户端与集群的连接就会被关闭。建立与节点的新连接需要时间。因此，由于对副本的写入请求而断开连接并重新连接也会影响通过相同连接处理的读取请求的延迟。在建立新连接之前，此影响会持续存在。如果您的应用程序主要执行大量读取操作或者对延迟非常敏感，您可能使客户端保持连接，以避免降低读取性能。

Redis 2.8.22（加强版）增加的参数

参数组系列： redis2.8

对于 Redis 2.8.22，不支持附加参数。

> **Important**
>
> - 从 Redis 版本 2.8.22 开始，`repl-backlog-size` 应用于主集群以及副本集群。
> - 从 Redis 版本 2.8.22 开始，不支持 `repl-timeout` 参数。如果它被更改， ElastiCache 将像我们一样用默认值（60 秒）覆盖。`appendonly`

不再支持以下参数。

- appendonly
- appendfsync
- repl-timeout

Redis 2.8.21 增加的参数

参数组系列： redis2.8

Redis 2.8.21 没有增加支持的参数。

Redis 2.8.19 增加的参数

参数组系列： redis2.8

对于 Redis 2.8.19，不支持附加参数。

Redis 2.8.6 增加的参数

参数组系列： redis2.8

对于 Redis 2.8.6，支持以下附加参数。

名称	详细信息	描述
`min-slaves-max-lag`	默认值：10 类型：整数 可修改：是 更改生效：立即	主节点必须从只读副本收到 Ping 请求的秒数。如果此时间量已过，但主节点未收到 Ping，则不再将副本视为可用。如果可用副本的数量降至以下 min-slaves-to-write，则主副本将在此时停止接受写入。

名称	详细信息	描述
		如果此参数或 min-slaves-to-write 为 0，则即使没有副本可用，主节点也将始终接受写入请求。
min-slaves-to-write	默认：0 类型：整数 可修改：是 更改生效：立即	使主节点可以从客户端接受写入所必需的可用只读副本的最小数目。如果可用副本数下降到低于此数字，则主节点不再接受写入请求。 如果此参数或 min-slaves-max-lag 为 0，则即使没有副本可用，主节点也将始终接受写入请求。

名称	详细信息	描述
notify-keyspace-events	默认值：（空字符串） 类型：字符串 可修改：是 更改生效：立即	Redis 可以向客户端通知的键空间事件类型。每种事件类型由单个字母表示： • K – 键空间事件，发布时带有前缀 __keyspace@<db>__ • E – 键事件类型事件，发布时带有前缀 __keyevent@<db>__ • g – 通用的非特定命令，如 DEL、EXPIRE、RENAME 等。 • $ – 字符串命令 • l – 列表命令 • s – 集命令 • h – 哈希命令 • z – 排序集命令 • x – 过期事件（每次键过期时生成的事件） • e – 移出事件（针对 maxmemory 移出键时生成的事件） • A – g$lshzxe 的别名 您可以使用这些事件类型的任何组合。例如，AKE 表示 Redis 可以发布所有事件类型的通知。

名称	详细信息	描述
		请勿使用上面未列出的任何字符；尝试这样操作会导致错误消息。 默认情况下，此参数设置为空字符串，这表示禁用了键空间事件通知。
repl-backlog-size	默认值：1048576 类型：整数 可修改：是 更改生效：立即	主节点积压缓冲区的大小（以字节为单位）。积压用于记录主节点上数据的更新。只读副本连接到主节点时，它尝试执行部分同步（psync），其中它应用积压中的数据以与主节点同步。如果 psync 失败，则需要完整同步。 此参数的最小值为 16384。 **Note** 从 Redis 2.8.22 开始，此参数应用于主集群以及只读副本。
repl-backlog-ttl	默认值：3600 类型：整数 可修改：是 更改生效：立即	主节点保留积压缓冲区的秒数。从上一个副本节点断开连接时开始，积压中的数据将保持不变，直到 repl-backlog-ttl 过期。如果副本未在此时间内连接到主节点，则主节点会释放积压缓冲区。当副本最终重新连接时，它必须执行与主节点的完整同步。 如果此参数设置为 0，则永不释放积压缓冲区。

名称	详细信息	描述
repl-timeout	默认值：60 类型：整数 可修改：是 更改生效：立即	表示超时期限（以秒为单位），适用于： • 同步过程中的批量数据传输（从只读副本的角度） • 主节点超时（从副本的角度） • 副本超时（从主节点的角度）

Redis 2.6.13 参数

参数组系列： redis2.6

Redis 2.6.13 是第一个支持的 Redis 版本。 ElastiCache下表显示了支持的 Redis 2.6.13 参数。 ElastiCache

名称	详细信息	描述
activerehashing	默认值：yes 类型：字符串（yes/no） 可修改：是 发生更改：在创建时	确定是否启用 Redis 的活动重新哈希功能。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。
appendonly	默认值：no 类型：字符串 可修改：是 更改生效：立即	启用或禁用 Redis 的仅附加文件功能（AOF）。AOF 捕获在缓存中更改数据的任何 Redis 命令，并可用于从特定节点故障中恢复。 默认值为 no，这表示 AOF 已关闭。将此参数设置为 yes 可启用 AOF。 有关更多信息，请参阅 缓解故障。

名称	详细信息	描述
		Note cache.t1.micro 和 cache.t2.* 节点不支持仅附加文件（AOF）。对于此类型的节点，将忽略 appendonly 参数值。 **Note** 对于多可用区复制组，不允许使用 AOF。
appendfsync	默认值：everysec 类型：字符串 可修改：是 更改生效：立即	当 appendonly 设置为是时，控制将 AOF 输出缓冲区写入磁盘的频率： • no – 缓冲区根据需要刷新到磁盘中。 • everysec – 缓冲区每秒刷新一次。这是默认模式。 • always – 每当集群中的数据有修改时，缓冲区便会进行刷新。 • 2.8.22 及更高版本不支持 Appendfsync。
client-output-buffer-limit-normal-hard-limit	默认：0 类型：整数 可修改：是 更改生效：立即	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）。

名称	详细信息	描述
client-output-buffer-limit-normal-soft-limit	默认：0 类型：整数 可修改：是 更改生效：立即	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	默认：0 类型：整数 可修改：是 更改生效：立即	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）。
client-output-buffer-limit-pubsub-hard-limit	默认值：33554432 类型：整数 可修改：是 更改生效：立即	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	默认值：8388608 类型：整数 可修改：是 更改生效：立即	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，仅当此条件保持 client-output-buffer-limit-pubsub-soft-seconds 时间时，客户端将断开连接。
client-output-buffer-limit-pubsub-soft-seconds	默认值：60 类型：整数 可修改：是 更改生效：立即	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。

名称	详细信息	描述
client-output-buffer-limit-slave-hard-limit	默认值：有关值的信息，请参阅 特定于 Redis 节点类型的参数 类型：整数 可修改：否	对于 Redis 只读副本：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-slave-soft-limit	默认值：有关值的信息，请参阅 特定于 Redis 节点类型的参数 类型：整数 可修改：否	对于 Redis 只读副本：如果客户端的输出缓冲区达到指定字节数，仅当此条件保持 client-output-buffer-limit-slave-soft-seconds 时间时，客户端将断开连接。
client-output-buffer-limit-slave-soft-seconds	默认值：60 类型：整数 可修改：否	对于 Redis 只读副本：如果客户端的输出缓冲区保持 client-output-buffer-limit-slave-soft-limit 字节的时间长于此秒数，则客户端将断开连接。
databases	默认值：16 类型：整数 可修改：否 发生更改：在创建时	数据库拆分到的逻辑分区的数量。建议将此值保持较低。 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。
hash-max-ziplist-entries	默认值：512 类型：整数 可修改：是 更改生效：立即	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	描述
hash-max-ziplist-value	默认值：64 类型：整数 可修改：是 更改生效：立即	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。
list-max-ziplist-entries	默认值：512 类型：整数 可修改：是 更改生效：立即	确定用于列表的内存量。条目少于指定数量的列表使用节省空间的特殊编码进行存储。
list-max-ziplist-value	默认值：64 类型：整数 可修改：是 更改生效：立即	确定用于列表的内存量。条目小于指定字节数的列表使用节省空间的特殊编码进行存储。
lua-time-limit	默认值：5000 类型：整数 可修改：否	Lua 脚本在采取行动停止脚本之前 ElastiCache 的最大执行时间，以毫秒为单位。 如果超过 lua-time-limit ，则所有 Redis 命令都会返回形式为 ____-BUSY 的错误。由于这种状态可能会干扰许多基本的 Redis 操作，因此 ElastiCache 将首先发出 SCR IPT KILL 命令。如果不成功， ElastiCache 将强制重启 Redis。
maxclients 除明确指定的外，此值适用于所有实例类型	默认值：65000 类型：整数 可修改：否	可以一次连接的最大客户端连接数。

名称	详细信息	描述
	t2.medium 原定设置：20000 类型：整数 可修改：否	
	t2.small 原定设置：20000 类型：整数 可修改：否	
	t2.micro 原定设置：20000 类型：整数 可修改：否	
	t4g.micro 原定设置：20000 类型：整数 可修改：否	
	t3.medium 原定设置：46000 类型：整数 可修改：否	
	t3.small 原定设置：46000 类型：整数 可修改：否	
	t3.micro 原定设置：20000 类型：整数 可修改：否	

名称	详细信息	描述
maxmemory-policy	默认值：volatile-lru 类型：字符串 可修改：是 更改生效：立即	达到最大内存使用率时密钥的移出策略。 有效值为：volatile-lru | allkeys-lru | volatile-random | allkeys-random | volatile-ttl | noeviction 有关更多信息，请参阅将 Redis 用作 LRU 缓存。
maxmemory-samples	原定设置值：3 类型：整数 可修改：是 更改生效：立即	对于 least-recently-used (LRU) 和 time-to-live (TTL) 计算，此参数表示要检查的密钥的样本量。默认情况下，Redis 选择 3 个密钥并使用最近最少使用的一个密钥。
reserved-memory	默认：0 类型：整数 可修改：是 更改生效：立即	为非数据使用预留的内存总量，以字节为单位。默认情况下，Redis 节点将增长，直到它使用了节点的 maxmemory （请参阅特定于 Redis 节点类型的参数）。如果发生这种情况，可能会因内存分页过多而影响节点性能。通过预留内存，您可以为非 Redis 用途留出一些可用内存，以帮助减少分页量。 此参数特定于标准 Redis 发行版 ElastiCache，不属于标准 Redis 发行版的一部分。 有关更多信息，请参阅 reserved-memory-percent 和 管理预留内存。

名称	详细信息	描述
set-max-intset-entries	默认值：512 类型：整数 可修改：是 更改生效：立即	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。
slave-allow-chaining	默认值：no 类型：字符串 可修改：否	确定 Redis 中的只读副本是否可以具有自己的只读副本。
slowlog-log-slower-than	默认值：10000 类型：整数 可修改：是 更改生效：立即	Redis 慢速日志功能记录的命令的最大执行时间（单位：微秒）。
slowlog-max-len	默认值：128 类型：整数 可修改：是 更改生效：立即	Redis 慢速日志的最大长度。
tcp-keepalive	默认：0 类型：整数 可修改：是 更改生效：立即	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。 **Important** 此参数的某些方面在 Redis 版本 3.2.4 中有所更改。请参阅 Redis 3.2.4 中已更改的参数（加强版）。

名称	详细信息	描述
timeout	默认：0 类型：整数 可修改：是 更改生效：立即	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。
zset-max-ziplist-entries	默认值：128 类型：整数 可修改：是 更改生效：立即	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	默认值：64 类型：整数 可修改：是 更改生效：立即	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。

> **Note**
>
> 如果您没有为 Redis 2.6.13 集群指定参数组，则将使用默认的参数组（default.redis2.6）。您不能在默认的参数组中更改任何参数的值；但是，您始终可以创建自定义参数组，然后随时将其分配给您的集群。

特定于 Redis 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。下表显示了每种节点类型的 maxmemory、client-output-buffer-limit-slave-hard-limit 和 client-output-

buffer-limit-slave-soft-limit 参数的默认值。maxmemory 的值是节点上可供您使用（数据和其他用途）的最大字节数。有关更多信息，请参阅可用内存。

> **Note**
>
> 无法修改 maxmemory 参数。

节点类型	Maxmemory	Client-output-buffer-limit-slave-hard-limit	Client-output-buffer-limit-slave-soft-limit
cache.t1.micro	142606336	14260633	14260633
cache.t2.micro	581959680	58195968	58195968
cache.t2.small	1665138688	166513868	166513868
cache.t2.medium	3461349376	346134937	346134937
cache.t3.micro	536870912	53687091	53687091
cache.t3.small	1471026299	147102629	147102629
cache.t3.medium	3317862236	331786223	331786223
cache.t4g.micro	536870912	53687091	53687091
cache.t4g.small	1471026299	147102629	147102629
cache.t4g.medium	3317862236	331786223	331786223
cache.m1.small	943718400	94371840	94371840
cache.m1.medium	3093299200	309329920	309329920
cache.m1.large	7025459200	702545920	702545920
cache.m1.xlarge	14889779200	1488977920	1488977920
cache.m2.xlarge	17091788800	1709178880	1709178880

节点类型	Maxmemory	Client-output-buffer-limit-slave-hard-limit	Client-output-buffer-limit-slave-soft-limit
cache.m2.2xlarge	35022438400	3502243840	3502243840
cache.m2.4xlarge	70883737600	7088373760	7088373760
cache.m3.medium	2988441600	309329920	309329920
cache.m3.large	6501171200	650117120	650117120
cache.m3.xlarge	14260633600	1426063360	1426063360
cache.m3.2xlarge	29989273600	2998927360	2998927360
cache.m4.large	6892593152	689259315	689259315
cache.m4.xlarge	15328501760	1532850176	1532850176
cache.m4.2xlarge	31889126359	3188912636	3188912636
cache.m4.4xlarge	65257290629	6525729063	6525729063
cache.m4.10xlarge	166047614239	16604761424	16604761424
cache.m5.large	6854542746	685454275	685454275
cache.m5.xlarge	13891921715	1389192172	1389192172
cache.m5.2xlarge	27966669210	2796666921	2796666921
cache.m5.4xlarge	56116178125	5611617812	5611617812
cache.m5.12xlarge	168715971994	16871597199	16871597199
cache.m5.24xlarge	337500562842	33750056284	33750056284
cache.m6g.large	6854542746	685454275	685454275
cache.m6g.xlarge	13891921715	1389192172	1389192172

节点类型	Maxmemory	Client-output-buffer-limit-slave-hard-limit	Client-output-buffer-limit-slave-soft-limit
cache.m6g.2xlarge	27966669210	2796666921	2796666921
cache.m6g.4xlarge	56116178125	5611617812	5611617812
cache.m6g.8xlarge	111325552312	11132555231	11132555231
cache.m6g.12xlarge	168715971994	16871597199	16871597199
cache.m6g.16xlarge	225000375228	22500037523	22500037523
cache.c1.xlarge	6501171200	650117120	650117120
cache.r3.large	14470348800	1468006400	1468006400
cache.r3.xlarge	30513561600	3040870400	3040870400
cache.r3.2xlarge	62495129600	6081740800	6081740800
cache.r3.4xlarge	126458265600	12268339200	12268339200
cache.r3.8xlarge	254384537600	24536678400	24536678400
cache.r4.large	13201781556	1320178155	1320178155
cache.r4.xlarge	26898228839	2689822883	2689822883
cache.r4.2xlarge	54197537997	5419753799	5419753799
cache.r4.4xlarge	108858546586	10885854658	10885854658
cache.r4.8xlarge	218255432090	21825543209	21825543209
cache.r4.16xlarge	437021573120	43702157312	43702157312
cache.r5.large	14037181030	1403718103	1403718103
cache.r5.xlarge	28261849702	2826184970	2826184970

节点类型	Maxmemory	Client-output-buffer-limit-slave-hard-limit	Client-output-buffer-limit-slave-soft-limit
cache.r5.2xlarge	56711183565	5671118356	5671118356
cache.r5.4xlarge	113609865216	11360986522	11360986522
cache.r5.12xlarge	341206346547	34120634655	34120634655
cache.r5.24xlarge	682485973811	68248597381	68248597381
cache.r6g.large	14037181030	1403718103	1403718103
cache.r6g.xlarge	28261849702	2826184970	2826184970
cache.r6g.2xlarge	56711183565	5671118356	5671118356
cache.r6g.4xlarge	113609865216	11360986522	11360986522
cache.r6g.8xlarge	225000375228	22500037523	22500037523
cache.r6g.12xlarge	341206346547	34120634655	34120634655
cache.r6g.16xlarge	450000750456	45000075046	45000075046
cache.r6gd.xlarge	28261849702	2826184970	2826184970
cache.r6gd.2xlarge	56711183565	5671118356	5671118356
cache.r6gd.4xlarge	113609865216	11360986522	11360986522
cache.r6gd.8xlarge	225000375228	22500037523	22500037523
cache.r6gd.12xlarge	341206346547	34120634655	34120634655
cache.r6gd.16xlarge	450000750456	45000075046	45000075046
cache.r7g.large	14037181030	1403718103	1403718103
cache.r7g.xlarge	28261849702	2826184970	2826184970

节点类型	Maxmemory	Client-output-buffer-limit-slave-hard-limit	Client-output-buffer-limit-slave-soft-limit
cache.r7g.2xlarge	56711183565	5671118356	5671118356
cache.r7g.4xlarge	113609865216	11360986522	11360986522
cache.r7g.8xlarge	225000375228	22500037523	22500037523
cache.r7g.12xlarge	341206346547	34120634655	34120634655
cache.r7g.16xlarge	450000750456	45000075046	45000075046
cache.m7g.large	6854542746	685454275	685454275
cache.m7g.xlarge	13891921715	1389192172	1389192172
cache.m7g.2xlarge	27966669210	2796666921	2796666921
cache.m7g.4xlarge	56116178125	5611617812	5611617812
cache.m7g.8xlarge	111325552312	11132555231	11132555231
cache.m7g.12xlarge	168715971994	16871597199	16871597199
cache.m7g.16xlarge	225000375228	22500037523	22500037523
cache.c7gn.large	3317862236	1403718103	1403718103
cache.c7gn.xlarge	6854542746	2826184970	2826184970
cache.c7gn.2xlarge	13891921715	5671118356	5671118356
cache.c7gn.4xlarge	27966669210	11360986522	11360986522
cache.c7gn.8xlarge	56116178125	22500037523	22500037523
cache.c7gn.12xlarge	84357985997	34120634655	34120634655
cache.c7gn.16xlarge	113609865216	45000075046	45000075046

> **Note**
>
> 默认情况下，所有最新一代实例类型将在 Amazon Virtual Private Cloud VPC 中创建。
> T1 实例不支持多可用区。
> T1 和 T2 实例不支持 Redis AOF。
> Redis 版本 2.8.22 及更高版本不支持 Redis 配置变量 `appendonly` 和 `appendfsync`。

针对 Redis ElastiCache 进行扩展

扩展 ElastiCache 无服务器

ElastiCache 当您的工作负载流量上升或下降时，Serverless 会自动适应您的工作负载流量。对于每个 ElastiCache 无服务器缓存， ElastiCache 持续跟踪 CPU、内存和网络等资源的利用率。当这些资源中的任何一个受到限制时， ElastiCache Serverless会通过添加新分片并将数据重新分配到新分片来进行扩展，而不会使您的应用程序停机。您可以通过监控缓存数据存储和计算使用率 `ElastiCacheProcessingUnits` (ECPU) `BytesUsedForCache` 指标 CloudWatch 来监控缓存消耗的资源。

设置扩展限制以管理成本

您可以选择为缓存数据存储和缓存的每秒 ECPU 数配置最大使用量，以控制缓存成本。这样做可以确保缓存使用量永远不会超过配置的最大值。

如果您设置了扩展最大值，则当缓存达到最大值时，应用程序的缓存性能可能会降低。当您设置了最大缓存数据存储空间且缓存数据存储空间达到最大值时， ElastiCache 将开始使用 LRU 逻辑逐出缓存中设置了存活时间 (TTL) 的数据。如果没有可以驱逐的数据，则写入更多数据的请求将收到内存不足（OOM）错误消息。当您设置了 ecpu/秒的最大值，并且工作负载的计算利用率超过此值时， ElastiCache 将开始限制 Redis 请求。

如果您将最大限制设置为`BytesUsedForCache`或`ElastiCacheProcessingUnits`，我们强烈建议将CloudWatch 警报设置为低于最大限制的值，以便在缓存接近这些限制时收到通知。我们建议将警报设置为所设最大值限制的 75%。请参阅有关如何设置 CloudWatch 闹钟的文档。

使用 ElastiCache 无服务器进行预扩展

ElastiCache 无服务器预扩展

使用预缩放（也称为预热），您可以为缓存设置支持的最低限制。 ElastiCache 您可以为每秒 ElastiCache 处理单元 (ECPU) 或数据存储设置这些最小值。这在为预期的扩展事件做准备时非常有用。例如，如果一家游戏公司预计在其新游戏发布的第一分钟内登录量将增加5倍，那么他们就可以为使用量的大幅激增做好缓存准备。

您可以使用 ElastiCache 控制台、CLI 或 API 进行预扩展。 ElastiCache Serverless 会在 60 分钟内更新缓存中可用的 ECPU /秒，并在最低限制更新完成后发送事件通知。

预缩放的工作原理

通过控制台、CLI 或 API 更新 ECPU /秒或数据存储的最低限制后，新的限制将在 1 小时内生效。 ElastiCache Serverless 在空缓存上支持 30K ecpus/秒，使用从副本读取功能时支持高达 90K ecpus/秒。 ElastiCache 每 10-12 分钟可以使每秒 ecpu 翻一番。这种扩展速度足以满足大多数工作负载的需求。如果您预计即将到来的扩展事件可能会超过此速率，那么我们建议将最低 ECPU /秒设置为峰值事件发生前至少 60 分钟您预计的峰值 ECPU/秒。否则，应用程序可能会遇到延迟增加和请求受限的情况。

最低限制更新完成后， ElastiCache Serverless 将开始计量新的每秒 ECPU 最小值或新的最低存储空间。即使您的应用程序没有在缓存上执行请求，或者您的数据存储使用量低于最低限度，也会发生这种情况。当您从当前设置中降低最低限制时，更新会立即生效，因此 ElastiCache Serverless 将立即以新的最低限制开始计量。

> **Note**
>
> - 当您设置最低使用限制时，即使您的实际使用量低于最低使用限制，也会按该限制收费。超出最低使用限制的 ECPU 或数据存储使用量按常规费率收费。例如，如果您将最低使用限制设置为 100,000 ecpu/秒，那么即使您的使用量低于设定的最低限额，也将按每小时至少 1.224 美元的费用（使用 us-east-1 中的 ECPU 价格）。
> - ElastiCache Serverless 在缓存的聚合级别上支持请求的最小缩放比例。 ElastiCache Serverless 还支持每个插槽最多 30K ECPU /秒（使用只读连接使用从副本读取时，每秒支持 90K ECPU）。作为最佳实践，您的应用程序应确保跨Redis插槽的密钥分布和密钥之间的流量尽可能均匀。

使用控制台设置缩放限制 AWS CLI

使用 AWS 控制台设置缩放限制

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格中，选择在要修改的缓存上运行的引擎。
3. 此时会显示运行所选引擎的缓存的列表。
4. 选择缓存名称左侧的单选按钮来选择要修改的缓存。
5. 选择 Actions（操作），然后选择 Modify（修改）。
6. 在 “使用限制” 下，设置相应的内存或计算限制。
7. 单击预览更改，然后保存更改。

使用设置缩放限制 AWS CLI

要使用 CLI 更改扩展限制，请使用 modify-serverless-cache API。

Linux：

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> \
--cache-usage-limits 'DataStorage={Minimum=10,Maximum=100,Unit=GB},
 ECPUPerSecond={Minimum=1000,Maximum=100000}'
```

Windows:

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> ^
--cache-usage-limits 'DataStorage={Minimum=10,Maximum=100,Unit=GB},
 ECPUPerSecond={Minimum=1000,Maximum=100000}'
```

使用 CLI 取消扩展限制

要使用 CLI 删除扩展限制，请将最小和最大限制参数设置为 0。

Linux：

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> \
--cache-usage-limits 'DataStorage={Minimum=0,Maximum=0,Unit=GB},
 ECPUPerSecond={Minimum=0,Maximum=0}'
```

Windows:

```
aws elasticache modify-serverless-cache --serverless-cache-name <cache name> ^
```

```
--cache-usage-limits 'DataStorage={Minimum=0,Maximum=0,Unit=GB},
 ECPUPerSecond={Minimum=0,Maximum=0}'
```

扩展 Red ElastiCache is 自行设计的集群

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。如果您自行管理缓存，则需要预配置足量硬件来满足您的需求高峰，这会产生很高的费用。通过使用 Amazon， ElastiCache 您可以根据当前需求进行扩展，只需按实际用量付费。 ElastiCache 使您能够扩展缓存以满足需求。

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 Redis 集群

操作	Redis（已禁用集群模式）	Redis（已启用集群模式）
缩减	从集群中移除节点	扩展 Redis（启用集群模式）集群
横向扩展	向集群添加节点	Redis（已启用集群模式）的在线重新分片和分片重新平衡
更改节点类型	为更大的节点类型： • 纵向扩展单节点 Redis（已禁用集群模式）集群 • 纵向扩展具有副本的 Redis 集群 为更小的节点类型： • 缩减单节点 Redis 集群 • 缩减具有副本的 Redis 集群	通过修改节点类型来在线纵向扩展
更改节点组数量	Redis（已禁用集群模式）集群不支持	扩展 Redis（启用集群模式）集群

主题

扩展 Redis（已禁用集群模式）集群

Redis（已禁用集群模式）集群可以是具有 0 个分区的单节点集群，也可以是具有 1 个分区的多节点集群。单节点集群使用一个节点执行读取和写入。多节点集群始终有 1 个节点作为读/写主节点，以及 0 到 5 个只读副本节点。

目录

扩展单节点 Redis（已禁用集群模式）集群

Redis（已禁用集群模式）节点必须足够大，以包含所有缓存数据及 Redis 开销。若要更改 Redis（已禁用集群模式）集群的数据容量，必须纵向扩展；纵向扩展为更大的节点类型可提高数据容量，缩减到较小的节点类型可降低数据容量。

ElastiCache for Redis 纵向扩展过程旨在尽最大努力保留您的现有数据，并成功实现 Redis 复制。对于 Redis（已禁用集群模式）集群，建议留有足够的内存可供 Redis 使用。

您不能跨多个 Redis（已禁用集群模式）集群对数据进行分区。不过，如果您只需提高或降低集群的读取容量，则可以创建具有副本节点的 Redis（已禁用集群模式）集群并添加或删除只读副本。要使用单节点 Redis 缓存集群作为主集群来创建具有副本节点的 Redis（已禁用集群模式）集群，请参阅 创建 Redis（已禁用集群模式）集群（控制台）。

创建具有副本的集群后，您可以通过添加只读副本来增加读取容量。之后，您可以根据需要通过删除只读副本来降低读取容量。有关更多信息，请参阅 增加读取容量 或 降低读取容量。

除了可以扩展读取容量之外，具有副本的 Redis（已禁用集群模式）集群还具备其他业务优势。有关更多信息，请参阅使用复制组时的高可用性。

> **Important**
>
> 如果您的参数组使用 `reserved-memory` 为 Redis 开销留出一些内存，则在开始扩展之前，请确保您具有为新节点类型预留正确内存量的自定义参数组。或者，您可以修改自定义参数组以便使用 `reserved-memory-percent`，并为您的新集群使用该参数组。
> 如果您在使用 `reserved-memory-percent`，则这不是必需的。
> 有关更多信息，请参阅管理预留内存。

主题

- 纵向扩展单节点 Redis（已禁用集群模式）集群
- 缩减单节点 Redis 集群

纵向扩展单节点 Redis（已禁用集群模式）集群

当您纵向扩展单节点 Redis 集群时，ElastiCache 会执行以下过程，无论您使用的是 ElastiCache 控制台、AWS CLI 还是 ElastiCache API。

1. 在现有缓存集群所在的可用区中启动具有新节点类型的新缓存集群。
2. 将现有缓存集群中的缓存数据复制到新缓存集群。此过程所需的时间取决于您的节点类型以及缓存集群中的数据量。
3. 使用新缓存集群进行读写操作。由于新缓存集群的终端节点与旧缓存集群的终端节点相同，因此您不需要更新应用程序中的终端节点。在更新 DNS 条目时，您会发现主节点的读取和写入出现短暂中断（几秒钟）。
4. ElastiCache 会删除旧缓存集群。由于与旧节点之间的连接会断开，您会发现旧节点的读取和写入出现短暂中断（几秒钟）。

> **Note**
>
> 对于运行 r6gd 节点类型的集群，您只能在 r6gd 节点系列的节点大小范围内扩缩。

如下表所示，如果您在下一维护时段内安排有引擎升级，则会阻止 Redis 纵向扩展操作。有关维护时段的更多信息，请参阅管理维护。

阻止的 Redis 操作

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

如果有待处理的操作正在阻止您，您可以执行以下操作之一。

- 通过清除 Apply immediately 复选框（CLI 使用：`--no-apply-immediately`，API 使用：`ApplyImmediately=false`），将 Redis 扩展操作安排在下一维护时段内。

- 等到下一维护时段（或之后）再执行 Redis 纵向扩展操作。
- 将 Redis 引擎升级操作添加到此选中了 Apply Immediately 复选框 (CLI 使用：`--apply-immediately`，API 使用：`ApplyImmediately=true`) 的缓存集群修改中。这将导致立即执行引擎升级，从而取消阻止纵向扩展操作。

您可以使用 ElastiCache 控制台、AWS CLI 或 ElastiCache API 纵向扩展单节点 Redis（已禁用集群模式）集群。

> **Important**
>
> 如果您的参数组使用 `reserved-memory` 为 Redis 开销留出一些内存，则在开始扩展之前，请确保您具有为新节点类型预留正确内存量的自定义参数组。或者，您可以修改自定义参数组以便使用 `reserved-memory-percent`，并为您的新集群使用该参数组。
> 如果您在使用 `reserved-memory-percent`，则这不是必需的。
> 有关更多信息，请参阅管理预留内存。

纵向扩展单节点 Redis（已禁用集群模式）集群（控制台）

以下过程介绍如何使用 ElastiCache 管理控制台纵向扩展单节点 Redis 集群。在此过程中，Redis 集群将继续处理请求，且停机时间降至最短。

纵向扩展单节点 Redis 集群（控制台）

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 从导航窗格中，选择 Redis clusters（Redis 集群）。
3. 从集群列表中选择要扩展的集群（它必须运行 Redis 引擎，而不是集群化 Redis 引擎）。
4. 选择 Modify（修改）。
5. 在 Modify Cluster 向导中：
   a. 从 Node type 列表中选择您希望扩展到的节点类型。
   b. 如果您在使用 `reserved-memory` 管理内存，请从 Parameter Group 列表中，选择为新节点类型预留正确内存量的自定义参数组。
6. 如果您要立即执行纵向扩展过程，请选中 Apply immediately 框。如果 Apply immediately 框处于未选中状态，则在此集群的下一维护时段内执行纵向扩展过程。

7. 选择 Modify（修改）。

   如果您在上一步选择了 Apply immediately，则集群的状态将变为 modifying。当状态变为 available 时，即表示修改完成，您可以开始使用新集群。

纵向扩展单节点 Redis 缓存集群 (AWS CLI)

以下过程介绍如何使用 AWS CLI 向上扩展单节点 Redis 缓存集群。在此过程中，Redis 集群将继续处理请求，且停机时间降至最短。

纵向扩展单节点 Redis 缓存集群 (AWS CLI)

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

   - `--cache-cluster-id`

   对于 Linux、macOS 或 Unix：

```
aws elasticache list-allowed-node-type-modifications \
     --cache-cluster-id my-cache-cluster-id
```

   对于 Windows：

```
aws elasticache list-allowed-node-type-modifications ^
     --cache-cluster-id my-cache-cluster-id
```

   以上命令的输出类似于此处所示 (JSON 格式)。

```
{
     "ScaleUpModifications": [
         "cache.m3.2xlarge",
         "cache.m3.large",
         "cache.m3.xlarge",
         "cache.m4.10xlarge",
         "cache.m4.2xlarge",
         "cache.m4.4xlarge",
         "cache.m4.large",
         "cache.m4.xlarge",
         "cache.r3.2xlarge",
```

```
          "cache.r3.4xlarge",
          "cache.r3.8xlarge",
          "cache.r3.large",
          "cache.r3.xlarge"
     ]
        "ScaleDownModifications": [
          "cache.t2.micro",
          "cache.t2.small ",
          "cache.t2.medium ",
             "cache.t1.small ",
     ],

 }
```

有关更多信息，请参阅 AWS CLI 参考中的 list-allowed-node-type-modifications。

2. 使用 AWS CLI `modify-cache-cluster` 命令和以下参数，修改现有缓存集群，并指定要纵向扩展的缓存集群和较大的新节点类型。

   - `--cache-cluster-id` – 要纵向扩展的缓存集群的名称。
   - `--cache-node-type` – 要扩展缓存集群的新节点类型。此值必须是步骤 1 中由 `list-allowed-node-type-modifications` 命令返回的节点类型之一。
   - `--cache-parameter-group-name` – [可选] 如果您使用 `reserved-memory` 管理集群的预留内存，则使用此参数。指定为您的新节点类型预留正确内存量的自定义缓存参数组。如果您在使用 `reserved-memory-percent`，则可以忽略此参数。
   - `--apply-immediately` – 使纵向扩展过程立即得到应用。要将纵向扩展流程推迟到此集群的下一维护时段，请使用 `--no-apply-immediately` 参数。

   对于 Linux、macOS 或 Unix：

```
aws elasticache modify-cache-cluster \
     --cache-cluster-id my-redis-cache-cluster \
     --cache-node-type cache.m3.xlarge \
     --cache-parameter-group-name redis32-m2-xl \
     --apply-immediately
```

   对于 Windows：

```
aws elasticache modify-cache-cluster ^
     --cache-cluster-id my-redis-cache-cluster ^
```

```
    --cache-node-type cache.m3.xlarge ^
    --cache-parameter-group-name redis32-m2-xl ^
    --apply-immediately
```

以上命令的输出类似于此处所示 (JSON 格式)。

```
{
    "CacheCluster": {
        "Engine": "redis",
        "CacheParameterGroup": {
            "CacheNodeIdsToReboot": [],
            "CacheParameterGroupName": "default.redis6.x",
            "ParameterApplyStatus": "in-sync"
        },
        "SnapshotRetentionLimit": 1,
        "CacheClusterId": "my-redis-cache-cluster",
        "CacheSecurityGroups": [],
        "NumCacheNodes": 1,
        "SnapshotWindow": "00:00-01:00",
        "CacheClusterCreateTime": "2017-02-21T22:34:09.645Z",
        "AutoMinorVersionUpgrade": true,
        "CacheClusterStatus": "modifying",
        "PreferredAvailabilityZone": "us-west-2a",
        "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
        "CacheSubnetGroupName": "default",
        "EngineVersion": "6.0",
        "PendingModifiedValues": {
            "CacheNodeType": "cache.m3.2xlarge"
        },
        "PreferredMaintenanceWindow": "tue:11:30-tue:12:30",
        "CacheNodeType": "cache.m3.medium",
         "DataTiering": "disabled"
    }
 }
```

有关更多信息，请参阅 AWS CLI 参考中的 modify-cache-cluster。

3. 如果您使用了 `--apply-immediately`，请使用带以下参数的 AWS CLI `describe-cache-clusters` 命令检查新缓存集群的状态。当状态变为 available 时，您便可开始使用较大的新缓存集群。

- `--cache-cache cluster-id` – 单节点 Redis 缓存集群的名称。使用此参数可描述特定缓存集群而非所有缓存集群。

```
aws elasticache describe-cache-clusters --cache-cluster-id my-redis-cache-cluster
```

有关更多信息，请参阅 AWS CLI 参考中的 describe-cache-clusters。

纵向扩展单节点 Redis 缓存集群 (ElastiCache API)

以下过程介绍如何使用 ElastiCache API 纵向扩展单节点 Redis 缓存集群。在此过程中，Redis 集群将继续处理请求，且停机时间降至最短。

纵向扩展单节点 Redis 缓存集群 (ElastiCache API)

1. 运行带以下参数的 ElastiCache API `ListAllowedNodeTypeModifications` 操作，确定您可纵向扩展为的节点类型。

   - `CacheClusterId` – 要纵向扩展的单节点 Redis 缓存集群的名称。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ListAllowedNodeTypeModifications
    &CacheClusterId=MyRedisCacheCluster
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

   有关更多信息，请参阅 Amazon ElastiCache API 参考中的 ListAllowedNodeTypeModifications。

2. 使用 `ModifyCacheCluster` ElastiCache API 操作和以下参数，修改现有缓存集群，并指定要纵向扩展的缓存集群和较大的新节点类型。

   - `CacheClusterId` – 要纵向扩展的缓存集群的名称。
   - `CacheNodeType` – 要通过纵向扩展使缓存集群达到的较大新节点类型。此值必须是步骤 1 中由 `ListAllowedNodeTypeModifications` 操作返回的节点类型之一。

- CacheParameterGroupName – [可选] 如果您使用 reserved-memory 管理集群的预留内存，则使用此参数。指定为您的新节点类型预留正确内存量的自定义缓存参数组。如果您在使用 reserved-memory-percent，则可以忽略此参数。
- ApplyImmediately – 设置为 true 可促使立即开始执行纵向扩展过程。要将纵向扩展流程推迟到此集群的下一维护时段，请使用 ApplyImmediately=false。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=true
    &CacheClusterId=MyRedisCacheCluster
    &CacheNodeType=cache.m3.xlarge
    &CacheParameterGroupName redis32-m2-xl
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 Amazon ElastiCache API 参考中的 ModifyCacheCluster。

3. 如果您使用了 ApplyImmediately=true，请使用带以下参数的 ElastiCache API DescribeCacheClusters 操作检查新缓存集群的状态。当状态变为 available 时，您便可开始使用较大的新缓存集群。

   - CacheClusterId – 单节点 Redis 缓存集群的名称。使用此参数可描述特定缓存集群而非所有缓存集群。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterId=MyRedisCacheCluster
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 Amazon ElastiCache API 参考中的 DescribeCacheClusters。

缩减单节点 Redis 集群

以下部分介绍了如何将单节点 Redis 集群缩减为较小的节点类型。确保较小的新节点类型足以容纳所有数据和 Redis 开销，这一点对于新 Redis 集群的长期成功至关重要。有关更多信息，请参阅确保具有用于创建 Redis 快照的足够内存。

> **Note**
>
> 对于运行 r6gd 节点类型的集群，您只能在 r6gd 节点系列的节点大小范围内扩缩。

主题

- 缩减单节点 Redis 集群（控制台）
- 缩减单节点 Redis 缓存集群 (AWS CLI)
- 缩减单节点 Redis 缓存集群 (ElastiCache API)

缩减单节点 Redis 集群（控制台）

以下过程介绍如何使用 ElastiCache 控制台将单节点 Redis 集群缩减为较小的节点类型。

> **Important**
>
> 如果您的参数组使用 `reserved-memory` 为 Redis 开销留出一些内存，则在开始扩展之前，请确保您具有为新节点类型预留正确内存量的自定义参数组。或者，您可以修改自定义参数组以便使用 `reserved-memory-percent`，并为您的新集群使用该参数组。
> 如果您在使用 `reserved-memory-percent`，则这不是必需的。
> 有关更多信息，请参阅管理预留内存。

缩减单节点 Redis 集群（控制台）

1. 确保较小的节点类型足以满足您的数据和开销需求。
2. 如果您的参数组使用 `reserved-memory` 为 Redis 开销留出一些内存，请确保您具有为新节点类型预留正确内存量的自定义参数组。

   或者，您可以修改自定义参数组以使用 `reserved-memory-percent`。有关更多信息，请参阅管理预留内存。

3. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
4. 从集群列表中，选择要缩减的集群。该集群必须运行 Redis 引擎，而不是集群化 Redis 引擎。
5. 选择 Modify（修改）。
6. 在 Modify Cluster 向导中：

   a. 从 Node type (节点类型) 列表中选择您希望缩减到的节点类型。

   b. 如果您在使用 `reserved-memory` 管理内存，请从 Parameter Group 列表中，选择为新节点类型预留正确内存量的自定义参数组。
7. 如果您要立即执行缩减过程，请选中 Apply immediately (立即应用) 复选框。如果 Apply immediately (立即应用) 复选框处于未选中状态，则在此集群的下一维护时段内执行缩减过程。
8. 选择 Modify（修改）。
9. 当集群的状态从 modifying 变为 available 时，即表示您的集群已扩展为新的节点类型。无需更新应用程序中的终端节点。

缩减单节点 Redis 缓存集群 (AWS CLI)

以下过程介绍如何使用 AWS CLI 收缩单节点 Redis 缓存集群。

缩减单节点 Redis 缓存集群 (AWS CLI)

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

   - `--cache-cluster-id`

   对于 Linux、macOS 或 Unix：

   ```
   aws elasticache list-allowed-node-type-modifications \
        --cache-cluster-id my-cache-cluster-id
   ```

   对于 Windows：

   ```
   aws elasticache list-allowed-node-type-modifications ^
        --cache-cluster-id my-cache-cluster-id
   ```

   以上命令的输出类似于此处所示 (JSON 格式)。

```
{
     "ScaleUpModifications": [
         "cache.m3.2xlarge",
         "cache.m3.large",
         "cache.m3.xlarge",
         "cache.m4.10xlarge",
         "cache.m4.2xlarge",
         "cache.m4.4xlarge",
         "cache.m4.large",
         "cache.m4.xlarge",
         "cache.r3.2xlarge",
         "cache.r3.4xlarge",
         "cache.r3.8xlarge",
         "cache.r3.large",
         "cache.r3.xlarge"
     ]
        "ScaleDownModifications": [
         "cache.t2.micro",
         "cache.t2.small ",
         "cache.t2.medium ",
            "cache.t1.small ",
     ],

 }
```

有关更多信息，请参阅 AWS CLI 参考中的 list-allowed-node-type-modifications。

2. 使用 AWS CLI `modify-cache-cluster` 命令和以下参数，修改现有缓存集群，并指定要收缩的缓存集群和较小的新节点类型。

   - `--cache-cluster-id` – 要缩减的缓存集群的名称。
   - `--cache-node-type` – 要扩展缓存集群的新节点类型。此值必须是步骤 1 中由 `list-allowed-node-type-modifications` 命令返回的节点类型之一。
   - `--cache-parameter-group-name` – [可选] 如果您使用 `reserved-memory` 管理集群的预留内存，则使用此参数。指定为您的新节点类型预留正确内存量的自定义缓存参数组。如果您在使用 `reserved-memory-percent`，则可以忽略此参数。
   - `--apply-immediately` – 促使立即应用缩减流程。要将纵向扩展流程推迟到此集群的下一维护时段，请使用 `--no-apply-immediately` 参数。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-cache-cluster \
     --cache-cluster-id my-redis-cache-cluster \
     --cache-node-type cache.m3.xlarge \
     --cache-parameter-group-name redis32-m2-xl \
     --apply-immediately
```

对于 Windows：

```
aws elasticache modify-cache-cluster ^
     --cache-cluster-id my-redis-cache-cluster ^
     --cache-node-type cache.m3.xlarge ^
     --cache-parameter-group-name redis32-m2-xl ^
     --apply-immediately
```

以上命令的输出类似于此处所示 (JSON 格式)。

```
{
     "CacheCluster": {
         "Engine": "redis",
         "CacheParameterGroup": {
             "CacheNodeIdsToReboot": [],
             "CacheParameterGroupName": "default.redis6,x",
             "ParameterApplyStatus": "in-sync"
         },
         "SnapshotRetentionLimit": 1,
         "CacheClusterId": "my-redis-cache-cluster",
         "CacheSecurityGroups": [],
         "NumCacheNodes": 1,
         "SnapshotWindow": "00:00-01:00",
         "CacheClusterCreateTime": "2017-02-21T22:34:09.645Z",
         "AutoMinorVersionUpgrade": true,
         "CacheClusterStatus": "modifying",
         "PreferredAvailabilityZone": "us-west-2a",
         "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
         "CacheSubnetGroupName": "default",
         "EngineVersion": "6.0",
         "PendingModifiedValues": {
```

```
            "CacheNodeType": "cache.m3.2xlarge"
        },
        "PreferredMaintenanceWindow": "tue:11:30-tue:12:30",
        "CacheNodeType": "cache.m3.medium",
         "DataTiering": "disabled"
    }
}
```

有关更多信息，请参阅 AWS CLI 参考中的 modify-cache-cluster。

3. 如果您使用了 `--apply-immediately`，请使用带以下参数的 AWS CLI `describe-cache-clusters` 命令检查新缓存集群的状态。当状态变为 available 时，您便可开始使用较大的新缓存集群。

   - `--cache-cache cluster-id` – 单节点 Redis 缓存集群的名称。使用此参数可描述特定缓存集群而非所有缓存集群。

```
aws elasticache describe-cache-clusters --cache-cluster-id my-redis-cache-cluster
```

有关更多信息，请参阅 AWS CLI 参考中的 describe-cache-clusters。

缩减单节点 Redis 缓存集群 (ElastiCache API)

以下过程介绍如何使用 ElastiCache API 扩展/缩减单节点 Redis 缓存集群。

缩减单节点 Redis 缓存集群 (ElastiCache API)

1. 通过运行带以下参数的 ElastiCache API `ListAllowedNodeTypeModifications` 操作，确定您可缩减为的节点类型。

   - `CacheClusterId` –要缩减的单节点 Redis 缓存集群的名称。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ListAllowedNodeTypeModifications
    &CacheClusterId=MyRedisCacheCluster
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
```

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

有关更多信息，请参阅 Amazon ElastiCache API 参考中的 ListAllowedNodeTypeModifications。

2. 使用 ModifyCacheCluster ElastiCache API 操作和以下参数，修改现有缓存集群，并指定要纵向扩展的缓存集群和较大的新节点类型。

   - CacheClusterId – 要缩减的缓存集群的名称。
   - CacheNodeType – 要将缓存集群缩减到的较小的新节点类型。此值必须是步骤 1 中由 ListAllowedNodeTypeModifications 操作返回的节点类型之一。
   - CacheParameterGroupName – [可选] 如果您使用 reserved-memory 管理集群的预留内存，则使用此参数。指定为您的新节点类型预留正确内存量的自定义缓存参数组。如果您在使用 reserved-memory-percent，则可以忽略此参数。
   - ApplyImmediately – 设置为 true 可促使立即执行缩减过程。要将纵向扩展流程推迟到此集群的下一维护时段，请使用 ApplyImmediately=false。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=true
    &CacheClusterId=MyRedisCacheCluster
    &CacheNodeType=cache.m3.xlarge
    &CacheParameterGroupName redis32-m2-xl
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 Amazon ElastiCache API 参考中的 ModifyCacheCluster。

3. 如果您使用了 ApplyImmediately=true，请使用带以下参数的 ElastiCache API DescribeCacheClusters 操作检查新缓存集群的状态。当状态变为 available 时，您便可开始使用较小的新缓存集群。

   - CacheClusterId – 单节点 Redis 缓存集群的名称。使用此参数可描述特定缓存集群而非所有缓存集群。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
```

```
    &CacheClusterId=MyRedisCacheCluster
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 Amazon ElastiCache API 参考中的 DescribeCacheClusters。

扩展具有副本节点的 Redis（已禁用集群模式）集群

具有副本节点的 Redis 集群（在 API/CLI 中称作复制组）通过启用了自动故障转移功能的多可用区的复制来提供高可用性。具有副本节点的集群是最多包含 6 个 Redis 节点的逻辑集合（其中一个节点为主节点），能够处理读写请求。该集群中的所有其他节点均为主集群的只读副本。写入主集群的数据异步复制到集群中的所有只读副本。由于 Redis（禁用集群模式）不支持跨多个集群对数据进行分区，因此 Redis（禁用集群模式）复制组中的每个节点都包含整个缓存数据集。Redis（已启用集群模式）集群支持跨多达 500 个分区对数据进行分区。

要更改集群的数据容量，必须将复制组纵向扩展为较大的节点类型或收缩为较小的节点类型。

要更改集群的读取容量，可添加更多只读副本（最多 5 个）或移除只读副本。

ElastiCache 纵向扩展过程旨在尽最大努力保留您的现有数据，并成功实现 Redis 复制。对于具有副本的 Redis 集群，建议有足够的内存可供 Redis 使用。

相关主题

- 使用复制组时的高可用性
- 复制：Redis（已禁用集群模式）与 Redis（已启用集群模式）对比
- 使用多可用区最大限度地 ElastiCache 缩短 Redis 的停机时间
- 确保具有用于创建 Redis 快照的足够内存

主题

- 纵向扩展具有副本的 Redis 集群
- 缩减具有副本的 Redis 集群
- 增加读取容量
- 降低读取容量

纵向扩展具有副本的 Redis 集群

Amazon ElastiCache 提供控制台、CLI 和 API 支持，用于纵向扩展 Redis（已禁用集群模式）复制组。

启动纵向扩展流程时，ElastiCache 执行以下操作：

1. 使用新节点类型启动复制组。
2. 将当前主节点中的所有数据复制到新的主节点。
3. 将新的只读副本与新的主节点同步。
4. 更新 DNS 条目使其指向新的节点。因此，您便不需要更新应用程序中的终端节点。对于 Redis 5.0.5 及更高版本，您可以在该集群继续保持在线并处理传入请求时扩展启用自动故障转移的集群。在版本 4.0.10 及更低版本上，更新 DNS 条目时，您可能会发现先前版本上来自主节点的读取和写入短暂中断。
5. 删除旧节点（CLI/API：复制组）。由于与旧节点之间的连接会断开，您会发现旧节点的读取和写入出现短暂中断（几秒钟）。

此过程所需的时间取决于您的节点类型以及集群中的数据量。

如下表所示，如果您在集群的下一维护时段内安排有引擎升级，则会阻止 Redis 纵向扩展操作。

阻止的 Redis 操作

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

如果有待处理的操作正在阻止您，您可以执行以下操作之一。

- 通过清除 Apply immediately 复选框（CLI 使用：`--no-apply-immediately`，API 使用：`ApplyImmediately=false`），将 Redis 扩展操作安排在下一维护时段内。
- 等到下一维护时段（或之后）再执行 Redis 纵向扩展操作。

- 将 Redis 引擎升级操作添加到此选中了 Apply Immediately 复选框 (CLI 使用：`--apply-immediately`，API 使用：`ApplyImmediately=true`) 的缓存集群修改中。这将导致立即执行引擎升级，从而取消阻止纵向扩展操作。

以下部分介绍如何使用 ElastiCache 控制台、AWS CLI 和 ElastiCache API 纵向扩展具有副本的 Redis 集群。

> **Important**
>
> 如果您的参数组使用 `reserved-memory` 为 Redis 开销留出一些内存，则在开始扩展之前，请确保您具有为新节点类型预留正确内存量的自定义参数组。或者，您可以修改自定义参数组以便使用 `reserved-memory-percent`，并为您的新集群使用该参数组。
> 如果您在使用 `reserved-memory-percent`，则这不是必需的。
> 有关更多信息，请参阅管理预留内存。

纵向扩展具有副本的 Redis 集群（控制台）

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。

以下过程使用 ElastiCache 控制台将具有副本的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，更新 DNS 条目时，其他版本的主节点可能会短暂中断读取和写入。对于在 5.0.6 版本以更高版本上运行的节点，可能会出现不到一秒钟的停机时间；对于较早的版本，则可能会出现几秒钟的停机时间。

纵向扩展具有副本的 Redis 集群（控制台）

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 从导航窗格中，选择 Redis clusters（Redis 集群）
3. 从集群列表中，选择要扩展的集群。该集群必须运行 Redis 引擎，而不是集群化 Redis 引擎。
4. 选择 Modify（修改）。
5. 在 Modify Cluster 向导中：

    a. 从 Node type 列表中选择您希望扩展到的节点类型。请注意，并不是可缩减到所有节点类型。

    b. 如果您在使用 `reserved-memory` 管理内存，请从 Parameter Group 列表中，选择为新节点类型预留正确内存量的自定义参数组。

6. 如果您要立即执行纵向扩展流程，请选中 Apply immediately 复选框。如果 Apply immediately 复选框处于未选中状态，则在此集群的下一维护时段内执行纵向扩展过程。
7. 选择 Modify（修改）。
8. 当集群的状态从 modifying 变为 available 时，即表示您的集群已扩展为新的节点类型。无需更新应用程序中的终端节点。

纵向扩展 Redis 复制组 (AWS CLI)

以下过程使用 AWS CLI 将复制组从其当前节点类型扩展为较大的新节点类型。在此过程中，ElastiCache for Redis 会更新 DNS 条目使其指向新的节点。因此，您便不需要更新应用程序中的终端节点。对于 Redis 5.0.5 及更高版本，您可以在该集群继续保持在线并处理传入请求时扩展启用自动故障转移的集群。在版本 4.0.10 及更低版本上，更新 DNS 条目时，您可能会发现先前版本上来自主节点的读取和写入短暂中断。

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缓存集群中的数据量不同而异。

纵向扩展 Redis 复制组 (AWS CLI)

1. 通过运行带以下参数的 AWS CLI `list-allowed-node-type-modifications` 命令，确定您可纵向扩展到的节点类型。

   - `--replication-group-id` – 复制组的名称。使用此参数可描述特定复制组而非所有复制组。

   对于 Linux、macOS 或 Unix：

   ```
   aws elasticache list-allowed-node-type-modifications \
        --replication-group-id my-repl-group
   ```

   对于 Windows：

   ```
   aws elasticache list-allowed-node-type-modifications ^
        --replication-group-id my-repl-group
   ```

   该操作的输出内容应类似如下所示 (JSON 格式)。

   ```
   {
        "ScaleUpModifications": [
   ```

```
            "cache.m3.2xlarge",
            "cache.m3.large",
            "cache.m3.xlarge",
            "cache.m4.10xlarge",
            "cache.m4.2xlarge",
            "cache.m4.4xlarge",
            "cache.m4.large",
            "cache.m4.xlarge",
            "cache.r3.2xlarge",
            "cache.r3.4xlarge",
            "cache.r3.8xlarge",
            "cache.r3.large",
            "cache.r3.xlarge"
        ]
    }
```

有关更多信息，请参阅 AWS CLI 参考中的 list-allowed-node-type-modifications。

2. 使用带以下参数的 AWS CLI `modify-replication-group` 命令将当前复制组扩展为新的节点类型。

   - `--replication-group-id` – 复制组的名称。
   - `--cache-node-type` – 此复制组中缓存集群的较大新节点类型。此值必须是步骤 1 中由 `list-allowed-node-type-modifications` 命令返回的实例类型之一。
   - `--cache-parameter-group-name` – [可选] 如果您使用 `reserved-memory` 管理集群的预留内存，则使用此参数。指定为您的新节点类型预留正确内存量的自定义缓存参数组。如果您在使用 `reserved-memory-percent`，则可以忽略此参数。
   - `--apply-immediately` – 使纵向扩展过程立即得到应用。要将扩展操作推迟到下一维护时段，请使用 `--no-apply-immediately`。

   对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
     --replication-group-id my-repl-group \
     --cache-node-type cache.m3.xlarge \
     --cache-parameter-group-name redis32-m3-2xl \
     --apply-immediately
```

   对于 Windows：

```
aws elasticache modify-replication-group ^
     --replication-group-id my-repl-group ^
     --cache-node-type cache.m3.xlarge ^
     --cache-parameter-group-name redis32-m3-2xl \
     --apply-immediately
```

该命令的输出内容应类似如下所示 (JSON 格式)。

```
{
 "ReplicationGroup": {
  "Status": "available",
  "Description": "Some description",
  "NodeGroups": [{
   "Status": "available",
   "NodeGroupMembers": [{
     "CurrentRole": "primary",
     "PreferredAvailabilityZone": "us-west-2b",
     "CacheNodeId": "0001",
     "ReadEndpoint": {
      "Port": 6379,
      "Address": "my-repl-group-001.8fdx4s.0001.usw2.cache.amazonaws.com"
     },
     "CacheClusterId": "my-repl-group-001"
    },
    {
     "CurrentRole": "replica",
     "PreferredAvailabilityZone": "us-west-2c",
     "CacheNodeId": "0001",
     "ReadEndpoint": {
      "Port": 6379,
      "Address": "my-repl-group-002.8fdx4s.0001.usw2.cache.amazonaws.com"
     },
     "CacheClusterId": "my-repl-group-002"
    }
   ],
   "NodeGroupId": "0001",
   "PrimaryEndpoint": {
    "Port": 6379,
    "Address": "my-repl-group.8fdx4s.ng.0001.usw2.cache.amazonaws.com"
   }
  }],
```

```
  "ReplicationGroupId": "my-repl-group",
  "SnapshotRetentionLimit": 1,
  "AutomaticFailover": "disabled",
  "SnapshotWindow": "12:00-13:00",
  "SnapshottingClusterId": "my-repl-group-002",
  "MemberClusters": [
   "my-repl-group-001",
   "my-repl-group-002"
  ],
  "PendingModifiedValues": {}
 }
}
```

有关更多信息，请参阅 AWS CLI 参考中的 modify-replication-group。

3. 如果您使用了 `--apply-immediately` 参数，请使用带以下参数的 AWS CLI `describe-replication-group` 命令监控复制组的状态。当状态仍处于正在修改时，在更新 DNS 条目时，在 5.0.6 版本及更高版本上运行的节点可能会出现不到一秒钟的停机时间，以及较早版本上来自主节点的读取和写入出现短暂中断。

   - `--replication-group-id` – 复制组的名称。使用此参数可描述特定复制组而非所有复制组。

   对于 Linux、macOS 或 Unix：

   ```
   aws elasticache describe-replication-groups \
        --replication-group-id my-replication-group
   ```

   对于 Windows：

   ```
   aws elasticache describe-replication-groups ^
        --replication-group-id my-replication-group
   ```

   有关更多信息，请参阅 AWS CLI 参考中的 describe-replication-groups。

纵向扩展 Redis 复制组 (ElastiCache API)

以下过程使用 ElastiCache API 将复制组从其当前节点类型扩展为较大的新节点类型。对于 Redis 5.0.5 及更高版本，您可以在该集群继续保持在线并处理传入请求时扩展启用自动故障转移的集群。在

版本 4.0.10 及更低版本上，更新 DNS 条目时，您可能会发现先前版本上来自主节点的读取和写入短暂中断。

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缓存集群中的数据量不同而异。

纵向扩展 Redis 复制组 (ElastiCache API)

1. 使用带以下参数的 ElastiCache API `ListAllowedNodeTypeModifications` 操作确定您可纵向扩展为的节点类型。

   - `ReplicationGroupId` – 复制组的名称。使用此参数可描述特定复制组而非所有复制组。

   ```
   https://elasticache.us-west-2.amazonaws.com/
       ?Action=ListAllowedNodeTypeModifications
       &ReplicationGroupId=MyReplGroup
       &Version=2015-02-02
       &SignatureVersion=4
       &SignatureMethod=HmacSHA256
       &Timestamp=20150202T192317Z
       &X-Amz-Credential=<credential>
   ```

   有关更多信息，请参阅 Amazon ElastiCache API 参考中的 ListAllowedNodeTypeModifications。

2. 使用带以下参数的 `ModifyRedplicationGroup` ElastiCache API 操作将当前复制组扩展为新的节点类型。

   - `ReplicationGroupId` – 复制组的名称。
   - `CacheNodeType` – 此复制组中缓存集群的较大新节点类型。此值必须是步骤 1 中由 `ListAllowedNodeTypeModifications` 操作返回的实例类型之一。
   - `CacheParameterGroupName` – [可选] 如果您使用 `reserved-memory` 管理集群的预留内存，则使用此参数。指定为您的新节点类型预留正确内存量的自定义缓存参数组。如果您在使用 `reserved-memory-percent`，则可以忽略此参数。
   - `ApplyImmediately` – 设置为 `true` 可促使立即应用纵向扩展流程。要将扩展流程推迟到下一维护时段，请使用 `ApplyImmediately=false`。

   ```
   https://elasticache.us-west-2.amazonaws.com/
       ?Action=ModifyReplicationGroup
       &ApplyImmediately=true
       &CacheNodeType=cache.m3.2xlarge
   ```

```
    &CacheParameterGroupName=redis32-m3-2xl
    &ReplicationGroupId=myReplGroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

有关更多信息，请参阅 Amazon ElastiCache API 参考中的 ModifyReplicationGroup。

3. 如果您使用了 `ApplyImmediately=true`，请使用带以下参数的 ElastiCache API `DescribeReplicationGroups` 操作监控复制组的状态。当状态从 modifying 变为 available 时，您便可开始写入已扩展的新复制组。

   - `ReplicationGroupId` – 复制组的名称。使用此参数可描述特定复制组而非所有复制组。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReplicationGroups
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 Amazon ElastiCache API 参考中的 DescribeReplicationGroups。

缩减具有副本的 Redis 集群

以下部分介绍了如何将具有副本节点的 Redis（已禁用集群模式）缓存集群缩减为较小的节点类型。确保较小的新节点类型足以容纳所有数据和开销对成功非常重要。有关更多信息，请参阅确保具有用于创建 Redis 快照的足够内存。

> **Note**
>
> 对于运行 r6gd 节点类型的集群，您只能在 r6gd 节点系列的节点大小范围内扩缩。

> **Important**
>
> 如果您的参数组使用 `reserved-memory` 为 Redis 开销留出一些内存，则在开始扩展之前，请确保您具有为新节点类型预留正确内存量的自定义参数组。或者，您可以修改自定义参数组以便使用 `reserved-memory-percent`，并为您的新集群使用该参数组。
> 如果您在使用 `reserved-memory-percent`，则这不是必需的。
> 有关更多信息，请参阅管理预留内存。

缩减具有副本的 Redis 集群（控制台）

以下过程使用 ElastiCache 控制台将具有副本节点的 Redis 集群缩减为较小的节点类型。

缩减具有副本节点的 Redis 集群（控制台）

1. 确保较小的节点类型足以满足您的数据和开销需求。
2. 如果您的参数组使用 `reserved-memory` 为 Redis 开销留出一些内存，请确保您具有为新节点类型预留正确内存量的自定义参数组。

   或者，您可以修改自定义参数组以使用 `reserved-memory-percent`。有关更多信息，请参阅管理预留内存。
3. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
4. 从集群列表中，选择要缩减的集群。该集群必须运行 Redis 引擎，而不是集群化 Redis 引擎。
5. 选择 Modify（修改）。
6. 在 Modify Cluster 向导中：

a. 从 Node type (节点类型) 列表中选择您希望缩减到的节点类型。

b. 如果您在使用 `reserved-memory` 管理内存，请从 Parameter Group 列表中，选择为新节点类型预留正确内存量的自定义参数组。

7. 如果您要立即执行缩减过程，请选中 Apply immediately (立即应用) 复选框。如果 Apply immediately (立即应用) 复选框处于未选中状态，则在此集群的下一维护时段内执行缩减过程。
8. 选择 Modify（修改）。
9. 当集群的状态从 modifying 变为 available 时，即表示您的集群已扩展为新的节点类型。无需更新应用程序中的终端节点。

缩减 Redis 复制组 (AWS CLI)

以下过程使用 AWS CLI 将复制组从其当前节点类型收缩为较小的新节点类型。在此过程中，ElastiCache for Redis 会更新 DNS 条目使其指向新的节点。因此，您便不需要更新应用程序中的终端节点。对于 Redis 5.0.5 及更高版本，您可以在该集群继续保持在线并处理传入请求时扩展启用自动故障转移的集群。在版本 4.0.10 及更低版本上，更新 DNS 条目时，您可能会发现先前版本上来自主节点的读取和写入短暂中断。

但是，只读副本缓存集群的读取继续不受干扰地进行。

缩减为较小的节点类型所需的时间因节点类型和当前缓存集群中的数据量而异。

缩减 Redis 复制组 (AWS CLI)

1. 通过运行带以下参数的 AWS CLI `list-allowed-node-type-modifications` 命令，确定您可收缩到的节点类型。

   - `--replication-group-id` – 复制组的名称。使用此参数可描述特定复制组而非所有复制组。

   对于 Linux、macOS 或 Unix：

   ```
   aws elasticache list-allowed-node-type-modifications \
        --replication-group-id my-repl-group
   ```

   对于 Windows：

   ```
   aws elasticache list-allowed-node-type-modifications ^
   ```

```
    --replication-group-id my-repl-group
```

该操作的输出内容应类似如下所示 (JSON 格式)。

```
{
    "ScaleDownModifications": [
        "cache.m3.2xlarge",
        "cache.m3.large",
        "cache.m3.xlarge",
        "cache.m4.10xlarge",
        "cache.m4.2xlarge",
        "cache.m4.4xlarge",
        "cache.m4.large",
        "cache.m4.xlarge",
        "cache.r3.2xlarge",
        "cache.r3.4xlarge",
        "cache.r3.8xlarge",
        "cache.r3.large",
        "cache.r3.xlarge"
    ]
}
```

有关更多信息，请参阅 AWS CLI 参考中的 list-allowed-node-type-modifications。

2. 使用带以下参数的 AWS CLI `modify-replication-group` 命令将当前复制组扩展为新的节点类型。

   - `--replication-group-id` – 复制组的名称。
   - `--cache-node-type` – 此复制组中缓存集群的较小新节点类型。此值必须是步骤 1 中由 `list-allowed-node-type-modifications` 命令返回的实例类型之一。
   - `--cache-parameter-group-name` – [可选] 如果您使用 `reserved-memory` 管理集群的预留内存，则使用此参数。指定为您的新节点类型预留正确内存量的自定义缓存参数组。如果您在使用 `reserved-memory-percent`，则可以忽略此参数。
   - `--apply-immediately` – 使纵向扩展过程立即得到应用。要将扩展操作推迟到下一维护时段，请使用 `--no-apply-immediately`。

   对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
    --replication-group-id my-repl-group \
```

```
    --cache-node-type cache.t2.small  \
    --cache-parameter-group-name redis32-m3-2xl \
    --apply-immediately
```

对于 Windows：

```
aws elasticache modify-replication-group ^
    --replication-group-id my-repl-group ^
    --cache-node-type cache.t2.small  ^
    --cache-parameter-group-name redis32-m3-2xl \
    --apply-immediately
```

该命令的输出内容应类似如下所示 (JSON 格式)。

```
{"ReplicationGroup": {
         "Status": "available",
         "Description": "Some description",
         "NodeGroups": [
             {
                 "Status": "available",
                 "NodeGroupMembers": [
                     {
                         "CurrentRole": "primary",
                         "PreferredAvailabilityZone": "us-west-2b",
                         "CacheNodeId": "0001",
                         "ReadEndpoint": {
                             "Port": 6379,
                             "Address": "my-repl-
group-001.8fdx4s.0001.usw2.cache.amazonaws.com"
                         },
                         "CacheClusterId": "my-repl-group-001"
                     },
                     {
                         "CurrentRole": "replica",
                         "PreferredAvailabilityZone": "us-west-2c",
                         "CacheNodeId": "0001",
                         "ReadEndpoint": {
                             "Port": 6379,
                             "Address": "my-repl-
group-002.8fdx4s.0001.usw2.cache.amazonaws.com"
                         },
```

```
                        "CacheClusterId": "my-repl-group-002"
                    }
                ],
                "NodeGroupId": "0001",
                "PrimaryEndpoint": {
                    "Port": 6379,
                    "Address": "my-repl-
group.8fdx4s.ng.0001.usw2.cache.amazonaws.com"
                }
            }
        ],
        "ReplicationGroupId": "my-repl-group",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "disabled",
        "SnapshotWindow": "12:00-13:00",
        "SnapshottingClusterId": "my-repl-group-002",
        "MemberClusters": [
            "my-repl-group-001",
            "my-repl-group-002",
        ],
        "PendingModifiedValues": {}
    }
 }
```

有关更多信息，请参阅 AWS CLI 参考中的 modify-replication-group。

3. 如果您使用了 `--apply-immediately` 参数，请使用带以下参数的 AWS CLI `describe-replication-group` 命令监控复制组的状态。当状态从 modifying 变为 available 时，您便可开始写入已收缩的新复制组。

   - `--replication-group-id` – 复制组的名称。使用此参数可描述特定复制组而非所有复制组。

   对于 Linux、macOS 或 Unix：

   ```
   aws elasticache describe-replication-group \
       --replication-group-id my-replication-group
   ```

   对于 Windows：

   ```
   aws elasticache describe-replication-groups ^
   ```

```
    --replication-group-id my-replication-group
```

有关更多信息，请参阅 AWS CLI 参考中的 describe-replication-groups。

缩减 Redis 复制组 (ElastiCache API)

以下过程使用 ElastiCache API 将复制组从其当前节点类型缩减为较小的新节点类型。在此过程中，ElastiCache for Redis 会更新 DNS 条目使其指向新的节点。因此，您便不需要更新应用程序中的终端节点。对于 Redis 5.0.5 及更高版本，您可以在该集群继续保持在线并处理传入请求时扩展启用自动故障转移的集群。在版本 4.0.10 及更低版本上，更新 DNS 条目时，您可能会发现先前版本上来自主节点的读取和写入短暂中断。但是，只读副本缓存集群的读取继续不受干扰地进行。

缩减为较小的节点类型所需的时间因节点类型和当前缓存集群中的数据量而异。

缩减 Redis 复制组 (ElastiCache API)

1. 使用带以下参数的 ElastiCache API `ListAllowedNodeTypeModifications` 操作确定您可缩减为的节点类型。

   - `ReplicationGroupId` – 复制组的名称。使用此参数可描述特定复制组而非所有复制组。

   ```
   https://elasticache.us-west-2.amazonaws.com/
       ?Action=ListAllowedNodeTypeModifications
       &ReplicationGroupId=MyReplGroup
       &Version=2015-02-02
       &SignatureVersion=4
       &SignatureMethod=HmacSHA256
       &Timestamp=20150202T192317Z
       &X-Amz-Credential=<credential>
   ```

   有关更多信息，请参阅 Amazon ElastiCache API 参考中的 ListAllowedNodeTypeModifications。

2. 使用带以下参数的 `ModifyRedplicationGroup` ElastiCache API 操作将当前复制组扩展为新的节点类型。

   - `ReplicationGroupId` – 复制组的名称。
   - `CacheNodeType` – 此复制组中缓存集群的较小新节点类型。此值必须是步骤 1 中由 `ListAllowedNodeTypeModifications` 操作返回的实例类型之一。

- CacheParameterGroupName – [可选] 如果您使用 reserved-memory 管理集群的预留内存，则使用此参数。指定为您的新节点类型预留正确内存量的自定义缓存参数组。如果您在使用 reserved-memory-percent，则可以忽略此参数。
- ApplyImmediately – 设置为 true 可促使立即应用纵向扩展流程。要将缩减流程推迟到下一维护时段，请使用 ApplyImmediately=false。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyReplicationGroup
    &ApplyImmediately=true
    &CacheNodeType=cache.m3.2xlarge
    &CacheParameterGroupName=redis32-m3-2xl
    &ReplicationGroupId=myReplGroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

有关更多信息，请参阅 Amazon ElastiCache API 参考中的 ModifyReplicationGroup。

3. 如果您使用了 ApplyImmediately=true，请使用带以下参数的 ElastiCache API DescribeReplicationGroups 操作监控复制组的状态。当状态从 modifying 变为 available 时，您便可开始写入已收缩的新复制组。

   - ReplicationGroupId – 复制组的名称。使用此参数可描述特定复制组而非所有复制组。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReplicationGroups
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 Amazon ElastiCache API 参考中的 DescribeReplicationGroups。

增加读取容量

要增加读取容量，可向 Redis 复制组中添加只读副本（最多 5 个）。

您可以使用 ElastiCache 控制台、AWS CLI 或 ElastiCache API 扩展 Redis 集群的读取容量。有关更多信息，请参阅向 Redis（已禁用集群模式）复制组添加只读副本。

降低读取容量

要降低读取容量，请从具有副本的 Redis 集群中删除一个或多个只读副本（在 API/CLI 中称作复制组）。如果集群是启用自动故障转移功能的多可用区，则在未先禁用多可用区的情况下，无法删除上一个只读副本。有关更多信息，请参阅修改复制组。

有关更多信息，请参阅删除 Redis（已禁用集群模式）复制组的只读副本 。

扩展 Redis（启用集群模式）集群

由于对集群的需求发生变化，您可能决定通过更改 Redis（已启用集群模式）集群中的分区数量来提高性能或降低成本。我们建议使用在线水平扩展来实现这一目的，因为采用这种方法，您的集群在扩展过程中可以继续为请求提供服务。

您决定重新调节集群的情况包括以下几种：

- 内存压力：

  如果集群中的节点存在内存压力，您可能会决定进行横向扩展，以便获得更多资源来更好地存储数据并为请求提供服务。

  您可以通过监控以下指标来确定您的节点是否承受内存压力：FreeableMemorySwapUsage、和BytesUseForCache。
- 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延迟和吞吐量级别：CPU利用率、输NetworkBytes入CurrConnections、NetworkBytes输出和。NewConnections
- 您的集群过度扩展：

  对集群的当前需求是缩减集群不会降低性能，并可以降低成本。

  您可以使用以下指标监控集群的使用情况，以确定是否可以安全地进行扩展：FreeableMemory、、、CPUUssization SwapUsage、BytesUseForCacheIn、NetworkBytes Ou NetworkBytes t 和。CurrConnectionsNewConnections

扩展的性能影响

当使用离线过程进行扩展时，您的集群在大部分过程中处于离线状态，因此无法为请求提供服务。当使用在线方法进行扩展时，由于扩展是计算密集型操作，因此会导致一定程度的性能下降，但是在整个扩展操作过程中您的集群仍然会继续为请求提供服务。性能的降低程度取决于您的常规 CPU 利用率和数据。

有两种方法可以扩展您的 Redis（已启用集群模式）集群：横向和纵向扩展。

- 利用横向扩展，可以通过添加或删除节点组（分片）来更改复制组中的节点组（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。

采用与旧集群不同的方法来配置新集群中的槽。仅采用离线方法。

- 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。

有关更多信息，请参阅选择缓存节点大小。

目录

Redis（已启用集群模式）的离线重新分区和分区重新平衡

离线分片重新配置带来的主要优势便是，除了在复制组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，除了更改复制组中的分片数量，您还可以执行以下操作：

> **Note**
>
> 启用了数据分层的 Redis 集群不支持离线重新分片。有关更多信息，请参阅数据分层。

- 更改复制组的节点类型。
- 为复制组中的每个节点指定可用区。
- 升级为更新的引擎版本。
- 单独指定每个分片中的副本节点数量。
- 为每个分片指定密钥空间。

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。

离线重新配置分区 Redis（已启用集群模式）集群

1. 创建现有 Redis 集群的手动备份。有关更多信息，请参阅进行手动备份。
2. 通过从备份中还原来创建新集群。有关更多信息，请参阅从备份还原到新缓存。
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。

Redis（已启用集群模式）的在线重新分片和分片重新平衡

通过在 A ElastiCache mazon for Redis 版本 3.2.10 或更高版本中使用在线重新分片和分片再平衡，您可以在不停机的情况下动态扩展 ElastiCache Redis（已启用集群模式）。此方法意味着，即使在进行扩展或重新平衡的过程中，您的集群也可以继续为请求提供服务。

您可执行以下操作：

- 横向扩展 – 通过向 Redis（已启用集群模式）集群（复制组）添加分片（节点组）来增加读写容量。

  如果您向复制组添加一个或多个分片，则每个新分片中的节点数量与最小的现有分片中的节点数量相同。
- 横向缩减 – 通过删除 Redis（已启用集群模式）集群中的分片来降低读写容量，从而降低成本。

- 重新平衡 — 在 for Redis（已启用集群模式）集群中的 ElastiCache 分片之间移动密钥空间，使其在分片之间尽可能均匀分布。

您无法执行以下操作：

- 单独配置分片：

  您无法单独指定分片的键空间。要执行此操作，您必须使用离线过程。

目前，以下限制适用于 Redi ElastiCache s 在线重新分片和重新平衡：

- 这些过程需要 Redis 引擎版本 3.2.10 或更新版本。有关升级引擎版本的信息，请参阅引擎版本和升级。
- 槽或键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最小的现有分片中的节点数量。
- 在横向扩展时，所有现有分片的任何常见标签将被复制到新分片中。
- 扩展全局数据存储集群时，不会自动 ElastiCache 将函数从一个现有节点复制到新节点。我们建议在横向扩展集群后将您的函数加载到新的分片中，这样每个分片都具有相同的函数。

> **Note**
>
> ElastiCache 适用于 Redis 版本 7 及更高版本：在扩展集群时， ElastiCache 会自动将现有节点之一（随机选择）中加载的函数复制到新节点。如果您的应用程序使用 Redis Functions，我们建议您在扩展之前将所有函数加载到所有分片中，这样您 ElastiCache 的 for Redis 集群就不会在不同的分片上出现不同的函数定义。

有关更多信息，请参阅 在线集群大小调整。

您可以使用、和 API 水平扩展或重新平衡 ElastiCache 适用于 Redis（已启用集群模式）的 AWS CLI 集群。 AWS Management Console ElastiCache

通过在线重新分片功能添加分片

您可以使用、或 ElastiCache API 向 Redis（已启用集群模式）集群添加分片。 AWS Management Console AWS CLI向 Redis（已启用集群模式）集群添加分片时，现有分片的所有标签都将复制到新分片。

添加分片（控制台）

您可以使用 AWS Management Console 向您的 Redis (已启用集群模式) 集群添加一个或多个分片。以下步骤描述了这个过程。

向 Redis（已启用集群模式）集群添加分片

1. 打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 从导航窗格中，选择 Redis clusters（Redis 集群）。
3. 找到并选择要将分片添加到其中的 Redis（已启用集群模式）集群的名称，而不是选择集群名称左侧的框。

   > **Tip**
   >
   > Redis（已启用集群模式）的 Mode（模式）列中会显示 Clustered Redis（集群化 Redis）

4. 选择 Add shard。

   a. 对于 Number of shards to be added，请选择要添加到此集群的分片数量。

   b. 对于 Availability zone(s)，请选择 No preference 或 Specify availability zones。

   c. 如果您选择 Specify availability zones，则对于每个分片中的每个节点，请从可用区列表中选择节点的可用区。

   d. 选择添加。

添加分片（AWS CLI）

以下过程介绍了如何通过使用 AWS CLI添加分片来重新配置 Redis（已启用集群模式）集群中的分片。

在`modify-replication-group-shard-configuration`中使用以下参数：

参数

- `--apply-immediately` – 必需。指定分片重新配置操作立即开始。
- `--replication-group-id` – 必需。指定在哪个复制组（集群）上执行分片重新配置操作。
- `--node-group-count` – 必需。指定操作完成时存在的分片（节点组）数量。添加分片后，`--node-group-count` 的值必须大于当前分片数量。

  您也可以使用 `--resharding-configuration` 为复制组中的每个节点指定可用区。
- `--resharding-configuration` – 可选。复制组中每个分片中的每个节点的首选可用区列表。只有当 `--node-group-count` 的值大于当前分片数量时，才能使用此参数。如果在添加分片时省略此参数，Amazon 将为新节点 ElastiCache 选择可用区。

以下示例将重新配置 Redis（已启用集群模式）集群 `my-cluster` 中四个分片中的键空间。该示例还为每个分片中的每个节点指定可用区。操作将立即开始。

Example - 添加分片

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group-shard-configuration \
    --replication-group-id my-cluster \
    --node-group-count 4 \
    --resharding-configuration \
        "PreferredAvailabilityZones=us-east-2a,us-east-2c" \
        "PreferredAvailabilityZones=us-east-2b,us-east-2a" \
        "PreferredAvailabilityZones=us-east-2c,us-east-2d" \
        "PreferredAvailabilityZones=us-east-2d,us-east-2c" \
    --apply-immediately
```

对于 Windows：

```
aws elasticache modify-replication-group-shard-configuration ^
    --replication-group-id my-cluster ^
    --node-group-count 4 ^
    --resharding-configuration ^
        "PreferredAvailabilityZones=us-east-2a,us-east-2c" ^
        "PreferredAvailabilityZones=us-east-2b,us-east-2a" ^
        "PreferredAvailabilityZones=us-east-2c,us-east-2d" ^
        "PreferredAvailabilityZones=us-east-2d,us-east-2c" ^
    --apply-immediately
```

有关更多信息，请参阅文档中的修改复制组分片配置。 AWS CLI

添加分片 (ElastiCache API)

您可以通过操作使用 ElastiCache API 在线重新配置 Redis（已启用集群模式）集群中的分片。ModifyReplicationGroupShardConfiguration

在ModifyReplicationGroupShardConfiguration中使用以下参数：

参数

- ApplyImmediately=true – 必需。指定分片重新配置操作立即开始。
- ReplicationGroupId – 必需。指定在哪个复制组（集群）上执行分片重新配置操作。
- NodeGroupCount – 必需。指定操作完成时存在的分片（节点组）数量。添加分片后，NodeGroupCount 的值必须大于当前分片数量。

  您也可以使用 ReshardingConfiguration 为复制组中的每个节点指定可用区。
- ReshardingConfiguration – 可选。复制组中每个分片中的每个节点的首选可用区列表。只有当 NodeGroupCount 的值大于当前分片数量时，才能使用此参数。如果在添加分片时省略此参数，Amazon 将为新节点 ElastiCache 选择可用区。

以下过程介绍如何通过使用 API 添加分片来重新配置 Redis（已启用集群模式）集群中的分片。ElastiCache

Example - 添加分片

以下示例将节点组添加到 Redis（已启用集群模式）集群 my-cluster 中，因此当操作完成后共有 4 个节点组。该示例还为每个分片中的每个节点指定可用区。操作将立即开始。

```
https://elasticache.us-east-2.amazonaws.com/
    ?Action=ModifyReplicationGroupShardConfiguration
    &ApplyImmediately=true
    &NodeGroupCount=4
    &ReplicationGroupId=my-cluster

 &ReshardingConfiguration.ReshardingConfiguration.1.PreferredAvailabilityZones.AvailabilityZone
east-2a

 &ReshardingConfiguration.ReshardingConfiguration.1.PreferredAvailabilityZones.AvailabilityZone
east-2c
```

```
 &ReshardingConfiguration.ReshardingConfiguration.2.PreferredAvailabilityZones.AvailabilityZone
east-2b

 &ReshardingConfiguration.ReshardingConfiguration.2.PreferredAvailabilityZones.AvailabilityZone
east-2a

 &ReshardingConfiguration.ReshardingConfiguration.3.PreferredAvailabilityZones.AvailabilityZone
east-2c

 &ReshardingConfiguration.ReshardingConfiguration.3.PreferredAvailabilityZones.AvailabilityZone
east-2d

 &ReshardingConfiguration.ReshardingConfiguration.4.PreferredAvailabilityZones.AvailabilityZone
east-2d

 &ReshardingConfiguration.ReshardingConfiguration.4.PreferredAvailabilityZones.AvailabilityZone
east-2c
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20171002T192317Z
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 ElastiCache API 参考中的ModifyReplicationGroupShard配置。

通过在线重新分片功能删除分片

您可以使用、或 ElastiCache API 从 Redis（已启用集群模式）集群中 AWS Management Console移除分片。 AWS CLI

主题

- 删除分片（控制台）
- 删除分片（AWS CLI）
- 移除分片 (ElastiCacheAPI)

删除分片（控制台）

以下过程介绍了如何通过使用 AWS Management Console删除分片来重新配置 Redis（已启用集群模式）集群中的分片。

在从复制组中移除节点组（分片）之前，请 ElastiCache 确保所有数据都适合剩余的分片。如果数据将适合，将根据要求从复制组中删除指定分片。如果数据不适合剩余的节点组，则过程将终止，并且复制组的节点组配置将保留为与发出请求之前相同。

您可以使用从 Redis (已启用集群模式) 集群中移除一个或多个分片。 AWS Management Console 您无法删除某个复制组中的所有分片。而必须删除复制组。有关更多信息，请参阅 删除复制组。以下步骤描述了删除一个或多个分片的过程。

从 Redis（已启用集群模式）集群中删除分片

1. 打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 从导航窗格中，选择 Redis clusters（Redis 集群）。
3. 找到并选择要从中删除分片的 Redis（已启用集群模式）集群的名称，而不是选择集群名称左侧的框。

   > **Tip**
   >
   > Redis（已启用集群模式）集群的 Shards（分片）列中会显示 1 或大于 1 的值。

4. 从分片列表中，选择要删除的每个分片的名称左侧的框。
5. 选择 Delete shard。

删除分片（AWS CLI）

以下过程介绍了如何通过使用 AWS CLI删除分片来重新配置 Redis（已启用集群模式）集群中的分片。

> **Important**
>
> 在从复制组中移除节点组（分片）之前，请 ElastiCache 确保所有数据都适合剩余的分片。如果数据将适合，将根据要求从复制组中删除指定分片（`--node-groups-to-remove`），并将其密钥空间映射到其余分片。如果数据不适合剩余的节点组，则过程将终止，并且复制组的节点组配置将保留为与发出请求之前相同。

您可以使用从 Redis (已启用集群模式) 集群中移除一个或多个分片。 AWS CLI 您无法删除某个复制组中的所有分片。而必须删除复制组。有关更多信息，请参阅 删除复制组。

在`modify-replication-group-shard-configuration`中使用以下参数：

参数

- `--apply-immediately` – 必需。指定分片重新配置操作立即开始。
- `--replication-group-id` – 必需。指定在哪个复制组（集群）上执行分片重新配置操作。
- `--node-group-count` – 必需。指定操作完成时存在的分片（节点组）数量。删除分片后，`--node-group-count` 的值必须小于当前分片数量。
- `--node-groups-to-remove` – 当 `--node-group-count` 小于当前节点组（分片）数量时，此参数为必需。要从复制组中删除的分片（节点组）ID 列表。

以下步骤描述了删除一个或多个分片的过程。

Example - 删除分片

以下示例从 Redis（已启用集群模式）集群 `my-cluster` 中删除 2 个节点组，因此当操作完成后共有 2 个节点组。删除分片的键空间会均匀地分布在其余分片上。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group-shard-configuration \
    --replication-group-id my-cluster \
    --node-group-count 2 \
    --node-groups-to-remove "0002" "0003" \
    --apply-immediately
```

对于 Windows：

```
aws elasticache modify-replication-group-shard-configuration ^
    --replication-group-id my-cluster ^
    --node-group-count 2 ^
    --node-groups-to-remove "0002" "0003" ^
    --apply-immediately
```

移除分片 (ElastiCacheAPI)

您可以通过操作使用 ElastiCache API 在线重新配置 Redis（已启用集群模式）集群中的分片。`ModifyReplicationGroupShardConfiguration`

以下过程介绍如何通过使用 API 移除分片来重新配置 Redis（已启用集群模式）集群中的分片。ElastiCache

> **Important**
>
> 在从复制组中移除节点组（分片）之前，请 ElastiCache 确保所有数据都适合剩余的分片。如果数据将适合，将根据要求从复制组中删除指定分片（NodeGroupsToRemove），并将其密钥空间映射到其余分片。如果数据不适合剩余的节点组，则过程将终止，并且复制组的节点组配置将保留为与发出请求之前相同。

您可以使用 ElastiCache API 从 Redis（已启用集群模式）集群中移除一个或多个分片。您无法删除某个复制组中的所有分片。而必须删除复制组。有关更多信息，请参阅 删除复制组。

在ModifyReplicationGroupShardConfiguration中使用以下参数：

参数

- ApplyImmediately=true – 必需。指定分片重新配置操作立即开始。
- ReplicationGroupId – 必需。指定在哪个复制组（集群）上执行分片重新配置操作。
- NodeGroupCount – 必需。指定操作完成时存在的分片（节点组）数量。删除分片后，NodeGroupCount 的值必须小于当前分片数量。
- NodeGroupsToRemove – 当 --node-group-count 小于当前节点组（分片）数量时，此参数为必需。要从复制组中删除的分片（节点组）ID 列表。

以下步骤描述了删除一个或多个分片的过程。

Example - 删除分片

以下示例从 Redis（已启用集群模式）集群 my-cluster 中删除 2 个节点组，因此当操作完成后共有 2 个节点组。删除分片的键空间会均匀地分布在其余分片上。

```
https://elasticache.us-east-2.amazonaws.com/
    ?Action=ModifyReplicationGroupShardConfiguration
    &ApplyImmediately=true
    &NodeGroupCount=2
    &ReplicationGroupId=my-cluster
    &NodeGroupsToRemove.member.1=0002
    &NodeGroupsToRemove.member.2=0003
    &Version=2015-02-02
    &SignatureVersion=4
```

```
    &SignatureMethod=HmacSHA256
    &Timestamp=20171002T192317Z
    &X-Amz-Credential=<credential>
```

在线分片重新平衡

您可以使用、或 API 重新平衡 Redis（已启用集群模式）集群中的 AWS Management Console分片。AWS CLI ElastiCache

主题

- 在线分片重新平衡（控制台）
- 在线分片重新平衡（AWS CLI）
- 在线分片再平衡 (API) ElastiCache

在线分片重新平衡（控制台）

以下过程介绍了如何通过使用 AWS Management Console重新平衡分片来重新配置 Redis（已启用集群模式）集群中的分片。

在 Redis（已启用集群模式）集群的分片之间重新平衡键空间

1. 打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 从导航窗格中，选择 Redis clusters（Redis 集群）。
3. 选择要重新平衡的 Redis（已启用集群模式）集群的名称，而不是名称左侧的框。

   > **Tip**
   >
   > Redis（已启用集群模式）集群的 Shards（分区）列中会显示 1 或大于 1 的值。

4. 选择 Rebalance。
5. 系统提示时，请选择 Rebalance。您可能会看到一条类似于这样的消息：*################ ########AmazonElasti#######400######InvalidReplicationGroupState###### 2246cebd-9721-11e7-8d5b-e1b0f086* c8c8cf）。如果如此，请选择 Cancel。

在线分片重新平衡（AWS CLI）

在`modify-replication-group-shard-configuration`中使用以下参数：

参数

- `-apply-immediately` – 必需。指定分片重新配置操作立即开始。
- `--replication-group-id` – 必需。指定在哪个复制组（集群）上执行分片重新配置操作。
- `--node-group-count` – 必需。要在集群中的所有分片之间重新平衡键空间，该值必须与当前分片数量相同。

以下过程介绍了如何通过使用 AWS CLI重新平衡分区来重新配置 Redis（已启用集群模式）集群中的分区。

Example - 重新平衡集群中的分片

以下示例演示重新平衡 Redis（已启用集群模式）集群 `my-cluster` 中的槽，以便使槽尽可能均匀分布。`--node-group-count` (4) 的值为集群中的当前分片数量。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group-shard-configuration \
    --replication-group-id my-cluster \
    --node-group-count 4 \
    --apply-immediately
```

对于 Windows：

```
aws elasticache modify-replication-group-shard-configuration ^
    --replication-group-id my-cluster ^
    --node-group-count 4 ^
    --apply-immediately
```

在线分片再平衡 (API) ElastiCache

您可以通过操作使用 ElastiCache API 在线重新配置 Redis（已启用集群模式）集群中的分片。`ModifyReplicationGroupShardConfiguration`

在`ModifyReplicationGroupShardConfiguration`中使用以下参数：

参数

- `ApplyImmediately=true` – 必需。指定分片重新配置操作立即开始。
- `ReplicationGroupId` – 必需。指定在哪个复制组（集群）上执行分片重新配置操作。

- NodeGroupCount – 必需。要在集群中的所有分片之间重新平衡键空间，该值必须与当前分片数量相同。

以下过程介绍如何通过使用 API 重新平衡分片来重新配置 Redis（已启用集群模式）集群中的分片。ElastiCache

Example - 集群重新平衡

以下示例演示重新平衡 Redis（已启用集群模式）集群 my-cluster 中的槽，以便使槽尽可能均匀分布。NodeGroupCount (4) 的值为集群中的当前分片数量。

```
https://elasticache.us-east-2.amazonaws.com/
    ?Action=ModifyReplicationGroupShardConfiguration
    &ApplyImmediately=true
    &NodeGroupCount=4
    &ReplicationGroupId=my-cluster
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20171002T192317Z
    &X-Amz-Credential=<credential>
```

通过修改节点类型来在线纵向扩展

通过使用 Amazon ElastiCache for Redis 版本 3.2.10 或更高版本的在线垂直扩展，您可以在最短的停机时间内动态扩展 Redis 集群。这样，即使在扩展时，Redis 集群也可以处理请求。

> **Note**
>
> 不支持在使用数据分层功能的集群（例如，使用 r6gd 节点类型的集群）和不使用数据分层功能的集群（例如，使用 r6g 节点类型的集群）之间扩缩。有关更多信息，请参阅 数据分层。

您可执行以下操作：

- 纵向扩展 – 通过调整 Redis 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。

  ElastiCache 在保持在线状态并处理请求的同时，动态调整集群的大小。
- 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，在保持在线状态并处理请求的同时， ElastiCache 动态调整集群的大小。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：
>
> - 确保您具有足够的 ENI（弹性网络接口）容量。如果要缩减，请确保较小的节点具有足够的内存来承受预期流量。
>
> 有关内存管理的最佳实践，请参阅 管理预留内存。
> - 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。

目录

在线纵向扩展

主题

纵向扩展 Redis 缓存集群（控制台）

以下过程介绍如何使用 ElastiCache 管理控制台扩展 Redis 集群。在此过程中，Redis 集群将继续处理请求，且停机时间降至最短。

纵向扩展 Redis 集群（控制台）

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 从导航窗格中，选择 Redis clusters（Redis 集群）。
3. 从集群列表中，选择集群。
4. 选择 Modify(修改)。
5. 在 Modify Cluster 向导中：
   - 从 Node type 列表中选择您希望扩展到的节点类型。要扩展，请选择大于现有节点的节点类型。
6. 如果您要立即执行扩展过程，请选中立即应用框。如果 Apply immediately 框处于未选中状态，则在此集群的下一维护时段内执行纵向扩展过程。
7. 选择 Modify(修改)。

   如果您在上一步选择了 Apply immediately，则集群的状态将变为 modifying。当状态变为 available 时，即表示修改完成，您可以开始使用新集群。

纵向扩展 Redis 缓存集群 (AWS CLI)

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

纵向扩展 Redis 缓存集群 (AWS CLI)

1. 通过运行带有以下参数的 AWS CLI `list-allowed-node-type-modifications`命令来确定可以扩展到的节点类型。

   对于 Linux、macOS 或 Unix：

   ```
   aws elasticache list-allowed-node-type-modifications \
        --replication-group-id my-replication-group-id
   ```

   对于 Windows：

   ```
   aws elasticache list-allowed-node-type-modifications ^
        --replication-group-id my-replication-group-id
   ```

以上命令的输出类似于此处所示（JSON 格式）。

```
{
     "ScaleUpModifications": [
         "cache.m3.2xlarge",
         "cache.m3.large",
         "cache.m3.xlarge",
         "cache.m4.10xlarge",
         "cache.m4.2xlarge",
         "cache.m4.4xlarge",
         "cache.m4.large",
         "cache.m4.xlarge",
         "cache.r3.2xlarge",
         "cache.r3.4xlarge",
         "cache.r3.8xlarge",
         "cache.r3.large",
         "cache.r3.xlarge"
     ]
        "ScaleDownModifications": [
         "cache.t2.micro",
         "cache.t2.small ",
         "cache.t2.medium",
         "cache.t1.small "
     ],
}
```

有关更多信息，请参阅 AWS CLI 参考中的 list-allowed-node-type-modifications。

2. 使用 AWS CLI `modify-replication-group`命令和以下参数修改您的复制组以向上扩展到新的更大的节点类型。

   - `--replication-group-id` – 要纵向扩展到的复制组的名称。
   - `--cache-node-type` – 要扩展缓存集群的新节点类型。此值必须是步骤 1 中由 `list-allowed-node-type-modifications` 命令返回的节点类型之一。
   - `--cache-parameter-group-name` – [可选] 如果您使用 `reserved-memory` 管理集群的预留内存，则使用此参数。指定为您的新节点类型预留正确内存量的自定义缓存参数组。如果您在使用 `reserved-memory-percent`，则可以忽略此参数。
   - `--apply-immediately` – 使纵向扩展过程立即得到应用。要将纵向扩展流程推迟到此集群的下一维护时段，请使用 `--no-apply-immediately` 参数。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group  \
     --replication-group-id my-redis-cluster \
     --cache-node-type cache.m3.xlarge \
     --apply-immediately
```

对于 Windows：

```
aws elasticache modify-replication-group ^
     --replication-group-id my-redis-cluster ^
     --cache-node-type cache.m3.xlarge ^
     --apply-immediately
```

以上命令的输出类似于此处所示（JSON 格式）。

```
{
  "ReplicationGroup": {
        "Status": "modifying",
        "Description": "my-redis-cluster",
        "NodeGroups": [
            {
                "Status": "modifying",
                "Slots": "0-16383",
                "NodeGroupId": "0001",
                "NodeGroupMembers": [
                    {
                        "PreferredAvailabilityZone": "us-east-1f",
                        "CacheNodeId": "0001",
                        "CacheClusterId": "my-redis-cluster-0001-001"
                    },
                    {
                        "PreferredAvailabilityZone": "us-east-1d",
                        "CacheNodeId": "0001",
                        "CacheClusterId": "my-redis-cluster-0001-002"
                    }
                ]
            }
        ],
```

```
        "ConfigurationEndpoint": {
            "Port": 6379,
            "Address": "my-redis-
cluster.r7gdfi.clustercfg.use1.cache.amazonaws.com"
        },
        "ClusterEnabled": true,
        "ReplicationGroupId": "my-redis-cluster",
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "enabled",
        "SnapshotWindow": "07:30-08:30",
        "MemberClusters": [
            "my-redis-cluster-0001-001",
            "my-redis-cluster-0001-002"
        ],
        "CacheNodeType": "cache.m3.xlarge",
         "DataTiering": "disabled"
        "PendingModifiedValues": {}
    }
}
```

有关更多信息，请参阅 AWS CLI 参考中的 modify-replication-group。

3. 如果您使用了`--apply-immediately`，请使用带有以下参数的 AWS CLI `describe-cache-clusters`命令检查缓存集群的状态。当状态变为 available 时，您便可开始使用较大的新缓存集群节点。

扩展 Redis 缓存集群 (ElastiCache API)

以下过程使用 ElastiCache API 将您的缓存集群从其当前节点类型扩展到新的更大的节点类型。在此过程中， ElastiCache for Redis 会更新 DNS 条目，使其指向新节点。因此，您便不需要更新应用程序中的终端节点。对于 Redis 5.0.5 及更高版本，您可以在该集群继续保持在线并处理传入请求时扩展启用自动故障转移的集群。在版本 4.0.10 及更低版本上，更新 DNS 条目时，您可能会发现先前版本上来自主节点的读取和写入短暂中断。

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缓存集群中的数据量不同而异。

扩展 Redis 缓存集群 (ElastiCache API)

1. 使用带有以下参数的 ElastiCache API `ListAllowedNodeTypeModifications` 操作来确定可以扩展到哪些节点类型。

   - `ReplicationGroupId` – 复制组的名称。使用此参数可描述特定复制组而非所有复制组。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ListAllowedNodeTypeModifications
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 Amazon ElastiCache API 参考ListAllowedNodeTypeModifications中的。

2. 使用 ModifyReplicationGroup ElastiCache API 操作并使用以下参数将当前的复制组扩展到新的节点类型。

   - ReplicationGroupId – 复制组的名称。
   - CacheNodeType – 此复制组中缓存集群的较大新节点类型。此值必须是步骤 1 中由 ListAllowedNodeTypeModifications 操作返回的实例类型之一。
   - CacheParameterGroupName – [可选] 如果您使用 reserved-memory 管理集群的预留内存，则使用此参数。指定为您的新节点类型预留正确内存量的自定义缓存参数组。如果您在使用 reserved-memory-percent，则可以忽略此参数。
   - ApplyImmediately – 设置为 true 可促使立即应用纵向扩展流程。要将扩展流程推迟到下一维护时段，请使用 ApplyImmediately=false。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyReplicationGroup
    &ApplyImmediately=true
    &CacheNodeType=cache.m3.2xlarge
    &CacheParameterGroupName=redis32-m3-2xl
    &ReplicationGroupId=myReplGroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
```

```
    &X-Amz-Signature=<signature>
```

有关更多信息，请参阅 Amazon ElastiCache API 参考ModifyReplicationGroup中的。

3. 如果您使用了 `ApplyImmediately=true`，请使用带有以下参数的 ElastiCache API `DescribeReplicationGroups` 操作监控复制组的状态。当状态从 modifying 变为 available 时，您便可开始写入已扩展的新复制组。

   - `ReplicationGroupId` – 复制组的名称。使用此参数可描述特定复制组而非所有复制组。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReplicationGroups
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 Amazon ElastiCache API 参考DescribeReplicationGroups中的。

在线缩减

主题

- 缩减 Redis 缓存集群（控制台）
- 缩减 Redis 缓存集群 (AWS CLI)
- 缩小 Redis 缓存集群规模 (ElastiCache API)

缩减 Redis 缓存集群（控制台）

以下过程介绍如何使用 ElastiCache 管理控制台缩小 Redis 集群。在此过程中，Redis 集群将继续处理请求，且停机时间降至最短。

缩减 Redis 集群（控制台）

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 从导航窗格中，选择 Redis clusters（Redis 集群）。

3. 从集群列表中，选择首选集群。
4. 选择 Modify(修改)。
5. 在 Modify Cluster 向导中：
   - 从 Node type 列表中选择您希望扩展到的节点类型。要缩减，请选择小于现有节点的节点类型。请注意，并不是可缩减到所有节点类型。
6. 如果您要立即执行缩减过程，请选中立即应用框。如果立即应用框处于未选中状态，则在此集群的下一维护时段内执行缩减过程。
7. 选择 Modify(修改)。

   如果您在上一步选择了 Apply immediately，则集群的状态将变为 modifying。当状态变为 available 时，即表示修改完成，您可以开始使用新集群。

缩减 Redis 缓存集群 (AWS CLI)

以下过程介绍如何使用 AWS CLI缩减 Redis 缓存集群。在此过程中，Redis 集群将继续处理请求，且停机时间降至最短。

缩减 Redis 缓存集群 (AWS CLI)

1. 通过运行带有以下参数的 AWS CLI `list-allowed-node-type-modifications`命令来确定可以缩减到的节点类型。

   对于 Linux、macOS 或 Unix：

   ```
   aws elasticache list-allowed-node-type-modifications \
        --replication-group-id my-replication-group-id
   ```

   对于 Windows：

   ```
   aws elasticache list-allowed-node-type-modifications ^
        --replication-group-id my-replication-group-id
   ```

   以上命令的输出类似于此处所示（JSON 格式）。

   ```
   {
        "ScaleUpModifications": [
            "cache.m3.2xlarge",
            "cache.m3.large",
   ```

```
          "cache.m3.xlarge",
          "cache.m4.10xlarge",
          "cache.m4.2xlarge",
          "cache.m4.4xlarge",
          "cache.m4.large",
          "cache.m4.xlarge",
          "cache.r3.2xlarge",
          "cache.r3.4xlarge",
          "cache.r3.8xlarge",
          "cache.r3.large",
          "cache.r3.xlarge"
     ]

        "ScaleDownModifications": [
          "cache.t2.micro",
          "cache.t2.small ",
          "cache.t2.medium ",
          "cache.t1.small"
     ]
}
```

有关更多信息，请参阅 AWS CLI 参考中的 list-allowed-node-type-modifications。

2. 使用 AWS CLI `modify-replication-group`命令和以下参数修改您的复制组以缩小到新的较小节点类型。

   - `--replication-group-id` – 要缩减到的复制组的名称。
   - `--cache-node-type` – 要扩展缓存集群的新节点类型。此值必须是步骤 1 中由 `list-allowed-node-type-modifications` 命令返回的节点类型之一。
   - `--cache-parameter-group-name` – [可选] 如果您使用 `reserved-memory` 管理集群的预留内存，则使用此参数。指定为您的新节点类型预留正确内存量的自定义缓存参数组。如果您在使用 `reserved-memory-percent`，则可以忽略此参数。
   - `--apply-immediately` – 使纵向扩展过程立即得到应用。要将收缩流程推迟到此集群的下一维护时段，请使用 `--no-apply-immediately` 参数。

   对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group  \
     --replication-group-id my-redis-cluster \
     --cache-node-type cache.t2.micro \
```

```
    --apply-immediately
```

对于 Windows：

```
aws elasticache modify-replication-group ^
    --replication-group-id my-redis-cluster ^
    --cache-node-type cache.t2.micro ^
    --apply-immediately
```

以上命令的输出类似于此处所示（JSON 格式）。

```
{
  "ReplicationGroup": {
        "Status": "modifying",
        "Description": "my-redis-cluster",
        "NodeGroups": [
            {
                "Status": "modifying",
                "Slots": "0-16383",
                "NodeGroupId": "0001",
                "NodeGroupMembers": [
                    {
                        "PreferredAvailabilityZone": "us-east-1f",
                        "CacheNodeId": "0001",
                        "CacheClusterId": "my-redis-cluster-0001-001"
                    },
                    {
                        "PreferredAvailabilityZone": "us-east-1d",
                        "CacheNodeId": "0001",
                        "CacheClusterId": "my-redis-cluster-0001-002"
                    }
                ]
            }
        ],
        "ConfigurationEndpoint": {
            "Port": 6379,
            "Address": "my-redis-
cluster.r7gdfi.clustercfg.use1.cache.amazonaws.com"
        },
        "ClusterEnabled": true,
        "ReplicationGroupId": "my-redis-cluster",
```

```
        "SnapshotRetentionLimit": 1,
        "AutomaticFailover": "enabled",
        "SnapshotWindow": "07:30-08:30",
        "MemberClusters": [
            "my-redis-cluster-0001-001",
            "my-redis-cluster-0001-002"
        ],
        "CacheNodeType": "cache.t2.micro",
         "DataTiering": "disabled"
        "PendingModifiedValues": {}
    }
}
```

有关更多信息，请参阅 AWS CLI 参考中的 modify-replication-group。

3. 如果您使用了`--apply-immediately`，请使用带有以下参数的 AWS CLI `describe-cache-clusters`命令检查缓存集群的状态。当状态变为 available 时，您便可开始使用较小的新缓存集群节点。

缩小 Redis 缓存集群规模 (ElastiCache API)

以下过程使用 ElastiCache API 将您的复制组从其当前节点类型扩展到新的较小节点类型。在此过程中，Redis 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缓存集群中的数据量而异。

缩小规模 (ElastiCache API)

1. 使用带有以下参数的 ElastiCache API `ListAllowedNodeTypeModifications` 操作来确定可以缩减为哪些节点类型。

   - `ReplicationGroupId` – 复制组的名称。使用此参数可描述特定复制组而非所有复制组。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ListAllowedNodeTypeModifications
    &ReplicationGroupId=MyReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 Amazon ElastiCache API 参考ListAllowedNodeTypeModifications中的。

2. 使用 ModifyReplicationGroup ElastiCache API 操作并使用以下参数将当前的复制组缩小到新的节点类型。

   - ReplicationGroupId – 复制组的名称。
   - CacheNodeType – 此复制组中缓存集群的较小新节点类型。此值必须是步骤 1 中由 ListAllowedNodeTypeModifications 操作返回的实例类型之一。
   - CacheParameterGroupName – [可选] 如果您使用 reserved-memory 管理集群的预留内存，则使用此参数。指定为您的新节点类型预留正确内存量的自定义缓存参数组。如果您在使用 reserved-memory-percent，则可以忽略此参数。
   - ApplyImmediately – 设置为 true 可促使立即应用缩减流程。要将缩减流程推迟到下一维护时段，请使用 ApplyImmediately=false。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyReplicationGroup
    &ApplyImmediately=true
    &CacheNodeType=cache.t2.micro
    &CacheParameterGroupName=redis32-m3-2xl
    &ReplicationGroupId=myReplGroup
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

有关更多信息，请参阅 Amazon ElastiCache API 参考ModifyReplicationGroup中的。

开始在 ElastiCache for Redis 中使用 JSON

ElastiCache for Redis 支持原生 JavaScript 对象表示法（JSON）格式，这是在 Redis 集群中对复杂数据集进行编码的一种简单的无 Schema 的方法。您可以使用 JavaScript 对象表示法（JSON）格式在

Redis 集群中进行数据的本地存储和访问，并更新在这些集群中存储的 JSON 数据 ， 无需管理自定义代码来对其进行序列化和反序列化。

除了对通过 JSON 运行的应用程序使用 Redis API 操作之外，您现在还可以有效地检索和更新 JSON 文档的特定部分，而无需对整个对象进行操作。这可以提高性能并降低成本。您还可以使用 Goessner 样式的 `JSONPath` 查询来搜索您的 JSON 文档内容。

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

与 JSON 相关的指标 `JsonBasedCmds` 和 `JsonBasedCmdsLatency` 合并到 CloudWatch 中，以监控此数据类型的使用情况。有关更多信息，请参阅 Redis 的指标。

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

主题

Redis JSON 数据类型概述

ElastiCache for Redis 支持许多用于处理 JSON 数据类型的 Redis 命令。以下是 JSON 数据类型的概述和支持的 Redis 命令的详细列表。

术语

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

支持的 JSON 标准

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

根元素

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

文档大小限制

JSON 文档以针对快速访问和修改而优化的格式在内部存储。此格式通常会导致比同一文档的等效序列化表示使用稍多的内存。

单个 JSON 文档的内存使用限制为 64 MB，这是内存中数据结构的大小，而不是 JSON 字符串的大小。您可以使用 `JSON.DEBUG MEMORY` 命令检查 JSON 文档所使用的内存量。

JSON ACL

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

ACL

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

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

嵌套深度限制

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

命令语法

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

表示法：

- 必需的参数括在尖括号内。例如：<key>
- 可选的参数括在方括号内。例如：[path]
- 其他可选参数由省略号（“…”）来表示。例如：[json ...]

路径语法

Redis JSON 支持两种路径语法：

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

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

如果查询路径以“$”开头，则使用的是增强的语法。否则使用的是受限的语法。

增强的语法

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

符号/表达式	描述
==	等于，用于筛选表达式。
!=	不等于，用于筛选表达式。
>	大于，用于筛选表达式。
>=	大于或等于，用于筛选表达式。
<	小于，用于筛选表达式。
<=	小于或等于，用于筛选表达式。
&&	逻辑 AND，用于组合多个筛选表达式。
||	L逻辑 OR，用于组合多个筛选表达式。

示例

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

```
{ "store": {
    "book": [
      { "category": "reference",
        "author": "Nigel Rees",
        "title": "Sayings of the Century",
        "price": 8.95,
        "in-stock": true,
        "sold": true
      },
      { "category": "fiction",
        "author": "Evelyn Waugh",
        "title": "Sword of Honour",
        "price": 12.99,
        "in-stock": false,
        "sold": true
      },
      { "category": "fiction",
        "author": "Herman Melville",
        "title": "Moby Dick",
        "isbn": "0-553-21311-3",
```

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

路径	描述
$.store.book[*].author	商店中所有书籍的作者。
$..author	所有作者。
$.store.*	商店的所有成员。
$["store"].*	商店的所有成员。
$.store..price	商店中所有商品的价格。
$..*	JSON 结构的所有递归成员。
$..book[*]	所有书籍。
$..book[0]	第一本书籍。
$..book[-1]	最后一本书籍。

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

其他筛选表达式示例：

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

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

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

受限的语法

符号/表达式	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	第一本书籍的作者。
.store.book[-1].author	最后一本书籍的作者。
.address.city	城市名称。
["store"]["book"][0]["title"]	第一本书籍的书名。
["store"]["book"][-1]["title"]	最后一本书籍的书名。

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

常见错误前缀

每条错误消息均有一个前缀。以下是常见错误前缀的列表。

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

JSON 相关指标

提供了以下 JSON 信息指标：

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

要查询核心指标，请运行以下 Redis 命令：

```
info json_core_metrics
```

ElastiCache for Redis 如何与 JSON 交互

以下部分介绍了 ElastiCache for Redis 如何与 JSON 数据类型交互。

运算符优先顺序

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

最大路径嵌套限制行为

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

处理数字值

JSON 没有针对整数和浮点数的单独数据类型。它们都被称为数字。

数字的表示形式：

当在输入时接收 JSON 数字时，它会转换为两种内部二进制表示之一：64 位有符号整数或 64 位 IEEE 双精度浮点。不保留原始字符串及其所有格式。因此，当数字作为 JSON 响应的一部分输出时，它会从内部二进制表示转换为使用通用格式规则的可打印字符串。这些规则可能会导致生成的字符串与收到的字符串不同。

算术命令 NUMINCRBY 和 NUMMULTBY：

- 如果两个数字都是整数并且结果超出 int64 的范围，则它会自动变成一个 64 位 IEEE 双精度浮点数。
- 如果至少有一个数字是浮点，则结果是 64 位 IEEE 双精度浮点数。
- 如果结果超出 64 位 IEEE 双精度值的范围，则该命令将返回 OVERFLOW 错误。

有关可用命令的详细列表，请参阅 支持的 Redis JSON 命令。

直接数组筛选

ElastiCache for Redis 可直接筛选数组对象。

对于像 [0,1,2,3,4,5,6] 这样的数据和像 $[?(@<4)] 这样的路径查询，或者像 {"my_key": [0,1,2,3,4,5,6]} 这样的数据和像 $.my_key[?(@<4)] 这样的路径查询，在这两种情况下，ElastiCache for Redis 都会返回 [1,2,3]。

数组索引行为

ElastiCache for Redis 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。

为确保客户行为可预测，ElastiCache for Redis 不会向下或向上舍入数组索引，因此如果您有一个长度为 5 的数组，则调用索引 5 及更高或 -6 及更低将不会产生结果。

严格语法评估

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

支持的 Redis JSON 命令

ElastiCache for Redis 支持 以下 Redis JSON 命令：

主题

- JSON.ARRAPPEND
- JSON.ARRINDEX

- JSON.ARRINSERT
- JSON.ARRLEN
- JSON.ARRPOP
- JSON.ARRTRIM
- JSON.CLEAR
- JSON.DEBUG
- JSON.DEL
- JSON.FORGET
- JSON.GET
- JSON.MGET
- JSON.NUMINCRBY
- JSON.NUMMULTBY
- JSON.OBJLEN
- JSON.OBJKEYS
- JSON.RESP
- JSON.SET
- JSON.STRAPPEND
- JSON.STRLEN
- JSON.TOGGLE
- JSON.TYPE

JSON.ARRAPPEND

将一个或多个值附加到路径上的数组值。

语法

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

- key（必需）– JSON 文档类型的 Redis 键。
- path（必需）– 一个 JSON 路径。

- json（必需）– 要附加到数组的 JSON 值。

Return

如果路径是增强的语法：

- 表示每个路径处数组的新长度的整数数组。
- 如果值不是数组，则其对应的返回值为 Null。
- 如果输入 json 参数之一不是有效的 JSON 字符串，则为 SYNTAXERR 错误。
- 如果路径不存在，则为 NONEXISTENT 错误。

如果路径是受限的语法：

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

示例

增强的路径语法：

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

受限的路径语法：

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

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

JSON.ARRINDEX

搜索标量 JSON 值在路径的数组中的首次出现。

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

语法

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

- key（必需）– JSON 文档类型的 Redis 键。
- path（必需）– 一个 JSON 路径。
- json-scalar（必需）- 要搜索的标量值。JSON 标量是指不是对象或数组的值。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（不含）。如果未提供，则默认为 0，这意味着包含最后一个元素。0 或 -1 意味着包含最后一个元素。

Return

如果路径是增强的语法：

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

如果路径是受限的语法：

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

示例

增强的路径语法：

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

受限的路径语法：

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

JSON.ARRINSERT

将一个或多个值插入到索引之前路径处的数组值中。

语法

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

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

Return

如果路径是增强的语法：

- 表示每个路径处数组的新长度的整数数组。

- 如果值为空数组，则其对应的返回值为 Null。
- 如果值不是数组，则其对应的返回值为 Null。
- 如果索引参数超出界限，则为 OUTOFBOUNDARIES 错误。

如果路径是受限的语法：

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

示例

增强的路径语法：

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

受限的路径语法：

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

JSON.ARRLEN

获取路径中数组值的长度。

语法

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

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

Return

如果路径是增强的语法：

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

如果路径是受限的语法：

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

示例

增强的路径语法：

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

127.0.0.1:6379> JSON.SET k2 . '[[], "a", ["a", "b"], ["a", "b", "c"], 4]'
```

```
OK
127.0.0.1:6379> JSON.ARRLEN k2 $[*]
1) (integer) 0
2) (nil)
3) (integer) 2
4) (integer) 3
5) (nil)
```

受限的路径语法：

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

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

JSON.ARRPOP

从数组中删除并返回索引处的元素。弹出空数组会返回 Null。

语法

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

- key（必需）– JSON 文档类型的 Redis 键。
- path（可选）– 一个 JSON 路径。如果未提供，则默认为根目录。
- index（可选）– 数组中要开始弹出的位置。
  - 如果未提供，则默认为 -1，这表示最后一个元素。

- 负值表示距离最后一个元素的位置。
- 超出边界的索引会舍入到其各自的数组边界。

Return

如果路径是增强的语法：

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

如果路径是受限的语法：

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

示例

增强的路径语法：

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

受限的路径语法：

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

```
"[[],[\"a\"]]"

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

JSON.ARRTRIM

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

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

语法

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

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

Return

如果路径是增强的语法：

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

如果路径是受限的语法：

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

示例

增强的路径语法：

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

受限的路径语法：

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

JSON.CLEAR

清除路径中的数组或对象。

语法

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

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

Return

- 整数，已清除的容器数量。
- 清除空数组或对象会导致清除 1 个容器。
- 清除非容器值会返回 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：

- MEMORY <key> [path] – 报告 JSON 值的内存使用量（以字节为单位）。如果未提供，则路径默认为根目录。
- FIELDS <key> [path] – 报告指定文档路径中的字段数。如果未提供，则路径默认为根目录。每个非容器 JSON 值均计为一个字段。对象和数组以递归方式为其包含的每个 JSON 值计数一个字段。除根容器外，每个容器值均计为一个额外字段。
- HELP – 打印命令的帮助消息。

语法

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

取决于子命令：

MEMORY

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

FIELDS

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

HELP – 返回一个帮助消息数组。

示例

增强的路径语法：

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

```
6) (integer) 16
7) (integer) 16
8) (integer) 50
9) (integer) 64
127.0.0.1:6379> JSON.DEBUG FIELDS k1 $[*]
1) (integer) 1
2) (integer) 1
3) (integer) 1
4) (integer) 1
5) (integer) 1
6) (integer) 0
7) (integer) 0
8) (integer) 2
9) (integer) 3
```

受限的路径语法：

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

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

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

JSON.DEL

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

语法

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

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

Return

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

示例

增强的路径语法：

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

受限的路径语法：

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

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

JSON.FORGET

JSON.DEL 的别名。

JSON.GET

返回一个或多个路径的序列化 JSON。

语法

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

- key（必需）– JSON 文档类型的 Redis 键。
- 缩进/换行/空格（可选）– 控制返回的 JSON 字符串的格式，也就是“漂亮的打印格式”。每个字符串的默认值均为空字符串。他们在任意组合中都可被覆盖。可以按任意顺序指定这些字符串。
- NOESCAPE – 可选，允许存在以实现与旧版的兼容性，且没有其他影响。
- path（可选）– 零个或多个 JSON 路径，如果没有给出，则默认为根目录。路径参数必须放在末尾。

Return

增强的路径语法：

如果给出了一条路径：

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

如果给出了多条路径：

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

示例

增强的路径语法：

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

受限的路径语法：

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

```
"{\n\t\"street\": \"21 2nd Street\",\n\t\"city\": \"New York\",\n\t\"state\": \"NY\",\n
\t\"zipcode\": \"10021-3100\"\n}"
127.0.0.1:6379> JSON.GET k1 .firstName .lastName .age
"{\".firstName\":\"John\",\".lastName\":\"Smith\",\".age\":27}"
```

JSON.MGET

从多个文档键获取路径的序列化 JSON。对于不存在的键或 JSON 路径，将返回 Null。

语法

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

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

Return

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

示例

增强的路径语法：

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

受限的路径语法：

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

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

JSON.NUMINCRBY

将路径上的数字值增加给定的数字。

语法

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

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

Return

如果路径是增强的语法：

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

- 如果文档键不存在，则为 NONEXISTENT。

如果路径是受限的语法：

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

示例

增强的路径语法：

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

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

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

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

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

受限的路径语法：

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

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

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

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

127.0.0.1:6379> JSON.SET k3 . '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMINCRBY k3 .a.* 1
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMINCRBY k3 .b.* 1
"2"
127.0.0.1:6379> JSON.NUMINCRBY k3 .c.* 1
(error) WRONGTYPE JSON element is not a number
127.0.0.1:6379> JSON.NUMINCRBY k3 .d.* 1
"4"
```

JSON.NUMMULTBY

将路径上的数值乘以给定的数字。

语法

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

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

Return

如果路径是增强的语法：

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

如果路径是受限的语法：

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

示例

增强的路径语法：

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

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

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

127.0.0.1:6379> JSON.SET k3 $ '{"a":{"a":"a"}, "b":{"a":"a", "b":1}, "c":{"a":"a",
 "b":"b"}, "d":{"a":1, "b":"b", "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k3 $.a.* 2
"[null]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.b.* 2
"[null,2]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.c.* 2
"[null,null]"
127.0.0.1:6379> JSON.NUMMULTBY k3 $.d.* 2
"[2,null,6]"
```

受限的路径语法：

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

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

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

127.0.0.1:6379> JSON.SET k2 . '{"a":{}, "b":{"a":1}, "c":{"a":1, "b":2}, "d":{"a":1,
 "b":2, "c":3}}'
OK
127.0.0.1:6379> JSON.NUMMULTBY k2 .a.* 2
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.NUMMULTBY k2 .b.* 2
"2"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":1,\"b\":2},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k2 .c.* 2
"4"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":4},\"d\":{\"a\":1,\"b\":2,\"c\":3}}"
127.0.0.1:6379> JSON.NUMMULTBY k2 .d.* 2
"6"
127.0.0.1:6379> JSON.GET k2
"{\"a\":{},\"b\":{\"a\":2},\"c\":{\"a\":2,\"b\":4},\"d\":{\"a\":2,\"b\":4,\"c\":6}}"

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

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

JSON.OBJLEN

获取路径对象值中的键数。

语法

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

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

Return

如果路径是增强的语法：

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

如果路径是受限的语法：

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

- 如果文档键不存在，则为 Null。

示例

增强的路径语法：

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

受限的路径语法：

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

```
(integer) 0
127.0.0.1:6379> JSON.OBJLEN k1 .a.*
(error) NONEXISTENT JSON path does not exist
127.0.0.1:6379> JSON.OBJLEN k1 .b
(integer) 1
127.0.0.1:6379> JSON.OBJLEN k1 .b.*
(error) WRONGTYPE JSON element is not an object
127.0.0.1:6379> JSON.OBJLEN k1 .c
(integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 .c.*
(error) WRONGTYPE JSON element is not an object
127.0.0.1:6379> JSON.OBJLEN k1 .d
(integer) 3
127.0.0.1:6379> JSON.OBJLEN k1 .d.*
(integer) 2
127.0.0.1:6379> JSON.OBJLEN k1 .*
(integer) 0
```

JSON.OBJKEYS

获取路径对象值中的键名。

语法

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

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

Return

如果路径是增强的语法：

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

如果路径是受限的语法：

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

示例

增强的路径语法：

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

受限的路径语法：

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

JSON.RESP

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

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

语法

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

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

Return

如果路径是增强的语法：

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

如果路径是受限的语法：

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

示例

增强的路径语法：

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

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

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

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

127.0.0.1:6379> JSON.RESP k1 $.phoneNumbers[*]
1) 1) {
   2) 1) "type"
      2) "home"
```

```
   3) 1) "number"
      2) "212 555-1234"
2) 1) {
   2) 1) "type"
      2) "office"
   3) 1) "number"
      2) "555 555-4567"
```

受限的路径语法：

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

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

127.0.0.1:6379> JSON.RESP k1
 1) {
 2) 1) "firstName"
    2) "John"
 3) 1) "lastName"
    2) "Smith"
 4) 1) "age"
    2) (integer) 27
 5) 1) "weight"
    2) "135.25"
 6) 1) "isAlive"
    2) true
 7) 1) "address"
```

```
    2) 1) {
       2) 1) "street"
          2) "21 2nd Street"
       3) 1) "city"
          2) "New York"
       4) 1) "state"
          2) "NY"
       5) 1) "zipcode"
          2) "10021-3100"
 8) 1) "phoneNumbers"
    2) 1) [
       2) 1) {
          2) 1) "type"
             2) "home"
          3) 1) "number"
             2) "212 555-1234"
       3) 1) {
          2) 1) "type"
             2) "office"
          3) 1) "number"
             2) "555 555-4567"
 9) 1) "children"
    2) 1) [
10) 1) "spouse"
    2) (nil)
```

JSON.SET

在路径中设置 JSON 值。

如果路径调用对象成员：

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

如果路径调用数组索引：

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

如果路径调用对象或数组，该值（对象或数组）将被新的 JSON 值替换。

语法

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

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

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

Return

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

示例

增强的路径语法：

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

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

受限的路径语法：

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

JSON.STRAPPEND

将字符串附加到路径的 JSON 字符串。

语法

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

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

Return

如果路径是增强的语法：

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

如果路径是受限的语法：

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

示例

增强的路径语法：

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

受限的路径语法：

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

JSON.STRLEN

获取路径中 JSON 字符串值的长度。

语法

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

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

Return

如果路径是增强的语法：

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

如果路径是受限的语法：

- 整数，该字符串的长度。

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

示例

增强的路径语法：

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

受限的路径语法：

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

```
(integer) 1
```

JSON.TOGGLE

在路径的 true 和 false 之间切换布尔值。

语法

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

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

Return

如果路径是增强的语法：

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

如果路径是受限的语法：

- 表示结果布尔值的字符串 ("true"/"false")。
- 如果文档键不存在，则为 NONEXISTENT。
- 如果路径中的值不是布尔值，则为 WRONGTYPE 错误。

示例

增强的路径语法：

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

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

受限的路径语法：

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

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

JSON.TYPE

报告给定路径的值类型。

语法

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

- key（必需）– JSON 文档类型的 Redis 键。

- path（可选）– 一个 JSON 路径。如果未提供，则默认为根目录。

Return

如果路径是增强的语法：

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

如果路径是受限的语法：

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

示例

增强的路径语法：

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

受限的路径语法：

```
127.0.0.1:6379> JSON.SET k1 .
 '{"firstName":"John","lastName":"Smith","age":27,"weight":135.25,"isAlive":true,"address":
{"street":"21 2nd Street","city":"New
 York","state":"NY","zipcode":"10021-3100"},"phoneNumbers":
```

```
[{"type":"home","number":"212 555-1234"},{"type":"office","number":"646
 555-4567"}],"children":[],"spouse":null}'
OK
127.0.0.1:6379> JSON.TYPE k1
object
127.0.0.1:6379> JSON.TYPE k1 .children
array
127.0.0.1:6379> JSON.TYPE k1 .firstName
string
127.0.0.1:6379> JSON.TYPE k1 .age
integer
127.0.0.1:6379> JSON.TYPE k1 .weight
number
127.0.0.1:6379> JSON.TYPE k1 .isAlive
boolean
127.0.0.1:6379> JSON.TYPE k1 .spouse
null
```

标记 ElastiCache 资源

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

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

标签基本知识

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

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

标签对 ElastiCache 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。此外，如果添加或删除复制组的标签，则也将向该复制组中的所有节点添加或删除其标签。

您可以使用 AWS Management Console、AWS CLI 和 ElastiCache API 处理标签。

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

您可以为之添加标签的资源

您可以标记账户中已存在的大多数 ElastiCache 资源。下表列出了支持标记的资源。如果您使用的是 AWS 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。控制台可能根据 Name（名称）标签对资源进行组织，但此标签对 ElastiCache 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。

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

- CreateServerlessCache
- CreateCacheCluster
- CreateReplicationGroup
- CopyServerlessCacheSnapshot
- CopySnapshot
- CreateCacheParameterGroup
- CreateCacheSecurityGroup
- CreateCacheSubnetGroup
- CreateServerlessCacheSnapshot
- CreateSnapshot

- `CreateUserGroup`
- `CreateUser`
- `PurchaseReservedCacheNodesOffering`

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

支持用于 ElastiCache 资源的标签

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

支持标签	支持在创建时标记
是	是
是	是

> **Note**
>
> 无法标记全局数据存储。

对于支持在创建时标记的 ElastiCache 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。

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。

标记缓存和快照

以下规则适用于请求操作中的标记：

- CreateReplicationGroup：
  - 如果请求中包含 `--primary-cluster-id` 和 `--tags` 参数，则会向复制组添加请求标签且标签会传播到复制组中的所有缓存群集。如果主缓存群集具有现有标签，则这些标签将被请求标签覆盖，以便所有节点上的标签保持一直。

    如果没有请求标签，则主缓存群集标记将添加到复制组并传播到所有缓存群集。
  - 如果提供了 `--snapshot-name` 或 `--serverless-cache-snapshot-name`：

    如果请求中包含标签，则仅使用这些标签对复制组进行标记。如果请求中未包含任何标签，则快照标签将添加到复制组。
  - 如果提供了 `--global-replication-group-id`：

    如果请求中包含标签，则请求标签将添加到复制组并传播到所有缓存群集。

- CreateCacheCluster：
  - 如果提供了 `--replication-group-id`：

    如果请求中包含标签，则仅使用这些标签对缓存群集进行标记。如果请求中未包含任何标签，则缓存群集将继承复制组标签，而不是主缓存群集的标签。
  - 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缓存群集进行标记。如果请求中未包含任何标签，则将向缓存群集添加快照标签。
- CreateServerlessCache：
  - 如果请求中包含标签，则仅将请求标签添加到无服务器缓存。
- CreateSnapshot：
  - 如果提供了 `--replication-group-id`：

    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，则复制组标签将添加到快照。
  - 如果提供了 `--cache-cluster-id`：

    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，则缓存群集标签将添加到快照。
  - 自动快照：

    标签将传播自复制组标签。
- CreateServerlessCacheSnapshot：
  - 如果请求中包含标签，则仅将请求标签添加到无服务器缓存快照。
- 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。
- CopyServerlessCacheSnapshot：
  - 如果请求中包含标签，则仅将请求标签添加到无服务器缓存快照。
- AddTagsToResource 和 RemoveTagsFromResource：
  - 向复制组添加标签或删除其标签，并将此操作传播到复制组中的所有集群。

> **Note**
>
> AddTagsToResource 和 RemoveTagsFromResource 不能用于默认参数和安全组。

- IncreaseReplicaCount 和 ModifyReplicationGroupShardConfiguration：
  - 对添加到复制组的所有新集群应用与复制组相同的标签。

标签限制

下面是适用于标签的基本限制：

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

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

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

标记资源示例

- 使用标签创建无服务器缓存

```
aws elasticache create-serverless-cache \
    --serverless-cache-name CacheName \
    --engine redis
    --tags Key="Cost Center", Value="1110001" Key="project",Value="XYZ"
```

- 向无服务器缓存添加标签

```
aws elasticache add-tags-to-resource \
--resource-name arn:aws:elasticache:us-east-1:111111222233:serverlesscache:my-cache \
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

- 向复制组添加标签。

```
aws elasticache add-tags-to-resource \
--resource-name arn:aws:elasticache:us-east-1:111111222233:replicationgroup:my-rg \
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

- 使用标签创建缓存群集。

```
aws elasticache create-cache-cluster \
--cluster-id testing-tags \
--cluster-description cluster-test \
--cache-subnet-group-name test \
--cache-node-type cache.t2.micro \
--engine redis \
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

- 创建具有标签的无服务器快照。

```
aws elasticache create-serverless-cache-snapshot \
--serverless-cache-name testing-tags \
--serverless-cache-snapshot-name bkp-testing-tags-scs \
--tags Key="work",Value="foo"
```

- 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，即使复制组包含标签，快照也将仅接收请求标签。

```
aws elasticache create-snapshot \
--replication-group-id testing-tags \
--snapshot-name bkp-testing-tags-rg \
--tags Key="work",Value="foo"
```

基于标签的访问控制策略示例

1. 允许仅当集群具有 Project=XYZ 标签时才对该集群应用 AddTagsToResource 操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticache:AddTagsToResource",
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Project": "XYZ"
                }
            }
        }
    ]
}
```

2. 当复制组包含 Project 和 Service 标签且密钥与 Project 和 Service 不同时，允许该复制组执行 RemoveTagsFromResource 操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticache:RemoveTagsFromResource",
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Service": "Elasticache",
                    "aws:ResourceTag/Project": "XYZ"
                },
                "ForAnyValue:StringNotEqualsIgnoreCase": {
                    "aws:TagKeys": [
                        "Project",
                        "Service"
                    ]
                }
            }
```

```
        }
    ]
}
```

3. 允许仅当标签与 Project 和 Service 不同时才能对任何资源应用 AddTagsToResource。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticache:AddTagsToResource",
            "Resource": [
                "arn:aws:elasticache:*:*:*:*"
            ],
            "Condition": {
                "ForAnyValue:StringNotEqualsIgnoreCase": {
                    "aws:TagKeys": [
                        "Service",
                        "Project"
                    ]
                }
            }
        }
    ]
}
```

4. 在请求具有 Tag Project=Foo 时拒绝 CreateReplicationGroup 操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "elasticache:CreateReplicationGroup",
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/Project": "Foo"
                }
            }
```

```
        }
    ]
}
```

5. 在源快照具有 Project=XYZ 标签且请求标签为 Service=Elasticache 时拒绝 CopySnapshot 操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": "elasticache:CopySnapshot",
            "Resource": [
                "arn:aws:elasticache:*:*:snapshot:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Project": "XYZ",
                    "aws:RequestTag/Service": "Elasticache"
                }
            }
        }
    ]
}
```

6. 如果请求标签 CreateCacheCluster 丢失或不等于 Project、Dev 或 QA，则拒绝 Prod 操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
          {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*",
                "arn:aws:elasticache:*:*:securitygroup:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ]
        },
        {
```

```
            "Effect": "Deny",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "Null": {
                    "aws:RequestTag/Project": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:AddTagsToResource"
            ],
            "Resource": "arn:aws:elasticache:*:*:cluster:*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/Project": [
                        "Dev",
                        "Prod",
                        "QA"
                    ]
                }
            }
        }
    ]
}
```

有关条件键的相关信息，请参阅使用条件键。

使用成本分配标签监控成本

在 Amazon ElastiCache 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。

ElastiCache 成本分配标签是您定义的一个键值对，此标签与 ElastiCache 资源关联。键和值区分大小写。您可以使用标签键定义类别，而标签值作为该类别中的项目。例如，通过定义标签键

CostCenter 和标签值 10010，可以表示将资源分配给 10010 成本中心。再如，通过为标签使用 Environment 键和 test 或 production 值，可以将资源指定为测试或生产用途。我们建议您使用一组一致的标签键，从而方便跟踪与资源相关联的成本。

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

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

您可以向 Redis 节点添加 ElastiCache 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节点。

ElastiCache 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 ElastiCache 资源。资源类型将是 "cluster" 。

  示例 ARN：arn:aws:elasticache:*<region>*:*<customer-id>*:*<resource-type>*:*<resource-name>*

  示例 arn：arn:aws:elasticache:us-west-2:1234567890: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 ElastiCache 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 `ElastiCache`。

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

使用 AWS CLI 管理成本分配标签

您可以使用 AWS CLI 添加、修改或删除成本分配标签。

示例 arn：`arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

成本分配标签应用到 ElastiCache for Redis 节点。要添加标签的节点是使用 ARN (Amazon 资源名称) 指定的。

示例 arn：`arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

主题

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

使用 AWS CLI 列出标签

您可以使用 AWS CLI 通过 list-tags-for-resource 操作列出现有 ElastiCache 资源上的标签。

以下代码使用 AWS CLI 列出 us-west-2 区域中的 `my-cluster` 集群中的 Redis 节点 `my-cluster-001` 上的标签。

对于 Linux、macOS 或 Unix：

```
aws elasticache list-tags-for-resource \
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
```

对于 Windows：

```
aws elasticache list-tags-for-resource ^
```

```
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
```

此操作的输出类似于下文，即列出资源上的所有标签。

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

如果资源上没有任何标签，则输出空标签列表。

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

有关更多信息，请参阅适用于 ElastiCache 的 AWS CLI (list-tags-for-resource)。

使用 AWS CLI 添加标签

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

下面的代码使用 AWS CLI 向 us-west-2 区域中集群 `my-cluster` 的节点 `my-cluster-001` 添加键 `Service` 和 `Region`，这两个键的值分别为 `elasticache` 和 `us-west-2`。

对于 Linux、macOS 或 Unix：

```
aws elasticache add-tags-to-resource \
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 \
 --tags Key=Service,Value=elasticache \
        Key=Region,Value=us-west-2
```

对于 Windows：

```
aws elasticache add-tags-to-resource ^
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 ^
 --tags Key=Service,Value=elasticache ^
        Key=Region,Value=us-west-2
```

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": "elasticache",
         "Key": "Service"
      },
      {
         "Value": "us-west-2",
         "Key": "Region"
      }
   ]
}
```

有关更多信息，请参阅适用于 ElastiCache 的 AWS CLI (add-tags-to-resource)。

还可以在创建新集群时使用 AWS CLI 向集群添加标签，方法是使用操作 create-cache-cluster。使用 ElastiCache 管理控制台创建集群时，您不能添加标签。创建集群之后，随后可以使用控制台向集群添加标签。

使用 AWS CLI 修改标签

您可以使用 AWS CLI 修改 ElastiCache for Redis 集群中节点上的标签。

修改标签：

- 使用 add-tags-to-resource 可添加新标签和值，或更改与现有标签关联的值。
- 使用 remove-tags-from-resource 删除资源的指定标签。

以上任意操作的输出将是指定集群上标签及其值的列表。

使用 AWS CLI 删除标签

您可以使用 AWS CLI 通过 remove-tags-from-resource 操作删除 ElastiCache for Redis 集群中现有节点的标签。

下面的代码使用 AWS CLI 移除了 us-west-2 区域中集群 `my-cluster` 的节点 `my-cluster-001` 包含键 `Service` 和 `Region` 的标签。

对于 Linux、macOS 或 Unix：

```
aws elasticache remove-tags-from-resource \
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 \
 --tag-keys PM Service
```

对于 Windows：

```
aws elasticache remove-tags-from-resource ^
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001 ^
 --tag-keys PM Service
```

此操作的输出将类似于下文，先列出资源上的所有标签，后面跟随操作。

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

有关更多信息，请参阅适用于 ElastiCache 的 AWS CLI (remove-tags-from-resource)。

使用 ElastiCache API 管理成本分配标签

您可以使用 ElastiCache API 添加、修改或删除成本分配标签。

成本分配标签应用到 ElastiCache for Memcached 集群。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:elasticache:us-west-2:1234567890:cluster:my-cluster`

主题

- 使用 ElastiCache API 列出标签
- 使用 ElastiCache API 添加标签
- 使用 ElastiCache API 修改标签
- 使用 ElastiCache API 删除标签

使用 ElastiCache API 列出标签

您可以使用 ElastiCache API 通过 ListTagsForResource 操作列出现有资源上的标签。

以下代码使用 ElastiCache API 列出 us-west-2 区域中 my-cluster-001 资源上的标签。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ListTagsForResource
   &ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Version=2015-02-02
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

使用 ElastiCache API 添加标签

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

以下代码使用 ElastiCache API 向 us-west-2 区域中的 my-cluster-001 资源添加键 Region 和 elasticache，两个键的值分别为 Service和 us-west-2。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=AddTagsToResource
   &ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Tags.member.1.Key=Service
   &Tags.member.1.Value=elasticache
   &Tags.member.2.Key=Region
   &Tags.member.2.Value=us-west-2
   &Version=2015-02-02
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

有关更多信息，请参阅 Amazon ElastiCache API 参考中的 AddTagsToResource。

使用 ElastiCache API 修改标签

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

修改标签的值：

- 使用 AddTagsToResource 操作可添加新标签和值，或更改现有标签的值。
- 使用 RemoveTagsFromResource 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 RemoveTagsFromResource 可删除资源的标签。

使用 ElastiCache API 删除标签

您可以使用 ElastiCache API 通过 RemoveTagsFromResource 操作删除现有 ElastiCache for Redis 节点的标签。

以下代码使用 ElastiCache API 从 us-west-2 区域中集群 `my-cluster` 中的节点 `my-cluster-001` 上删除具有键 `Service` 和 `Region` 的标签。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=RemoveTagsFromResource
   &ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster-001
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &TagKeys.member.1=Service
   &TagKeys.member.2=Region
   &Version=2015-02-02
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

使用 Amazon ElastiCache Well-Architected Lens

本节介绍了 Amazon ElastiCache Well-Architected Lens，这是一组用于设计架构完善的 ElastiCache 工作负载的设计原则和指南。

- ElastiCache Lens 是对 AWS Well-Architected Framework 的补充。
- 每个支柱都有一组问题，有助于围绕 ElastiCache 架构展开讨论。
  - 每个问题都有一些领先的做法及其报告分数。
    - 必需 - 在进入生产环境之前是必需的（如果没有，会导致高风险）
    - 最佳 - 客户可能所处的最佳状态

    - 良好 - 我们建议客户具备的条件（如果没有，会导致中度风险）
- Well-Architected 术语
    - 组件 – 共同满足某项要求的代码、配置和 AWS 资源。组件与其他组件交互，通常等同于微服务架构中的服务。
    - 工作负载 – 共同提供业务价值的一组组件。这样的工作负载有营销网站、电子商务网站、移动应用程序的后端、分析平台等。

主题

- Amazon ElastiCache Well-Architected Lens 卓越运营支柱
- Amazon ElastiCache Well-Architected Lens 安全支柱
- Amazon ElastiCache Well-Architected Lens 可靠性支柱
- Amazon ElastiCache Well-Architected Lens 性能效率支柱
- Amazon ElastiCache Well-Architected Lens 成本优化支柱

Amazon ElastiCache Well-Architected Lens 卓越运营支柱

卓越运营支柱侧重于运行和监控系统以提供业务价值，并不断改进流程和程序。关键主题包括自动变更、响应事件和定义管理日常运营的标准。

主题

- OE 1：您如何理解和响应 ElastiCache 集群触发的提示和事件？
- OE 2：您何时以及如何扩展现有的 ElastiCache 集群？
- OE 3：如何管理您的 ElastiCache 集群资源并使集群保持最新状态？
- OE 4：如何管理客户端与 ElastiCache 集群的连接？
- OE 5：如何为工作负载部署 ElastiCache 组件？
- OE 6：如何针对故障进行规划和缓解故障？
- OE 7：如何排查 Redis 引擎事件的问题？

OE 1：您如何理解和响应 ElastiCache 集群触发的提示和事件？

问题级简介：当您运营 ElastiCache 集群时，您可以选择在发生特定事件时接收通知和提示。原定设置情况下，ElastiCache 会记录与您的资源相关的事件，例如失效转移、节点更换、扩展操作、定期维护等。每个事件都包括日期和时间、来源名称和来源类型以及描述。

问题级优势：能够理解和管理事件（即触发由集群生成的提示）背后的根本原因，将使您能够更有效地运营并对事件做出适当的响应。

- [必需] 在 ElastiCache 控制台（选择您的区域后）上或使用 Amazon 命令行界面（AWS CLI）describe-events 命令和 ElastiCache API 查看由 ElastiCache 生成的事件。配置 ElastiCache 以使用 Amazon Simple Notification Service（Amazon SNS）发送重要集群事件的通知。将 Amazon SNS 与集群结合使用允许您以编程方式对 ElastiCache 事件采取措施。
  - 事件分为两大类：当前事件和计划的事件。当前事件列表包括：资源创建和删除、扩展操作、失效转移、节点重启、创建的快照、集群的参数修改、CA 证书续订、故障事件（集群预调配失败 -VPC 或 ENI-、扩展失败 -ENI- 和快照故障）。计划的事件列表包括：计划在维护时段更换的节点和重新安排的节点更换。
  - 尽管您可能不需要立即对其中一些事件做出反应，但首先查看所有故障事件至关重要：
    - ElastiCache:AddCacheNodeFailed
    - ElastiCache:CacheClusterProvisioningFailed
    - ElastiCache:CacheClusterScalingFailed
    - ElastiCache:CacheNodesRebooted
    - ElastiCache:SnapshotFailed（仅限 Redis）
  - [资源]：
    - 管理 ElastiCache Amazon SNS 通知
    - 事件通知和 Amazon SNS
- [最佳] 要自动响应事件，请利用 SNS 和 Lambda 函数等 AWS 产品和服务功能。遵循最佳实践，进行小的、频繁的、可逆的更改，作为代码来随着时间推移发展您的运营。您应该使用 Amazon CloudWatch 指标来监控您的集群。

  [资源]：使用 AWS Lambda、Amazon Route 53 和 Amazon SNS 监控 Amazon ElastiCache for Redis（已禁用集群模式）只读副本端点，了解使用 Lambda 和 SNS 的使用案例。

OE 2：您何时以及如何扩展现有的 ElastiCache 集群？

问题级简介：合理调整您的 ElastiCache 集群规模是一种平衡行为，每次底层工作负载类型发生变化时都需要进行评估。您的目标是在适合您的工作负载的规模合适的环境中运行。

问题级优势：资源过度利用可能会导致延迟时间增加和整体性能下降。另一方面，利用率不足可能导致资源预调配过多，成本优化不够理想。通过适当地调整环境规模，您可以在性能效率与成本优化之间取

得平衡。为了修正资源利用率过高或不足的情况，ElastiCache 可以在两个维度上进行扩展。您可以通过增加或减少节点容量来纵向扩展。您也可以通过添加和移除节点来横向扩展。

- [必需] 应通过分流读取操作并将读取操作重定向到副本节点，来解决主节点上的 CPU 和网络过度利用问题。使用副本节点进行读取操作以降低主节点利用率。这可以在您的 Redis 客户端库中进行配置，方法是在禁用集群模式时连接到 ElastiCache 读取器端点，或者在启用集群模式时使用 Redis READONLY 命令。

  [资源]：

  - 查找连接端点
  - 合理调整集群规模
  - Redis READONLY 命令

- [必需] 监控 CPU、内存和网络等关键集群资源的利用率。需要跟踪这些特定集群资源的利用率，以便为您的扩展决定和扩展操作的类型提供信息。如果禁用了 ElastiCache for Redis 集群模式，则主节点和副本节点可以纵向扩展。副本节点也可以从 0 个节点横向扩展到 5 个节点。如果启用了集群模式，则这同样适用于集群的每个分片。此外，您可以增加或减少分片的数量。

  [资源]：

  - 使用 Amazon CloudWatch 监控 Amazon ElastiCache for Redis 的最佳实践
  - 扩展 ElastiCache for Redis 集群
  - 扩展 ElastiCache for Memcached 集群

- [最佳] 监控随时间推移的趋势可以帮助您检测工作负载变化，如果在特定时间点进行监控，这些变化将不会被注意到。要检测长期趋势，请使用 CloudWatch 指标扫描更长的时间范围。从长时间观察 CloudWatch 指标中获得的经验应为您预测集群资源利用率提供信息。CloudWatch 数据点和指标的可用时间长达 455 天。

  [资源]：

  - 使用 CloudWatch 指标监控 ElastiCache for Redis
  - 使用 CloudWatch 指标监控 Memcached
  - 使用 Amazon CloudWatch 监控 Amazon ElastiCache for Redis 的最佳实践

- [最佳] 如果您的 ElastiCache 资源是使用 CloudFormation 创建的，则最佳实践是使用 CloudFormation 模板执行更改，以保持操作一致性并避免非托管式配置更改和堆栈偏移。

  [资源]：

  - CloudFormation 的 ElastiCache 资源类型参考

- [最佳] 使用集群运营数据自动执行扩展操作，并在 CloudWatch 中定义阈值以设置警报。使用 CloudWatch Events 和 Simple Notification Service（SNS）触发 Lambda 函数并执行 ElastiCache API 以自动扩展您的集群。例如，当 `EngineCPUUtilization` 指标在很长一段时间内达到 80% 时，向您的集群添加分片。另一种选择是使用 `DatabaseMemoryUsedPercentages` 来设置基于内存的阈值。

  [资源]：

  - 使用 Amazon CloudWatch 警报
  - 什么是 Amazon CloudWatch Events？
  - 将 AWS Lambda 与 Amazon Simple Notification Service 结合使用
  - ElastiCache API 参考

OE 3：如何管理您的 ElastiCache 集群资源并使集群保持最新状态？

问题级简介：大规模运营时，必须能够查明和识别所有 ElastiCache 资源。在推出新的应用程序功能时，您需要跨所有 ElastiCache 环境类型（开发、测试和生产）创建集群版本对称性。资源属性允许您针对不同的运营目标（例如，在推出新功能和启用新的安全机制时）将环境分开。

问题级优势：将开发、测试和生产环境分开是最佳运营实践。以下方法也是最佳实践：跨环境的集群和节点使用众所周知和有据可查的流程应用最新的软件补丁。利用原生 ElastiCache 功能，可以让您的工程团队专注于实现业务目标，而不是 ElastiCache 的维护。

- [最佳] 在可用的最新引擎版本上运行，并在自助服务更新可用时尽快应用这些更新。ElastiCache 会在您指定的集群维护时段内自动更新其底层基础设施。但是，集群中运行的节点会通过自助更新进行更新。这些更新可以分为两种类型：安全补丁或次要软件更新。确保您了解补丁类型之间的区别及其应用时间。

  [资源]：

  - Amazon ElastiCache 中的自助更新
  - Amazon ElastiCache 托管式维护和服务更新帮助页面
- [最佳] 使用标签整理 ElastiCache 资源。在复制组上使用标签，而不是在单个节点上使用标签。您可以配置要在查询资源时显示的标签，也可以使用标签来执行搜索和应用筛选条件。您应该使用资源组来轻松创建和维护共享通用标签集的资源集合。

  [资源]：

  - 标记最佳实践

- CloudFormation 的 ElastiCache 资源类型参考
- 参数组

OE 4：如何管理客户端与 ElastiCache 集群的连接？

问题级简介：大规模运营时，您需要了解您的客户端如何与 ElastiCache 集群连接，以管理应用程序的运营环节（如响应时间）。

问题级优势：选择最合适的连接机制，可确保您的应用程序不会因连接错误（如超时）而断开连接。

- [必需] 将读取操作与写入操作分开，并连接到副本节点以执行读取操作。但请注意，当您将写入与读取分开时，由于 Redis 复制的异步性质，您将失去在写入密钥后立即读取密钥的能力。可以利用 WAIT 命令来提高现实世界的数据安全性，并强制副本在响应客户端之前确认写入，但代价是总体性能降低。使用禁用集群模式的 ElastiCache 读取器端点，可以在 ElastiCache for Redis 客户端库中配置使用副本节点进行读取操作。如果启用了集群模式，请使用 ElastiCache For Redis READONLY 命令。对于许多 ElastiCache for Redis 客户端库，ElastiCache for Redis READONLY 是原定设置情况下或通过配置设置实现的。

  [资源]：

  - 查找连接端点
  - READONLY
- [必需] 使用连接池。建立 TCP 连接在客户端和服务器端都会消耗 CPU 时间，而池化允许您重用 TCP 连接。

  为了减少连接开销，您应该使用连接池。有了连接池，您的应用程序可以“随意”重用和释放连接，而无需支付建立连接的成本。您可以通过 ElastiCache for Redis 客户端库（如果支持），使用适用于您的应用程序环境的框架实现连接池，也可以从头开始构建连接池。
- [最佳] 确保将客户端的套接字超时设置为至少一秒（相比之下，多个客户端的典型原定设置值为“无”）。
  - 当服务器负载较高时，将超时值设置得过低可能会导致超时。如果将其设置得过高，则可能会导致您的应用程序花费很长时间才能检测到连接问题。
  - 通过在客户端应用程序中实现连接池来控制新连接的量。这样可以减少打开和关闭连接所需的延迟和 CPU 利用率，并且如果在集群上启用了 TLS，则执行 TLS 握手。

  [资源]：配置 Amazon ElastiCache for Redis 以提高可用性
- [良好] 使用管道传输（在您的使用案例允许的情况下）可以显著提高性能。

- 通过管道传输，可以减少应用程序客户端和集群之间的往返时间（RTT），即使客户端尚未读取之前的响应，也可以处理新的请求。
- 使用管道传输，您可以向服务器发送多个命令，而无需等待回复/确认。管道传输的缺点是，当您最终批量获取所有响应时，可能出现了一个直到最后您才会发现的错误。
- 实现在返回遗漏错误请求的错误时重试请求的方法。

[资源]：管道传输

OE 5：如何为工作负载部署 ElastiCache 组件？

问题级简介：ElastiCache 环境可以通过 AWS 控制台手动部署，也可以通过 API、CLI、工具包等以编程方式部署。卓越运营最佳实践建议尽可能通过代码自动部署。此外，ElastiCache 集群可以按工作负载进行隔离，也可以组合起来进行成本优化。

问题级优势：为您的 ElastiCache 环境选择最合适的部署机制，可以随着时间的推移改善卓越运营。建议尽可能以代码形式执行操作，以最大限度地减少人为错误并改善可重复性、灵活性以及对事件的响应时间。

通过了解工作负载隔离要求，您可以选择为每个工作负载提供专用 ElastiCache 环境，或者将多个工作负载合并为单个集群，或者将它们组合在一起。了解权衡利弊有助于在卓越运营和成本优化之间取得平衡

- [必需] 了解 ElastiCache 可用的部署选项，并尽可能自动执行这些过程。可能的自动化途径包括 CloudFormation、AWS CLI/SDK 和 API。

  [资源]：

  - Amazon ElastiCache 资源类型参考
  - elasticache
  - Amazon ElastiCache API 参考
- [必需] 对于所有工作负载，确定所需的集群隔离级别。
  - [最佳]：高度隔离 – 工作负载与集群的映射为 1:1。允许在每个工作负载的基础上对 ElastiCache 资源的访问、大小、扩展和管理进行最精细的控制。
  - [更佳]：中等隔离 – M:1 按目的隔离，但可能在多个工作负载之间共享（例如，一个集群专门用于缓存工作负载，另一个集群专门用于消息传递）。
  - [良好]：低度隔离 – M:1 全用途，完全共享。建议用于可接受共享访问的工作负载。

OE 6：如何针对故障进行规划和缓解故障？

问题级简介：卓越运营包括通过定期进行“崩溃前”练习来预测故障，以确定潜在的故障来源，从而消除或缓解故障。ElastiCache 提供失效转移 API，允许模拟节点故障事件，用于测试目的。

问题级优势：通过提前测试故障情景，您可以了解它们如何影响您的工作负载。这样可以安全地测试响应过程及其有效性，并让您的团队熟悉其执行情况。

[必需] 定期在开发/测试账户中执行失效转移测试。TestFailover

OE 7：如何排查 Redis 引擎事件的问题？

问题级简介：卓越运营要求能够调查服务级和引擎级信息，以分析集群的运行状况和状态。Amazon ElastiCache for Redis 可以向 Amazon CloudWatch 和 Amazon Kinesis Data Firehose 发送 Redis 引擎日志。

问题级优势：在 Amazon ElastiCache for Redis 集群上启用 Redis 引擎日志后，可以深入了解影响集群运行状况和性能的事件。Redis 引擎日志直接提供来自 Redis 引擎的数据，而这些数据无法通过 ElastiCache 事件机制获得。通过仔细观察 ElastiCache 事件（请参阅前面的 OE-1）和 Redis 引擎日志，可以从 ElastiCache 服务的角度和 Redis 引擎的角度确定排查问题时的事件顺序。

- [必需] 确保已启用 Redis 引擎日志记录功能，该功能在 ElastiCache for Redis 6.2 及更高版本中可用。这可以在集群创建期间执行，也可以在创建后通过修改集群来执行。
  - 确定是 Amazon CloudWatch Logs 还是 Amazon Kinesis Data Firehose 是 Redis 引擎日志的合适目标。
  - 在 CloudWatch 或 Kinesis Data Firehose 中选择相应的目标日志来保存日志。如果您有多个集群，请考虑为每个集群使用不同的目标日志，因为这将有助于在进行故障排除时隔离数据。

  [资源]：
  - 日志传输：日志传输
  - 日志记录目标：Amazon CloudWatch Logs
  - Amazon CloudWatch Logs 简介：什么是 Amazon CloudWatch Logs？
  - Amazon Kinesis Data Firehose 简介：什么是 Amazon Kinesis Data Firehose？
- [最佳] 如果使用 Amazon CloudWatch Logs，可以考虑利用 Amazon CloudWatch Logs Insights 查询 Redis 引擎日志以获取重要信息。

  例如，针对包含 Redis 引擎日志的 CloudWatch 日志组创建查询，这将返回日志级别为“警告”的事件，例如：

```
fields @timestamp, LogLevel, Message
| sort @timestamp desc
| filter LogLevel = "WARNING"
```

[资源]：使用 CloudWatch Logs Insights 分析日志数据

Amazon ElastiCache Well-Architected Lens 安全支柱

安全支柱侧重于保护信息和系统。关键主题包括数据的机密性和完整性、识别和管理谁能通过基于权限的管理做什么、保护系统以及建立用以检测安全事件的控制措施。

主题

- SEC 1：您在控制对 ElastiCache 数据的授权访问方面采取了哪些举措？
- SEC 2：除了基于网络的控制之外，您的应用程序是否需要对于 ElastiCache 的额外授权？
- SEC 3：是否存在无意中执行命令从而导致数据丢失或故障的风险？
- SEC 4：如何使用 ElastiCache 确保静态数据加密
- SEC 5：如何使用 ElastiCache 加密传输中的数据？
- SEC 6：如何限制对控制面板资源的访问？
- SEC 7：如何检测和响应安全事件？

SEC 1：您在控制对 ElastiCache 数据的授权访问方面采取了哪些举措？

问题级简介：所有 ElastiCache 集群均设计为从 VPC 中的 Amazon Elastic Compute Cloud 实例、无服务器函数（AWS Lambda）或容器（Amazon Elastic Container Service）进行访问。最常遇到的情况是从同一个 Amazon Virtual Private Cloud（Amazon VPC）内的 Amazon Elastic Compute Cloud 实例访问 ElastiCache 集群。必须先授权 Amazon EC2 实例对集群的访问权限，然后您才能从 Amazon EC2 实例连接到集群。要访问在 VPC 中运行的 ElastiCache 集群，需要授予进入该集群的网络入口。

问题级优势：通过 VPC 安全组控制进入集群的网络入口。安全组充当 Amazon EC2 实例的虚拟防火墙，用于控制传入和传出流量。入站规则控制传入到实例的流量，出站规则控制从实例传出的流量。就 ElastiCache 而言，启动集群时，需要关联安全组。这样可以确保组成集群的所有节点都有入站和出站流量规则。此外，ElastiCache 配置为仅在私有子网上部署，因此只能通过 VPC 的私有网络进行访问。

- [必需] 与您的集群关联的安全组控制进入集群的网络入口以及对集群的访问权限。原定设置情况下，安全组将不会定义任何入站规则，因此不会有指向 ElastiCache 的入口路径。要启用此功能，请在安全组上配置入站规则，指定源 IP 地址/范围、TCP 类型流量和 ElastiCache 集群的端口（例如 ElastiCache for Redis 的原定设置端口 6379）。尽管允许非常广泛的入口来源，例如 VPC 中的所有资源（0.0.0.0/0），但建议尽可能详细地定义入站规则，例如，仅授权入站访问在与特定安全组关联的 Amazon EC2 实例上运行的 Redis 客户端。

  [资源]：

  - 子网和子网组
  - 访问您的集群或复制组
  - 使用安全组控制到资源的流量
  - 适用于 Linux 实例的 Amazon Elastic Compute Cloud 安全组
- [必需] 可以为 AWS Lambda 函数分配 AWS Identity and Access Management 策略，允许其访问 ElastiCache 数据。要启用此功能，请创建具有 `AWSLambdaVPCAccessExecutionRole` 权限的 IAM 执行角色，然后将该角色分配给 AWS Lambda 函数。

  [资源]：配置 Lambda 函数以访问 Amazon VPC 中的 Amazon ElastiCache：教程：配置 Lambda 函数以访问 Amazon VPC 中的 Amazon ElastiCache

SEC 2：除了基于网络的控制之外，您的应用程序是否需要对于 ElastiCache 的额外授权？

问题级简介：对于需要在单个客户端级别限制或控制对 ElastiCache for Redis 集群的访问权限的场景中，建议通过 ElastiCache for Redis AUTH 命令进行身份验证。ElastiCache for Redis 身份验证令牌以及可选的用户和用户组管理，让 ElastiCache for Redis 在允许客户端运行命令和访问密钥之前需要密码，进而提高数据面板的安全性。

问题级优势：为了保护您的数据安全，ElastiCache for Redis 提供了旨在防止未经授权访问您数据的机制。这些机制包括强制实施基于角色的访问控制（RBAC）AUTH 或 AUTH 令牌（密码），供客户端在执行授权命令之前连接到 ElastiCache。

- [最佳] 对于 ElastiCache for Redis 6.x 及更高版本，通过定义用户组、用户和访问字符串来定义身份验证和授权控制。将用户分配给用户组，然后将用户组分配给集群。要使用 RBAC，必须在创建集群时将其选中，并且必须启用传输中加密。确保您使用的是支持 TLS 的 Redis 客户端，以便能够利用 RBAC。

  [资源]：

  - 将 RBAC 应用于 ElastiCache for Redis 的复制组
  - 使用访问字符串指定权限
  - ACL
  - 支持的 ElastiCache for Redis 版本
- [最佳] 对于 6.x 之前的 ElastiCache for Redis 版本，除了为 ElastiCache for Redis AUTH 设置强令牌/密码和维持严格的密码策略外，最佳实践是轮换密码/令牌。在任何给定时间，ElastiCache 最多可管理两（2）个身份验证令牌。您也可以修改集群以明确要求使用身份验证令牌。

  [资源]：修改现有 ElastiCache for Redis 集群上的 AUTH 令牌

SEC 3：是否存在无意中执行命令从而导致数据丢失或故障的风险？

问题级简介：许多 Redis 命令一旦错误执行或由恶意行为者执行，就可能会对运营产生不利影响。从性能和数据安全的角度来看，这些命令可能会产生意想不到的后果。例如，开发人员可能会在开发环境中定期调用 FLUSHALL 命令，并且由于错误，可能会无意中尝试在生产系统上调用此命令，从而导致数据意外丢失。

问题级优势：从 ElastiCache 上的 ElastiCache for Redis 5.0.3 开始，您可以重命名某些可能会中断工作负载的命令。重命名命令有助于防止无意中在集群上执行这些命令。

- [必需]

  [资源]：
  - ElastiCache for Redis 版本 5.0.3（已弃用，请使用 5.0.6 版本）
  - Redis 5.0.3 参数更改
  - Redis 安全

SEC 4：如何使用 ElastiCache 确保静态数据加密

问题级简介：虽然 ElastiCache for Redis 是一种内存数据存储，但可以加密任何在集群标准操作中保留（在存储上）的数据。这包括写入 Amazon S3 的计划备份和手动备份，以及在执行同步和交换操作后保存到磁盘存储中的数据。m6g 和 R6g 系列中的实例类型还会始终开启内存加密。

问题级优势：ElastiCache for Redis 提供可选的静态加密，以提高数据安全性。

- [必需] 只有在创建 ElastiCache 集群（复制组）时，才能在该集群上启用静态加密。无法修改现有集群以开始加密静态数据。原定设置情况下，ElastiCache 将提供和管理静态加密中使用的密钥。

[资源]：

- 静态加密条件
- 启用静态加密

- [最佳] 利用当数据在内存中时对数据进行加密的 Amazon EC2 实例类型（例如 m6g 或 R6g）。如果可能，请考虑管理自己的静态加密密钥。对于更严格的数据安全环境，可以使用 AWS Key Management Service（KMS）来自行管理客户主密钥（CMK）。通过 ElastiCache 与 AWS Key Management Service 集成，您可以创建、拥有和管理用于为 ElastiCache for Redis 集群加密静态数据的密钥。

  [资源]：

  - 使用 AWS Key Management Service 中的客户托管密钥
  - AWS Key Management Service
  - AWS KMS 概念

SEC 5：如何使用 ElastiCache 加密传输中的数据？

问题级简介：防止数据在传输过程中被泄露是常见的要求。这些数据指分布式系统的组件内以及应用程序客户端和集群节点之间的数据。ElastiCache for Redis 通过允许对客户端和集群之间以及集群节点自身之间的传输中数据进行加密，从而支持这一要求。m6g 和 R6g 系列中的实例类型还会始终开启内存加密。

问题级简介：Amazon ElastiCache 传输中加密是一项可选功能，您可以通过该功能在数据最脆弱的时候（从一个位置传输到另一个位置时）提高数据的安全性。

- [必需] 只有在创建 ElastiCache for Redis 集群（复制组）时才能在该集群上启用传输中加密。请注意，由于加密/解密数据需要额外的处理，因此，实施传输中加密将会对性能有一些影响。要了解具体会有什么影响，建议在启用传输中加密之前和之后分别对您的工作负载进行基准测试。

  [资源]：

  - 传输中加密概览

SEC 6：如何限制对控制面板资源的访问？

问题级简介：IAM policy 和 ARN 为 ElastiCache for Redis 提供了精细的访问控制，允许通过更严格的控制来管理 ElastiCache for Redis 集群的创建、修改和删除。

问题级优势：可以将 Amazon ElastiCache 资源（例如复制组、节点等）的管理限制为根据 IAM policy 拥有特定权限的 AWS 账户，从而提高资源的安全性和可靠性。

- [必需] 通过为 AWS 用户分配特定 AWS Identity and Access Management 策略来管理对 Amazon ElastiCache 资源的访问权限，从而可以更精细地控制哪些账户可以对集群执行哪些操作。

  [资源]：

  - 管理对 ElastiCache 资源的访问权限的概览
  - 将基于身份的策略（IAM policy）用于 Amazon ElastiCache

SEC 7：如何检测和响应安全事件？

问题级简介：ElastiCache 在启用 RBAC 的情况下部署时，会导出 CloudWatch 指标以向用户通知安全事件。这些指标有助于识别连接的 RBAC 用户未获授权进行身份验证、访问密钥或运行命令的失败尝试。

此外，AWS 产品和服务资源通过自动执行部署和记录所有操作及修改以供日后审查/审计，帮助保护您的整体工作负载。

问题级优势：通过监控事件，可以让您的组织能够根据您的要求、策略和过程做出响应。自动监控和响应这些安全事件可增强您的整体安全态势。

- [必需] 自行熟悉已发布的与 RBAC 身份验证和授权失败有关的 CloudWatch 指标。
  - AuthenticationFailures = 尝试向 Redis 进行身份验证失败
  - KeyAuthorizationFailures = 用户未经许可尝试访问密钥失败
  - CommandAuthorizationFailures = 用户未经许可尝试运行命令失败

  [资源]：

  - Redis 的指标
- [最佳] 建议针对这些指标设置提示和通知，并在必要时做出响应。

  [资源]：

  - 使用 Amazon CloudWatch 告警
- [最佳] 使用 Redis ACL LOG 命令收集进一步的详细信息

  [资源]：

  - ACL LOG

- [最佳] 自行熟悉与监控、记录和分析 ElastiCache 部署和事件相关的 AWS 产品和服务功能

  [资源]：

  - 使用 AWS CloudTrail 记录 Amazon ElastiCache API 调用
  - elasticache-redis-cluster-automatic-backup-check
  - 使用 CloudWatch 指标监控使用情况

Amazon ElastiCache Well-Architected Lens 可靠性支柱

主题

REL 1：您如何支持高可用性（HA）架构部署？

问题级简介：了解 Amazon ElastiCache 的高可用性架构，将使您能够在可用性事件期间以弹性状态运行。

问题级优势：设计您的 ElastiCache 集群的架构，使之具有故障恢复能力，可确保您的 ElastiCache 部署具有更高的可用性。

- [必需] 确定您的 ElastiCache 集群所需的可靠性级别。不同的工作负载具有不同的弹性标准，从完全的临时工作负载到任务关键型工作负载。定义您运行的每种环境类型（例如开发、测试和生产）的需求。

  缓存引擎：Memcached 与 ElastiCache for Redis

  1. Memcached 不提供任何复制机制，主要用于临时工作负载。
  2. ElastiCache for Redis 提供了下面所讨论的 HA 功能

- [最佳] 对于需要 HA 的工作负载，请在集群模式下使用 ElastiCache for Redis，每个分片至少有两个副本，即使对于吞吐量要求较小且只需要一个分片的工作负载也是如此。

  1. 如果启用了集群模式，将自动启用多可用区。

在发生任何计划内或计划外维护以及缓解可用区故障时，多可用区通过执行从主节点到副本的自动失效转移来最大限度地减少停机时间。

2. 对于分片工作负载，由于 Redis 集群协议要求大多数主节点可用才能实现仲裁，因此至少有三个分片可以在失效转移事件期间提供更快的恢复。
3. 跨可用性设置两个或更多副本。

   拥有两个副本可以提高读取可扩展性，也可以在一个副本处于维护状态的场景中提供读取可用性。
4. 使用基于 Graviton2 的节点类型（大多数区域中的原定设置节点）。

   Amazon ElastiCache for Redis 在这些节点上添加了优化的性能。因此，您可以获得更佳的复制和同步性能，从而提高整体可用性。
5. 监控并适当调整规模以应对预期的流量高峰：在高负载下，ElastiCache for Redis 引擎可能会变得无响应，从而影响可用性。`BytesUsedForCache` 和 `DatabaseMemoryUsagePercentage` 是衡量内存使用情况的良好指标，而 `ReplicationLag` 是基于写入速率衡量复制运行状况的指标。您可以使用这些指标来触发集群扩展。
6. 通过在生产失效转移事件之前使用失效转移 API 进行测试，确保客户端恢复能力。

[资源]：

- 配置 Amazon ElastiCache for Redis 以提高可用性
- 使用复制组时的高可用性

REL 2：您如何使用 ElastiCache 实现恢复点目标（RPO）？

问题级简介：了解工作负载 RPO，为有关 ElastiCache 备份和恢复策略的决策提供依据。

问题级优势：制定适当的 RPO 策略，可以提高灾难恢复情景下的业务连续性。设计备份和还原策略有助于您实现 ElastiCache 数据的恢复点目标（RPO）。ElastiCache for Redis 提供存储在 Amazon S3 中的快照功能以及可配置的保留策略。这些快照是在定义的备份时段内拍摄的，并由服务自动处理。如果您的工作负载需要额外的备份粒度，则可以选择每天创建多达 20 个手动备份。手动创建的备份没有服务保留策略，可以无限期保留。

- [必需] 了解并记录您的 ElastiCache 部署的 RPO。
  - 请注意，Memcached 不提供任何备份流程。
  - 查看 ElastiCache 备份和还原特性的功能。
- [最佳] 制定一个沟通良好的集群备份流程。

  - 根据需要启动手动备份。
  - 查看自动备份的保留策略。
  - 请注意，手动备份将会无限期保留。
  - 将自动备份安排在使用率比较低的时段内进行。
  - 对只读副本执行备份操作，以确保将对集群性能的影响降至最低。
- [良好] 利用 ElastiCache 的计划备份功能，在规定的时段内定期备份数据。
  - 定期测试从备份中执行的还原。
- [资源]：
  - Redis
  - ElastiCache for Redis 的备份和还原
  - 进行手动备份
  - 计划自动备份
  - 备份和还原 ElastiCache Redis 集群

REL 3：您如何支持灾难恢复（DR）要求？

问题级简介：灾难恢复对于任何工作负载规划都是一个重要的方面。ElastiCache for Redis 提供了多种选择，可根据工作负载弹性要求实施相应的灾难恢复。使用 Amazon ElastiCache for Redis 全局数据存储，您可以在一个区域中写入您的 ElastiCache for Redis 集群，并让数据可供从另外两个跨区域副本集群读取，从而实现跨区域的低延迟读取和灾难恢复。

问题级优势：了解各种灾难情景并相应进行规划可以确保业务连续性。灾难恢复策略必须在成本、性能影响和数据丢失可能性之间达到平衡。

- [必需] 根据工作负载要求，为所有 ElastiCache 组件制定和记录灾难恢复策略。ElastiCache 的独特之处在于，有些使用案例是完全是临时的，不需要任何灾难恢复策略；而另一些使用案例则截然相反，需要极其稳健的灾难恢复策略。所有选项都必须针对成本优化进行权衡 – 恢复能力越高，则需要的基础设施就越多。

  了解区域级别和多区域级别上可用的灾难恢复选项。
  - 建议进行多可用区部署以防出现可用区故障。确保在多可用区架构中启用集群模式进行部署，且至少提供 3 个可用区。
  - 建议使用全局数据存储以防出现区域故障。
- [最佳] 为需要区域级恢复能力的工作负载启用全局数据存储。

  - 制定计划，以便在主区域出现性能下降时失效转移到辅助区域。
  - 在生产环境中进行失效转移之前，测试多区域失效转移过程。
  - 监控 ReplicationLag 指标，以了解失效转移事件期间数据丢失带来的潜在影响。
- [资源]：
  - 缓解故障
  - 使用全局数据存储跨 AWS 区域进行复制
  - 从备份还原（可选择调整集群大小）
  - 利用多可用区最大限度减少 ElastiCache for Redis 中的停机时间

REL 4：如何有效地规划失效转移？

问题级简介：启用具有自动失效转移功能的多可用区是 ElastiCache 最佳实践。某些情况下，在服务操作过程中，ElastiCache for Redis 会取代主节点。这些情况包括计划维护事件，以及节点故障或可用区出现问题等此类不太可能发生的情况。失效转移的成功与否依赖于 ElastiCache 和您的客户端库配置。

问题级优势：将 ElastiCache 失效转移的最佳实践与特定的 ElastiCache for Redis 客户端库相结合，有助于您最大限度地减少失效转移事件期间的潜在停机时间。

- [必需] 如果禁用集群模式，请使用超时，以便客户端检测是否需要断开与旧的主节点的连接，然后使用更新后的主端点 IP 地址重新连接到新的主节点。如果启用了集群模式，则客户端库负责检测底层集群拓扑的变化。此操作通常是通过 ElastiCache for Redis 客户端库中的配置设置来实现的，该操作还允许您配置刷新的频率和方法。每个客户端库都提供自己的设置，更多详细信息可在相应的文档中找到。

  [资源]：
  - 利用多可用区最大限度减少 ElastiCache for Redis 中的停机时间
  - 查看 ElastiCache for Redis 客户端库的最佳实践。
- [必需] 失效转移的成功取决于主节点和副本节点之间运行状况正常的复制环境。查看并了解 Redis 复制的异步性质，以及可用的 CloudWatch 指标，以便报告主节点和副本节点之间的复制延迟。对于需要更高数据安全性的使用案例，请利用 Redis WAIT 命令强制副本在响应连接的客户端之前确认写入。

  [资源]：
  - Redis 的指标
  - 使用 Amazon CloudWatch 监控 Amazon ElastiCache for Redis 的最佳实践

- [最佳] 在失效转移期间，使用 ElastiCache 测试失效转移 API 定期验证应用程序的响应能力。

  [资源]：

  - 在 Amazon ElastiCache for Redis 上测试到只读副本的自动失效转移
  - 测试自动失效转移

REL 5：您的 ElastiCache 组件是否设计为可扩展？

问题级简介：通过了解扩展能力和可用的部署拓扑，您的 ElastiCache 组件可以不断进行调整，以满足不断变化的工作负载要求。ElastiCache 提供 4 种扩展方式：横向缩减/横向扩展（横向）和纵向扩展/缩减（纵向）。

问题级优势：遵循 ElastiCache 部署的最佳实践可提供最大的扩展灵活性，同时还符合 Well Architected 原则，即横向扩展以最大限度地减少故障的影响。

- [必需] 了解启用集群模式的拓扑与禁用集群模式的拓扑之间的区别。在几乎所有情况下，均建议在启用集群模式的情况下进行部署，因为这可以不断提高可扩展性。禁用集群模式的组件通过添加只读副本进行水平扩展的能力受到限制。
- [必需] 了解何时以及如何扩展。
  - 要获得更多 READIOPS：添加副本
  - 要获得更多 WRITEOPS：添加分片（横向扩展）
  - 要获得更多网络 IO：使用网络优化型实例，纵向扩展
- [最佳] 在启用集群模式的情况下部署 ElastiCache 组件，偏向于更多、更小的节点，而不是更少、更大的节点。这可有效地限制节点故障的影响范围。
- [最佳] 在集群中加入副本，以增强扩展事件期间的响应能力
- [良好] 如果禁用了集群模式，请利用只读副本增加总体读取容量。ElastiCache 在禁用集群模式的情况下最多支持 5 个只读副本，还支持纵向扩展。
- [资源]：
  - 扩展 ElastiCache for Redis 集群
  - 在线纵向扩展
  - 扩展 ElastiCache for Memcached 集群

Amazon ElastiCache Well-Architected Lens 性能效率支柱

性能效率支柱侧重于高效率地使用 IT 和计算资源。关键主题包括：根据工作负载要求选择合适的资源类型和大小、监控性能以及做出明智的决策以跟随业务需求的变化保持效率。

主题

PE 1：如何监控 Amazon ElastiCache 集群的性能？

问题级简介：通过了解现有的监控指标，您可以确定当前的利用率。适当的监控有助于识别影响集群性能的潜在瓶颈。

问题级优势：了解与您的集群关联的指标有助于指导优化技术，从而减少延迟和增加吞吐量。

- [必需] 使用一部分工作负载进行基准性能测试。
  - 您应该使用负载测试等机制监控实际工作负载的性能。
  - 在运行这些测试时监控 CloudWatch 指标，以了解可用指标并建立性能基准。
- [最佳] 对于 ElastiCache for Redis 工作负载，重命名计算成本高的命令（例如 `KEYS`），以限制用户在生产集群上运行阻止性命令的能力。
  - 运行引擎 6.x 的 ElastiCache for Redis 工作负载可以利用基于角色的访问控制来限制某些命令。通过使用 AWS 控制台或 CLI 创建用户和用户组，并将用户组与 ElastiCache for Redis 集群关联，可以控制对命令的访问权限。在 Redis 6 中，启用 RBAC 后，我们可以使用“-@dangerous”，它将禁止该用户使用诸如 KEYS、MONITOR、SORT 等昂贵的命令。
  - 对于引擎版本 5.x，使用 Amazon ElastiCache for Redis 集群参数组上的 `rename-commands` 参数重命名命令。
- [更佳] 分析慢速查询并寻找优化技巧。

  - 对于 ElastiCache for Redis 工作负载，请通过分析慢速日志详细了解查询。例如，您可以使用以下命令 `redis-cli slowlog get 10` 来显示最近 10 条超过延迟阈值（原定设置为 10 秒）的命令。
  - 使用复杂的 ElastiCache for Redis 数据结构，可以更高效地执行某些查询。例如，对于数字样式范围查找，应用程序可以使用排序集来实现简单的数字索引。管理这些索引可以减少对数据集执行的扫描，并以更高的性能效率返回数据。
  - 对于 ElastiCache for Redis 工作负载，`redis-benchmark` 提供了一个简单的界面，用于使用用户定义的输入（如客户端数量和数据大小）测试不同命令的性能。
  - 由于 Memcached 仅支持简单的键级命令，因此可以考虑构建其他键作为索引，以避免遍历键空间来服务于客户端查询。
- [资源]：
  - 使用 CloudWatch 指标监控使用情况
  - 使用 CloudWatch 指标监控使用情况
  - 使用 Amazon CloudWatch 告警
  - Redis 特定的参数
  - SLOWLOG
  - Redis 基准测试

PE 2：如何在 ElastiCache 集群节点间分配工作？

问题级简介：您的应用程序连接到 Amazon ElastiCache 节点的方式可能会影响集群的性能和可扩展性。

问题级优势：正确使用集群中的可用节点将确保在可用的资源中分配工作。以下技巧也有助于避免闲置资源。

- [必需] 让客户端连接到正确的 ElastiCache 端点。
  - Amazon ElastiCache for Redis 根据所使用的集群模式实现不同的端点。如果启用了集群模式，ElastiCache 将提供配置端点。如果禁用了集群模式，ElastiCache 将提供主端点（通常用于写入）和用于平衡副本间读取的读取器端点。正确实现这些端点将会提高性能并让扩展操作更轻松。除非有特定要求，否则请避免连接到各个节点端点。
  - 对于多节点 Memcached 集群，ElastiCache 提供了支持自动发现的配置端点。建议使用哈希算法在缓存节点之间均匀分配工作。许多 Memcached 客户端库可实现一致性哈希。请参阅您要使用的库的文档，了解其是否支持一致性哈希以及如何实现一致性哈希。您可以在此处找到有关实现这些功能的更多信息。

- [更佳] 利用启用的 ElastiCache for Redis 集群模式来提高可扩展性。
  - ElastiCache for Redis（已启用集群模式）集群支持在线扩展操作（横向扩展/横向缩减和纵向扩展/缩减），以帮助在分片之间动态分配数据。使用配置端点将确保您的集群感知客户端能够适应集群拓扑的变化。
  - 您也可以通过在 ElastiCache for Redis（已启用集群模式）集群中的可用分片之间移动哈希槽来重新平衡集群。此举有助于在可用分片之间更高效地分配工作。
- [更佳] 实施用于识别和修复工作负载中热键的策略。
  - 考虑多维 Redis 数据结构（例如列表、流、集合等）的影响。这些数据结构存储在单个 Redis 键中，而这些键位于单个节点上。与其他数据类型相比，非常大的多维键有可能占用更多的网络容量和内存，因此可能导致过度使用该节点。如果可能，请将您的工作负载设计为将数据访问分散到许多离散的键上。
  - 工作负载中的热键可能会影响正在使用的节点的性能。对于 ElastiCache for Redis 工作负载，如果存在 LFU 最大内存策略，则可以使用 `redis-cli --hotkeys` 检测热键。
  - 考虑在多个节点上复制热键，以便更均匀地分配对它们的访问。这种方法要求客户端写入多个主节点（Redis 节点本身不提供此功能），除原始键名称外，还需要维护一个可供读取的键名称列表。
  - ElastiCache for Redis 版本 6 支持服务器辅助客户端缓存。这让应用程序能够等待更改键后再向 ElastiCache 进行网络调用。
- [资源]：
  - 配置 Amazon ElastiCache for Redis 以提高可用性
  - 查找连接端点
  - 负载均衡最佳实践
  - Redis（已启用集群模式）的在线重新分片和分片重新平衡
  - Redis 中的客户端缓存

PE 3：对于缓存工作负载，如何跟踪和报告缓存的有效性和性能？

问题级简介：缓存是 ElastiCache 上经常遇到的工作负载，了解如何管理缓存的有效性和性能非常重要。

问题级优势：您的应用程序可能显示出性能不佳的迹象。您能够使用特定于缓存的指标来决定如何提高应用程序性能，这对您的缓存工作负载至关重要。

- [必需] 测量并跟踪一段时间内的缓存命中率。缓存的效率由其“缓存命中率”决定。缓存命中率由键命中总数除以命中和未命中总数来定义。比率越接近 1，您的缓存就越有效。缓存命中率低是由缓存未命中数量造成的。当在缓存中找不到请求的键时，就会出现缓存未命中。键不在缓存中，因为它要么

已被驱逐或删除，要么已过期，要么从未存在。了解为什么键不在缓存中，并制定适当的策略将其放入缓存。

[资源]：

- Redis 的指标

- [必需] 测量和收集应用程序缓存性能以及延迟和 CPU 利用率值，以便了解是否需要调整生存时间或其他应用程序组件。ElastiCache 为每种数据结构的聚合延迟提供了一组 CloudWatch 指标。这些延迟指标是使用 ElastiCache for Redis INFO 命令中的命令统计数据计算得出的，不包括网络和 I/O 时间。这只是 ElastiCache for Redis 处理操作所耗费的时间。

  [资源]：

  - Redis 的指标
  - 使用 Amazon CloudWatch 监控 Amazon ElastiCache for Redis 的最佳实践

- [最佳] 根据您的需求选择合适的缓存策略。缓存命中率低是由缓存未命中数量造成的。如果您的工作负载设计为缓存未命中数量较低（例如实时通信），则最好对缓存策略进行审查，并为您的工作负载应用最合适的解决方案，例如用于测量内存和性能的查询检测。您用于为填充并维护缓存而实施的实际策略取决于客户端需要缓存的数据以及针对这些数据的访问模式。例如，您不太可能对流媒体应用程序的个性化推荐和热门新闻报道使用相同的策略。

  [资源]：

  - 缓存策略
  - 缓存最佳实践
  - 利用 Amazon ElastiCache 实现规模性能白皮书

PE 4：您的工作负载如何优化网络资源和连接的使用？

问题级简介：许多应用程序客户端都支持 ElastiCache for Redis 和 Memcached，实现方式可能会有所不同。您需要了解现有的网络和连接管理，以分析潜在的性能影响。

问题级优势：高效地使用网络资源可以提高集群的性能效率。以下建议可以减少网络需求，并改善集群延迟和吞吐量。

- [必需] 主动管理与您的 ElastiCache 集群的连接。
  - 应用程序中的连接池减少了通过打开和关闭连接在集群上产生的开销量。使用 `CurrConnections` 和 `NewConnections` 监控 Amazon CloudWatch 中的连接行为。

  - 通过在适当的位置正确关闭客户端连接来避免连接泄露。连接管理策略包括正确关闭未使用的连接，以及设置连接超时。
  - 对于 Memcached 工作负载，为处理连接预留了可配置的内存量，称为 `memcached_connections_overhead`。
- [更佳] 压缩大型对象以减少内存并提高网络吞吐量。
  - 数据压缩可以减少所需的网络吞吐量（Gbps），但会增加应用程序压缩和解压缩数据的工作量。
  - 压缩还会减少键所消耗的内存量
  - 根据您的应用程序需求，考虑压缩比与压缩速度之间的权衡。
- [资源]：
  - Amazon ElastiCache for Redis - 全局数据存储
  - Memcached 特定的参数
  - Amazon ElastiCache for Redis 5.0.3 增强了 I/O 处理以提高性能
  - Redis 的指标
  - 配置 Amazon ElastiCache for Redis 以提高可用性

PE 5：如何管理键删除和/或驱逐？

问题级简介：当集群节点接近内存消耗限制时，工作负载具有不同的要求和预期行为。Amazon ElastiCache for Redis 具有不同的策略来处理这些情况。

问题级优势：适当管理可用内存和了解驱逐策略，将有助于确保了解在超过实例内存限制时的集群行为。

- [必需] 检测数据访问权限以评估要应用的策略。确定适当的最大内存策略，以控制是否以及如何对集群执行驱逐。
  - 当集群上的最大内存消耗完毕并且制定了允许驱逐的策略时，就会发生驱逐。在这种情况下，集群的行为取决于指定的驱逐策略。此策略可以使用 ElastiCache for Redis 集群参数组上的 `maxmemory-policy` 进行管理。
  - 原定设置策略 `volatile-lru` 通过驱逐设置了过期时间（TTL 值）的键来释放内存。最少使用（LFU）和最近最少使用（LRU）策略会根据使用情况删除键。
  - 对于 Memcached 工作负载，有一个原定设置 LRU 策略来控制每个节点上的驱逐。您可以使用 Amazon CloudWatch 上的驱逐指标来监控您的 Amazon ElastiCache 集群上的驱逐次数。
- [更佳] 对删除行为进行标准化来控制对集群的性能影响，从而避免意外的性能瓶颈。

  - 对于 ElastiCache for Redis 工作负载，当从集群中明确删除键时，UNLINK 就像 DEL：它会删除指定的键。但是，该命令在不同的线程中执行实际内存回收，因此它不会阻止，而 DEL 会阻止。实际的删除将在稍后异步进行。
  - 对于 ElastiCache for Redis 6.x 工作负载，可以使用 `lazyfree-lazy-user-del` 参数在参数组中修改 DEL 命令的行为。
- [资源]：
  - 使用参数组配置引擎参数
  - UNLINK
  - 使用 AWS 进行云财务管理

PE 6：如何在 ElastiCache 中对数据建模以及与数据交互？

问题级简介：ElastiCache 应用程序在很大程度上依赖于所使用的数据结构和数据模型，但它还需要考虑底层数据存储（如果存在）。了解可用的 ElastiCache for Redis 数据结构，并确保使用最适合您需求的数据结构。

问题级优势：ElastiCache 中的数据建模有多个层，包括应用程序使用案例、数据类型以及数据元素之间的关系。此外，每个 ElastiCache for Redis 数据类型和命令都有自己有据可查的性能签名。

- [最佳] 最佳实践是减少无意中覆盖数据的情况。使用可最大限度地减少重叠键名称的命名约定。数据结构的传统命名使用分层方法，例如：`APPNAME:CONTEXT:ID`（如 `ORDER-APP:CUSTOMER:123`）。

  [资源]：

  - 键命名
- [最佳] ElastiCache for Redis 命令的时间复杂度由 Big O 表示法定义。命令的这种时间复杂度是其影响的算法/数学表示形式。在应用程序中引入新的数据类型时，需要仔细检查相关命令的时间复杂度。时间复杂度为 O(1) 的命令在时间上是恒定的，不依赖于输入的大小，但时间复杂度为 O(N) 的命令在时间上是线性的，受输入大小影响。由于 ElastiCache for Redis 采用单线程设计，因此，大量时间复杂度高的操作将导致性能下降，并可能会引起操作超时。

  [资源]：

  - 命令
- [最佳] 使用 API 获取 GUI 对集群中数据模型的可见性。

  [资源]：

- Redis Commander
- Redis 浏览器
- Redsmin

PE 7：如何在 Amazon ElastiCache 集群中记录运行缓慢的命令？

问题级简介：通过捕获、聚合和通知长时间运行的命令，可以改善性能调优效果。通过了解执行命令所需的时长，您可以确定哪些命令会导致性能不佳，以及哪些命令阻止引擎以最佳方式执行。Amazon ElastiCache for Redis 还可以将这些信息转发到 Amazon ElastiCache 或 Amazon Kinesis Data Firehose。

问题级优势：记录到专用的永久位置并为慢速命令提供通知事件，有助于进行详细的性能分析，并可用于触发自动事件。

- [必需] 运行引擎版本 6.0 或更高版本的 Amazon ElastiCache for Redis，在集群上正确配置参数组并启用了 SLOWLOG 日志记录。
  - 仅当引擎版本兼容性设置为 Redis 版本 6.0 或更高版本时，必需的参数才可用。
  - 当命令的服务器执行时间超过指定的值时，就会发生 SLOWLOG 日志记录。集群的行为取决于关联的参数组参数，即 `slowlog-log-slower-than` 和 `slowlog-max-len`。
  - 更改将立即生效。
- [最佳] 利用 CloudWatch 或 Kinesis Data Firehose 功能。
  - 使用 CloudWatch、CloudWatch Logs Insights 和 Amazon Simple Notification Services 的筛选和警报功能来实现性能监控和事件通知。
  - 使用 Kinesis Data Firehose 的流式传输功能，将 SLOWLOG 日志归档到永久存储空间或触发自动集群参数调优。
  - 确定 JSON 还是纯文本格式最适合您的需求。
  - 提供 IAM 权限以发布到 CloudWatch 或 Kinesis Data Firehose。
- [更佳] 将 `slowlog-log-slower-than` 配置为原定设置值以外的值。
  - 此参数确定命令在 Redis 引擎中执行多长时间后会被记录为慢速运行命令。原定设置值为 10000 微秒（10 毫秒）。对于某些工作负载，原定设置值可能过高。
  - 根据应用程序需求和测试结果确定更适合您工作负载的值；但是，值过低可能会生成过多的数据。
- [更佳] 将 `slowlog-max-len` 保留为原定设置值。

- 此参数决定了任何给定时间在 Redis 内存中捕获的慢速运行命令数上限。值为 0 会有效地禁用捕获。该值越高，存储在内存中的条目就越多，从而减少了重要信息在查看之前被驱逐的可能性。原定设置值为 128。
- 原定设置值适用于大多数工作负载。如果需要通过 SLOWLOG 命令在 redis-cli 的扩展时段内分析数据，请考虑增加此值。这允许在 Redis 内存中保留更多命令。

  如果您将 SLOWLOG 数据发送到 CloudWatch Logs 或 Kinesis Data Firehose，则数据将保留并可以在 ElastiCache 系统之外进行分析，从而减少在 Redis 内存中存储大量慢速运行命令的需求。

- [资源]：
  - 如何在 ElastiCache for Redis 缓存群集中开启 Redis 慢速日志？
  - 日志传输
  - Redis 特定的参数
  - https://aws.amazon.com/cloudwatch/Amazon CloudWatch
  - Amazon Kinesis Data Firehose

PE8：自动扩缩如何帮助提高 ElastiCache 集群的性能？

问题级简介：通过实施 Redis 自动扩缩的功能，您的 ElastiCache 组件可以逐渐进行调整，以自动增加或减少所需的分片或副本。这可以通过实施目标跟踪或计划的扩展策略来实现。

问题级优势：了解和规划工作负载的峰值，可以确保提高缓存性能和业务连续性。ElastiCache for Redis 自动扩缩会持续监控您的 CPU/内存利用率，以确保您的集群以所需的性能水平运行。

- [必需] 启动 ElastiCache for Redis 的集群时：
  1. 确保已启用集群模式
  2. 确保该实例属于支持自动扩缩的特定类型和大小的系列
  3. 确保集群未在全局数据存储、Outposts 或本地区域中运行

  [资源]：
  - 扩展 Redis 中的集群（启用集群模式）
  - 将自动扩缩与分片结合使用
  - 将自动扩缩与副本结合使用
- [最佳] 确定您的工作负载是读取密集型还是写入密集型，以定义扩展策略。要想获得最佳性能，请仅使用一个跟踪指标。建议避免针对每个维度使用多个策略，因为自动扩缩策略会在达到目标时横向扩展，但只有在所有目标跟踪策略都准备好横向缩减时才会进行横向缩减。

[资源]：

- 自动扩缩策略
- 定义扩展策略

- [最佳] 在一段时间内持续监控性能有助于您检测工作负载变化，如果在特定时间点进行监控，将不会注意到这些变化。您可以分析四周内集群利用率的相应 CloudWatch 指标，以确定目标值阈值。如果您仍然不确定要选择哪个值，我们建议您从支持的最小预定义指标值开始。

  [资源]：

  - 使用 CloudWatch 指标监控使用情况
- [更佳] 我们建议使用预期的最小和最大工作负载测试您的应用程序，以确定集群制定扩展策略和减轻可用性问题所需的分片/副本的确切数量。

  [资源]：

  - 注册可扩展目标
  - 注册可扩展目标

Amazon ElastiCache Well-Architected Lens 成本优化支柱

成本优化支柱侧重于避免不必要的成本。关键主题包括了解和控制资金花在哪里、选择最合适的节点类型（使用支持基于工作负载需求进行数据分层的实例）、相应数量的资源类型（有多少只读副本）、分析一段时间内的支出，以及在不超支的情况下进行扩展以满足业务需求。

主题

- 成本 1：如何识别和跟踪与 ElastiCache 资源相关的成本？ 您如何建立让用户能够创建、管理和处置已创建资源的机制？
- 成本 2：如何使用持续监控工具来优化与 ElastiCache 资源关联的成本？
- 成本 3：您是否应该使用支持数据分层的实例类型？ 数据分层有哪些优点？ 何时不使用数据分层实例？

成本 1：如何识别和跟踪与 ElastiCache 资源相关的成本？ 您如何建立让用户能够创建、管理和处置已创建资源的机制？

问题级简介：要了解成本指标，需要多个团队参与和协作：软件工程、数据管理、产品负责人、财务和领导层。要确定关键成本驱动因素，则要求所有相关方了解服务使用控制杠杆和成本管理的利弊，这通

常是成功与不太成功的成本优化工作之间的关键区别。确保您有适当的流程和工具来跟踪从开发到生产和停用期间创建的资源，这有助于您管理与 ElastiCache 关联的成本。

问题级优势：要持续跟踪与您工作负载关联的所有成本，需要深入了解将 ElastiCache 作为其组件之一的架构。此外，您应该制定成本管理计划，以收集使用情况并将其与预算进行比较。

- [必需] 建立一个云卓越中心（CCoE），作为其创始章程之一，负责定义、跟踪组织的 ElastiCache 使用情况，并根据相关指标采取措施。如果 CCoE 存在且正常运行，请确保其知道如何读取和跟踪与 ElastiCache 关联的成本。创建资源时，使用 IAM 角色和 IAM policy 来验证只有特定的团队和组才能实例化资源。这确保了成本与业务成果相关联，并从成本角度建立了明确的问责制。
  1. CCoE 应基于分类数据识别、定义和发布与关键 ElastiCache 使用情况相关的成本指标（每月定期更新这），例如：
     a. 使用的节点类型及其属性：标准与内存优化型、按需实例与预留实例、区域和可用区
     b. 环境类型：免费环境、开发环境、测试环境和生产环境
     c. 备份存储和保留策略
     d. 区域内和跨区域的数据传输
     e. 在 Amazon Outposts 上运行的实例
  2. CCoE 由一个跨职能团队组成，其代表来自组织中软件工程、数据管理、产品团队、财务团队和领导团队的非专属代表。

  [资源]：
  - 创建云卓越中心
  - Amazon ElastiCache 定价
- [必需] 使用成本分配标签以较低的粒度跟踪成本。使用 AWS 成本管理来可视化、了解和管理一段时间内的 AWS 成本和使用情况。
  1. 使用标签来整理资源，并可以使用成本分配标签来细致地跟踪 AWS 成本。在您激活成本分配标签后，AWS 将使用成本分配标签来整理您的资源分配报告中的资源成本，以方便您对 AWS 成本进行分类和跟踪。AWS 提供了两种类型的成本分配标签：AWS 生成的标签和用户定义的标签。AWS 将为您定义、创建和应用 AWS 生成的标签，而您将定义、创建和应用用户定义的标签。您必须先分别激活这两种类型的标签，然后这些标签才能显示在成本管理中或成本分配报告上。
  2. 使用成本分配标签整理 AWS 账单，以反映您自己的成本结构。在 Amazon ElastiCache 中向资源添加成本分配标签时，将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。您应该考虑合并标签，以采用更高详细信息级别跟踪成本。

  [资源]：

- 使用 AWS 成本分配标签
- 使用成本分配标签监控成本
- AWS Cost Explorer

- [最佳] 将 ElastiCache 成本与涵盖整个组织的指标联系起来。

  1. 考虑业务指标以及延迟等运营指标 - 您业务模式中有哪些概念可以被不同角色理解？ 这些指标需要让组织中尽可能多的角色能够理解。
  2. 示例 - 同时服务的用户数、每个操作和用户的最大和平均延迟、用户参与度分数、用户每周返回率、会话长度/用户、放弃率、缓存命中率以及跟踪的键

  [资源]：

  - 使用 CloudWatch 指标监控使用情况

- [良好] 在使用 ElastiCache 的整个工作负载中，保持最新的架构和运营指标和成本可见性。

  1. 了解您的整个解决方案生态系统，ElastiCache 往往是其技术组合中完整 AWS 服务生态系统的一部分，例如，从客户端到 API Gateway、Redshift 和 QuickSight（用于报告工具）。
  2. 在架构图上映射解决方案的各个组成部分，包括客户端、连接、安全性、内存操作、存储、资源自动化、数据访问和管理。每层都连接到整个解决方案且具有其自身的需求和功能，可以助力和/或帮助您管理总体成本。
  3. 您的图表应包括计算、网络、存储、生命周期策略、指标收集的使用情况以及应用程序的操作和功能 ElastiCache 元素
  4. 工作负载的要求可能会不断变化，为了在工作负载成本管理中保持主动性，您必须继续维护和记录您对基本组件以及主要功能目标的理解。
  5. 管理层在可见性、问责制、优先级划分和资源方面的支持对于您为 ElastiCache 制定有效的成本管理策略至关重要。

成本 2：如何使用持续监控工具来优化与 ElastiCache 资源关联的成本？

问题级简介：您需要在您的 ElastiCache 成本和应用程序性能指标之间取得适当的平衡。Amazon CloudWatch 提供关键运营指标的可见性，可以帮助您评测，相对于您的需求，您的 ElastiCache 资源是过度使用还是未得到充分利用。从成本优化的角度来看，您需要了解何时过度预调配，并能够开发适当的机制来调整您的 ElastiCache 资源的规模，同时保持运营、可用性、弹性和性能需求。

问题级优势：在理想状态下，您将预调配足够的资源来满足工作负载的运维需求，并且不会出现未充分利用的资源，从而导致成本状态欠佳。您需要能够识别和避免长时间运行规模过大的 ElastiCache 资源。

- [必需] 使用 CloudWatch 监控您的 ElastiCache 集群，并分析这些指标与您 AWS Cost Explorer 成本管理服务控制面板的相关性。

  1. ElastiCache 提供主机层面级指标（例如 CPU 使用率）和特定于缓存引擎软件的指标（例如缓存获取次数和缓存未命中数）。这些指标每隔 60 秒对每个缓存节点进行测量并发布结果。
  2. ElastiCache 性能指标（CPUUtilization、EngineUtilization、SwapUsage、CurrConnections 和 Evictions）可能表明您需要纵向扩展/缩减（使用更大/更小的缓存节点类型）或横向缩减/横向扩展（添加更多/更少的分片）。通过创建 PlayBook 矩阵来了解扩展决策的成本影响，该矩阵可估算满足应用程序性能阈值所需的额外成本以及最小和最大时间长度。

  [资源]：

  - 使用 CloudWatch 指标监控使用情况
  - 应监控哪些指标？
  - Amazon ElastiCache 定价

- [必需] 了解并记录您的备份策略和成本影响。

  1. 使用 ElastiCache，备份存储在 Amazon S3 中，而 Amazon S3 可提供持久存储。您需要了解与故障恢复能力有关的成本影响。
  2. 启用自动备份，这将删除超过保留期限的备份文件。

  [资源]：

  - 计划自动备份
  - Amazon Simple Storage Service 定价

- [最佳] 作为一种深思熟虑的策略，应对实例使用预留节点，以管理已充分了解和记录的工作负载的成本。预留节点需支付预付费用，此费用取决于节点类型和预留时间长短（一年或三年）。此费用远低于按需节点产生的每小时使用费。

  1. 在收集到足够的数据来评估预留实例需求之前，您可能需要使用按需节点运行您的 ElastiCache 集群。规划和记录满足需求所需的资源，并比较不同实例类型（按需型与预留）的预期成本
  2. 定期评测新的可用缓存节点类型，并从成本和运营指标的角度评测将您的实例集迁移到新的缓存节点类型是否合理

成本 3：您是否应该使用支持数据分层的实例类型？ 数据分层有哪些优点？ 何时不使用数据分层实例？

问题级简介：选择适当的实例类型不仅会对性能和服务级别产生影响，还会对财务状况产生影响。实例类型具有不同的关联成本。选择一种或几种可以满足内存中所有存储需求的大型实例类型可能是一个自

然而然的决定。但是，随着项目日趋成熟，这可能会对成本产生重大影响。要确保选择正确的实例类型，需要定期检查 ElastiCache 对象的空闲时间。

问题级优势：您应该清楚地了解各种实例类型对您当前和未来的成本有何影响。边际或定期的工作负载变化不应导致过多的成本变化。如果工作负载允许，支持数据分层的实例类型提供的每可用存储的价格将更优惠。这是因为每实例可用的 SSD 存储数据分层实例所支持的每实例的总数据容量要高得多。

- [必需] 了解数据分层实例的局限性
  1. 仅适用于 ElastiCache for Redis 集群。
  2. 支持数据分层的实例类型非常有限。
  3. 仅支持 ElastiCache for Redis 版本 6.2 及更高版本
  4. 大型项目不会交换到 SSD。超过 128MiB 的对象保留在内存中。

  [资源]：
  - 数据分层
  - Amazon ElastiCache 定价
- [必需] 了解您的工作负载定期访问数据库的百分比。
  1. 数据分层实例非常适合经常访问整个数据集的一小部分但仍需要快速访问其余数据的工作负载。换句话说，热数据与温数据的比例约为 20:80。
  2. 制定对象空闲时间的集群级跟踪。
  3. 超过 500Gb 数据的大型实现是不错的选择
- [必需] 了解数据分层实例对于某些工作负载不是可选的。
  1. 访问不常用的对象会产生少许性能成本，因为这些对象会被交换到本地 SSD。如果您的应用程序对响应时间敏感，请测试对工作负载的影响。
  2. 不适合主要存储大小超过 128MiB 的大型对象的缓存。

  [资源]：
  - 限制
- [最佳] 预留实例类型支持数据分层。这可确保在每个实例的数据存储量方面实现最低的成本。
  1. 在更好地了解您的需求之前，您可能需要使用非数据分层实例运行 ElastiCache 集群。
  2. 分析您的 ElastiCache 集群的数据使用模式。
  3. 创建定期收集对象空闲时间的自动作业。
  4. 如果您发现很大一部分对象（大约 80%）在认为适合您工作负载的时间段内处于空闲状态，请记录调查发现，并建议将集群迁移到支持数据分层的实例。

5. 定期评测新的可用缓存节点类型，并从成本和运营指标的角度评测将您的实例集迁移到新的缓存节点类型是否合理。

[资源]：

- 对象空闲时间
- Amazon ElastiCache 定价

常见故障排除步骤和最佳实践

主题

- 连接问题
- Redis 客户端错误
- 解决 ElastiCache 无服务器中的高延迟问题
- 对无服务器中的限制问题进行故障排除 ElastiCache
- 相关主题

连接问题

如果您无法连接到 ElastiCache 缓存，请考虑以下方法之一：

1. 使用 TLS：如果您在尝试连接到 ElastiCache 终端节点时遇到连接挂起的情况，则可能没有在客户端中使用 TLS。如果您使用的是 ElastiCache 无服务器，则传输中的加密始终处于启用状态。确保您的客户端使用 TLS 连接到缓存。在此处了解有关连接到启用 TLS 的缓存的更多信息。
2. VP ElastiCache C：只能从 VPC 内部访问缓存。确保您访问缓存的 EC2 实例和缓存是在同一 VPC 中创建的。 ElastiCache 或者，您必须在您的 EC2 实例所在的 VPC 和创建缓存的 VPC 之间启用 VPC 对等关系。
3. 安全组：ElastiCache 使用安全组来控制对缓存的访问权限。请考虑以下事项：
   a. 确保您的 ElastiCache 缓存使用的安全组允许从 EC2 实例对其进行入站访问。请参阅此处，了解如何在安全组中正确设置入站规则。
   b. 确保您的 ElastiCache 缓存使用的安全组允许访问缓存的端口（无服务器端口 6379 和 6380，自行设计的端口默认为 6379）。 ElastiCache 使用这些端口接受 Redis 命令。在此处详细了解如何设置端口访问权限。

Redis 客户端错误

ElastiCache 只有使用支持 Redis 集群模式协议的 Redis 客户端才能访问无服务器。根据集群配置，可以在任一模式下从 Redis 客户端访问自行设计的集群。

如果您在客户端中遇到 Redis 错误，请考虑以下几点：

1. 集群模式：如果您在使用 SE LECT Redis 命令时遇到 CROSSLOT 错误或错误，则可能正在尝试使用不支持 Redis 集群协议的 Redis 客户端访问已启用集群模式的缓存。 ElastiCache 无服务器仅支持支持 Redis 集群协议的 Redis 客户端。如果要在 “禁用集群模式” (CMD) 中使用 Redis，则必须设计自己的集群。
2. CROSSLOT 错误：如果您遇到`ERR CROSSLOT Keys in request don't hash to the same slot`错误，则可能正在尝试访问不属于集群模式缓存中同一插槽的密钥。提醒一下，ElastiCache Serverless 始终在集群模式下运行。仅当涉及的所有密钥都在同一个哈希槽中时，才允许使用涉及多个密钥的多密钥操作、事务或 Lua 脚本。

有关配置 Redis 客户端的其他最佳实践，请查看此博客文章。

解决 ElastiCache 无服务器中的高延迟问题

如果您的工作负载出现高延迟，则可以分析 CloudWatch `SuccessfulReadRequestLatency`和`SuccessfulWriteRequestLatency`指标，以检查延迟是否与 ElastiCache 无服务器有关。这些指标衡量的是 ElastiCache 无服务器内部的延迟，不包括客户端延迟以及您的客户端和 ElastiCache 无服务器端点之间的网络访问时间。

有些可变性和偶尔出现的峰值不应引起担忧。但是，如果`Average`统计数据显示急剧增长并持续存在，则应查看和您的 Personal Health Dashboard 以获取更多信息。 AWS Health Dashboard 如有必要，可以考虑向提出支持案例 AWS Support。

考虑以下减少延迟的最佳实践和策略：

- 启用从副本读取：如果您的应用程序允许，我们建议在您的 Redis 客户端中启用 “从副本读取” 功能，以扩展读取并降低延迟。启用后， ElastiCache Serverless 会尝试将您的读取请求路由到与您的客户端位于同一可用区 (AZ) 的副本缓存节点，从而避免跨可用区网络延迟。请注意，在客户端中启用 “从副本读取” 功能表示您的应用程序接受数据的最终一致性。如果您在写入密钥后尝试读取，您的应用程序可能会在一段时间内收到较旧的数据。
- 确保您的应用程序部署在与缓存相同的可用区中：如果您的应用程序未部署在与缓存相同的可用区中，则可能会出现更高的客户端延迟。创建无服务器缓存时，您可以提供您的应用程序将从中访问缓

存的子网， ElastiCache Serverless 将在这些子网中创建 VPC 终端节点。确保您的应用程序部署在相同的可用区中。否则，您的应用程序在访问缓存时可能会出现跨可用区跳跃，从而导致更高的客户端延迟。

- 重用连接：ElastiCache 无服务器请求是使用 RESP 协议通过启用 TLS 的 TCP 连接发出的。启动连接（包括对连接进行身份验证，如果已配置）需要时间，因此第一个请求的延迟要高于典型延迟。通过已初始化的连接发出的请求可提供始终如一 ElastiCache的低延迟。因此，您应该考虑使用连接池或重复使用现有的 Redis 连接。
- 扩展速度：ElastiCache Serverless 会随着请求速率的增长而自动扩展。请求速率的突然大幅增加（快 ElastiCache 于 Serverless 的扩展速度）可能会在一段时间内导致延迟升高。 ElastiCache Serverless 通常可以快速提高其支持的请求速率，最多需要 10-12 分钟才能将请求速率提高一倍。
- 检查长时间运行的命令：某些 Redis 命令，包括 Lua 脚本或大型数据结构上的命令，可能会运行很长时间。要识别这些命令，请 ElastiCache 发布命令级指标。借助ElastiCache 无服务器，您可以使用这些BasedECPUs指标。
- 受@@ 限制的请求：在 ElastiCache Serverless 中限制请求时，您的应用程序中的客户端延迟可能会增加。当请求在 ElastiCache Serverless 中受到限制时，您应该会看到无服务器指标有所增加。ThrottledRequests ElastiCache 请查看以下部分，了解受限请求的疑难解答。
- 密钥和请求的均匀分布：在 ElastiCache Redis 中，每个插槽的密钥或请求分布不均会导致热槽，从而导致延迟增加。 ElastiCache 在执行简单的 SET/GET 命令的工作负载中，Serverless 在单个插槽上支持高达 30,000 ecpu/秒（使用从副本读取时为90,000 ecpu/秒）。我们建议您评估密钥和请求在各个插槽中的分布，并确保在您的请求速率超过此限制时实现均匀分配。

对无服务器中的限制问题进行故障排除 ElastiCache

在服务导向型架构和分布式系统中，限制各种服务组件处理 API 调用的速率称为“限制”。这可以平滑峰值，控制组件吞吐量中的不匹配情况，并在出现意外操作事件时实现更可预测的恢复。 ElastiCache Serverless 专为这些类型的架构而设计，大多数 Redis 客户端都内置了针对受限请求的重试功能。一定程度上的限制对应用程序而言不一定是问题，但是持续限制数据工作流中对延迟敏感的部分可能会对用户体验产生负面影响，并会降低系统的整体效率。

当请求在 ElastiCache Serverless 中受到限制时，您应该会看到无服务器指标有所增加。ThrottledRequests ElastiCache 如果您注意到受限的请求数量很多，请考虑以下几点：

- 扩展速度：ElastiCache Serverless会随着您摄取更多数据或请求速率的增长而自动扩展。如果您的应用程序的扩展速度快于Serverless的扩展速度，则您的请求可能会受到限制，而 ElastiCache ElastiCache Serverless可以扩展以适应您的工作负载。 ElastiCache Serverless 通常可以快速增加存储大小，最多需要 10-12 分钟才能将缓存中的存储大小增加一倍。

- 密钥和请求的均匀分布：在 ElastiCache Redis 中，每个插槽的密钥或请求分布不均可能会导致出现热槽。在执行简单的 SET/GET 命令的工作负载中，如果单个插槽的请求速率超过 30,000 ecpus/秒，则热插槽可能会导致请求受限。
- 从副本读取：如果您的应用程序允许，请考虑使用 “从副本读取” 功能。大多数 Redis 客户端可以配置为 “扩展读取”，将读取定向到副本节点。此功能使您能够扩展读取流量。此外， ElastiCache Serverless 会自动将副本请求中的读取路由到与您的应用程序位于同一可用区的节点，从而降低延迟。启用从副本读取后，对于使用简单的 SET/GET 命令的工作负载，您可以在单个插槽上实现高达 90,000 ECPU /秒。

相关主题

- 其他疑难解答步骤
- the section called “最佳实践和缓存策略”

其他疑难解答步骤

在对持续连接问题进行故障排除时，必须验证以下各项 ElastiCache：

主题

- 安全组
- 网络 ACL
- 路由表
- DNS 解析
- 通过服务器端诊断识别问题
- 网络连接验证
- 网络相关限制
- CPU 使用率
- 从服务器端终止的连接
- Amazon EC2 实例的客户端问题排除
- 解剖完成单个请求所花费的时间

安全组

安全组是保护您的 ElastiCache 客户端（EC2 实例、 AWS Lambda 函数、Amazon ECS 容器等）和 ElastiCache 缓存的虚拟防火墙。安全组是有状态的，也就是说在允许传入或传出流量后，对该流量所做的响应将在该特定安全组的上下文中自动获得授权。

有状态功能要求安全组跟踪所有已授权的连接，而且对跟踪的连接有限制。如果达到该限制，新连接将会失败。有关如何识别客户端或 ElastiCache 端是否已达到限制的帮助，请参阅疑难解答部分。

您可以同时为客户机和 ElastiCache 群集分配单个安全组，也可以为每个安全组分配单独的安全组。

在这两种情况下，都需要允许来自源 ElastiCache 端口的 TCP 出站流量和来自同一端口的入站流量到 ElastiCache。Memcached 的默认端口为 11211，Redis 的默认端口为 6379。默认情况下，安全组允许所有出站流量。在这种情况下，只需要目标安全组中的入站规则。

有关更多信息，请参阅访问 Amazon VPC 中 ElastiCache 集群的访问模式。

网络 ACL

网络访问控制列表 (ACL) 是无状态规则。必须在入站和出站两个方向上都允许流量，才能成功。网络 ACL 将分配给子网，而不是特定资源。可以为客户端资源分配相同的 ACL，尤其是当它们位于同一个子网中时。 ElastiCache

默认情况下，网络 ACL 允许所有流量。但可对它们自定义，以拒绝或允许流量。此外，ACL 规则的评估是按顺序进行的，也就是说，匹配流量的编号最小的规则将允许或拒绝该流量。允许 Redis 流量的最低配置为：

客户端网络 ACL：

- 入站规则：
- 规则编号：最好低于所有拒绝规则；
- 类型：自定义 TCP 规则；
- 协议：TCP
- 端口范围：1024 – 65535
- 来源：0.0.0.0/0（或为集群子网创建单独的规则） ElastiCache
- 允许/拒绝：允许

- 出站规则：

- 规则编号：最好低于所有拒绝规则；
- 类型：自定义 TCP 规则；
- 协议：TCP
- 端口范围：6379
- 来源：0.0.0.0/0（或集群子网。 ElastiCache 请记住，使用特定 IP 可能会在故障转移或扩展集群时产生问题）
- 允许/拒绝：允许

ElastiCache 网络 ACL：

- 入站规则：
- 规则编号：最好低于所有拒绝规则；
- 类型：自定义 TCP 规则；
- 协议：TCP
- 端口范围：6379
- 来源：0.0.0.0/0（或为集群子网创建单独的规则） ElastiCache
- 允许/拒绝：允许

- 出站规则：
- 规则编号：最好低于所有拒绝规则；
- 类型：自定义 TCP 规则；
- 协议：TCP
- 端口范围：1024 – 65535
- 来源：0.0.0.0/0（或集群子网。 ElastiCache 请记住，使用特定 IP 可能会在故障转移或扩展集群时产生问题）
- 允许/拒绝：允许

有关更多信息，请参阅网络 ACL。

路由表

与网络 ACL 类似，每个子网可以具有不同的路由表。如果客户端和 ElastiCache 集群位于不同的子网中，请确保它们的路由表允许它们相互访问。

环境越复杂（涉及多个 VPC、动态路由或网络防火墙），排除问题可能会变得越难。请参阅 网络连接验证 以确认您的网络设置是否合适。

DNS 解析

ElastiCache 根据 DNS 名称提供服务端点。可用的端点包括 Configuration、Primary、Reader 和 Node 端点。有关更多信息，请参阅查找连接终端节点。

在故障转移或集群修改的情况下，与端点名称关联的地址可能会发生变化，并将自动更新。

自定义 DNS 设置（即不使用 VPC DNS 服务）可能不知道 ElastiCache提供的 DNS 名称。确保您的系统能够使用诸如dig（如下所示）或之类的系统工具成功解析 ElastiCache 端点nslookup。

```
$ dig +short example.xxxxxx.ng.0001.use1.cache.amazonaws.com
example-001.xxxxxx.0001.use1.cache.amazonaws.com.
1.2.3.4
```

您还可以通过 VPC DNS 服务强制进行名称解析：

```
$ dig +short example.xxxxxx.ng.0001.use1.cache.amazonaws.com @169.254.169.253
example-001.tihewd.0001.use1.cache.amazonaws.com.
1.2.3.4
```

通过服务器端诊断识别问题

CloudWatch 来自 ElastiCache 引擎的指标和运行时信息是识别连接问题潜在来源的常用来源或信息。良好的分析通常从以下项目开始：

- CPU 使用率：Redis 是一个多线程应用程序。但是，每个命令的执行都发生在一个（主）线程中。因此， ElastiCache 提供了指标CPUUtilization和EngineCPUUtilization。EngineCPUUtilization提供专用 Redis 进程的 CPU 使用率以及所有 vCPU CPUUtilization 的使用率。具有多个 vCPU 的节点通常具有不同的 CPUUtilization 和 EngineCPUUtilization 值，第二个值通常更高。高 EngineCPUUtilization 可能是由于请求数量增加或需要大量 CPU 时间才能完成的复杂操作引起的。您可以通过以下方式标识两者：
  - 增加的请求数：检查与 EngineCPUUtilization 模式匹配的其他指标的增加。有用的指标包括：
    - CacheHits 和 CacheMisses：成功的请求数量或在缓存中找不到有效项目的请求的数量。如果未命中与命中之比较高，则应用程序会因无效的请求浪费时间和资源。

  - SetTypeCmds 和 GetTypeCmds：这些与 EngineCPUUtilization 相关的指标可以帮助理解写入请求（由 SetTypeCmds 衡量）还是读取请求（由 GetTypeCmds 衡量）的负载明显更高。如果负载主要是读取，则使用多个只读副本可以在多个节点之间平衡请求，并将主节点留出用于写入。在禁用集群模式的集群中，可通过使用读取器终端节点在应用程序中创建其他连接配置来使用只读副本。 ElastiCache 有关更多信息，请参阅查找连接终端节点。读取操作必须提交到此额外连接。写入操作将通过常规主端点完成。在已启用集群模式的情况下，建议使用支持本机只读副本的库。使用正确的标记，库将能够自动发现集群拓扑、副本节点，通过 READONLY Redis 命令启用读取操作，然后将读取请求提交到副本。

- 增加的连接数：
  - CurrConnections 和 NewConnections：CurrConnection 是数据点集合时已建立的连接数，而 NewConnections 显示的是在期间内创建的连接数。

    创建和处理连接意味着大量的 CPU 开销。此外，创建新连接所需的 TCP 三向握手会对整体响应时间产生负面影响。

    NewConnections每分钟数千个的 ElastiCache 节点表示仅通过几个命令即可创建和使用连接，这不是最佳选择。最佳做法是保持已建立连接并将其重复用于新操作。当客户端应用程序支持并正确实现连接池或持久连接时，可采用此最佳做法。使用连接池时，currConnections 数量没有很大的变化，NewConnections 应该尽可能的低。Redis 通过少量的当前连接提供最佳性能。将当前连接保持为数十或数百个的顺序，可最大限度地减少支持单独连接（如客户端缓冲区和 CPU 周期）的资源使用，以便为连接提供服务。

- 网络吞吐量：
  - 确定带宽： ElastiCache 节点的网络带宽与节点大小成正比。由于应用程序具有不同的特征，因此结果可能会因工作负载而异。例如，小请求比率较高的应用程序对 CPU 使用率的影响往往大于网络吞吐量，而较大的密钥则会导致更高的网络利用率。因此，建议使用实际工作负载测试节点，以便更好地了解限制。

    模拟应用程序的负载可以提供更准确的结果。但是，通过基准工具可以很好地了解限制。
  - 对于主要是读取请求的情况，使用副本进行读取操作将减轻主节点上的负载。如果使用场景主要是写入，则使用许多副本将增加网络使用率。对于写入主节点的所有字节，有 N 个字节将被发送到副本，N 为副本数。写入密集型工作负载的最佳实践是使用 ElastiCache 启用了集群模式的 Redis，这样写入操作就可以在多个分片之间进行平衡，或者扩展到具有更多网络功能的节点类型。
  - CloudWatchmetrics NetworkBytesIn和分别NetworkBytesOut提供进入或离开节点的数据量。 ReplicationBytes是专用于数据复制的流量。

有关更多信息，请参阅 网络相关限制。

- 复杂命令：Redis 命令在单个线程上提供，这意味着按顺序处理请求。单个慢速命令可能会影响其他请求和连接，最终导致超时。对多个值、密钥或数据类型进行操作的命令的使用必须仔细完成。根据参数数量或其输入或输出值的大小，可以阻止或终止连接。

  一个显著例子就是 KEYS 命令。此命令扫描整个密钥空间来搜索给定模式，并在其执行过程中阻止其他命令的执行。Redis 使用“Big O”符号来描述其命令的复杂性。

  密钥命令具有 O (N) 时间复杂性，N 表示数据库中的密钥数。因此，密钥数越大，命令的速度就越慢。KEYS 可能会以不同的方式制造麻烦：如果未使用搜索模式，则该命令会返回所有可用的密钥名称。在含有数千或百万个项目的数据库中，这将会创建大量输出并充满网络缓冲区。

  如果使用了搜索模式，则只有匹配该模式的密钥才会返回到客户端。但是，引擎仍然会扫描整个密钥空间来搜索它，并且完成命令的时间将是相同的。

  KEYS 命令的一个替代选择是 SCAN 命令。此命令会遍历密钥空间并限制特定数量项目中的迭代，避免引擎上的长时间阻塞。

  扫描具有 COUNT 参数，用于设置迭代块的大小。默认值为 10（每次迭代 10 个项目）。

  取决于数据库中的项目数，较小的 COUNT 值数据块将需要更多的迭代才能完成全面扫描，而且较大的值将使引擎在每次迭代中处于繁忙状态。虽然小计数值将使 SCAN 在大数据库变慢，较大的值可能会导致出现 KEYS 中提及的相同问题。

  例如，运行 SCAN 命令（计数值为 10）将需要在具有 100 万个密钥的数据库上进行 10 万次重复操作。如果平均网络往返时间为 0.5 毫秒，则传输请求将耗时大约 5 万毫秒（50 秒）。

  另一方面，如果计数值为 100,0000，则需要一次迭代，并且传输它只需要 0.5 毫秒。但是，在命令完成扫描所有密钥空间之前，该引擎将完全阻止其他操作。

  除了 KEYS 之外，其他几个命令如果使用不当也可能会有害。要查看所有命令及其各自的时间复杂度的列表，请转到 https://redis.io/commands。

  潜在问题的示例：

  - Lua 脚本：Redis 提供了嵌入式 Lua 解释器，允许在服务器端执行脚本。Redis 上的 Lua 脚本在引擎级别执行，而且根据定义，其具有原子性，这意味着在脚本执行过程中不允许运行其他命令或脚本。Lua 脚本提供了直接在 Redis 引擎上运行多个命令、决策算法、数据解析和其他操作的可能性。虽然脚本的原子性和分载应用程序的可能性很诱人，但必须小心使用脚本，并且仅用于小型操作。开 ElastiCache启后，Lua 脚本的执行时间限制为 5 秒。未写入密钥空间的脚本

将在 5 秒后自动终止。为了避免数据损坏和不一致，如果脚本执行在 5 秒内未完成并且在执行过程中有任何写入，则节点将进行故障转移。事务是保证 Redis 中多个相关密钥修改的一致性的替代方案。事务允许执行一个命令块，监视现有密钥以进行修改。如果任何受监视的密钥在事务完成之前发生了更改，则会放弃所有修改。

- 批量删除项目：DEL 命令接受多个参数，这些参数是要删除的密钥名称。删除操作是同步的，如果参数列表很大，或者包含大列表、集合、排序集或哈希（包含多个子项的数据结构），则需要大量 CPU 时间。换句话说，如果具有许多元素，那么即使删除单个密钥也可能需要相当长的时间。DEL 的替代项选择为 UNLINK，这是自 Redis 4 以来可用的异步命令。UNLINK 必须尽可能优先于 DEL。从 Red ElastiCache is 6x 开始，该`lazyfree-lazy-user-del`参数使DEL命令的行为与启用UNLINK时相同。有关更多信息，请参阅 Redis 6.0 参数更改。
- 对多个密钥进行操作的命令：DEL 在先前是作为接受多个实际参数的命令提起，其执行时间直接与之成正比。但是，Redis 提供了更多工作原理类似的命令。例如，MSET 和 MGET 允许一次插入或检索多个字符串键。使用它们可能有助于降低多个单独的 SET 或 GET 命令固有的网络延迟。但是，参数列表过大会影响 CPU 使用率。

  虽然仅仅 CPU 使用率并不是导致连接问题的原因，但是通过多个密钥花费过多时间处理单个或少量命令可能会导致其他请求失败，并增加总体 CPU 使用率。

  密钥的数量及其大小将影响命令的复杂性，从而影响完成时间。

  其他可对多个密钥进行操作的命令示
  例：HMGET、HMSET、MSETNX、PFCOUNT、PFMERGE、SDIFF、SDIFFSTORE、SINTER、SINTERST
  或 ZINTERSTORE。
- 对多种数据类型进行操作的命令：Redis 还提供针对一个或多个密钥执行操作的命令，无论其数据类型如何。 ElastiCache for Redis 提供了监控此类命令KeyBasedCmds的指标。此指标汇总了以下命令在所选时间段内的执行情况：
  - O(N) 复杂性：
    - KEYS
  - O(1)
    - EXISTS
    - OBJECT
    - PTTL
    - RANDOMKEY
    - TTL
    - TYPE

    - EXPIRE
    - EXPIREAT
    - MOVE
    - PERSIST
    - PEXPIRE
    - PEXPIREAT
    - UNLINK (O(N) 来回收内存。但是，内存回收任务发生在一个单独的线程中，并且不会阻塞引擎
  - 根据数据类型不同的复杂性时间：
    - DEL
    - DUMP
    - RENAME 被认为是一个具有 O(1) 复杂性的命令，但在内部执行 DEL。执行时间将根据重命名密钥的大小而变。
    - RENAMENX
    - RESTORE
    - SORT
  - 大哈希：哈希是一种数据类型，允许单个密钥和多个键值子项目。每个哈希可以存储 4,294,967,295 个项目，并且对大哈希执行的操作可能会变得昂贵。类似于 KEYS，哈希具有 HKEYS 命令，该命令具有 O(N) 时间复杂度，N 表示哈希中的项目数。HSCAN 必须优先于 HKEYS 来避免长时间运行的命令。HDEL、HGETALL、HMGET、HMSET 和 HVALS 是应谨慎用于大哈希的命令。
- 其他大数据结构：除了哈希之外，其他数据结构可能是 CPU 密集型的。集、列表、排序集和 Hyperloglog 也可能需要相当长的时间来处理，具体取决于它们的大小和使用的命令。有关这些命令的更多信息，请参阅 https://redis.io/commands。

网络连接验证

查看与 DNS 解析、安全组、网络 ACL 和路由表相关的网络配置后，可以使用 VPC Reachability Analyzer 和系统工具验证连接性。

Reachability Analyzer 将测试网络连接并确认是否满足所有要求和权限。对于以下测试，您将需要您的 VPC 中可用 ElastiCache 节点之一的 ENI ID（弹性网络接口识别）。您可以执行以下操作来查找：

1. 转到 https://console.aws.amazon.com/ec2/v2/home?#NIC:

2. 根据您的 ElastiCache 集群名称或之前从 DNS 验证中获得的 IP 地址筛选接口列表。
3. 记下或以其他方式保存 ENI ID。如果显示多个接口，请查看描述以确认它们属于正确的 ElastiCache 集群，然后从中选择一个。
4. 继续执行下一步骤。
5. 在 https://console.aws.amazon.com/vpc/home? # 上创建分析路径ReachabilityAnalyzer并选择以下选项：

   - 源类型：如果您的 ElastiCache 客户端在 Amazon EC2 实例上运行，则选择实例；如果您的客户端使用其他服务（例如带有 awsvpc 网络的 A AWS Fargate mazon ECS 等） AWS Lambda，则选择网络接口，以及相应的资源 ID（EC2 实例或 ENI ID）；
   - Destination Type（目的地类型）：选择 Network Interface（网络接口），然后在列表中选择 ElastiCache ENI。
   - 目标端口：为 Redis 指定 6379， ElastiCache 为 Memcached 指定 11211。 ElastiCache 这些端口是使用默认配置定义的端口，本示例假定它们未更改。
   - 协议：TCP

创建分析路径并等待结果。如果状态为无法访问，请打开分析详细信息并查看 Analysis Explorer，了解请求被阻止的详细信息。

如果可到达性测试通过，请继续进行系统级别的验证操作。

要验证 ElastiCache 服务端口上的 TCP 连接，Nping请执行以下操作：在 Amazon Linux 上，包nmap中提供，可以测试 ElastiCache 端口上的 TCP 连接，还可以提供网络往返时间来建立连接。使用它来验证 ElastiCache 集群的网络连接和当前延迟，如下所示：

```
$ sudo nping --tcp -p 6379 example.xxxxxx.ng.0001.use1.cache.amazonaws.com

Starting Nping 0.6.40 ( http://nmap.org/nping ) at 2020-12-30 16:48 UTC
SENT (0.0495s) TCP ...
(Output suppressed )

Max rtt: 0.937ms | Min rtt: 0.318ms | Avg rtt: 0.449ms
Raw packets sent: 5 (200B) | Rcvd: 5 (220B) | Lost: 0 (0.00%)
Nping done: 1 IP address pinged in 4.08 seconds
```

默认情况下，nping 会发送 5 个探测器，探测器之间的延迟为 1 秒。您可以使用选项“-c”来增加探测器的数量，并使用“--delay”来更改发送新测试的时间。

如果带有 nping 的测试失败而 VPC Reachability Analyzer 测试通过，请您的系统管理员查看可能基于主机的防火墙规则、非对称路由规则或操作系统级别的任何其他可能的限制。

在 ElastiCache 控制台上，检查 ElastiCache 集群详细信息中是否启用了传输中加密。如果传输中加密已启用，请使用以下命令确认是否可以建立 TLS 会话：

```
openssl s_client -connect example.xxxxxx.use1.cache.amazonaws.com:6379
```

如果连接和 TLS 协商成功，则预计会有大量输出。检查最后一行中可用的返回代码，该值必须为 0 (ok)。如果 openssl 返回不同的内容，请在 https://www.openssl.org/docs/man1.0.2/man1/verify.html#DIAGNOSTICS 中检查错误原因。

如果所有基础架构和操作系统测试均已通过，但您的应用程序仍无法连接 ElastiCache，请检查应用程序配置是否符合 ElastiCache 设置。常见的错误有：

- 您的应用程序不支持 ElastiCache 集群模式，并且 ElastiCache 已启用集群模式；
- 您的应用程序不支持 TLS/SSL，并且 ElastiCache 已启用传输中加密；
- 应用程序支持 TLS/SSL，但没有正确的配置标记或受信任的证书颁发机构；

网络相关限制

- 最大连接数：同时连接的数量有硬限制。每个 ElastiCache 节点允许在所有客户端之间同时连接多达 65,000 个。可以通过上的CurrConnections指标来监控此限制 CloudWatch。但是，客户端也有出站连接限制。在 Linux 上，使用以下命令检查允许的临时端口范围：

  ```
  # sysctl net.ipv4.ip_local_port_range
  net.ipv4.ip_local_port_range = 32768 60999
  ```

  在前面的示例中，将允许从同一源到相同目标 IP（ElastiCache 节点）和端口的 28231 个连接。以下命令显示特定 ElastiCache 节点 (IP 1.2.3.4) 有多少连接：

  ```
  ss --numeric --tcp state connected "dst 1.2.3.4 and dport == 6379" | grep -vE
   '^State' | wc -l
  ```

  如果数量过高，您的系统可能会因尝试处理连接请求而变得过载。建议考虑实施连接池或持久连接等技术，以更好地处理连接。尽可能配置连接池以将最大连接数限制为几百个。此外，建议采用退避逻辑来处理超时或其他连接异常，以避免在出现问题时出现连接损失。

- 网络流量限制：检查 Redis 的以下CloudWatch 指标，以确定 ElastiCache 节点上可能达到的网络限制：
  - NetworkBandwidthInAllowanceExceeded/NetworkBandwidthOutAllowanceExceeded：由于吞吐量超过了聚合带宽限制而形成的网络数据包。

    请注意，写入主节点的每个字节都将被复制到 N 个副本，N 代表副本的数量。具有小节点类型、多个副本和密集型写入请求的集群可能无法应对复制积压。对于这种情况，最佳做法是纵向扩展（更改节点类型）、横向扩展（在已启用集群模式的集群中添加分区）、减少副本数量或最大程度减少写入次数。
  - NetworkConntrackAllowanceExceeded：由于超过了分配给节点的、跨所有安全组跟踪的连接最大数量而形成的数据包。在此期间，新连接可能会失败。
  - NetworkPackets PerSecondAllowanceExceeded：超过每秒最大数据包数。基于高比率小请求的工作负载可能会在达到最大带宽之前达到此限制。

  以上指标是确认节点达到网络限制的理想方法。但是，通过网络指标的高原也可以确认限制。

  如果平稳状态持续了很长时间，则它们之后可能会出现复制滞后、用于缓存的字节增加、可用内存减少、高交换和 CPU 使用率。Amazon EC2 实例同样具有网络限制，这些限制可通过 ENA 驱动程序指标跟踪。具有增强联网支持和 ENA 驱动程序 2.2.10 或更高版本的 Linux 实例可以使用以下命令查看限制计数器：

```
# ethtool -S eth0 | grep "allowance_exceeded"
```

CPU 使用率

CPU 使用率指标是调查的起点，以下项目可以帮助缩小可能存在 ElastiCache 的问题：

- Redis SlowLogs： ElastiCache 默认配置保留了花费超过 10 毫秒才完成的最后 128 条命令。慢速命令的历史记录会在引擎运行时保留，如果发生故障或重启，则会丢失。如果列表达到 128 个条目，旧事件将被删除来为新事件预留空间。慢速事件列表的大小和被视为慢的执行时间可以通过自定义参数组中的参数 slowlog-max-len 和 slowlog-log-slower-than 进行修改。慢速日志列表可以通过在引擎上运行 SLOWLOG GET 128 来进行检索，128 代表报告的最近 128 个慢速命令。每个条目都包含以下字段：

```
1) 1) (integer) 1 ------------> Sequential ID
   2) (integer) 1609010767 --> Timestamp (Unix epoch time)of the Event
   3) (integer) 4823378 -----> Time in microseconds to complete the command.
```

```
   4) 1) "keys" -------------> Command
      2) "*" ----------------> Arguments
   5) "1.2.3.4:57004"-> Source
```

上述事件发生在 12 月 26 日，UTC 19:26:07，耗时 4.8 秒（4823 毫秒）完成，该事件由从客户端 1.2.3.4 请求的 KEYS 命令导致。

在 Linux 上，时间戳可以使用命令日期进行转换：

```
$ date --date='@1609010767'
Sat Dec 26 19:26:07 UTC 2020
```

使用 Python：

```
>>> from datetime import datetime
>>> datetime.fromtimestamp(1609010767)
datetime.datetime(2020, 12, 26, 19, 26, 7)
```

或者在 Windows 上使用 PowerShell：

```
PS D:\Users\user> [datetimeoffset]::FromUnixTimeSeconds('1609010767')
DateTime      : 12/26/2020 7:26:07 PM
UtcDateTime
                : 12/26/2020 7:26:07 PM
LocalDateTime : 12/26/2020 2:26:07 PM
Date          : 12/26/2020 12:00:00 AM
Day           : 26
DayOfWeek
                : Saturday
DayOfYear     : 361
Hour          : 19
Millisecond   : 0
Minute        : 26
Month
                : 12
Offset        : 00:00:00Ticks         : 637446075670000000
UtcTicks
                : 637446075670000000
TimeOfDay     : 19:26:07
Year          : 2020
```

如果短时间内（一分钟内或更短时间）有许多慢速命令，则需要引起关注。查看命令的性质以及如何对其进行优化（请参阅前面的示例）。如果经常报告 O(1) 时间复杂度的命令，请检查前面提到的 CPU 使用率高的其他因素。

- 延迟指标： ElastiCache for Redis 提供了用于监控不同类别命令的平均延迟的 CloudWatch 指标。数据点的计算方法是将类别中命令的执行总数除以期间内的总执行时间。了解延迟指标结果是多个命令的聚合，这一点非常重要。单个命令可能会导致意外结果（如超时），不会对指标产生重大影响。对于这种情况，慢日志事件将是一个更准确的信息来源。以下列表包含可用的延迟指标以及影响它们的相应命令。
  - EvalBasedCmdsLatency: 与 Lua 脚本命令有关，`eval`，; `evalsha`
  - GeoSpatialBasedCmdsLatency: `geodist`, `geohash`, `geopos`, `georadius`, `georadiusbymember`, `geoadd`;
  - GetTypeCmdsLatency: 读取命令，无论数据类型如何；
  - HashBasedCmdsLatency: `hexists`, `hget`, `hgetall`, `hkeys`, `hlen`, `hmget`, `hvals`, `hstrlen`, `hdel`, `hincrby`, `hincrbyfloat`, `hmset`, `hset`, `hsetnx`;
  - HyperLogLogBasedCmdsLatency: `pfselftest`, `pfcount`, `pfdebug`, `pfadd`, `pfmerge`;
  - KeyBasedCmdsLatency: 可以对不同数据类型进行操作的命令：dump、、、、、、、、、、、、exists、、keys、、object、、pttl、、randomkey、、ttl、
  - ListBasedCmdsLatency: lindex、llen、lange、blop、brpop、brpop、broppush、linsert、lpop、lpush、lpush、lpush、ltrim、rprc
  - PubSubBasedCmdsLatency: 取消订阅、发布、发布订阅、取消订阅、订阅、取消订阅；
  - SetBasedCmdsLatency: `scard`, `sdiff`, `sinter`, `sismember`, `smembers`, `srandmember`, `sunion`, `sadd`, `sdiffstore`, `sinterstore`, `smove`, `spop`, `srem`, `sunionstore`;
  - SetTypeCmdsLatency: 无论数据类型如何，都要编写命令；
  - SortedSetBasedCmdsLatency: `zcard`, `zcount`, `zrange`, `zrangebyscore`, `zrank`, `zrevrange`, `zrevrangebyscore`, `zrevrank`, `zscore`, `zrangebylex`, `zrevrangebylex`, `zlexcount`, `zadd`. `zincrby`, `zinterstore`, `zrem`, `zremrangebyrank`, `zremrangebyscore`, `zunionstore`, `zremrangebylex`, `zpopmax`, `zpopmin`, `bzpopmin`, `bzpopmax`;
  - StringBasedCmdsLatency: `bitcount`, `get`, `getbit`, `getrange`, `mget`, `strlen`, `substr`, `bitpos`, `append`, `bitop`, `bitfield`, `decr`, `decrby`, `getset`, `incr`, `incrby`, `incrbyfloat`, `mset`, `msetnx`, `psetex`, `set`, `setbit`, `setex`, `setnx`, `setrange`;
  - StreamBasedCmdsLatency: `xrange`, `xrevrange`, `xlen`, `xread`, `xpending`, `xinfo`, `xadd`, `xgroup`, `readgroup`, `xack`, `xclaim`, `xdel`, `xtrim`, `xsetid`;

- Redis 运行时命令：
  - info commandstats：提供自 Redis 引擎启动以来执行的命令列表、其累积执行数、总执行时间以及每个命令的平均执行时间；
  - 客户端列表：提供当前已连接的客户端列表以及相关信息，如缓冲区使用情况、最后执行的命令等；
- 备份和复制： ElastiCache 对于 2.8.22 之前的 Redis 版本，使用分叉进程来创建备份并处理与副本的完全同步。对于写入密集型使用案例，此方法可能会产生大量内存开销。

  从 ElastiCache Redis 2.8.22 开始， AWS 引入了一种无分支备份和复制方法。新方法可能会延迟写入，以防止出现故障。这两种方法都可能产生较高的 CPU 使用率，导致更长的响应时间，从而导致客户端在执行过程中出现超时。始终检查客户端故障是否发生在备份窗口或在该期间内 `SaveInProgress` 指标是否为 1。建议将备份窗口安排在使用率低的时间段，以最大限度地减少客户端出现问题或备份失败的可能性。

从服务器端终止的连接

Redis ElastiCache 的默认配置可以无限期地保持客户端连接的建立。但是，在某些情况下，可能需要终止连接。例如：

- 客户端应用程序中的漏洞可能会导致连接被遗忘并在空闲状态仍保持为已建立状态。这被称为“连接泄漏”，其结果是在 `CurrConnections` 指标上观察到的已建立连接数量的稳步上升。这种行为可能会导致客户端或 ElastiCache 端出现饱和。当客户端无法立即修复时，一些管理员会在其 ElastiCache 参数组中设置 “超时” 值。超时是允许空闲连接保留的时间（以秒为单位）。如果客户端在此期间内未提交任何请求，则 Redis 引擎将在连接达到超时值后立即终止连接。较小的超时值可能会导致不必要的断开连接，客户端需要正确处理它们并重新连接，因此会导致延迟。
- 用于存储密钥的内存与客户端缓冲区共享。具有较大请求或响应的速度较慢的客户端可能需要大量内存来处理其缓冲区。Redis 配置 ElastiCache 的默认设置不限制常规客户端输出缓冲区的大小。如果达到 `maxmemory` 限制，引擎将尝试移出项目以满足缓冲区使用情况。在内存极低的情况下，fo ElastiCache r Redis 可能会选择断开消耗大量客户端输出缓冲区的客户端的连接，以释放内存并保持集群的运行状况。

  可以通过自定义配置来限制客户端缓冲区的大小，达到限制的客户端将断开连接。但客户端应能够处理意外断开的连接。用于处理常规客户端缓冲区大小的参数如下：
  - client-query-buffer-limit: 单个输入请求的最大大小；

- client-output-buffer-limit-normal-soft-limit：客户端连接的软限制。如果保持在软限制之上的时间超过（以秒为单位）上 client-output-buffer-limit定义的时间，normal-soft-seconds 或者达到硬限制，则连接将被终止；
- client-output-buffer-limit-normal-soft-seconds: 允许的连接时间超过 client-output-buffer-limit-normal-soft-limit;
- client-output-buffer-limit-normal-hard-limit：达到此限制的连接将立即终止。

除了常规客户端缓冲区之外，以下选项控制副本节点和 Pub/Sub（发布/订阅）客户端的缓冲区：

- client-output-buffer-limit-replica-hard-limit;
- client-output-buffer-limit-replica-soft-seconds;
- client-output-buffer-limit-replica-hard-limit;
- client-output-buffer-limit-pubsub-soft-limit;
- client-output-buffer-limit-pubsub-soft-seconds;
- client-output-buffer-limit-pubsub-hard-limit;

Amazon EC2 实例的客户端问题排除

客户端的负载和响应能力也会影响对的请求 ElastiCache。在排除间歇性连接或超时问题时，需要仔细检查 EC2 实例和操作系统限制。需要注意的一些关键点：

- CPU：
  - EC2 实例 CPU 使用率：确保 CPU 未饱和或接近 100%。历史分析可以通过以下方式完成 CloudWatch，但请记住，数据点的粒度要么是 1 分钟（启用详细监控），要么是 5 分钟；
  - 使用可突增 EC2 实例时，请确保他们的 CPU 积分余额没有被耗尽。此信息可在`CPUCreditBalance` CloudWatch 指标上找到。
  - 短时间的高CPU使用率可能会导致超时，而不会反映100%的利用率。 CloudWatch这种情况需要使用操作系统工具（如 `top`、`ps` 和 `mpstat`）进行实时监控。
- 网络
  - 根据实例功能，检查网络吞吐量是否在可接受的值之内。有关更多信息，请参阅 Amazon EC2 实例类型。
  - 在具有 `ena` 增强型网络驱动程序的实例上，请检查 ENA 统计数据查看超时或超出限制情况。以下统计数据对于确认网络限制饱和度很有用：
    - `bw_in_allowance_exceeded`/`bw_out_allowance_exceeded`：由于入站或出站吞吐量过高而形成的数据包数量；

- conntrack_allowance_exceeded：由于安全组连接跟踪限制而丢弃的数据包数。当此限制饱和时，新连接将失败；
- linklocal_allowance_exceeded：由于通过 VPC DNS 对实例元数据的请求过多而丢弃的数据包数。对所有服务的限制是每秒 1024 个数据包；
- pps_allowance_exceeded：由于每秒数据包过多而丢弃的数据包数。当网络流量由每秒数千个或数百万个非常小的请求组成时，可能会达到 PPS 限制。 ElastiCache 可以优化流量，以便通过管道或命令更好地利用网络数据包，这些管道或命令可以同时执行多个操作，比如，MGET而不是GET。

解剖完成单个请求所花费的时间

- 在网络上：Tcpdump和Wireshark（命令行上的 tshark）是便捷的工具，用于了解请求在网络上传输、启动 ElastiCache 引擎并获得返回所花费的时间。以下示例突出显示使用以下命令创建的单个请求：

```
$ echo ping | nc example.xxxxxx.ng.0001.use1.cache.amazonaws.com 6379
+PONG
```

与上面的命令并行，tcpdump 经过执行并返回：

```
$ sudo tcpdump -i any -nn port 6379 -tt
tcpdump: verbose output suppressed, use -v or -vv for full protocol decode
listening on any, link-type LINUX_SLL (Linux cooked), capture size 262144 bytes
1609428918.917869 IP 172.31.11.142.40966
    > 172.31.11.247.6379: Flags [S], seq 177032944, win 26883, options [mss
 8961,sackOK,TS val 27819440 ecr 0,nop,wscale 7], length 0
1609428918.918071 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [S.], seq
 53962565, ack 177032945, win
    28960, options [mss 1460,sackOK,TS val 3788576332 ecr 27819440,nop,wscale 7],
 length 0
1609428918.918091 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 1, win
 211, options [nop,nop,TS val 27819440 ecr 3788576332], length 0
1609428918.918122
    IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [P.], seq 1:6, ack 1, win 211,
 options [nop,nop,TS val 27819440 ecr 3788576332], length 5: RESP "ping"
1609428918.918132 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [F.], seq 6, ack
    1, win 211, options [nop,nop,TS val 27819440 ecr 3788576332], length 0
1609428918.918240 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [.], ack 6, win
 227, options [nop,nop,TS val 3788576332 ecr 27819440], length 0
```

```
1609428918.918295
    IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [P.], seq 1:8, ack 7, win 227,
 options [nop,nop,TS val 3788576332 ecr 27819440], length 7: RESP "PONG"
1609428918.918300 IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 8, win
    211, options [nop,nop,TS val 27819441 ecr 3788576332], length 0
1609428918.918302 IP 172.31.11.247.6379 > 172.31.11.142.40966: Flags [F.], seq 8, ack
 7, win 227, options [nop,nop,TS val 3788576332 ecr 27819440], length 0
1609428918.918307
    IP 172.31.11.142.40966 > 172.31.11.247.6379: Flags [.], ack 9, win 211, options
 [nop,nop,TS val 27819441 ecr 3788576332], length 0
^C
10 packets captured
10 packets received by filter
0 packets dropped by kernel
```

从上面的输出中，我们可以确认 TCP 三向握手是在 222 微秒（918091 – 917869）内完成的，并且在 173 微秒（918295 – 918122）内提交并返回了 ping 命令。

请求关闭连接耗费了 438 微秒（918307 – 917869）。这些结果将确认网络和引擎响应时间良好，调查可以集中在其他组件上。

- 在操作系统上：Strace 可以帮助识别操作系统级别的时间差。对实际应用程序的分析将更加广泛，建议使用专门的应用程序分析器或调试器。以下示例仅显示操作系统基本组件是否按预期工作，其他内容可能需要进一步调查。通过 strace 使用相同的 Redis PING 命令，我们得到：

```
$ echo ping | strace -f -tttt -r -e trace=execve,socket,open,recvfrom,sendto
 nc example.xxxxxx.ng.0001.use1.cache.amazonaws.com (http://
example.xxxxxx.ng.0001.use1.cache.amazonaws.com/)
    6379
1609430221.697712 (+ 0.000000) execve("/usr/bin/nc", ["nc",
 "example.xxxxxx.ng.0001.use"..., "6379"], 0x7fffede7cc38 /* 22 vars */) = 0
1609430221.708955 (+ 0.011231) socket(AF_UNIX, SOCK_STREAM|SOCK_CLOEXEC|
SOCK_NONBLOCK, 0) = 3
1609430221.709084
    (+ 0.000124) socket(AF_UNIX, SOCK_STREAM|SOCK_CLOEXEC|SOCK_NONBLOCK, 0) = 3
1609430221.709258 (+ 0.000173) open("/etc/nsswitch.conf", O_RDONLY|O_CLOEXEC) = 3
1609430221.709637 (+ 0.000378) open("/etc/host.conf", O_RDONLY|O_CLOEXEC) = 3
1609430221.709923
    (+ 0.000286) open("/etc/resolv.conf", O_RDONLY|O_CLOEXEC) = 3
1609430221.711365 (+ 0.001443) open("/etc/hosts", O_RDONLY|O_CLOEXEC) = 3
1609430221.713293 (+ 0.001928) socket(AF_INET, SOCK_DGRAM|SOCK_CLOEXEC|SOCK_NONBLOCK,
 IPPROTO_IP) = 3
1609430221.717419
```

```
    (+ 0.004126) recvfrom(3, "\362|
\201\200\0\1\0\2\0\0\0\0\rnotls20201224\6tihew"..., 2048, 0, {sa_family=AF_INET,
 sin_port=htons(53), sin_addr=inet_addr("172.31.0.2")}, [28->16]) = 155
1609430221.717890 (+ 0.000469) recvfrom(3,
 "\204\207\201\200\0\1\0\1\0\0\0\0\rnotls20201224\6tihew"...,
    65536, 0, {sa_family=AF_INET, sin_port=htons(53),
 sin_addr=inet_addr("172.31.0.2")}, [28->16]) = 139
1609430221.745659 (+ 0.027772) socket(AF_INET, SOCK_STREAM, IPPROTO_TCP) = 3
1609430221.747548 (+ 0.001887) recvfrom(0, 0x7ffcf2f2ca50, 8192,
    0, 0x7ffcf2f2c9d0, [128]) = -1 ENOTSOCK (Socket operation on non-socket)
1609430221.747858 (+ 0.000308) sendto(3, "ping\n", 5, 0, NULL, 0) = 5
1609430221.748048 (+ 0.000188) recvfrom(0, 0x7ffcf2f2ca50, 8192, 0, 0x7ffcf2f2c9d0,
 [128]) = -1 ENOTSOCK
    (Socket operation on non-socket)
1609430221.748330 (+ 0.000282) recvfrom(3, "+PONG\r\n", 8192, 0, 0x7ffcf2f2c9d0,
 [128->0]) = 7
+PONG
1609430221.748543 (+ 0.000213) recvfrom(3, "", 8192, 0, 0x7ffcf2f2c9d0, [128->0]) = 0
1609430221.752110
    (+ 0.003569) +++ exited with 0 +++
```

在上面的示例中，完成该命令所耗费的时间稍多于 54 毫秒（752110 – 697712 = 54398 微秒）。

实例化 NC 并执行名称解析耗费了大量的时间（从 697712 到 717890，大约 20 毫秒），之后创建 TCP 套接字需要 2 毫秒（745659 到 747858），提交和接收请求的响应需要 0.4 毫秒（747858 到 748330）。

亚马逊的安全 ElastiCache

云安全 AWS 是重中之重。作为 AWS 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 AWS 的责任。责任共担模式将其描述为云的 安全性和云中 的安全性：

- 云安全 — AWS 负责保护在 AWS 云中运行 AWS 服务的基础架构。 AWS 还为您提供可以安全使用的服务。作为 AWS 合规性计划的一部分，第三方审核人员将定期测试和验证安全性的有效性。要了解适用于亚马逊的合规计划 ElastiCache，请参阅合规计划范围内的AWS 服务。
- 云端安全-您的责任由您使用的 AWS 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

本文档可帮助您了解在使用 Amazon 时如何应用分担责任模型 ElastiCache。以下主题向您展示如何配置 Amazon ElastiCache 以满足您的安全与合规目标。您还将学习如何使用其他 AWS 服务来帮助您监控和保护您的 Amazon ElastiCache 资源。

主题

- Amazon ElastiCache 中的数据保护
- 互联网络流量隐私保护
- 适用于亚马逊的身份和访问管理 ElastiCache
- Amazon 合规性验证 ElastiCache
- Amazon ElastiCache 中的恢复能力
- AWS ElastiCache 中的基础设施安全性
- 中的服务更新 ElastiCache
- 常见漏洞和披露 (CVE)：Redis 中 ElastiCache 已解决的安全漏洞

Amazon ElastiCache 中的数据保护

AWS 责任共担模式适用于 AWS ElastiCache（ElastiCache）中的数据保护。如该模式中所述，AWS 负责保护运行所有AWS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AWS服务的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。

出于数据保护目的，我们建议您保护 AWS 账户凭证，并使用 AWS Identity and Access Management（IAM）设置各个账户。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数据：

- 对每个账户使用多重身份验证 (MFA)。
- 使用 TLS 与 AWS 资源进行通信。
- 使用 AWS CloudTrail 设置 API 和用户活动日志记录。
- 使用 AWS 加密解决方案以及 AWS 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），其有助于发现和保护存储在 Simple Storage Service（Amazon S3）中的个人数据。

我们强烈建议您切勿将敏感的可识别信息（例如您客户的账号）放入自由格式字段（例如 Name（名称）字段）。这包括通过控制台、API、AWS CLI 或 AWS SDK 使用 ElastiCache 或其他 AWS 服务的情况。您输入到 ElastiCache 或其他服务中的任何数据都可能会被选取以包含在诊断日志中。当您向外部服务器提供 URL 时，请勿在 URL 中包含凭证信息来验证您对该服务器的请求。

主题

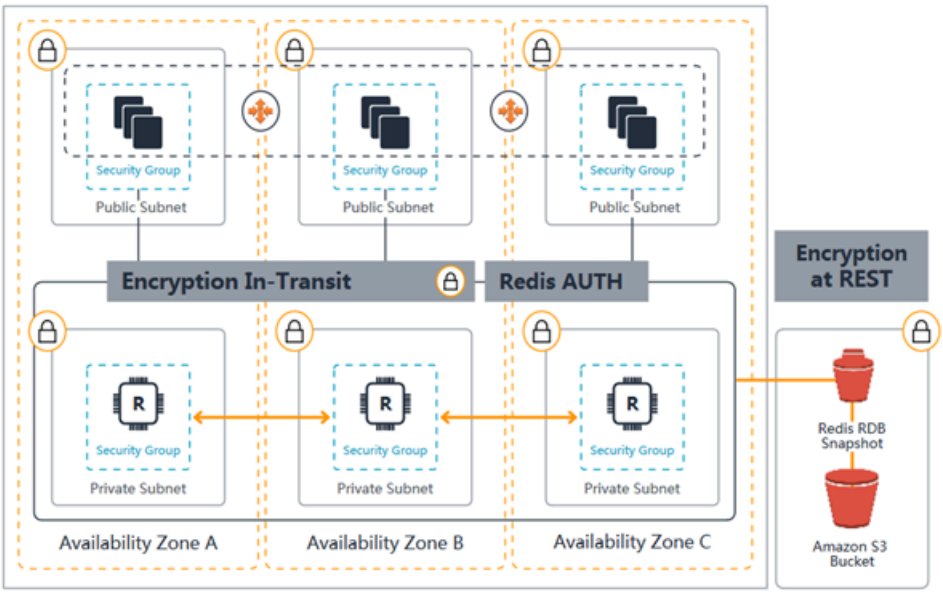
Amazon ElastiCache 中的数据安全性

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

Amazon ElastiCache for Redis 面向运行 Redis 版本 3.2.6（已计划终止生命周期，请参阅 Redis 版本生命周期终止计划）、4.0.10 或更高版本的缓存，为其中的数据提供了可选加密功能：

- 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缓存与应用程序之间移动数据。
- 静态加密可在同步和备份操作期间对磁盘上的数据进行加密。

Amazon ElastiCache for Redis 还支持使用 IAM 或 Redis AUTH 对用户进行身份验证，以及使用基于角色的访问控制（RBAC）对用户操作进行授权。



ElastiCache for Redis 安全图

主题

- ElastiCache 传输中加密 (TLS)
- ElastiCache 中的静态加密
- 身份验证和授权

ElastiCache 传输中加密 (TLS)

为了帮助保护您的数据安全，Amazon ElastiCache 和 Amazon EC2 提供了防止未经授权访问服务器上数据的机制。通过提供传输中加密功能， ElastiCache 为您提供一种工具，用于在数据从一个位置移动到另一个位置时帮助保护数据。

所有无服务器缓存均启用了传输中加密。对于自行设计的集群，在创建复制组时，您可将参数 `TransitEncryptionEnabled` 设置为 `true`（CLI：`--transit-encryption-enabled`），以此在复制组中启用传输中加密。无论您是使用、还是 ElastiCache API 创建复制组 AWS Management Console AWS CLI，都可以执行此操作。

主题

- 传输中加密概览
- 传输中加密的条件
- 传输中加密最佳实践

- 另请参阅
- 启用传输中加密
- ElastiCache 使用 redis-cli 通过传输加密连接到亚马逊 Redis
- 使用 Python 在自行设计的 Redis 集群上启用传输中加密
- 启用传输中加密时的最佳实践

传输中加密概览

Amazon ElastiCache 传输中加密是一项功能，可让您在数据从一个位置传输到另一个地点时，在最脆弱的地方提高数据的安全性。由于在端点加密和解密数据时需要进行一些处理，因此启用传输中加密会对性能产生一些影响。应对使用和不使用传输中加密的数据进行基准测试，以确定对使用案例的性能影响。

ElastiCache 传输中加密实现了以下功能：

- 加密客户端连接：客户端与缓存节点的连接采用 TLS 加密。
- 加密服务器连接：对集群中在节点之间移动的数据进行了加密。
- 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。
- 客户端身份验证 – 使用 Redis AUTH 功能，服务器可以对客户端进行身份验证。

传输中加密的条件

在规划自己设计的集群实施时，应牢记以下对 Amazon ElastiCache 传输中加密的限制：

- 在运行以下 Redis 版本的复制组上支持传输中加密：3.2.6、4.0.10 和更高版本。
- 运行 Redis 版本 7 及更高版本的复制组支持修改现有集群的传输中加密设置。
- 只有在 Amazon VPC 中运行的复制组支持传输中加密。
- 运行以下节点类型的复制组不支持传输中加密：M1、M2。

  有关更多信息，请参阅 受支持的节点类型。
- 通过显式将参数 `TransitEncryptionEnabled` 设置为 `true` 可启用传输中加密。
- 确保您的缓存客户端支持 TLS 连接，并且您已在客户端配置中启用传输中加密。
- 对于 ElastiCache 版本 6 及更高版本，所有 AWS 区域都不建议使用旧的 TLS 1.0 和 TLS 1.1。ElastiCache 将在 2025 年 5 月 8 日之前继续支持 TLS 1.0 和 1.1。客户必须在该日期之前更新其客户端软件。

传输中加密最佳实践

- 由于在端点加密和解密数据时需要进行一些处理，因此实现传输中加密会降低性能。使用自己的数据，对传输中加密进行基准测试，然后与不加密情况进行比较，以确定其对实现性能的影响。
- 由于创建新连接的成本可能非常高，您可以通过保留 TLS 连接来减小传输中加密对性能的影响。

另请参阅

- ElastiCache 中的静态加密
- 使用 Redis AUTH 命令进行身份验证
- 使用基于角色的访问控制（RBAC）对用户进行身份验证
- Amazon VPC 和 ElastiCache 安全性
- 适用于亚马逊的身份和访问管理 ElastiCache

启用传输中加密

所有无服务器缓存均启用了传输中加密。在自行设计的集群上，您可以使用 AWS Management Console、AWS CLI 或 ElastiCache API 启用传输中加密。

使用 AWS Management Console 启用传输中加密

使用 AWS Management Console 为新的自行设计集群启用传输中加密

在设计自己的集群时，采用“轻松创建”方法的“开发/测试”和“生产”配置均启用了传输中加密。在您选择自己的配置时，请进行以下选择：

- 选择 3.2.6、4.0.10 或更高的引擎版本。
- 单击传输中加密选项的启用旁边的复选框。

有关这个分步过程，请参阅以下内容：

- 创建 Redis（已禁用集群模式）集群（控制台）
- 创建 Redis（已启用集群模式）集群（控制台）

使用 AWS Management Console 为现有的自行设计集群启用传输中加密

启用传输中加密分为两步，您必须先将传输加密模式设置为 `preferred`。此模式允许您的 Redis 客户端使用加密和未加密的连接进行连接。将所有 Redis 客户端迁移为使用加密连接后，您可以修改集群配置以将传输加密模式设置为 `required`。将传输加密模式设置为 `required` 将删除所有未加密的连接，并且仅允许加密连接。

第 1 步：将 Transit encryption mode（传输加密模式）设置为 Preferred（首选）

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 在左侧导航窗格上列出的 ElastiCache 资源中，选择 Redis 缓存。
3. 选择要更新的 Redis 缓存。
4. 选择 Actions（操作）下拉列表，然后选择 Modify（修改）。
5. 在 Security（安全）部分的 Encryption in transit（传输中加密）下，选择 Enable（启用）。
6. 选择 Preferred（首选）作为 Transit encryption mode（传输加密模式）。
7. 选择 Preview changes（预览更改），然后保存更改。

将所有 Redis 客户端迁移为使用加密连接后：

第 2 步：将 Transit encryption mode（传输加密模式）设置为 Required（必需）

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 在左侧导航窗格上列出的 ElastiCache 资源中，选择 Redis 缓存。
3. 选择要更新的 Redis 缓存。
4. 选择 Actions（操作）下拉列表，然后选择 Modify（修改）。
5. 在 Security（安全）部分中，选择 Required（必需）作为 Transit encryption mode（传输加密模式）。
6. 选择 Preview changes（预览更改），然后保存更改。

使用 AWS CLI 启用传输中加密

要在使用 AWS CLI 创建 Redis 复制组时启用传输中加密，请使用参数 `transit-encryption-enabled`。

在新的 Redis 自行设计集群（已禁用集群模式）上启用传输中加密（CLI）

使用 AWS CLI 操作 `create-replication-group` 和以下参数创建启用传输中加密的有副本 Redis 复制组：

关键参数：

- **`--engine`** – 必须为 `redis`。
- **`--engine-version`** – 必须是 3.2.6、4.0.10 或更高版本。
- **`--transit-encryption-enabled`** – 必填项。如果要启用传输中加密，还必须为 `--cache-subnet-group` 参数提供值。
- **`--num-cache-clusters`** – 必须至少为 1。此参数的最大值为 6。

有关更多信息，请参阅下列内容：

- 从头开始创建 Redis（已禁用集群模式）复制组（AWS CLI）
- create-replication-group

在新的 Redis 自行设计集群（已启用集群模式）上启用传输中加密（CLI）

使用 AWS CLI 操作 `create-replication-group` 和以下参数创建启用了传输中加密的 Redis（已启用集群模式）复制组：

关键参数：

- **`--engine`** – 必须为 `redis`。
- **`--engine-version`** – 必须是 3.2.6、4.0.10 或更高版本。
- **`--transit-encryption-enabled`** – 必填项。如果要启用传输中加密，还必须为 `--cache-subnet-group` 参数提供值。
- 使用以下参数集之一指定复制组的节点组配置：
  - **`--num-node-groups`** – 指定此复制组中的分片数（节点组）。此参数的最大值为 500。

    `--replicas-per-node-group` – 指定每个节点组中的副本节点数。此处指定的值适用于此复制组中的所有分片。此参数的最大值为 5。
  - **`--node-group-configuration`** – 分别指定每个分片的配置。

有关更多信息，请参阅下列内容：

- 从头开始创建 Redis（已启用集群模式）复制组（AWS CLI）
- create-replication-group

使用 AWS CLI 为现有集群启用传输中加密

启用传输中加密分为两步，您必须先将传输加密模式设置为 `preferred`。此模式允许您的 Redis 客户端使用加密和未加密的连接进行连接。将所有 Redis 客户端迁移为使用加密连接后，您可以修改集群配置以将传输加密模式设置为 `required`。将传输加密模式设置为 `required` 将删除所有未加密的连接，并且仅允许加密连接。

使用 AWS CLI 操作 `modify-replication-group` 和以下参数更新禁用了传输中加密的 Redis（已启用集群模式）复制组。

启用传输中加密

1. 使用以下参数将传输加密模式设置为 `preferred`
   - **`--transit-encryption-enabled`** – 必填项。
   - **`--transit-encryption-mode`** - 必须设置为 `preferred`。
2. 使用以下参数将传输加密模式设置为 `required`：
   - **`--transit-encryption-enabled`** – 必填项。
   - **`--transit-encryption-mode`** - 必须设置为 `required`。

ElastiCache 使用 redis-cli 通过传输加密连接到亚马逊 Redis

要访问启用传输中加密 ElastiCache 的 Redis 缓存的数据，您可以使用支持安全套接字层 (SSL) 的客户端。您也可以在 Amazon Linux 和 Amazon Linux 2 上使用具有 TLS/SSL 的 redis-cli。如果您的客户端不支持 TLS，则可以在客户端主机上，使用 `stunnel` 命令创建连接到 Redis 节点的 SSL 隧道。

与 Linux 的加密连接

要使用 redis-cli 连接到亚马逊 Linux 2023、亚马逊 Linux 2 或亚马逊 Linux 上启用传输加密的 Redis 集群，请按照以下步骤操作。

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. 在您的 EC2 实例的命令提示符处，键入适用于您正在使用的 Linux 版本的相应命令。

   Amazon Linux 2023

如果使用亚马逊 Linux 2023，请输入以下内容：

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

然后键入以下命令，用集群的终端节点和端口替换本示例中显示的内容。

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

有关查找端点的更多信息，请参阅查找您的节点端点。

Amazon Linux 2

如果使用亚马逊 Linux 2，请输入以下内容：

```
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

如果使用亚马逊 Linux，请输入以下内容：

```
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/
```

在 Amazon Linux 上，您可能还需要执行以下额外步骤：

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

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

4. 要连接到启用了加密和身份验证的集群，请输入以下命令：

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

> **Note**
>
> 如果你在亚马逊 Linux 2023 上安装 redis6，你现在可以使用`redis6-cli`以下命令代替：`redis-cli`
>
> ```
> redis6-cli -h Primary or Configuration Endpoint --tls -p 6379
> ```

与 stunnel 的加密连接

要使用 redis-cli 通过 stunnel 连接到启用传输中加密的 Redis 集群，请按照以下步骤操作。

1. 使用 SSH 连接到您的客户端并安装 `stunnel`。

```
sudo yum install stunnel
```

2. 使用下面提供的输出作为模板，运行以下命令`'/etc/stunnel/redis-cli.conf'`同时创建和编辑文件，将 ElastiCache 适用于 Redis 的集群终端节点添加到一个或多个连接参数中。

```
vi /etc/stunnel/redis-cli.conf


fips = no
setuid = root
setgid = root
pid = /var/run/stunnel.pid
debug = 7
delay = yes
options = NO_SSLv2
options = NO_SSLv3
[redis-cli]
   client = yes
   accept = 127.0.0.1:6379
   connect = primary.ssltest.wif01h.use1.cache.amazonaws.com:6379
[redis-cli-replica]
   client = yes
```

```
  accept = 127.0.0.1:6380
  connect = ssltest-02.ssltest.wif01h.use1.cache.amazonaws.com:6379
```

在此示例中，配置文件具有两个连接，即 `redis-cli` 和 `redis-cli-replica`。参数设置如下所示：

- client 设置为 yes（是）以指定此 Stunnel 实例是客户端。
- accept 设置为客户端 IP。在此示例中，主 IP 设置为端口 6379 上的 Redis 默认 127.0.0.1。副本必须调用另一个端口并设置为 6380。您可以使用临时端口 1024-65535。有关更多信息，请参阅 Amazon VPC 用户指南中的临时端口。
- connect 设置为 Redis 服务器端点。有关更多信息，请参阅查找连接端点。

3. 启动 `stunnel`。

```
sudo stunnel /etc/stunnel/redis-cli.conf
```

使用 `netstat` 命令确认隧道已启动。

```
sudo netstat -tulnp | grep -i stunnel

tcp        0      0 127.0.0.1:6379              0.0.0.0:*                   LISTEN
     3189/stunnel
tcp        0      0 127.0.0.1:6380              0.0.0.0:*                   LISTEN
     3189/stunnel
```

4. 使用隧道的本地端点连接到加密的 Redis 节点。

- 如果在创建 Redis 集群时 ElastiCache 未使用身份验证密码，则此示例使用 redis-cli 在 Amazon Linux 上使用 redis-cli ElastiCache 的完整路径连接到 for Redis 服务器：

```
/home/ec2-user/redis-stable/src/redis-cli -h localhost -p 6379
```

如果在创建 Redis 集群期间未使用 AUTH 密码，此示例将在 Amazon Linux 上使用 redis-cli 通过其完整路径连接到 Redis 服务器：

```
 /home/ec2-user/redis-stable/src/redis-cli -h localhost -p 6379 -a my-secret-
password
```

或

- 将目录更改为 redis-stable 并执行以下操作：

  如果在创建 Redis 集群时 ElastiCache 未使用身份验证密码，则此示例使用 redis-cli 在 Amazon Linux 上使用 redis-cli ElastiCache 的完整路径连接到 for Redis 服务器：

  ```
  src/redis-cli -h localhost -p 6379
  ```

  如果在创建 Redis 集群期间未使用 AUTH 密码，此示例将在 Amazon Linux 上使用 redis-cli 通过其完整路径连接到 Redis 服务器：

  ```
  src/redis-cli -h localhost -p 6379 -a my-secret-password
  ```

此示例使用 Telnet 连接到 Redis 服务器。

```
telnet localhost 6379

Trying 127.0.0.1...
Connected to localhost.
Escape character is '^]'.
auth MySecretPassword
+OK
get foo
$3
bar
```

5. 要停止并关闭 SSL 隧道，请对 Stunnel 过程执行 `pkill` 操作。

```
sudo pkill stunnel
```

使用 Python 在自行设计的 Redis 集群上启用传输中加密

以下指南将演示如何在 Redis 7.0 集群上启用传输中加密，该集群最初是在禁用传输中加密的情况下创建的。在此过程中，TCP 和 TLS 客户端将继续与集群通信，无需停机。

Boto3 将从环境变量中获取它所需的凭证（`aws_access_key_id`、`aws_secret_access_key` 和 `aws_session_token`）。这些凭证将提前粘贴到一个 bash 终端中，我们将在该同一个终端中运行 `python3` 来处理本指南中显示的 Python 代码。以下示例中的代码从一个 EC2 实例进行处理，该实例在一个 VPC 中启动，该同一个 VPC 将用于在其中创建 ElastiCache Redis 集群。

> **Note**
>
> - 以下示例使用 boto3 SDK for ElastiCache 管理操作（集群或用户创建），并使用 redis-py/redis-py-cluster 处理数据。
> - 您必须至少使用 boto3 版本 (=~) 1.26.39，才能通过集群修改 API 使用在线 TLS 迁移。
> - ElastiCache 仅对于版本 7.0 或更高版本的 Redis 集群支持在线 TLS 迁移。因此，如果您的集群运行的是 7.0 之前的 Redis 版本，则需要升级集群的 Redis 版本。有关版本差异的更多信息，请参阅主要版本行为和兼容性差异。

主题

定义将启动 ElastiCache Redis 集群的字符串常量

首先，让我们定义一些简单的 Python 字符串常量，这些常量将保留创建 ElastiCache 集群所需的 AWS 实体的名称，例如 `security-group`、`Cache Subnet group` 和 `default parameter group`。所有这些 AWS 实体都必须事先在您愿意使用的区域的 AWS 账户中创建。

```
#Constants definitions
SECURITY_GROUP = "sg-0492aa0a29c558427"
CLUSTER_DESCRIPTION = "This cluster has been launched as part of the online TLS
 migration user guide"
EC_SUBNET_GROUP = "client-testing"
DEFAULT_PARAMETER_GROUP_REDIS_7_CLUSTER_MODE_ENABLED = "default.redis7.cluster.on"
```

为集群配置定义类

现在，让我们定义一些简单的 Python 类，这些类将表示集群的配置，而这些配置将保留有关集群的元数据，例如 Redis 版本、实例类型以及是启用还是禁用传输中加密（TLS）。

```
#Class definitions
```

```
class Config:
    def __init__(
        self,
        instance_type: str = "cache.t4g.small",
        version: str = "7.0",
        multi_az: bool = True,
        TLS: bool = True,
        name: str = None,
    ):
        self.instance_type = instance_type
        self.version = version
        self.multi_az = multi_az
        self.TLS = TLS
        self.name = name or f"tls-test"

    def create_base_launch_request(self):
        return {
            "ReplicationGroupId": self.name,
            "TransitEncryptionEnabled": self.TLS,
            "MultiAZEnabled": self.multi_az,
            "CacheNodeType": self.instance_type,
            "Engine": "redis",
            "EngineVersion": self.version,
            "CacheSubnetGroupName": EC_SUBNET_GROUP ,
            "CacheParameterGroupName":
 DEFAULT_PARAMETER_GROUP_REDIS_7_CLUSTER_MODE_ENABLED ,
            "ReplicationGroupDescription": CLUSTER_DESCRIPTION,
            "SecurityGroupIds": [SECURITY_GROUP],
        }

class ConfigCME(Config):
    def __init__(
        self,
        instance_type: str = "cache.t4g.small",
        version: str = "7.0",
        multi_az: bool = True,
        TLS: bool = True,
        name: str = None,
        num_shards: int = 2,
        num_replicas_per_shard: int = 1,
    ):
        super().__init__(instance_type, version, multi_az, TLS, name)
        self.num_shards = num_shards
        self.num_replicas_per_shard = num_replicas_per_shard
```

```
    def create_launch_request(self) -> dict:
        launch_request = self.create_base_launch_request()
        launch_request["NumNodeGroups"] = self.num_shards
        launch_request["ReplicasPerNodeGroup"] = self.num_replicas_per_shard
        return launch_request
```

定义一个表示集群本身的类

现在，让我们定义一些简单的 Python 类，它们将表示 ElastiCache Redis 集群本身。该类将有一个客户端字段，此字段将保留用于执行 ElastiCache 管理操作（例如，创建集群和查询 ElastiCache API）的 boto3 客户端。

```
import botocore.config
import boto3

# Create boto3 client
def init_client(region: str = "us-east-1"):
    config = botocore.config.Config(retries={"max_attempts": 10, "mode": "standard"})
    init_request = dict()
    init_request["config"] = config
    init_request["service_name"] = "elasticache"
    init_request["region_name"] = region
    return boto3.client(**init_request)


class ElastiCacheClusterBase:
    def __init__(self, name: str):
        self.name = name
        self.elasticache_client = init_client()

    def get_first_replication_group(self):
        return self.elasticache_client.describe_replication_groups(
        ReplicationGroupId=self.name
        )["ReplicationGroups"][0]

    def get_status(self) -> str:
        return self.get_first_replication_group()["Status"]

    def get_transit_encryption_enabled(self) -> bool:
        return self.get_first_replication_group()["TransitEncryptionEnabled"]

    def is_available(self) -> bool:
```

```
        return self.get_status() == "available"

    def is_modifying(self) -> bool:
        return self.get_status() == "modifying"

    def wait_for_available(self):
        while True:
            if self.is_available():
                break
            else:
                time.sleep(5)

    def wait_for_modifying(self):
        while True:
            if self.is_modifying():
                break
            else:
                time.sleep(5)

    def delete_cluster(self) -> bool:
        self.elasticache_client.delete_replication_group(
            ReplicationGroupId=self.name, RetainPrimaryCluster=False
        )

    def modify_transit_encryption_mode(self, new_transit_encryption_mode: str):
        # generate api call to migrate the cluster to TLS preffered or to TLS required
            self.elasticache_client.modify_replication_group(
                ReplicationGroupId=self.name,
                TransitEncryptionMode=new_transit_encryption_mode,
                TransitEncryptionEnabled=True,
                ApplyImmediately=True,
            )
        self.wait_for_modifying()

class ElastiCacheClusterCME(ElastiCacheClusterBase):
    def __init__(self, name: str):
        super().__init__(name)

    @classmethod
    def launch(cls, config: ConfigCME = None) -> ElastiCacheClusterCME:
        config = config or ConfigCME()
        print(config)
        new_cluster = ElastiCacheClusterCME(config.name)
        launch_request = config.create_launch_request()
```

```
        new_cluster.elasticache_client.create_replication_group(**launch_request)
        new_cluster.wait_for_available()
        return new_cluster

    def get_configuration_endpoint(self) -> str:
        return self.get_first_replication_group()["ConfigurationEndpoint"]["Address"]

#Since the code can throw exceptions, we define this class to make the code more
 readable and
#so we won't forget to delete the cluster
class ElastiCacheCMEManager:
    def __init__(self, config: ConfigCME = None):
        self.config = config or ConfigCME()

    def __enter__(self) -> ElastiCacheClusterCME:
        self.cluster = ElastiCacheClusterCME.launch(self.config)
        return self.cluster

    def __exit__(self, exc_type, exc_val, exc_tb):
        self.cluster.delete_cluster()
```

（可选）创建一个包装器类来演示客户端与 Redis 集群的连接

现在，让我们为 `redis-py-cluster` 客户端创建一个包装器类。这个包装器类将支持在集群中预先填充一些密钥，然后执行随机重复的 `get` 命令。

> **Note**
>
> 这是一个可选步骤，但它简化了后面步骤中出现的主函数的代码。

```
import redis
improt random
from time import perf_counter_ns, time


class DowntimeTestClient:
    def __init__(self, client):
        self.client = client

        # num of keys prefilled
        self.prefilled = 0
```

```
        # percent of get above prefilled
        self.percent_get_above_prefilled = 10 # nil result expected when get hit above
prefilled
        # total downtime in nano seconds
        self.downtime_ns = 0
        # num of success and fail operations
        self.success_ops = 0
        self.fail_ops = 0
        self.connection_errors = 0
        self.timeout_errors = 0


    def replace_client(self, client):
        self.client = client

    def prefill_data(self, timelimit_sec=60):
        end_time = time() + timelimit_sec
        while time() < end_time:
            self.client.set(self.prefilled, self.prefilled)
            self.prefilled += 1

    # unsuccesful operations throw exceptions
    def _exec(self, func):
        try:
            start_ns = perf_counter_ns()
            func()
            self.success_ops += 1
            elapsed_ms = (perf_counter_ns() - start_ns) // 10 ** 6
            # upon succesful execution of func
            # reset random_key to None so that the next command
            # will use a new random key
            self.random_key = None

        except Exception as e:
            elapsed_ns = perf_counter_ns() - start_ns
            self.downtime_ns += elapsed_ns
            # in case of failure- increment the relevant counters so that we will keep
track
            # of how many connection issues we had while trying to communicate with
            # the cluster.
            self.fail_ops += 1
            if e.__class__ is redis.exceptions.ConnectionError:
                self.connection_errors += 1
            if e.__class__ is redis.exceptions.TimeoutError:
```

```
                self.timeout_errors += 1

    def _repeat_exec(self, func, seconds):
        end_time = time() + seconds
        while time() < end_time:
            self._exec(func)

    def _new_random_key_if_needed(self, percent_above_prefilled):
        if self.random_key is None:
            max = int((self.prefilled * (100 + percent_above_prefilled)) / 100)
            return random.randint(0, max)
        return self.random_key

    def _random_get(self):
        key = self._new_random_key_if_needed(self.percent_get_above_prefilled)
        result = self.client.get(key)
        # we know the key was set for sure only in the case key < self.prefilled
        if key < self.prefilled:
            assert result.decode("UTF-8") == str(key)


    def repeat_get(self, seconds=60):
        self._repeat_exec(self._random_get, seconds)

    def get_downtime_ms(self) -> int:
        return self.downtime_ns // 10 ** 6


    def do_get_until(self, cond_check):
        while not cond_check():
            self.repeat_get()
        # do one more get cycle once condition is met
        self.repeat_get()
```

创建主函数，用于演示更改传输中加密配置的过程

现在，让我们定义主函数，它将执行以下操作：

1. 使用 boto3 ElastiCache 客户端创建集群。
2. 初始化将使用不带 TLS 的清晰 TCP 连接来连接到集群的 `redis-py-cluster` 客户端。
3. `redis-py-cluster` 客户端在集群中预先填充一些数据。
4. boto3 客户端将触发 TLS 从无 TLS 迁移到首选 TLS。

5. 当将集群迁移到 TLS Preferred 时，redis-py-cluster TCP 客户端将向集群发送重复的 get 操作，直到迁移完成。
6. 完成向 TLS Preferred 的迁移后，我们将断言集群支持传输中加密。之后，我们将创建一个使用 TLS 连接到集群的 redis-py-cluster 客户端。
7. 我们将使用新的 TLS 客户端和旧的 TCP 客户端发送一些 get 命令。
8. boto3 客户端将触发 TLS 从 TLS Preferred 迁移到需要 TLS。
9. 当将集群迁移到需要 TLS 时，redis-py-cluster TLS 客户端将向集群发送重复的 get 操作，直到迁移完成。

```
import redis

def init_cluster_client(
    cluster: ElastiCacheClusterCME, prefill_data: bool, TLS: bool = True) ->
 DowntimeTestClient:
    # we must use for the host name the cluster configuration endpoint.
    redis_client = redis.RedisCluster(
        host=cluster.get_configuration_endpoint(), ssl=TLS, socket_timeout=0.25,
 socket_connect_timeout=0.1
    )
    test_client = DowntimeTestClient(redis_client)
    if prefill_data:
        test_client.prefill_data()
    return test_client

if __name__ == '__main__':
    config = ConfigCME(TLS=False, instance_type="cache.m5.large")

    with ElastiCacheCMEManager(config) as cluster:
        # create a client that will connect to the cluster with clear tcp connection
        test_client_tcp = init_cluster_client(cluster, prefill_data=True, TLS=False)

       # migrate the cluster to TLS Preferred
        cluster.modify_transit_encryption_mode(new_transit_encryption_mode="preferred")

        # do repeated get commands until the cluster finishes the migration to TLS
 Preferred
        test_client_tcp.do_get_until(cluster.is_available)

      # verify that in transit encryption is enabled so that clients will be able to
 connect to the cluster with TLS
```

```
        assert cluster.get_transit_encryption_enabled() == True

       # create a client that will connect to the cluster with TLS connection.
        # we must first make sure that the cluster indeed supports TLS
        test_client_tls = init_cluster_client(cluster, prefill_data=True, TLS=True)

        # by doing get commands with the tcp client for 60 more seconds
       # we can verify that the existing tcp connection to the cluster still works
        test_client_tcp.repeat_get(seconds=60)

        # do get commands with the new TLS client for 60 more seconds
        test_client_tcp.repeat_get(seconds=60)

       # migrate the cluster to TLS required
        cluster.modify_transit_encryption_mode(new_transit_encryption_mode="required")

       # from this point the tcp clients will be disconnected and we must not use them
anymore.
       # do get commands with the TLS client until the cluster finishes migartion to
TLS required mode.
        test_client_tls.do_get_until(cluster.is_available)
```

启用传输中加密时的最佳实践

在启用传输中加密之前：确保您正确处理 DNS 记录

> **Note**
>
> 在此过程中，我们正在更改和删除旧端点。不正确地使用端点可能会导致 Redis 客户端使用旧的和已删除的端点，从而使其无法连接到集群。

当集群从无 TLS 迁移到首选 TLS 时，旧的每节点 DNS 记录会被保留，并将以不同的格式生成新的每节点 DNS 记录。启用 TLS 的集群与未启用 TLS 的集群所使用的 DNS 记录格式不同。当集群配置为“加密模式：首选”时，ElastiCache 将保留这两种 DNS 记录，以便应用程序和其他 Redis 客户端可以在它们之间切换。在 TLS 迁移过程中，DNS 记录发生以下更改：

启用传输中加密时发生的 DNS 记录更改的描述

对于 CME 集群

当集群设置为“传输加密模式：首选”时：

- 未启用 TLS 的集群的原始集群端点将保持活动状态。将集群从 TLS 加密模式“无”重新配置为“首选”时，不会出现停机。
- 当集群设置为首选 TLS 模式时，将生成新的 TLS Redis 端点。这些新端点将解析为与旧端点（非 TLS）相同的 IP。
- 新的 TLS Redis 配置端点将在 ElastiCache 控制台和对 `describe-replication-group` API 的响应中公开。

当集群设置为“传输加密模式：需要”时：

- 将删除未启用 TLS 的旧端点。TLS 集群端点将不会停机。
- 您可以从 ElastiCache 控制台或 `describe-replication-group` API 检索新的 `cluster-configuration-endpoint` 文件。

对于启用了自动失效转移或禁用了自动失效转移的 CMD 集群

当复制组设置为“传输加密模式：首选”时：

- 未启用 TLS 的集群的原始主端点和读取器端点将保持活动状态。
- 当集群设置为 TLS `Preferred` 模式时，将生成新的 TLS 主端点和读取器端点。此新端点将解析为与旧端点（非 TLS）相同的 IP。
- 新的主端点和读取器端点将在 ElastiCache 控制台和对 `describe-replication-group` API 的响应中公开。

当复制组设置为“传输加密模式：需要”时：

- 未启用 TLS 的集群的原始主端点和读取器端点将保持活动状态。
- 将删除旧的非 TLS 主端点和读取器端点。TLS 集群端点将不会停机。
- 您可以从 ElastiCache 控制台或 `describe-replication-group` API 检索新的主端点和读取器端点。

DNS 记录的建议用法

对于 CME 集群

- 在应用程序代码中使用集群配置端点，而不是每节点 DNS 记录。不建议直接使用每节点 DNS 名称，因为在添加或删除分片时它们可能会发生变化。

- 不要在应用程序中对集群配置端点进行硬编码，因为在此过程中此端点会发生变化。
- 在应用程序中对集群配置端点进行硬编码是一种不好的做法，因为在此过程中可能会对此端点进行更改。传输中加密完成后，使用 `describe-replication-group` API 查询集群配置端点 [如上所示（粗体）]，然后从此时起使用从响应中获得的 DNS。

对于启用了自动失效转移的 CMD 集群

- 在应用程序代码中使用主端点和读取器端点而不是每节点 DNS 名称，因为将集群从无 TLS 迁移到首选 TLS 时，会删除旧的每节点 DNS 名称并生成新的 DNS 名称。不建议直接使用每节点 DNS 名称，因为将来您可能会向集群添加副本。此外，启用自动失效转移后，ElastiCache 服务会自动更改主集群和副本的角色，建议使用主端点和读取器端点来帮助您跟踪这些更改。最后，使用读取器端点将帮助您在集群中的副本之间平均分配从副本进行的读取操作。
- 在应用程序中对主端点和读取器端点进行硬编码是不好的做法，因为在 TLS 迁移过程中可能会对端点进行更改。完成向首选 TLS 的迁移更改后，使用 describe-replication-group API 查询主端点和读取器端点，然后从此时起使用从响应中获得的 DNS。这样，您将能够以动态方式跟踪端点的变化。

对于禁用了自动失效转移的 CMD 集群

- 在应用程序的代码中使用主端点和读取器端点，而不是每节点 DNS 名称。禁用自动失效转移时，启用自动失效转移时由 ElastiCache 服务自动管理的扩缩、修补、失效转移以及其他过程将改为由您完成。这使您能够更轻松地手动跟踪不同的端点。由于在将集群从无 TLS 迁移到首选 TLS 时，会删除旧的每节点 DNS 名称并生成新的每节点 DNS 名称，因此请勿直接使用每节点 DNS 名称。这一点是强制性的，以便客户端能够在 TLS 迁移期间连接到集群。此外，在使用读取器端点时会在副本之间均匀分布读取操作，并且在集群中添加或删除副本时会跟踪 DNS 记录，因此您将受益匪浅。
- 在应用程序中对集群配置端点进行硬编码是一种不好的做法，因为在 TLS 迁移过程中可能会对端点进行更改。

在传输中加密期间：注意迁移过程何时完成

传输加密模式的更改并非立即生效，而可能需要一些时间。对于大型集群而言，这种情况尤其如此。只有当集群完成向首选 TLS 的迁移后，集群才能接受和提供 TCP 和 TLS 连接。因此，在传输中加密完成之前，不应创建尝试与集群建立 TLS 连接的客户端。

有几种方法可以在传输中加密成功完成或失败时获得通知：（未显示在上面的代码示例中）：

- 使用 SNS 服务在加密完成时收到通知
- 使用将在加密完成后发出事件的 `describe-events` API

- 在 ElastiCache 控制台中看到一条表明加密已完成的消息

您还可以在应用程序中实现逻辑以了解加密是否已完成。在上面的示例中，我们看到了几种确保集群完成迁移的方法：

- 等到迁移过程开始（集群状态更改为“正在修改”），然后等到修改完成（集群状态变回为“可用”）
- 通过查询 `describe-replication-group` API 断言集群已将 `transit_encryption_enabled` 设置为 True。

启用传输中加密后：确保正确配置了您使用的客户端

当集群处于首选 TLS 模式时，您的应用程序应建立与集群的 TLS 连接并仅使用这些连接。这样，在启用传输中加密时，应用程序就不会出现停机。您可以使用 SSL 部分下的 Redis info 命令，确保与 Redis 引擎之间没有更清晰的 TCP 连接。

```
# SSL
ssl_enabled:yes
ssl_current_certificate_not_before_date:Mar 20 23:27:07 2017 GMT
ssl_current_certificate_not_after_date:Feb 24 23:27:07 2117 GMT
ssl_current_certificate_serial:D8C7DEA91E684163
tls_mode_connected_tcp_clients:0   (should be zero)
tls_mode_connected_tls_clients:100
```

ElastiCache 中的静态加密

为了帮助保护您的数据，Amazon ElastiCache 和 Amazon S3 提供了不同的方法来限制对缓存中的数据的访问。有关更多信息，请参阅 Amazon VPC 和 ElastiCache 安全性 和 适用于亚马逊的身份和访问管理 ElastiCache：

ElastiCache 静态加密功能可加密磁盘数据，从而提高数据安全性。无服务器缓存上始终启用该功能。在启用后，它会对以下方面进行加密：

- 同步、备份和交换操作期间的磁盘
- 存储在 Amazon S3 中的备份

在启用数据分层的集群中，存储在 SSD（固态硬盘）上的数据始终加密。

ElastiCache 提供默认（服务托管式）的静态加密，以及使用 AWS Key Management Service（KMS）中您自己的对称客户自主管理型 AWS KMS 密钥的功能。备份缓存后，在加密选项下，选择是使用默认加密密钥还是客户自主管理型密钥。有关更多信息，请参阅启用静态加密。

> **Note**
>
> 默认加密（服务托管式）是 GovCloud (US) 区域中唯一可用选项。

> **Important**
>
> 在现有自行设计 Redis 集群上启用静态加密，涉及到在现有的复制组上运行备份和还原之后删除该复制组。

静态加密只能在创建缓存时在缓存上启用。由于加密和解密数据时需要进行一些处理，因此启用静态加密会对这些操作期间的性能产生影响。应对使用和不使用静态加密的数据进行基准测试，以确定对使用案例的性能影响。

主题

静态加密条件

在规划 ElastiCache 静态加密的实现时，应牢记有关 ElastiCache 静态加密的以下约束：

- 在运行 Redis 版本 3.2.6（计划终止生命周期，请参阅 Redis 版本生命周期终止计划）、4.0.10 或更高版本的复制组上支持静态加密。
- 只有在 Amazon VPC 中运行的复制组支持静态加密。
- 只有运行以下节点类型的复制组才支持静态加密。
  - R6gd、R6g、R5、R4、R3
  - M6g、M5、M4、M3
  - T4g、T3、T2

  有关更多信息，请参阅受支持的节点类型
- 通过将参数 `AtRestEncryptionEnabled` 明确设置为 `true` 可启用静态加密。
- 只有在创建复制组时才能在复制组中启用静态加密。无法通过修改复制组来开启和关闭静态加密。有关在现有复制组中实现静态加密的信息，请参阅启用静态加密。
- 如果集群使用 r6gd 系列的节点类型，则无论是否启用静态加密，存储在 SSD 上的数据都会加密。
- 使用客户自主管理型密钥进行静态加密的选项在 AWS GovCloud（us-gov-east-1 和 us-gov-west-1）区域中不可用。
- 如果集群使用 r6gd 系列的节点类型，则存储在 SSD 上的数据将使用所选客户自主管理型 AWS KMS 密钥进行加密（或在 AWS GovCloud 区域中使用服务托管加密）。

在备份和节点同步操作期间，实施静态加密可能会降低性能。使用自己的数据，对静态加密进行基准测试，然后与不加密情况进行比较，以确定其对实现性能的影响。

使用 AWS KMS 中的客户自主管理型密钥

ElastiCache 支持使用对称的客户自主管理型 AWS KMS 密钥（简称 KMS 密钥）进行静态加密。客户自主管理型 KMS 密钥是您在自己的 AWS 账户中创建、拥有并管理的加密密钥。有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的 AWS KMS 密钥。必须先在 AWS KMS 中创建密钥，然后才能将其与 ElastiCache 一起使用。

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

ElastiCache 允许与 AWS KMS 集成。有关更多信息，请参阅 AWS Key Management Service 开发人员指南中的使用授权。无需任何客户操作即可实现 Amazon ElastiCache 与 AWS KMS 的集成。

`kms:ViaService` 条件键将 AWS KMS 密钥（KMS 密钥）限制为仅用于指定 AWS 服务发送的请求。要将 `kms:ViaService` 与 ElastiCache 结合使用，请将两个 ViaService 名称包含在条件键值中：`elasticache.AWS_region.amazonaws.com` 和 `dax.AWS_region.amazonaws.com`。有关更多信息，请参阅 kms:ViaService。

您可以使用 AWS CloudTrail 来跟踪 Amazon ElastiCache 代表您向 AWS Key Management Service 发送的请求。对 AWS Key Management Service 发出的与客户自主管理型密钥相关的所有 API 调用都具有相应的 CloudTrail 日志。您还可以通过调用 ListGrants KMS API 调用来查看 ElastiCache 创建的授权。

使用客户自主管理型密钥对复制组进行加密后，复制组的所有备份都将按如下方式进行加密：

- 使用与集群关联的客户自主管理型密钥对每日自动备份进行加密。
- 删除复制组时创建的最终备份也使用与复制组关联的客户自主管理型密钥进行加密。
- 默认情况下，使用与复制组关联的密钥对手动创建的备份进行加密。您可以通过选择其他客户自主管理型密钥来覆此行为。
- 复制备份将默认使用与源备份关联的客户自主管理型密钥。您可以通过选择其他客户自主管理型密钥来覆此行为。

> **Note**
>
> - 将备份导出到所选的 Amazon S3 存储桶时，无法使用客户自主管理型密钥。但是，导出到 Amazon S3 的所有备份都将使用服务器端加密进行加密。您可以选择将备份文件复制到新的 S3 对象并使用客户自主管理型 KMS 密钥进行加密，将文件复制到使用 KMS 密钥通过默认加密设置的另一个 S3 存储桶，或者更改文件本身中的加密选项。
> - 对于未使用客户自主管理型密钥进行加密的复制组的手动创建备份，您还可以使用客户自主管理型密钥对其进行加密。使用此选项，即使未在原始复制组上加密数据，也可以使用 KMS 密钥对存储在 Amazon S3 中的备份文件进行加密。
>
> 从备份还原允许您从可用的加密选项中进行选择，类似于创建新复制组时可用的加密选项。

- 如果删除密钥或禁用密钥并为用于加密缓存的密钥撤销授权，则缓存将变得不可恢复。换句话说，复制组在硬件故障后无法修改或恢复。AWSKMS 在至少七天的等待期限之后才会删除根密钥。删除密钥后，您可以使用其他客户自主管理型密钥创建备份以用于存档目的。

- 自动密钥轮换将保留 AWS KMS 根密钥的属性，因此轮换不会影响您访问 ElastiCache 数据的能力。加密的 Amazon ElastiCache 缓存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。要了解详情，请参阅 AWS Key Management Service 开发人员指南中的轮换 AWS KMS 密钥。
- 使用 KMS 密钥对 ElastiCache 缓存进行加密时，每个缓存都需要一个授权。此授权在缓存的整个生命周期中使用。此外，在备份创建期间每个备份要使用一个授权。在创建备份后，此授权将停用。
- 有关 AWS KMS 授权和限制的更多信息，请参阅 AWS Key Management Service 开发人员指南中的限制。

启用静态加密

所有无服务器缓存均启用了静态加密。

创建自行设计的集群时，您可以通过将参数 `AtRestEncryptionEnabled` 设置为 `true` 来启用静态加密。不能对现有复制组启用静态加密。

您可以在创建 ElastiCache 缓存时启用静态加密。您可以使用 AWS Management Console、AWS CLI 或 ElastiCache API 执行此操作。

在创建缓存时，您可以选取以下选项之一：

- 默认 – 此选项使用服务管理的静态加密。
- 客户自主管理型密钥 – 此选项允许您提供 AWS KMS 中的密钥 ID/ARN 以进行静态加密。

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

目录

对现有的自行设计 Redis 集群启用静态加密

只能在创建 Redis 复制组时启用静态加密。如果要对现有复制组启用静态加密，请执行以下操作。

要对现有复制组启用静态加密

1. 创建现有复制组的手动备份。有关更多信息，请参阅进行手动备份。

2. 通过从备份中还原来创建新复制组。对新复制组启用静态加密。有关更多信息，请参阅从备份还原到新缓存。
3. 在您的应用程序中，将终端节点更新为新复制组的节点。
4. 删除旧复制组。有关更多信息，请参阅 删除集群或 删除复制组。

使用 AWS Management Console 启用静态加密

在无服务器缓存上启用静态加密（控制台）

所有无服务器缓存均启用了静态加密。默认情况下，使用 AWS 拥有的 KMS 密钥来加密数据。要选择您自己的 AWS KMS 密钥，请进行以下选择：

- 展开默认设置部分。
- 在默认设置部分下选择自定义默认设置。
- 在安全部分下选择自定义您的安全设置。
- 在加密密钥设置下选择客户自主管理型密钥。
- 在 AWS KMS 密钥设置下选择一个密钥。

在自行设计的集群上启用静态加密（控制台）

在设计自己的缓存时，采用“轻松创建”方法的“开发/测试”和“生产”配置均启用了使用默认密钥的静态加密。在您选择自己的配置时，请进行以下选择：

- 选择 3.2.6、4.0.10 或更高版本作为引擎版本。
- 单击静态加密选项的启用旁边的复选框。
- 选择默认密钥或客户自主管理型密钥。

有关这个分步过程，请参阅以下内容：

- 创建 Redis（已禁用集群模式）集群（控制台）
- 创建 Redis（已启用集群模式）集群（控制台）

使用 AWS CLI 启用静态加密

要在创建 Redis 集群时使用 AWS CLI 启用静态加密，请在创建复制组时使用 --at-rest-encryption-enabled 参数。

在 Redis（已禁用集群模式）集群上启用静态加密 (CLI)

以下操作创建具有三个节点（--num-cache-clusters，一个主节点和两个只读副本）的 Redis（已禁用集群模式）复制组 `my-classic-rg`。为此复制组启用了静态加密（--at-rest-encryption-enabled）。

在对此复制组启用加密时，需要以下参数及其值：

关键参数

- **`--engine`** – 必须为 `redis`。
- **`--engine-version`** – 必须是 3.2.6、4.0.10 或更高版本。
- **`--at-rest-encryption-enabled`** – 启用静态加密所必需的。

Example 1：具有副本的 Redis（已禁用集群模式）集群

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
    --replication-group-id my-classic-rg \
    --replication-group-description "3 node replication group" \
    --cache-node-type cache.m4.large \
    --engine redis \
    --at-rest-encryption-enabled \
    --num-cache-clusters 3
```

对于 Windows：

```
aws elasticache create-replication-group ^
    --replication-group-id my-classic-rg ^
    --replication-group-description "3 node replication group" ^
    --cache-node-type cache.m4.large ^
    --engine redis ^
    --at-rest-encryption-enabled ^
    --num-cache-clusters 3 ^
```

有关更多信息，请参阅以下内容：

- 从头开始创建 Redis（已禁用集群模式）复制组（AWS CLI）
- create-replication-group

在 Redis（已启用集群模式）集群上启用静态加密 (CLI)

以下操作创建具有三个节点组/分区 (--num-node-groups) 的 Redis（已启用集群模式）复制组 `my-clustered-rg`。每个复制组有三个节点，一个主节点和两个只读副本 (--replicas-per-node-group)。为此复制组启用了静态加密（--at-rest-encryption-enabled）。

在对此复制组启用加密时，需要以下参数及其值：

关键参数

- **`--engine`** – 必须为 `redis`。
- **`--engine-version`** – 必须是 4.0.10 或更高版本。
- **`--at-rest-encryption-enabled`** – 启用静态加密所必需的。
- **`--cache-parameter-group`** – 必须为 `default-redis4.0.cluster.on` 或派生自此值，以便为此集群启用复制组模式。

Example 2：Redis（已启用集群模式）集群

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
   --replication-group-id my-clustered-rg \
   --replication-group-description "redis clustered cluster" \
   --cache-node-type cache.m3.large \
   --num-node-groups 3 \
   --replicas-per-node-group 2 \
   --engine redis \
   --engine-version 6.2 \
   --at-rest-encryption-enabled \
   --cache-parameter-group default.redis6.x.cluster.on
```

对于 Windows：

```
aws elasticache create-replication-group ^
   --replication-group-id my-clustered-rg ^
   --replication-group-description "redis clustered cluster" ^
   --cache-node-type cache.m3.large ^
   --num-node-groups 3 ^
```

```
    --replicas-per-node-group 2 ^
    --engine redis ^
    --engine-version 6.2 ^
    --at-rest-encryption-enabled ^
    --cache-parameter-group default.redis6.x.cluster.on
```

有关更多信息，请参阅以下内容：

- 从头开始创建 Redis（已启用集群模式）复制组（AWS CLI）
- create-replication-group

另请参阅

- Amazon VPC 和 ElastiCache 安全性
- 适用于亚马逊的身份和访问管理 ElastiCache

身份验证和授权

ElastiCache 支持使用 IAM 和 Redis AUTH 命令对用户进行身份验证，以及使用基于角色的访问控制（RBAC）对用户进行操作授权。

主题

- 基于角色的访问控制（RBAC）
- 使用 Redis AUTH 命令进行身份验证
- 在 ElastiCache Redis 缓存上禁用访问控制

基于角色的访问控制（RBAC）

在 Redis 6.0 及以上版本中，您可以使用称为基于角色的访问控制（RBAC）的功能，而无需使用使用 Redis AUTH 命令进行身份验证中描述的 Redis AUTH 命令进行用户身份验证。此外，只能通过 RBAC 控制对无服务器缓存的访问。

与 Redis AUTH（如果对客户端令牌进行身份验证，则所有经过身份验证的客户端都对缓存具有完全的访问权限）不同的是，RBAC 使您能够通过用户组控制缓存访问权限。这些用户组旨在作为一种管理对缓存的访问权限的方式。

通过 RBAC，您可以使用访问字符串创建用户并为其分配特定权限，如下所述。您可以将用户分配给与特定角色（管理员、人力资源）匹配的用户组，然后将这些用户部署到一个或多个 ElastiCache for

Redis 缓存。这样，您可以在使用相同 Redis 缓存的客户端之间建立安全边界，并阻止客户端彼此访问数据。

RBAC 旨在支持在 Redis 6 中引入 Redis ACL。在将 RBAC 与 ElastiCache for Redis 缓存一起使用时，有一些限制：

- 不能在访问字符串中指定密码。您可以使用 CreateUser 或 ModifyUser 调用设置密码。
- 对于用户权利，您可以将 on 和 off 作为访问字符串的一部分进行传递。如果两者在访问字符串中均未指定，则将为用户分配 off 并且用户没有访问缓存的权限。
- 您不能使用已禁止和重命名的命令。如果您指定了已禁止的命令或重命名命令，则会引发异常。如果要对重命名的命令使用访问控制列表（ACL），请指定命令的原始名称（换句话说，该命令在重命名之前的名称）。
- 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，ElastiCache for Redis 会管理这些密码。因此，您不能使用 reset，因为此命令会移除用户的所有密码。
- Redis 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。ElastiCache for Redis 支持 ACL LIST 命令，但不像 Redis 那样支持哈希密码。通过 ElastiCache for Redis，您可以使用 describe-users 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，describe-users 不会检索用户密码。

  ElastiCache for Redis 支持的其他只读命令包括 ACL WHOAMI、ACL USERS 和 ACL CAT。ElastiCache for Redis 不支持任何其他基于写入的 ACL 命令。
- 以下限制适用：

资源	允许的最大值
每个用户组的用户数	100
用户数	1000
用户组数	100

下面将更详细地介绍如何通过 ElastiCache for Redis 使用 RBAC。

主题

- 使用访问字符串指定权限
- 将 RBAC 应用于 ElastiCache for Redis 的缓存

- 从 Redis AUTH 迁移到 RBAC
- 从 RBAC 迁移到 Redis AUTH
- 为用户自动轮换密码
- 使用 IAM 进行身份验证

使用访问字符串指定权限

要指定对 ElastiCache for Redis 缓存的权限，请使用 AWS CLI 或 AWS Management Console 创建一个访问字符串并将其分配给用户。

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::”键空间开头的键进行读取访问

```
on ~app::* -@all +@read
```

您可以通过列出用户有权访问的命令来进一步优化这些权限：

+*command1* – 用户对命令的访问被限制为 *command1*。

+@category – 用户的访问被限制为某个类别的命令。

有关向用户分配访问字符串的信息，请参阅 使用控制台和 CLI 创建用户和用户组。

如果要将现有工作负载迁移到 ElastiCache，则可以通过调用 `ACL LIST`（不包括用户和任何哈希密码）来检索访问字符串。

对于 Redis 版本 6.2 及更高版本，还支持以下访问字符串语法：

- `&*` – 具有对所有可用频道的访问权限。

对于 Redis 版本 7.0 及更高版本，还支持以下访问字符串语法：

- `|` - 可用于屏蔽子命令（例如 “-config|set”）。
- `%R~<pattern>` - 添加指定的读取密钥模式。此行为与常规密钥模式类似，但仅授予读取与给定模式匹配的密钥的权限。有关更多信息，请参阅密钥权限。
- `%W~<pattern>` - 添加指定的写入密钥模式。此行为与常规密钥模式类似，但仅授予写入与给定模式匹配的密钥的权限。有关更多信息，请参阅密钥权限。
- `%RW~<pattern>` - `~<pattern>` 的别名。
- `(<rule list>)` - 创建一个新的选择器作为匹配规则的依据。选择器在获得用户权限后进行评估，并根据其定义顺序进行评估。如果命令与用户权限或任何选择器匹配，则允许使用。有关更多信息，请参阅 ACL selectors（ACL 选择器）。
- `clearselectors` - 删除附加到用户的所有选择器。

将 RBAC 应用于 ElastiCache for Redis 的缓存

要使用 ElastiCache for Redis RBAC，请执行以下步骤：

1. 创建一个或多个用户。
2. 创建用户组并将用户添加到该组中。
3. 将用户组分配给已启用了传输中加密的缓存。

下方详细地说明了这些步骤。

主题

- 使用控制台和 CLI 创建用户和用户组
- 使用控制台和 CLI 管理用户组
- 将用户组分配给无服务器缓存
- 将用户组分配给复制组

使用控制台和 CLI 创建用户和用户组

RBAC 用户的用户信息为用户 ID、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。用户 ID 对于用户是唯一的，用户名则是传递给引擎的内容。

确保您提供的用户权限符合用户组的预期目的。例如，如果您创建一个名为 `Administrators` 的用户组，则添加到该组的任何用户都应将其访问字符串设置为对密钥和命令具有完全访问权限。对于 `e-commerce` 用户组中的用户，您可以将其访问字符串设置为只读访问。

ElastiCache 使用用户 ID 和用户名 `"default"` 自动配置默认用户并将其添加到所有用户组。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。

要向缓存添加适当的访问控制，请将此默认用户替换为未启用的或使用强密码的新用户。要更改默认用户，请创建一个新用户，其用户名设置为 `default`。然后，您可以用其替换原始默认用户。

以下过程演示了如何用另一个具有已修改访问字符串的 `default` 用户替换原始 `default` 用户。

修改控制台上的默认用户

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 在导航窗格中，选择用户组管理。
3. 对于用户组 ID，选择要修改的 ID。确保选择的是链接，而不是复选框。
4. 选择 Modify(修改)。
5. 在修改窗口中，选择管理，对于默认用户，选择您想要作为默认用户的用户。
6. 选择选择。
7. 选择 Modify(修改)。执行此操作时，原始默认用户与缓存之间的任何现有连接将被终止。

使用 AWS CLI 修改默认用户

1. 使用以下命令，创建用户名为 `default` 的新用户。

   对于 Linux、macOS 或 Unix：

```
aws elasticache create-user \
  --user-id "new-default-user" \
  --user-name "default" \
  --engine "REDIS" \
```

```
 --passwords "a-str0ng-pa))word" \
 --access-string "off +get ~keys*"
```

对于 Windows：

```
aws elasticache create-user ^
 --user-id "new-default-user" ^
 --user-name "default" ^
 --engine "REDIS" ^
 --passwords "a-str0ng-pa))word" ^
 --access-string "off +get ~keys*"
```

2. 创建用户组并添加先前创建的用户。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-user-group \
  --user-group-id "new-group-2" \
  --engine "REDIS" \
  --user-ids "new-default-user"
```

对于 Windows：

```
aws elasticache create-user-group ^
  --user-group-id "new-group-2" ^
  --engine "REDIS" ^
  --user-ids "new-default-user"
```

3. 用新 `default` 用户替换原始 `default` 用户。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-user-group \
    --user-group-id test-group \
    --user-ids-to-add "new-default-user" \
    --user-ids-to-remove "default"
```

对于 Windows：

```
aws elasticache modify-user-group ^
    --user-group-id test-group ^
```

```
    --user-ids-to-add "new-default-user" ^
    --user-ids-to-remove "default"
```

调用此修改操作时，原始默认用户与缓存之间的任何现有连接将被终止。

创建用户时，最多可以设置两个密码。修改密码时，将保持与缓存之间的所有现有连接。

特别是，在使用 ElastiCache for Redis 的 RBAC 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

使用以下过程在控制台上管理用户。

在控制台上管理用户

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 在 Amazon ElastiCache 控制面板上，选择用户管理。以下选项可用：
   - 创建用户 – 创建用户时，输入用户 ID、用户名、身份验证模式和访问字符串。访问字符串设置允许用户使用的密钥和命令的权限级别。

     创建用户时，最多可以设置两个密码。修改密码时，将保持与缓存之间的所有现有连接。
   - 修改用户 – 允许您更新用户的身份验证设置或更改其访问字符串。
   - 删除用户 – 将账户从它所属的任何用户组中移除。

按以下过程使用 AWS CLI 管理用户。

使用 CLI 修改用户

- 使用 `modify-user` 命令更新用户密码或者更改用户的访问权限。

  修改用户后，将更新与该用户关联的用户组以及与该用户组关联的任何缓存。将会保持所有现有连接。示例如下。

  对于 Linux、macOS 或 Unix：

```
aws elasticache modify-user \
  --user-id user-id-1 \
  --access-string "~objects:* ~items:* ~public:*" \
  --no-password-required
```

对于 Windows：

```
aws elasticache modify-user ^
  --user-id user-id-1 ^
  --access-string "~objects:* ~items:* ~public:*" ^
  --no-password-required
```

> **Note**
>
> 建议您不要使用 nopass 选项。如果您按此建议操作，我们建议将用户的权限设置为只读，且仅能访问一组有限的密钥。

使用 CLI 删除用户

- 使用 delete-user 命令删除用户。此账户将被删除并从其所属的任何用户组中移除。以下是示例。

  对于 Linux、macOS 或 Unix：

  ```
  aws elasticache delete-user \
    --user-id user-id-2
  ```

  对于 Windows：

  ```
  aws elasticache delete-user ^
    --user-id user-id-2
  ```

要查看用户列表，请调用 describe-users 操作。

```
aws elasticache describe-users
```

使用控制台和 CLI 管理用户组

您可以创建用户组来组织和控制用户对一个或多个缓存的访问权限，如下所示。

使用控制台通过以下过程管理用户组。

使用控制台管理用户组

1. 登录 AWS Management Console 并打开 Amazon ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 在 Amazon ElastiCache 控制面板上，选择用户组管理。

   以下操作可用于创建新用户组：

   - 创建 – 创建用户组时，您可以添加用户，然后将用户组分配给缓存。例如，您可以为在缓存上具有管理角色的用户创建一个 `Admin` 用户组。

     > **Important**
     > 创建用户组时，需要包括默认用户。

   - Add Users（添加用户）– 向用户组添加用户。
   - Remove Users（移除用户）– 从用户组中移除用户。当用户从用户组中移除时，他们与缓存之间的任何现有连接都将终止。
   - Delete（删除）– 使用此操作删除用户组。请注意，将删除用户组本身，而不是属于该组的用户。

   对于现有用户组，您可执行以下操作：

   - Add User（添加用户）– 将现有用户添加到用户组。
   - Delete Users（删除用户）– 从用户组中移除现有用户。

     > **Note**
     > 用户将从用户组中移除，但不会从系统中删除。

使用 CLI 通过以下过程管理用户组。

使用 CLI 创建新用户组并添加用户

- 使用 `create-user-group` 命令，如下所示：

  对于 Linux、macOS 或 Unix：

```
aws elasticache create-user-group \
  --user-group-id "new-group-1" \
  --engine "REDIS" \
  --user-ids user-id-1, user-id-2
```

  对于 Windows：

```
aws elasticache create-user-group ^
  --user-group-id "new-group-1" ^
  --engine "REDIS" ^
  --user-ids user-id-1, user-id-2
```

使用 CLI 通过添加新用户或移除当前成员来修改用户组

- 使用 `modify-user-group` 命令，如下所示：

  对于 Linux、macOS 或 Unix：

```
aws elasticache modify-user-group --user-group-id new-group-1 \
--user-ids-to-add user-id-3 \
--user-ids-to-remove user-id-2
```

  对于 Windows：

```
aws elasticache modify-user-group --user-group-id new-group-1 ^
--user-ids-to-add userid-3 ^
--user-ids-to-removere user-id-2
```

> **Note**
>
> 此命令将结束属于从用户组中移除的用户的任何打开的连接。

使用 CLI 删除用户组

- 使用 `delete-user-group` 命令，如下所示：将删除用户组本身，而不是属于该组的用户。

  对于 Linux、macOS 或 Unix：

```
aws elasticache delete-user-group /
   --user-group-id
```

  对于 Windows：

```
aws elasticache delete-user-group ^
   --user-group-id
```

要查看用户组列表，您可用调用 describe-user-groups 操作。

```
aws elasticache describe-user-groups \
  --user-group-id test-group
```

将用户组分配给无服务器缓存

创建用户组并添加用户后，实施 RBAC 的最后步骤是将用户组分配给无服务器缓存。

使用控制台将用户组分配给无服务器缓存

要使用 AWS Management Console 将用户组添加到无服务器缓存中，请执行以下操作：

- 对于已禁用集群模式，请参阅 创建 Redis（已禁用集群模式）集群（控制台）
- 对于已启用集群模式，请参阅 创建 Redis（已启用集群模式）集群（控制台）

使用 AWS CLI 将用户组分配给无服务器缓存

以下 AWS CLI 操作使用带值 *my-user-group-id* 的 user-group-id 参数创建无服务器缓存。用已存在的子网组替换子网组 `sng-test`。

关键参数

- **`--engine`** – 必须为 `redis`。
- **`--user-group-id`** – 此值提供用户组的 ID，该用户组由对缓存具有指定访问权限的用户组成。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-serverless-cache \
    --serverless-cache-name "new-serverless-cache" \
    --description "new-serverless-cache" \
    --engine "redis" \
    --user-group-id "new-group-1"
```

对于 Windows：

```
aws elasticache create-serverless-cache ^
    --serverless-cache-name "new-serverless-cache" ^
    --description "new-serverless-cache" ^
    --engine "redis" ^
    --user-group-id "new-group-1"
```

以下 AWS CLI 操作使用带值 *my-user-group-id* 的 user-group-id 参数修改无服务器缓存。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-serverless-cache \
    --serverless-cache-name serverless-cache-1 \
    --user-group-id "new-group-2"
```

对于 Windows：

```
aws elasticache modify-serverless-cache ^
    --serverless-cache-name serverless-cache-1 ^
    --user-group-id "new-group-2"
```

请注意，对缓存所做的任何修改都将异步更新。您可通过查看事件来监控进度。有关更多信息，请参阅查看 ElastiCache 事件。

将用户组分配给复制组

创建用户组并添加用户后，实施 RBAC 的最后步骤是将用户组分配给复制组。

使用控制台将用户组分配给复制组

要使用 AWS Management Console 将用户组添加到复制中，请执行以下操作：

- 对于已禁用集群模式，请参阅 创建 Redis（已禁用集群模式）集群（控制台）

- 对于已启用集群模式，请参阅 创建 Redis（已启用集群模式）集群（控制台）

使用 AWS CLI 将用户组分配给复制组

以下 AWS CLI 操作将创建一个启用了传输中加密（TLS）且具有 user-group-ids 参数（值为 *my-user-group-id*）的复制组。用已存在的子网组替换子网组 `sng-test`。

关键参数

- **`--engine`** – 必须为 `redis`。
- **`--engine-version`** – 必须是 6.0 或更高版本。
- **`--transit-encryption-enabled`** – 必需，用于身份验证和关联用户组。
- **`--user-group-ids`** – 此值提供用户组的 ID，该用户组由对缓存具有指定访问权限的用户组成。
- **`--cache-subnet-group`** – 必需，用于关联用户组。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
    --replication-group-id "new-replication-group" \
    --replication-group-description "new-replication-group" \
    --engine "redis" \
    --cache-node-type cache.m5.large \
    --transit-encryption-enabled \
    --user-group-ids "new-group-1" \
    --cache-subnet-group "cache-subnet-group"
```

对于 Windows：

```
aws elasticache create-replication-group ^
    --replication-group-id "new-replication-group" ^
    --replication-group-description "new-replication-group" ^
    --engine "redis" ^
    --cache-node-type cache.m5.large ^
    --transit-encryption-enabled ^
    --user-group-ids "new-group-1" ^
    --cache-subnet-group "cache-subnet-group"
```

以下 AWS CLI 操作将修改启用了传输中加密（TLS）且具有 user-group-ids 参数（值为 *my-user-group-id*）的复制组。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
    --replication-group-id replication-group-1 \
    --user-group-ids-to-remove "new-group-1" \
    --user-group-ids-to-add "new-group-2"
```

对于 Windows：

```
aws elasticache modify-replication-group ^
    --replication-group-id replication-group-1 ^
    --user-group-ids-to-remove "new-group-1" ^
    --user-group-ids-to-add "new-group-2"
```

记下响应中的 PendingChanges。对缓存所做的任何修改都将异步更新。您可通过查看事件来监控进度。有关更多信息，请参阅查看 ElastiCache 事件。

从 Redis AUTH 迁移到 RBAC

如果您正在使用 使用 Redis AUTH 命令进行身份验证 中所述的 Redis AUTH，并希望迁移到使用 RBAC，请使用以下过程。

使用控制台通过以下过程从 Redis AUTH 迁移到 RBAC。

使用控制台从 Redis AUTH 迁移到 RBAC

1. 登录 AWS Management Console 并打开 ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 从右上角的列表中，选择要修改的缓存所在的 AWS 区域。
3. 在导航窗格中，选择在要修改的缓存上运行的引擎。

   此时会显示选定引擎的缓存列表。
4. 在缓存列表中，对于要修改的缓存，选择其名称。
5. 对于 Actions（操作），选择 Modify（修改）。

   此时将显示修改窗口。
6. 对于访问控制，选择用户组访问控制列表。
7. 对于用户组访问控制列表，选择一个用户组。
8. 选择预览更改，然后在下一个屏幕上选择修改。

使用 CLI 通过以下过程从 Redis AUTH 迁移到 RBAC。

使用 CLI 从 Redis AUTH 迁移到 RBAC

- 使用 `modify-replication-group` 命令，如下所示：

  对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group --replication-group-id test \
    --auth-token-update-strategy DELETE \
    --user-group-ids-to-add user-group-1
```

  对于 Windows：

```
aws elasticache modify-replication-group --replication-group-id test ^
    --auth-token-update-strategy DELETE ^
    --user-group-ids-to-add user-group-1
```

从 RBAC 迁移到 Redis AUTH

如果您正在使用 RBAC 并希望迁移到 Redis AUTH，请参阅 从 RBAC 迁移到 Redis AUTH。

> **Note**
>
> 如果您需要在 ElastiCache 缓存上禁用访问控制，则需要通过 AWS CLI 来实现。有关更多信息，请参阅the section called “在 ElastiCache Redis 缓存上禁用访问控制”。

为用户自动轮换密码

借助 AWS Secrets Manager，您可以将代码中的硬编码凭证（包括密码）替换为对 Secrets Manager 的 API 调用，从而以编程方式检索密钥。这有助于确保检查您的代码的人不会泄露密钥，因为其中根本不包含密钥。此外，您还可以配置 Secrets Manager 以根据指定的计划自动轮换密钥。这使您能够将长期密钥替换为短期密钥，这有助于显著减少泄露风险。

借助 Secrets Manager，您可以使用 Secrets Manager 提供的 AWS Lambda 函数自动轮换 ElastiCache for Redis 的密码（即密钥）。

有关 AWS Secrets Manager 的更多信息，请参阅什么是 AWS Secrets Manager？

ElastiCache 如何使用密钥

在 Redis 6 中，ElastiCache for Redis 引入 基于角色的访问控制（RBAC） 来保护 Redis 集群。此功能允许在可以执行的命令和可以访问的密钥方面限制某些连接。使用 RBAC，当客户使用密码创建用户时，密码值需要以明文形式手动输入，且对操作员可见。

使用 Secrets Manager，应用程序从 Secrets Manager 获取密码，而不是手动输入密码并将其存储在应用程序的配置中。有关如何执行此操作的信息，请参阅 ElastiCache 是如何与密钥关联的。

使用密钥会产生费用。有关定价信息，请参阅 AWS Secrets Manager 定价。

ElastiCache 是如何与密钥关联的

Secrets Manager 将在密钥的 `SecretString` 字段中为关联用户保留引用。ElastiCache 方面不会提及这个密钥。

```
{
    "password": "strongpassword",
    "username": "user1",
    "user_arn": "arn:aws:elasticache:us-east-1:xxxxxxxxxx918:user:user1" //this is the
 bond between the secret and the user
}
```

Lambda 轮换函数

要启用 Secrets Manager 自动密码轮换，您需要创建一个 Lambda 函数，该函数将与 modify-user API 交互以更新用户的密码。

有关其工作原理的信息，请参阅轮换的工作原理。

> **Note**
>
> 对于某些 AWS 服务，为了避免混淆代理情况，AWS 建议您同时使用 `aws:SourceArn` 和 `aws:SourceAccount` 全局条件密钥。但如果轮换函数策略中包括 `aws:SourceArn` 条件，则轮换函数只能用于轮换该 ARN 指定的密钥。我们建议您仅在其中包括上下文键 `aws:SourceAccount`，以便对多个密钥使用轮换函数。

有关您可能遇到的任何问题，请参阅 AWS Secrets Manager 轮换疑难解答。

如何创建 ElastiCache 用户并将其与 Secrets Manager 关联

以下步骤说明如何创建用户并将其与 Secrets Manager 关联：

1. 创建不活跃的用户

   对于 Linux、macOS 或 Unix：

```
aws elasticache create-user \
 --user-id user1 \
 --user-name user1 \
 --engine "REDIS" \
 --no-password \ // no authentication is required
 --access-string "*off* +get ~keys*" // this disables the user
```

   对于 Windows：

```
aws elasticache create-user ^
 --user-id user1 ^
 --user-name user1 ^
 --engine "REDIS" ^
 --no-password ^ // no authentication is required
 --access-string "*off* +get ~keys*" // this disables the user
```

   您看到的响应与以下内容类似：

```
{
    "UserId": "user1",
    "UserName": "user1",
    "Status": "active",
    "Engine": "redis",
    "AccessString": "off ~keys* -@all +get",
    "UserGroupIds": [],
    "Authentication": {
        "Type": "no_password"
    },
    "ARN": "arn:aws:elasticache:us-east-1:xxxxxxxxxx918:user:user1"
}
```

2. 创建密钥

   对于 Linux、macOS 或 Unix：

```
aws secretsmanager create-secret \
--name production/ec/user1 \
--secret-string \
'{
   "user_arn": "arn:aws:elasticache:us-east-1:123456xxxx:user:user1",
    "username":"user1"
 }'
```

对于 Windows：

```
aws secretsmanager create-secret ^
--name production/ec/user1 ^
--secret-string ^
'{
   "user_arn": "arn:aws:elasticache:us-east-1:123456xxxx:user:user1",
    "username":"user1"
 }'
```

您看到的响应与以下内容类似：

```
{
 "ARN": "arn:aws:secretsmanager:us-east-1:123456xxxx:secret:production/ec/user1-
eaFois",
 "Name": "production/ec/user1",
 "VersionId": "aae5b963-1e6b-4250-91c6-ebd6c47d0d95"
}
```

3. 配置 Lambda 函数来轮换您的密码

   a. 登录到 AWS Management Console，然后通过以下网址打开 Lambda 控制台：https://console.aws.amazon.com/lambda/

   b. 在导航面板上，选择 Functions（函数），然后选择您创建的函数。选择函数名称，而不是其左边的复选框。

   c. 选择配置选项卡。

   d. 在 General configuration（常规配置）中，选择 Edit（编辑），然后将 Timeout（超时）设置为至少 12 分钟。

   e. 选择 Save（保存）。

   f. 选择 Environment variables（环境变量），然后设置以下内容：

i. SECRETS_MANAGER_ENDPOINT – https://secretsmanager.**REGION**.amazonaws.com

ii. SECRET_ARN – 您在第 2 步中创建的密钥的 Amazon 资源名称（ARN）。

iii. USER_NAME – ElastiCache 用户的用户名，

iv. 选择 Save（保存）。

g. 选择 Permissions（权限）。

h. 在 Execution role（执行角色）下，选择要在 IAM 控制台上查看的 Lambda 函数角色的名称。

i. Lambda 函数需要以下权限才能修改用户和设置密码：

ElastiCache

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Action": [
            "elasticache:DescribeUsers",
            "elasticache:ModifyUser"
        ],
        "Resource": "arn:aws:elasticache:us-east-1:xxxxxxxxxx918:user:user1"
        }
    ]
}
```

Secrets Manager

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "secretsmanager:GetSecretValue",
                "secretsmanager:DescribeSecret",
                "secretsmanager:PutSecretValue",
                "secretsmanager:UpdateSecretVersionStage"
            ],
```

```
            "Resource": "arn:aws:secretsmanager:us-
east-1:xxxxxxxxxxx:secret:XXXX"
        },
        {
            "Effect": "Allow",
            "Action": "secretsmanager:GetRandomPassword",
            "Resource": "*"
        }
    ]
}
```

4. 设置 Secrets Manager 密钥轮换

   a. 使用 AWS Management Console，请参阅使用控制台设置 AWS Secrets Manager 密钥的自动轮换

      有关设置轮换计划的更多信息，请参阅 Secrets Manager 轮换中的计划表达式。

   b. 使用 AWS CLI，请参阅使用AWS Command Line Interface 为 AWS Secrets Manager 设置自动轮换

使用 IAM 进行身份验证

主题

- 概述
- 限制
- 设置
- 连接

概述

使用 IAM 身份验证，您可以在缓存配置为使用 Redis 版本 7 或更高版本时，使用 AWS IAM 身份对与 ElastiCache for Redis 的连接进行身份验证。这使您可以增强安全模型并简化许多管理安全任务。您还可以使用 IAM 身份验证，遵循最低权限原则，为每个单独的 ElastiCache 缓存和 ElastiCache 用户配置精细的访问控制。ElastiCache for Redis 的 IAM 身份验证的工作原理是在 Redis AUTH 或 HELLO 命令中提供有效期很短的 IAM 身份验证令牌，而不是有效期很长的 ElastiCache 用户密码。有关 IAM 身份验证令牌的更多信息，请参阅《AWS 一般参考指南》中的 Signature Version 4 签名流程和下面的代码示例。

您可以使用 IAM 身份及其关联策略进一步限制 Redis 访问权限。您还可以直接从联合身份提供商向用户授予对 Redis 缓存的访问权限。

要将 AWS IAM 与 ElastiCache for Redis 一起使用，您首先需要创建身份验证模式设置为 IAM 的 ElastiCache 用户，然后创建或重用 IAM 身份。IAM 身份需要关联策略来向 ElastiCache 缓存和 ElastiCache 用户授予 `elasticache:Connect` 操作权限。配置完成后，您可以使用 IAM 用户或角色的 AWS 凭证创建 IAM 身份验证令牌。最后，在连接到 Redis 缓存时，您需要在 Redis 客户端中提供有效期较短的 IAM 身份验证令牌作为密码。支持凭证提供程序的 Redis 客户端可以为每个新连接自动生成临时凭证。ElastiCache for Redis 将对启用 IAM 的 ElastiCache 用户的连接请求执行 IAM 身份验证，并将通过 IAM 验证连接请求。

限制

使用 IAM 身份验证时，以下限制适用：

- 使用 ElastiCache for Redis 版本 7.0 或更高版本时，IAM 身份验证可用。
- 对于启用了 IAM 的 ElastiCache 用户，用户名和用户 ID 属性必须相同。
- IAM 身份验证令牌的有效期为 15 分钟。对于长时间的连接，建议使用支持凭证提供程序接口的 Redis 客户端。
- 经过 IAM 身份验证的 ElastiCache for Redis 连接将在 12 小时后自动断开。通过使用新 IAM 身份验证令牌发送 `AUTH` 或 `HELLO` 命令，可以将连接延长 12 小时。
- `MULTI EXEC` 命令不支持 IAM 身份验证。
- 目前，IAM 身份验证支持以下全局条件上下文键：
  - 对无服务器缓存使用 IAM 身份验证时，支持 `aws:VpcSourceIp`、`aws:SourceVpc`、`aws:SourceVpce`、`aws:CurrentTime`、`aws:EpochTime` 和 `aws:ResourceTag/%s`（从关联的无服务器缓存和用户）。
  - 对复制组使用 IAM 身份验证时，支持 `aws:SourceIp` 和 `aws:ResourceTag/%s`（从关联的复制组和用户）。

  有关全局条件上下文键的更多信息，请参阅《IAM 用户指南》中的 AWS 全局条件上下文键。

设置

要设置 IAM 身份验证，请执行以下操作：

1. 创建缓存

```
aws elasticache create-serverless-cache \
```

```
  --serverless-cache-name cache-01  \
  --description "ElastiCache IAM auth application" \
  --engine redis
```

2. 为您的角色创建 IAM 信任政策文档，如下所示，允许您的账户承担新角色。将策略保存到名为 trust-policy.json 的文件中。

```
{
    "Version": "2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
        "Action": "sts:AssumeRole"
    }
}
```

3. 创建 IAM policy 文档，如下所示。将策略保存到名为 policy.json 的文件中。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect" : "Allow",
      "Action" : [
        "elasticache:Connect"
      ],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:serverlesscache:cache-01",
        "arn:aws:elasticache:us-east-1:123456789012:user:iam-user-01"
      ]
    }
  ]
}
```

4. 创建一个 IAM 角色。

```
aws iam create-role \
--role-name "elasticache-iam-auth-app" \
--assume-role-policy-document file://trust-policy.json
```

5. 创建 IAM policy。

```
aws iam create-policy \
```

```
  --policy-name "elasticache-allow-all" \
  --policy-document file://policy.json
```

6. 向角色附加 IAM policy。

```
aws iam attach-role-policy \
 --role-name "elasticache-iam-auth-app" \
 --policy-arn "arn:aws:iam::123456789012:policy/elasticache-allow-all"
```

7. 创建启用 IAM 的新用户。

```
aws elasticache create-user \
  --user-name iam-user-01 \
  --user-id iam-user-01 \
  --authentication-mode Type=iam \
  --engine redis \
  --access-string "on ~* +@all"
```

8. 创建用户组并附加用户。

```
aws elasticache create-user-group \
  --user-group-id iam-user-group-01 \
  --engine redis \
  --user-ids default iam-user-01

aws elasticache modify-serverless-cache \
  --serverless-cache-name cache-01  \
  --user-group-id iam-user-group-01
```

连接

使用令牌作为密码进行连接

您首先需要使用 AWS SigV4 预签名请求生成有效期较短的 IAM 身份验证令牌。之后，您需要在连接到 Redis 缓存时提供 IAM 身份验证令牌作为密码，如下例所示。

```
String userId = "insert user id";
String cacheName = "insert cache name";
boolean isServerless = true;
String region = "insert region";

// Create a default AWS Credentials provider.
```

```
// This will look for AWS credentials defined in environment variables or system
 properties.
AWSCredentialsProvider awsCredentialsProvider = new
 DefaultAWSCredentialsProviderChain();

// Create an IAM authentication token request and signed it using the AWS credentials.
// The pre-signed request URL is used as an IAM authentication token for ElastiCache
 Redis.
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userId, cacheName,
 region, isServerless);
String iamAuthToken =
 iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());

// Construct Redis URL with IAM Auth credentials provider
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
    .withSsl(ssl)
    .withAuthentication(userId, iamAuthToken)
    .build();

// Create a new Lettuce Redis client
RedisClient client = RedisClient.create(redisURI);
client.connect();
```

以下为 IAMAuthTokenRequest 的定义。

```
public class IAMAuthTokenRequest {
    private static final HttpMethodName REQUEST_METHOD = HttpMethodName.GET;
    private static final String REQUEST_PROTOCOL = "http://";
    private static final String PARAM_ACTION = "Action";
    private static final String PARAM_USER = "User";
    private static final String PARAM_RESOURCE_TYPE = "ResourceType";
    private static final String RESOURCE_TYPE_SERVERLESS_CACHE = "ServerlessCache";
    private static final String ACTION_NAME = "connect";
    private static final String SERVICE_NAME = "elasticache";
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final String userId;
    private final String cacheName;
    private final String region;
    private final boolean isServerless;
```

```
    public IAMAuthTokenRequest(String userId, String cacheName, String region, boolean
isServerless) {
        this.userId = userId;
        this.cacheName = cacheName;
        this.region = region;
        this.isServerless = isServerless;
    }

    public String toSignedRequestUri(AWSCredentials credentials) throws
URISyntaxException {
        Request<Void> request = getSignableRequest();
        sign(request, credentials);
        return new URIBuilder(request.getEndpoint())
            .addParameters(toNamedValuePair(request.getParameters()))
            .build()
            .toString()
            .replace(REQUEST_PROTOCOL, "");
    }

    private <T> Request<T> getSignableRequest() {
        Request<T> request  = new DefaultRequest<>(SERVICE_NAME);
        request.setHttpMethod(REQUEST_METHOD);
        request.setEndpoint(getRequestUri());
        request.addParameters(PARAM_ACTION, Collections.singletonList(ACTION_NAME));
        request.addParameters(PARAM_USER, Collections.singletonList(userId));
        if (isServerless) {
            request.addParameters(PARAM_RESOURCE_TYPE,
Collections.singletonList(RESOURCE_TYPE_SERVERLESS_CACHE));
        }
        return request;
    }

    private URI getRequestUri() {
        return URI.create(String.format("%s%s/", REQUEST_PROTOCOL, cacheName));
    }

    private <T> void sign(SignableRequest<T> request, AWSCredentials credentials) {
        AWS4Signer signer = new AWS4Signer();
        signer.setRegionName(region);
        signer.setServiceName(SERVICE_NAME);

        DateTime dateTime = DateTime.now();
        dateTime = dateTime.plus(Duration.standardSeconds(TOKEN_EXPIRY_SECONDS));
```

```
        signer.presignRequest(request, credentials, dateTime.toDate());
    }

    private static List<NameValuePair> toNamedValuePair(Map<String, List<String>> in) {
        return in.entrySet().stream()
            .map(e -> new BasicNameValuePair(e.getKey(), e.getValue().get(0)))
            .collect(Collectors.toList());
    }
}
```

使用凭证提供程序进行连接

以下代码显示了如何使用 IAM 身份验证凭证提供程序通过 ElastiCache for Redis 进行身份验证。

```
String userId = "insert user id";
String cacheName = "insert cache name";
boolean isServerless = true;
String region = "insert region";

// Create a default AWS Credentials provider.
// This will look for AWS credentials defined in environment variables or system
 properties.
AWSCredentialsProvider awsCredentialsProvider = new
 DefaultAWSCredentialsProviderChain();

// Create an IAM authentication token request. Once this request is signed it can be
 used as an
// IAM authentication token for ElastiCache Redis.
IAMAuthTokenRequest iamAuthTokenRequest = new IAMAuthTokenRequest(userId, cacheName,
 region, isServerless);

// Create a Redis credentials provider using IAM credentials.
RedisCredentialsProvider redisCredentialsProvider = new
 RedisIAMAuthCredentialsProvider(
    userId, iamAuthTokenRequest, awsCredentialsProvider);

// Construct Redis URL with IAM Auth credentials provider
RedisURI redisURI = RedisURI.builder()
    .withHost(host)
    .withPort(port)
    .withSsl(ssl)
    .withAuthentication(redisCredentialsProvider)
    .build();
```

```
// Create a new Lettuce Redis client
RedisClient client = RedisClient.create(redisURI);
client.connect();
```

以下是 Lettuce Redis 客户端的示例，该客户端将 IAMAuthTokenRequest 封装在凭证提供程序中，以便在需要时自动生成临时凭证。

```
public class RedisIAMAuthCredentialsProvider implements RedisCredentialsProvider {
    private static final long TOKEN_EXPIRY_SECONDS = 900;

    private final AWSCredentialsProvider awsCredentialsProvider;
    private final String userId;
    private final IAMAuthTokenRequest iamAuthTokenRequest;
    private final Supplier<String> iamAuthTokenSupplier;

    public RedisIAMAuthCredentialsProvider(String userId,
        IAMAuthTokenRequest iamAuthTokenRequest,
        AWSCredentialsProvider awsCredentialsProvider) {
        this.userName = userName;
        this.awsCredentialsProvider = awsCredentialsProvider;
        this.iamAuthTokenRequest = iamAuthTokenRequest;
        this.iamAuthTokenSupplier =
 Suppliers.memoizeWithExpiration(this::getIamAuthToken, TOKEN_EXPIRY_SECONDS,
 TimeUnit.SECONDS);
    }

    @Override
    public Mono<RedisCredentials> resolveCredentials() {
        return Mono.just(RedisCredentials.just(userId, iamAuthTokenSupplier.get()));
    }

    private String getIamAuthToken() {
        return
 iamAuthTokenRequest.toSignedRequestUri(awsCredentialsProvider.getCredentials());
    }
}
```

使用 Redis AUTH 命令进行身份验证

> **Note**
>
> Redis AUTH 已被取代。the section called “基于角色的访问控制（RBAC）”所有无服务器缓存都必须使用 RBAC 进行身份验证。

Redis 身份验证令牌或密码使 Redis 能够在允许客户端运行命令之前要求输入密码，从而提高数据安全性。Redis AUTH 仅适用于自行设计的集群。

主题

- ElastiCache 适用于 Redis 的 AUTH 概述
- 对适用 ElastiCache 于 Redis 的集群应用身份验证
- 在现有 ElastiCache 的 Redis 集群上修改身份验证令牌
- 从 RBAC 迁移到 Redis AUTH

ElastiCache 适用于 Redis 的 AUTH 概述

当你将 Redis AUTH 与 for Redis 集群配合使用时， ElastiCache 需要进行一些改进。

特别是，在将 AUTH 与 Redis 一起 ElastiCache 使用时，请注意以下身份验证令牌或密码限制：

- 令牌（或密码）必须是 16-128 个可打印字符。
- 非字母数字字符仅限使用 !、&、#、$、^、<、>、-。
- 只能为为 Redis 集群启用传输中加密功能启用身份验证 ElastiCache 。

要设置增强令牌，我们建议您遵循严格的密码策略，例如，要求满足以下条件：

- 令牌或密码必须至少包含以下三种字符类型：
  - 大写字符
  - 小写字符
  - 数字
  - 非字母数字字符（!、&、#、$、^、<、>、-）
- 令牌或密码不得包含字典单词或稍作修改的字典单词。
- 令牌或密码不得与最近使用的令牌相同或相似。

对适用 ElastiCache 于 Redis 的集群应用身份验证

您可以要求用户在受令牌保护的 Redis 服务器上输入令牌（密码）。为此，在创建复制组或集群时，请在 `--auth-token` 参数（API：`AuthToken`）中包含正确的令牌。还要在复制组或集群的所有后续命令中包含该令牌。

以下 AWS CLI 操作创建启用了传输中加密 (TLS) 和AUTH令牌的复制组*`This-is-a-sample-token`*。用已存在的子网组替换子网组 `sng-test`。

关键参数

- **`--engine`** – 必须为 `redis`。
- **`--engine-version`** – 必须是 3.2.6、4.0.10 或更高版本。
- **`--transit-encryption-enabled`** – 对于身份验证和 HIPAA 资格是必需的。
- **`--auth-token`** – 对于 HIPAA 资格是必需的。该值必须是该受令牌保护的 Redis 服务器的正确令牌。
- **`--cache-subnet-group`** – 对于 HIPAA 资格是必需的。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
    --replication-group-id authtestgroup \
    --replication-group-description authtest \
    --engine redis \
    --cache-node-type cache.m4.large \
    --num-node-groups 1 \
    --replicas-per-node-group 2 \
    --transit-encryption-enabled \
    --auth-token This-is-a-sample-token \
    --cache-subnet-group sng-test
```

对于 Windows：

```
aws elasticache create-replication-group ^
    --replication-group-id authtestgroup ^
    --replication-group-description authtest ^
    --engine redis ^
    --cache-node-type cache.m4.large ^
    --num-node-groups 1 ^
    --replicas-per-node-group 2 ^
```

```
    --transit-encryption-enabled ^
    --auth-token This-is-a-sample-token ^
    --cache-subnet-group sng-test
```

在现有 ElastiCache 的 Redis 集群上修改身份验证令牌

为了更轻松地更新您的身份验证，您可以修改 for Redis 集群上使用的AUTH令牌。 ElastiCache 如果引擎版本为 5.0.6 或更高版本，且 Redis 启用了传输中加密，则可以 ElastiCache 进行此修改。

修改 AUTH 令牌支持两种策略：ROTATE 和 SET。ROTATE 策略在保留先前令牌的同时，向服务器添加一个额外的身份验证令牌。SET 策略会更新服务器，使其仅支持单个 AUTH 令牌。请使用 `--apply-immediately` 参数进行这些修改调用以立即应用更改。

轮换 AUTH 令牌

要使用新的身份验证令牌更新 Redis 服务器，请调用 `ModifyReplicationGroup` API，`--auth-token`参数为新AUTH令牌，值为 ROTAT `--auth-token-update-strategy` E。ROTATE 修改完成后，除了`auth-token`参数中指定的身份验证令牌外，集群还将支持之前的身份验证令牌。如果在 AUTH 令牌轮换之前未在复制组上配置身份验证令牌，则集群除了支持无需身份验证即可连接外，还支持`--auth-token`参数中指定的身份验证令牌。设置 AUTH 令牌要使用更新策略集更新所需的身份验证令牌，请参阅。

> **Note**
>
> 如果您之前未配置 AUTH 令牌，则修改完成后，除在 auth-token 参数中指定的一个令牌外，集群将不支持任何 AUTH 令牌。

如果在已经支持两个 AUTH 令牌的服务器上执行此修改，则在此操作期间，最旧的身份验证令牌也将被移除。这允许服务器在给定时间最多支持两个最新的身份验证令牌。

此时，您可以通过更新客户端以使用最新的 AUTH 令牌来继续。在更新客户端后，您可以使用 SET 策略轮换 AUTH 令牌（在下一节中说明）以开始使用唯一的新令牌。

以下 AWS CLI 操作修改复制组以轮换令AUTH牌*This-is-the-rotated-token*。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
--replication-group-id authtestgroup \
--auth-token This-is-the-rotated-token \
--auth-token-update-strategy ROTATE \
```

```
--apply-immediately
```

对于 Windows：

```
aws elasticache modify-replication-group ^
--replication-group-id authtestgroup ^
--auth-token This-is-the-rotated-token ^
--auth-token-update-strategy ROTATE ^
--apply-immediately
```

设置 AUTH 令牌

要更新 Redis 服务器以支持单个必需AUTH令牌，请使用与最后一个 AUTH 令牌值相同的`--auth-token`参数和具有该值的`--auth-token-update-strategy`参数调用 `ModifyReplicationGroup` API 操作。SETSET 策略只能用于之前使用 ROTATE 策略的 2 个身份验证令牌或 1 个可选身份验证令牌的集群。修改完成后，Redis 服务器仅支持 auth-token 参数中指定的身份验证令牌。

以下 AWS CLI 操作修改了要将 AUTH 令牌设置为的复制组。*This-is-the-set-token*

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
--replication-group-id authtestgroup \
--auth-token This-is-the-set-token \
--auth-token-update-strategy SET \
--apply-immediately
```

对于 Windows：

```
aws elasticache modify-replication-group ^
--replication-group-id authtestgroup ^
--auth-token This-is-the-set-token ^
--auth-token-update-strategy SET ^
--apply-immediately
```

在现有的 Redis 集 ElastiCache 群上启用身份验证

要在现有 Redis 服务器上启用身份验证，请调用 `ModifyReplicationGroup` API 操作。调用 `ModifyReplicationGroup`，将 `--auth-token` 参数作为新令牌，并将 `--auth-token-update-strategy` 参数值设置为 ROTATE。

ROTATE 修改完成后，集群除了支持无需身份验证即可连接外，还支持--auth-token参数中指定的AUTH令牌。更新所有客户端应用程序以使用身份验证令牌向 Redis 进行身份验证后，请使用 SET 策略将身份验证令牌标记为必填项。仅在启用了传输中加密（TLS）的 Redis 服务器上支持启用身份验证。

从 RBAC 迁移到 Redis AUTH

如果您正在使用中所述的 Redis 基于角色的访问控制 (RBAC) 对用户进行身份验证基于角色的访问控制（RBAC），并且想要迁移到 Redis AUTH，请使用以下步骤。您可以使用控制台或 CLI 进行迁移。

使用控制台从 RBAC 迁移到 Redis AUTH

1. 登录 AWS Management Console 并打开 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 从右上角的列表中，选择要修改的集群所在的 AWS 区域。
3. 在导航窗格中，选择在您要修改的集群上运行的引擎。

   此时会显示选定引擎的集群列表。
4. 在集群列表中，对于要修改的集群，选择其名称。
5. 对于 Actions（操作），选择 Modify（修改）。

   此时将显示修改窗口。
6. 对于访问控制，请选择 Redis AUTH 默认用户访问。
7. 在 Redis AUTH 令牌下，设置一个新令牌。
8. 选择预览更改，然后在下一个屏幕上选择修改。

要从 RBAC 迁移到 Redis AUTH，请使用 AWS CLI

使用以下命令之一为您的 Redis 复制组配置新的可选AUTH令牌。请注意，在身份验证令牌被标记为必填项之前，可选的身份验证令牌将允许对复制组进行未经身份验证的访问，使用下一步SET中的更新策略。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
    --replication-group-id test \
    --remove-user-groups \
    --auth-token This-is-a-sample-token \
```

```
    --auth-token-update-strategy ROTATE \
    --apply-immediately
```

对于 Windows：

```
aws elasticache modify-replication-group ^
    --replication-group-id test ^
    --remove-user-groups ^
    --auth-token This-is-a-sample-token ^
    --auth-token-update-strategy ROTATE ^
    --apply-immediately
```

执行上述命令后，您可以使用新配置的可选 AUTH 令牌更新您的 Redis 应用程序，以向 ElastiCache 复制组进行身份验证。要完成身份验证令牌的轮换，请在下面的后续命令SET中使用更新策略。这将根据需要标记到可选的身份验证令牌上。身份验证令牌更新完成后，复制组的状态将显示为，ACTIVE并且与该复制组的所有 Redis 连接都需要身份验证。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
   --replication-group-id test \
   --auth-token This-is-a-sample-token \
   --auth-token-update-strategy SET \
   --apply-immediately
```

对于 Windows：

```
aws elasticache modify-replication-group ^
   --replication-group-id test ^
   --remove-user-groups ^
   --auth-token This-is-a-sample-token ^
   --auth-token-update-strategy SET ^
   --apply-immediately
```

有关更多信息，请参阅 使用 Redis AUTH 命令进行身份验证。

> **Note**
>
> 如果您需要在 ElastiCache 集群上禁用访问控制，请参阅the section called “在 ElastiCache Redis 缓存上禁用访问控制”。

在 ElastiCache Redis 缓存上禁用访问控制

按照以下说明操作，在启用 Redis TLS 的缓存上禁用访问控制。您的 Redis 缓存将采用两种不同类型的配置之一：Redis AUTH 默认用户访问权限或用户组访问控制（RBAC）列表。如果您的缓存在创建时使用 AUTH 配置，则必须先将其更改为 RBAC 配置，然后才能通过删除用户组来禁用缓存访问控制。如果您的缓存在创建时使用了 RBAC 配置，则可以直接将其禁用。

禁用使用 RBAC 配置的 Redis 无服务器缓存

1. 删除用户组以禁用访问控制。

```
aws elasticache modify-serverless-cache --serverless-cache-name <serverless-cache>
 --remove-user-group
```

2. （可选）验证没有用户组与无服务器缓存关联。

```
aws elasticache describe-serverless-caches --serverless-cache-name <serverless-
cache>
{
    "..."
    "UserGroupId": ""
    "..."
}
```

禁用通过 AUTH 令牌配置的 Redis 缓存

1. 将 AUTH 令牌更改为 RBAC 并指定要添加的用户组。

```
aws elasticache modify-replication-group --replication-group-id <replication-group-
id-value> --auth-token-update-strategy DELETE --user-group-ids-to-add <user-group-
value>
```

2. 验证 AUTH 令牌是否已禁用以及是否添加了用户组。

```
aws elasticache describe-replication-groups --replication-group-id <replication-
group-id-value>
{
    "..."
    "AuthTokenEnabled": false,
    "UserGroupIds": [
        "<user-group-value>"
```

```
    ]
    "..."
}
```

3. 删除用户组以禁用访问控制。

```
aws elasticache modify-replication-group --replication-group-id <replication-group-
value> --user-group-ids-to-remove <user-group-value>
{
    "..."
    "PendingModifiedValues": {
    "UserGroups": {
      "UserGroupIdsToAdd": [],
      "UserGroupIdsToRemove": [
        "<user-group-value>"
      ]
    }
    "..."
}
```

4. （可选）验证没有用户组与集群关联。AuthTokenEnabled 字段也应显示为 false。

```
aws elasticache describe-replication-groups --replication-group-id <replication-
group-value>
"AuthTokenEnabled": false
```

禁用配置了 RBAC 的 Redis 集群

1. 删除用户组以禁用访问控制。

```
aws elasticache modify-replication-group --replication-group-id <replication-group-
value> --user-group-ids-to-remove <user-group-value>
{
    "..."
    "PendingModifiedValues": {
    "UserGroups": {
      "UserGroupIdsToAdd": [],
      "UserGroupIdsToRemove": [
        "<user-group-value>"
      ]
    }
    "..."
```

```
}
```

2. （可选）验证没有用户组与集群关联。AuthTokenEnabled 字段也应显示为 false。

```
aws elasticache describe-replication-groups --replication-group-id <replication-
group-value>
"AuthTokenEnabled": false
```

互联网络流量隐私保护

Amazon ElastiCache 使用以下技术保护您的缓存数据免受未经授权的访问：

- Amazon VPC 和 ElastiCache 安全性 说明了安装所需的安全组类型。
- 适用于亚马逊的身份和访问管理 ElastiCache 用于授予和限制用户、组和角色的操作。

Amazon VPC 和 ElastiCache 安全性

由于数据安全性非常重要，ElastiCache 为您提供了控制哪些人可访问您的数据的方法。如何控制对数据的访问取决于您是在 Amazon Virtual Private Cloud (Amazon VPC) 中还是在 Amazon EC2-Classic 中启动您的集群。

> **Important**
>
> 我们已经弃用了使用 Amazon EC2-Classic 来启动 ElastiCache 集群。所有当前生成节点都仅在 Amazon Virtual Private Cloud 中启动。

Amazon Virtual Private Cloud (Amazon VPC) 服务定义一个与传统数据中心非常相似的虚拟网络。在配置您的 Amazon 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。您还可以将缓存集群添加到虚拟网络，并使用 Amazon VPC 安全组控制对缓存集群的访问。

本部分说明如何在 Amazon VPC 中手动配置 ElastiCache 集群。这些信息适用于希望更深入地了解 ElastiCache 和 Amazon VPC 如何协同工作的用户。

主题

- 了解 ElastiCache 和 Amazon VPC
- 访问 Amazon VPC 中 ElastiCache 缓存的访问模式

- 创建虚拟私有云（VPC）
- 连接到在 Amazon VPC 中运行的缓存

了解 ElastiCache 和 Amazon VPC

ElastiCache 与 Amazon Virtual Private Cloud (Amazon VPC) 完全集成。对于 ElastiCache 用户，这具有以下意义：

- 如果您的 AWS 账户仅支持 EC2-VPC 平台，ElastiCache 将始终在 Amazon VPC 中启动您的集群。
- 如果您刚开始使用 AWS，则您的集群将会部署到 Amazon VPC 中。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。

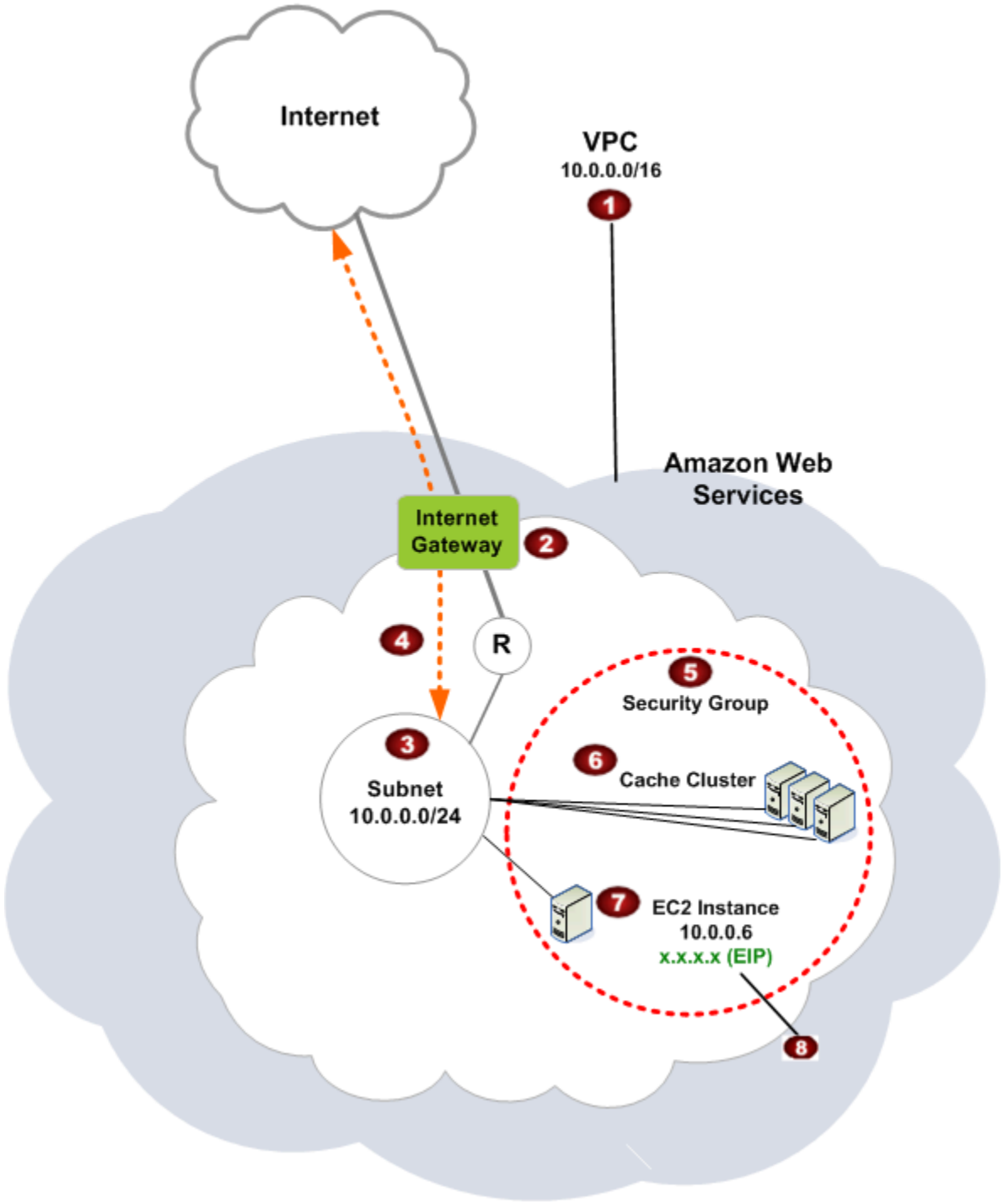
使用 Amazon Virtual Private Cloud，您可以在非常类似于传统数据中心的 AWS 云中创建虚拟网络。您可以配置您的 Amazon VPC，包括选择其 IP 地址范围、创建子网以及配置路由表、网关和安全设置。

ElastiCache 的基本功能与 Virtual Private Cloud 的基本功能相同；无论您的集群部署在 Amazon VPC 内部还是外部，ElastiCache 均可管理软件升级、修补、故障检测和恢复。

部署在 Amazon VPC 外部的 ElastiCache 缓存节点会分配有外部 IP 地址，端点/DNS 名称会解析为该地址。这可以实现连接 Amazon Elastic Compute Cloud (Amazon EC2) 实例。当您将 ElastiCache 集群启动到 Amazon VPC 私有子网时，每个缓存节点都在该子网内分配了一个私有 IP 地址。

Amazon VPC 中的 ElastiCache 概览

下图和表格介绍了 Amazon VPC 环境，以及在 Amazon VPC 中启动的 ElastiCache 集群和 Amazon EC2 实例。



1	Amazon VPC 是 AWS 云的一个独立部分，分配有自己的 IP 地址数据块。
2	

	互联网网关将您的 Amazon VPC 直接连接到互联网，并提供对其他 AWS 资源 [例如在 Amazon VPC 外部运行的 Amazon Simple Storage Service (Amazon S3)] 的访问。
3	Amazon VPC 子网是 Amazon VPC 的 IP 地址范围的一部分，在其中您可以根据您的安全和操作需求隔离 AWS 资源。
4	Amazon VPC 中的路由表可定向子网与互联网之间的网络流量。Amazon VPC 有一个隐式路由器，在本图中使用带圈的 R 表示。
5	Amazon VPC 安全组可以控制 ElastiCache 集群和 Amazon EC2 实例的入站和出站流量。
6	您可以在子网中启动 ElastiCache 集群。缓存节点具有子网地址范围内的私有 IP 地址。
7	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缓存节点。
8	要可以从互联网访问您的 Amazon 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 Amazon VPC 内创建 ElastiCache 集群，您的 Amazon VPC 必须满足以下要求：

- Amazon 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 Amazon VPC 中使用 ElastiCache。
- 必须为您的 Amazon VPC 定义缓存子网组。ElastiCache 使用该缓存子网组选择与 VPC 端点或缓存节点关联的子网和子网中的 IP 地址。
- 每个子网的 CIDR 块必须足够大，以便为 ElastiCache 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 Amazon VPC 中配置路由，以控制流量的流向（例如，流向互联网网关或虚拟私有网关）。使用互联网网关，您的 Amazon VPC 可以直接访问不在您的 Amazon VPC 中运行的其他 AWS 资源。如果您选择只使用一个虚拟专用网关连接至贵组织的本地网络，那么您可以通过 VPN 设置您的

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，当您通过互联网访问 AWS 资源时，便会产生额外的带宽费用。

您可以使用 Amazon VPC 安全组来帮助保护 Amazon VPC 中的 ElastiCache 集群和 Amazon EC2 实例。安全组在实例级上（而非子网级上）与防火墙的功能类似。

> **Note**
>
> 我们强烈建议您使用 DNS 名称连接到您的缓存节点，因为基础 IP 地址可能会改变。

Amazon VPC 文档

Amazon VPC 有一套专门文档，介绍如何创建和使用您的 Amazon VPC。下表提供指向 Amazon VPC 指南的链接。

描述	文档
如何开始使用 Amazon VPC	Amazon VPC 入门
如何通过 AWS 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	AWS Site-to-Site VPN 是什么？

有关 Amazon Virtual Private Cloud 的更多详细信息，请参阅 Amazon Virtual Private Cloud。

访问 Amazon VPC 中 ElastiCache 缓存的访问模式

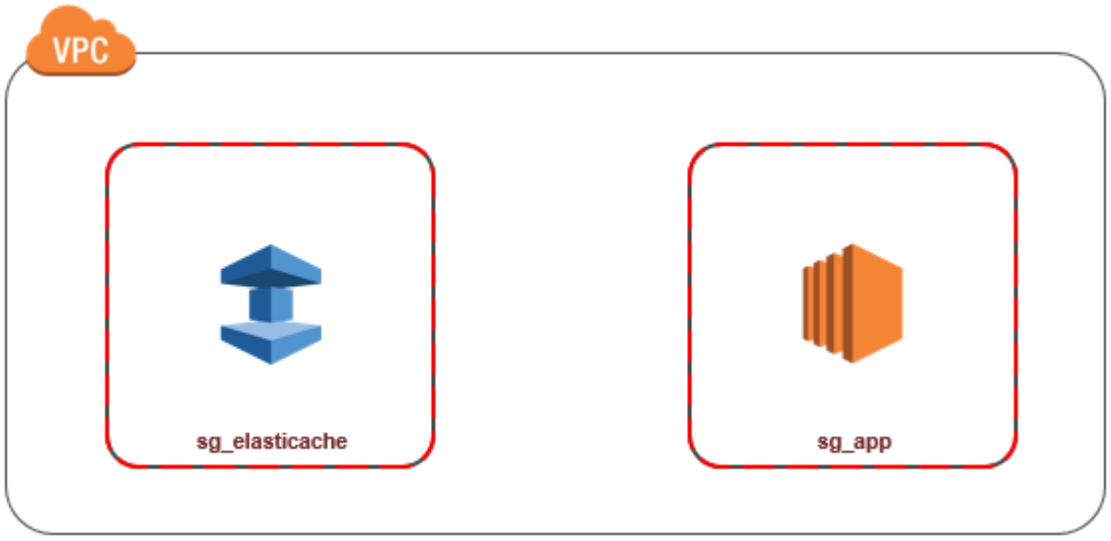
亚马逊 ElastiCache 支持以下场景访问亚马逊 VPC 中的缓存：

目录

当 ElastiCache 缓存和 Amazon EC2 实例位于同一 Amazon VPC 中时访问缓存

最常见的使用案例是，当 EC2 实例上部署的应用程序需要连接到同一 VPC 中的缓存时。

下图阐明了此方案。



通过执行以下操作，可以最轻松地管理同一 VPC 中的 EC2 实例与缓存之间的访问：

1. 为缓存创建 VPC 安全组。此安全组可用于限制对缓存的访问。例如，您可为此安全组创建自定义规则，允许使用您创建缓存时所分配的端口以及将用来访问缓存的 IP 地址进行 TCP 访问。

   Redis 缓存的默认端口为 6379。

2. 为 EC2 实例（Web 和应用程序服务器）创建 VPC 安全组。如果需要，此安全组可允许通过 VPC 的路由表从 Internet 访问 EC2 实例。例如，您可设置此安全组的规则以允许通过端口 22 对 EC2 实例进行 TCP 访问。
3. 在安全组中为缓存创建自定义规则，允许从为 EC2 实例创建的安全组进行连接。这将允许安全组的任何成员访问缓存。

> **Note**
>
> 如果您计划使用 Local Zones，请确保已将其启用。当您在该本地区域中创建子网组时，您的 VPC 也会扩展到该本地区域，并且您的 VPC 会将该子网视为任何其他可用区中的任何子网。所有相关网关和路由表都将自动调整。

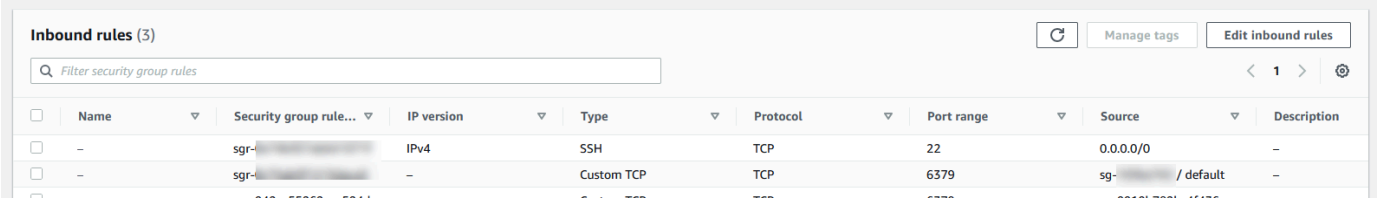
在 VPC 安全组中创建允许从另一安全组连接的规则

1. 登录 AWS 管理控制台并打开 Amazon VPC 控制台，网址为 https://console.aws.amazon.com/vpc。
2. 在导航窗格中，选择安全组。
3. 选择或创建将用于缓存的安全组。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. 对于端口范围，请指定在创建缓存时使用的端口。

      Redis 缓存和复制组的默认端口为 6379。

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



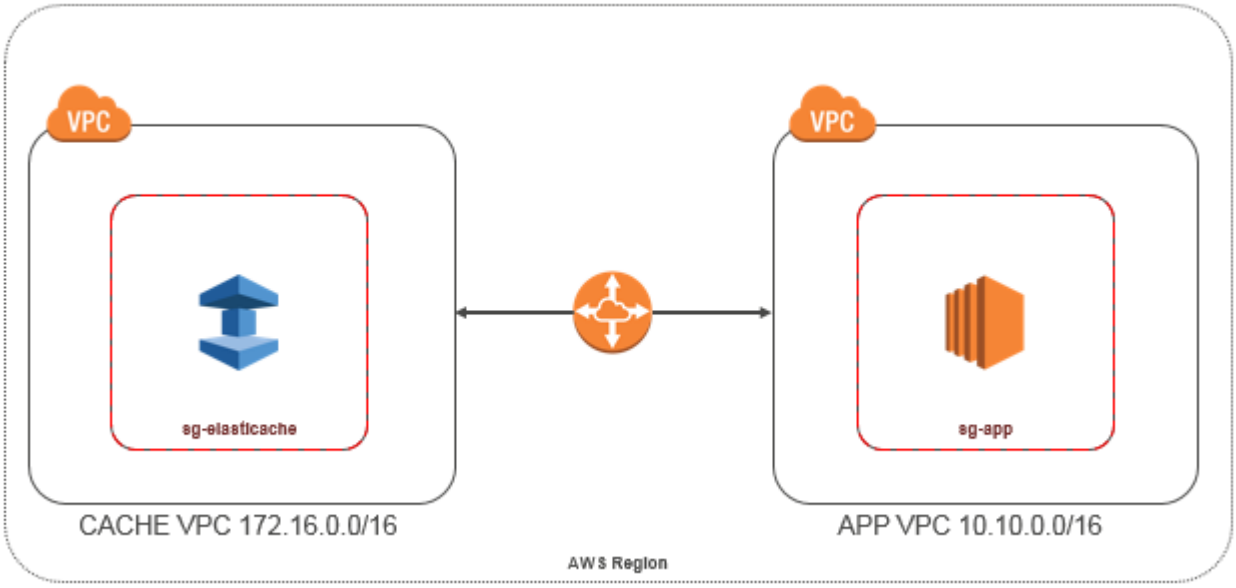
当 ElastiCache 缓存和 Amazon EC2 实例位于不同的 Amazon VPC 中时访问缓存

当您的缓存与用来访问它的 EC2 实例位于不同的 VPC 中时，可通过多种方式访问缓存。如果缓存和 EC2 实例位于同一区域的不同的 VPC 中，可以使用 VPC 对等连接。如果缓存和 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 ElastiCache 缓存和 Amazon EC2 实例位于同一区域的不同 Amazon VPC 中时访问缓存

下图演示了当 Amazon EC2 实例与缓存位于同一区域的不同 Amazon VPC 中时，如何使用 Amazon VPC 对等连接访问缓存。



缓存由同一区域的不同 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。您可以在自己的 Amazon VPC 之间创建 VPC 对等连接，也可以与单个区域内的其他 AWS 账户中的 Amazon VPC 创建 VPC 对等连接。要了解有关 Amazon VPC 对等连接的更多信息，请参阅 VPC 文档。

> **Note**
>
> 对等 VPC 的 DNS 名称解析可能会失败，具体取决于应用于 VPC ElastiCache 的配置。要解决此问题，必须为 DNS 主机名和 DNS 解析启用两种 VPC。有关更多信息，请参阅实现对 VPC 对等连接的 DNS 解析。

通过对等连接访问不同 Amazon VPC 中的缓存

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

   下面是上图中示例的路由表的形式。请注意，pcx-a894f1c1 是对等连接。

Destination	Target
172.16.0.0/16	local
10.10.0.0/16	pcx-a894f1c1

Destination	Target
10.10.0.0/16	local
0.0.0.0/0	igw-bfdcccd8
172.16.0.0/16	pcx-a894f1c1

   VPC 路由表
4. 修改 ElastiCache 缓存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。

通过对等连接访问缓存会产生额外的数据传输费用。

使用 Transit Gateway

通过传输网关，您可以连接同一 AWS 区域中的 VPC 和 VPN 连接，并在它们之间路由流量。公交网关跨 AWS 账户运行，您可以使用 Resource Acc AWS ess Manager 与其他账户共享您的公交网关。在您与其他账户共享公交网关后， AWS 账户所有者可以将其的 VPC 连接到您的公交网关。任一账户的用户都可以随时删除此挂载。

您可以在中转网关上启用多播，然后创建一个中转网关多播域，允许通过与域关联的 VPC 挂载，将多播流量从多播源发送到多播组成员。

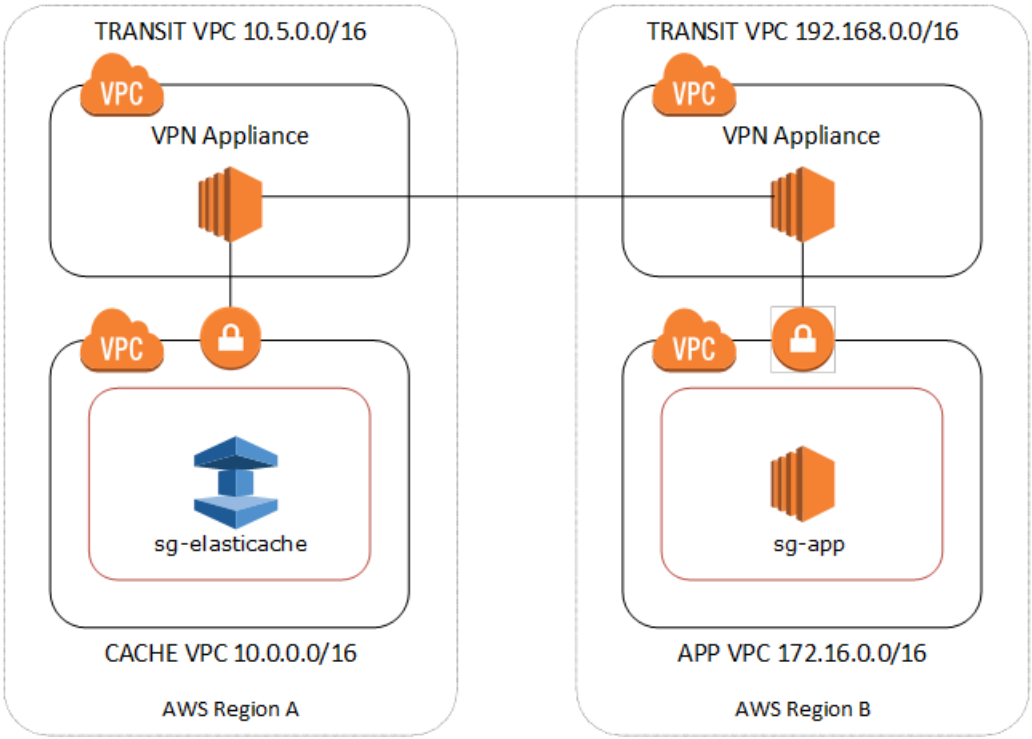
您还可以在不同 AWS 区域的中转网关之间创建对等连接连接。这使您能够跨不同区域在中转网关的挂载之间路由流量。

有关更多信息，请参阅中转网关。

当 ElastiCache 缓存和 Amazon EC2 实例位于不同区域的不同 Amazon VPC 中时访问缓存

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，部署在一个区域的 “分支” VPC 中的应用程序可以连接到另一个区域中 “分支” VPC 中的 ElastiCache 缓存。

访问不同 AWS 区域内不同 VPC 中的缓存

1. 部署传输 VPC 解决方案。有关更多信息，请参阅 AWS Transit Gateway。
2. 更新应用和缓存 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 ElastiCache 缓存的安全组以允许来自应用程序实例 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。

跨区域访问缓存会引入网络连接延迟，而且会产生额外的跨区域数据传输费用。

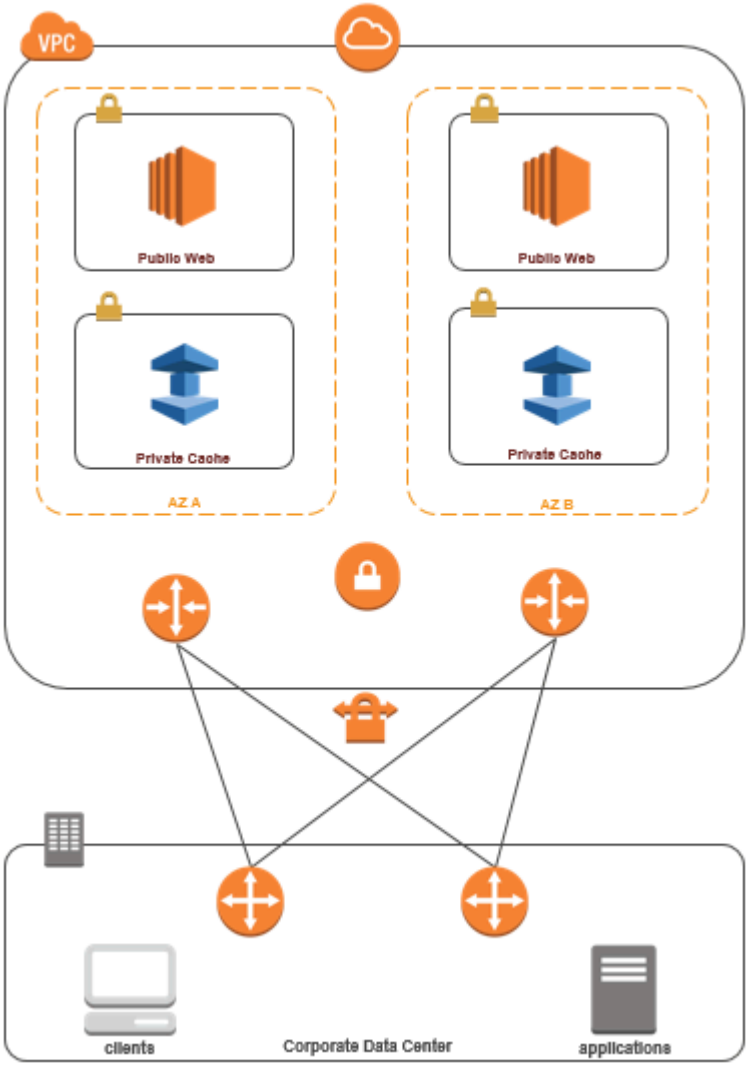
从客户数据中心运行的应用程序访问 ElastiCache 缓存

另一种可能的情况是混合架构，在这种架构中，客户数据中心中的客户端或应用程序可能需要访问 VPC 中的 ElastiCache 缓存。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 连接从客户数据中心运行的应用程序访问 ElastiCache 缓存

下图说明了使用 VPN 连接从企业网络中运行的应用程序访问 ElastiCache 缓存。



通过 VPN ElastiCache 从您的数据中心连接

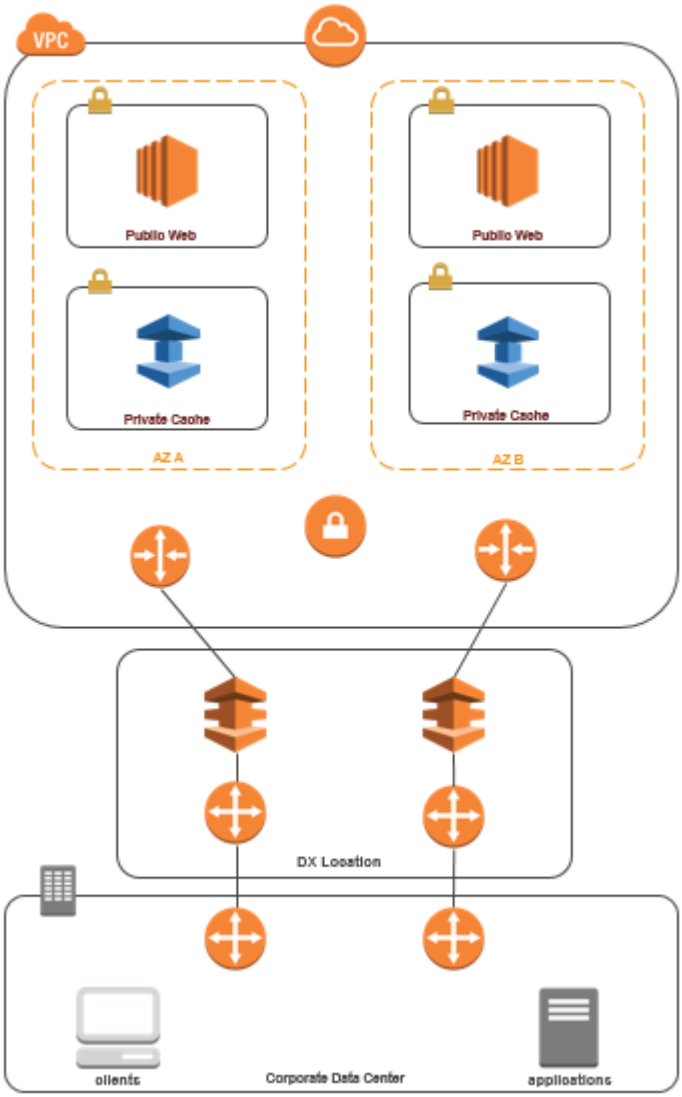
通过 VPN 连接从本地应用程序访问 VPC 中的缓存

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 ElastiCache 缓存的子网的 VPC 路由表，以允许来自本地应用服务器的流量。对于使用 BGP 的动态路由，可自动传播您的路由。
3. 修改 ElastiCache 缓存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 连接访问缓存将造成网络连接延迟并产生其他数据传输费用。

使用 Direct Connect 从客户数据中心运行的应用程序访问 ElastiCache 缓存

下图说明了使用 Direct Connect 从企业网络上运行的应用程序访问 ElastiCache 缓存。



通过 Direct Connect ElastiCache 从您的数据中心进行连接

使用 Direct Connect 从网络中运行的应用程序访问 ElastiCache 缓存

1. 建立 Direct Connect 连接。有关更多信息，请参阅 Di AWS rect Connect 入门。
2. 修改 ElastiCache 缓存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 连接访问缓存可能会造成网络连接延迟并产生其他数据传输费用。

创建虚拟私有云（VPC）

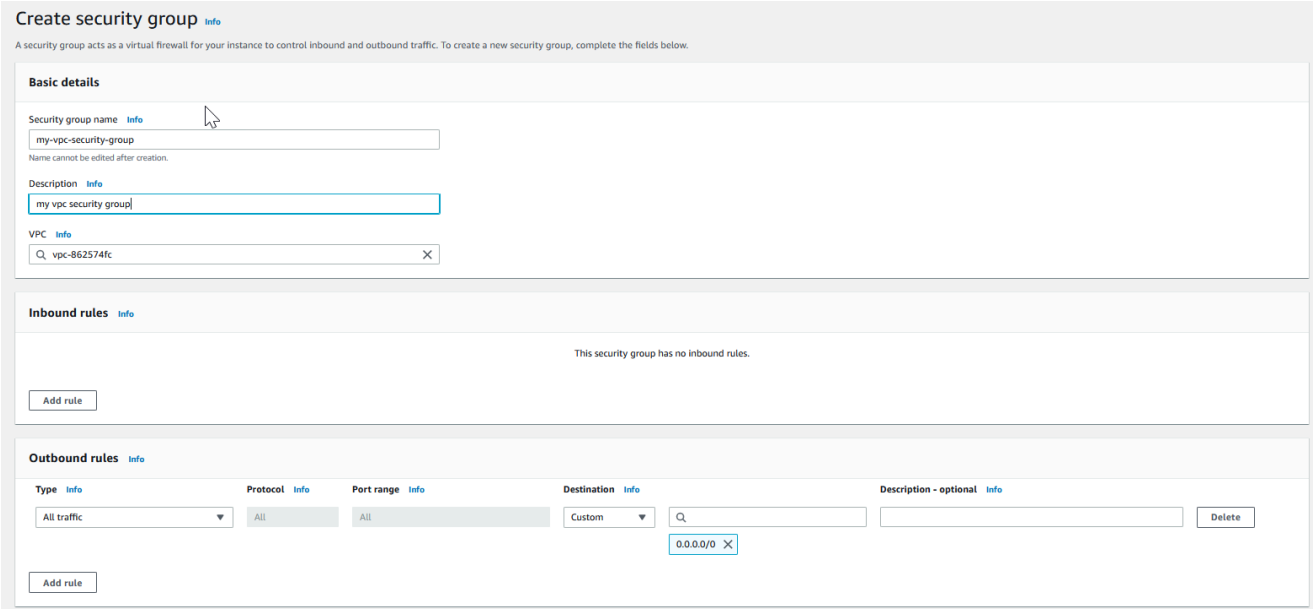
在本示例中，您创建一个 Amazon VPC，其中每个可用区域都有一个私有子网。

创建 Amazon VPC（控制台）

1. 登录 AWS 管理控制台并通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
2. 在 VPC 控制面板上，选择 Create VPC（创建 VPC）。
3. 在要创建的 Resources（资源）下，选择 VPC and more（VPC 等）。
4. 在 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，以及哪个是公有的，哪个是私有的。稍后，当您启动集群以及向 Amazon VPC 添加 Amazon EC2 实例时，您将需要此类信息。

7. 创建 Amazon VPC 安全组。您将对缓存集群和 Amazon EC2 实例使用此安全组。
   a. 在 Amazon VPC 管理控制台的导航窗格中，选择 Security Groups（安全组）。
   b. 选择创建安全组。
   c. 在相应的框内，为您的安全组键入名称和描述。在 VPC 框中，选择 Amazon VPC 标识符。



d. 根据需要完成所有设置后，选择 Yes, Create。

8. 为您的安全组定义一个网络入口规则。此规则将允许您使用 Secure Shell (SSH) 连接至 Amazon EC2 实例。

   a. 在导航列表中，选择 Security Groups。

   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。

   d. 为新入站规则设置以下值以允许 HTTP 访问：

      - 类型：HTTP
      - 来源：0.0.0.0/0

      选择 Apply Rule Changes。

现在，您已准备就绪，可在 Amazon VPC 中创建缓存子网组并启动缓存集群。

- 创建子网组
- 创建 Redis（已禁用集群模式）集群（控制台）.

连接到在 Amazon VPC 中运行的缓存

此示例演示如何在 Amazon VPC 中启动 Amazon EC2 实例。然后，您可以登录此实例并访问正在 Amazon VPC 中运行的 ElastiCache 缓存。

连接到在 Amazon VPC 中运行的缓存（控制台）

在本示例中，您将在 Amazon VPC 中创建 Amazon EC2 实例。您可以使用此 Amazon EC2 实例连接到在 Amazon VPC 中运行的缓存节点。

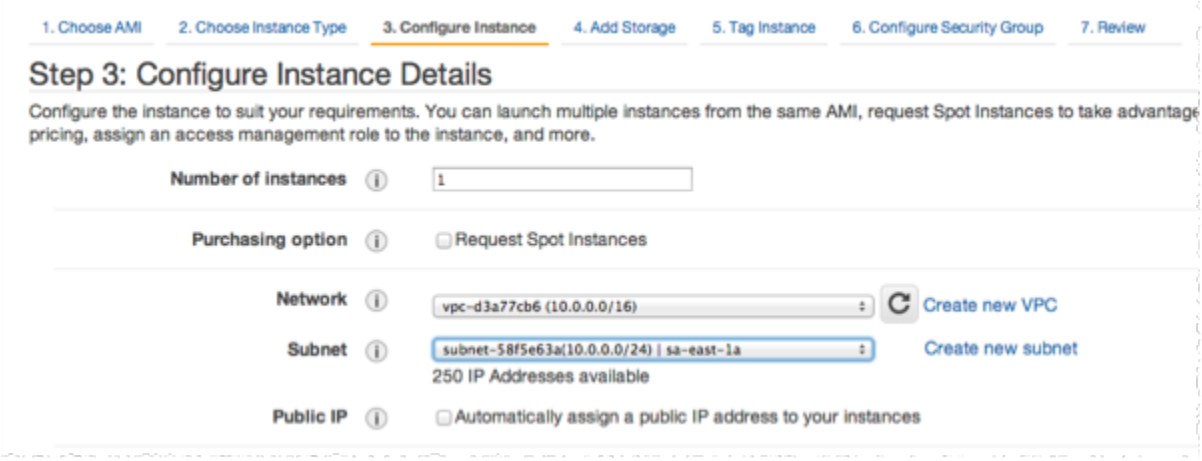
> **Note**
>
> 有关使用 Amazon EC2 的信息，请参阅 Amazon EC2 文档中的 Amazon EC2 入门指南。

使用 Amazon EC2 控制台在 Amazon VPC 中创建 Amazon EC2 实例

1. 登录到 AWS Management Console 并打开 Amazon EC2 控制台（https://console.aws.amazon.com/ec2/）。
2. 在控制台中，选择启动实例并执行下列步骤：
3. 在选择一个 Amazon Machine Image (AMI) 页上，选择 64 位 Amazon Linux AMI，然后选择选择。
4. 在 Choose an Instance Type（选择实例类型）页面上，选择 3. Configure Instance（配置实例）。
5. 在配置实例详细信息页上，进行以下选择：

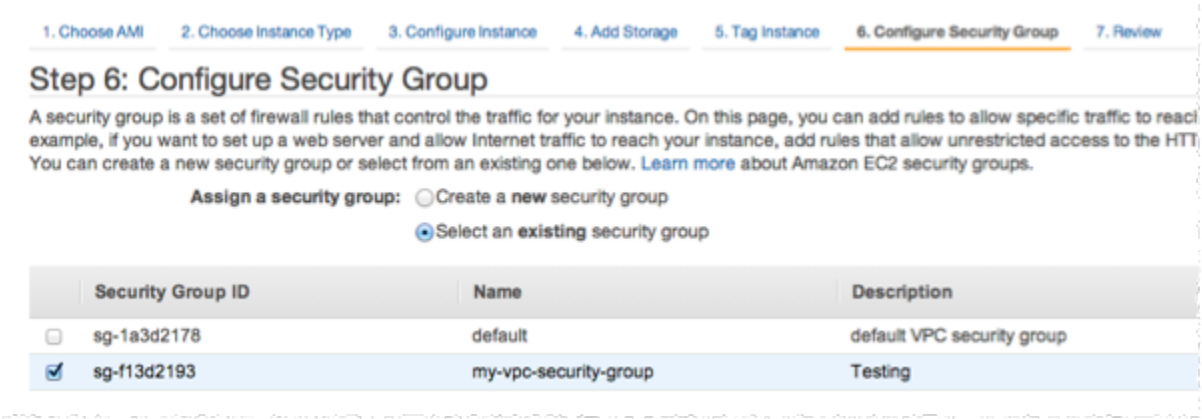
   a. 在 Network（网络）列表中，选择您的 Amazon VPC。

   b. 在子网列表中，选择您的公有子网。



根据需要进行设置后，选择 4. Add Storage（添加存储）。

6. 在 Add Storage（添加存储）页面上，选择 5. Tag Instance（标记实例）。
7. 在 Tag Instance（标记实例）页面上，为您的 Amazon EC2 实例键入名称，然后选择 6. Configure Security Group（配置安全组）。
8. 在配置安全组页上，选择选择一个现有的安全组。有关安全组的更多信息，请参阅 Linux 实例的 Amazon EC2 安全组。



选择 Amazon VPC 安全组的名称，然后选择 Review and Launch（审核和启动）。

9. 在 Review Instance and Launch (审核实例并启动) 页上，选择启动。
10. 在 Selct an existing key pair or create a new key pair（选择现有密钥对或创建新密钥对）窗口中，指定您要用于此实例的密钥对。

> **Note**
>
> 有关管理密钥对的信息，请参阅 Amazon EC2 入门指南。

11. 当您准备好启动您的 Amazon EC2 实例时，请选择 Launch（启动）。

您现在可以向刚创建的 Amazon EC2 实例分配弹性 IP 地址。您需要使用此 IP 地址连接到 Amazon EC2 实例。

分配弹性 IP 地址（控制台）

1. 通过以下网址打开 Amazon VPC 控制台：https://console.aws.amazon.com/vpc/。
2. 在导航列表中，选择弹性 IP。
3. 选择 Allocate Elastic IP address (分配弹性 IP 地址)。

4. 在分配弹性 IP 地址对话框中，接受默认的网络边界组，然后选择分配。
5. 选择您刚刚从列表中分配的弹性 IP 地址，然后选择关联地址。
6. 在 Associate Address（关联地址）对话框的 Instance（实例）框中，选择您启动的 Amazon EC2 实例的 ID。

   在私有 IP 地址框中，选中要获取私有 IP 地址的框，然后选择关联。

   您现在可以通过您创建的弹性 IP 地址，使用 SSH 连接到 Amazon EC2 实例。

连接到您的 Amazon EC2 实例

- 打开一个命令窗口。在命令提示符处，发出以下命令，将 mykeypair.pem 替换为您的密钥对文件的名称，并将 54.207.55.251 替换为弹性 IP 地址。

```
ssh -i mykeypair.pem ec2-user@54.207.55.251
```

> **Important**
>
> 暂时不要退出 Amazon EC2 实例。

您现在已准备好与 ElastiCache 集群进行交互。如果您尚未执行此操作，则需要先安装 telnet 实用工具，然后才能执行此操作。

安装 telnet 并与缓存群集（AWS CLI）交互

1. 打开一个命令窗口。在命令提示符下，发布以下命令。在确认提示符处，键入 y。

```
sudo yum install telnet
Loaded plugins: priorities, security, update-motd, upgrade-helper
Setting up Install Process
Resolving Dependencies
--> Running transaction check

...(output omitted)...

Total download size: 63 k
Installed size: 109 k
Is this ok [y/N]: y
```

```
Downloading Packages:
telnet-0.17-47.7.amzn1.x86_64.rpm                          |  63 kB     00:00

...(output omitted)...

Complete!
```

2. 使用 telnet 通过端口 6379 连接至您的缓存节点端点。将下面显示的主机名替换为缓存节点的主机名。

```
telnet my-cache-cluster.7wufxa.0001.use1.cache.amazonaws.com 6379
```

现在，您的系统已连接至缓存引擎并且能够发出命令。在本示例中，将一个数据项添加到缓存中，然后立即获取该数据项。最后，断开与缓存节点的连接。

要存储键和值，请键入以下两行：

```
set mykey myvalue
```

缓存引擎响应以下内容：

```
OK
```

要检索 mykey 的值，请键入以下内容：

```
get mykey
```

要断开与缓存引擎的连接，请键入以下命令：

```
quit
```

3. 转到 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)，并获取缓存群集中某个节点的端点。有关更多信息，请参阅适用于 Redis 的查找连接端点。

4. 使用 telnet 通过端口 6379 连接至您的缓存节点端点。将下面显示的主机名替换为缓存节点的主机名。

```
telnet my-cache-cluster.7wufxa.0001.use1.cache.amazonaws.com 6379
```

现在，您的系统已连接至缓存引擎并且能够发出命令。在本示例中，将一个数据项添加到缓存中，然后立即获取该数据项。最后，断开与缓存节点的连接。

要存储密钥和值，请键入以下内容：

```
set mykey myvalue
```

缓存引擎响应以下内容：

```
OK
```

要检索 mykey 的值，请键入以下内容：

```
get mykey
```

缓存引擎响应以下内容：

```
get mykey
myvalue
```

要断开与缓存引擎的连接，请键入以下命令：

```
quit
```

> **Important**
>
> 为了避免您的 AWS 账户产生额外费用，在您使用这些示例实验之后，请务必删除您不再需要的任何 AWS 资源。

Amazon ElastiCache API 和接口 VPC 端点（AWS PrivateLink）

您可以通过创建 接口 VPC 端点在 VPC 和 Amazon ElastiCache API 端点之间建立私有连接。接口端点由 AWS PrivateLink 提供支持。AWS PrivateLink 使您能够私下访问 Amazon ElastiCache API 操作，而无需互联网网关、NAT 设备、VPN 连接或 AWS Direct Connect 连接。

VPC 中的实例不需要公有 IP 地址便可与 Amazon ElastiCache API 端点进行通信。您的实例也不需要公有 IP 地址即可使用任何可用的 ElastiCache API 操作。您的 VPC 和 Amazon ElastiCache 之间的流量不会脱离 Amazon 网络。每个接口终端节点均由子网中的一个或多个弹性网络接口表示。有关弹性网络接口的更多信息，请参阅《Amazon EC2 用户指南》中的弹性网络接口。

- 有关 VPC 终端节点的更多信息，请参阅 Amazon VPC 用户指南中的接口 VPC 终端节点 (AWS PrivateLink)。
- 有关 ElastiCache API 操作的更多信息，请参阅 ElastiCache API 操作。

在创建接口 VPC 端点后，如果您为端点启用私有 DNS 主机名，则默认 ElastiCache 端点（https://elasticache.*Region*.amazonaws.com）将解析为您的 VPC 端点。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.elasticache.Region.vpce.amazonaws.com
```

有关更多信息，请参阅《Amazon VPC 用户指南》中的接口 VPC 终端节点 (AWS PrivateLink)。ElastiCache 支持调用您的 VPC 中的所有 API 操作。

> **Note**
>
> 只能为 VPC 中的一个 VPC 端点启用私有 DNS 主机名。如果要创建额外的 VPC 端点，则应为其禁用私有 DNS 主机名。

VPC 端点注意事项

在为 Amazon ElastiCache API 端点设置接口 VPC 端点之前，请务必查看 Amazon VPC 用户指南中的接口端点属性和限制。可以从使用 AWS PrivateLink 的 VPC 中获取与管理 Amazon ElastiCache 资源相关的所有 ElastiCache API 操作。

ElastiCache API 端点支持 VPC 端点策略。默认情况下，允许通过端点对 ElastiCache 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 端点控制对服务的访问权限。

为 ElastiCache API 创建接口 VPC 端点

您可以使用 Amazon VPC 控制台或 AWS CLI 为 Amazon ElastiCache API 创建 VPC 端点。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 端点后，您可以为端点启用私有 DNS 主机名。当您执行此操作时，默认的 Amazon ElastiCache 端点（https://elasticache.*Region*.amazonaws.com）将解析为您的 VPC 端点。对于中国（北京）和中国（宁夏）AWS 区域，您可以通过 VPC 端点分别使用 `elasticache.cn-north-1.amazonaws.com.cn`（对于北京）和 `elasticache.cn-northwest-1.amazonaws.com.cn`（对于宁夏）发出 API 请求。有关更多信息，请参阅 Amazon VPC 用户指南中的通过接口端点访问服务。

为 Amazon ElastiCache API 创建 VPC 端点策略

您可以为 VPC 端点附加控制对 ElastiCache API 的访问的端点策略。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 端点控制对服务的访问。

Example ElastiCache API 操作的 VPC 端点策略

下面是用于 ElastiCache API 的端点策略示例。当附加到端点时，此策略会向所有委托人授予对列出的针对所有资源的 ElastiCache API 操作的访问权限。

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "elasticache:CreateCacheCluster",
   "elasticache:ModifyCacheCluster",
   "elasticache:CreateSnapshot"
  ],
  "Resource": "*"
 }]
}
```

Example 拒绝来自指定 AWS 账户的所有访问的 VPC 端点策略

以下 VPC 终端节点策略拒绝 AWS 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"
    ]
   }
  }
 ]
}
```

子网和子网组

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

如果您在 Amazon VPC 中创建自行设计集群，则必须使用一个子网组。ElastiCache 使用该子网组选择与节点关联的子网和子网中的 IP 地址。

ElastiCache 提供了一个默认 IPv4 子网组，或者您可以选择创建一个新的 IPv4 子网组。对于 IPv6，您需要创建一个带有 IPv6 CIDR 块的子网组。如果您选择双堆栈，则必须选择发现 IP 类型，即 IPv6 或 IPv4。

ElastiCache 无服务器不使用子网组资源，而是在创建子网时直接获取子网列表。

本部分介绍如何创建和利用子网以及子网组来管理对 ElastiCache 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 访问您的集群或复制组。

主题

- 创建子网组
- 将子网组分配到缓存

- 修改子网组
- 删除子网组

创建子网组

缓存子网组是您要为 VPC 中的缓存指定的子网集合。当您在 VPC 中启动缓存时，您需要选择一个缓存子网组。然后，ElastiCache 使用该缓存子网组，向缓存中的每个缓存节点分配子网范围内的 IP 地址。

当您创建新的子网组时，请记下可用 IP 地址的数量。如果子网只有很少的几个空闲 IP 地址，则您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。

如果您选择 IPv4 作为网络类型，则默认的子网组将可用，或者您可以选择创建一个新的子网组。ElastiCache 使用该子网组选择与节点关联的子网和子网中的 IP 地址。如果您选择双堆栈或 IPV6，则系统将引导您创建双堆栈或 IPV6 子网。有关网络类型的更多信息，请参阅网络类型。有关更多信息，请参阅在 VPC 中创建子网。

以下过程演示如何创建名为 `mysubnetgroup` 的子网组（控制台、AWS CLI 和 ElastiCache API）。

创建子网组（控制台）

以下过程介绍如何创建子网组（控制台）。

创建子网组（控制台）

1. 登录 AWS 管理控制台并打开 ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 在导航列表中，选择子网组。
3. 选择 Create subnet group (创建子网组)。
4. 在创建子网组向导中，执行以下操作。根据需要完成所有设置后，选择 Create（创建）。
    a. 在 Name 框中，为子网组键入名称。
    b. 在 Description 框中，为子网组键入描述。
    c. 在 VPC ID 框中，选择您的 Amazon VPC。
    d. 默认情况下会选择所有子网。在选定的子网面板中，单击管理，选择私有子网的可用区或 Local Zones 以及 ID，然后选择选择。
5. 在出现的确认信息中，选择 Close。

您的新子网组会显示在 ElastiCache 控制台的 Subnet Groups（子网组）列表中。您可以在窗口底部选择子网组以查看详细信息，例如与此组关联的所有子网。

创建子网组（AWS CLI）

在命令提示符处，使用命令 `create-cache-subnet-group` 创建子网组。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup \
    --cache-subnet-group-description "Testing" \
    --subnet-ids subnet-53df9c3a
```

对于 Windows：

```
aws elasticache create-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup ^
    --cache-subnet-group-description "Testing" ^
    --subnet-ids subnet-53df9c3a
```

该命令应该生成类似于下述信息的输出：

```
{
    "CacheSubnetGroup": {
        "VpcId": "vpc-37c3cd17",
        "CacheSubnetGroupDescription": "Testing",
        "Subnets": [
            {
                "SubnetIdentifier": "subnet-53df9c3a",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                }
            }
        ],
        "CacheSubnetGroupName": "mysubnetgroup"
    }
}
```

有关更多信息，请参阅 AWS CLI 主题 create-cache-subnet-group。

将子网组分配到缓存

创建子网组后，您便可以在 Amazon VPC 中启动缓存。有关更多信息，请参阅下列内容。

- 独立 Redis 集群 – 要启动单节点 Redis 集群，请参阅 创建 Redis（已禁用集群模式）集群（控制台）。在步骤 7.a [Advanced Redis Settings（高级 Redis 设置）] 中，选择 VPC 子网组。
- Redis（已禁用集群模式）复制组 – 要在 VPC 中启动 Redis（已禁用集群模式）复制组，请参阅 从头创建 Redis（已禁用集群模式）复制组。在步骤 7.b [Advanced Redis Settings（高级 Redis 设置）] 中，选择 VPC 子网组。
- Redis（已启用集群模式）复制组 – 创建 Redis（已启用集群模式）集群（控制台）。在步骤 6.i [Advanced Redis Settings（高级 Redis 设置）] 中，选择 VPC 子网组。

修改子网组

您可以修改子网组的描述，或者修改与子网组关联的子网 ID 列表。如果缓存目前正在使用某个子网，那么您不能从子网组中删除该子网的 ID。

以下过程介绍如何修改子网组。

修改子网组（控制台）

修改子网组

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在导航窗格中，选择子网组。
3. 在子网组列表中，选择您希望修改的子网组的单选按钮，然后选择修改。
4. 在选定的子网面板中，选择管理。
5. 对所选子网进行任何更改，然后单击选择。
6. 单击保存更改以保存您的更改。

修改子网组 (AWS CLI)

在命令提示符处，使用命令 `modify-cache-subnet-group` 修改子网组。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup \
    --cache-subnet-group-description "New description" \
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

对于 Windows：

```
aws elasticache modify-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup ^
    --cache-subnet-group-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：

```
{
    "CacheSubnetGroup": {
        "VpcId": "vpc-73cd3c17",
        "CacheSubnetGroupDescription": "New description",
        "Subnets": [
            {
                "SubnetIdentifier": "subnet-42dcf93a",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                }
            },
            {
                "SubnetIdentifier": "subnet-48fc12a9",
                "SubnetAvailabilityZone": {
                    "Name": "us-west-2a"
                }
            }
        ],
        "CacheSubnetGroupName": "mysubnetgroup"
    }
}
```

有关更多信息，请参阅 AWS CLI 主题 modify-cache-subnet-group。

删除子网组

如果您决定不再需要您的子网组，则可删除它。如果缓存目前正在使用某个子网组，则无法删除该子网组。

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 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在导航窗格中，选择子网组。
3. 在子网组列表中，选择要删除的子网组，然后选择 Delete。
4. 当系统要求您确认此操作时，请在文本输入字段中键入子网组的名称，然后选择删除。

删除子网组 (AWS CLI)

通过使用 AWS CLI，调用带以下参数的命令 delete-cache-subnet-group：

- `--cache-subnet-group-name` *`mysubnetgroup`*

对于 Linux、macOS 或 Unix：

```
aws elasticache delete-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup
```

对于 Windows：

```
aws elasticache delete-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup
```

此命令不生成任何输出。

有关更多信息，请参阅 AWS CLI 主题 delete-cache-subnet-group。

适用于亚马逊的身份和访问管理 ElastiCache

AWS Identity and Access Management (IAM) AWS 服务 可帮助管理员安全地控制对 AWS 资源的访问权限。IAM 管理员控制谁可以进行身份验证（登录）和授权（拥有权限）使用 ElastiCache 资源。您可以使用 IAM AWS 服务 ，无需支付额外费用。

主题

受众

您的使用方式 AWS Identity and Access Management (IAM) 会有所不同，具体取决于您所做的工作 ElastiCache。

服务用户-如果您使用 ElastiCache 服务完成工作，则管理员会为您提供所需的凭证和权限。当你使用更多 ElastiCache 功能来完成工作时，你可能需要额外的权限。了解如何管理访问权限有助于您向管理员请求适合的权限。如果您无法访问中的功能 ElastiCache，请参阅对 Amazon ElastiCache 身份和访问进行故障排除。

服务管理员-如果您负责公司的 ElastiCache 资源，则可能拥有完全访问权限 ElastiCache。您的工作是确定您的服务用户应访问哪些 ElastiCache 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要详细了解您的公司如何使用 IAM ElastiCache，请参阅亚马逊如何 ElastiCache 与 IAM 合作。

IAM 管理员 — 如果您是 IAM 管理员，则可能需要详细了解如何编写策略来管理访问权限 ElastiCache。要查看您可以在 IAM 中使用的 ElastiCache 基于身份的策略示例，请参阅。适用于 Amazon ElastiCache 的基于身份的策略示例

使用身份进行身份验证

身份验证是您 AWS 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 AWS 账户根用户任 IAM 角色进行身份验证（登录 AWS）。

您可以使用通过身份源提供的凭据以 AWS 联合身份登录。 AWS IAM Identity Center （ IAM Identity Center ）用户、贵公司的单点登录身份验证以及您的 Google 或 Facebook 凭据就是联合身份的示例。当您以联合身份登录时，您的管理员以前使用 IAM 角色设置了身份联合验证。当你使用联合访问 AWS 时，你就是在间接扮演一个角色。

根据您的用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录的更多信息 AWS，请参阅《AWS 登录 用户指南》中的如何登录到您 AWS 账户的。

如果您 AWS 以编程方式访问，则会 AWS 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 AWS 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅 IAM 用户指南中的签署 AWS API 请求。

无论使用何种身份验证方法，您可能需要提供其他安全信息。例如， AWS 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《AWS IAM Identity Center 用户指南》中的 多重身份验证和《IAM 用户指南》中的 在 AWS中使用多重身份验证（ MFA ）。

AWS 账户 root 用户

创建时 AWS 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 AWS 服务 和资源。此身份被称为 AWS 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。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《IAM 用户指南》中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 AWS 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C enter 目录中的用户，或者任何使用 AWS 服务 通过身份源提供的凭据进行访问的用户。 AWS Directory Service当联合身份访问时 AWS 账户，他们将扮演角色，角色提供临时证书。

要集中管理访问权限，建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和群组，也可以连接并同步到您自己的身份源中的一组用户和群组，以便在您的所有 AWS 账户 和应用程序中使用。有关 IAM Identity Center 的信息，请参阅《AWS IAM Identity Center 用户指南》中的什么是 IAM Identity Center？

IAM 用户和群组

I AM 用户是您 AWS 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（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，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

I AM 角色是您内部具有特定权限 AWS 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。您可以 AWS Management Console 通过切换角色在中临时担任 IAM 角色。您可以通过调用 AWS CLI 或 AWS API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问 – 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。如果您使用 IAM Identity Center，则需要配置权限集。为控制您的身份在进行身份验证后可以访问的内容，IAM Identity Center 将权限集与 IAM 中的角色相关联。有关权限集的信息，请参阅《AWS IAM Identity Center 用户指南》中的 权限集。
- 临时 IAM 用户权限 – IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取 – 您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 AWS 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解角色和基于资源的跨账户访问策略之间的区别，请参阅 IA M 用户指南中的跨账户资源访问。
- 跨服务访问 — 有些 AWS 服务 使用其他 AWS 服务服务中的功能。例如，当您在某个服务中进行调用时，该服务通常会在 Amazon EC2 中运行应用程序或在 Simple Storage Service (Amazon S3) 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。

- 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。
- 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。
- 服务相关角色-服务相关角色是一种链接到的服务角色。 AWS 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

- 在 A@@ mazon 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 AWS CLI 或 AWS API 请求的应用程序的临时证书。这优先于在 EC2 实例中存储访问密钥。要向 EC2 实例分配 AWS 角色并使其可供其所有应用程序使用，您需要创建附加到该实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅IAM 用户指南中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您可以 AWS 通过创建策略并将其附加到 AWS 身份或资源来控制中的访问权限。策略是其中的一个对象 AWS ，当与身份或资源关联时，它会定义其权限。 AWS 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 AWS 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM policy，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 AWS Management Console AWS CLI、或 AWS API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM policy。

基于身份的策略可以进一步归类为内联策略或托管策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 AWS 账户。托管策略包括 AWS 托管策略和客户托管策略。要了解如何在托管式策略和内联策略之间进行选择，请参阅 IAM 用户指南中的在托管式策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 AWS 托管策略。

访问控制列表 (ACL)

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持 ACL 的服务示例。 AWS WAF要了解有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的访问控制列表（ACL）概览。

其他策略类型

AWS 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界：权限边界是一个高级特征，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCP)-SCP 是 JSON 策略，用于指定组织或组织单位 (OU) 的最大权限。 AWS Organizations AWS Organizations 是一项用于对您的企业拥有的多 AWS 账户 项进行分组和集中管

理的服务。如果在组织内启用了所有功能，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中的实体（包括每个 AWS 账户根用户实体）的权限。有关 Organizations 和 SCP 的更多信息，请参阅《AWS Organizations 用户指南》中的 SCP 的工作原理。

- 会话策略 – 会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅《IAM 用户指南》中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 AWS 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

亚马逊如何 ElastiCache 与 IAM 合作

在使用 IAM 管理访问权限之前 ElastiCache，请先了解有哪些 IAM 功能可供使用 ElastiCache。

您可以在 Amazon 上使用的 IAM 功能 ElastiCache

IAM 功能	ElastiCache 支持
基于身份的策略	是
基于资源的策略	否
策略操作	是
策略资源	是
策略条件键	是
ACL	是
ABAC（策略中的标签）	是
临时凭证	是
主体权限	是

IAM 功能	ElastiCache 支持
服务角色	是
服务相关角色	是

要全面了解 ElastiCache 以及其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中的与 IAM 配合使用的AWS 服务。

基于身份的策略 ElastiCache

支持基于身份的策略	是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM policy。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

适用于 ElastiCache 的基于身份的策略示例

要查看 ElastiCache 基于身份的策略的示例，请参阅。适用于 Amazon ElastiCache 的基于身份的策略示例

ElastiCache 内基于资源的策略

支持基于资源的策略	否

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 AWS 服务。

要启用跨账户存取，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同位置时 AWS 账户，可信账户中的 IAM 管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅 IAM 用户指南中的跨账户在 IAM 中访问资源。

的政策行动 ElastiCache

支持策略操作	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 AWS API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

要查看 ElastiCache 操作列表，请参阅《服务授权参考》 ElastiCache中的 Amazon 定义的操作。

正在执行的策略操作在操作前 ElastiCache 使用以下前缀：

```
elasticache
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "elasticache:action1",
      "elasticache:action2"
         ]
```

您也可以使用通配符（*）指定多个操作。例如，要指定以单词 `Describe` 开头的所有操作，包括以下操作：

```
"Action": "elasticache:Describe*"
```

要查看 ElastiCache 基于身份的策略的示例，请参阅。适用于 Amazon ElastiCache 的基于身份的策略示例

的政策资源 ElastiCache

支持策略资源	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon 资源名称 (ARN) 指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符 (*) 指示语句应用于所有资源。

```
"Resource": "*"
```

要查看 ElastiCache 资源类型及其 ARN 的列表，请参阅《服务授权参考》 ElastiCache中的 Amazon 定义的资源。要了解您可以使用哪些操作来指定每种资源的 ARN，请参阅 Amazon 定义的操作。 ElastiCache

要查看 ElastiCache 基于身份的策略的示例，请参阅。适用于 Amazon ElastiCache 的基于身份的策略示例

ElastiCache 的策略条件键

支持特定于服务的策略条件键	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 Condition 元素（或 Condition 块）中，可以指定语句生效的条件。Condition 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 Condition 元素，或在单个 Condition 元素中指定多个键，则 AWS 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 AWS 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM policy 元素：变量和标签。

AWS 支持全局条件密钥和特定于服务的条件密钥。要查看所有 AWS 全局条件键，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

要查看 ElastiCache 条件密钥列表，请参阅《服务授权参考》 ElastiCache中的 Amazon 条件密钥。要了解您可以使用条件键的操作和资源，请参阅 Amazon 定义的操作 ElastiCache。

要查看 ElastiCache 基于身份的策略的示例，请参阅。适用于 Amazon ElastiCache 的基于身份的策略示例

中的访问控制列表 (ACL) ElastiCache

支持 ACL	是

访问控制列表（ACL）控制哪些主体（账户成员、用户或角色）有权访问资源。ACL 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

基于属性的访问控制 (ABAC) ElastiCache

支持 ABAC（策略中的标签）	是

基于属性的访问控制 (ABAC) 是一种授权策略，该策略基于属性来定义权限。在中 AWS，这些属性称为标签。您可以将标签附加到 IAM 实体（用户或角色）和许多 AWS 资源。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息,请参阅《IAM 用户指南》中的什么是 ABAC？。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

将临时凭证与配合使用 ElastiCache

支持临时凭证	是

当你使用临时证书登录时，有些 AWS 服务 不起作用。有关更多信息，包括哪些 AWS 服务 适用于临时证书，请参阅 IAM 用户指南中的AWS 服务 与 IA M 配合使用的信息。

如果您使用除用户名和密码之外的任何方法登录，则 AWS Management Console 使用的是临时证书。例如，当您 AWS 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的 切换到角色（控制台）。

您可以使用 AWS CLI 或 AWS API 手动创建临时证书。然后，您可以使用这些临时证书进行访问 AWS。 AWS 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

ElastiCache 的跨服务主体权限

支持转发访问会话 (FAS)	是

当您使用 IAM 用户或角色在中执行操作时 AWS，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 AWS 服务 向下游服务发出请求的请求。 AWS 服务只有当服务收到需要与其他 AWS 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两个操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

的服务角色 ElastiCache

支持服务角色	是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会中断 ElastiCache 功能。只有在 ElastiCache 提供操作指导时才编辑服务角色。

的服务相关角色 ElastiCache

支持服务相关角色	是

服务相关角色是一种与服务相关联的 AWS 服务服务角色。服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 AWS 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理服务相关角色的详细信息，请参阅能够与 IAM 搭配使用的AWS 服务。在表中查找服务相关角色列中包含 Yes 的表。选择是链接以查看该服务的服务相关角色文档。

适用于 Amazon ElastiCache 的基于身份的策略示例

原定设置情况下，用户和角色没有创建或修改 ElastiCache 资源的权限。他们也无法使用 AWS Management Console、AWS Command Line Interface (AWS CLI) 或 AWS API 执行任务。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。然后，管理员可以向角色添加 IAM 策略，并且用户可以担任角色。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略。

有关 ElastiCache 定义的操作和资源类型的详细信息，包括每种资源类型的 ARN 格式，请参阅《服务授权参考》中的 Amazon ElastiCache 的操作、资源和条件键。

主题

- 策略最佳实操
- 使用 ElastiCache 控制台

- 允许用户查看他们自己的权限

策略最佳实操

基于身份的策略确定某个用户是否可以创建、访问或删除您账户中的 ElastiCache 资源。这些操作可能会使 AWS 账户产生成本。创建或编辑基于身份的策略时，请遵循以下准则和建议：

- AWS 托管式策略及转向最低权限许可入门 - 要开始向用户和工作负载授予权限，请使用 AWS 托管式策略来为许多常见使用场景授予权限。您可以在 AWS 账户 中找到这些策略。建议通过定义特定于您的使用场景的 AWS 客户管理型策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略或工作职能的 AWS 托管式策略。
- 应用最低权限 – 在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限 – 您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果通过特定 AWS 服务（例如 AWS CloudFormation）使用服务操作，您还可以使用条件来授予对服务操作的访问权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言 (JSON) 和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，有助于制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM Access Analyzer 策略验证。
- 需要多重身份验证 (MFA) - 如果您所处的场景要求您的 AWS 账户 中有 IAM 用户或根用户，请启用 MFA 来提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的配置受 MFA 保护的 API 访问。

有关 IAM 中的最佳实践的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

使用 ElastiCache 控制台

要访问 Amazon ElastiCache 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关您的 AWS 账户中的 ElastiCache 资源的详细信息。如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于只需要调用 AWS CLI 或 AWS API 的用户，您无需为其提供最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍可使用 ElastiCache 控制台，请同时将 ElastiCache `ConsoleAccess` 或 `ReadOnly` AWS 托管式策略添加到实体。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联策略和托管式策略。此策略包括在控制台上完成此操作或者以编程方式使用 AWS CLI 或 AWS API 所需的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

对 Amazon ElastiCache 身份和访问进行故障排除

使用以下信息来帮助您诊断和修复在使用 ElastiCache 和 IAM 时可能遇到的常见问题。

主题

我无权在以下位置执行操作 ElastiCache

如果 AWS Management Console 告诉您您无权执行某项操作，则必须联系管理员寻求帮助。管理员是指提供用户名和密码的人员。

当 `mateojackson` 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不拥有虚构 `elasticache:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 elasticache:GetWidget on resource: my-example-widget
```

在这种情况下，Mateo 请求他的管理员更新其策略，以允许他使用 `elasticache:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。

我无权执行 iam：PassRole

如果您收到错误消息，提示您无权执行`iam:PassRole`操作，则必须更新您的策略以允许您将角色传递给 ElastiCache。

有些 AWS 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为的 IAM 用户`marymajor`尝试使用控制台在中执行操作时，会出现以下示例错误 ElastiCache。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。您的管理员是提供登录凭证的人。

我想允许 AWS 账户之外的人访问我的 ElastiCache 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。对于支持基于资源的策略或访问控制列表（ACL）的服务，您可以使用这些策略向人员授予对您的资源的访问权。

要了解更多信息，请参阅以下内容：

- 要了解是否 ElastiCache 支持这些功能，请参阅亚马逊如何 ElastiCache 与 IAM 合作。
- 要了解如何提供对您拥有的资源的访问权限 AWS 账户 ，请参阅 IAM 用户指南中的向您拥有 AWS 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 AWS 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 AWS 账户
- 要了解如何通过联合身份验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问的区别，请参阅 IA M 用户指南中的跨账户资源访问。

访问控制

您可以拥有有效的凭证来验证您的请求，但是除非您拥有权限，否则您无法创建或访问 ElastiCache 资源。例如，您必须具有创建集 ElastiCache 群的权限。

以下各节介绍如何管理的权限 ElastiCache。我们建议您先阅读概述。

- 管理对 ElastiCache 资源的访问权限的概览
- 将基于身份的策略（IAM 策略）用于 Amazon ElastiCache

管理对 ElastiCache 资源的访问权限的概览

每个 AWS 资源都归某个 AWS 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，Amazon ElastiCache 还支持向资源附加权限策略。

> **Note**
>
> 账户管理员（或管理员用户）是具有管理员权限的用户。有关更多信息，请参阅《IAM 用户指南》中的 IAM 最佳实操。

要提供访问权限，请为您的用户、组或角色添加权限：

- AWS IAM Identity Center 中的用户和群组：

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

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

主题

- ElastiCache API 权限：操作、资源和条件参考

Amazon ElastiCache 资源和操作

有关 ElastiCache 资源类型及其 ARN 的列表，请参阅《服务授权参考》中的 Amazon ElastiCache 定义的资源。要了解您可以使用哪些操作指定每个资源的 ARN，请参阅 Amazon ElastiCache 定义的操作。

了解资源所有权

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

- 假设您使用 AWS 账户的根账户凭证来创建缓存群集。在这种情况下，您的 AWS 账户是资源的拥有者。在 ElastiCache 中，该资源为缓存群集。
- 假设您在自己的 AWS 账户中创建了一个 IAM 用户并向该用户授予了创建缓存群集的权限。在这种情况下，用户可以创建缓存群集。但是，您的 AWS 账户（即该用户所属的账户）拥有缓存群集资源。
- 假设您在有权创建缓存群集的 AWS 账户中创建 IAM 角色。在这种情况下，任何可以代入该角色的人都可以创建缓存群集。该角色所属的 AWS 账户拥有缓存群集资源。

管理对资源的访问

权限策略规定谁可以访问哪些内容。下一节介绍创建权限策略时的可用选项。

> **Note**
>
> 本节讨论在 Amazon ElastiCache 范围内使用 IAM。这里不提供有关 IAM 服务的详细信息。有关完整的 IAM 文档，请参阅《IAM 用户指南》中的什么是 IAM？。有关 IAM 策略语法和说明的信息，请参阅《IAM 用户指南》中的 AWS IAM 策略参考。

附加到 IAM 身份的策略称为基于身份的策略（IAM 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）
- 指定策略元素：操作、效果、资源和主体

- 在策略中指定条件

基于身份的策略（IAM 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 ElastiCache 资源，例如缓存群集、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AWS账户（如账户 B）或某项AWS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 AWS 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 AWS 服务委托人。

  有关使用 IAM 委托权限的更多信息，请参阅 IAM 用户指南中的访问权限管理。

以下是允许用户对您的 AWS 账户执行 `DescribeCacheClusters` 操作的示例策略。ElastiCache 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": "DescribeCacheClusters",
      "Effect": "Allow",
      "Action": [
         "elasticache:DescribeCacheClusters"],
      "Resource": resource-arn
      }
   ]
}
```

有关对 ElastiCache 使用基于身份的策略的更多信息，请参阅 将基于身份的策略（IAM 策略）用于 Amazon ElastiCache。有关用户、组、角色和权限的更多信息，请参阅 IAM 用户指南中的身份（用户、组和角色）。

指定策略元素：操作、效果、资源和主体

对于每个 Amazon ElastiCache 资源（请参阅 Amazon ElastiCache 资源和操作），该服务都定义了一组 API 操作（请参阅操作）。为授予这些 API 操作的权限，ElastiCache 定义了一组您可以在策略中指定的操作。例如，对于 ElastiCache 集群资源，定义了以下操作：`CreateCacheCluster`、`DeleteCacheCluster` 和 `DescribeCacheCluster`。执行一个 API 操作可能需要多个操作的权限。

以下是最基本的策略元素：

- 资源 – 在策略中，您可以使用 Amazon Resource Name (ARN) 标识策略应用到的资源。有关更多信息，请参阅Amazon ElastiCache 资源和操作。
- 操作 – 您可以使用操作关键字标识要允许或拒绝的资源操作。例如，根据指定的 `Effect`，`elasticache:CreateCacheCluster` 权限允许或拒绝执行 Amazon ElastiCache `CreateCacheCluster` 操作的用户权限。
- 效果 — 您可以指定当用户请求特定操作（可以是允许或拒绝）时的效果。如果没有显式授予（允许）对资源的访问权限，则隐式拒绝访问。您也可显式拒绝对资源的访问。例如，您可以执行此操作，以确保用户无法访问资源，即使有其他策略授予了访问权限也是如此。
- 主体 – 在基于身份的策略（IAM 策略）中，附加了策略的用户是隐式主体。对于基于资源的策略，您可以指定要接收权限的用户、账户、服务或其他实体（仅适用于基于资源的策略）。

有关 IAM policy 语法和描述的更多信息，请参阅 IAM 用户指南中的AWS IAM policy 参考。

有关显示所有 Amazon ElastiCache API 操作的表，请参阅 ElastiCache API 权限：操作、资源和条件参考。

在策略中指定条件

当您授予权限时，可使用 IAM 策略语言来指定规定策略何时生效的条件。例如，您可能希望策略仅在特定日期后应用。有关使用策略语言指定条件的更多信息，请参阅《IAM 用户指南》中的条件。

要表示条件，您可以使用预定义的条件键。要使用 ElastiCache 特定的条件键，请参阅 使用条件键。您可以根据需要使用 AWS 范围内的条件键。有关 AWS 范围内的键的完整列表，请参阅《IAM 用户指南》中的条件的可用键。

适用于 Amazon ElastiCache 的 AWS 托管策略

AWS 托管式策略是由 AWS 创建和管理的独立策略。AWS 托管式策略旨在为许多常见用例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住，AWS 托管式策略可能不会为您的特定使用场景授予最低权限，因为它们可供所有 AWS 客户使用。我们建议通过定义特定于您的使用场景的客户管理型策略来进一步减少权限。

您无法更改 AWS 托管式策略中定义的权限。如果 AWS 更新在 AWS 托管式策略中定义的权限，则更新会影响该策略所附加到的所有主体身份（用户、组和角色）。当新的 AWS 服务启动或新的 API 操作可用于现有服务时，AWS 最有可能更新 AWS 托管式策略。

有关更多信息，请参阅《IAM 用户指南》中的 AWS 托管式策略。

AWS 托管策略：ElastiCacheServiceRolePolicy

您无法将 ElastiCacheServiceRolePolicy 附加至 IAM 实体。此策略附加到服务相关角色，允许 ElastiCache 代表您执行操作。

此策略允许 ElastiCache 根据管理缓存的需要，代表您管理 AWS 资源：

- `ec2` – 管理要连接到缓存节点的 EC2 联网资源，包括 VPC 端点（用于无服务器缓存）、弹性网络接口（ENI）（用于自行设计的集群）和安全组。
- `cloudwatch`：从服务将指标数据发送到 CloudWatch。
- `outposts`：允许在 AWS Outposts 上创建缓存节点。

您可以在 IAM 控制台上找到 ElastiCacheServiceRolePolicy 策略，也可以在《AWS Managed Policy Reference Guide》中找到 ElastiCacheServiceRolePolicy 策略。

AWS 托管策略：AmazonElastiCacheFullAccess

您可以将 `AmazonElastiCacheFullAccess` 策略附加到 IAM 身份。

此策略允许主体在使用 AWS 管理控制台时获得对 ElastiCache 的完整访问权限：

- `elasticache`：访问所有 API。

- iam：创建服务操作所需的服务相关角色。
- ec2 – 描述创建缓存所需的依赖 EC2 资源（VPC、子网、安全组），并允许创建 VPC 端点（用于无服务器缓存）。
- kms – 允许使用客户自主管理型密钥进行静态加密。
- cloudwatch：允许对指标的访问权限，以便在控制台中显示 ElastiCache 指标。
- application-autoscaling：允许访问权限，以便描述缓存的自动缩放策略。
- logs：用于填充日志流，以便在控制台中使用日志传输功能。
- firehose – 用于填充传输送流，以便在控制台中使用日志传输功能。
- s3：用于填充 S3 存储桶，以便在控制台中使用快照还原功能。
- outposts：用于填充 AWS Outposts，以便在控制台中用于缓存创建。
- sns：用于填充 SNS 主题，以便在控制台中使用通知功能。

您可以在 IAM 控制台上找到 AmazonElastiCacheFullAccess 策略，也可以在《AWS Managed Policy Reference Guide》中找到 AmazonElastiCacheFullAccess。

AWS 托管策略：AmazonElastiCacheReadOnlyAccess

您可以将 AmazonElastiCacheReadOnlyAccess 策略附加到 IAM 身份。

此策略允许主体在使用 AWS 管理控制台时获得对 ElastiCache 的只读访问权限：

- elasticache：只读 Describe API 的访问权限。

您可以在 IAM 控制台上找到 AmazonElastiCacheReadOnlyAccess 策略，也可在《AWS Managed Policy Reference Guide》中找到 AmazonElastiCacheReadOnlyAccess。

对 AWS 托管策略的 ElastiCache 更新

查看有关 ElastiCache 的 AWS 托管策略更新的详细信息（始于此服务开始跟踪这些更改）。有关此页面更改的自动提示，请订阅“ElastiCache 文档历史记录”页面上的 RSS 信息源。

更改	描述	日期
AmazonElastiCacheFullAccess：对现有策略的更新	ElastiCache 添加了用于管理无服务器缓存的新权限，并允	2023 年 11 月 27 日

更改	描述	日期
	许通过控制台使用所有服务功能。	
ElastiCacheServiceRolePolicy：对现有策略的更新	ElastiCache 添加了新的权限，允许管理无服务器缓存资源的 VPC 端点。	2023 年 11 月 27 日
ElastiCache 开启了更改跟踪	ElastiCache 开启了对其 AWS 托管策略更改的跟踪。	2020 年 2 月 7 日

将基于身份的策略（IAM 策略）用于 Amazon ElastiCache

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 Amazon ElastiCache 资源的访问的基本概念和选项的主题。有关更多信息，请参阅管理对 ElastiCache 资源的访问权限的概览。

本主题的各个部分涵盖以下内容：

- 适用于 Amazon ElastiCache 的 AWS 托管策略
- 客户管理型策略示例

下面介绍权限策略示例。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowClusterPermissions",
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache",
                "elasticache:CreateCacheCluster",
```

```
                "elasticache:DescribeServerlessCaches",
                "elasticache:DescribeReplicationGroups",
                "elasticache:DescribeCacheClusters",
                "elasticache:ModifyServerlessCache",
                "elasticache:ModifyReplicationGroup",
                "elasticache:ModifyCacheCluster"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUserToPassRole",
            "Effect": "Allow",
            "Action": [ "iam:PassRole" ],
            "Resource": "arn:aws:iam::123456789012:role/EC2-roles-for-cluster"
        }
    ]
}
```

该策略包含两条语句：

- 第一条语句授予执行 Amazon ElastiCache 操作（`elasticache:Create*`、`elasticache:Describe*`、`elasticache:Modify*`）的权限
- 第二条语句授予对 `Resource` 值末尾指定的 IAM 角色名称的 IAM 操作（`iam:PassRole`）的权限。

该策略不指定 `Principal` 元素，因为在基于身份的策略中，您未指定获取权限的委托人。附加了策略的用户是隐式委托人。向 IAM 角色附加权限策略后，该角色的信任策略中标识的主体将获取权限。

有关显示所有 Amazon ElastiCache API 操作及它们适用的资源的表，请参阅 ElastiCache API 权限：操作、资源和条件参考。

客户管理型策略示例

如果您未使用默认策略并选择使用自定义托管策略，请确保以下两项之一。您应该有权调用 `iam:createServiceLinkedRole`（有关更多信息，请参阅示例 4：允许用户调用 IAM CreateServiceLinkedRole API）。或者您应该已经创建了 ElastiCache 服务相关角色。

与使用 Amazon ElastiCache 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 AWS 开发工具包 和 AWS CLI 相关。

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 ElastiCache 控制台时，一些看起来简单的 ElastiCache 操作可能需要其他操作来支持它们。例如，`elasticache:CreateCacheCluster` 授予创建 ElastiCache 缓存群集的权限。但是，为执行此操作，ElastiCache 控制台使用一些 `Describe` 和 `List` 操作来填充控制台列表。

示例

示例 1：允许用户对 ElastiCache 资源进行只读访问

以下策略授予允许用户列出资源的 ElastiCache 操作权限。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": "ECReadOnly",
      "Effect":"Allow",
      "Action": [
          "elasticache:Describe*",
          "elasticache:List*"],
      "Resource":"*"
      }
   ]
}
```

示例 2：允许用户执行常见的 ElastiCache 系统管理员任务

常见的系统管理员任务包括修改资源。系统管理员还可能需要获得有关 ElastiCache 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 ElastiCache 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": "ECAllowMutations",
      "Effect":"Allow",
      "Action":[
          "elasticache:Modify*",
          "elasticache:Describe*",
          "elasticache:ResetCacheParameterGroup"
      ],
      "Resource":"*"
      }
   ]
}
```

示例 3：允许用户访问所有 ElastiCache API 操作

以下策略允许用户访问所有 ElastiCache 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": "ECAllowAll",
      "Effect":"Allow",
      "Action":[
          "elasticache:*"
      ],
      "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 ElastiCache 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:AWSServiceName":"elasticache.amazonaws.com"
        }
      }
    }
  ]
}
```

示例 5：允许用户使用 IAM 身份验证连接到无服务器缓存

以下策略允许任何用户在 2023 年 4 月 1 日到 2023 年 6 月 30 日之间，使用 IAM 身份验证连接到任何无服务器缓存。

```
{
  "Version" : "2012-10-17",
  "Statement" :
  [
    {
      "Effect" : "Allow",
      "Action" : ["elasticache:Connect"],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:serverlesscache:*"
      ],
      "Condition": {
        "DateGreaterThan": {"aws:CurrentTime": "2023-04-01T00:00:00Z"},
        "DateLessThan": {"aws:CurrentTime": "2023-06-30T23:59:59Z"}
      }
    },
    {
      "Effect" : "Allow",
      "Action" : ["elasticache:Connect"],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:user:*"
      ]
    }
  ]
}
```

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。多数 ElastiCache API 操作均支持根据操作行为而有所不同的资源类型。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。

有关 ElastiCache 资源类型及其 ARN 的列表，请参阅《服务授权参考》中的 Amazon ElastiCache 定义的资源。要了解您可以使用哪些操作指定每个资源的 ARN，请参阅 Amazon ElastiCache 定义的操作。

示例

- 示例 1：允许用户完全访问特定 ElastiCache 资源类型
- 示例 2：拒绝用户访问无服务器缓存。

示例 1：允许用户完全访问特定 ElastiCache 资源类型

以下策略明确允许无服务器缓存类型的所有资源。

```
{
        "Sid": "Example1",
        "Effect": "Allow",
        "Action": "elasticache:*",
        "Resource": [
             "arn:aws:elasticache:us-east-1:account-id:serverlesscache:*"
        ]
}
```

示例 2：拒绝用户访问无服务器缓存。

以下示例明确拒绝访问特定无服务器缓存。

```
{
        "Sid": "Example2",
        "Effect": "Deny",
        "Action": "elasticache:*",
        "Resource": [
            "arn:aws:elasticache:us-east-1:account-id:serverlesscache:name"
        ]
```

```
}
```

使用条件键

您可以指定决定 IAM policy 如何生效的条件。在 ElastiCache 中，您可以使用 JSON 策略的 `Condition` 元素将请求上下文中的键与您在策略中指定的键值进行比较。有关更多信息，请参阅 IAM JSON 策略元素：条件。

要查看 ElastiCache 条件键的列表，请参阅《服务授权参考》中的 Amazon ElastiCache 的条件键。

有关全局条件键的列表，请参阅 AWS 全局条件上下文键。

指定条件：使用条件键

要实现精细控制，您需要编写 IAM 权限策略，用于指定控制某些请求上单独参数集的条件。然后，将该策略应用于您使用 IAM 控制台创建的 IAM 用户、组或角色。

要应用条件，请将条件信息添加到 IAM policy 语句。在以下示例中，您指定了条件，为创建的所有自行设计缓存群集使用节点类型 `cache.r5.large`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
```

```
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:CacheNodeType": [
                        "cache.r5.large"
                    ]
                }
            }
        }
    ]
}
```

有关更多信息，请参阅基于标签的访问控制策略示例。

有关使用策略条件运算符的更多信息，请参阅 ElastiCache API 权限：操作、资源和条件参考。

策略示例：使用条件实现精细参数控制

此部分介绍对之前列出的 ElastiCache 参数实现精细访问控制的示例策略。

1. elasticache:MaximumDataStorage：指定无服务器缓存的最大数据存储量。使用提供的条件，客户不能创建存储超过特定数量数据的缓存。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDependentResources",
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:serverlesscachesnapshot:*",
                "arn:aws:elasticache:*:*:snapshot:*",
                "arn:aws:elasticache:*:*:usergroup:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
```

```
            "Resource": [
                "arn:aws:elasticache:*:*:serverlesscache:*"
            ],
            "Condition": {
                "NumericLessThanEquals": {
                    "elasticache:MaximumDataStorage": "30"
                },
                "StringEquals": {
                    "elasticache:DataStorageUnit": "GB"
                }
            }
        }
    ]
}
```

2. elasticache:MaximumECPUPerSecond：指定无服务器缓存的每秒最大 ECPU 值。使用提供的条件，客户不能创建每秒执行的 ECPU 数超过特定数量的缓存。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowDependentResources",
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:serverlesscachesnapshot:*",
                "arn:aws:elasticache:*:*:snapshot:*",
                "arn:aws:elasticache:*:*:usergroup:*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:serverlesscache:*"
            ],
            "Condition": {
                "NumericLessThanEquals": {
```

```
                    "elasticache:MaximumECPUPerSecond": "100000"
                }
            }
        }
    ]
}
```

3. elasticache:CacheNodeType：指定用户可以创建哪些 NodeType。使用提供的条件，客户可以为节点类型指定单个值或范围值。

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:CacheNodeType": [
                        "cache.t2.micro",
                        "cache.t2.medium"
                    ]
                }
            }
```

```
        }
    ]
}
```

4. elasticache:NumNodeGroups：创建节点组少于 20 个的复制组。

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
             "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
             "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "NumericLessThanEquals": {
                    "elasticache:NumNodeGroups": "20"
                }
            }
        }
    ]
}
```

5. elasticache:ReplicasPerNodeGroup：将每个节点的副本数指定为介于 5 到 10 之间。

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

```
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "NumericGreaterThanEquals": {
                    "elasticache:ReplicasPerNodeGroup": "5"
                },
                "NumericLessThanEquals": {
                    "elasticache:ReplicasPerNodeGroup": "10"
                }
            }
        }
    ]
}
```

6. elasticache:EngineVersion：指定引擎版本 5.0.6 的使用情况。

```
{
    "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
```

```
            ]
        },

        {
           "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:EngineVersion": "5.0.6"
                }
            }
        }
    ]
}
```

7. elasticache:EngineType：指定仅使用 Redis 引擎。

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
```

```
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:EngineType": "redis"
                }
            }
        }
    ]
}
```

8. elasticache:AtRestEncryptionEnabled：指定仅在已启用加密的情况下创建复制组。

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

         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:AtRestEncryptionEnabled": "true"
                }
            }
```

```
        }
    ]
}
```

9. elasticache:TransitEncryptionEnabled

   a. 针对 CreateReplicationGroup 操作将 `elasticache:TransitEncryptionEnabled` 条件键设置为 `false`，以指定只有在不使用 TLS 时才能创建复制组：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:TransitEncryptionEnabled": "false"
                }
            }
        }
    ]
}
```

   当在策略中针对 CreateReplicationGroup 操作将 `elasticache:TransitEncryptionEnabled` 条件键设置为 `false` 时，只有在不使用 TLS

时（即请求不包含设置为 `true` 的 `TransitEncryptionEnabled` 参数或设置为 `required` 的 `TransitEncryptionMode` 参数），才允许 `CreateReplicationGroup` 请求。

b. 针对 CreateReplicationGroup 操作将 `elasticache:TransitEncryptionEnabled` 条件键设置为 `true`，以指定只有在使用 TLS 时才能创建复制组：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:TransitEncryptionEnabled": "true"
                }
            }
        }
    ]
}
```

当在策略中针对 CreateReplicationGroup 操作将 `elasticache:TransitEncryptionEnabled` 条件键设置为 `true` 时，只有在请求包含设置为 `true` 的 `TransitEncryptionEnabled` 参数或设置为 `required` 的 `TransitEncryptionMode` 参数时，才允许 `CreateReplicationGroup` 请求。

c. 针对 ModifyReplicationGroup 操作将 elasticache:TransitEncryptionEnabled 设置为 true，以指定只有在使用 TLS 时才能修改复制组：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:ModifyReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "BoolIfExists": {
                    "elasticache:TransitEncryptionEnabled": "true"
                }
            }
        }
    ]
}
```

当在策略中针对 ModifyReplicationGroup 操作将 elasticache:TransitEncryptionEnabled 条件键设置为 true 时，只有在请求包含设置为 required 的 TransitEncryptionMode 参数时，才允许 ModifyReplicationGroup 请求。也可以选择包含设置为 true 的 TransitEncryptionEnabled 参数，但在这种情况下，该设置并不是启用 TLS 所必需的。

10elasticache:AutomaticFailoverEnabled：指定仅在已启用自动故障转移的情况下创建复制组。

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
```

```
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:AutomaticFailoverEnabled": "true"
                }
            }
        }
    ]
}
```

11.elasticache:MultiAZEnabled：指定不能在已禁用多可用区的情况下创建复制组。

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
```

```
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:MultiAZEnabled": "false"
                }
            }
        }
    ]
}
```

12.elasticache:ClusterModeEnabled：指定仅在已启用集群模式的情况下创建复制组。

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:ClusterModeEnabled": "true"
                }
            }
        }
    ]
}
```

13elasticache:AuthTokenEnabled：指定仅在已启用 AUTH 令牌的情况下创建复制组。

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

         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "Bool": {
                    "elasticache:AuthTokenEnabled": "true"
                }
            }
        }
    ]
}
```

14elasticache:SnapshotRetentionLimit：指定保留快照的天数（或最少/最多天数）。以下策略强制将备份存储至少 30 天。

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

```
         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup",
                "elasticache:CreateServerlessCache"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*",
                "arn:aws:elasticache:*:*:serverlesscache:*"
            ],
            "Condition": {
                "NumericGreaterThanEquals": {
                    "elasticache:SnapshotRetentionLimit": "30"
                }
            }
        }
    ]
}
```

15elasticache:KmsKeyId：指定客户自主管理型 AWS KMS 密钥的使用情况。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
        "Sid": "AllowDependentResources",
        "Effect": "Allow",
        "Action": [
            "elasticache:CreateServerlessCache"
```

```
        ],
        "Resource": [
            "arn:aws:elasticache:*:*:serverlesscachesnapshot:*",
            "arn:aws:elasticache:*:*:snapshot:*",
            "arn:aws:elasticache:*:*:usergroup:*"
        ]
    },
    {
        "Effect": "Allow",
        "Action": [
            "elasticache:CreateServerlessCache"
        ],
        "Resource": [
            "arn:aws:elasticache:*:*:serverlesscache:*"
        ],
        "Condition": {
            "StringEquals": {
                "elasticache:KmsKeyId": "my-key"
            }
        }
    }
  ]
}
```

16elasticache:CacheParameterGroupName：使用集群上某个企业的特定参数，指定非默认参数组。您还可以为参数组指定命名模式，或阻止删除特定参数组名称。以下是限制使用仅“my-org-param-group”的示例。

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

         {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },
```

```
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:CreateReplicationGroup"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:CacheParameterGroupName": "my-org-param-group"
                }
            }
        }
    ]
}
```

17.elasticache:CreateCacheCluster：如果请求标签 `Project` 丢失或不等于 `Dev`、`QA` 或 `Prod`，则拒绝 `CreateCacheCluster` 操作。

```
{
    "Version": "2012-10-17",
    "Statement": [
          {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*",
                "arn:aws:elasticache:*:*:securitygroup:*",
                "arn:aws:elasticache:*:*:replicationgroup:*"
            ]
        },
        {
            "Effect": "Deny",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
```

```
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "Null": {
                    "aws:RequestTag/Project": "true"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster",
                "elasticache:AddTagsToResource"
            ],
            "Resource": "arn:aws:elasticache:*:*:cluster:*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/Project": [
                        "Dev",
                        "Prod",
                        "QA"
                    ]
                }
            }
        }
    ]
}
```

18elasticache:CacheNodeType：允许使用 cacheNodeType cache.r5.large 或 cache.r6g.4xlarge 以及标签 Project=XYZ 来 CreateCacheCluster。

```
{
  "Version": "2012-10-17",
  "Statement": [
      {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster",
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:parametergroup:*",
```

```
        "arn:aws:elasticache:*:*:subnetgroup:*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "elasticache:CreateCacheCluster"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:cluster:*"
      ],
      "Condition": {
        "StringEqualsIfExists": {
          "elasticache:CacheNodeType": [
            "cache.r5.large",
            "cache.r6g.4xlarge"
          ]
        },
        "StringEquals": {
          "aws:RequestTag/Project": "XYZ"
        }
      }
    }
  ]
}
```

> **Note**
>
> 在创建策略以将标签和其他条件键一起强制执行时，由于使用 `--tags` 参数创建请求的额外 `elasticache:AddTagsToResource` 策略要求，条件键元素可能需要条件 `IfExists`。

将服务相关角色用于 Amazon ElastiCache

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

服务相关角色可让您更轻松地设置 Amazon ElastiCache，因为您不必手动添加必要的权限。这些角色已存在于您的 AWS 账户中，但与 Amazon ElastiCache 使用案例有关并有预定义的权限。只有

Amazon ElastiCache 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。这样可以保护您的 Amazon ElastiCache 资源，因为您不会无意中删除访问资源所需的必要权限。

有关支持服务相关角色的其他服务的信息，请参阅使用 IAM 的 AWS 服务并查找 Service-Linked Role（服务相关角色）列中显示为 Yes（是）的服务。选择是，可转到查看该服务的服务相关角色文档的链接。

目录

Amazon ElastiCache 的服务相关角色权限

创建服务相关角色所需的权限

允许 IAM 实体创建 AWSServiceRoleForElastiCache 服务相关角色

向该 IAM 实体的权限中添加以下策略声明：

```
{
    "Effect": "Allow",
    "Action": [
```

```
        "iam:CreateServiceLinkedRole",
        "iam:PutRolePolicy"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticache.amazonaws.com/
AWSServiceRoleForElastiCache*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "elasticache.amazonaws.com"}}
}
```

允许 IAM 实体删除 AWSServiceRoleForElastiCache 服务相关角色

向该 IAM 实体的权限中添加以下策略声明：

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticache.amazonaws.com/
AWSServiceRoleForElastiCache*",
    "Condition": {"StringLike": {"iam:AWSServiceName": "elasticache.amazonaws.com"}}
}
```

或者，您可以使用 AWS 托管式策略提供对 Amazon ElastiCache 的完全访问权限。

创建服务相关角色 (IAM)

您可以使用 IAM 控制台、CLI 或 API 创建服务相关角色。

创建服务相关角色（IAM 控制台）

您可使用 IAM 控制台创建服务相关角色。

创建服务相关角色 (控制台)

1. 登录 AWS 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 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 AWS Service（亚马逊云科技服务）。
4. 在 Or select a service to view its use cases（或选择服务以查看其使用案例）下，选择 ElastiCache。
5. 选择下一步: 权限。

6. 在 Policy name（策略名称）下，请注意此角色需要 ElastiCacheServiceRolePolicy。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）对于角色描述，编辑新服务相关角色的描述。
9. 检查角色，然后选择创建角色。

创建服务相关角色 (IAM CLI)

您可以从 AWS 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 elasticache.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 elasticache.amazonaws.com 的服务名称。

编辑 Amazon ElastiCache 的服务相关角色的描述

Amazon ElastiCache 不允许您编辑 AWS ServiceRoleForElastiCache 服务相关角色。创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。

编辑服务相关角色描述（IAM 控制台）

您可以使用 IAM 控制台编辑服务相关角色的描述。

编辑服务相关角色的描述 (控制台)

1. 在 IAM 控制台的导航窗格中，选择角色。
2. 以下代码示例显示如何将 IAM policy 附加到用户。

3. 在 Role description 的最右侧，选择 Edit。
4. 在框中输入新描述，然后选择 Save（保存）。

编辑服务相关角色描述 (IAM CLI)

您可以从 AWS 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 AWS CLI 执行 IAM 操作 get-role。

   Example

   ```
   $ aws iam get-role --role-name AWSServiceRoleForElastiCache
   ```

   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 AWS CLI 执行 IAM 操作 update-role-description。

   对于 Linux、macOS 或 Unix：

   ```
   $ aws iam update-role-description \
       --role-name AWSServiceRoleForElastiCache \
       --description "new description"
   ```

   对于 Windows：

   ```
   $ aws iam update-role-description ^
       --role-name AWSServiceRoleForElastiCache ^
       --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=AWSServiceRoleForElastiCache
   &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=AWSServiceRoleForElastiCache
   &Version=2010-05-08
   &Description="New description"
```

删除 Amazon ElastiCache 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特征或服务，建议您删除该角色。这样就没有未被主动监控或维护的未使用实体。但是，您必须先清除您的服务相关角色，然后才能将其删除。

Amazon ElastiCache 不会删除您的服务相关角色。

清除服务相关角色

您必须先确认该角色没有与之关联的资源（集群或复制组），然后才能使用 IAM 删除服务相关角色。

在 IAM 控制台中检查服务相关角色是否具有活动会话

1. 登录 AWS 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 AWS ServiceRoleForElastiCache 角色的名称（而不是复选框）。
3. 在所选角色的摘要页面上，选择访问顾问选项卡。
4. 在访问顾问选项卡查看服务相关角色的近期活动。

删除需要 AWSServiceRoleForElastiCache 的 Amazon ElastiCache 资源

- 要删除集群，请参阅以下内容：
  - 使用 AWS Management Console
  - 使用 AWS CLI
  - 使用 ElastiCache API
- 要删除复制组，请参阅以下内容：
  - 删除复制组（控制台）
  - 删除复制组 (AWS CLI)
  - 删除复制组 (ElastiCache API)

删除服务相关角色（IAM 控制台）

您可以使用 IAM 控制台删除服务相关角色。

删除服务相关角色（控制台）

1. 登录 AWS 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. 在确认对话框中，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AWS服务的时间。这样可帮助您确认角色当前是否处于活动状态。如果要继续，请选择 Yes, Delete 以提交服务相关角色进行删除。
5. 监视 IAM 控制台通知，以监控服务相关角色的删除进度。由于 IAM 服务相关角色删除是异步的，因此，在您提交角色进行删除后，删除任务可能成功，也可能失败。如果任务失败，您可以从通知中选择 View details 或 View Resources 以了解删除失败的原因。

删除服务相关角色 (IAM CLI)

您可以从 AWS 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。

ElastiCache API 权限：操作、资源和条件参考

在设置访问控制以及编写可附加到 IAM policy 的权限策略（基于身份或基于资源）时，可将下表作为参考。该表列出了每个 Amazon ElastiCache API 操作以及您可以授予执行该操作的权限的相应操作。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（*）。

您可以在 ElastiCache 策略中使用条件密钥来表达条件。要查看 ElastiCache特定条件键的列表以及它们适用的操作和资源类型，请参阅使用条件键。有关 AWS范围密钥的完整列表，请参阅 IAM 用户指南中的AWS 全局条件上下文密钥。

> **Note**
>
> 要指定操作，请在 API 操作名称之前使用 `elasticache:` 前缀（例如，`elasticache:DescribeCacheClusters`）。

要查看 ElastiCache 操作列表，请参阅《服务授权参考》 ElastiCache中的 Amazon 定义的操作。

Amazon 合规性验证 ElastiCache

作为多个合 AWS 规计划（例如 SOC、PCI、FedRAMP 和 HIPAA）的一部分，第三方审计师评估 AWS 服务的安全性和合规性。

要了解是否属于特定合规计划的范围，请参阅AWS 服务 “按合规计划划分的范围” ”，然后选择您感兴趣的合规计划。 AWS 服务 有关一般信息，请参阅AWS 合规计划AWS。

您可以使用下载第三方审计报告 AWS Artifact。有关更多信息，请参阅中的 “下载报告” 中的 “ AWS Artifact。

您在使用 AWS 服务 时的合规责任取决于您的数据的敏感性、贵公司的合规目标以及适用的法律和法规。 AWS 提供了以下资源来帮助实现合规性：

- 安全与合规性快速入门指南 — 这些部署指南讨论了架构注意事项，并提供了在这些基础上 AWS 部署以安全性和合规性为重点的基准环境的步骤。
- 在 A@@ mazon Web Services 上构建 HIPAA 安全与合规性 — 本白皮书描述了各公司如何使用 AWS 来创建符合 HIPAA 资格的应用程序。

> **Note**
>
> 并非所有 AWS 服务 人都符合 HIPAA 资格。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 合规资源AWS — 此工作簿和指南集可能适用于您所在的行业和所在地区。
- AWS 客户合规指南 — 从合规角度了解责任共担模式。这些指南总结了保护的最佳实践， AWS 服务并将指南映射到跨多个框架（包括美国国家标准与技术研究院 (NIST)、支付卡行业安全标准委员会 (PCI) 和国际标准化组织 (ISO)）的安全控制。
- 使用AWS Config 开发人员指南中的规则评估资源 — 该 AWS Config 服务评估您的资源配置在多大程度上符合内部实践、行业指导方针和法规。
- AWS Security Hub— 这 AWS 服务 提供了您内部安全状态的全面视图 AWS。Security Hub 通过安全控件评估您的 AWS 资源并检查其是否符合安全行业标准和最佳实践。有关受支持服务及控件的列表，请参阅 Security Hub 控件参考。
- Amazon GuardDuty — 它通过监控您的 AWS 账户环境中是否存在可疑和恶意活动，来 AWS 服务检测您的工作负载、容器和数据面临的潜在威胁。 GuardDuty 通过满足某些合规性框架规定的入侵检测要求，可以帮助您满足各种合规性要求，例如 PCI DSS。
- AWS Audit Manager— 这 AWS 服务 可以帮助您持续审计 AWS 使用情况，从而简化风险管理以及对法规和行业标准的合规性。

更多信息

有关 AWS 云合规性的一般信息，请参阅以下内容：

- 按服务分类的 FIPS 端点
- 中的服务更新 ElastiCache
- AWS 云合规性
- 责任共担模式
- AWS PCI DSS 合规计划

Amazon ElastiCache 中的恢复能力

AWS全球基础设施围绕AWS区域和可用区构建。AWS区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和操作在可用

区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础设施相比，可用区具有更高的可用性、容错性和可扩展性。

有关 AWS 区域和可用区的更多信息，请参阅 AWS 全球基础设施。

除 AWS 全球基础设施外，Amazon ElastiCache 还提供多种功能，以帮助支持您的数据恢复能力和备份需求。

主题

缓解故障

在规划 Amazon ElastiCache 实施时，您应做好计划，尽量减少故障对应用程序和数据的影响。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。

主题

缓解运行 Redis 时发生的故障

运行 Redis 引擎时，您有以下选项来最大程度地减小节点故障或可用区故障的影响。

缓解节点故障

无服务器缓存使用多可用区架构，自动缓解节点故障，因此节点故障对您的应用程序是透明的。自行设计集群必须得到妥善配置，以缓解单个节点出现故障的情况。

要缓解自行设计集群中 Redis 节点故障的影响，您有以下选择：

主题

缓解故障：Redis 复制组

Redis 复制组包含一个应用程序可从中读取和写入的主节点和 1 至 5 个只读副本节点。在向主节点写入数据时，也会在只读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. ElastiCache 检测失败的只读副本。
2. ElastiCache 使失败的节点下线。
3. ElastiCache 在同一 AZ 中启动并配置替换节点。
4. 新节点与主节点同步。

在此期间，应用程序可使用其他节点继续读取和写入。

Redis 多可用区

您可以在 Redis 复制组上启用多可用区。无论是否启用多可用区，都将自动检测并替换发生故障的主节点。执行此操作的方式因是否启用多可用区而异。

启用多可用区时

1. ElastiCache 检测主节点故障。
2. ElastiCache 将复制延迟最小的只读副本节点提升到主节点。
3. 其他副本将与新的主节点同步。
4. ElastiCache 在出现故障的主节点的 AZ 中启动只读副本。
5. 新节点将与新提升的主节点同步。

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，与未启用多可用区的情况相比，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 使用多可用区最大限度地 ElastiCache 缩短 Redis 的停机时间。

禁用多可用区时

1. ElastiCache 检测主故障。
2. ElastiCache 使主服务器脱机。
3. ElastiCache 创建并置备一个新的主节点来替换出现故障的主节点。
4. ElastiCache 将新的主副本与其中一个现有副本同步。
5. 同步完成时，新节点将发挥集群主节点的功能。

在此过程的步骤 1 到 4 中，您的应用程序无法写入主节点。不过，您的应用程序会继续从副本节点进行读取。

要提供额外保护，建议您启动在不同可用区 (AZ) 的复制组中的节点。如果这样做，可用区故障将仅影响该可用区中的节点，而不会影响其他节点。

有关更多信息，请参阅 使用复制组时的高可用性。

缓解可用区故障

无服务器缓存使用复制的多可用区架构，自动缓解可用区故障，因此可用区故障对您的应用程序是透明的。

要缓解自行设计集群中可用区故障的影响，对于每个分片，请将节点置于尽可能多的可用区中。

无论您的一个分片有多少个节点，如果所有这些节点都位于相同的可用区内，则该可用区的灾难性故障会导致您丢失分片的所有数据。但是，如果您将节点置于多个可用区内，则任一可用区的故障只会导致您丢失该可用区内的节点。

只要您丢失节点，就会导致性能下降，因为现在共享读取操作的节点更少了。在替换节点之前，性能下降将继续。

有关为 Redis 节点指定可用区的信息，请参阅创建 Redis（已禁用集群模式）集群（控制台）。

有关区域和可用区的更多信息，请参阅选择区域和可用区。

建议

我们建议对自行设计的集群创建无服务器缓存，因为这样您无需额外配置即可自动获得更好的容错能力。但是，在创建自行设计集群时，您需要规划两种类型的故障：单个节点故障和广泛的可用区故障。最佳的故障缓解计划将解决这两种故障。

尽可能减少节点故障的影响

为了尽可能减少节点故障的影响，我们建议您的实施在每个分片中使用多个节点，并将节点分布在多个可用区上。对于无服务器缓存，此过程自动完成。

对于自行设计的集群，我们建议您在复制组上启用多可用区， ElastiCache 以便在主节点出现故障时自动故障转移到副本。

最大程度地减小可用区故障的影响

要最大程度地减小可用区故障的影响，建议您在提供的不同可用区内启动节点。跨可用区均匀分布节点将最大程度地减小极少发生的可用区故障的影响。对于无服务器缓存，此过程自动完成。

其他预防措施

如果您正在运行 Redis，除了上述预防措施之外，建议您定期对集群进行备份。备份（快照）会创建一个 .rdb 文件，在出现故障或损坏时，可使用此文件还原缓存。有关更多信息，请参阅 快照和还原。

AWS ElastiCache 中的基础设施安全性

作为托管式服务，AWS ElastiCache 受到 AWS 全球网络安全程序（如 AWS 架构中心中的“安全性与合规性”部分所述）的保护。

您可以使用 AWS 发布的 API 调用，通过网络访问 ElastiCache。客户端必须支持传输层安全性协议（TLS）1.2 或更高版本。建议使用 TLS 1.3 或更高版本。客户端还必须支持具有完全向前保密 (PFS) 的密码套件，例如 Ephemeral Diffie-Hellman (DHE) 或 Elliptic Curve Ephemeral Diffie-Hellman (ECDHE)。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

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

中的服务更新 ElastiCache

ElastiCache 自动监控您的缓存、集群和节点队列，以便在服务更新可用时对其进行应用。无服务器缓存的服务更新会自动地透明应用。对于自行设计的集群，您可以设置一个预定义的维护时段，以便 ElastiCache 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，您可以控制何时对自行设计集群应用更新以及应用哪些更新。您还可以实时监控所选 ElastiCache 集群的更新进度。

管理服务更新

ElastiCache 定期发布自行设计的集群的服务更新。如果您有一个或多个符合条件的自设计集群用于这些服务更新，则更新发布后，您将通过电子邮件、SNS、Personal Health Dashboard (PHD) 和亚马逊 CloudWatch 活动收到通知。更新还会显示在 ElastiCache 控制台的 “服务更新” 页面上。通过使用此控制面板，您可以查看 ElastiCache 车队的所有服务更新及其状态。无服务器缓存的服务更新会透明地应用，无法通过服务更新进行管理。

您可以控制在自动更新开始前应用更新的时间。我们强烈建议您尽快应用任何安全更新类型的更新，以确保您的 ElastiCache 集群始终 up-to-date 安装最新的安全补丁。

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 available（可用）状态起开始向实例集应用服务更新。服务更新为累积更新。换句话说，您尚未应用的任何更新都包含在您的最新更新中。

如果服务更新启用了自动更新，则可以选择在服务更新可用时不采取任何操作。 ElastiCache 将安排在自动更新开始日期之后的某个集群即将到来的维护时段内应用更新。您将收到更新每个阶段的相关通知。

> **Note**
>
> 您只能应用那些具有 available（可用）或 scheduled（已安排）状态的服务更新。

有关查看和应用任何特定于服务的更新到适用 ElastiCache 集群的更多信息，请参阅使用控制台应用服务更新。

当您的一个或多个 ElastiCache 集群有新的服务更新可用时，您可以使用 ElastiCache 控制台、API 或 AWS CLI 来应用更新。以下各节说明了可用于应用更新的选项。

使用控制台应用服务更新

要查看可用服务更新的列表和其他信息，请转到控制台中的 Service Updates（服务更新）页面。

1. 登录 AWS Management Console 并打开亚马逊 ElastiCache 控制台，网址为 https://console.aws.amazon.com/elasticache/。
2. 在导航窗格中，选择 Service Updates（服务更新）。
3. 在 Service updates（服务更新）下，您可以查看以下内容：
   - Service update name（服务更新名称）：服务更新的唯一名称
   - Update type（更新类型）：服务更新的类型，可以是 security-update 或 engine-update
   - Update severity（更新严重性）：应用更新的优先级：
     - critical (关键)：我们建议您立即应用此更新（14 天或更短时间内）。

  - important (重要)：只要您的业务流程允许，我们建议您尽快应用此更新（30 天或更短时间内）。
  - medium (中等)：我们建议您尽快应用此更新（60 天或更短时间内）。
  - low (低)：我们建议您尽快应用此更新（90 天或更短时间内）。
- Engine version（引擎版本）：如果更新类型为 engine-update，则为正在更新的引擎版本。
- 发布日期：更新发布且可应用于集群的时间。
- 推荐申请截止日期:应用更新的 ElastiCache指导日期.
- Status (状态)：更新的状态，可为下列状态之一：
  - 可用：更新适用于必需的集群。
  - complete（完成）：已应用更新。
  - cancelled (已取消)：更新已被取消且不再需要。
  - expired (已过期)：再也无法应用更新。

4. 选择单个更新（不是其左侧的按钮）以查看服务更新的详细信息。

   在 Cluster update status（集群更新状态）部分中，您可以查看尚未应用服务更新或最近才应用服务更新的集群的列表。您可以查看每个集群的以下内容：

   - Cluster name（集群名称）：集群的名称
   - Nodes updated（已更新节点）：特定集群中已更新或仍对特定服务更新可用的各个节点的比率。
   - Update Type（更新类型）：服务更新的类型，可以是 security-update 或 engine-update
   - Status（状态）：集群服务更新的状态，为下列状态之一：
     - available（可用）：更新适用于必需的集群。
     - 进行中：正在对此集群应用更新。
     - 已计划：已计划更新日期。
     - 完成：已成功应用更新。完成状态的集群将在完成后显示 7 天。

     如果您选择任何或所有具有 available（可用）或 scheduled（已安排）状态的集群，然后选择 Apply now（立即应用），将开始对这些集群应用更新。

使用 AWS 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 AWS CLI检测和应用这些更新：

- 要检索可用的服务更新的描述，请运行以下命令：

  ```
  aws elasticache describe-service-updates --service-update-status
  available
  ```

  有关更多信息，请参阅describe-service-updates。

- 要对集群列表应用服务更新，请运行以下命令：

  ```
  aws elasticache batch-apply-update-action --service-update
  ServiceUpdateNameToApply=sample-service-update --cluster-names cluster-1
  cluster2
  ```

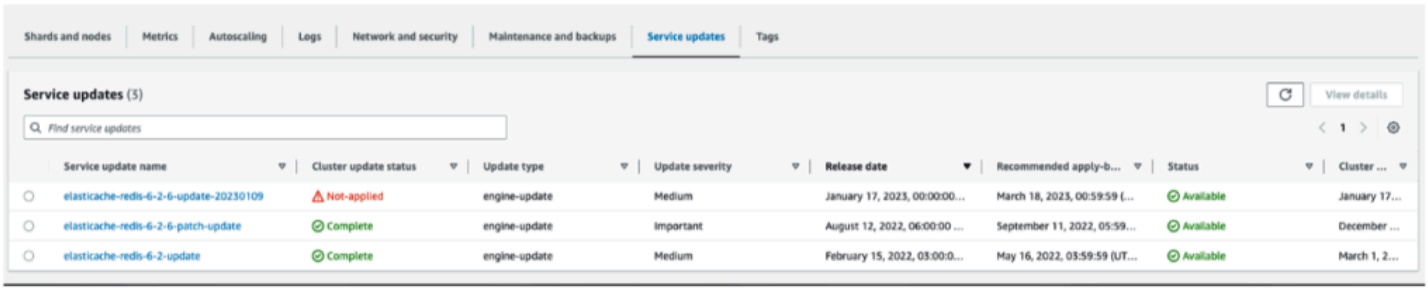
  有关更多信息，请参阅batch-apply-update-action。

使用 AWS 控制台验证是否应用了最新的服务更新

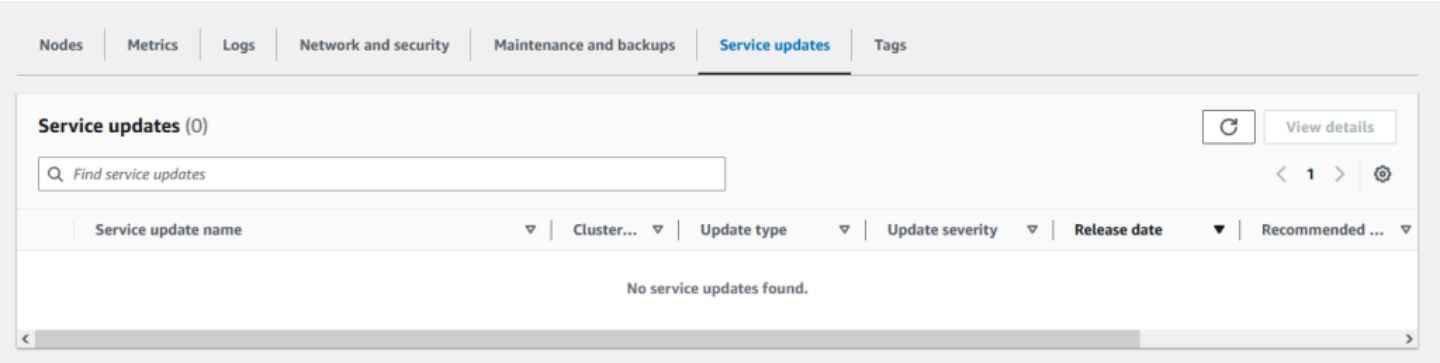
您可以按照以下步骤验证您 ElastiCache 的 For Redis 集群是否正在运行最新的服务更新：

1. 在 Redis 集群页面上选择适用的集群
2. 在导航窗格中选择 “服务更新”，查看适用于该集群的服务更新（如果有）。

如果控制台显示服务更新列表，则可以选择服务更新并选择立即应用。



如果控制台显示 “未找到服务更新”，则表示适用 ElastiCache 于 Redis 的集群已经应用了最新的服务更新。



停止服务更新

如果需要，您可以停止对集群的更新。例如，如果正在进行更新的集群出现意外激增，您可能希望停止更新。或者，如果更新花费时间过长并在高峰时间中断您的业务流程，您可能希望停止更新。

正在停止操作将立即中断对这些集群和任何尚未更新的节点的所有更新。它将继续完成对具有 in progress (正在进行中) 状态的任何节点的更新。不过，它将停止对同一集群中其他具有 update available (更新可用) 状态的节点的更新并将其状态恢复到 Stopping (正在停止) 状态。

在 Stopping (正在停止) 工作流程完成后，具有 Stopping (正在停止) 状态的节点将变为 Stopped (已停止) 状态。根据更新的工作流程，一些集群将不会具有任何更新的节点。其他集群可能包含一些已更新的节点和另一些仍具有 update available (更新可用) 状态的节点。

您可以稍后返回以在业务流程允许时完成更新过程。在此情况下，选择要完成更新的适用的集群，然后选择 Apply Now (立即应用)。有关更多信息，请参阅 应用服务更新。

使用 控制台

您可以使用 ElastiCache 控制台中断服务更新。以下内容演示了如何执行此操作：

- 在选定集群上进行服务更新后， ElastiCache 控制台将在仪表板顶部显示 “查看/停止更新” 选项卡。 ElastiCache
- 要中断更新，请选择 Stop Update (停止更新)。
- 在停止更新时，请选择集群并检查状态。它恢复到 Stopping (正在停止) 状态并最终变为 Stopped (已停止) 状态。

使用 AWS CLI

您可以使用 AWS CLI中断服务更新。以下代码示例演示如何执行此操作。

对于复制组，请执行以下操作：

```
aws elasticache batch-stop-update-action --service-update-name sample-service-update --replication-group-ids my-replication-group-1 my-replication-group-2
```

对于缓存群集，请执行以下操作：

```
aws elasticache batch-stop-update-action --service-update-name sample-service-update --cache-cluster-ids my-cache-cluster-1 my-cache-cluster-2
```

有关更多信息，请参阅BatchStopUpdateAction。

常见漏洞和披露 (CVE)：Redis 中 ElastiCache 已解决的安全漏洞

常见漏洞和风险 (CVE) 是公开已知网络安全漏洞的条目列表。每个条目都是一个链接，其中包含标识号、描述和至少一个公共参考文献。您可以在此页面上找到 Redis 中 ElastiCache 已解决的安全漏洞列表。

我们建议您始终将 Redis 版本升级到最新 ElastiCache 版本，以防范已知漏洞。操作 ElastiCache 无服务器缓存时，CVE 修复会自动应用于您的缓存。在操作自己设计的集群时，f ElastiCache or Redis 会公开 PATCH 组件。例如，在 Redis 版本 6.2.6 中使用 ElastiCache 时，主要版本为 6，次要版本为 2，补丁版本为 6。PATCH 版本用于向后兼容的错误修复、安全修复和不起作用的更改。

您可以使用此页面来验证特定版本的 for Redis 是否有针对特定安全漏洞的修复程序。 ElastiCache 如果您 ElastiCache 的 For Redis 集群运行的版本没有安全补丁，请参阅下表并采取措施。您可以升级到包含该修复程序的 Redis 的最新 ElastiCache 版本，或者如果您使用的是包含该修复程序 ElastiCache 的 Redis 版本，请通过参阅来确保应用了最新的服务更新。管理服务更新有关 Redis 引擎版本 ElastiCache 支持以及如何升级的更多信息，请参阅引擎版本和升级。

> **Note**
>
> - 如果在 for Redis 版本中 ElastiCache 寻址了 CVE，则意味着在较新的版本中也会对其进行寻址。因此，举例来说，如果在 Redis 版本 6.0.5 中 ElastiCache 解决了一个漏洞，那么在 6.2.6、7.0.7 和 7.1 版本中，这种情况会继续下去。
> - 下表中的星号 (*) 表示您必须为运行指定的 Redis 版本的 Redis 集群应用最新的服务更新，才能解决安全漏洞。 ElastiCache ElastiCache 有关如何验证您是否已为运行集群的 For Redis 版本应用了 ElastiCache 最新的服务更新的更多信息，请参阅管理服务更新。

ElastiCache 适用于 Redis 版本	已解决 CVE
Redis 6.0.5	CVE-2022-24735 *、CVE-2022-24736 *
Redis 6.2.6	CVE-2022-24834 *、CVE-2022-35977 *、CVE-2022-36021 *、CVE-2022-24735、CVE-2022-24736
Redis 7.0.7	CVE-2023-41056 *、CVE-2022-24834 *、CVE-2022-35977、CVE-2022-36021、CVE-2022-24735、CVE-2022-24736
Redis 7.1.0	CVE-2023-41056、CVE-2022-24834、CVE-2022-35977、CVE-2022-36021、CVE-2022-24735、CVE-2022-24736

Amazon ElastiCache 中的日志记录和监控

要管理缓存，您务必要了解缓存的性能情况。ElastiCache 会生成指标，这些指标发布到 Amazon CloudWatch Logs 用于监控缓存性能。此外，当您的缓存资源发生重大变化时（例如，创建新缓存或删除了缓存时），ElastiCache 会生成事件。

主题

无服务器指标和事件

此部分介绍在使用无服务器缓存时，您可以监控的指标和事件。

主题

无服务器缓存指标

`AWS/ElastiCache` 命名空间包括您的 Redis 无服务器缓存的以下 CloudWatch 指标。

指标	描述	单位
BytesUsedForCache	存储在缓存中的数据使用的总字节数。	字节
ElastiCacheProcessingUnits	在缓存上执行请求所消耗的 ElastiCacheProcessingUnits（ECPU）总数	计数
SuccessfulReadRequestLatency	成功读取请求的延迟。	微秒

指标	描述	单位
SuccessfulWriteRequestLatency	成功写入请求的延迟	微秒
TotalCmdsCount	在缓存中执行的所有命令的总数	计数
CacheHitRate	表示缓存的命中率。这是使用 `cache_hits` 和 `cache_misses` 统计数据按以下方式计算的：`cache_hits /(cache_hits + cache_misses)` 。	百分比
CacheHits	缓存中成功的只读键查找次数。	计数
CurrConnections	缓存的客户端连接数。	计数
ThrottledCmds	由于工作负载的扩展速度超过 ElastiCache 所能扩展的速度，因而被 ElastiCache 节流的请求数量。	计数
NewConnections	在此期间，服务器接受的连接总数。	计数
CurrItems	缓存中的项目数。	计数
CurrVolatileItems	带 TTL 的缓存中的项目数。	计数
NetworkBytesIn	传输到缓存的字节总数	字节
NetworkBytesOut	从缓存传出的字节总数	字节
移出	缓存驱逐的键的数量	计数

指标	描述	单位
IamAuthenticationExpirations	已过期的经过 IAM 身份验证的 Redis 连接总数。您可以在用户指南中找到有关 使用 IAM 进行身份验证 的更多信息。	计数
IamAuthenticationThrottling	受限的经过 IAM 身份验证的 Redis AUTH 或 HELLO 请求的总数。您可以在用户指南中找到有关 使用 IAM 进行身份验证 的更多信息。	计数
KeyAuthorizationFailures	用户访问其无权限访问的密钥的失败尝试次数。我们建议为此设置告警以检测未经授权的访问尝试。	计数
AuthenticationFailures	使用 AUTH 命令向 Redis 进行身份验证的失败尝试总次数。我们建议为此设置告警以检测未经授权的访问尝试。	计数
CommandAuthorizationFailures	用户运行其无权限调用的命令的失败尝试次数。我们建议为此设置告警以检测未经授权的访问尝试。	计数

命令级指标

ElastiCache 还会发出以下命令级别的指标。对于每个命令类型，ElastiCache 都会发出命令的总数以及该命令类型使用的 ECPU 数。

指标	描述	单位
EvalBasedCmds	缓存已收到的 get 命令数。	计数

指标	描述	单位
EvalBasedCmdsECPUs	基于 eval 的命令使用的 ECPU 数。	计数
GeoSpatialBasedCmds	基于地理空间的命令的命令总数。此值是从 Redis commandstats 统计数据得出。此值通过计算所有地理空间类型的命令的总和得出：geoadd、geodist、geohash、geopos、georadius 和 georadiusbymember。	计数
GeoSpatialBasedCmdsECPUs	基于地理空间的命令使用的 ECPU 数。	计数
GetTypeCmds	只读类型命令的总数。此值是通过计算所有只读类型的命令（get、hget、scard、lrange 等）的总和，从 Redis commandstats 统计数据得出。	计数
GetTypeCmdsECPUs	读取命令使用的 ECPU 数。	计数
HashBasedCmds	基于哈希的命令总数。此值是通过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，从 Redis commandstats 统计数据得出。	计数
HashBasedCmdsECPUs	基于哈希的命令使用的 ECPU 数。	计数

指标	描述	单位
HyperLogLogBasedCmds	基于 HyperLogLog 的命令的总数。此值是通过计算所有 pf 类型的命令（pfadd、pfcount、pfmerge 等）的总和，从 Redis commandstats 统计数据得出。	计数
HyperLogLogBasedCmdsECPUs	基于 HyperLogLog 的命令使用的 ECPU 数。	计数
JsonBasedCmds	JSON 命令的总数，包括读取和写入命令。此值是通过计算所有作用于 JSON 键的 JSON 命令的总和，从 Redis commandstats 统计数据得出。	计数
JsonBasedCmdsECPUs	所有 JSON 命令使用的 ECPU 数，包括读取和写入命令。	计数
JsonBasedGetCmds	JSON 只读命令的总数。此值是通过计算所有作用于 JSON 键的 JSON 读取命令的总和，从 Redis commandstats 统计数据得出。	计数
JsonBasedGetCmdsECPUs	JSON 只读命令使用的 ECPU 数。	计数
JsonBasedSetCmds	JSON 写入命令的总数。此值是通过计算所有作用于 JSON 键的 JSON 写入命令的总和，从 Redis commandstats 统计数据得出。	计数

指标	描述	单位
JsonBasedSetCmdsECPUs	JSON 写入命令使用的 ECPU 数。	计数
KeyBasedCmds	基于密钥的命令总数。此值是通过计算所有作用于多个数据结构中的一个或多个键的命令（del、expire、rename 等）的总和，从 Redis commandstats 统计数据得出。	计数
KeyBasedCmdsECPUs	基于键的命令使用的 ECPU 数。	计数
ListBasedCmds	基于列表的命令总数。此值是通过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，从 Redis commandstats 统计数据得出。	计数
ListBasedCmdsECPUs	基于列表的命令使用的 ECPU 数。	计数
NonKeyTypeCmds	不基于键的命令总数。此值是通过计算所有不作用于某个键的命令（acl、dbsize 或 info）的总和，从 Redis commandstats 统计数据得出。	计数
NonKeyTypeCmdsECPUs	未基于键的命令使用的 ECPU 数。	计数

指标	描述	单位
PubSubBasedCmds	用于发布/订阅功能的命令总数。此值是通过计算所有用于 pub/sub 功能的命令（psubscribe、publish、pubsub、punsubscribe、ssubscribe、sunsubscribe、spublish、subscribe 和 unsubscribe）的总和，从 Redis commandstats 统计数据得出。	计数
PubSubBasedCmdsECPUs	基于 pub/sub 的命令使用的 ECPU 数。	计数
SetBasedCmds	基于设置的命令总数。此值是通过计算所有作用于一个或多个集的命令（scard、sdiff、sadd、sunion 等）的总和，从 Redis commandstats 统计数据得出。	计数
SetBasedCmdsECPUs	基于集的命令使用的 ECPU 数。	计数
SetTypeCmds	写入类型命令的总数。此值是通过计算所有作用于数据的变化类型的命令（set、hset、sadd、lpop 等）的总和，从 Redis commandstats 统计数据得出。	计数
SetTypeCmdsECPUs	写入命令使用的 ECPU 数。	计数

指标	描述	单位
SortedSetBasedCmds	基于设置的已排序命令总数。此值是通过计算所有作用于一个或多个已排序集的命令（zcount、zrange、zrank、zadd 等）的总和，从 Redis commandstats 统计数据得出。	计数
SortedSetBasedCmdsECPUs	基于排序的命令使用的 ECPU 数。	计数
StringBasedCmds	基于字符串的命令总数。此值是通过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，从 Redis commandstats 统计数据得出。	计数
StringBasedCmdsECPUs	基于字符串的命令使用的 ECPU 数。	计数
StreamBasedCmds	基于流的命令总数。此值是通过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，从 Redis commandstats 统计数据得出。	计数
StreamBasedCmdsECPUs	基于流的命令使用的 ECPU 数。	计数

无服务器缓存事件

ElastiCache 会记录与您的无服务器缓存相关的事件。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。通过使用 ElastiCache 控制台、AWS CLI describe-events 命令或 ElastiCache API 操作 `DescribeEvents`，您可以轻松从日志中检索事件。

您可以选择使用 Amazon EventBridge 监控、摄取、转换和处理 ElastiCache 事件。了解有关 Amazon EventBridge 的更多信息（https://docs.aws.amazon.com/eventbridge/latest/userguide/）。

查看 ElastiCache 事件（控制台）

要使用 ElastiCache 控制台查看事件，请执行以下操作：

1. 登录 AWS Management Console 并打开 ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 要查看所有可用事件的列表，请在导航窗格中选择 Events (事件)。
3. 在事件屏幕上，列表的每一行表示一个事件，并显示事件源、事件类型、事件的 GMT 时间及事件的描述。通过使用 Filter，您可以指定是要查看事件列表中的所有事件，还是仅查看特定类型的事件。

查看 ElastiCache 事件（AWS CLI）

要使用 AWS CLI 生成 ElastiCache 事件的列表，请使用命令 describe-events。您可以使用可选参数来控制所列事件的类型、所列事件的时间范围、要列出的事件的最大数目等。

以下代码列出最多 40 个无服务器缓存事件。

```
aws elasticache describe-events --source-type serverless-cache --max-items 40
```

以下代码列出了过去 24 小时（1440 分钟）内的所有无服务器缓存事件。

```
aws elasticache describe-events --source-type serverless-cache --duration 1440
```

无服务器事件

此部分记录了您可能收到的有关无服务器缓存的不同类型的事件。

无服务器缓存创建事件

Detail-Type	描述	单位	来源	消息
缓存已创建	缓存 arn	创建	serverless-cache	缓存 <cache-name> 已创建，可供使用。
缓存已创建	缓存 arn 快照 arn	创建	serverless-cache	缓存 <cache-name> 已创建，并且已从快照中还原数据。您的缓存已就绪，可供使用。
缓存创建失败	缓存 arn	失败	serverless-cache	缓存 <cache-name> 的创建失败。可用 IP 地址不足，无法创建 VPC 端点。
缓存创建失败	缓存 arn	失败	serverless-cache	缓存 <cache-name> 的创建失败。请求中提供的子网无效。
缓存创建失败	缓存 arn	失败	serverless-cache	缓存 <cache-name> 的创建失败。已达到创建 VPC 端点的配额限制。
缓存创建失败	缓存 arn	失败	serverless-cache	缓存 <cache-name> 的创建失败。您无权创建 VPC 端点。
缓存创建失败	缓存 arn	失败	serverless-cache	缓存 <cache-name> 的创建失败。用户组

Detail-Type	描述	单位	来源	消息
				<user-group-name> 中存在具有不兼容的 Redis 版本的用户。
缓存创建失败	缓存 arn 缓存快照 arn	失败	serverless-cache	缓存 <cache-name> 的创建失败。提供的用户组 <user-group-name> 不存在。
缓存创建失败	缓存 arn	失败	serverless-cache	缓存 <cache-name> 的创建失败。从快照还原数据失败，因为 <reason>。 失败原因： • 无法从 S3 检索文件。 • 预期的 md5 与实际的 md5 不匹配。 • 提供的 RDB 文件的版本不受支持。

无服务器缓存更新事件

Detail-Type	资源列表	类别	来源	消息
缓存已更新	缓存 arn	配置更改	serverless-cache	缓存 <cache-name> 的

Detail-Type	资源列表	类别	来源	消息
				SecurityGroups 已更新。
缓存已更新	缓存 arn	配置更改	serverless-cache	缓存 <cache-name> 的标签已更新。
缓存更新失败	缓存 arn	配置更改	serverless-cache	缓存 <cache-name> 的更新失败。用户组 <user-group-name> 中存在具有不兼容的 Redis 版本的用户。
缓存更新失败	缓存 arn	配置更改	serverless-cache	缓存 <cache-name> 的更新失败。SecurityGroups 更新失败。
缓存更新失败	缓存 arn	配置更改	serverless-cache	缓存 <cache-name> 的更新失败。由于权限不足，SecurityGroups 更新失败。
缓存更新失败	缓存 arn	配置更改	serverless-cache	缓存 <cache-name> 的更新失败。SecurityGroups 更新失败，因为 SecurityGroups 无效。

无服务器缓存删除事件

Detail-Type	资源列表	类别	来源	消息
缓存已删除	缓存 arn	删除	serverless-cache	缓存 <cache-name> 已删除。

无服务器缓存使用限制事件

Detail-Type	描述	单位	来源	消息
缓存已更新	缓存 arn	配置更改	serverless-cache	限制对缓存 <cache-name> 的更新。
即将达到缓存限制	缓存 arn	通知	serverless-cache	槽 <X> 使用的容量超过每个槽 32 GB 限制的 <Y> %。例如，槽 10 使用的容量超过每个槽 32 GB 限制的 90%。
缓存更新失败	缓存 arn	失败	serverless-cache	由于缓存 <cache-name> 已删除，对缓存的限制更新失败。
缓存更新失败	缓存 arn	失败	serverless-cache	由于配置无效，对缓存 <cache-name> 的限制更新失败。
缓存更新失败	缓存 arn	失败	serverless-cache	对缓存 <cache-name> 的有限更新失败，因

Detail-Type	描述	单位	来源	消息
				为当前缓存的数据超过了新的限制。在应用限制之前，请刷新数据。

无服务器缓存快照事件

Detail-Type	Resources-list	类别	来源	消息
快照已创建	缓存 arn 快照 arn	创建	serverless-cache-snapshot	已为缓存 <cache-name> 创建快照 <snapshot-name>。
快照创建失败	缓存 arn 快照 arn	失败	serverless-cache-snapshot	未能为缓存 <cache-name> 创建快照。使用客户自主管理型密钥 <key-id> 创建快照 <snapshot-name> 失败，因为 <reason>。 失败原因消息： • 因为客户自主管理型密钥已禁用 • 因为找不到客户自主管理型密钥

Detail-Type	Resources-list	类别	来源	消息
				• 因为请求超时
快照创建失败	缓存 arn 快照 arn	失败	serverless-cache-snapshot	未能为缓存 <cache-name> 创建快照。未能创建快照 <snapshot-name>，因为 <reason>。 默认原因： • 因为内部错误
快照导出失败	快照 arn	失败	serverless-cache-snapshot	未能为缓存 <cache-name> 导出快照。无法将快照导出到存储桶 %s，因为 ElastiCache 不具有存储桶权限。
快照导出失败	快照 arn	失败	serverless-cache-snapshot	未能为缓存 <cache-name> 导出快照。无法将快照导出到存储桶“%s”，因为存储桶中已存在同名对象。

Detail-Type	Resources-list	类别	来源	消息
快照导出失败	快照 arn	失败	serverless-cache-snapshot	未能为缓存 <cache-name> 导出快照。无法将快照导出到存储桶“%s”，因为存储桶所有者账户 ID 已更改。
快照导出失败	快照 arn	失败	serverless-cache-snapshot	未能为缓存 <cache-name> 导出快照。无法将快照导出到存储桶“%s”，因为 S3 存储桶不可访问。
快照导出失败	快照 arn	失败	serverless-cache-snapshot	未能为缓存 <cache-name> 导出快照。无法将快照导出到存储桶“%s”，因为存储桶不可访问。
快照导出失败	快照 arn	失败	serverless-cache-snapshot	未能为缓存 <cache-name> 导出快照。无法将快照导出到存储桶“%s”，因为存储桶不存在。

Detail-Type	Resources-list	类别	来源	消息
快照导出失败	快照 arn	失败	serverless-cache-snapshot	未能为缓存 <cache-name> 导出快照。无法使用源快照客户自主管理型密钥 %s 将快照导出到存储桶“%s”，因为 <reason>。
快照导出失败	快照 arn	失败	serverless-cache-snapshot	未能为缓存 <cache-name> 导出快照。无法将快照导出到存储桶“%s”。
快照复制失败	快照 arn-1 快照 arn-2	失败	serverless-cache-snapshot	未能复制快照 <snapshot-name>。无法使用源快照客户自主管理型密钥 <key-id> 将快照“%s”复制到快照“%s”，因为 <reason-name>。
快照复制失败	快照 arn-1 快照 arn-2	失败	serverless-cache-snapshot	未能复制快照 <snapshot-name>。无法将快照“%s”复制到快照“%s”（使用目标快照客户自主管理型密钥“%s”“%s”）。

自行设计集群的指标和事件

此部分介绍在使用自行设计的集群时，预计您可能会看到的指标、事件和日志。

主题

自行设计集群的指标

当您自行设计集群时，ElastiCache 会在各个节点级别发布指标，包括主机级别的指标和缓存指标。

有关主机级别指标的更多信息，请参阅主机级指标。

有关节点级别指标的更多信息，请参阅Redis 的指标。

自行设计集群的事件

ElastiCache 会记录与您的自行设计缓存相关的事件。使用自行设计的集群时，您可以在 ElastiCache 控制台中、使用 AWS CLI 或使用 Amazon Simple Notification Service（SNS）查看集群的事件。自行设计集群的事件不会发布到 Amazon EventBridge。

自行设计集群的事件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。通过使用 ElastiCache 控制台、AWS CLI describe-events 命令或 ElastiCache API 操作 DescribeEvents，您可以轻松从日志中检索事件。

查看 ElastiCache 事件（控制台）

以下过程演示了使用 ElastiCache 控制台查看事件。

使用 ElastiCache 控制台查看事件

1. 登录 AWS Management Console 并打开 ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。

2. 要查看所有可用事件的列表，请在导航窗格中选择 Events (事件)。
3. 在事件屏幕上，列表的每一行表示一个事件，并显示事件源、事件类型、事件的 GMT 时间及事件的描述。通过使用 Filter，您可以指定是要查看事件列表中的所有事件，还是仅查看特定类型的事件。

查看 ElastiCache 事件 (AWS CLI)

要使用 AWS CLI 生成 ElastiCache 事件的列表，请使用命令 describe-events。您可以使用可选参数来控制所列事件的类型、所列事件的时间范围、要列出的事件的最大数目等。

以下代码列出最多 40 个自行设计集群的事件。

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
```

以下代码列出了过去 24 小时（1440 分钟）内自行设计缓存的所有事件。

```
aws elasticache describe-events --source-type cache-cluster --duration 1440
```

自行设计集群的事件

本部分包含对于自行设计的集群，预计您会收到的事件列表。

以下 ElastiCache 事件会触发 Amazon SNS 通知。有关事件详细信息的信息，请参阅 查看 ElastiCache 事件。

事件名称	消息	描述
ElastiCache:AddCacheNodeComplete	`ElastiCache:AddCacheNodeComplete : cache-cluster`	缓存节点已添加到缓存群集，并准备就绪，可供可用。
由于空闲 IP 地址不足导致的 ElastiCache:AddCacheNodeFailed	`ElastiCache:AddCacheNodeFailed : cluster-name`	因为没有足够的可用 IP 地址，所以无法添加缓存节点。
ElastiCache:CacheClusterParametersChanged	`ElastiCache:CacheClusterParametersChanged : cluster-name`	一个或多个缓存群集参数已更改。

事件名称	消息	描述
ElastiCache:CacheClusterProvisioningComplete	`ElastiCache:CacheClusterProvisioningComplete` *`cluster-name-0001-005`*	缓存群集预配置已完成，并且缓存群集中的缓存节点准备就绪，可供使用。
由于不兼容网络状态导致的 ElastiCache:CacheClusterProvisioningFailed	`ElastiCache:CacheClusterProvisioningFailed :` *`cluster-name`*	尝试将新缓存群集启动到不存在的 Virtual Private Cloud (VPC) 中。
ElastiCache:CacheClusterScalingComplete	`CacheClusterScalingComplete :` *`cluster-name`*	已成功完成缓存群集扩展。
ElastiCache:CacheClusterScalingFailed	ElastiCache:CacheClusterScalingFailed : *`cluster-name`*	对缓存群集的纵向扩展操作已失败。
ElastiCache:CacheClusterSecurityGroupModified	`ElastiCache:CacheClusterSecurityGroupModified :` *`cluster-name`*	发生下列事件之一： • 已修改授权用于缓存群集的缓存安全组列表。 • 已在与缓存群集相关的任何缓存安全组上授权一个或多个新的 EC2 安全组。 • 已从与缓存群集相关的缓存安全组中撤销一个或多个 EC2 安全组。

事件名称	消息	描述
ElastiCache:CacheNodeReplaceStarted	`ElastiCache:CacheNodeReplaceStarted : cluster-name`	ElastiCache 已检测到运行缓存节点的主机性能下降或无法访问，并已开始缓存节点的替换工作。 **Note** 针对替换之缓存节点的 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，某些缓存客户端库可能停止使用缓存节点，即使在 ElastiCache 已替换缓存节点之后亦是如此；在这种情况下，应用程序应该在此事件发生时刷新服务器列表。

事件名称	消息	描述
ElastiCache:CacheNodeReplaceComplete	`ElastiCache:CacheNodeReplaceComplete : cluster-name`	ElastiCache 已检测到运行缓存节点的主机性能下降或无法访问，并已完成缓存节点的替换工作。 **Note** 针对替换之缓存节点的 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，某些缓存客户端库可能停止使用缓存节点，即使在 ElastiCache 已替换缓存节点之后亦是如此；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。
ElastiCache:CacheNodesRebooted	`ElastiCache:CacheNodesRebooted : cluster-name`	一个或多个缓存节点已重启。 消息 (Memcached)：`"Cache node %s shutdown"` ，然后是第二条消息：`"Cache node %s restarted"`
ElastiCache:CertificateRenewalComplete（仅限 Redis）	`ElastiCache:CertificateRenewalComplete`	已成功续订 Amazon CA 证书。

事件名称	消息	描述
ElastiCache:CreateReplicationGroupComplete	ElastiCache:CreateReplicationGroupComplete : *cluster-name*	已成功创建复制组。
ElastiCache:DeleteCacheClusterComplete	ElastiCache:DeleteCacheClusterComplete : *cluster-name*	已完成缓存群集和所有关联缓存节点的删除工作。
ElastiCache:FailoverComplete（仅限 Redis）	ElastiCache:FailoverComplete : *mycluster*	已成功故障转移至副本节点。
ElastiCache:ReplicationGroupIncreaseReplicaCountFinished	ElastiCache:ReplicationGroupIncreaseReplicaCountFinished : *cluster-name-0001-005*	已增加集群中的副本数量。
ElastiCache:ReplicationGroupIncreaseReplicaCountStarted	ElastiCache:ReplicationGroupIncreaseReplicaCountStarted : *cluster-name-0003-004*	已开始向集群添加副本的过程。
ElastiCache:NodeReplacementCanceled	ElastiCache:NodeReplacementCanceled : *cluster-name*	计划替换的集群中的节点不再计划替换。
ElastiCache:NodeReplacementRescheduled	ElastiCache:NodeReplacementRescheduled : *cluster-name*	之前计划替换的集群中的节点已计划在通知中所述的新时段内替换。 有关您可以执行的操作的信息，请参阅 替换节点。

事件名称	消息	描述
ElastiCache:NodeReplacementScheduled	`ElastiCache:NodeReplacementScheduled : cluster-name`	您集群中的节点计划在通知所述的时段内替换。 有关您可以执行的操作的信息，请参阅 替换节点。
ElastiCache:RemoveCacheNodeComplete	`ElastiCache:RemoveCacheNodeComplete : cluster-name`	缓存节点已从缓存群集中移除。
ElastiCache:ReplicationGroupScalingComplete	`ElastiCache:ReplicationGroupScalingComplete : cluster-name`	已成功完成对复制组的纵向扩展操作。
ElastiCache:ReplicationGroupScalingFailed	`"Failed applying modification to cache node type to %s."`	对复制组的纵向扩展操作失败。
ElastiCache:ServiceUpdateAvailableForNode	`"Service update is available for cache node %s."`	自助服务更新可用于节点。
ElastiCache:SnapshotComplete（仅限 Redis）	`ElastiCache:SnapshotComplete : cluster-name`	缓存快照已成功完成。
ElastiCache:SnapshotFailed（仅限 Redis）	`SnapshotFailed : cluster-name`	缓存快照失败。有关失败原因的详细信息，请参阅该集群的缓存事件。 要对快照加以说明，请参阅 `DescribeSnapshots`，状态将是 `failed`。

日志传输

> **Note**
>
> 使用 6.0 和更高版本引擎的 Redis 缓存集群和复制组支持 Redis 慢日志。
> 使用 6.2 和更高版本引擎的 Redis 缓存集群和复制组支持 Redis 引擎日志。

日志传输可让您将 Redis 慢日志或 Redis 引擎日志流式传输到以下两个目的地之一：

- Amazon Data Firehose
- Amazon CloudWatch 日志

使用 ElastiCache API 创建或修改集群时，您可以启用和配置日志传输。每个日志条目将以两种格式之一传输到指定的目的地：JSON 或 TEXT。

将定期从 Redis 引擎中检索固定数量的慢日志条目。根据针对引擎参数 `slowlog-max-len` 指定的值，可能不会将其他慢日志条目传送到目的地。

您可以随时使用 AWS 控制台或其中一个修改 API（或）选择更改传输配置或禁用日志传输modify-replication-group。modify-cache-cluster

您必须为所有日志传输修改设置 `apply-immediately` 参数。

> **Note**
>
> 启用 CloudWatch 日志传输后，即使日志直接传送到 Amazon Data Firehose，也会收取亚马逊日志费用。有关更多信息，请参阅 Amazon CloudWatch 定价中的 “销售日志” 部分。

慢日志条目的内容

ElastiCache 适用于 Redis 的慢速日志包含以下信息：

- CacheClusterId— 缓存集群的 ID
- CacheNodeId— 缓存节点的 ID
- Id – 每个慢日志条目的唯一渐进式标识符
- Timestamp – 处理已录入的命令时的 Unix 时间戳

- Duration – 其执行所需的时间（以微秒为单位）
- Command – 客户端使用的命令。例如，`set foo bar`其中`foo`是键，其中`bar`是值。 ElastiCache for Redis 将实际的密钥名称和值替换`(2 more arguments)`为以避免暴露敏感数据。
- ClientAddress— 客户端 IP 地址和端口
- ClientName— 客户端名称（如果通过`CLIENT SETNAME`命令设置）

引擎日志条目的内容

ElastiCache 适用于 Redis 的引擎日志包含以下信息：

- CacheClusterId— 缓存集群的 ID
- CacheNodeId— 缓存节点的 ID
- 日志级别 — LogLevel 可以是以下任一项：`VERBOSE("-")`、`NOTICE("*")`、`WARNING("#")`。
- Time – 日志消息的 UTC 时间。时间采用以下格式：`"DD MMM YYYY hh:mm:ss.ms UTC"`
- Role – 发出日志的节点的角色。它可以是以下值之一：“M”表示主角色，“S”表示副本，“C”表示在 RDB/AOF 上工作的写入器子进程，“X”表示 Sentinel。
- Message - Redis 引擎日志消息。

配置日志记录的权限

您需要在您的 IAM 用户/角色策略中包含以下 IAM 权限：

- `logs:CreateLogDelivery`
- `logs:UpdateLogDelivery`
- `logs:DeleteLogDelivery`
- `logs:GetLogDelivery`
- `logs:ListLogDeliveries`

有关更多信息，请参阅访问管理概览：权限和策略。

日志类型和日志格式规范

慢日志

慢日志支持 JSON 和 TEXT 两种格式

以下示例演示了 JSON 格式：

```
{
  "CacheClusterId": "logslowxxxxmsxj",
  "CacheNodeId": "0001",
  "Id": 296,
  "Timestamp": 1605631822,
  "Duration (us)": 0,
  "Command": "GET ... (1 more arguments)",
  "ClientAddress": "192.168.12.104:55452",
  "ClientName": "logslowxxxxmsxj##"
}
```

以下示例演示了 TEXT 格式：

```
logslowxxxxmsxj,0001,1605631822,30,GET ... (1 more
 arguments),192.168.12.104:55452,logslowxxxxmsxj##
```

引擎日志

引擎日志支持 JSON 和 TEXT 两种格式

以下示例演示了 JSON 格式：

```
{
  "CacheClusterId": "xxxxxxxxxzy-engine-log-test",
  "CacheNodeId": "0001",
  "LogLevel": "VERBOSE",
  "Role": "M",
  "Time": "12 Nov 2020 01:28:57.994 UTC",
  "Message": "Replica is waiting for next BGSAVE before synchronizing with the primary.
 Check back later"
}
```

以下示例演示了 TEXT 格式：

```
xxxxxxxxxxxzy-engine-log-test/0001:M 29 Oct 2020 20:12:20.499 UTC * A slow-running Lua
 script detected that is still in execution after 10000 milliseconds.
```

ElastiCache 登录目的地

本节介绍您可以为日志选择的 ElastiCache 日志记录目的地。每个部分都提供了有关配置目标类型日志记录的指导，以及有关特定于目标类型的任何行为的信息。配置日志目标后，您可以向日志记录配置提供其规格以开始向其登录。 ElastiCache

主题

- Amazon CloudWatch 日志
- Amazon Data Firehose

Amazon CloudWatch 日志

- 您可以指定一个 CloudWatch 日志日志组，用于传送日志。
- 来自多个 Redis 集群和复制组的日志可以发送到同一个日志组。
- 将为缓存集群或复制组中的每个节点创建一个新的日志流，并将日志发送到各自的日志流。日志流名称将使用以下格式：`elasticache/${engine-name}/${cache-cluster-id}/${cache-node-id}/${log-type}`

向日志发布 CloudWatch 日志的权限

您必须具有以下权限设置才能配置 ElastiCache Redis 才能将日志发送到 CloudWatch 日志日志组：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "ElastiCacheLogging"
```

```
        },
        {
            "Sid": "ElastiCacheLoggingCWL",
            "Action": [
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

有关更多信息，请参阅发送到日志的 CloudWatch 日志。

Amazon Data Firehose

- 您可以指定一个 Firehose 传输流，将日志传送到哪里。
- 来自多个 Redis 集群和复制组的日志可以传输到同一个传输流。
- 缓存集群或复制组中每个节点的日志将传输到同一个传输流。您可以根据各个日志消息中包括的 `cache-cluster-id` 和 `cache-node-id` 来区分不同缓存节点的日志消息。
- 亚太地区（大阪）区域目前不支持向 Firehose 传送日志。

向 Firehose 发布日志的权限

您必须具有以下权限才能配置 ElastiCache Redis 才能向 Amazon Kinesis Data Firehose 传输流发送日志。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
```

```
                "logs:ListLogDeliveries"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow",
            "Sid": "ElastiCacheLogging"
        },
        {
            "Sid": "ElastiCacheLoggingFHSLR",
            "Action": [
                "iam:CreateServiceLinkedRole"
            ],
            "Resource": "*",
            "Effect": "Allow"
        },
        {
            "Sid": "ElastiCacheLoggingFH",
            "Action": [
                "firehose:TagDeliveryStream"
            ],
            "Resource": "Amazon Kinesis Data Firehose delivery stream ARN",
            "Effect": "Allow"
        }
    ]
}
```

使用控制台指定日志传输

通过使用 AWS Management Console，您可以按照 创建 Redis（已禁用集群模式）集群（控制台） 中的步骤创建 Redis（已禁用集群模式）集群，或按照 创建 Redis（已启用集群模式）集群（控制台） 中的步骤创建 Redis（已启用集群模式）集群。在上述任何一种情况下，您都可以通过执行以下操作来配置日志传输；

1. 在 Advanced Redis settings（高级 Redis 设置）中，选择 Logs（日志），然后查看 Slow logs（慢日志）或 Engine logs （引擎日志）。
2. 在 Log format（日志格式）下，选择 Text（文本）或 JSON。
3. 在 Destination Type（目标类型）下，选择 CloudWatch Logs 或 Kinesis Firehose。
4. 在 Log destination（日志目标）下，选择 Create new（创建新的），然后输入您的 Amazon S3 存储桶名称、CloudWatchLogs 日志组名称或 Kinesis Data Firehose 流名称，或选择 Select

existing（选择现有），然后选择您的 CloudWatch Logs 日志组名称或您的 Kinesis Data Firehose 流名称，

修改集群时：

您可以选择启用/禁用日志传输，也可以更改目的地类型、格式或目的地：

1. 登录到控制台并打开 ElastiCache 控制台（https://console.aws.amazon.com/elasticache/）。
2. 从导航窗格中，选择 Redis clusters（Redis 集群）。
3. 从集群列表中，选择要修改的集群。选择 Cluster name（集群名称）而不是旁边的复选框。
4. 在 Cluster name（集群名称）页面上，选择 Logs（日志）选项卡。
5. 要启用/禁用慢日志，请选择 Enable slow logs（启用慢日志）或 Disable slow logs（禁用慢日志）。
6. 要启用/禁用引擎日志，请选择 Enable engine logs（启用引擎日志）或 Disable engine logs（禁用引擎日志）。
7. 要更改配置，请选择 Modify slow logs（修改慢日志）或 Modify engine logs（修改引擎日志）：
   - 在 Destination Type（目标类型）下，选择 CloudWatch Logs 或 Kinesis Firehose。
   - 在 Log destination（日志目标）下，选择 Create new（创建新的），然后输入您的 CloudWatch Logs 日志组名称或 Kinesis Data Firehose 流名称。或者选择 Select existing（选择现有），然后选择您的 CloudWatch Logs 日志组名称或 Kinesis Data Firehose 流名称。

使用指定日志传输 AWS CLI

慢日志

创建一个向日志传送缓慢 CloudWatch 日志的复制组。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
    --replication-group-id test-slow-log \
    --replication-group-description test-slow-log \
    --engine redis \
    --cache-node-type cache.r5.large \
    --num-cache-clusters 2 \
    --log-delivery-configurations '{
        "LogType":"slow-log",
```

```
        "DestinationType":"cloudwatch-logs",
        "DestinationDetails":{
          "CloudWatchLogsDetails":{
            "LogGroup":"my-log-group"
          }
        },
        "LogFormat":"json"
      }'
```

对于 Windows：

```
aws elasticache create-replication-group ^
    --replication-group-id test-slow-log ^
    --replication-group-description test-slow-log ^
    --engine redis ^
    --cache-node-type cache.r5.large ^
    --num-cache-clusters 2 ^
    --log-delivery-configurations '{
        "LogType":"slow-log",
        "DestinationType":"cloudwatch-logs",
        "DestinationDetails":{
          "CloudWatchLogsDetails":{
            "LogGroup":"my-log-group"
          }
        },
        "LogFormat":"json"
      }'
```

修改复制组以将慢速日志传送到 CloudWatch 日志

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
      "LogType":"slow-log",
      "DestinationType":"cloudwatch-logs",
      "DestinationDetails":{
        "CloudWatchLogsDetails":{

          "LogGroup":"my-log-group"
```

```
        }
      },
      "LogFormat":"json"
    }'
```

对于 Windows：

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
      "LogType":"slow-log",
      "DestinationType":"cloudwatch-logs",
      "DestinationDetails":{
        "CloudWatchLogsDetails":{
          "LogGroup":"my-log-group"
        }
      },
      "LogFormat":"json"
    }'
```

修改复制组以禁用慢日志传递

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
      "LogType":"slow-log",
      "Enabled":false
    }'
```

对于 Windows：

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
```

```
      "LogType":"slow-log",
      "Enabled":false
    }'
```

引擎日志

创建将引擎日志传送到 CloudWatch 日志的复制组。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-replication-group \
    --replication-group-id test-slow-log \
    --replication-group-description test-slow-log \
    --engine redis \
    --cache-node-type cache.r5.large \
    --num-cache-clusters 2 \
    --log-delivery-configurations '{
        "LogType":"engine-log",
        "DestinationType":"cloudwatch-logs",
        "DestinationDetails":{
          "CloudWatchLogsDetails":{
            "LogGroup":"my-log-group"
          }
        },
        "LogFormat":"json"
      }'
```

对于 Windows：

```
aws elasticache create-replication-group ^
    --replication-group-id test-slow-log ^
    --replication-group-description test-slow-log ^
    --engine redis ^
    --cache-node-type cache.r5.large ^
    --num-cache-clusters 2 ^
    --log-delivery-configurations '{
        "LogType":"engine-log",
        "DestinationType":"cloudwatch-logs",
        "DestinationDetails":{
          "CloudWatchLogsDetails":{
            "LogGroup":"my-log-group"
          }
        },
```

```
        "LogFormat":"json"
      }'
```

修改复制组以将引擎日志传送到 Firehose

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
      "LogType":"engine-log",
      "DestinationType":"kinesis-firehose",
      "DestinationDetails":{
      "KinesisFirehoseDetails":{
         "DeliveryStream":"test"
       }
     },
      "LogFormat":"json"
    }'
```

对于 Windows：

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
      "LogType":"engine-log",
      "DestinationType":"kinesis-firehose",
      "DestinationDetails":{
      "KinesisFirehoseDetails":{
         "DeliveryStream":"test"
       }
      },
      "LogFormat":"json"
    }'
```

修改复制组以切换到引擎格式

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
       "LogType":"engine-log",
       "LogFormat":"json"
    }'
```

对于 Windows：

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
       "LogType":"engine-log",
       "LogFormat":"json"
    }'
```

修改复制组以禁用引擎日志传递

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-replication-group \
    --replication-group-id test-slow-log \
    --apply-immediately \
    --log-delivery-configurations '
    {
      "LogType":"engine-log",
      "Enabled":false
    }'
```

对于 Windows：

```
aws elasticache modify-replication-group ^
    --replication-group-id test-slow-log ^
    --apply-immediately ^
    --log-delivery-configurations '
    {
      "LogType":"engine-log",
      "Enabled":false
```

```
    }'
```

使用 CloudWatch 指标监控使用情况

ElastiCache 提供可用于监控集群的指标。您可以通过 CloudWatch 访问这些指标。有关 CloudWatch 的信息，请参阅 Amazon CloudWatch 文档。

ElastiCache 提供主机层面级指标（例如 CPU 使用率）和特定于缓存引擎软件的指标（例如缓存获取次数和缓存未命中数）。这些指标每隔 60 秒对每个缓存节点进行测量并发布结果。

> **Important**
>
> 您应考虑对特定重要指标设置 CloudWatch 告警，以便在缓存集群性能开始下降时收到通知。有关更多信息，请参阅本指南中的 应监控哪些指标？。

主题

主机级指标

AWS/ElastiCache 命名空间包含各个缓存节点的以下主机级指标。这些指标每隔 60 秒对每个缓存节点进行测量并发布结果。

另请参阅

- Redis 的指标

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，我们建议您监控有 4 个或更多 vCPU 的节点的 EngineCPUUtilization 指标。	百分比

指标	描述	单位
CPUCreditBalance	实例自启动后已累积获得的 CPU 积分数。对于 T2 标准，CPUCreditBalance 还包含已累积的启动积分数。 在获得积分后，积分将在积分余额中累积；在花费积分后，将从积分余额中扣除积分。积分余额具有最大值限制，这是由实例大小决定的。在达到限制后，将丢弃获得的任何新积分。对于 T2 标准，启动积分不计入限制。 实例可以花费 CPUCreditBalance 中的积分，以便突增到基准 CPU 使用率以上。 CPU 信用指标仅每 5 分钟提供一次。 这些指标对 T2 可突增性能实例不可用。	积分（vCPU 分钟）
CPUCreditUsage	实例为保持 CPU 使用率而花费的 CPU 积分数。一个 CPU 积分等于一个 vCPU 按 100% 利用率运行一分钟，或者 vCPU、利用率和时间的等效组合（例如， 一个 vCPU 按 50% 利用率运行两分钟，或者两个 vCPU 按 25% 利用率运行两分钟）。 CPU 信用指标仅每 5 分钟提供一次。如果您指定一个大于五分钟的时间段，请使用“Sum（总和）”统计数据，而非“Average（平均值）”统计数据。 这些指标对 T2 可突增性能实例不可用。	积分（vCPU 分钟）
FreeableMemory	主机上可用的闲置内存量。这是从操作系统报告为空闲的 RAM、缓冲区和缓存中派生出来的。	字节
NetworkBytesIn	主机已从网络读取的字节数。	字节

指标	描述	单位
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排队或丢弃的数据包的数量。	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丢弃的数据包的数量。这可能会导致进出实例的流量丢失数据包。	计数
NetworkBandwidthOutAllowanceExceeded	因出站聚合带宽超过实例的最大值而排队或丢弃的数据包的数量。	计数
NetworkPacketsPerSecondAllowanceExceeded	因每秒双向数据包数量超过实例的最大值而排队或丢弃的数据包数量。	计数
NetworkMaxBytesIn	每分钟内接收的最大突增字节数。	字节
NetworkMaxBytesOut	每分钟内传输的最大突增字节数。	字节
NetworkMaxPacketsIn	每分钟内接收的最大突增数据包数。	计数
NetworkMaxPacketsOut	每分钟内传输的最大突增数据包数。	计数
SwapUsage	主机上的交换区使用量。	字节

Redis 的指标

AWS/ElastiCache 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。每项指标均是按照缓存节点级计算的。

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

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
BytesUsedForCache	Redis 为所有目的（包括数据集、缓冲区等）分配的字节的总数。	字节
	Dimension: Tier=Memory （对于使用数据分层功能的 Redis 集群）：内存中用于缓存的总字节数。这是 Redis INFO 的 used_memory 统计数据的值。	字节
	Dimension: Tier=SSD （对于使用数据分层功能的 Redis 集群）：SSD 中用于缓存的总字节数。	字节
BytesReadFromDisk	每分钟从磁盘读取的总字节数。仅支持使用数据分层功能的集群。	字节
BytesWrittenToDisk	每分钟写入磁盘的总字节数。仅支持使用数据分层功能的集群。	字节

指标	描述	单位
CacheHits	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
CacheMisses	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
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
CacheHitRate	指示 Redis 实例的使用效率。如果缓存比率低于 0.8 左右，则意味着大量的密钥被移出、过期或不存在。这是使用 cache_hits 和 cache_misses 统计数据按以下方式计算的：cache_hits /(cache_hits + cache_misses) 。	百分比
ChannelAuthorizationFailures	用户访问其无权限访问的通道的失败尝试次数。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。我们建议为此指标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。ElastiCache 在每种情况下，都使用两到四个连接来监视集群。这是根据 Redis INFO 中的 connected_clients 统计数据得出的。	计数
CurrItems	缓存中的项目数。此值根据以下方法获得的 Redis keyspace 统计数据得出：计算整个键空间中所有键的总和。	计数
	Dimension: Tier=Memory （对于使用数据分层功能的集群）。内存中的项目数。	计数

指标	描述	单位
	Dimension: Tier=SSD （固态硬盘）（对于使用数据分层功能的 Redis 集群）。SSD 中的项目数。	计数
CurrVolatileItems	所有数据库中具有 ttl 集的键的总数。此值根据 Redis expires 统计数据得出，方式是将在整个键空间内有 ttl 集的所有键相加。	计数
DatabaseCapacityUsagePercentage	集群的总数据容量中正在使用的百分比。 在数据分层实例上，该指标的计算方式为 (used_memory - mem_not_counted_for_evict + SSD used) / (maxmemory + SSD total capacity) 、位置used_memory 和maxmemory 取自 Redis INFO。 在所有其他情况下，使用计算指标used_memory/maxmemory 。	百分比
DatabaseCapacityUsageCountedForEvictPercentage	集群的总数据容量中正在使用的百分比（不含用于开销和 COB 的内存）。该指标的计算方式如下： used_memory - mem_not_counted_for_evict/maxmemory 在数据分层实例上，该指标的计算方式如下： (used_memory + SSD used) / (maxmemory + SSD total capacity) 其中，used_memory 和 maxmemory 取自 Redis 信息	百分比

指标	描述	单位
DatabaseMemoryUsagePercentage	集群中正在使用的内存的百分比。这是使用 used_memory/maxmemory 从 Redis INFO 计算得来的。	百分比
DatabaseMemoryUsageCountedForEvictPercentage	集群中正在使用的内存的百分比（不含用于开销和 COB 的内存）。这是使用 used_memory-mem_not_counted_for_evict/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。副本不会使密钥过期，而是等待主节点使密钥过期。当主节点使密钥过期（或由于 LRU 而将其逐出）时，它将合成一个 DEL 命令，该命令将传送到所有副本。因此，对于副本节点，DB0AverageTTL 为 0，因为它们不会使密钥过期，因而不会跟踪 TTL。	毫秒

指标	描述	单位
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 ElastiCache 主机上，后台进程监视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，我们建议对于具有不超过两个 vCPU 的主机，还需要监控 CPUUtilization 指标。	百分比
Evictions	由于 maxmemory 限制而被驱逐的密钥数。这是根据 Redis INFO 中的 evicted_keys 统计数据得出的。	计数

指标	描述	单位
GlobalDatastoreReplicationLag	此为辅助区域的主节点与主区域的主节点之间的滞后。对于已启用集群模式的 Redis，滞后表示分片之间的最大延迟。	秒
IamAuthenticationExpirations	已过期的经过 IAM 身份验证的 Redis 连接总数。您可以在用户指南中找到有关 使用 IAM 进行身份验证 的更多信息。	计数
IamAuthenticationThrottling	受限的经过 IAM 身份验证的 Redis AUTH 或 HELLO 请求的总数。您可以在用户指南中找到有关 使用 IAM 进行身份验证 的更多信息。	计数
IsMaster	指示节点是否为当前分片/集群的主节点。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
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
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

指标	描述	单位
NewConnections	在此期间，服务器接受的连接总数。这是根据 Redis INFO 中的 total_connections_received 统计数据得出的。 **Note** 如果您使用 ElastiCache 的是 Redis 版本 5 或更低版本，则使用此指标报告的两到四个连接 ElastiCache 来监控集群。但是，在 Redis 版本 6 或更高版本中使用 ElastiCache 时， ElastiCache 用于监控集群的连接不包含在此指标中。	计数
NumItemsReadFromDisk	每分钟从磁盘检索的项目总数。仅支持使用数据分层功能的集群。	计数
NumItemsWrittenToDisk	每分钟写入磁盘的项目总数。仅支持使用数据分层功能的集群。	计数
MasterLinkHealthStatus	此状态有两个值：0 或 1。值 0 表示 ElastiCache主节点中的数据与 EC2 上的 Redis 不同步。值为 1 表示数据已同步。要完成迁移，请使用 CompleteMigrationAPI 操作。	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。此指标代表复制组上的写入负载。这是根据 Redis INFO 中的 master_repl_offset 统计数据得出的。	字节

指标	描述	单位
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）。对于 Redis 引擎版本 5.0.6 和更高版本，滞后以毫秒计。	秒
SaveInProgress	只要背景保存（forked 或 forkless）在进行中，此二进制指标均返回 1，否则会返回 0。在快照和同步期间，通常使用背景保存进程。这些操作会导致性能下降。使用 SaveInProgress 指标，您可以诊断性能下降是否由背景保存进程造成。这是根据 Redis INFO 中的 rdb_bgsave_in_progress 统计数据得出的。	布尔值
TrafficManagementActive	指示 Redis 是否 ElastiCache 通过调整分配给传入命令、监控或复制的流量来主动管理流量。当发送到节点的命令多于 Redis 可以处理的命令时，流量就会受到管理，并用于保持引擎的稳定性和最佳运行状态。任何为 1 的数据点都可能表示节点对于所提供的工作负载而言规模过小。 **Note** 如果此指标持续处于活动状态，请评估集群以确定是否需要纵向扩展或横向扩展。相关指标包括 NetworkBandwidthOutAllowanceExceeded 和 EngineCPUUtilization 。	布尔值

EngineCPUUtilization 可用性

AWS 以下列出的区域适用于所有支持的节点类型。

区域	区域名称
us-east-2	美国东部（俄亥俄）
us-east-1	美国东部（弗吉尼亚州北部）
us-west-1	美国西部（加利福尼亚北部）
us-west-2	美国西部（俄勒冈）
ap-northeast-1	亚太地区（东京）
ap-northeast-2	亚太地区 (首尔)
ap-northeast-3	亚太地区（大阪）
ap-east-1	亚太地区（香港）
ap-south-1	亚太地区（孟买）
ap-southeast-1	亚太地区（新加坡）
ap-southeast-2	亚太地区（悉尼）
ap-southeast-3	亚太地区（雅加达）
ca-central-1	加拿大（中部）
cn-north-1	中国（北京）
cn-northwest-2	中国（宁夏）
me-south-1	中东（巴林）
eu-central-1	欧洲地区（法兰克福）
eu-west-1	欧洲地区（爱尔兰）
eu-west-2	欧洲地区（伦敦）
eu-west-3	欧洲地区（巴黎）

区域	区域名称
eu-south-1	欧洲地区（米兰）
af-south-1	非洲（开普敦）
eu-north-1	欧洲地区（斯德哥尔摩）
sa-east-1	南美洲（圣保罗）
us-gov-west-1	AWS GovCloud （美国西部）
us-gov-east-1	AWS GovCloud （美国东部）

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、这些命令消耗的总 CPU 时间以及每个命令执行所消耗的平均 CPU 时间。对于每种命令类型，都会添加以下行：`cmdstat_XXX: calls=XXX,usec=XXX,usec_per_call=XXX`。

下面列出的延迟指标是使用 Redis INFO 中的 commandstats 统计数据计算得出的。计算方式如下：`delta(usec)/delta(calls)`。`delta` 计算为一分钟内的差异。延迟定义为处理命令所花费 ElastiCache 的 CPU 时间。请注意，对于使用数据分层的集群，这些测量值并未包含从 SSD 提取项目所需的时间。

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
ClusterBasedCmds	基于集群的命令总数。此值根据 Redis commandstats 统计数据得出，方式将所有作用于集群的命令（cluster slot、cluster info 等）相加。	计数
ClusterBasedCmdsLatency	基于集群的命令的延迟。	微秒
EvalBasedCmds	基于 eval 的命令的命令总数。这是根据 Redis commandstats 统计数据通过计算 eval、evalsha 的总和得出的。	计数

指标	描述	单位
EvalBasedCmdsLatency	基于 Eval 的命令的延迟。	微秒
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
GeoSpatialBasedCmdsLatency	基于地理空间的命令的延迟。	微秒
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
GetTypeCmdsLatency	读取命令的延迟。	微秒
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
HashBasedCmdsLatency	基于哈希的命令的延迟。	微秒
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
HyperLogLogBasedCmdsLatency	HyperLogLog基于命令的延迟。	微秒

指标	描述	单位
JsonBasedCmds	JSON 命令的总数，包括读取和写入命令。此值根据以下方法获得的 Redis commandstats 统计数据得出：计算所有作用于 JSON 键的 JSON 命令的总和。	计数
JsonBasedCmdsLatency	所有 JSON 命令的延迟，包括读取和写入命令。	微秒
JsonBasedGetCmds	JSON 只读命令的总数。此值根据以下方法获得的 Redis commandstats 统计数据得出：计算所有作用于 JSON 键的 JSON 读取命令的总和。	计数
JsonBasedGetCmds延迟	JSON 只读命令的延迟。	微秒
JsonBasedSetCmds	JSON 写入命令的总数。此值根据以下方法获得的 Redis commandstats 统计数据得出：计算所有作用于 JSON 键的 JSON 写入命令的总和。	计数
JsonBasedSetCmds延迟	JSON 写入命令的延迟。	微秒
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
KeyBasedCmdsLatency	基于键的命令的延迟。	微秒
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
ListBasedCmdsLatency	基于列表的命令的延迟。	微秒

指标	描述	单位
NonKeyTypeCmds	不基于键的命令总数。此值根据以下方法获得的 Redis commandstats 统计数据得出：计算所有不作用于某个键的命令（acl、dbsize 或 info 等）的总和。	计数
NonKeyTypeCmds延迟	non-key-based 命令延迟。	微秒
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、ssubscribe、sunsubscribe、spublish、subscribe 和 unsubscribe。	计数
PubSubBasedCmdsLatency	PubSub-based 命令的延迟。	微秒
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
SetBasedCmdsLatency	基于集合的命令的延迟。	微秒
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
SetTypeCmdsLatency	写入命令的延迟。	微秒

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
SortedSetBasedCmdsLatency	基于排序的命令的延迟。	微秒
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
StringBasedCmdsLatency	基于字符串的命令的延迟。	微秒
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
StreamBasedCmdsLatency	基于流的命令的延迟。	微秒

应监控哪些指标？

通过以下 CloudWatch 指标可深入了解 ElastiCache 性能。在许多情况下，我们建议对这些指标设置 CloudWatch 告警，以便您可以在性能问题出现之前采取纠正措施。

监控指标

- CPU 利用率
- EngineCPUUtilization
- SwapUsage
- 移出
- 当前连接
- 内存
- Network
- 延迟
- 复制
- 流量管理

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。

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%。

您需要根据所使用的缓存节点中的核心数，来确定自己的阈值。如果超过此阈值，并且主要工作负载来自读取请求，则请通过添加只读副本来扩展缓存集群。如果主要工作负载来自写入请求，我们的建议取决于您的集群配置：

- Redis（已禁用集群模式）集群：使用更大的缓存实例类型进行纵向扩展。
- Redis（已启用集群模式）集群：添加更多分区，在更多主节点中分配写入工作负载。

> **Tip**
>
> Redis 用户可能能够使用向您报告有关 Redis 引擎核心的使用率百分比的 Redis 指标 `EngineCPUUtilization`，而不是使用主机级指标 `CPUUtilization`。要了解此指标在您的节点上是否可用并了解更多信息，请参阅 Redis 的指标。

对于有 4 个或更多 vCPU 的较大节点类型，您可能希望使用 `EngineCPUUtilization` 指标，该指标可以向您报告 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 Redis 的指标。

EngineCPUUtilization

对于有 4 个或更多 vCPU 的较大节点类型，您可能希望使用 `EngineCPUUtilization` 指标，该指标可以向您报告 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 Redis 的指标。

有关更多信息，请参阅使用 Amazon CloudWatch 监控 Amazon ElastiCache for Redis 的最佳实践的 CPU 部分。

SwapUsage

这是以字节为单位报告的主机级指标。有关更多信息，请参阅主机级指标。

如果 `FreeableMemory` CloudWatch 指标接近 0（即低于100MB）或 `SwapUsage` 指标大于 `FreeableMemory` 指标，则表示节点处于内存压力下。如果发生这种情况，请参阅以下主题：

- 确保具有用于创建 Redis 快照的足够内存
- 管理预留内存

移出

这是缓存引擎指标。我们建议您根据应用程序需求，为此指标确定自己的警报阈值。

当前连接

这是缓存引擎指标。我们建议您根据应用程序需求，为此指标确定自己的警报阈值。

当前连接的数量不断增加，可能表示应用程序出现问题；您需要调查应用程序行为以解决此问题。

有关更多信息，请参阅使用 Amazon CloudWatch 监控 Amazon ElastiCache for Redis 的最佳实践的连接部分。

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。

有关更多信息，请参阅使用 Amazon CloudWatch 监控 Amazon ElastiCache for Redis 的最佳实践的内存部分。

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 ElastiCache 定价。

有关更多信息，请参阅使用 Amazon CloudWatch 监控 Amazon ElastiCache for Redis 的最佳实践的网络部分。

延迟

您可以使用一组 CloudWatch 指标来衡量命令的延迟，这些指标提供了每个数据结构的聚合延迟。这些延迟指标是使用来自 Redis INFO 命令的 `commandstats` 统计数据计算的。

有关更多信息，请参阅使用 Amazon CloudWatch 监控 Amazon ElastiCache for Redis 的最佳实践的延迟部分。

复制

可通过 `ReplicationBytes` 指标了解被复制的数据量。尽管此指标表示复制组上的写入负载，但它不提供有关复制运行状况的信息。要了解此信息，您可以使用 `ReplicationLag` 指标。

有关更多信息，请参阅使用 Amazon CloudWatch 监控 Amazon ElastiCache for Redis 的最佳实践的复制部分。

流量管理

当发送到节点的传入命令超过 Redis 可以处理的数量时，ElastiCache for Redis 会自动管理节点的流量。这样做是为了保持引擎的最佳运行和稳定性。

在节点上主动管理流量时，指标 `TrafficManagementActive` 将发出为 1 的数据点。这表示节点对于所提供的工作负载而言可能规模过小。如果此指标在很长一段时间内保持为 1，请评估集群以确定是否需要纵向扩展或横向扩展。

有关更多信息，请参阅指标页面上的 TrafficManagementActive 指标。

选择指标统计数据和周期

虽然 CloudWatch 将允许您为每个指标选择统计数据和周期，但并非所有的组合都有用。例如，CPU 利用率的平均、最小和最大统计数据均有用，但求和统计数据却无用。

对于每个单独缓存节点，发布所有 ElastiCache 示例的持续时间均为 60 秒。对于任何 60 秒期间，缓存节点的度量标准将只包含一个单一示例。

有关如何检索您的缓存节点的度量标准之更多信息，请参阅 监控 CloudWatch 集群和节点指标。

监控 CloudWatch 集群和节点指标

ElastiCache 和 CloudWatch 集成在一起，因此您可收集多种指标。您可使用 CloudWatch 监控这些指标。

> **Note**
>
> 下列示例需要使用 CloudWatch 命令行工具。有关 CloudWatch 的更多信息和下载开发工具，请参阅 CloudWatch 产品页面。

以下程序将介绍如何使用 CloudWatch 收集过去一小时内缓存集群的存储空间统计数据。

> **Note**
>
> 下述示例中提供的 `StartTime` 和 `EndTime` 值均供说明之用。您必须针对您的缓存节点使用适当的开始和结束时间值替代示例中的相应值。

有关 ElastiCache 限制的信息，请参阅 ElastiCache 的 AWS Service Limits。

监控 CloudWatch 集群和节点指标（控制台）

收集缓存集群的 CPU 利用率统计数据

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 选择您希望查看其度量标准的缓存节点。

> **Note**
>
> 若选择 20 个以上的节点，则将禁用在控制台上查看指标。

a. 在 AWS 管理控制台的 Cache Clusters（缓存集群）页面上，单击一个或多个缓存集群的名称。

   显示缓存集群的详情页面。

b. 单击位于窗口顶部的 Nodes 选项卡。

c. 在详情窗口的 Nodes 选项卡上，选择您希望查看其度量标准的缓存节点。

   一份可用 CloudWatch 度量标准列表会显示在控制台窗口的底部。

d. 单击 CPU 利用率 度量标准。

   CloudWatch 控制台将打开，其中会显示您选择的指标。您可以使用 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。

使用 CloudWatch CLI 监控 CloudWatch 集群和节点指标

收集缓存集群的 CPU 利用率统计数据

- 对于 Linux、macOS 或 Unix：

```
aws cloudwatch get-metric-statistics \
    --namespace AWS/ElastiCache \
    --metric-name CPUUtilization \
    --dimensions='[{"Name":"CacheClusterId","Value":"test"},
{"Name":"CacheNodeId","Value":"0001"}]' \
    --statistics=Average \
    --start-time 2018-07-05T00:00:00 \
    --end-time 2018-07-06T00:00:00 \
    --period=3600
```

  对于 Windows：

```
aws cloudwatch get-metric-statistics ^
    --namespace AWS/ElastiCache ^
    --metric-name CPUUtilization ^
```

```
    --dimensions='[{"Name":"CacheClusterId","Value":"test"},
{"Name":"CacheNodeId","Value":"0001"}]' ^
    --statistics=Average ^
    --start-time 2018-07-05T00:00:00 ^
    --end-time 2018-07-06T00:00:00 ^
    --period=3600
```

使用 CloudWatch API 监控 CloudWatch 集群和节点指标

收集缓存集群的 CPU 利用率统计数据

- 使用以下参数调用 CloudWatch API GetMetricStatistics（请注意，此处的开始和结束时间仅作为示例；您需要替换为适合您自己的开始和结束时间）：
  - Statistics.member.1=Average
  - Namespace=AWS/ElastiCache
  - StartTime=2013-07-05T00:00:00
  - EndTime=2013-07-06T00:00:00
  - Period=60
  - MeasureName=CPUUtilization
  - Dimensions=CacheClusterId=mycachecluster,CacheNodeId=0002

  Example

```
http://monitoring.amazonaws.com/
    ?Action=GetMetricStatistics
    &SignatureVersion=4
    &Version=2014-12-01
    &StartTime=2018-07-05T00:00:00
    &EndTime=2018-07-06T23:59:00
    &Period=3600
    &Statistics.member.1=Average
    &Dimensions.member.1="CacheClusterId=mycachecluster"
    &Dimensions.member.2="CacheNodeId=0002"
    &Namespace=&AWS;/ElastiCache
    &MeasureName=CPUUtilization
    &Timestamp=2018-07-07T17%3A48%3A21.746Z
    &AWS;AccessKeyId=<&AWS; Access Key ID>
```

```
    &Signature=<Signature>
```

Amazon SNS 监控 ElastiCache 事件

当集群上发生重大事件时，ElastiCache 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 ElastiCache Amazon SNS 通知
- 查看 ElastiCache 事件
- 事件通知和 Amazon SNS

管理 ElastiCache Amazon SNS 通知

您可以配置 ElastiCache 以使用 Amazon Simple Notification Service (Amazon SNS) 发送重要集群事件的通知。在这些示例中，您将使用 Amazon SNS 主题的 Amazon 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 AWS 控制台、AWS CLI 或 ElastiCache API 添加 Amazon SNS 主题。

添加 Amazon SNS 主题（控制台）

以下过程说明了如何为集群添加 Amazon SNS 主题。要为复制组添加 Amazon SNS 主题，请在步骤 2 中选择复制组（而不是选择集群），然后按照相同的剩余步骤操作。

> **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 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在 Clusters（集群）中，选择要为其添加或修改 Amazon SNS 主题 ARN 的集群。
3. 选择 Modify（修改）。
4. 在 Topic for SNS Notification（SNS 通知的主题）下的 Modify Cluster（修改集群）中，选择要添加的 SNS 主题，或选择 Manual ARN input（手动 ARN 输入）并键入 Amazon SNS 主题的 ARN。
5. 选择 Modify（修改）。

添加 Amazon SNS 主题 (AWS CLI)

要为集群添加或修改 Amazon SNS 主题，请使用 AWS CLI 命令 `modify-cache-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 elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --notification-topic-arn arn:aws:sns:us-
west-2:123456789xxx:ElastiCacheNotifications
```

对于 Windows：

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --notification-topic-arn arn:aws:sns:us-west-2:123456789xx:ElastiCacheNotifications
```

有关更多信息，请参阅 modify-cache-cluster。

添加 Amazon SNS 主题 (ElastiCache API)

若要为集群添加或修改 Amazon SNS 主题，请使用下列参数调用 `ModifyCacheCluster` 操作：

- `CacheClusterId=my-cluster`
- `TopicArn=arn%3Aaws%3Asns%3Aus-west-2%3A565419523791%3AElastiCacheNotifications`

Example

```
https://elasticache.amazon.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=false
    &CacheClusterId=my-cluster
    &NotificationTopicArn=arn%3Aaws%3Asns%3Aus-
west-2%3A565419523791%3AElastiCacheNotifications
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

有关更多信息，请参阅 ModifyCacheCluster。

启用和禁用 Amazon SNS 通知

您可以打开或关闭针对集群的通知。下面将介绍如何禁用 Amazon SNS 通知。

启用和禁用 Amazon SNS 通知（控制台）

使用 AWS Management Console 禁用 Amazon SNS 通知

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 要查看运行 Redis 的集群的列表，请在导航窗格中选择 Redis。
3. 选择要修改其通知的集群左侧的框。
4. 选择 Modify（修改）。
5. 在 Topic for SNS Notification 下的 Modify Cluster 中，选择 Disable Notifications。
6. 选择 Modify（修改）。

启用和禁用 Amazon SNS 通知 (AWS CLI)

若要禁用 Amazon SNS 通知，请使用包含以下参数的命令 `modify-cache-cluster`：

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --notification-topic-status inactive
```

对于 Windows：

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --notification-topic-status inactive
```

启用和禁用 Amazon SNS 通知 (ElastiCache API)

若要禁用 Amazon SNS 通知，请使用下列参数调用 `ModifyCacheCluster` 操作：

- `CacheClusterId=my-cluster`
- `NotificationTopicStatus=inactive`

此调用返回类似于下述信息的输出：

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=false
    &CacheClusterId=my-cluster
    &NotificationTopicStatus=inactive
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Date=20141201T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20141201T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

查看 ElastiCache 事件

ElastiCache 会记录与您的集群实例、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。通过使用 ElastiCache 控制台、AWS CLI `describe-events` 命令或 ElastiCache API 操作 `DescribeEvents`，您可以轻松从日志中检索事件。

以下过程说明了如何查看过去 24 小时（1440 分钟）内的所有 ElastiCache 事件。

查看 ElastiCache 事件（控制台）

以下过程演示了使用 ElastiCache 控制台查看事件。

要查看使用 ElastiCache 控制台的事件

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 要查看所有可用事件的列表，请在导航窗格中选择 Events (事件)。

   在 Events (事件) 屏幕上，列表的每一行表示一个事件，并显示事件源、事件类型 (cache-cluster、cache-parameter-group、cache-security-group 或 cache-subnet-group)、事件的 GMT 时间及事件的描述。

   通过使用 Filter，您可以指定是要查看事件列表中的所有事件，还是仅查看特定类型的事件。

查看 ElastiCache 事件 (AWS CLI)

要使用 AWS CLI 生成 ElastiCache 事件的列表，请使用命令 `describe-events`。您可以使用可选参数来控制所列事件的类型、所列事件的时间范围、要列出的事件的最大数目等。

以下代码列出最多 40 个缓存集群事件。

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
```

以下代码列出了过去 24 小时 (1440 分钟) 内的所有事件。

```
aws elasticache describe-events --source-type cache-cluster --duration 1440
```

`describe-events` 命令的输出类似于此处所示。

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
{
```

```
    "Events": [
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Finished modifying number of nodes from 1 to 3",
            "Date": "2020-06-09T02:01:21.772Z"
        },
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0002 in availability zone us-west-2a",
            "Date": "2020-06-09T02:01:21.716Z"
        },
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0003 in availability zone us-west-2a",
            "Date": "2020-06-09T02:01:21.706Z"
        },
        {
            "SourceIdentifier": "my-mem-cluster",
            "SourceType": "cache-cluster",
            "Message": "Increasing number of requested nodes",
            "Date": "2020-06-09T01:58:34.178Z"
        },
        {
            "SourceIdentifier": "mycluster-0003-004",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0001 in availability zone us-west-2c",
            "Date": "2020-06-09T01:51:14.120Z"
        },
        {
            "SourceIdentifier": "mycluster-0003-004",
            "SourceType": "cache-cluster",
            "Message": "This cache cluster does not support persistence (ex:
 'appendonly').  Please use a different instance type to enable persistence.",
            "Date": "2020-06-09T01:51:14.095Z"
        },
        {
            "SourceIdentifier": "mycluster-0003-004",
            "SourceType": "cache-cluster",
            "Message": "Cache cluster created",
            "Date": "2020-06-09T01:51:14.094Z"
        },
```

```
        {
            "SourceIdentifier": "mycluster-0001-005",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0001 in availability zone us-west-2b",
            "Date": "2020-06-09T01:42:55.603Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-005",
            "SourceType": "cache-cluster",
            "Message": "This cache cluster does not support persistence (ex:
 'appendonly').  Please use a different instance type to enable persistence.",
            "Date": "2020-06-09T01:42:55.576Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-005",
            "SourceType": "cache-cluster",
            "Message": "Cache cluster created",
            "Date": "2020-06-09T01:42:55.574Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-004",
            "SourceType": "cache-cluster",
            "Message": "Added cache node 0001 in availability zone us-west-2b",
            "Date": "2020-06-09T01:28:40.798Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-004",
            "SourceType": "cache-cluster",
            "Message": "This cache cluster does not support persistence (ex:
 'appendonly').  Please use a different instance type to enable persistence.",
            "Date": "2020-06-09T01:28:40.775Z"
        },
        {
            "SourceIdentifier": "mycluster-0001-004",
            "SourceType": "cache-cluster",
            "Message": "Cache cluster created",
            "Date": "2020-06-09T01:28:40.773Z"
        }
    ]
}
```

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 ElastiCache 事件 (ElastiCache API)

要使用 ElastiCache API 生成 ElastiCache 事件的列表，请使用 `DescribeEvents` 操作。您可以使用可选参数来控制所列事件的类型、所列事件的时间范围、要列出的事件的最大数目等。

以下代码列出了 40 个最新的 cache-cluster 事件。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeEvents
   &MaxRecords=40
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cache-cluster
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

以下代码列出了过去 24 小时（1440 分钟）内的 cache-cluster 事件。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeEvents
   &Duration=1440
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &SourceType=cache-cluster
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

以上操作应生成类似于以下内容的输出。

```
<DescribeEventsResponse xmlns="http://elasticache.amazonaws.com/doc/2015-02-02/">
    <DescribeEventsResult>
        <Events>
            <Event>
                <Message>Cache cluster created</Message>
                <SourceType>cache-cluster</SourceType>
                <Date>2015-02-02T18:22:18.202Z</Date>
                <SourceIdentifier>mem01</SourceIdentifier>
            </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

当缓存群集上发生重要事件时，ElastiCache 可以使用 Amazon Simple Notification Service (SNS) 发布消息。此功能可用于在连接到缓存群集的各个缓存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：

- 仅能为 ElastiCache 通知配置一个主题。
- 拥有 Amazon SNS 主题的 AWS 账户必须是拥有已启用通知的缓存群集的同一账户。
- 您要向其发布通知的 Amazon SNS 主题不得加密。

> **Note**
>
> 可将加密的（静态）Amazon SNS 主题附加到集群。但是，ElastiCache 控制台中的主题状态将显示为非活动状态，当 ElastiCache 将消息推送到主题时，这会有效地取消主题与集群的关联。

- Amazon SNS 主题必须与 ElastiCache 集群位于同一区域。

ElastiCache 事件

以下 ElastiCache 事件会触发 Amazon SNS 通知。有关事件详细信息的信息，请参阅 查看 ElastiCache 事件。

事件名称	消息	描述
ElastiCache:AddCacheNodeComplete	ElastiCache:AddCacheNodeComplete : *cache-cluster*	缓存节点已添加到缓存群集，并准备就绪，可供可用。
由于空闲 IP 地址不足导致的 ElastiCache:AddCacheNodeFailed	ElastiCache:AddCacheNodeFailed : *cluster-name*	因为没有足够的可用 IP 地址，所以无法添加缓存节点。
ElastiCache:CacheClusterParametersChanged	ElastiCache:CacheClusterParametersChanged : *cluster-name*	一个或多个缓存群集参数已更改。
ElastiCache:CacheClusterProvisioningComplete	ElastiCache:CacheClusterProvisioningComplete *cluster-name-0001-005*	缓存群集预配置已完成，并且缓存群集中的缓存节点准备就绪，可供使用。
由于不兼容网络状态导致的 ElastiCache:CacheClusterProvisioningFailed	ElastiCache:CacheClusterProvisioningFailed : *cluster-name*	尝试将新缓存群集启动到不存在的 Virtual Private Cloud (VPC) 中。
ElastiCache:CacheClusterScalingComplete	CacheClusterScalingComplete : *cluster-name*	已成功完成缓存群集扩展。
ElastiCache:CacheClusterScalingFailed	ElastiCache:CacheClusterScalingFailed : *cluster-name*	对缓存群集的纵向扩展操作已失败。
ElastiCache:CacheClusterSecurityGroupModified	ElastiCache:CacheClusterSecurityGroupModified : *cluster-name*	发生下列事件之一： • 已修改授权用于缓存群集的缓存安全组列表。 •

事件名称	消息	描述
		已在与缓存群集相关的任何缓存安全组上授权一个或多个新的 EC2 安全组。 • 已从与缓存群集相关的缓存安全组中撤销一个或多个 EC2 安全组。
ElastiCache:CacheNodeReplaceStarted	`ElastiCache:CacheNodeReplaceStarted : cluster-name`	ElastiCache 已检测到运行缓存节点的主机性能下降或无法访问，并已开始缓存节点的替换工作。 **Note** 针对替换之缓存节点的 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，某些缓存客户端库可能停止使用缓存节点，即使在 ElastiCache 已替换缓存节点之后亦是如此；在这种情况下，应用程序应该在此事件发生时刷新服务器列表。

事件名称	消息	描述
ElastiCache:CacheNodeReplaceComplete	`ElastiCache:CacheNodeReplaceComplete : cluster-name`	ElastiCache 已检测到运行缓存节点的主机性能下降或无法访问，并已完成缓存节点的替换工作。 **Note** 针对替换之缓存节点的 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，某些缓存客户端库可能停止使用缓存节点，即使在 ElastiCache 已替换缓存节点之后亦是如此；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。
ElastiCache:CacheNodesRebooted	`ElastiCache:CacheNodesRebooted : cluster-name`	一个或多个缓存节点已重启。 消息 (Memcached)：`"Cache node %s shutdown"` ，然后是第二条消息：`"Cache node %s restarted"`
ElastiCache:CertificateRenewalComplete（仅限 Redis）	`ElastiCache:CertificateRenewalComplete`	已成功续订 Amazon CA 证书。

事件名称	消息	描述
ElastiCache:CreateReplicationGroupComplete	ElastiCache:CreateReplicationGroupComplete : *cluster-name*	已成功创建复制组。
ElastiCache:DeleteCacheClusterComplete	ElastiCache:DeleteCacheClusterComplete : *cluster-name*	已完成缓存群集和所有关联缓存节点的删除工作。
ElastiCache:FailoverComplete（仅限 Redis）	ElastiCache:FailoverComplete : *mycluster*	已成功故障转移至副本节点。
ElastiCache:ReplicationGroupIncreaseReplicaCountFinished	ElastiCache:ReplicationGroupIncreaseReplicaCountFinished : *cluster-name-0001-005*	已增加集群中的副本数量。
ElastiCache:ReplicationGroupIncreaseReplicaCountStarted	ElastiCache:ReplicationGroupIncreaseReplicaCountStarted : *cluster-name-0003-004*	已开始向集群添加副本的过程。
ElastiCache:NodeReplacementCanceled	ElastiCache:NodeReplacementCanceled : *cluster-name*	计划替换的集群中的节点不再计划替换。
ElastiCache:NodeReplacementRescheduled	ElastiCache:NodeReplacementRescheduled : *cluster-name*	之前计划替换的集群中的节点已计划在通知中所述的新时段内替换。 有关您可以执行的操作的信息，请参阅 替换节点。

事件名称	消息	描述
ElastiCache:NodeReplacementScheduled	`ElastiCache:NodeReplacementScheduled : cluster-name`	您集群中的节点计划在通知所述的时段内替换。 有关您可以执行的操作的信息，请参阅 替换节点。
ElastiCache:RemoveCacheNodeComplete	`ElastiCache:RemoveCacheNodeComplete : cluster-name`	缓存节点已从缓存群集中移除。
ElastiCache:ReplicationGroupScalingComplete	`ElastiCache:ReplicationGroupScalingComplete : cluster-name`	已成功完成对复制组的纵向扩展操作。
ElastiCache:ReplicationGroupScalingFailed	`"Failed applying modification to cache node type to %s."`	对复制组的纵向扩展操作失败。
ElastiCache:ServiceUpdateAvailableForNode	`"Service update is available for cache node %s."`	自助服务更新可用于节点。
ElastiCache:SnapshotComplete（仅限 Redis）	`ElastiCache:SnapshotComplete : cluster-name`	缓存快照已成功完成。
ElastiCache:SnapshotFailed（仅限 Redis）	`SnapshotFailed : cluster-name`	缓存快照失败。有关失败原因的详细信息，请参阅该集群的缓存事件。 要对快照加以说明，请参阅 `DescribeSnapshots` ，状态将是 `failed`。

相关 主题

- 查看 ElastiCache 事件

使用 AWS CloudTrail 记录 Amazon ElastiCache API 调用

Amazon ElastiCache 与 AWS CloudTrail 集成，后者是在 Amazon ElastiCache 中提供由用户、角色或 AWS 服务所采取操作的记录的服务。CloudTrail 将对 Amazon ElastiCache 的所有 API 调用作为事件捕获，包括来自 Amazon ElastiCache 控制台的调用、来自代码对 Amazon ElastiCache API 操作的调用。如果您创建跟踪记录，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶，包括 Amazon ElastiCache 的事件。如果您不配置跟踪，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 Amazon ElastiCache 发出的请求内容、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅《AWS CloudTrail 用户指南》。

CloudTrail 中的 Amazon ElastiCache 信息

在您创建 AWS 账户时，将在该账户上启用 CloudTrail。当 Amazon ElastiCache 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 AWS 服务事件一同保存在事件历史记录中。您可以在 AWS 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录 AWS 账户中的事件（包括 Amazon ElastiCache 的事件），请创建跟踪记录。通过跟踪，CloudTrail 可将日志文件传送至 Amazon S3 桶。默认情况下，在控制台中创建跟踪记录时，此跟踪记录应用于所有 区域。此跟踪记录在 AWS 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Simple Storage Service（Amazon S3）存储桶。此外，您可以配置其他 AWS 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅下列内容：

- 创建跟踪概览
- 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 会记录所有 Amazon ElastiCache 操作，ElastiCache API 参考中介绍了这些操作。例如，对 `CreateCacheCluster`、`DescribeCacheCluster` 和 `ModifyCacheCluster` 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。
- 请求是否由其它 AWS 服务发出。

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

了解 Amazon ElastiCache 日志文件条目

跟踪是一种配置，可用于将事件作为日志文件传送到您指定的 Amazon S3 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 `CreateCacheCluster` 操作。

```
{
    "eventVersion":"1.01",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"EXAMPLEEXAMPLEEXAMPLE",
        "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "userName":"elasticache-allow"
    },
    "eventTime":"2014-12-01T22:00:35Z",
    "eventSource":"elasticache.amazonaws.com",
    "eventName":"CreateCacheCluster",
    "awsRegion":"us-west-2",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS CLI/ElastiCache 1.10 API 2014-12-01",
    "requestParameters":{
        "numCacheNodes":2,
        "cacheClusterId":"test-memcached",
        "engine":"memcached",
        "aZMode":"cross-az",
        "cacheNodeType":"cache.m1.small",

    },
    "responseElements":{
        "engine":"memcached",
        "clientDownloadLandingPage":"https://console.aws.amazon.com/elasticache/
home#client-download:",
        "cacheParameterGroup":{
            "cacheParameterGroupName":"default.memcached1.4",
            "cacheNodeIdsToReboot":{
            },
```

```
            "parameterApplyStatus":"in-sync"
        },
        "preferredAvailabilityZone":"Multiple",
        "numCacheNodes":2,
        "cacheNodeType":"cache.m1.small",

        "cacheClusterStatus":"creating",
        "autoMinorVersionUpgrade":true,
        "preferredMaintenanceWindow":"thu:05:00-thu:06:00",
        "cacheClusterId":"test-memcached",
        "engineVersion":"1.4.14",
        "cacheSecurityGroups":[
            {
                "status":"active",
                "cacheSecurityGroupName":"default"
            }
        ],
        "pendingModifiedValues":{
        }
    },
    "requestID":"104f30b3-3548-11e4-b7b8-6d79ffe84edd",
    "eventID":"92762127-7a68-42ce-8787-927d2174cde1"
}
```

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了 `DescribeCacheCluster` 操作。请注意，对于所有的 Amazon ElastiCache Describe 调用 (`Describe*`)，`ResponseElements` 部分将被移除并显示为 `null`。

```
{
    "eventVersion":"1.01",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"EXAMPLEEXAMPLEEXAMPLE",
        "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "userName":"elasticache-allow"
    },
    "eventTime":"2014-12-01T22:01:00Z",
    "eventSource":"elasticache.amazonaws.com",
    "eventName":"DescribeCacheClusters",
    "awsRegion":"us-west-2",
    "sourceIPAddress":"192.0.2.01",
```

```
    "userAgent":"AWS CLI/ElastiCache 1.10 API 2014-12-01",
    "requestParameters":{
        "showCacheNodeInfo":false,
        "maxRecords":100
    },
    "responseElements":null,
    "requestID":"1f0b5031-3548-11e4-9376-c1d979ba565a",
    "eventID":"a58572a8-e81b-4100-8e00-1797ed19d172"
}
```

下面的示例显示了一个 CloudTrail 日志条目，该条目记录了 ModifyCacheCluster 操作。

```
{
    "eventVersion":"1.01",
    "userIdentity":{
        "type":"IAMUser",
        "principalId":"EXAMPLEEXAMPLEEXAMPLE",
        "arn":"arn:aws:iam::123456789012:user/elasticache-allow",
        "accountId":"123456789012",
        "accessKeyId":"AKIAIOSFODNN7EXAMPLE",
        "userName":"elasticache-allow"
    },
    "eventTime":"2014-12-01T22:32:21Z",
    "eventSource":"elasticache.amazonaws.com",
    "eventName":"ModifyCacheCluster",
    "awsRegion":"us-west-2",
    "sourceIPAddress":"192.0.2.01",
    "userAgent":"AWS CLI/ElastiCache 1.10 API 2014-12-01",
    "requestParameters":{
        "applyImmediately":true,
        "numCacheNodes":3,
        "cacheClusterId":"test-memcached"
    },
    "responseElements":{
        "engine":"memcached",
        "clientDownloadLandingPage":"https://console.aws.amazon.com/elasticache/
home#client-download:",
        "cacheParameterGroup":{
            "cacheParameterGroupName":"default.memcached1.4",
            "cacheNodeIdsToReboot":{
            },
            "parameterApplyStatus":"in-sync"
        },
```

```
        "cacheClusterCreateTime":"Dec 1, 2014 10:16:06 PM",
        "preferredAvailabilityZone":"Multiple",
        "numCacheNodes":2,
        "cacheNodeType":"cache.m1.small",
        "cacheClusterStatus":"modifying",
        "autoMinorVersionUpgrade":true,
        "preferredMaintenanceWindow":"thu:05:00-thu:06:00",
        "cacheClusterId":"test-memcached",
        "engineVersion":"1.4.14",
        "cacheSecurityGroups":[
            {
                "status":"active",
                "cacheSecurityGroupName":"default"
            }
        ],
        "configurationEndpoint":{
            "address":"test-memcached.example.cfg.use1prod.cache.amazonaws.com",
            "port":11211
        },
        "pendingModifiedValues":{
            "numCacheNodes":3
        }
    },
    "requestID":"807f4bc3-354c-11e4-9376-c1d979ba565a",
    "eventID":"e9163565-376f-4223-96e9-9f50528da645"
}
```

ElastiCache 的配额

您的 AWS 账户对于每项 AWS 服务都具有默认配额（以前称为限制）。除非另有说明，否则，每个限额都特定于 区域。您可以请求增加某些限额，但其他一些限额无法增加。

要查看 ElastiCache 的配额，请打开 Service Quotas 控制台。在导航窗格中，选择 AWS services（亚马逊云科技服务），然后选择 ElastiCache。

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 AWS 账户具有以下与 ElastiCache 相关的配额。

资源	默认
每个区域的无服务器缓存	40
每个缓存每天的无服务器快照	24
每个区域的节点数	300
每个实例类型的每个集群的节点数（已启用 Redis 集群模式）	90
每个分区的节点数（已禁用 Redis 集群模式）	6
每个区域的参数组数	300
每个区域的安全组数	50
每个区域的子网组数	300
每个子网组的子网数	20
每个用户组的用户数	100
最大用户数	1000
最大用户组数	100

参考

本部分中的主题涵盖了有关使用 Amazon ElastiCache API 和 AWS CLI 的 ElastiCache 部分的信息。本部分还包含常见错误消息和服务通知。

- 使用 ElastiCache API
- ElastiCache API 参考
- AWS CLI 参考的 ElastiCache 部分
- Amazon ElastiCache 错误消息
- 通知

使用 ElastiCache API

本部分提供一些针对任务的说明，介绍如何使用和实施 ElastiCache 操作。有关这些操作的完整描述，请参阅 Amazon ElastiCache API 参考

主题

- 使用查询 API
- 可用的库
- 对应用程序进行问题排查

使用查询 API

查询参数

HTTP 基于查询的请求是指使用 HTTP 动作 GET 或 POST 的 HTTP 请求，查询参数的名称为 `Action`。

每个查询请求必须包括一些通用参数，以处理操作的身份验证和选择事宜。

有些操作会使用参数列表。这些列表都是使用 `param.`*n* 表示法指定的。*n* 值是从 1 开始的整数。

查询请求身份验证

您只可以通过 HTTP 发送查询请求，并且每个查询请求中必须包含您的签名。本部分描述了如何创建签名。以下过程中说明的方法称为签名版本 4。

下面介绍了对发送至 AWS 的请求进行身份验证的基本步骤。其中假定您注册了 AWS，并且有一个访问密钥 ID 和秘密访问密钥。

查询身份验证流程

1. 发件人构建一个将要发送至 AWS 的请求。
2. 发件人计算请求签名，即带有一个 SHA-1 哈希函数的键控式哈希信息验证码 (HMAC)，如本主题下一部分中所定义的那样。
3. 该请求的发件人将请求数据、签名和访问密钥 ID（即所使用的秘密访问密钥的密钥标识符）发送至 AWS。
4. AWS 使用访问密钥 ID 来查询秘密访问密钥。
5. AWS 使用与计算请求中签名所用的相同算法根据请求数据和秘密访问密钥生成一个签名。
6. 如果签名匹配，那么请求将被视为可信。如果比较签名这一操作失败，那么请求将被丢弃，同时 AWS 将返回错误响应。

> **Note**
>
> 如果请求包含一个 `Timestamp` 参数，那么针对请求计算的签名将在被赋予值后的 15 分钟失效。
> 如果请求包含一个 `Expires` 参数，那么签名将在 `Expires` 参数指定的时间失效。

计算请求签名

1. 创建标准化的查询字符串，您在此过程的稍后部分需要用到它：
   a. 根据参数名称、按照自然字节排序对 UTF-8 查询字符串组成部分进行分类。参数可取自 GET URI 或 POST 正文 (当内容类型为 application/x-www-form-urlencoded 时)。
   b. URL 根据以下规则对参数名称和值进行编码：
      i. 不对任何由 RFC 3986 定义的非预留字符进行 URL 编码。这些未预留字符是 A–Z、a–z、0–9、连字符 (-)、下划线 (_)、句点 (.) 和波形符 (~)。
      ii. 使用 %XY 对所有其他参数进行百分比编码，其中“X”和“Y”分别代表十六进制字符 0-9 和大写字母 A-F。
      iii. 以 %XY%ZA... 格式对扩展的 UTF-8 字符进行百分号编码。
      iv. 将空白字符百分号编码为 %20 (不是普通编码方案中的 +)。

c. 使用等号 (=) (ASCII 字符 61) 将编码的参数名称与它们的编码值分隔开，即使参数值为空，亦应如此。

d. 使用“和”符号 (&) (ASCII 代码 38) 隔开名称/值对。

2. 依照下列伪语法创建用以签名的字符串 (“\n”代表 ASCII 换行)。

```
StringToSign = HTTPVerb + "\n" +
ValueOfHostHeaderInLowercase + "\n" +
HTTPRequestURI + "\n" +
CanonicalizedQueryString <from the preceding step>
```

HTTPRequestURI 组件是 URI 的 HTTP 绝对路径组件，但不包括查询字符串。如果 HTTPRequestURI 为空，则使用正斜杠 (/)。

3. 利用您刚创建的字符串计算符合 RFC 2104 的 HMAC，将您的秘密访问密钥当作密钥，并将 SHA256 或 SHA1 作为哈希算法。

有关更多信息，请参阅 https://www.ietf.org/rfc/rfc2104.txt。

4. 将结果值转换为 base64。

5. 将此值作为请求中的 `Signature` 参数值。

例如，下面是一个示例请求（为清晰起见，添加了换行符）。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterIdentifier=myCacheCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-12-01
```

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
    elasticache.amazonaws.com\n
    Action=DescribeCacheClusters
    &CacheClusterIdentifier=myCacheCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
```

```
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE%2F20140523%2Fus-west-2%2Felasticache
%2Faws4_request
    &X-Amz-Date=20141201T223649Z
    &X-Amz-SignedHeaders=content-type%3Bhost%3Buser-agent%3Bx-amz-content-sha256%3Bx-
amz-date
        content-type:
        host:elasticache.us-west-2.amazonaws.com
        user-agent:CacheServicesAPICommand_Client
    x-amz-content-sha256:
    x-amz-date:
```

结果是下面的已签名请求。

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterIdentifier=myCacheCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-12-01
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=AKIADQKE4SARGYLE/20141201/us-west-2/elasticache/aws4_request
    &X-Amz-Date=20141201T223649Z
    &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-amz-date
    &X-Amz-Signature=2877960fced9040b41b4feaca835fd5cfeb9264f768e6a0236c9143f915ffa56
```

有关签名流程和计算请求签名的详细信息，请参阅主题签名版本 4 签名流程及其副主题。

可用的库

AWS 为喜欢使用特定于语言的 API（而不是查询 API）构建应用程序的软件开发人员提供了软件开发工具包 (SDK)。这些开发工具包提供了一些基本功能（未包括在 API 中），如请求身份验证、请求重试和错误处理，以便您更轻松地开始工作。现已推出适用以下编程语言的开发工具包和其他资源：

- Java
- Windows 和 .NET
- PHP
- Python
- Ruby

有关其他语言的信息，请参阅示例代码和库。

对应用程序进行问题排查

ElastiCache 提供具体的描述性错误，帮助您排查与 ElastiCache API 互动时遇到的问题。

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 ElastiCache API 中做出响应的 `Error` 节点。

XPath 语法规则不仅提供了一种搜索 `Error` 节点存在情况的简单方法，而且提供了一种检索错误代码和信息的简单方法。下面的代码片段采用 Perl 和 XML::XPath 模块来确定在请求期间是否出现错误。如果出现了错误，那么代码会刊载第一个错误代码和响应信息。

```
use XML::XPath;
my $xp = XML::XPath->new(xml =>$response);
if ( $xp->find("//Error") )
{print "There was an error processing your request:\n", " Error code: ",
$xp->findvalue("//Error[1]/Code"), "\n", " ",
$xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

故障排除技巧

我们建议采用下列流程来诊断和解决 ElastiCache API 问题。

- 验证 ElastiCache 是否正确运行。

  如要执行此操作，只需打开一个浏览器窗口，然后提交一个查询请求至 ElastiCache 服务（例如 https://elasticache.amazonaws.com）。MissingAuthenticationTokenException 或 500 Internal Server Error 可确认服务有效并对请求做出响应。
- 检查您的请求结构。

  每个 ElastiCache 操作在 ElastiCache API 参考中都有一个参考页面。复查您正在使用的参数是否正确。为了给予您关于潜在错误内容的意见，请考虑示例请求或用户场景，以查看这些示例是否正在执行类似操作。
- 检查论坛。

  ElastiCache 有一个论坛，您可以在其中搜索他人在开发过程中遇到的问题的解决方案。如要查看论坛，请参阅

https://forums.aws.amazon.com/。

设置 ElastiCache Command Line Interface

本部分描述了运行命令行工具的先决条件、在何处获取命令行工具以及如何设置工具及其环境，同时包含了一系列常见的工具用途示例。

只有您打算使用 AWS CLI for ElastiCache 时，才按照此主题的说明操作。

> **Important**
>
> Amazon ElastiCache Command Line Interface (CLI) 不支持 API 版本 2014-09-30 之后的任何 ElastiCache 改进。要通过命令行使用较新的 ElastiCache 功能，请使用 AWS Command Line Interface。

主题

先决条件

本文件假定您能够在 Linux/UNIX 或 Windows 环境中操作。Amazon ElastiCache 命令行工具也可在 Mac OS X（这是基于 UNIX 的环境）上运行；但是，本指南中不包含有关 Mac OS X 的具体说明。

就惯例而言，所有命令行文本以通配的 **PROMPT>** 命令行提示符作为前缀。您的机器上的实际命令行提示符可能有所不同。我们还使用 **$** 表示 Linux/UNIX 特定命令，使用 **C:\>** 表示 Windows 特定命令。由命令得出的示例输出在其后立即显示，同时不带任何前缀。

Java 运行时环境

本指南中使用的命令行工具需要 Java 版本 5 或更高版本，方可运行。JRE 或 JDK 安装均可行。如要查看并下载适用于一系列平台 (包括 Linux/UNIX 和 Windows) 的 JRE，请参阅 Java SE 下载。

设置 Java home 变量

命令行工具根据环境变量 (JAVA_HOME) 定位 Java Runtime。此环境变量应该被设为目录的完整路径，其中包含一个名称为 bin 的子目录，而该子目录中包含可执行的 java 文件（在 Linux 和 UNIX 上）或 java.exe可执行文件（在 Windows 上）。

设置 Java Home 变量

1. 设置 Java Home 变量。
   - 在 Linux 和 UNIX 操作系统上，输入以下命令：

     ```
     $ export JAVA_HOME=<PATH>
     ```

   - 在 Windows 操作系统上，输入以下命令：

     ```
     C:\> set JAVA_HOME=<PATH>
     ```

2. 通过运行 **$JAVA_HOME/bin/java -version** 并检查输出，确认路径设置。
   - 在 Linux/UNIX 上，您将看到类似于下述信息的输出：

     ```
     $ $JAVA_HOME/bin/java -version
     java version "1.6.0_23"
     Java(TM) SE Runtime Environment (build 1.6.0_23-b05)
     Java HotSpot(TM) Client VM (build 19.0-b09, mixed mode, sharing)
     ```

   - 在 Windows 上，您将看到类似于下述信息的输出：

     ```
     C:\> %JAVA_HOME%\bin\java -version
     java version "1.6.0_23"
     Java(TM) SE Runtime Environment (build 1.6.0_23-b05)
     Java HotSpot(TM) Client VM (build 19.0-b09, mixed mode, sharing)
     ```

获得命令行工具

命令行工具可作为 ZIP 格式文件在 ElastiCache 开发工具网站上提供。这些工具是用 JAVA 编写，包括适用于 Windows 2000/XP/Vista/Windows 7、Linux/UNIX 和 Mac OSX 的 Shell 脚本。ZIP 文件是一种自含式文件，无需安装；只需下载 Zip 文件，然后将其解压到本地计算机的目录上即可。

设置工具

命令行工具依靠环境变量 (AWS_ELASTICACHE_HOME) 来查找支持库。您需要设置此环境变量后，方可使用工具。请将它设为您解压缩命令行工具的目录路径。这个目录的名称为 ElastiCacheCli-A.B.nnnn（A、B 和 n 都是版本/版本号），其中包含名称为 BIN 和 LIB 的子目录。

设置 AWS_ELASTICACHE_HOME 环境变量

- 打开一个命令行窗口，然后输入下列命令之一，以设置 AWS_ELASTICACHE_HOME 环境变量。
  - 在 Linux 和 UNIX 操作系统上，输入以下命令：

```
$ export &AWS;_ELASTICACHE_HOME=<path-to-tools>
```

  - 在 Windows 操作系统上，输入以下命令：

```
C:\> set &AWS;_ELASTICACHE_HOME=<path-to-tools>
```

为了使工具更易使用，我们建议您将工具的 BIN 目录添加至您的系统路径。本指南的其余部分假定 BIN 目录位于您的系统路径中。

将工具的 BIN 目录添加至您的系统路径

- 输入下述命令，即可将工具的 BIN 目录添加至您的系统路径。
  - 在 Linux 和 UNIX 操作系统上，输入以下命令：

```
$ export PATH=$PATH:$&AWS;_ELASTICACHE_HOME/bin
```

  - 在 Windows 操作系统上，输入以下命令：

```
C:\> set PATH=%PATH%;%&AWS;_ELASTICACHE_HOME%\bin
```

> **Note**
>
> 当您关闭命令窗口时，Windows 环境变量会重置。您可能想要永久性设置它们。请参阅文档，了解有关您的 Windows 版本的更多信息。

> **Note**
>
> 如果路径中包含空格，必须使用双引号将路径括起来，例如：
> "C:\Program Files\Java"

提供工具凭证

命令行工具需要随您的 AWS 账户提供的 AWS 访问密钥和秘密访问密钥。可以使用命令行或从位于您本地系统上的证书文件获取它们。

部署包括一份您需要使用您的信息进行编辑的模板文件 ${AWS_ELASTICACHE_HOME}/credential-file-path.template。模板文件内容如下：

```
AWSAccessKeyId=<Write your AWS access ID>
AWSSecretKey=<Write your AWS secret key>
```

> **Important**
>
> 在 UNIX 上，限制凭证文件拥有者的权限：
>
> ```
> $ chmod 600 <the file created above>
> ```

使用凭证文件设置，您将需要设置 AWS_CREDENTIAL_FILE 环境变量，以便 ElastiCache 工具能够找到您的信息。

设置 AWS_CREDENTIAL_FILE 环境变量

1. 设置 环境变量：

- 在 Linux 和 UNIX 上，使用以下命令更新变量：

```
$ export &AWS;_CREDENTIAL_FILE=<the file created above>
```

- 在 Windows 上，使用以下命令设置变量：

```
C:\> set &AWS;_CREDENTIAL_FILE=<the file created above>
```

2. 检查您的设置是否正常工作，然后运行以下命令：

```
elasticache --help
```

您应该参阅所有 ElastiCache 命令的使用页面。

环境变量

在编写脚本、配置默认值或临时覆盖这些值时，环境变量很有用处。

除了环境变量 AWS_CREDENTIAL_FILE 以外，ElastiCache 命令行界面中包含的大部分 API 工具还支持以下变量：

- EC2_REGION – 要使用的 AWS 区域。
- AWS_ELASTICACHE_URL – 要用于服务调用的 URL。如果指定了 EC2_REGION 或传递了 --region 参数，则无需指定不同的区域终端节点。

以下示例演示如何设置环境变量 EC2_REGION 以配置 API 工具所使用的区域：

Linux、OS X 或 Unix

```
$ export EC2_REGION=us-west-1
```

Windows

```
$ set EC2_REGION=us-west-1
```

Amazon ElastiCache 错误消息

Amazon ElastiCache 返回以下错误消息。您可能会收到 ElastiCache、其他 AWS 服务或者 Redis 返回的其他错误消息。有关 ElastiCache 之外来源返回的错误消息的说明，请参阅生成该错误消息的来源的文档。

错误消息：超过集群节点配额。此区域中每个集群最多可以有 %n 个节点。

原因：您尝试创建或修改集群，结果导致集群将具有超过 %n 个节点。

解决方案：更改您的请求，以使集群的节点数不超过 %n 个。或者，如果需要的节点多于 %n 个，请使用 Amazon ElastiCache 节点请求表发出请求。

有关更多信息，请参阅《Amazon Web Services 一般参考》中的 Amazon ElastiCache 限制。

错误消息：超过客户节点配额。您在此区域最多可以有 %n 个节点，。或者，您已经达到此区域中 %s 个节点的配额。

原因：您尝试创建或修改集群，结果在此区域的所有集群中您账户的节点数超过了 %n。

解决方案：更改您的请求，以使此账户下该区域所有集群中的节点总数不超过 %n。或者，如果需要的节点多于 %n 个，请使用 Amazon ElastiCache 节点请求表发出请求。

有关更多信息，请参阅《Amazon Web Services 一般参考》中的 Amazon ElastiCache 限制。

错误消息：已达到此集群 24 小时内创建的最大手动快照数量 或者此节点 24 小时内创建的最大手动快照数量已达到其配额 %n

原因：您在尝试创建集群的手动快照，但您已经创建了 24 小时内允许的最大手动快照数量。

解决方案：请等待 24 小时再尝试创建集群的其他快照。或者，如果需要立即创建手动快照，请创建具有相同数据的其他节点 (如集群中的不同节点) 的快照。

错误消息：InsufficientCacheClusterCapacity

原因：AWS 当前没有足够的可用按需容量来服务您的请求。

解决方案：

- 等待几分钟，然后再次提交您的请求；容量可能经常转移。
- 提交减少了节点数或分片数（节点组数）的新请求。例如，如果您要提交 1 个启动包含 15 个节点的请求，请改为尝试提交 3 个包含 5 个节点的请求或 15 个包含 1 个节点的请求。
- 如果您要启动集群，请提交新请求，无需指定可用区。
- 如果您要启动集群，请使用其他节点类型（可在后期扩展）提交新请求。有关更多信息，请参阅针对 Redis ElastiCache 进行扩展。

通知

本主题涵盖了您可能会感兴趣的 ElastiCache 通知。大部分情况下，通知是一种临时的情况或事件，只会持续到找到解决方案并实施为止。通知一般有开始日期和解决日期，在该时间之后通知不再相关。任何一条通知可能与您相关，也可能无关。我们推荐您阅读实施指南，遵循该指南可以改进您集群的性能。

通知不会公布新增或改进的 ElastiCache 特性或功能。

一般 ElastiCache 通知

当前没有非特定于引擎的未完成 ElastiCache 通知。

ElastiCache for Redis 特定通知

当前没有未完成的 ElastiCache for Redis 通知。

ElastiCache 适用于 Redis 文档历史记录

- API 版本：2015-02-02
- 文档最新更新时间：2023 年 11 月 27 日

下表描述了 2018 年 3 月之后《适用于 Redis ElastiCache 的每个版本中的重要更改。如需对此文档更新的通知，您可以订阅 RSS 源。

Red ElastiCache is 最新更新

变更	说明	日期
ElastiCache 适用于 Redis 的新增了对额外 c7gN 节点大小的支持	ElastiCache 适用于 Redis 的新增了对额外 c7gN 节点大小的支持。	2024 年 1 月 10 日
ElastiCache for Redis 现在支持创建无服务器缓存	现在，您可以创建无服务器缓存，从而简化缓存管理并可即时扩展，用于支持具有极其苛刻要求的应用程序。有关更多信息，请参阅选择部署选项。作为此功能的一部分，对 `ElastiCacheServiceRolePolicy` 和 `AmazonElastiCacheFullAccess` 添加了新的权限，以允许将无服务器缓存与托管 VPC 端点关联起来。此外，还添加了权限以支持使用 `AmazonElastiCacheFullAccess` 策略的修改后控制台体验。	2023 年 11 月 27 日
ElastiCache for Redis 现在支持修改集群模式	现在，您可以将集群从“已禁用集群模式”（CMD）迁移到“已启用集群模式”（CME）。有关	2023 年 5 月 11 日

	更多信息，请参阅修改集群节点。	
ElastiCache for Redis 现在支持修改传输中的加密设置	现在，您可以更改 Redis 集群的 TLS 配置，无需重新构建或重新预置集群，也不会影响应用程序可用性。有关更多信息，请参阅 Enabling in-transit encryption for an existing cluster（为现有集群启用传输中加密）。	2022 年 12 月 28 日
ElastiCache for Redis 现在支持使用 IAM 对用户进行身份验证	IAM 身份验证允许您使用 IAM 身份对与 For Redis ElastiCache 的连接进行身份验证。AWS 这使您可以增强安全模型并简化许多管理安全任务。有关更多信息，请参阅使用 IAM 进行身份验证。	2022 年 11 月 16 日
ElastiCache for Redis 现在支持 Redis 7	此版本为 Amazon ElastiCache for Redis 带来了多项新功能：Redis 功能、ACL 改进和 Sharded Pub/Sub。有关更多信息，ElastiCache 请参阅 Redis 版本 7.0。	2022 年 11 月 8 日

ElastiCache 适用于 Redis 的版本现在支持 IPV6	ElastiCache 支持互联网协议版本 4 和 6（IPv4 和 IPv6），允许您将集群配置为仅接受 IPv4 连接、仅接受 IPv6 连接或同时接受 IPv4 和 IPv6 连接（双堆栈）。在 Nitro 系统上构建的所有实例上使用 Redis 引擎版本 6.2 及更高版本的工作负载均支持 IPv6。通过 IPv6 进行访问 ElastiCache 不收取额外费用。有关更多信息，请参阅 Choosing a network type（选择网络类型）。	2022 年 11 月 7 日
ElastiCache 适用于 Redis 的版本现在支持原生 JavaScript 对象表示法 (JSON) 格式	原生 JavaScript 对象表示法 (JSON) 格式是一种对 Redis 集群内的复杂数据集进行编码的简单、无架构的方法。您可以在 Redis 集群中使用 JavaScript 对象表示法 (JSON) 格式在本地存储和访问数据，并更新存储在这些集群中的 JSON 数据，而无需管理用于序列化和反序列化的自定义代码。有关更多信息，请参阅 JSON 入门。	2022 年 5 月 25 日

ElastiCache 现在支持 PrivateLink	AWS PrivateLink 允许您在没有互联网网关、NAT 设备、VPN 连接或 Di AWS rect Connect 连接的情况下私密访问 ElastiCache API 操作。有关更多信息，请参阅适用于 Redis 的亚马逊 ElastiCache API 和接口 VPC 终端节点 (AWS PrivateLink) 或亚马逊 ElastiCache API 和适用于 Memcached 的接口 VPC 终端节点 (AWS PrivateLink)。	2022 年 1 月 24 日
ElastiCache 适用于 Redis 的版本现在支持 Redis 6.2 和数据分层	Amazon ElastiCache for Redis 推出了亚马逊支持的下一版 Redis 引擎。 ElastiCache ElastiCache 适用于 Redis 6.2 的集群包括使用 8 个 vCPU 或更多 vCPU 的 x86 节点类型或具有 4 个 vCPU 或更多 VCPU 的 Graviton2 节点类型的集群的性能改进。ElastiCache 适用于 Redis 还引入了数据分层。借助数据分层功能，您将能够以更低的成本将集群容量最高扩展到数百 TB。有关更多信息，ElastiCache 请参阅 Redis 版本 6.2（增强）和数据分层。	2021 年 11 月 23 日

支持 Auto Scaling	ElastiCache 适用于 Redis 的版本现在支持 Auto Scaling。ElastiCache 对于 Redis 来说，自动缩放是指能够自动增加或减少 for Redis 服务中所需的分片或副本。 ElastiCache ElastiCache 利用 Application Auto Scaling 服务来提供此功能。有关更多信息，请参阅适用于 Redis 集群的 Auto Scaling ElastiCache 。	2021 年 8 月 19 日
对 Redis 慢日志传输的支持	ElastiCache 现在允许你将 Redis SLOWLOG 直播到两个目的地之一：Amazon Data Firehose 或 Amazon Logs。CloudWatch 有关更多信息，请参阅日志传输。	2021 年 4 月 22 日
对标记资源和条件键的支持	ElastiCache 现在支持标记，以帮助您管理集群和其他 ElastiCache 资源。有关更多信息，请参阅为 ElastiCache 资源添加标签。 ElastiCache 还引入了对条件键的支持。您可以指定决定 IAM policy 如何生效的条件。有关更多信息，请参阅使用条件键。	2021 年 4 月 7 日

ElastiCache 现已在 O AWS utposts 上线	AWS Outpost s 为几乎任何数据中心、托管空间或本地设施 AWS 提供原生服务、基础设施和运营模式。您可以 ElastiCache 在 Outposts 上部署，以便在本地设置、操作和使用缓存，就像在云中一样。有关更多信息，请参阅适用于 Redis 的使用 Outposts 或适用于 Memcached 的使用 Outposts。	2020 年 10 月 8 日
ElastiCache 现在支持 Redis 6	Amazon ElastiCache for Redis 推出了亚马逊支持的下一版 Redis 引擎。 ElastiCache此版本包括使用基于角色的访问控制对用户进行身份验证、无版本支持、客户端缓存以及重要操作改进。有关更多信息，ElastiCache 请参阅 Redis 版本 6.0（增强版）。	2020 年 10 月 7 日
ElastiCache 现在支持 Local Zones	本地区域是地理位置靠近您的用户的 AWS 区域的扩展。通过创建新的子网并将其分配给本地 AWS 区域，您可以将任何虚拟私有云 (VPC) 从父区域扩展到本地区域。有关更多信息，请参阅使用 Local Zones。	2020 年 9 月 25 日

ElastiCache for Redis 现在支持将你的 Redis 集群环境扩展到最多 500 个节点或 500 个分片	Redis 集群模式使得如下配置成为可能：您可以使用这些配置跨多个分片对数据进行分区，并提供更好的可扩展性、性能和可用性。此功能已在 Amazon 上推出， ElastiCache 适用于 Redis 版本 5.0.6 及更高版本的所有 AWS 区域，也适用于所有现有和新增的 Redis ElastiCache 集群环境。有关更多信息，请参阅 Redis 节点和分区。	2020 年 8 月 13 日
ElastiCache 现在支持资源级权限	现在，您可以通过在 AWS Identity and Access Management (IAM) 策略中指定 ElastiCache 资源来限制用户的权限范围。有关更多信息，请参阅资源级别权限。	2020 年 8 月 12 日
ElastiCache 适用于 Redis 添加了其他亚马逊指标 CloudWatch	ElastiCache for Redis 现在支持新的 CloudWatch 指标，包括`PubSubCmds` 和。`HyperLogLogBasedCmds` 有关完整列表，请参阅 Redis 的指标。	2020 年 6 月 10 日
ElastiCache 现在支持集群的 ElastiCache 自动更新	Amazon ElastiCache 现在支持在服务更新的 “建议申请截止日期” 过后自动更新 ElastiCache 集群。 ElastiCache 将使用您的维护窗口来安排适用集群的自动更新。有关更多信息，请参阅自助更新。	2020 年 5 月 13 日

ElastiCache 适用于 Redis 的全局数据存储现在支持 Redis 的全球数据存储	适用于 Redis 的全球数据存储库功能提供完全托管、快速、可靠和安全的跨 AWS 区域复制。使用此功能，您可以为 Redis 创建跨区域只读副本集群，以实现跨区域的低延迟读取和灾难恢复。 ElastiCache AWS 您可以创建、修改和描述全局数据存储。您还可以在全球数据存储中添加或删除 AWS 区域，并将某个 AWS 区域提升为全球数据存储中的主区域。有关更多信息，请参阅使用全球数据存储跨 AWS 区域复制。	2020 年 3 月 16 日
ElastiCache 适用于 Redis 现在支持 Redis 版本 5.0.6	有关更多信息，ElastiCache请参阅 Redis 版本 5.0.6（增强版）。	2019 年 12 月 18 日
亚马逊 ElastiCache 现在支持 T3 标准缓存节点	现在，您可以在 Amazon 中启动下一代通用可突发 T3 标准缓存节点。 ElastiCacheAmazon EC2 的 T3-Standard 实例提供基准水平的 CPU 性能，并能随时突增 CPU 使用量，直至累积的积分耗尽。有关更多信息，请参阅支持的节点类型。	2019 年 11 月 12 日

亚马逊 ElastiCache 现在支持在现有 ElastiCache 的 Redis 服务器上修改身份验证令牌	ElastiCache for Redis 5.0.6 现在允许您通过设置和轮换新令牌来修改身份验证令牌。现在，您可以在使用活动令牌时对其进行修改。您还可以将全新的令牌添加到启用了传输中加密的现有集群中，这些集群以前是在没有身份验证令牌的情况下设置的。这是一个两步过程，可以通过该过程设置和轮换令牌，而不会中断客户端请求。目前不支持此功能 AWS CloudFormation。有关更多信息，请参阅使用 Redis AUTH 命令对用户进行身份验证。	2019 年 10 月 30 日
亚马逊 ElastiCache 现在支持在亚马逊 EC2 上从 Redis 进行在线数据迁移	现在，您可以使用在线迁移将您的数据从 Amazon EC2 上的自托管 Redis 迁移到亚马逊。ElastiCache有关更多信息，请参阅在线迁移到 ElastiCache。	2019 年 10 月 28 日
ElastiCache 适用于 Redis 的 Redis 集群模式引入了在线垂直扩展。	现在，您可以按需扩展或缩小分片 Redis 集群。 ElastiCache for Redis 通过更改节点类型来调整集群的大小，同时集群继续保持在线状态并处理传入的请求。有关更多信息，请参阅通过修改节点类型来进行在线纵向扩展。	2019 年 8 月 20 日
ElastiCache for Redis 现在允许用户为你的 Amazon ElastiCache for Redis 集群使用单个读取器终端节点。	此功能允许您通过单个集群级终端节点将所有读取流量 ElastiCache 定向到您的 for Redis 集群，以利用负载平衡和更高的可用性。有关更多信息，请参阅查找连接端点。	2019 年 6 月 13 日

ElastiCache for Redis 现在允许用户按照自己的计划应用服务更新	使用此功能，您可以选择在所选时间应用可用的服务更新，而不仅仅是在维护时段。这将最大限度地减少服务中断，尤其是在业务流高峰期，并有助于确保您的集群在 ElastiCache受支持的合规计划中时保持合规性。有关更多信息，请参阅 Amazon 中的自助服务更新 ElastiCache 和亚马逊合规性验证 ElastiCache。	2019 年 6 月 4 日
ElastiCache 标准预留实例产品：部分预付、全额预付和不预付。	预留实例使您可以根据 ElastiCache 实例类型和 AWS 地区灵活地将 Amazon 实例预留一年或三年。有关更多信息，请参阅使用预留节点管理成本。	2019 年 1 月 18 日
ElastiCache 对于 Redis，每个 Redis 集群最多支持 250 个节点	Redis 集群的节点或分片限制可以提高到每个 250 ElastiCache 个上限。有关更多信息，请参阅分片。	2018 年 11 月 19 日
ElastiCache for Redis 支持所有 T2 节点上的自动故障转移以及备份和恢复	ElastiCache for Redis 引入了对所有 T2 节点上的自动故障转移、创建快照以及备份和还原的支持。有关更多信息，ElastiCache 请参阅 Redis Backup 和还原以及快照。	2018 年 11 月 19 日
ElastiCache 对于 Redis 支持 M5 和 R5 节点	ElastiCache for Redis 现在支持 M5 和 R5 节点、基于 Nitro 系统的通用和内存优化型实例类型。 AWS 有关更多信息，请参阅支持的节点类型。	2018 年 10 月 23 日

支持动态更改只读副本的数量	ElastiCache for Redis 增加了对在不停机集群的情况下向任何集群添加和删除只读副本的支持。有关本版本中的这些更改和其他变更的更多信息，请参阅《for Redis 用户指南》中的 “ElastiCache 更改副本数量”。另请参阅 ElastiCache API 参考IncreaseReplicaCount中的DecreaseReplicaCount和。	2018 年 9 月 17 日
FedRAMP 合规性认证	ElastiCache for Redis 现已通过 FedRAMP 合规认证。有关更多信息，请参阅 Amazon 合规性验证 ElastiCache。	2018 年 8 月 30 日
Redis（已启用集群模式）引擎升级	Amazon ElastiCache for Redis 增加了对升级 Redis（已启用集群模式）引擎版本的支持。有关更多信息，请参阅升级引擎版本。	2018 年 8 月 20 日
PCI DSS 合规性认证	ElastiCache for Redis 现已通过 PCI DSS 合规性认证。有关更多信息，请参阅 Amazon 合规性验证 ElastiCache。	2018 年 7 月 5 日
对 Redis ElastiCache 的 Support 4.0.10	ElastiCache for Redis 现在支持 Redis 4.0.10，在单个版本中同时支持加密和在线集群大小调整。有关更多信息，ElastiCache 请参阅 Redis 版本 4.0.10（增强版）。	2018 年 6 月 14 日

用户指南重组	现在，对单一ElastiCache 用户指南进行了重组，因此有单独的 Redis 用户指南（适用于 Redis 用户指南）和 Memcach ed（ElastiCache 适用于 Memcached 用户指南）ElastiCache 的用户指南。AWS CLI 命令参考：elasticache 部分和亚马逊 ElastiCache API 参考中的文档结构保持不变。	2018 年 20 月 4 日
支持 EngineCPUUtilization 指标	ElastiCache for Redis 添加了一个新指标`EngineCPU Utilization` ，用于报告当前使用的 CPU 容量的百分比。有关更多信息，请参阅 Redis 的指标。	2018 年 4 月 9 日

下表描述了 2018 年 3 月之前对《》用户指南所ElastiCache 做的重要更改。

更改	描述	更改日期
对亚太地区（大阪本地）区域的支持。	ElastiCache 增加了对亚太地区（大阪本地）区域的支持。亚太地区（大阪）区域目前支持一个可用区，且仅采用邀请方式。有关更多信息，请参阅下列内容： • 支持的区域 • 支持的缓存节点类型	2018 年 2 月 12 日
对欧洲（巴黎）区域的支持。	ElastiCache 增加了对欧盟（巴黎）地区的支持。有关更多信息，请参阅下列内容： • 支持的区域	2017 年 12 月 18 日

更改	描述	更改日期
	• 支持的缓存节点类型	
对中国（宁夏）区域的支持	Amazon ElastiCache 增加了对中国（宁夏）地区的支持。有关更多信息，请参阅下列内容： • 支持的区域 • 支持的缓存节点类型	2017 年 12 月 11 日
对服务相关角色的支持	此版本 ElastiCache 增加了对服务关联角色 (SLR) 的支持。有关更多信息，请参阅下列内容： • 将服务相关角色用于 Amazon ElastiCache • 设置您的权限（仅限新 ElastiCache 用户）	2017 年 12 月 7 日
支持 R4 节点类型	此版本在支持的所有 AWS 区域中 ElastiCache 增加了对 R4 节点类型的支持。 ElastiCache您可以将 R4 节点类型作为按需或预留缓存节点进行购买。有关更多信息，请参阅下列内容： • 支持的缓存节点类型 • 特定于 Redis 节点类型的参数	2017 年 11 月 20 日

更改	描述	更改日期
ElastiCache 适用于 Redis 3.2.10 并支持在线重新分片	Amazon ElastiCache for Redis 增加了对 Redis ElastiCache 3.2.10 的支持。 ElastiCache for Redis 还引入了在线集群大小调整功能，以便在集群继续处理传入的 I/O 请求的同时向集群添加或删除分片。有关更多信息，请参阅下列内容： • 在线集群大小调整 • Redis（已启用集群模式）的在线重新分片和分片重新平衡	2017 年 11 月 9 日
HIPAA 资格	ElastiCache 在您的集群上启用加密后，适用于 Redis 的 3.2.6 版现已通过 HIPAA 资格认证。有关更多信息，请参阅下列内容： • Amazon 合规性验证 ElastiCache • Amazon ElastiCache 中的数据安全性	2017 年 11 月 2 日
ElastiCache 适用于 Redis 3.2.6 并支持加密	ElastiCache 增加了对 Redi ElastiCache s 3.2.6 的支持，其中包括两个加密功能： • 传输中加密可对移动中的数据进行加密，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。 • 静态加密可在同步和备份操作期间对磁盘上的数据进行加密。 有关更多信息，请参阅下列内容： • Amazon ElastiCache 中的数据安全性 • 支持的 ElastiCache for Redis 版本	2017 年 10 月 25 日

更改	描述	更改日期
连接模式主题	ElastiCache 文档添加了一个涵盖访问 Amazon VPC 中 ElastiCache集群的各种模式的主题。 有关更多信息，请参阅《ElastiCache 用户指南》中的 访问 Amazon VPC 中 ElastiCache 缓存的访问模式。	2017 年 4 月 24 日
支持测试自动故障转移	ElastiCache 增加了对在支持复制的 Redis 集群上测试自动故障转移的支持。有关更多信息，请参阅下列内容： • 测试自动故障转移（在 ElastiCache 用户指南中）。 • 《ElastiCache API 参考》中的 TestFailover。 • AWS CLI 参考中的 test-failover。	2017 年 4 月 4 日
改进 Redis 还原	ElastiCache 通过调整集群大小添加增强的 Redis 备份和恢复。此功能支持将备份还原到与创建备份所用集群具有不同分片数量的集群。（对于 API 和 CLI，此功能可以还原不同数量的节点组，而不是不同数量的分片。）此更新还支持不同的 Redis 槽配置。有关更多信息，请参阅从备份还原到新缓存。	2017 年 3 月 15 日
新 Redis 内存管理参数	ElastiCache 添加了一个新的 Redis 参数`reserved-memory-percent`，这样可以更轻松地管理您的预留内存。此参数适用于所有版本的 Red ElastiCache is。有关更多信息，请参阅下列内容： • 管理预留内存 • Redis 3.2.4 的新参数	2017 年 3 月 15 日

更改	描述	更改日期
欧洲西部（伦敦）区域的支持	ElastiCache 增加了对欧洲（伦敦）地区的支持。当前只支持类型为 T2 和 M4 的节点。有关更多信息，请参阅下列内容： • 支持的区域 • 支持的缓存节点类型	2016 年 12 月 13 日
加拿大（蒙特利尔）区域的支持	ElastiCache 增加了对加拿大（蒙特利尔）地区的支持。该区域目前仅支持节点类型 M4 和 T2。 AWS 有关更多信息，请参阅下列内容： • 支持的区域 • 支持的缓存节点类型	2016 年 12 月 8 日
支持 M4 和 R3 节点类型	ElastiCache 增加了对南美洲（圣保罗）地区的 R3 和 M4 节点类型以及中国（北京）地区的 M4 节点类型的支持。有关更多信息，请参阅下列内容： • 支持的区域 • 支持的缓存节点类型	2016 年 11 月 1 日
美国东部 2（俄亥俄）区域支持	ElastiCache 添加了对具有 M4、T 2 和 R3 节点类型的美国东部（俄亥俄州）区域 (us-east -2) 的支持。有关更多信息，请参阅下列内容： • 支持的区域 • 支持的缓存节点类型	2016 年 10 月 17 日

更改	描述	更改日期
对 Redis 集群的支持	ElastiCache 添加了对 Redis 集群的支持（增强版）。使用 Redis 集群的客户可将其数据分配到最多 15 个分片（节点组）。每个分片都支持每分片最多 5 个只读副本的复制。Redis 集群自动故障转移用时约为早期版本的四分之一。 本版本包含重新设计的管理控制台，采用与业界用法一致的术语。 有关更多信息，请参阅下列内容： • 比较 Memcached 和 Redis • ElastiCache 适用于 Redis 组件和功能 – 请注意与“节点”、“分片”、“集群”和“复制”有关的部分。 • ElastiCache for Redis 术语	2016 年 10 月 12 日
M4 节点类型支持	ElastiCache 在支持的大多数 AWS 区域中添加了对 M4 系列节点类型的支持ElastiCache。您可以将 M4 节点类型作为按需或预留缓存节点进行购买。有关更多信息，请参阅下列内容： • 支持的缓存节点类型 • 特定于 Redis 节点类型的参数	2016 年 8 月 3 日
孟买区域支持	ElastiCache 增加了对亚太地区（孟买）地区的支持。有关更多信息，请参阅下列内容： • 支持的缓存节点类型 • 特定于 Redis 节点类型的参数	2016 年 6 月 27 日

更改	描述	更改日期
快照导出	ElastiCache 添加了导出 Redis 快照的功能，以便您可以从外部ElastiCache访问快照。有关更多信息，请参阅下列内容： • 导出备份在 Amazon ElastiCache 用户指南中 • CopySnapshot在 Amazon ElastiCache API 参考中	2016 年 26 月 5 日
节点类型纵向扩展	ElastiCache 添加了扩展 Redis 节点类型的功能。有关更多信息，请参阅针对 Redis ElastiCache 进行扩展 。	2016 年 3 月 24 日
轻松升级引擎	ElastiCache 添加了轻松升级 Redis 缓存引擎的功能。有关更多信息，请参阅引擎版本和升级 。	2016 年 3 月 22 日
支持 R3 节点类型	ElastiCache 增加了对中国（北京）区域和南美洲（圣保罗）区域的 R3 节点类型的支持。有关更多信息，请参阅支持的缓存节点类型。	2016 年 3 月 16 日
ElastiCache 使用 Lambda 函数进行访问	添加了有关配置 Lambda 函数以 ElastiCache 在亚马逊 VPC 中访问的教程。有关更多信息，请参阅ElastiCache 教程和视频。	2016 年 2 月 12 日
对 Redis 2.8.24 的支持	ElastiCache 增加了对 Redis 版本 2.8.24 的支持，并增加了自 Redis 2.8.23 以来的改进。这些改进包括错误修复以及对记录错误内存访问地址的支持。有关更多信息，请参阅下列内容： • ElastiCache for Redis 版本 2.8.24（加强版） • Redis 2.8 发布说明	2016 年 1 月 20 日
对亚太地区（首尔）区域的支持	ElastiCache 添加了对具有 t2、m3 和 r3 节点类型的亚太地区（首尔）（ap-northeast-2）区域的支持。	2016 年 1 月 6 日

更改	描述	更改日期
亚马逊 ElastiCache 控制台变更。	由于较新的 Redis 版本提供了更好、更稳定的用户体验，因此管理控制台中不再列出 Redis 版本 2.6.13、2.8.6 和 2.8.19。 ElastiCache 有关其他选项和更多信息，请参见支持的 ElastiCache for Redis 版本。	2015 年 12 月 15 日
对 Redis 2.8.23 的支持。	ElastiCache 增加了对 Redis 版本 2.8.23 的支持，并增加了自 Redis 2.8.22 以来的改进。改进功能包括错误修复以及对新参数 `close-on-slave-write` 的支持；如果启用该参数，尝试写入只读副本的客户端将会断开连接。有关更多信息，请参阅ElastiCache for Redis 版本 2.8.23（加强版）。	2015 年 11 月 13 日
对 Redis 2.8.22 的支持。	ElastiCache 增加了对 Redis 版本 2.8.22 的支持，并 ElastiCache 增加了自 2.8.21 版本以来的增强和改进。改进功能包括： • 实施了无分支保存过程，此过程可在可用内存不足而可能导致分支的保存失败时确保保存成功。 • 其他 CloudWatch 指标 — SaveInProgress和ReplicationBytes。 • 为了启用部分同步，Redis 参数 `repl-backlog-size` 现在应用于所有集群。 有关更改的完整列表和更多信息，请参阅ElastiCache for Redis 版本 2.8.22（增强版）。 本文档版本包括文档的重组和 ElastiCache 命令行界面 (CLI) 文档的删除。要使用命令行，请参阅AWS 命令行 ElastiCache。	2015 年 9 月 28 日

更改	描述	更改日期
对 Redis 2.8.21 的支持	ElastiCache 增加了对 Redis 版本 2.8.21 和 Redis 自 2.8.19 版本以来的改进的支持。此 Redis 版本包含几个错误修复。有关更多信息，请参阅 Redis 2.8 发布说明。	2015 年 7 月 29 日
新主题：ElastiCache 从外部访问 AWS	添加了有关如何从外部访问 ElastiCache 资源的新主题 AWS。有关更多信息，请参阅 ElastiCache 从外部访问 AWS。	2015 年 7 月 9 日
增加了节点替换消息	ElastiCache 添加了三条与定时节点替换相关的消息 ElastiCacheElastiCache：NodeReplacementScheduledNodeReplacementRescheduled、:和 ElastiCache:NodeReplacementCanceled。 有关计划替换节点时可以采取的更多信息和操作，请参阅 ElastiCache事件通知和 Amazon SNS。	2015 年 6 月 11 日

更改	描述	更改日期
支持 Redis 版本 2.8.19。	ElastiCache 增加了对 Redis 版本 2.8.19 和 Redis 自 2.8.6 版本以来的改进的支持。增加的支持包括： • HyperLogLog 数据结构，使用 Redis 命令 PFADD、PFCOUNT 和 PFMERGE。 • 使用新命令 ZRANGEBYLEX、ZLEXCOUNT 和 ZREMRANGEBYLEX 的 Lexicographic 范围查询。 • 推出了几个错误修复，即，通过在后台保存（bgsave）子进程意外终止时使主 SYNC 失败来防止主节点向副本节点发送陈旧数据。 有关的更多信息 HyperLogLog，请参阅 Redis 的新数据结构：. HyperLogLog 有关 PFADD、PFCOUNT 和 PFMERGE 的更多信息，请参阅 Redis 文档并单击。HyperLogLog	2015 年 3 月 11 日
对成本分配标签的支持	ElastiCache 添加了对成本分配标签的支持。有关更多信息，请参阅使用成本分配标签监控成本。	2015 年 2 月 9 日
Support AWS GovCloud for（美国西部）区域	ElastiCache 添加了对 AWS GovCloud（美国西部）(-us-gov-west1) 区域的支持。	2015 年 1 月 29 日
对欧洲地区（法兰克福）区域的支持	ElastiCache 增加了对欧洲（法兰克福）（eu-central-1）地区的支持。	2015 年 1 月 19 日

更改	描述	更改日期
对 Redis 复制组的多可用区支持	ElastiCache 添加了对从主节点到 Redis 复制组中只读副本的多可用区的支持。 ElastiCache 监控复制组的运行状况。如果主副本出现故障，则 ElastiCache 自动将副本升级为主副本，然后替换该副本。有关更多信息，请参阅使用多可用区最大限度地 ElastiCache 缩短 Redis 的停机时间。	2014 年 10 月 24 日
AWS CloudTrail 支持 API 调用记录	ElastiCache 添加了 AWS CloudTrail 对使用记录所有 ElastiCache API 调用的支持。有关更多信息，请参阅使用 AWS CloudTrail 记录 Amazon ElastiCache API 调用。	2014 年 9 月 15 日
支持新的实例大小	ElastiCache 增加了对其他通用型 (T2) 实例的支持。有关更多信息，请参阅使用参数组配置引擎参数。	2014 年 9 月 11 日
支持新的实例大小	ElastiCache 增加了对其他通用型 (M3) 实例和内存优化 (R3) 实例的支持。有关更多信息，请参阅使用参数组配置引擎参数。	2014 年 7 月 1 日
Redis 集群的备份和还原	在此版本中， ElastiCache 允许客户创建其 Redis 集群的快照，并使用这些快照创建新集群。备份是集群在特定时刻的副本，包含集群元数据以及 Redis 缓存中的所有数据。备份存储在 Amazon S3 中，客户可以随时将数据从快照还原到新集群中。有关更多信息，请参阅快照和还原。	2014 年 4 月 24 日
Redis 2.8.6	ElastiCache 除了 Redis 2.6.13 之外，还支持 Redis 2.8.6。Redis 2.8.6 支持部分重新同步，支持用户定义随时可用的只读副本的最小数量，因此，客户可以改进只读副本的弹性和容错功能。Redis 2.8.6 还提供对服务器上发生的事件的全面支持 publish-and-subscribe，客户可以收到服务器上发生的事件的通知。	2014 年 3 月 13 日

更改	描述	更改日期
Redis 缓存引擎	ElastiCache 除了 Memcached 之外，还提供 Redis 缓存引擎软件。当前使用 Redis 的客户可以使用 Red ElastiCache is 快照文件中的现有数据 “播种” 新的 Redis 缓存集群，从而轻松迁移到托管环境。ElastiCache 为了支持 Redis 复制功能， ElastiCache API 现在支持复制组。客户可以创建包含一个主 Redis 缓存节点的复制组，然后添加一个或多个只读副本节点，这些节点可自动与主节点中的缓存数据保持同步。读取操作密集型应用程序可以将工作负载转移给只读副本，从而减轻主节点上的负载。只读副本还可以在主缓存节点发生故障时防止数据丢失。	2013 年 9 月 3 日
对默认 Amazon Virtual Private Cloud（VPC）的支持	在本版本中，与亚马逊虚拟私 ElastiCache 有云 (VPC) Virtual Private Cloud 完全集成。对于新客户，默认情况下会在 Amazon VPC 中创建缓存群集。有关更多信息，请参阅Amazon VPC 和 ElastiCache 安全性。	2013 年 1 月 8 日
支持 Amazon Virtual Private Cloud（VPC）	在此版本中，可以在亚马逊虚拟私有云 (VPC) 中启动 ElastiCache 集群 Amazon Private Cloud。默认情况下，新客户的缓存群集会在 Amazon VPC 中自动创建；现有客户可以按自己的步调迁移到 Amazon VPC。有关更多信息，请参阅Amazon VPC 和 ElastiCache 安全性。	2012 年 12 月 20 日
新缓存节点类型	此版本提供四种额外的缓存节点类型。	2012 年 11 月 13 日
预留缓存节点	此版本增加了对预留缓存节点的支持。	2012 年 4 月 5 日
新指南	这是 A mazon ElastiCache 用户指南的第一个版本。	2011 年 8 月 22 日

AWS 词汇表

有关最新 AWS 术语，请参阅《AWS 词汇表 参考资料》中的AWS 词汇表。

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。