ElastiCache 适用于 Memcached 用户指南

Amazon ElastiCache

API 版本 2015-02-02

Amazon ElastiCache: ElastiCache 适用于 Memcached 用户指南

Table of Contents

Memcached 版 ElastiCache 的 Amazon 是什么？

欢迎阅读亚马逊 Mem ElastiCache cached 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 Memcached 提供无服务器缓存，可简化为应用程序添加和操作基于 Memcached 的缓存。 ElastiCache for Memcached Serverless 使您能够在不到一分钟的时间内创建高可用性缓存，并且无需配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 无服务器还无需规划和管理缓存容量。 ElastiCache 持续监控应用程序使用的缓存内存和计算，并自动扩展容量以满足应用程序的需求。 ElastiCache 通过抽象底层缓存基础设施和软件，为开发人员提供简单的端点体验。 ElastiCache 自动透明地管理硬件配置、监控、节点更换和软件修补，这样您就可以专注于应用程序开发，而不是操作缓存。

ElastiCache 适用于 Memcached Serverless 与 Memcached 1.6 及更高

ElastiCache 为 Memcached 集群设计自己的集群

如果您需要精细控制您 ElastiCache 的 for Memcached 集群，则可以选择使用设计自己的 Memcached 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，因此它会自动管理集群的硬件配置、监控、节点更换和软件修补。

ElastiCache 为 Memcached 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跨可用区可用性。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 Memcached 集群应用新的软件补丁。

在 ElastiCache 为 Memcached 设计自己的版本时，你可以选择运行 Memcached 1.4 及更高版本。

工作方式

在这里，您可以找到 for Memcached 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 Memcached 提供了两种部署选项：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

- Memcac ElastiCache hed 的工作原理
- 定价维度

Memcac ElastiCache hed 的工作原理

ElastiCache 适用于 Memcached 无服务器

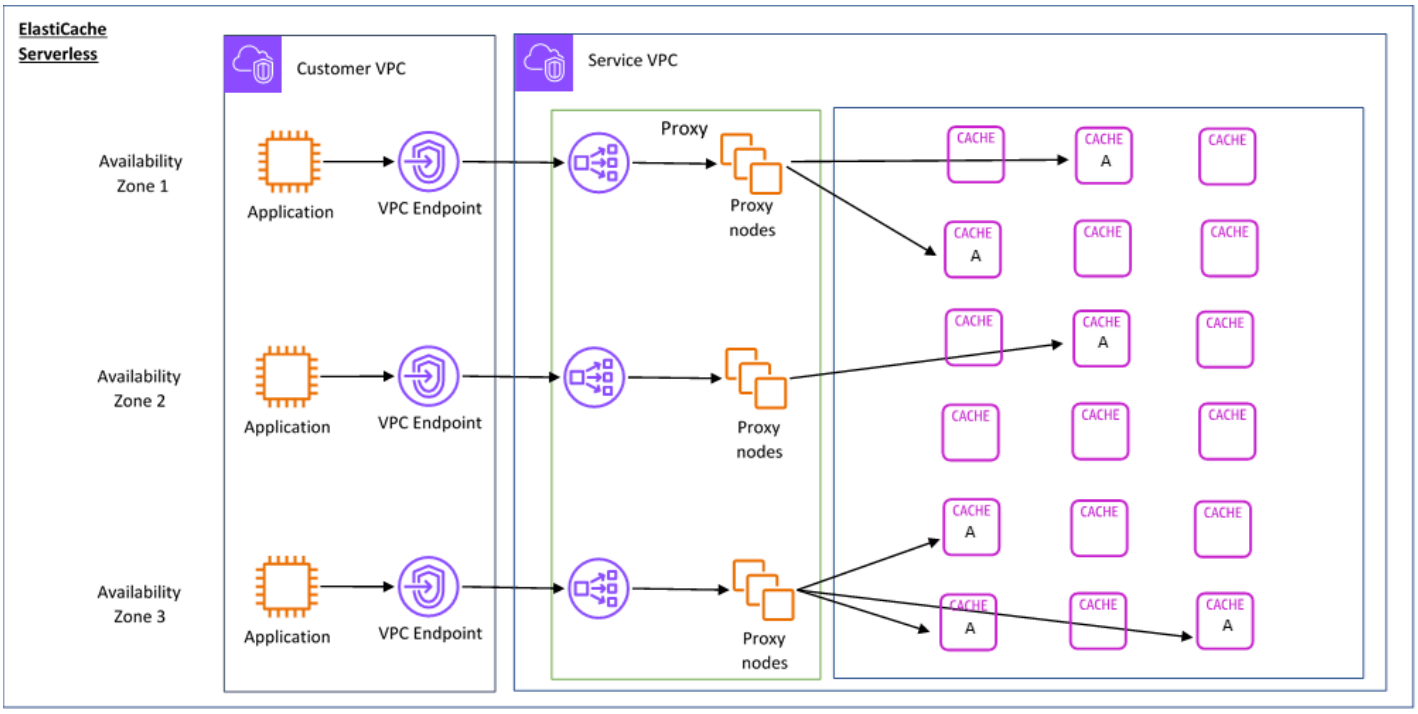
ElastiCache for Memcached Serverless 允许您创建缓存，而不必担心容量规划、硬件管理或集群设计。您只需为缓存提供一个名称，即可收到一个端点，可以在 Memcached 客户端中配置此端点以开始访问缓存。

> **Note**
>
> ElastiCache for Memcached 服务器仅与支持 TLS 的 Memcached 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 Memcached 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。

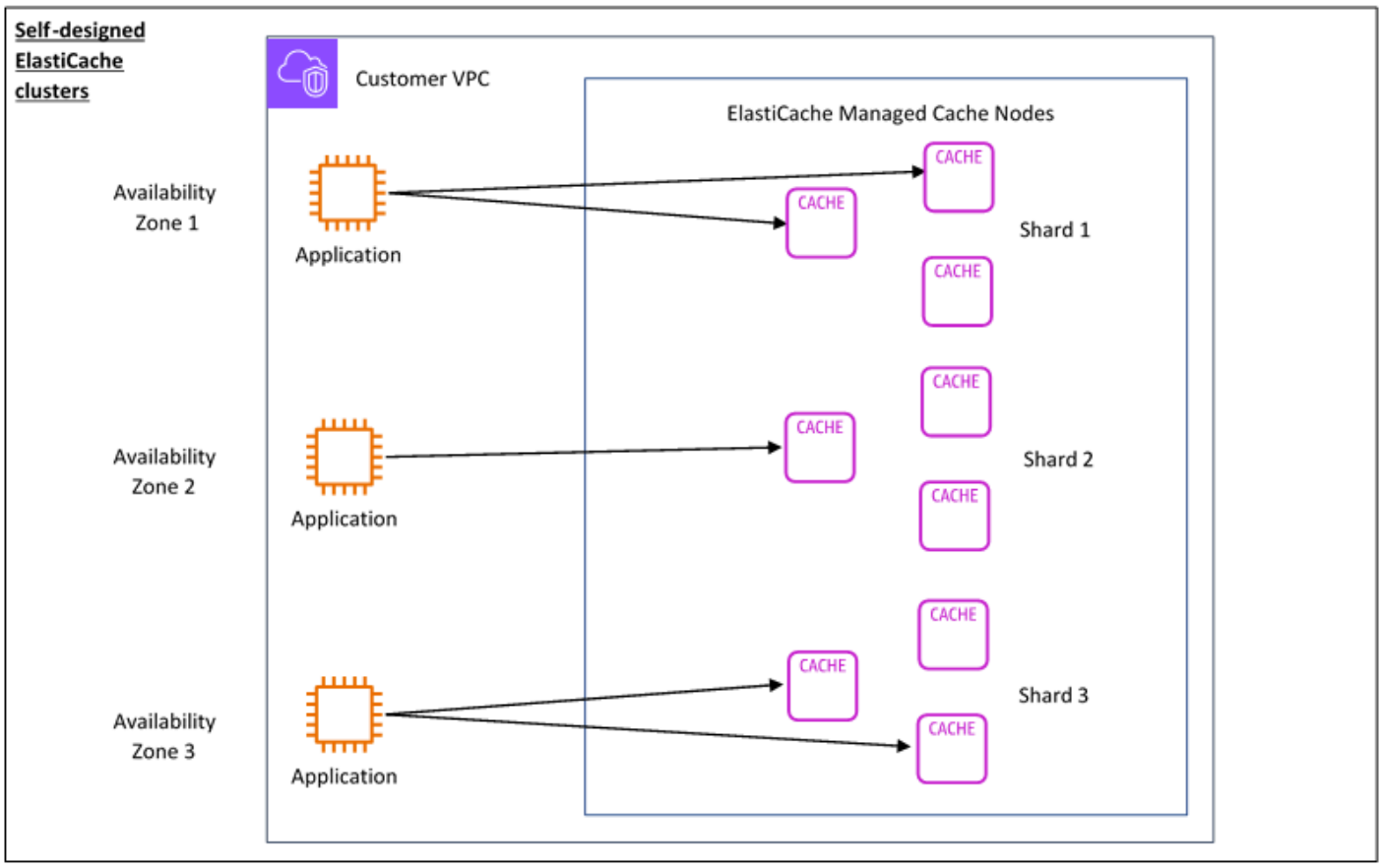
自行设计 ElastiCache 的集群

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。通过设计自己的集群，您可以更精细地控制集群的配置和扩展。

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，您可以使用使用参数组配置引擎参数 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。您也可以通过添加节点来横向扩展。

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 Memcached 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 会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。
- 对多个项目执行操作的命令将按比例消耗更多的 ECPU。例如，如果您的应用程序对 3 个项目执行 multiget，则它将消耗 3 个 ECPU。
- 操作更多项目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 GET 命令，检索 3 个项目并传输 3.2 KB 的数据，则它将使用 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 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、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。

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 Memcached 用户指南中的缓存策略

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。

ElastiCache for Memcached 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 for Memcached 的视频、常见使用案例以及说明如何使用 ElastiCache for Memcached 来减少延迟并提高应用程序吞吐量的演示。
- 入门 – Amazon ElastiCache for Memcached 入门 部分包含一个示例，介绍创建缓存群集、授予对缓存群集的访问权限、连接到缓存节点以及删除缓存群集的过程。
- 规模性能 – 利用 Amazon ElastiCache 实现规模性能白皮书介绍了能够使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 for Memcached 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）。

用于管理实施的工具

在向 Amazon EC2 实例授予对 ElastiCache 集群的访问权限后，您可使用以下四项来管理 ElastiCache 集群：AWS Management Console、AWS CLI for ElastiCache、AWS SDK for ElastiCache 和 ElastiCache API。

使用 AWS Management Console

AWS Management Console 是管理 Amazon ElastiCache for Memcached 的最简单方法。通过该控制台可以创建缓存集群、添加和移除缓存节点以及执行其他管理任务，而不必编写任何代码。此控制台还提供来自 CloudWatch 的缓存节点性能图，图上显示了缓存引擎活动、内存和 CPU 利用率以及其他指标。有关更多信息，请参阅此用户指南 中的特定主题。

使用 AWS CLI

您也可以使用 AWS Command Line Interface (AWS CLI) for ElastiCache。通过 AWS CLI 可轻松执行一次一个的操作，如启动或停止缓存集群。您还可以通过所选的脚本语言调用 AWS CLI for ElastiCache，从而自动执行重复任务。有关 AWS CLI 的更多信息，请参阅用户指南和 AWS CLI 命令参考。

使用 AWS SDK

如果要从应用程序访问 ElastiCache，则可以使用任一 AWS 软件开发工具包 (SDK)。开发工具包可包装 ElastiCache API 调用，将应用程序与 ElastiCache API 的低级别详细信息隔离。您提供证书，开发工具包库处理身份验证和请求签名。有关使用 AWS 软件开发工具包的更多信息，请参阅用于 Amazon Web Services 的工具。

使用 ElastiCache API

您还可以直接针对 ElastiCache Web 服务 API 编写应用程序代码。使用 API 时，您必须编写必要的代码以构建 HTTP 请求并验证其身份、分析来自 ElastiCache 的结果以及处理任何错误。有关该 API 的更多信息，请参阅使用 ElastiCache API。

另请参阅

有关管理 Amazon ElastiCache for Memcached 部署的更多详细信息，请参阅以下内容：

- 与 ElastiCache
- 互联网络流量隐私保护

- Amazon ElastiCache 中的日志记录和监控

选择部署选项

Amazon ElastiCache 有两个部署选项：

- 无服务器缓存
- 设计自己的集群

无服务器缓存

Amazon ElastiCache 无服务器简化了缓存的创建，并可即时扩展以支持客户具有苛刻要求的应用程序。借助 ElastiCache 无服务器，您可以在一分钟内创建高度可用且可扩展的缓存，而无需预置、规划和管理缓存群集容量。ElastiCache 无服务器自动跨三个可用区冗余存储数据，并提供 99.99% 可用性服务水平协议（SLA）。ElastiCache 提供跨可用区的自动数据复制，无需手动管理副本和自定义软件即可保持同步。

自行设计的集群

如果您需要对 ElastiCache for Memcached 集群进行精细控制，则可以选择通过 ElastiCache 设计自己的 Memcached 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。

选择部署选项

在以下情况下，选择无服务器缓存：

- 您正在为新的或未知的工作负载创建新缓存
- 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：

- 您已在运行 ElastiCache 无服务器，并且想更精细地控制运行 Memcached 的节点类型、节点数和节点放置。
- 您预计应用程序流量波动不会太大，或者您可以准确地预测应用程序流量的最大值和最小值。
- 您可以预测容量要求以控制成本。

相关主题:

- 设计和管理您自己的 ElastiCache for Memcached 集群

比较 Memcached 和 Redis 自行设计缓存

Amazon 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**
>
> 创建缓存群集或复制组后，您可以升级到较新的引擎版本，但不能降级到较早的引擎版本。如果您要使用较早的引擎版本，必须删除现有的缓存群集或复制组，并使用较早的引擎版本重新创建。

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。

如果以下情况适用于您，请选择带 ElastiCache for Redis 版本的 Redis：

- ElastiCache for Redis 版本 7.0（加强版）

  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 for Redis 版本 6.2（加强版）

  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 for Redis 版本 6.0（加强版）

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 for Redis 版本 5.0.0（加强版）

  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 for Redis 版本 4.0.10（加强版）

  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 for Redis 版本 3.2.10（加强版）

  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 for 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 for Redis 版本 3.2.6（加强版）

  如果您需要早期 Redis 版本的功能外加以下功能，请选择 ElastiCache for Redis 3.2.6：

  - 传输中加密。有关更多信息，请参阅 Amazon ElastiCache for Redis 传输中加密。

  - 静态加密。有关更多信息，请参阅 Amazon ElastiCache for Redis 静态加密。
- ElastiCache for 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 for Redis（非集群模式）2.8.x 和 3.2.4（加强版）

  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）
Engine versions+	1.4.5 and later	4.0.10 and later	4.0.10 and later
Data types	Simple	2.8.x - Complex * Complex	3.2.x and later - Complex
Data partitioning	Yes	No	Yes
Cluster is modifiable	Yes	Yes	3.2.10 and later - Limited
Online resharding	No	No	3.2.10 and later
Encryption	in-transit 1.6.12 and later	4.0.10 and later	4.0.10 and later
Data tiering	No	6.2 and later	6.2 and later
合规性认证			
Compliance Certification			
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
Multi-threaded	Yes	No	No
Node type upgrade	No	Yes	Yes
Engine upgrading	Yes	Yes	Yes
High availability (replication)	No	Yes	Yes

	Memcached	Redis（已禁用集群模式）	Redis（已启用集群模式）
Automatic failover	No	Optional	Required
Pub/Sub capabilities	No	Yes	Yes
Sorted sets	No	Yes	Yes
Backup and restore	No	Yes	Yes
Geospatial indexing	No	4.0.10 and later	Yes

注意:

string, objects (like databases)

* string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog

string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog, geospatial indexes

+ Excludes versions which are deprecated, have reached or soon to reach end of life.

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，请参阅支持的 ElastiCache for Memcached 版本或者支持的 ElastiCache for Redis 版本。

Amazon ElastiCache for Memcached 入门

本部分中的主题将引导您了解使用 ElastiCache 控制台，创建 Memcached 无服务器缓存、向其授予访问权限、进行连接和删除的过程。

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。作为安全最佳实操，请为管理用户分配管理访问权限，并且只使用根用户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 My Account（我的账户）来查看当前的账户活动并管理您的账户。

创建管理用户

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. 对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）。

创建管理用户

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，为管理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。

作为管理用户登录

- 要使用 IAM Identity Center 用户身份登录，请使用在创建 IAM Identity Center 用户时发送到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。

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 以使用。 • 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。
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

哪个用户需要编程式访问权限？	目的	方式
		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。按照《AWS IAM Identity Center 用户指南》中的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。

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 `iam:createServiceLinkedRole` 的权限或创建了 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 for Memcached 使用 11211 和 11212 端口接受 Memcached 命令。为了从您的 EC2 实例成功连接并执行 Memcached 命令，您的安全组必须允许访问这些端口。

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 11211。

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 11212。

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

第 1 步：创建缓存

您将要启动的缓存将是活动的，而且不在沙盒中运行。您需要为缓存支付标准 ElastiCache 使用费，直到您删除该缓存。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缓存，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。

创建无服务器缓存

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 AWS Management Console 并打开 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 Memcached 缓存。
3. 在控制台的右侧，选择创建 Memcached 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。

使用 AWS CLI 创建新的缓存

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 memcached
```

Windows

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

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。

默认情况下，ElastiCache 在您的默认 VPC 中创建缓存。请确保您同样在默认 VPC 中创建了 EC2 实例，以使实例能够连接到缓存。

查找缓存端点

AWS Management Console

要使用 ElastiCache 控制台查找缓存的端点，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 Memcached 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 命令查找新缓存的端点。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
```

使用 OpenSSL 进行连接

有关如何使用 OpenSSL 进行连接的信息，请参阅 ElastiCache 传输中加密（TLS）

使用 Memcached Java 客户端进行连接

有关如何使用 Memcached Java 客户端进行连接的信息，请参阅 ElastiCache 传输中加密（TLS）

使用 Memcached PHP 客户端进行连接

```
<?php
$cluster_endpoint = "mycluster.serverless.use1.cache.amazonaws.com";
$server_port = 11211;

/* Initialize a persistent Memcached client in TLS mode */
$tls_client = new Memcached('persistent-id');
$tls_client->addServer($cluster_endpoint, $server_port);
```

```
if(!$tls_client->setOption(Memcached::OPT_USE_TLS, 1)) {
    echo $tls_client->getLastErrorMessage(), "\n";
    exit(1);
}
$tls_config = new MemcachedTLSContextConfig();
$tls_config->hostname = '*.serverless.use1.cache.amazonaws.com';
$tls_config->skip_cert_verify = false;
$tls_config->skip_hostname_verify = false;
$tls_client->createAndSetTLSContext((array)$tls_config);

 /* store the data for 60 seconds in the cluster */
$tls_client->set('key', 'value', 60);
?>
```

使用 Memcached Python 客户端（Pymemcache）进行连接

请参阅 https://pymemcache.readthedocs.io/en/latest/getting_started.html

```
import ssl
from pymemcache.client.base import Client

context = ssl.create_default_context()
cluster_endpoint = <To be taken from the AWS CLI / console>
target_port = 11211
memcached_client = Client(("{cluster_endpoint}", target_port), tls_context=context)
memcached_client.set("key", "value", expire=500, noreply=False)
assert self.memcached_client.get("key").decode() == "value"
```

使用 Memcached NodeJS/TS 客户端（Electrode-IO memcache）进行连接

请参阅 https://github.com/electrode-io/memcache 和 https://www.npmjs.com/package/memcache-client

通过 `npm i memcache-client` 进行安装

在应用程序中，按以下所示创建 memcached TLS 客户端：

```
var memcache = require("memcache-client");
const client = new memcache.MemcacheClient({server: "{cluster_endpoint}:11211", tls:
 {}});
client.set("key", "value");
```

使用 Memcached Rust 客户端（rust-memcache）进行连接

请参阅 https://crates.io/crates/memcache 和 https://github.com/aisk/rust-memcache。

```
// create connection with to memcached server node:
let client = memcache::connect("memcache+tls://<cluster_endpoint>:11211?
verify_mode=none").unwrap();

// set a string value
client.set("foo", "bar", 0).unwrap();
```

使用 Memcached Go 客户端（Gomemcache）进行连接

请参阅 https://github.com/bradfitz/gomemcache

```
c := New(net.JoinHostPort("{cluster_endpoint}", strconv.Itoa(port)))
c.DialContext = func(ctx context.Context, network, addr string) (net.Conn, error) {
var td tls.Dialer
td.Config = &tls.Config{}
return td.DialContext(ctx, network, addr)
}
foo := &Item{Key: "foo", Value: []byte("fooval"), Flags: 123}
err := c.Set(foo)
```

使用 Memcached Ruby 客户端（Dalli）进行连接

请参阅 https://github.com/petergoldstein/dalli

```
require 'dalli'
ssl_context = OpenSSL::SSL::SSLContext.new
ssl_context.ssl_version = :SSLv23
ssl_context.verify_hostname = true
ssl_context.verify_mode = OpenSSL::SSL::VERIFY_PEER
client = Dalli::Client.new("<cluster_endpoint>:11211", :ssl_context => ssl_context);
client.get("abc")
```

使用 Memcached .NET 客户端（EnyimMemcachedCore）进行连接

请参阅 https://github.com/cnblogs/EnyimMemcachedCore

```
"MemcachedClient": {
"Servers": [
{
"Address": "{cluster_endpoint}",
"Port": 11211
}
],
"UseSslStream":  true
}
```

您现在可以继续执行第 3 步：（可选）清除。

第 3 步：（可选）清除

使用 AWS Management Console

以下过程从您的部署中删除单个缓存。要删除多个缓存，请对要删除的每个缓存重复此过程。您无需等待删除一个缓存的过程完成，即可开始删除另一个缓存。

删除缓存

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，选择您要删除的缓存当前所运行的引擎。此时会显示运行该引擎的所有缓存的列表。
3. 要选择缓存以便删除，请从缓存列表中选择缓存的名称。

   > **Important**
   >
   > 您从 ElastiCache 控制台一次只能删除一个缓存。选择多个缓存会禁用删除操作。

4. 对于操作，选择删除。
5. 在删除缓存确认屏幕上，选择删除可删除缓存，选择取消可保留集群。
6. 如果选择了删除，则缓存的状态将变为正在删除。

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。

使用 AWS CLI

下面的代码删除缓存 my-cache。

```
aws elasticache delete-serverless-cache --serverless-cache-name my-cache
```

delete-serverless-cache CLI 操作仅删除一个无服务器缓存。要删除多个缓存，请为要删除的每个无服务器缓存调用 delete-serverless-cache。您无需等待删除一个无服务器缓存的过程完成，即可开始删除另一个。

对于 Linux、macOS 或 Unix：

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

```
  --serverless-cache-name my-cache
```

对于 Windows：

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

有关更多信息，请参阅 AWS CLI 中的 ElastiCache 主题 delete-serverless-cache。

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：

- 与 ElastiCache
- 扩展 Mem ElastiCache cached
- ElastiCache 的配额
- ElastiCache 最佳实践和缓存策略
- 查看 ElastiCache 事件

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

在本教程中，您将执行以下操作：

- 在您 us-east-1 区域的默认 Amazon Virtual Private Cloud（Amazon VPC）中，创建 Amazon ElastiCache 缓存。
- 创建 Lambda 函数以访问 ElastiCache 缓存。在创建 Lambda 函数时，您需要提供 Amazon VPC 和 VPC 安全组中的子网 ID，以允许 Lambda 函数访问 VPC 中的资源。在本教程的图示中，Lambda 函数生成 UUID，将其写入到缓存，然后从缓存中检索。
- 手动调用 Lambda 函数，并确保其已访问您 VPC 中的 ElastiCache 缓存。

> **Important**
>
> 本教程使用了您账户的 us-east-1 区域中的默认 Amazon VPC。有关 Amazon VPC 的更多信息，请参阅 Amazon VPC 用户指南中的 Amazon VPC 入门指南。

主题

入门

第 1 步：创建 ElastiCache 缓存

第 1 步：创建 ElastiCache 缓存

在此步骤中，您在自己的账户中使用 AWS CLI，在 us-east-1 区域的默认 Amazon Virtual Private Cloud 中创建 Amazon ElastiCache 缓存。有关使用 ElastiCache 控制台或 API 创建 ElastiCache 无服务器缓存的信息，请参阅《ElastiCache for Memcached 用户指南》中的创建集群。

AWS Management Console

运行以下 AWS CLI 命令，在 us-east-1 区域的默认 VPC 中创建新的 Memcached 集群无服务器缓存。

Linux

```
aws elasticache create-serverless-cache \
--serverless-cache-name serverlessCacheForLambda \
--region us-east-1 \
--engine memcached
```

Windows

```
aws elasticache create-serverless-cache ^
--serverless-cache-name serverlessCacheForLambda ^
--region us-east-1 ^
--engine memcached
```

请注意，“状态”字段的值设置为 CREATING。ElastiCache 需要几分钟时间来完成集群的创建。

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 serverlessCacheforLambda \
--region us-east-1
```

Windows

```
aws elasticache describe-serverless-caches ^
--serverless-cache-name serverlessCacheforLambda ^
--region us-east-1
```

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。

创建新缓存后，继续执行步骤 2：创建 Lambda 函数。

下一步：

步骤 2：创建 Lambda 函数

步骤 2：创建 Lambda 函数

在此步骤中，您将执行以下操作：

1. 使用提供的示例代码创建 Lambda 函数部署包。
2. 创建 IAM 角色（执行角色）。在上传部署包时，需要指定此角色以便 Lambda 代入该角色，然后代表您执行该函数。权限策略将授予 AWS Lambda 设置弹性网络接口（ENI）的权限，从而使您的 Lambda 函数能够访问 VPC 中的资源。在本示例中，您的 Lambda 函数将访问 VPC 中的 ElastiCache 集群。
3. 通过上传部署包来创建 Lambda 函数。

下一步

步骤 2.1：创建部署包

步骤 2.1：创建部署包

目前 Lambda 函数的示例代码仅在 Python 中提供。

Python

以下 Python 代码示例将在 ElastiCache 集群中读取和写入项目。复制代码，并将其保存到名为 `app.py` 的文件中。请确保将代码中的 `elasticache_config_endpoint` 值替换为您在步骤 1 中复制的端点地址。

```
import uuid
import ssl
from pymemcache.client.base import Client

elasticache_config_endpoint = "serverlesscacheforlambda-
ces85m.serverless.use1.cache.amazonaws.com"
target_port = 11211

context = ssl.create_default_context()

memcached_client = Client((elasticache_config_endpoint, target_port),
 tls_context=context)

def lambda_handler(event, context):

    # create a random UUID - this will be the sample element we add to the cache
    uuid_in = uuid.uuid4().hex

    # put the UUID to the cache
    memcached_client.set("uuid", uuid_in, expire=500, noreply=False)
```

```
    # get the item (UUID) from the cache
    result = memcached_client.get("uuid")
    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 Memcached.")
    else:
        raise Exception(f"Bad value retrieved. Expected {uuid_in}, got
{decoded_result}")

    return "Fetched value from Memcached"
```

此代码使用 Python pymemcache 库将项目放入缓存并检索它们。要创建包含 pymemcache 的部署包，请执行以下步骤。

1. 在包含 app.py 源代码文件的项目目录中，创建一个 package 文件夹，用于在其中安装 pymemacache 库。

   ```
   mkdir package
   ```

2. 使用 pip 安装 pymemcache。

   ```
   pip install --target ./package pymemcache
   ```

3. 创建包含 pymemcache 库的 .zip 文件。在 Linux 和 macOS 中，运行以下命令：在 Windows 中，使用首选 zip 实用工具创建一个 .zip 文件，并将 pymemache 库置于根目录下。

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

4. 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，使用您首选的 zip 实用工具将 app.py 添加到 .zip 文件的根目录下。

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

下一步

步骤 2.2：创建 IAM 角色（执行角色）

步骤 2.2：创建 IAM 角色（执行角色）

在此步骤中，您使用以下预定义的角色类型和访问策略创建 AWS Identity and Access Management（IAM）角色：

- AWS Lambda 类型的 AWS 服务角色 – 此角色为 AWS 授予 Lambda 权限以担任该角色。
- AWSLambdaVPCAccessExecutionRole – 这是您附加到角色的访问权限策略。该策略授予对 EC2 操作的权限，AWS Lambda 需要该权限来管理 ENI。您可以在 IAM 控制台中查看此 AWS 托管策略。

有关 IAM 用户角色的更多信息，请参阅 IAM 用户指南中的角色（委托和联合）。

使用以下程序创建 IAM 角色。

创建 IAM（执行）角色

1. 登录 AWS 管理控制台，并通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择 Roles（角色），然后选择 Create role（创建角色）。
   - 在可信实体类型下，选择 AWS 服务，然后在使用案例下选择 Lambda。这将为 AWS Lambda 服务授予承担此角色的权限。选择下一步。
   - 在添加权限下，搜索 **AWSLambdaVPCAccessExecutionRole** 并选中策略旁边的复选框。
   - 选择下一步。
   - 在 Role Name（角色名称）中，使用在 AWS 账户内唯一的名称（例如，lambda-vpc-execution-role）。
   - 选择 Create role（创建角色）。
3. 复制角色 ARN。在下一步中，创建 Lambda 函数时需要用到它。

下一步

步骤 2.3：上传部署包（创建 Lambda 函数）

步骤 2.3：上传部署包（创建 Lambda 函数）

在此步骤中，您将使用 `create-function` AWS CLI 命令创建 Lambda 函数（`AccessMemcached`）。

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

对于 `role` 选项，请使用您在步骤 2.2 中创建的执行角色的 ARN。对于 `vpc-config`，请输入您的默认 VPC 子网列表和默认 VPC 安全组 ID 的列表，以逗号分隔。您还可以在 Amazon VPC 控制台中找到这些值。要查找您的默认 VPC 子网，请选择您的 VPC，然后选择 AWS 账户的默认 VPC。要查找此 VPC 的安全组，请在安全性下选择安全组。请确保您选择了 us-east-1 区域。

对于 Linux、macOS 或 Unix：

```
aws lambda create-function \
--function-name AccessMemcached  \
--region us-east-1 \
--zip-file fileb://my_deployment_package.zip \
--role arn:aws:iam::123456789012:role/lambda-vpc-execution-role \
--handler app.lambda_handler \
--runtime python3.11  \
--timeout 30 \
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-security-group-id \
```

对于 Windows：

```
aws lambda create-function ^
--function-name AccessMemcached  ^
--region us-east-1 ^
--zip-file fileb://path-to/my_deployment_package.zip ^
--role arn:aws:iam::123456789012:role/lambda-vpc-execution-role ^
--handler app.lambda_handler ^
--runtime python3.11  ^
--timeout 30 ^
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-security-group-id ^
```

或者，您也可以将 .zip 文件上传到同一 AWS 区域中的 Amazon S3 桶，然后在之前的命令中指定该桶和对象名称。您需要将 `--zip-file` 参数替换为 `--code` 参数，如下所示：

```
--code S3Bucket=bucket-name,S3Key=zip-file-object-key
```

您也可以使用 AWS Lambda 控制台创建 Lambda 函数。在创建函数时，为 Lambda 选择 VPC，然后从提供的字段中选择子网和安全组。

下一步

步骤 3：测试 Lambda 函数

步骤 3：测试 Lambda 函数

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

1. 使用 AWS Lambda `invoke` 命令调用 Lambda 函数（`AccessMemCache`）。

   对于 Linux、macOS 或 Unix：

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

   对于 Windows：

   ```
   aws lambda invoke ^
   --function-name AccessMemCache  ^
   --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 控制台并为您的函数（`/aws/lambda/AccessMemcached`）选择日志组，在 CloudWatch Logs 中验证结果。日志流应包含类似于以下内容的输出：

     ```
     Success: Inserted 05fcf2e4d6c942209acc89ea79b5b15e. Fetched
      05fcf2e4d6c942209acc89ea79b5b15e from Memcached.
     ```

   - 在 AWS Lambda 控制台中查看结果。

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

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 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)
- 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)

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)。

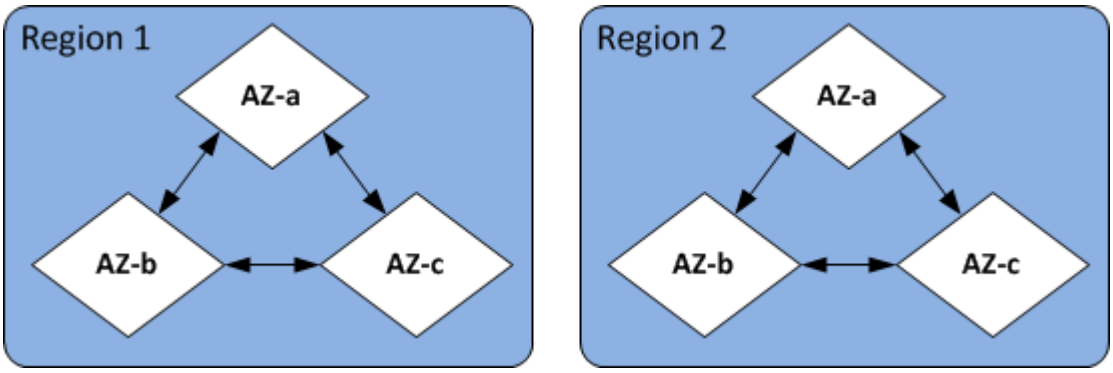
选择区域和可用区

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

- 可用区注意事项
- 支持的区域和端点
- 找到您的节点
- 将 Local Zones 与 ElastiCache 结合使用
- 使用 Outposts

可用区注意事项

将 Memcached 节点分配到一个区域内的多个可用区有助于保护您免受灾难性故障（如可用区内断电）的影响。

无服务器缓存

ElastiCache 无服务器缓存可创建跨越多个可用区域的高可用性缓存。您可以在创建缓存时指定来自不同可用区和相同 VPC 的子网，也可以自动从ElastiCache 默认 VPC 中选择子网。

ElastiCache 为 Memcached 集群设计自己的集群

一个 Memcached 集群最多可拥有 300 个节点。在您的 Memcached 集群中创建或添加节点时，您可以为所有节点指定一个可用区， ElastiCache 允许为所有节点选择一个可用区，为每个节点指定可用区，或者 ElastiCache 允许为每个节点选择一个可用区。当您将新节点添加到现有 Memcached 群集时，可以在不同的可用区中创建新节点。创建缓存节点后，无法修改其可用区。

如果您希望单个可用区集群中的集群的节点分布在多个可用区中，则 ElastiCache 可以在不同的可用区中创建新节点。然后，您可以删除部分或全部原始缓存节点。我们建议采用此方法。

将 Memcached 节点从单一可用区迁移到多个可用区

1. 通过在所需的可用区中创建新的缓存节点来修改您的集群。在您的请求中，执行以下操作：

   - 将 `AZMode`（CLI：`--az-mode`）设置为 `cross-az`。
   - 将 `NumCacheNodes`（CLI：`--num-cache-nodes`）设置为当前活动缓存节点数加上您要创建的新缓存节点数。
   - 将 `NewAvailabilityZones`（CLI：`--new-availability-zones`）设置为要在其中创建新缓存节点的区域列表。要ElastiCache 确定每个新节点的可用区，请不要指定列表。
   - 将 `ApplyImmediately`（CLI：`--apply-immediately`）设置为真。

   > **Note**
   >
   > 如果您未使用自动发现，请确保使用新的缓存节点端点更新客户端应用程序。

   继续执行下一步之前，请确保 Memcached 节点已完全创建且可用。
2. 通过删除原始可用区中不再需要的节点来修改您的集群。在您的请求中，执行以下操作：

- 将 NumCacheNodes（CLI：--num-cache-nodes）设置为应用此修改后所需的活动缓存节点数。
- 将 CacheNodeIdsToRemove（CLI：--nodes-to-remove）设置为要从集群中删除的缓存节点列表。

  列出的缓存节点 ID 的数量必须等于当前活动节点的数量减去NumCacheNodes。
- （可选）将 ApplyImmediately（CLI：--apply-immediately）设置为真。

  如果您未将 ApplyImmediately（CLI：--apply-immediately）置为真，则将在您的下一个维护时段中进行节点删除。

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 us-east-1	elasticache.us-east-1.amazonaws.com	HTTPS	
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	

区域名称/区域	终端节点	协议	
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	
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	

区域名称/区域	终端节点	协议	
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	
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	

区域名称/区域	终端节点	协议	
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	
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	

区域名称/区域	终端节点	协议	
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。

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。

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

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：

- 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 Memcached 集群。

   有关更多信息，请参阅创建 Memcached 集群（控制台）。

使用 Outposts

AWS 是一项完全托管式服务，可将 AWS 基础设施、服务、API 和工具扩展到客户场所。通过提供对 AWS 托管式基础设施的本地访问，AWS Outposts 使客户能够使用与 AWS 区域中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满足更低的延迟和本地数据处理需求。Outpost 是部署在客户站点的 AWS 计算和存储容量池。AWS 作为 AWS 区域的一部分运营、监控和管理此容量。您可以在 Outpost 上创建子网，并在创建 ElastiCache 集群等 AWS 资源时指定这些子网。

> **Note**
>
> 在此版本中，以下限制适用：
>
> - 用于 Outposts 的 ElastiCache 仅支持 M5 和 R5 节点系列。
> - 多可用区（不支持跨站点复制）。

- 以下区域不支持 ElastiCache for Outposts：cn-north-1、cn-northwest-1 和 ap-northeast-3。

将 Outposts 与 Memcached 控制台结合使用

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 Memcached。
3. 在 Cluster Engine（集群引擎）中，选择 Memcached。
4. 在 Location（位置）下，选择 On-Premises - Create your ElastiCache instances on AWS Outposts（本地 - 在 Amazon Outposts 上创建 ElastiCache 实例）。

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 Memcached settings（Memcached 设置）下：
   a. Name（名称）：输入 Memcached 集群的名称
   b. Description（描述）：输入 Memcached 集群的描述。
   c. Engine version compatilbility（引擎版本兼容性）：引擎版本基于 AWS Outpost 区域
   d. Port（端口）：接受默认端口 11211。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。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 nodes（节点数）：输入集群中所需的节点数。
2. 在 Advanced Memcached settings（高级 Memcached 设置）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。
- 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 Create (创建)。

查看 Outpost 集群详细信息

在 Memcached 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几点：

- 可用区：这将代表 Outpost，使用 ARN（Amazon 资源名称）和 AWS 资源编号。
- Outpost 名称：AWS Outpost 的名称。

将 Outposts 与 AWS CLI 结合使用

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 用户指南中安装 AWS CLI 和配置 AWS CLI 的说明进行操作。

将 AWS CLI 与 Outposts 结合使用

使用以下 CLI 操作创建使用 Outposts 的缓存集群：

- create-cache-cluster – 使用此操作时，`outpost-mode` 参数会接受一个值，该值指定缓存集群中的节点是在单一 Outpost 中创建还是在多个 Outpost 中创建。

> **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 \
```

设计和管理您自己的 ElastiCache for Memcached 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 Memcached 组件和功能

接下来，您可以找到亚马逊版 Memcached 部署的主要组件 ElastiCache 的概述。

主题

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 Memcached 的一个实例。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，请参阅扩展 Mem ElastiCache cached。

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。您也可以大大降低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，您可以购买多个预留节点以在大多数时间运行，并在偶尔需要添加 pay-as-you-go 节点的时间购买节点。有关预留节点的更多信息，请参阅ElastiCache 预留节点。

Memcached 引擎支持 Auto Discovery。自动发现能使客户端程序自动识别缓存群集中的所有节点，并且启动和维护所有这些节点的连接。有了 Auto Discovery，您的应用程序无需手动连接至各个节点。相

反，您的应用程序连接到配置终端节点。配置终端节点 DNS 条目包含各个缓存节点终端的 CNAME 条目。因此，通过连接到配置终端节点，您的应用程序便可立即知道集群中所有节点的信息，同时能够连接到所有节点。您无需对应用程序中的单个缓存节点终端节点进行硬编码。有关更多信息，请参阅自动发现。

有关节点的更多信息，请参阅管理节点。

ElastiCache 适用于内存缓存集群

Memcached 集群 是一个或多个ElastiCache 节点的逻辑分组。跨 Memcached 集群中的节点对数据进行分区。

许多 ElastiCache 操作都是针对集群的：

- 创建集群
- 修改集群
- 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。

  如果您需要超过这些限制，请使用 Amazon ElastiCache 缓存节点申请表进行申请。
- 缓解故障

  有关增强集群的容错能力的信息。

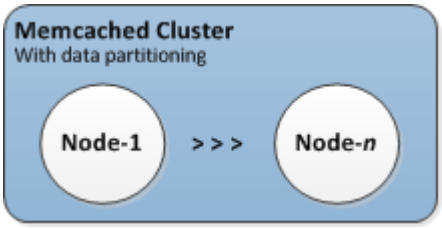
典型集群配置

Memcached 为每个 AWS 区域的每个客户最多支持 300 个节点，每个集群有 1-60 个节点。您可以跨 Memcached 集群中的节点对数据进行分区。

当你运行 Memcached 引擎时，集群可以由 1-60 个节点组成。您可以将数据库分配到多个节点上。应用程序会对每个节点的终端节点进行读写操作。有关更多信息，请参阅自动发现。

为了提高容错能力，请在集群区域内的各个可用区 (AZ) 中找到 Memcache AWS d 节点。这样一来，可最大程度地减小某个可用区内的故障对整个集群和应用程序的影响。有关更多信息，请参阅缓解故障。

由于 Memcached 集群需求不断变化的，您可通过添加或删除节点来进行向外扩展或向内扩展，从而跨新数量的节点对数据进行重新分区。对数据进行分区时，建议使用一致性哈希处理。有关一致性哈希处理的更多信息，请参阅配置 ElastiCache 客户端以实现高效负载均衡。在下图中，您可以看到单节点和多节点 Memcached 集群的示例。

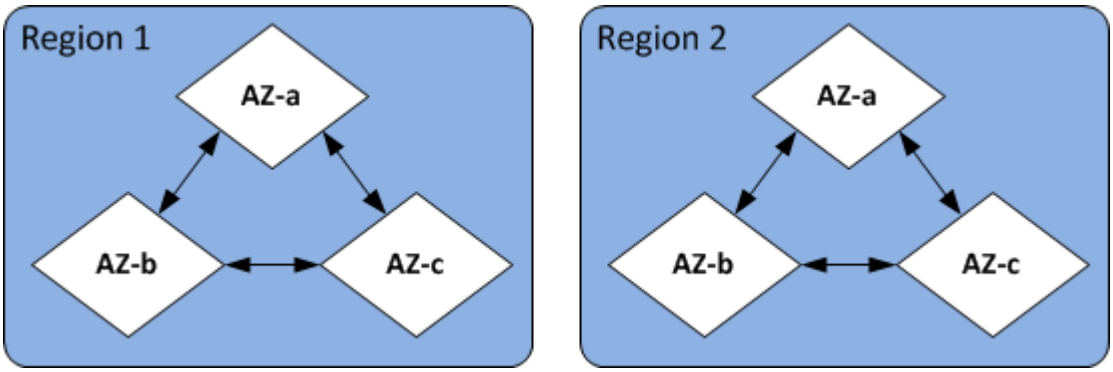


AWS 地区和可用区

Amazon ElastiCache for Memcached 已在全球多个 AWS 地区推出。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于您的 HTTP 请求、 AWS 软件开发工具包和 ElastiCache 控制台。 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。



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 用于内存缓存终端节点

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。

Memcached 集群中的每个节点都有自己的终端节点。该集群还具有一个称为配置终端节点 的终端节点。如果您启用 Auto Discovery 并连接到配置终端节点，则即使是在集群中添加或移除节点后，应用程序仍将自动获知每个节点终端节点。有关更多信息，请参阅自动发现。

有关更多信息，请参阅 终端节点。

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 (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 安全

为了增强安全性，只有在列入白名单的 Amazon EC2 实例上运行的应用程序才能访问 ElastiCache 节点。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 集群都是在亚马逊虚拟私有云 (Amazon 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。如果您选择在 Amzon VPC 外部运行您的集群，则可以创建安全组以向特定 Amazon EC2 安全组中运行的 Amazon EC2 实例授权。

ElastiCache 子网组

子网组是您可为在 Amazon Virtual Private Cloud (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 Amazon VPC 和 ElastiCache 安全性、授权访问和子网和子网组。

ElastiCache 用于内存缓存事件

当缓存集群上发生重大事件时， ElastiCache 会向特定的 Amazon SNS 主题发送通知。重要事件可以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，并且根据事件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。

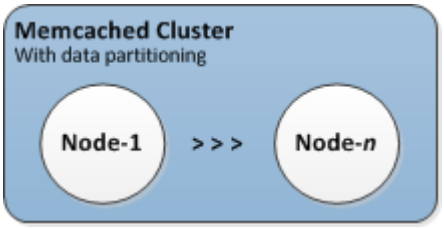
管理集群

集群 是一个或多个缓存节点的集合，其中所有节点都运行 Memcached 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 Memcached 集群。Memcached 集群包含 1 到 60 个节点，您可以将数据横向分区。

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）。

典型的 Memcached 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。此标识符用于指定与 ElastiCache API 和 AWS CLI 命令交互时的特定集群。集群标识符对于该客户在一个 AWS 区域中必须是唯一的。

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，请参阅用于访问 Amazon VPC 中的 ElastiCache 缓存的访问模式。

有关支持的 Memcached 版本列表，请参阅 M emcached 版本支持 ElastiCache 。

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 Memcached 引擎版本 1.6.6 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

创建缓存集群并选择双栈作为网络类型时，您需要指定 IP 发现类型，即 IPv4 或 IPv6。ElastiCache 会将网络类型和 IP 发现默认为 IPv6，但这是可以更改的。如果您使用 Auto Discovery，则仅将所选 IP 类型的 IP 地址返回给 Memcached 客户端。

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，但双栈仍然有益于自动发现，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 `config get cluster` 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

使用在连接下创建缓存集群时，选择网络类型，即 IPv4、IPv 6 或双堆栈。AWS Management Console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，请参阅创建 Memcached 集群（控制台）。

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 memcached \
    --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 memcached ^
    --cache-node-type cache.m5.large ^
    --num-cache-nodes 1 ^
    --network-type dual_stack ^
    --ip-discovery ipv4
```

自动识别集群中的节点

对于运行 Memcached 引擎的集群，ElastiCache 支持自动发现能使客户端程序自动识别缓存群集中的所有节点，并且启动和维护所有这些节点的连接。

> **Note**
>
> 已为 Amazon ElastiCache Memcached 上运行的缓存群集添加自动发现。

借助 Auto Discovery，您的应用程序无需手动连接至单个缓存节点；相反，您的应用程序连接至一个 Memcached 节点并检索节点列表。通过该列表，应用程序可知道集群中的其余节点并能连接至其中的任一节点。您无需对应用程序中的单个缓存节点端点进行硬编码。

如果您在集群上使用双堆栈网络类型，自动发现将仅返回 IPv4 或 IPv6 地址，具体取决于您选择的地址。有关更多信息，请参阅 选择网络类型。

集群中的所有缓存节点都会保留一份与所有其他节点有关的元数据列表。每当在集群中添加节点时或从集群中移除节点时，此类元数据都会进行更新。

主题

Auto Discovery 的优势

Auto Discovery 具备下列优点：

- 当您增加缓存集群中的节点数时，新节点会向配置终端节点和所有其他节点进行注册。当您从缓存集群中移除节点时，分离的节点会取消注册自己。在这两种情况下，集群中的所有其他节点都会使用最新的缓存节点元数据进行更新。
- 缓存节点故障会被自动检测到；故障节点会被自动替换。

> **Note**
>
> 在节点替换完成之前，节点仍将出现故障。

- 客户端程序只需连接至配置终端节点。之后，Auto Discovery 库会连接至集群中的所有其他节点。
- 客户端程序每分钟轮询集群一次（如有必要，可调整此时间间隔）。如果对集群配置做出任何更改（例如新添或删除节点），客户端会收到一份更新的元数据列表。然后，客户端会根据需要连接至这些节点或与这些节点断开。

在所有 ElastiCache Memcached 缓存集群上启用 Auto Discovery。您无需重启您的任何缓存节点，即可使用此功能。

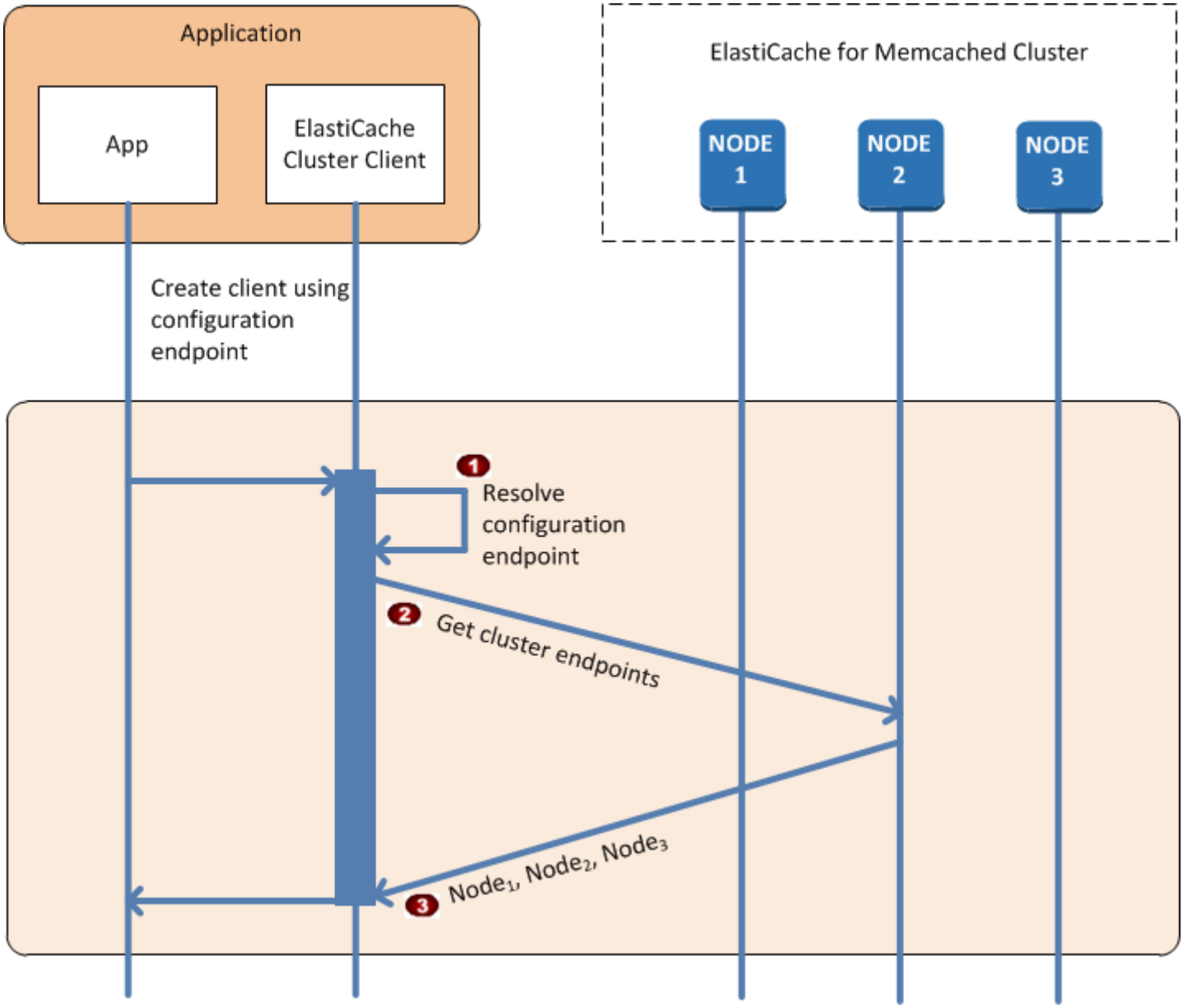
Auto Discovery 如何发挥作用

主题

本部分介绍了客户端应用程序如何使用 ElastiCache Cluster Client 来管理缓存节点连接，并与缓存中的数据项目互动。

连接至缓存节点

从应用程序的角度来看，连接至集群配置终端节点与直接连接至个别缓存节点并无差别。下面的序列图显示了连接至缓存节点的流程。



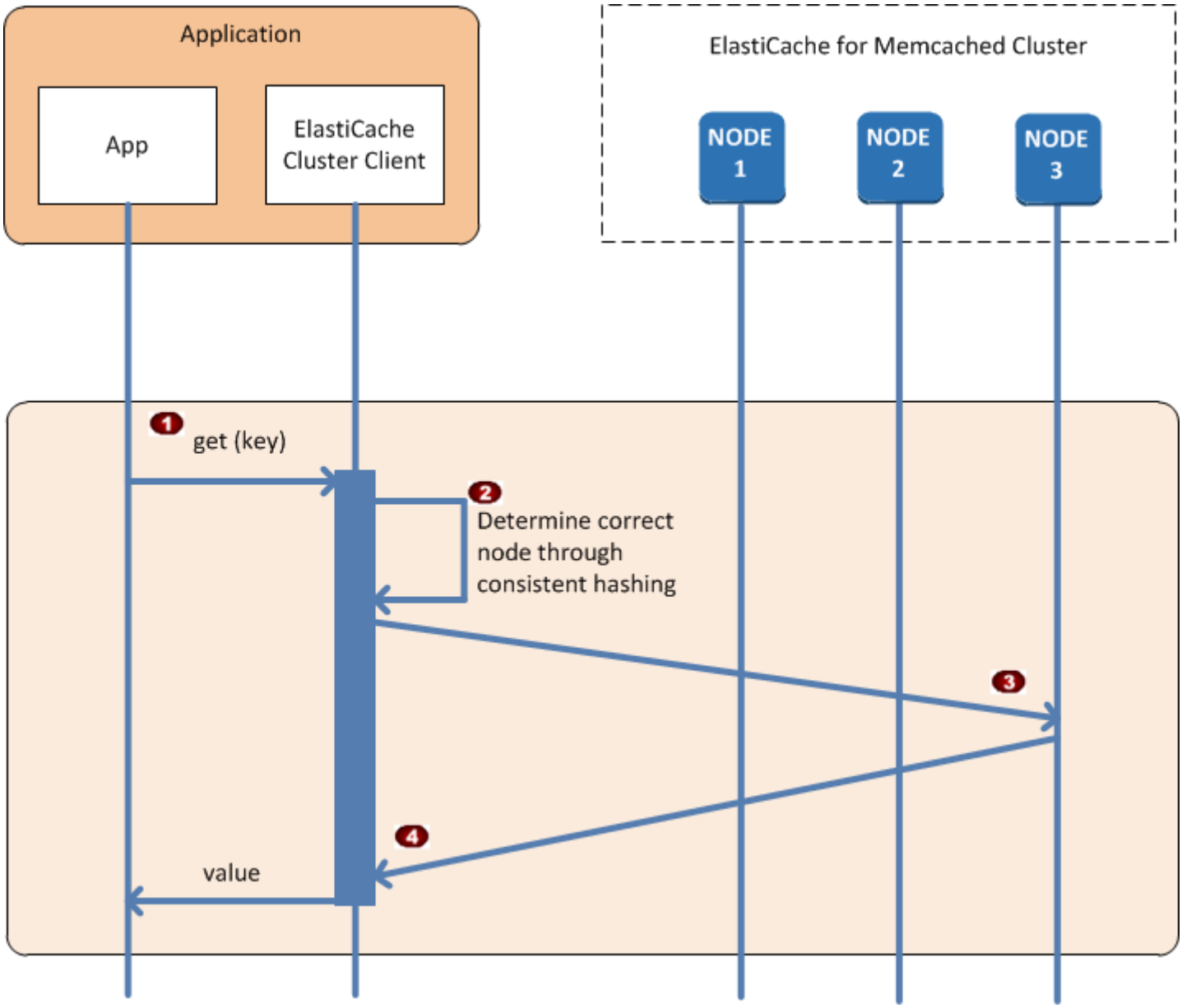
连接至缓存节点的流程

- 应用程序可以解析配置终端节点的 DNS 名称。由于配置终端节点为所有缓存节点保留 CNAME 条目，因此 DNS 名称被定为其中的一个节点；然后，客户端便可连接至这个节点。
- 客户端需要所有其他节点的配置信息。由于每个节点都为集群中的所有节点保留配置信息，因此任何节点都可以根据请求把配置信息传给客户端。
- 客户端则可收到当前的缓存节点主机名和 IP 地址列表。随后，它可以连接至集群中的所有其他节点。

> **Note**
>
> 客户端程序每分钟刷新一次缓存节点主机名和 IP 地址列表。如有必要，可以调整这个轮询间隔时间。

正常集群操作

当应用程序已连接至所有缓存节点时，ElastiCache Cluster Client 可以确定哪些节点应该存储单个数据项，以及哪些节点稍后应该用来查询此类数据项目。下面的序列图显示了正常集群操作的流程。



正常集群操作的流程

- 应用程序发出一个 get 请求，旨在获取一个由其密钥识别的特定数据项。
- 客户端针对密钥采用一种哈希算法，以确定哪一个缓存节点包含数据项。
- 数据项是从相关节点处请求的。
- 数据项会返还至应用程序。

其他操作

在某些情况下，您可能会更改集群的节点。例如，您可以添加附加节点来满足其他需求，或者删除节点以在需求减少期间节省资金。或者，您可能会因某种或另一种节点故障而替换节点。

当集群中发生需要对集群的终端节点进行元数据更新的更改时，将同时对所有节点进行此更改。因此，任何给定节点中的元数据将与集群中的所有其他节点中的元数据保持一致。

在所有这些情况下，元数据在所有节点间始终保持一致，因为将同时为集群中的所有节点更新元数据。您应始终使用配置终端节点来获取集群中各个节点的终端节点。通过使用配置终端节点，可确保您不会从对您“不可见”的节点获取终端节点数据。

添加节点

在启动节点的过程中，节点的终端节点不包含在元数据中。一旦该节点可用，就会将它添加到集群的每个节点的元数据中。在此方案中，元数据在所有节点间保持一致，并且您仅在新节点可用后能够与之交互。在节点可用之前，您不会知道该节点，并且您将与集群中的节点进行交互，就好像新节点不存在一样。

删除节点

在删除某个节点时，先从元数据中删除该节点的终端节点，然后从集群中删除该节点。在此方案中，所有节点中的元数据保持一致，如果要删除的节点不可用，则任何时候元数据都不会包含该节点的终端节点。在删除节点期间，将不会在元数据中进行报告，您的应用程序仅与其余 n-1 个节点进行交互，就好像该节点不存在一样。

替换节点

如果某个节点失败，ElastiCache 将中断该节点并启动替换节点。替换过程需要花费几分钟的时间。在此期间，所有节点中的元数据仍将显示失败节点的终端节点，但任何尝试与该节点进行交互的操作都将失败。因此，您的逻辑应始终包含重试逻辑。

使用 Auto Discovery

如要开始使用 Auto Discovery，请遵循下述步骤：

- 步骤 1：获取配置终端节点
- 步骤 2：下载 ElastiCache Cluster Client
- 步骤 3：修改您的应用程序

步骤 1：获取配置终端节点

如要连接至某个集群，客户端程序必须知道集群的配置终端节点。请参阅主题查找集群的端点（控制台）。

您也可以使用带有 `--show-cache-node-info` 参数的 `aws elasticache describe-cache-clusters` 命令：

不论您使用什么方法查找集群的终端节点，配置终端节点的地址中始终有 .cfg。

Example 使用适用于 ElastiCache 的 AWS CLI 查找端点

对于 Linux、macOS 或 Unix：

```
aws elasticache describe-cache-clusters \
    --cache-cluster-id mycluster \
    --show-cache-node-info
```

对于 Windows：

```
aws elasticache describe-cache-clusters ^
    --cache-cluster-id mycluster ^
    --show-cache-node-info
```

此操作将生成类似于以下内容的输出（JSON 格式）：

```
{
    "CacheClusters": [
        {
            "Engine": "memcached",
            "CacheNodes": [
                {
                    "CacheNodeId": "0001",
```

```
                    "Endpoint": {
                        "Port": 11211,
                        "Address": "mycluster.fnjyzo.cfg.0001.use1.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-10-12T21:39:28.001Z",
                    "CustomerAvailabilityZone": "us-east-1e"
                },
                {
                    "CacheNodeId": "0002",
                    "Endpoint": {
                        "Port": 11211,
                        "Address": "mycluster.fnjyzo.cfg.0002.use1.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-10-12T21:39:28.001Z",
                    "CustomerAvailabilityZone": "us-east-1a"
                }
            ],
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "CacheParameterGroupName": "default.memcached1.4",
                "ParameterApplyStatus": "in-sync"
            },
            "CacheClusterId": "mycluster",
            "PreferredAvailabilityZone": "Multiple",
            "ConfigurationEndpoint": {
                "Port": 11211,
                "Address": "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com"
            },
            "CacheSecurityGroups": [],
            "CacheClusterCreateTime": "2016-10-12T21:39:28.001Z",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterStatus": "available",
            "NumCacheNodes": 2,
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "CacheSubnetGroupName": "default",
            "EngineVersion": "1.4.24",
            "PendingModifiedValues": {},
            "PreferredMaintenanceWindow": "sat:06:00-sat:07:00",
            "CacheNodeType": "cache.r3.large"
```

```
        }
    ]
}
```

步骤 2：下载 ElastiCache Cluster Client

如要利用 Auto Discovery，客户端程序必须使用 ElastiCache Cluster Client。ElastiCache Cluster Client 可用于 Java、PHP 和 .NET，其中包含适用于发现和连接至您的所有缓存节点所需的全部逻辑。

下载 ElastiCache Cluster Client

1. 登录 AWS 管理控制台并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在 ElastiCache 控制台上，选择 ElastiCache Cluster Client，然后选择 Download（下载）。

ElastiCache Cluster Client for Java 的源代码可在 https://github.com/amazonwebservices/aws-elasticache-cluster-client-memcached-for-java 获取。此库是以常见的 Spymemcached 客户端为基础。ElastiCache Cluster Client 根据亚马逊软件许可 https://aws.amazon.com/asl 发布。您可以按照自己的方式自由修改源代码。您甚至可以将代码并入其他开放源 Memcached 库中或者您自己的客户端代码中。

> **Note**
>
> 如要使用适用于 PHP 的 ElastiCache Cluster Client，您首先需要将其安装在您的 Amazon EC2 实例上。有关更多信息，请参阅安装适用于 PHP 的 ElastiCache Cluster Client。
> 对于支持 TLS 的客户端，请下载 PHP 版本 7.4 或更高版本的二进制文件。
> 若要使用适用于 .NET 的 ElastiCache Cluster Client，您首先需要将其安装在您的 Amazon EC2 实例上。有关更多信息，请参阅安装适用于 .NET 的 ElastiCache Cluster Client。

步骤 3：修改您的应用程序

修改您的应用程序，以便它可以使用 Auto Discovery。以下部分介绍了如何使用适用于 Java、PHP 和 .NET 的 ElastiCache Cluster Client。

Important

指定集群的配置终端节点时，请确保其地址中有 ".cfg"，如此处所示。请勿使用 CNAME 或其中没有 ".cfg" 的终端节点。

```
"mycluster.fnjyzo.cfg.use1.cache.amazonaws.com";
```

未明确指定集群的配置终端节点会导致配置到特定节点。

使用适用于 Java 的 ElastiCache Cluster Client

下述程序演示了如何使用 ElastiCache Cluster Client 以连接到集群配置端点，并将数据项目添加至缓存中。借助 Auto Discovery，程序可在没有任何进一步干预的情况下连接至集群中的所有节点。

```
package com.amazon.elasticache;

import java.io.IOException;
import java.net.InetSocketAddress;

// Import the &AWS;-provided library with Auto Discovery support
import net.spy.memcached.MemcachedClient;

public class AutoDiscoveryDemo {

    public static void main(String[] args) throws IOException {

        String configEndpoint = "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com";
        Integer clusterPort = 11211;

        MemcachedClient client = new MemcachedClient(
                                 new InetSocketAddress(configEndpoint,
                                                       clusterPort));
        // The client will connect to the other cache nodes automatically.

        // Store a data item for an hour.
        // The client will decide which cache host will store this item.
        client.set("theKey", 3600, "This is the data value");
    }
}
```

使用适用于 PHP 的 ElastiCache Cluster Client

下述程序演示了如何使用 ElastiCache Cluster Client 以连接到集群配置端点，并将数据项目添加至缓存中。借助 Auto Discovery，程序将在没有任何进一步干预的情况下连接至集群中的所有节点。

如要使用适用于 PHP 的 ElastiCache Cluster Client，您首先需要将其安装在您的 Amazon EC2 实例上。有关更多信息，请参阅安装适用于 PHP 的 ElastiCache Cluster Client。

```
<?php

 /**
  * Sample PHP code to show how to integrate with the Amazon ElastiCache
  * Auto Discovery feature.
  */

  /* Configuration endpoint to use to initialize memcached client.
   * This is only an example.  */
  $server_endpoint = "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com";

  /* Port for connecting to the ElastiCache cluster.
   * This is only an example  */
  $server_port = 11211;

 /**
  * The following will initialize a Memcached client to utilize the Auto Discovery
 feature.
  *
  * By configuring the client with the Dynamic client mode with single endpoint, the
  * client will periodically use the configuration endpoint to retrieve the current
 cache
  * cluster configuration. This allows scaling the cache cluster up or down in number
 of nodes
  * without requiring any changes to the PHP application.
  *
  * By default the Memcached instances are destroyed at the end of the request.
  * To create an instance that persists between requests,
  *    use persistent_id to specify a unique ID for the instance.
  * All instances created with the same persistent_id will share the same connection.
  * See http://php.net/manual/en/memcached.construct.php for more information.
  */
  $dynamic_client = new Memcached('persistent-id');
  $dynamic_client->setOption(Memcached::OPT_CLIENT_MODE,
 Memcached::DYNAMIC_CLIENT_MODE);
```

```
 $dynamic_client->addServer($server_endpoint, $server_port);

 /**
 * Store the data for 60 seconds in the cluster.
 * The client will decide which cache host will store this item.
 */
 $dynamic_client->set('key', 'value', 60);


/**
 * Configuring the client with Static client mode disables the usage of Auto Discovery
 * and the client operates as it did before the introduction of Auto Discovery.
 * The user can then add a list of server endpoints.
 */
 $static_client = new Memcached('persistent-id');
 $static_client->setOption(Memcached::OPT_CLIENT_MODE, Memcached::STATIC_CLIENT_MODE);
 $static_client->addServer($server_endpoint, $server_port);

/**
 * Store the data without expiration.
 * The client will decide which cache host will store this item.
 */
 $static_client->set('key', 'value');
 ?>
```

有关如何在启用 TLS 的情况下使用 ElastiCache 集群客户端的示例，请参阅对 PHP 和 Memcached 使用传输中加密。

使用适用于 .NET 的 ElastiCache Cluster Client

> **Note**
>
> 截至 2022 年 5 月，ElastiCache .NET 集群客户端已被弃用。

适用于 ElastiCache 的 .NET 客户端是发布在 https://github.com/awslabs/elasticache-cluster-config-net 上的开源客户端。

.NET 应用程序通常会从其配置文件中获取其配置。下面是一个应用程序配置文件示例。

```
<?xml version="1.0" encoding="utf-8"?>
<configuration>
```

```
    <configSections>
        <section
            name="clusterclient"
            type="Amazon.ElastiCacheCluster.ClusterConfigSettings,
 Amazon.ElastiCacheCluster" />
    </configSections>

    <clusterclient>
        <!-- the hostname and port values are from step 1 above -->
        <endpoint hostname="mycluster.fnjyzo.cfg.use1.cache.amazonaws.com"
 port="11211" />
    </clusterclient>
</configuration>
```

下述 C# 程序演示了如何使用 ElastiCache Cluster Client 以连接到集群配置端点，并将数据项目添加至缓存中。借助 Auto Discovery，程序将在没有任何进一步干预的情况下连接至集群中的所有节点。

```
// *****************
// Sample C# code to show how to integrate with the Amazon ElastiCcache Auto Discovery
 feature.

using System;

using Amazon.ElastiCacheCluster;

using Enyim.Caching;
using Enyim.Caching.Memcached;

public class DotNetAutoDiscoveryDemo  {

    public static void Main(String[] args)  {

        // instantiate a new client.
        ElastiCacheClusterConfig config = new ElastiCacheClusterConfig();
        MemcachedClient memClient = new MemcachedClient(config);

        // Store the data for 3600 seconds (1hour) in the cluster.
        // The client will decide which cache host will store this item.
        memClient.Store(StoreMode.Set, 3600, "This is the data value.");

    }  // end Main

}  // end class DotNetAutoDiscoverDemo
```

手动连接至缓存节点

如果您的客户端程序未使用 Auto Discovery，那么它可以手动连接至每一个缓存节点。这是 Memcached 客户端的默认行为。

您可以从 AWS 管理控制台获取一份缓存节点主机名和端口号列表。您也可以使用带有 `--show-cache-node-info` 参数的 AWS CLI `aws elasticache describe-cache-clusters` 命令。

Example

下方 Java 代码片段显示了如何连接至四节点缓存集群中的所有节点：

```
...

ArrayList<String> cacheNodes = new ArrayList<String>(
 Arrays.asList(
     "mycachecluster.fnjyzo.0001.use1.cache.amazonaws.com:11211",
     "mycachecluster.fnjyzo.0002.use1.cache.amazonaws.com:11211",
     "mycachecluster.fnjyzo.0003.use1.cache.amazonaws.com:11211",
     "mycachecluster.fnjyzo.0004.use1.cache.amazonaws.com:11211"));

MemcachedClient cache = new MemcachedClient(AddrUtil.getAddresses(cacheNodes));

...
```

Important

如果您通过添加或移除节点来纵向扩展或缩减您的缓存集群，那么您将需要更新客户端代码中的节点列表。

把 Auto Discovery 添加至您的客户端库

Auto Discovery 的配置信息以冗余方式存储在各个缓存集群节点中。客户端应用程序可以查询任何缓存节点并获取集群中所有节点的配置信息。

应用程序完成此操作采用的方式取决于缓存引擎版本：

- 如果缓存引擎版本为 1.4.14 或更高版本，请使用 `config` 命令。
- 如果缓存引擎版本 低于 1.4.14，请使用 `get AmazonElastiCache:cluster` 命令。

这两个命令得到的输出结果是相同的，并且在下面的 输出格式 部分中加以描述。

缓存引擎版本 1.4.14 或更高版本

对于缓存引擎版本 1.4.14 或更高版本，请使用 `config` 命令。此命令已被 ElastiCache 添加至 Memcached ASCII 和二进制协议，并在 ElastiCache Cluster Client 中得以执行。如果您想将 Auto Discovery 与其他客户端库一同使用，那么将需要对此库进行扩展，以支持 `config` 命令。

> **Note**
>
> 下面的文档与 ASCII 协议有关；然而，`config` 命令支持 ASCII 和库。如果您想使用二进制协议添加 Auto Discovery 支持文件，请参阅 ElastiCache Cluster Client 的源代码。

语法

```
config [sub-command] [key]
```

选项

名称	描述	必填
`sub-command`	用于与缓存节点互动的子命令。对于 Auto Discovery，这个子命令为 `get`。	是
`key`	存储集群配置的密钥。对于 Auto Discovery，这个密钥的名称为 `cluster`。	是

名称	描述	必填

如要获取集群配置信息，请使用下述命令：

```
config get cluster
```

缓存引擎版本低于 1.4.14

如要获取集群配置信息，请使用下述命令：

```
get AmazonElastiCache:cluster
```

> **Note**
>
> 不要篡改“AmazonElastiCache:cluster”密钥，因为此处为集群配置信息所在的位置。如果您确实覆盖了这个密钥，那么在 ElastiCache 自动且正确地更新配置信息之前，客户端可能会在一小段时间内（不超过 15 秒）出现错误配置的情况。

输出格式

无论您使用 `config get cluster` 或 `get AmazonElastiCache:cluster`，回复都由两行组成：

- 配置信息的版本号。每当在缓存集群中添加一个节点或者从缓存集群中移除一个节点时，版本号都会增加一个数。
- 一份缓存节点列表。列表中的各个节点都由 用户名|IP 地址|端口 组加以表示，并且每个节点都由一个空格加以限定。

回车和换行字符 (CR + LF) 出现在每行末尾处。数据行末尾包含一个换行字符 (LF)，其中添加了 CR + LF。配置版本行以 LF 终止，无需 CR。

包含三个节点的缓存集群的表示方式如下：

```
configversion\n
hostname|ip-address|port hostname|ip-address|port hostname|ip-address|port\n\r\n
```

每个节点都会用 CNAME 和私有 IP 地址显示。CNAME 将会始终加以显示；如果私有 IP 地址不可用，则不会显示；然而，管道字符“|”仍会被印出。

Example

下面介绍了一个示例，即当您询问配置信息时返回的有效负载：

```
CONFIG cluster 0 136\r\n
12\n
myCluster.pc4ldq.0001.use1.cache.amazonaws.com|10.82.235.120|11211
 myCluster.pc4ldq.0002.use1.cache.amazonaws.com|10.80.249.27|11211\n\r\n
END\r\n
```

Note

- 第二行表示迄今为止已修改配置信息 12 次。
- 在第三行中，节点列表按照主机名的字母顺序进行排序。这种排序可能与您目前在客户端应用程序中采用的顺序不同。

带有 Auto Discovery 的 ElastiCache 客户端

本部分讨论如何安装和配置 ElastiCache PHP 和 .NET 客户端。

主题

- 安装和编译集群客户端
- 配置 ElastiCache 客户端

安装和编译集群客户端

本部分介绍如何安装、配置和编译 PHP 以及 .NET Amazon ElastiCache Auto Discovery 集群客户端。

主题

- 安装适用于 .NET 的 ElastiCache Cluster Client
- 安装适用于 PHP 的 ElastiCache Cluster Client
- 编译适用于 PHP 的 ElastiCache Cluster Client 的源代码

安装适用于 .NET 的 ElastiCache Cluster Client

> **Note**
>
> 截至 2022 年 5 月，ElastiCache .NET 集群客户端已被弃用。

您可以在 https://github.com/awslabs/elasticache-cluster-config-net 上找到开源的 ElastiCache .NET Cluster Client 代码。

本部分说明如何在 Amazon EC2 实例上安装、更新和移除适用于 ElastiCache Cluster Client 的 .NET 组件。有关自动发现的更多信息，请参阅自动发现。有关使用客户端的示例 .NET 代码，请参阅使用 DotNET 的自动发现。

主题

- 安装 .NET
- 下载适用于 ElastiCache 的 ElastiCache .NET 集群客户端
- 使用 NuGet 安装 AWS 程序集

安装 .NET

您必须安装了 .NET 3.5 或更高版本才能使用 AWS .NET SDK for ElastiCache。如果您未安装 .NET 3.5 或更高版本，则可从 http://www.microsoft.com/net 下载并安装最新版本。

下载适用于 ElastiCache 的 ElastiCache .NET 集群客户端

下载 ElastiCache .NET 集群客户端

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，单击 ElastiCache Cluster Client。
3. 在 Download ElastiCache Memcached Cluster Client 列表中，选择 .NET，然后单击 Download。

使用 NuGet 安装 AWS 程序集

NuGet 是 .NET 平台的包管理系统。NuGet 知道程序集依赖项并自动安装所有必需的文件。NuGet 安装的程序集将与您的解决方案存储在一起，而不是存储在 `Program Files` 这样的中央位置，因此您可安装特定于应用程序的版本，而不会产生兼容性问题。

安装 NuGet

NuGet 可从 MSDN 上的安装库进行安装；请参阅 https://visualstudiogallery.msdn.microsoft.com/27077b70-9dad-4c64-adcf-c7cf6bc9970c。如果您使用的是 Visual Studio 2010 或更高版本，则将自动安装 NuGet。

您可从 Solution Explorer (解决方案资源管理器) 或 Package Manager Console (包管理器控制台) 使用 NuGet。

从解决方案资源管理器使用 NuGet

从 Visual Studio 2010 的解决方案资源管理器中使用 NuGet

1. 从 Tools (工具) 菜单中，选择 Library Package Manager (库包管理器)。
2. 单击 Package Manager Console（软件包管理器控制台）。

从 Visual Studio 2012 或 Visual Studio 2013 的解决方案资源管理器中使用 NuGet

1. 从 Tools (工具) 菜单中，选择 NuGet Package Manager (NuGet 包管理器)。
2. 单击 Package Manager Console（软件包管理器控制台）。

从命令行中，您可使用 `Install-Package` 安装程序集，如下所示。

```
Install-Package Amazon.ElastiCacheCluster
```

若要查看可通过 NuGet 提供的每个程序包（例如 AWS 开发工具包和 AWS Extensions 程序集）对应的网页，请参阅 NuGet 网站 (http://www.nuget.org)。每个包对应的网页包括一个用于通过控制台安装包的示例命令行和一个列表（其中包含可通过 NuGet 使用的包的早期版本）。

有关 Package Manager Console (包管理器控制台) 命令的更多信息，请参阅 http://nuget.codeplex.com/wikipage?title=Package%20Manager%20Console%20Command%20Reference%20%28v1.3%29。

安装适用于 PHP 的 ElastiCache Cluster Client

本部分说明如何在 Amazon EC2 实例上安装、更新和移除适用于 ElastiCache Cluster Client 的 PHP 组件。有关 Auto Discovery 的更多信息，请参阅 自动识别集群中的节点。有关使用客户端的 PHP 示例代码，请参阅使用适用于 PHP 的 ElastiCache Cluster Client。

主题

- 下载安装包
- 针对已经安装 php-memcached 扩展的用户
- 新用户安装步骤
- 移除 PHP 集群客户端

下载安装包

为了确保适用于 PHP 的 ElastiCache Cluster Client 的版本正确，您需要了解您的 Amazon EC2 实例上安装的是哪一个版本的 PHP。此外，还需要知道您的 Amazon EC2 实例是在 64 位还是 32 位版本的 Linux 上运行的。

确定安装在 Amazon EC2 实例上的 PHP 版本

- 在命令提示符下，运行以下命令：

```
php -v
```

PHP 版本将在输出中显示，如本示例所示：

```
PHP 5.4.10 (cli) (built: Jan 11 2013 14:48:57)
Copyright (c) 1997-2012 The PHP Group
Zend Engine v2.4.0, Copyright (c) 1998-2012 Zend Technologies
```

> **Note**
>
> 如果您的 PHP 版本与 Memcached 版本不兼容，您会收到与下面类似的错误消息：
>
> ```
> PHP Warning: PHP Startup: memcached: Unable to initialize module
> Module compiled with module API=20100525
> PHP compiled with module API=20131226
> These options need to match
> ```

```
in Unknown on line 0
```

如果出现这种情况，您需要从源代码编译模块。有关更多信息，请参阅编译适用于 PHP 的 ElastiCache Cluster Client 的源代码。

确定您的 Amazon EC2 AMI 架构（64 位或 32 位）

1. 登录到 AWS Management Console 并打开 Amazon EC2 控制台（https://console.aws.amazon.com/ec2/）。
2. 在 Instances（实例）列表中，单击您的 Amazon EC2 实例。
3. 在 Description 选项卡上，查找 AMI: 字段。64 位实例应该把 x86_64 作为描述的一部分；对于 32 位实例，查找本字段中的 i386 或 i686。

您现在准备下载 ElastiCache Cluster Client。

下载适用于 PHP 的 ElastiCache Cluster Client

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 在 ElastiCache 控制台中，选择 ElastiCache Cluster Client。
3. 从 Download ElastiCache Memcached Cluster Client（下载 ElastiCache Memcached Cluster Client）列表中，选择与 PHP 版本和 AMI 架构匹配的 ElastiCache Cluster Client，然后选择 Download（下载）按钮。

   对于支持 TLS 的客户端，请下载 PHP 版本 7.4 或更高版本的二进制文件。

针对已经安装 php-memcached 扩展的用户

更新 `php-memcached` 安装

1. 移除先前安装的适用于 PHP 的 Memcached 扩展，如主题移除 PHP 集群客户端所述。
2. 如前面的 新用户安装步骤 中所述，安装新的 ElastiCache `php-memcached` 扩展。

新用户安装步骤

主题

- 为新用户安装 PHP 7.x – 8.x
- 为新用户安装 PHP 5.x

为新用户安装 PHP 7.x – 8.x

主题

- 在 Amazon Linux 2 AMI 上安装 PHP 7.x – 8.x
- 在 Amazon Linux 201609 AMI 上安装 PHP 7.x – 8.x
- 在 SUSE Linux 15 AMI 上安装 PHP 7.x – 8.x
- 在 Ubuntu 22.04 AMI 上安装 PHP 7.x – 8.x

在 Amazon Linux 2 AMI 上安装 PHP 7.x – 8.x

> **Note**
>
> 如有必要，请将 *PHP-7.x* 替换为您正在使用的版本。

1. 从 AMI 启动新实例。
2. 运行以下命令：

   ```
   sudo yum install gcc-c++ zlib-devel
   ```

3. 使用 `amazon-linux-extras` 安装 PHP 7.x

   利用 Amazon Linux 2，您可以使用 Extras 库以在您的实例上安装应用程序和软件更新。这些软件更新称为主题。您可以安装主题的某特定版本或忽略要使用最新版本的版本信息。有关更多信息，请参阅 Extras 库 (Amazon Linux 2)。

   为此，请按照下方步骤操作；

   a. 首先，验证 amazon-linux-extras 是否已安装。

   b. 如果尚未安装，请运行以下命令进行安装：

      ```
      sudo yum install -y amazon-linux-extras
      ```

   c. 确认 PHP 7.x 主题在 Amazon Linux 2 机器中可用：

```
sudo amazon-linux-extras | grep php
```

d. 从输出中，查看所有 PHP 7 主题并选择您想要的版本：

```
sudo amazon-linux-extras enable php7.x
```

e. 从存储库中安装 PHP 程序包。例如：

```
sudo yum clean metadata

sudo yum install php php-devel
```

4. 下载 Amazon ElastiCache Cluster Client。

- 打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。

  在 ElastiCache 控制面板下，转到 ElastiCache Cluster Client，然后选择您希望使用的 PHP7 版本。
- 在命令行中，将 PHP-7.X 替换为所需的 PHP 版本，并将 ARCH 替换为所需的架构（X86 或 arm），而对于版本 >= 7.4 的 PHP，将 OpenSSL 替换为所需的 OpenSSL 版本（openssl1.1 或 openssl3）。如果您使用的 PHP 版本 > 7.4，请删除 OpenSSL 后缀。

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.X/
latest-64bit-<ARCH>-<OpenSSL>
```

5. 使用 tar -zxvf 提取所下载的文件。

```
tar -zxvf latest-64bit-<ARCH>-<OpenSSL>
```

6. 在具有 Root 权限的情况下，将提取的工件文件 amazon-elasticache-cluster-client.so 复制到 /usr/lib64/php/modules。

```
sudo mv amazon-elasticache-cluster-client.so /usr/lib64/php/modules/
```

7. 将 extension=amazon-elasticache-cluster-client.so 添加到文件 /etc/php.ini

8. 如果您下载了 PHP 7.4 或更高版本的 ElastiCache Cluster Client，请安装 OpenSSL 1.1.x 或更高版本。适用于 OpenSSL 1.1.1 的安装说明：

```
sudo yum -y update
sudo yum install -y make gcc perl-core pcre-devel wget zlib-devel
wget https://www.openssl.org/source/openssl-1.1.1c.tar.gz
tar xvf openssl-1.1.1c.tar.gz
```

```
cd openssl-1.1.1c
./config
make
sudo make install
sudo ln -s /usr/local/lib64/libssl.so.1.1 /usr/lib64/libssl.so.1.1
```

在 Amazon Linux 201609 AMI 上安装 PHP 7.x – 8.x

> **Note**
>
> 如有必要，请将 *php7.x* 替换为您正在使用的版本。

1. 从 AMI 启动新实例。有关更多信息，请参阅 Amazon EC2 用户指南中的步骤 1：启动实例。
2. 运行以下命令：

```
sudo yum install gcc-c++
```

3. 安装 PHP

```
sudo yum install php7.x
```

4. 下载 Amazon ElastiCache Cluster Client。

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.x/
latest-64bit
```

5. 提取 `latest-64bit`。

```
tar -zxvf latest-64bit
```

6. 在具有 Root 权限的情况下，将提取的构件文件 `amazon-elasticache-cluster-client.so` 复制到 `/usr/lib64/php/7.x/modules/`。

```
sudo mv artifact/amazon-elasticache-cluster-client.so /usr/lib64/php/7.x/modules/
```

7. 创建 `50-memcached.ini` 文件。

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /
etc/php-7.x.d/50-memcached.ini
```

8. 启动或重启 Apache 服务器。

```
sudo /etc/init.d/httpd start
```

在 SUSE Linux 15 AMI 上安装 PHP 7.x – 8.x

> **Note**
>
> 如有必要，请将 *php7.x* 替换为您正在使用的版本。

1. 从 AMI 启动新实例。
2. 运行以下命令：

```
sudo zypper refresh
sudo zypper update -y
sudo zypper install gcc
```

3. 安装 PHP

```
sudo yum install php7.x
```

or

```
sudo zypper addrepo //download.opensuse.org/repositories/devel:/languages:/php/
openSUSE_Leap_15.3/ php
```

4. 下载 Amazon ElastiCache Cluster Client，将 <ARCH> 替换为所需的架构（X86 或 arm）。SUSE 15 内置了 OpenSSL1.1，因此对于版本 >= 7.4 的 PHP，请选择带有 OpenSSL1 的客户端二进制文件。如果您使用的 PHP 版本 < 7.4，请删除 OpenSSL 后缀。

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.x/
latest-64bit-<ARCH>-openssl1.1
```

5. 提取 latest-64bit。

```
tar -zxvf latest-64bit-<ARCH>-openssl1.1
```

6. 在具有 Root 权限的情况下，将提取的构件文件 amazon-elasticache-cluster-client.so 复制到 /usr/lib64/php7/extensions/。

```
sudo mv artifact/amazon-elasticache-cluster-client.so /usr/lib64/php7/extensions/
```

7. 将 extension=amazon-elasticache-cluster-client.so 行插入到文件 /etc/php7/cli/php.ini。

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/
php7/cli/php.ini
```

8. 启动或重启 Apache 服务器。

```
sudo /etc/init.d/httpd start
```

在 Ubuntu 22.04 AMI 上安装 PHP 7.x – 8.x

> **Note**
>
> 如有必要，请将 *php7.x* 替换为您正在使用的版本。

1. 从 AMI 启动新实例。
2. 运行以下命令：

```
sudo apt-get update
sudo apt-get install gcc g++ make zlib1g zlib1g-dev
```

3. 安装 PHP

   a. 适用于 PHP 8.1 的安装说明：

   ```
   sudo apt install php8.1-cli php8.1-dev
   ```

b. 适用于 PHP 7.4 的安装说明：

```
sudo apt -y install software-properties-common
sudo add-apt-repository ppa:ondrej/php
sudo apt-get update
sudo apt -y install php7.4
```

4. 下载 Amazon ElastiCache Cluster Client，将 <ARCH> 替换为所需的架构（X86 或 arm）。Ubuntu 22.04 内置了 OpenSSL3，因此对于版本 >= 7.4 的 PHP，请选择带有 OpenSSL3 的客户端二进制文件。如果您使用的 PHP 版本 < 7.4，请删除 OpenSSL 后缀。

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.x/
latest-64bit-<ARCH>-openssl3
```

5. 提取 latest-64bit。

```
tar -zxvf latest-64bit-<ARCH>-openssl3
```

6. 在具有 Root 权限的情况下，将提取的构件文件 amazon-elasticache-cluster-client.so 复制到 php 扩展目录 /usr/lib/php/20190902。如果该扩展目录不存在，您可以运行以下命令找到它：php -i | grep extension_dir
7. 将 extension=amazon-elasticache-cluster-client.so 行插入到文件 /etc/php/7.x/cli/php.ini。

为新用户安装 PHP 5.x

主题

- 在 Amazon Linux AMI 2014.03（64 位和 32 位）上安装 PHP 5
- 在 Red Hat Enterprise Linux 7.0 AMI（64 位和 32 位）上安装 PHP 5
- 在 Ubuntu Server 14.04 LTS AMI（64 位和 32 位）上安装 PHP 5
- 为 SUSE Linux Enterprise Server 11 AMI（64 位或 32 位）安装 PHP 5
- 其他 Linux 分配

在 Amazon Linux AMI 2014.03（64 位和 32 位）上安装 PHP 5

1. 启动一个 Amazon Linux 实例（64 位或 32 位），然后登录。

2. 安装 PHP 依赖项：

```
$ sudo yum install gcc-c++ php php-pear
```

3. 下载适用于您的 Amazon EC2 实例和 PHP 版本的正确 php-memcached 软件包。有关更多信息，请参阅下载安装包。

4. 安装 php-memcached。URI 应为安装包的下载路径：

```
$ sudo pecl install <package download path>
```

这是一个关于 PHP 5.4、64 位 Linux 的示例安装命令。在这个示例中，使用实际版本号代替 *X.Y.Z*：

```
$ sudo pecl install /home/AmazonElastiCacheClusterClient-X.Y.Z-PHP54-64bit.tgz
```

> **Note**
>
> 请务必使用最新版本的安装工件。

5. 在具有 Root/Sudo 权限的情况下，在 memcached.ini 目录中添加一个名为 /etc/php.d 的新文件，然后在该文件中插入“extension=amazon-elasticache-cluster-client.so”：

```
$ echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/
php.d/memcached.ini
```

6. 启动或重启 Apache 服务器。

```
sudo /etc/init.d/httpd start
```

在 Red Hat Enterprise Linux 7.0 AMI（64 位和 32 位）上安装 PHP 5

1. 启动一个 Red Hat Enterprise Linux 实例（64 位或 32 位），然后登录。

2. 安装 PHP 依赖项：

```
sudo yum install gcc-c++ php php-pear
```

3. 下载适用于您的 Amazon EC2 实例和 PHP 版本的正确 php-memcached 软件包。有关更多信息，请参阅下载安装包。
4. 安装 php-memcached。URI 应为安装包的下载路径：

```
sudo pecl install <package download path>
```

5. 在具有 Root/Sudo 权限的情况下，在 memcached.ini 目录中添加一个名为 /etc/php.d 的新文件，然后在该文件中插入 extension=amazon-elasticache-cluster-client.so。

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/
php.d/memcached.ini
```

6. 启动或重启 Apache 服务器。

```
sudo /etc/init.d/httpd start
```

在 Ubuntu Server 14.04 LTS AMI（64 位和 32 位）上安装 PHP 5

1. 启动一个 Ubuntu Linux 实例（64 位或 32 位），然后登录。
2. 安装 PHP 依赖项：

```
sudo apt-get update
sudo apt-get install gcc g++ php5 php-pear
```

3. 下载适用于您的 Amazon EC2 实例和 PHP 版本的正确 php-memcached 软件包。有关更多信息，请参阅下载安装包。
4. 安装 php-memcached。URI 应为安装包的下载路径。

```
$ sudo pecl install <package download path>
```

> **Note**
>
> 此安装步骤将构建工件 amazon-elasticache-cluster-client.so 安装到 /usr/lib/php5/20121212* 目录中。请核对构建工件的绝对路径，因为您在下一个步骤中需要使用此路径。

如果上一个命令不起作用，则需要从下载的 amazon-elasticache-cluster-client.so 文件中手动提取 PHP 客户端工件 *.tgz，将它复制到 /usr/lib/php5/20121212* 目录。

```
$ tar -xvf <package download path>
cp amazon-elasticache-cluster-client.so /usr/lib/php5/20121212/
```

5. 在具有 Root/Sudo 权限的情况下，在 /etc/php5/cli/conf.d 目录中添加一个名为 memcached.ini 的新文件，然后在该文件中插入“extension=<amazon-elasticache-cluster-client.so 的绝对路径>”。

```
$ echo "extension=<absolute path to amazon-elasticache-cluster-client.so>" | sudo
 tee --append /etc/php5/cli/conf.d/memcached.ini
```

6. 启动或重启 Apache 服务器。

```
sudo /etc/init.d/httpd start
```

为 SUSE Linux Enterprise Server 11 AMI（64 位或 32 位）安装 PHP 5

1. 启动一个 SUSE Linux 实例（64 位或 32 位），然后登录。
2. 安装 PHP 依赖项：

```
$ sudo zypper install gcc php53-devel
```

3. 下载适用于您的 Amazon EC2 实例和 PHP 版本的正确 php-memcached 软件包。有关更多信息，请参阅下载安装包。
4. 安装 php-memcached。URI 应为安装包的下载路径。

```
$ sudo pecl install <package download path>
```

5. 在具有 Root/Sudo 权限的情况下，在 memcached.ini 目录中添加一个名为 /etc/php5/conf.d 的新文件，然后在该文件中插入 **extension=amazon-elasticache-cluster-client.so**。

```
$ echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/
php5/conf.d/memcached.ini
```

6. 启动或重启 Apache 服务器。

```
sudo /etc/init.d/httpd start
```

> **Note**
>
> 如果步骤 5 不适用于任何以前的平台，请验证 amazon-elasticache-cluster-client.so 的安装路径。另外，在扩展中指定二进制文件的完整路径。此外，请确认所用的 PHP 是受支持的版本。我们支持版本 5.3 到 5.5。

其他 Linux 分配

在某些系统（特别是 CentOS7 和 Red Hat Enterprise Linux (RHEL) 7.1）上，libsasl2.so.3 已替代 libsasl2.so.2。在这些系统上，当您加载 ElastiCache 集群客户端时，它会尝试查找和加载 libsasl2.so.2，但此尝试将失败。要解决此问题，请创建一个指向 libsasl2.so.3 的符号链接，以便在客户端尝试加载 libsasl2.so.2 时将重定向到 libsasl2.so.3。以下代码将创建此符号链接。

```
cd /usr/lib64
$ sudo ln libsasl2.so.3 libsasl2.so.2
```

移除 PHP 集群客户端

主题

- 移除早期版本的 PHP 7 或更高版本
- 移除早期版本的 PHP 5

移除早期版本的 PHP 7 或更高版本

移除早期版本的 PHP 7 或更高版本

1. 依照前述安装说明，从相应的 PHP 库目录中删除 amazon-elasticache-cluster-client.so 文件。请参阅针对已经安装 php-memcached 扩展的用户处关于您的安装的部分。
2. 从 php.ini 文件中移除 extension=amazon-elasticache-cluster-client.so 行。
3. 启动或重启 Apache 服务器。

```
sudo /etc/init.d/httpd start
```

移除早期版本的 PHP 5

移除早期版本的 PHP 5

1. 移除 php-memcached 扩展：

```
sudo pecl uninstall __uri/AmazonElastiCacheClusterClient
```

2. 依照前述安装步骤，移除添加在相应目录中的 memcached.ini 文件。

编译适用于 PHP 的 ElastiCache Cluster Client 的源代码

本部分介绍如何获取和编译适用于 PHP 的 ElastiCache Cluster Client 的源代码。

有两个需要从 GitHub 提取并编译的数据包：aws-elasticache-cluster-client-libmemcached 和 aws-elasticache-cluster-client-memcached-for-php。

主题

- 编译 libmemcached 库
- 编译适用于 PHP 的 ElastiCache Memcached Auto Discovery 客户端

编译 libmemcached 库

必备的库

- OpenSSL 1.1.0 或更高版本（除非 TLS 支持已被 ./configure --disable-tls 禁用）。
- SASL（libsasl2，除非 SASL 支持已被 `./configure --disable-sasl` 禁用）。

编译 aws-elasticache-cluster-client-libmemcached 库

1. 启动 Amazon EC2 实例。
2. 安装库依赖项。
   - 在 Amazon Linux 201509 AMI / Amazon Linux 2 AMI 上

```
sudo yum -y update
sudo yum install gcc gcc-c++ autoconf libevent-devel make perl-core pcre-devel
 wget zlib-devel
// Install OpenSSL 1.1.1
wget https://www.openssl.org/source/openssl-1.1.1c.tar.gz
tar xvf openssl-1.1.1c.tar.gz
cd openssl-1.1.1c
./config
make
sudo make install
sudo ln -s /usr/local/lib64/libssl.so.1.1 /usr/lib64/libssl.so.1.1
```

   - 在 Ubuntu 14.04 AMI 上（OpenSSL >= 1.1 附带的 Ubuntu 版本不需要）

```
sudo apt-get update
```

```
sudo apt-get install libevent-dev gcc g++ make autoconf libsasl2-dev

// Install OpenSSL 1.1.1
wget https://www.openssl.org/source/openssl-1.1.1c.tar.gz
tar xvf openssl-1.1.1c.tar.gz
cd openssl-1.1.1c
./config
make
sudo make install
sudo ln -s /usr/local/lib/libssl.so.1.1 /usr/lib/x86_64-linux-gnu/libssl.so.1.1
```

3. 拉取存储库并编译代码。

```
git clone https://github.com/awslabs/aws-elasticache-cluster-client-
libmemcached.git
cd aws-elasticache-cluster-client-libmemcached
touch configure.ac aclocal.m4 configure Makefile.am Makefile.in
mkdir BUILD
cd BUILD
../configure --prefix=<libmemcached-install-directory> --with-pic --disable-sasl
```

如果运行 `../configure` 无法找到 `libssl`（OpenSSL 库），则可能需要调整 `PKG_CONFIG_PATH` 环境变量：

```
PKG_CONFIG_PATH=/path/to/ssl/lib/pkgconfig ../configure --prefix=<libmemcached-
install-directory> --with-pic --disable-sasl
```

或者，如果您没有使用 TLS，则可以通过运行以下操作来禁用它：

```
make
sudo make install
../configure –prefix=<libmemcached-install-directory> --with-pic --disable-sasl --
disable-tls
```

编译适用于 PHP 的 ElastiCache Memcached Auto Discovery 客户端

以下部分介绍如何编译 ElastiCache Memcached Auto Discovery 客户端

主题

- 编译适用于 PHP 7 或更高版本的 ElastiCache Memcached 客户端
- 编译适用于 PHP 5 的 ElastiCache Memcached 客户端

编译适用于 PHP 7 或更高版本的 ElastiCache Memcached 客户端

将 PHP-7.x 替换为您正在使用的版本。

安装 PHP：

```
sudo yum install -y amazon-linux-extras
sudo amazon-linux-extras enable php7.x
```

在代码目录下运行下面一组命令。

```
git clone  https://github.com/awslabs/aws-elasticache-cluster-client-memcached-for-
php.git
cd aws-elasticache-cluster-client-memcached-for-php
phpize
mkdir BUILD
CD BUILD
../configure --with-libmemcached-dir=<libmemcached-install-directory> --disable-
memcached-sasl
```

如果运行 ../configure 无法找到 libssl（OpenSSL 库），则可能需要将 PKG_CONFIG_PATH 环境变量调整为 OpenSSL 的 .PC 文件目录：

```
PKG_CONFIG_PATH=/path/to/ssl/lib/pkgconfig ../configure --with-libmemcached-dir=<path
 to libmemcached build directory> --disable-memcached-sasl
```

或者，如果您没有使用 TLS，则可以通过运行以下操作来禁用它：

```
make
make install
../configure --with-libmemcached-dir=<path to libmemcached build directory> --disable-
memcached-sasl --disable-memcached-tls
```

> **Note**
>
> 您可以将 libmemcached 库静态链接到 PHP 二进制文件，以使其可以跨各种 Linux 平台传输。为此，请在 make 之前运行以下命令：

```
sed -i "s#-lmemcached#<libmemcached-install-directory>/lib/libmemcached.a -
lcrypt -lpthread -lm -lstdc++ -lsasl2#" Makefile
```

编译适用于 PHP 5 的 ElastiCache Memcached 客户端

通过在 aws-elasticache-cluster-client-memcached-for-php/ 文件夹下运行以下命令，编译 aws-elasticache-cluster-client-memcached-for-php。

```
git clone  https:////github.com/awslabs/aws-elasticache-cluster-client-memcached-for-
php/tree/php.git
cd aws-elasticache-cluster-client-memcached-for-php
sudo yum install zlib-devel
phpize
./configure --with-libmemcached-dir=<libmemcached-install-directory>
make
make install
```

配置 ElastiCache 客户端

ElastiCache 集群符合 Memcached 协议标准。您目前与您的现有 Memcached 环境一起使用的代码、应用程序和最普及的工具都将与服务无缝结合。

本部分讨论有关在 ElastiCache 中连接到缓存节点的特定注意事项。

主题

查找节点端点和端口号

要连接到某个缓存节点，您的应用程序需要知道该节点的终端节点和端口号。

查找节点端点和端口号（控制台）

确定节点终端节点和端口号

1. 登录 Amazon ElastiCache 管理控制台，然后选择在集群上运行的引擎。

   此时会显示运行所选引擎的所有集群的列表。
2. 对您运行的引擎和配置继续下面的操作。
3. 选择所需集群的名称。
4. 找到所需节点的 Port 和 Endpoint 列。

查找缓存节点端点和端口号 (AWS CLI)

要确定缓存节点终端节点和端口号，请使用带 `--show-cache-node-info` 参数的命令 `describe-cache-clusters`。

```
aws elasticache describe-cache-clusters --show-cache-node-info
```

完全限定的 DNS 名称和端口号处于输出的终端节点部分。

查找缓存节点端点和端口号 (ElastiCache API)

要确定缓存节点终端节点和端口号，请使用带 `ShowCacheNodeInfo=true` 参数的操作 `DescribeCacheClusters`。

Example

```
https://elasticache.us-west-2.amazonaws.com /
    ?Action=DescribeCacheClusters
    &ShowCacheNodeInfo=true
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20140421T220302Z
    &Version=2014-09-30
    &X-Amz-Algorithm=&AWS;4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20140421T220302Z
    &X-Amz-Expires=20140421T220302Z
    &X-Amz-Signature=<signature>
    &X-Amz-SignedHeaders=Host
```

连接以使用 Auto Discovery

如果您的应用程序使用 Auto Discovery，则您只需知道集群的配置终端节点，而不是每个缓存节点的各个终端节点。有关更多信息，请参阅自动识别集群中的节点。

> **Note**
>
> 此时，Auto Discovery 只能用于运行 Memcached 的缓存集群。

DNS 名称和底层 IP

客户端维护一份服务器列表，其中包含保存缓存数据的服务器的地址和端口。在使用 ElastiCache 时，DescribeCacheClusters API（或 describe-cache-clusters 命令行实用工具）会返回可用于服务器列表的完全限定 DNS 条目和端口号。

> **Important**
>
> 重要的是配置客户端应用程序，以便在它们尝试连接到缓存节点终端节点时频繁地解析缓存节点的 DNS 名称。

VPC 安装

当缓存节点发生故障恢复时，ElastiCache 可确保缓存节点的 DNS 名称和 IP 地址均保持不变。

非 VPC 安装

当缓存节点发生故障恢复时，ElastiCache 可确保缓存节点的 DNS 名称保持不变；然而，缓存节点的基本 IP 地址可能会出现更改。

大多数客户端库默认支持持久性缓存节点连接。我们建议您在使用 ElastiCache 时采用持久性缓存节点连接。客户端 DNS 缓存可以出现在多个位置，包括客户端库、语言运行时或客户端操作系统。您应该检查每一层的应用程序配置，以确保您可以频繁地解析您的缓存节点 IP 地址。

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：

您想使用 ElastiCache 无服务器服务还是基于实例的服务？

如果您想使用无服务器缓存，只需确保已正确配置您的 VPC、子网和安全组即可。有关更多详细信息，请参阅用于访问 Amazon VPC 中的 ElastiCache 缓存的访问模式。如果您想使用基于实例 ElastiCache，请继续阅读。

- 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 Memcached 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，请参阅用于访问 Amazon VPC 中的 ElastiCache 缓存的访问模式。

- 您是否需要自定义任何参数值？

  如果这样做，请创建自定义参数组。有关更多信息，请参阅创建参数组。

- 您是否需要创建自己的 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

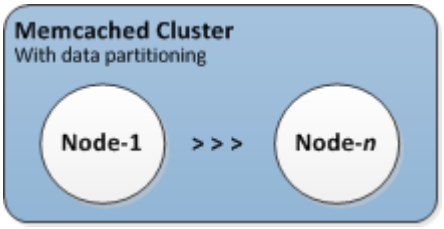
Amazon 的基本构建块 ElastiCache 是节点。配置单个节点，或成组配置节点以形成集群。在确定用于集群的节点类型时，请考虑集群的节点配置以及必须存储的数据量。

Memcached 引擎是多线程的，因此节点的内核数会影响可供集群使用的计算能力。

Memcached 集群配置

ElastiCache 对于 Memcached 集群由 1 到 60 个节点组成。Memcached 集群中的数据在集群中的节点间分区。您的应用程序使用称为终端节点的网络地址与 Memcached 集群连接。Memcached 集群中

的每个节点都具有自己的终端节点，应用程序可以使用它来对特定节点进行读取或写入。除了节点终端节点外，Memcached 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来读取或写入集群，从而由 自动识别集群中的节点 决定要读取或写入的节点。



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。当您扩展 Memcached 集群时，新集群开始为空。

ElastiCache 适用于 Memcached 的 Amazon 集群可以向外扩展或向内扩展。要扩展或收缩 Memcached 集群，您只需在集群中添加或删除节点即可。如果已启用 Automatic Discovery 并且您的应用程序已连接到集群的配置终端节点，则在添加或删除节点时不需要在应用程序中进行任何更改。

有关更多信息，请参阅本指南中的扩展 Mem ElastiCache cached。

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 访问您的集群。

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：

- 区域和可用区

- Local Zones
- 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 Memcached 节点大小

Memcached 集群包含一个或多个节点，该集群的数据会分区到各个节点中。因此，集群的内存需求和节点的内存相关但不相同。您可以通过拥有几个大型节点或多个小型节点来获得所需的集群内存容量。此外，由于您的需求是变化的，您可以在集群中添加节点或删除节点，从而仅为所需内容付费。

集群的总内存容量的计算方法是，在扣除系统开销后将集群中的节点数乘以每个节点的 RAM 容量。每个节点的容量都基于节点类型。

```
cluster_capacity = number_of_nodes * (node_capacity - system_overhead)
```

集群中的节点数是运行 Memcached 的集群可用性的一个关键因素。如果单一节点出现故障，则可能对应用程序的可用性以及后端数据库的负载产生影响。在这种情况下，ElastiCache 会更换出现故障的节点并对其进行重新填充。要减小这种可用性影响，请将内存和计算容量分布于更多节点上（每个节点的容量稍小），而非使用少量大容量节点。

在您希望拥有 35GB 缓存内存的情况下，可以设置以下任意配置：

- 11 `cache.t2.medium` 个节点，每个节点具有 3.22 GB 内存和 2 个线程，共 35.42 GB 和 22 个线程。
- 6 `cache.m4.large` 个节点，每个节点具有 6.42 GB 内存和 2 个线程，共 38.52 GB 和 12 个线程。
- 3 `cache.r4.large` 个节点，每个节点具有 12.3 GB 内存和 2 个线程，共 36.90 GB 和 6 个线程。
- 3 `cache.m4.xlarge` 个节点，每个节点具有 14.28 GB 内存和 4 个线程，共 42.84 GB 和 12 个线程。

比较节点选项

节点类型	内存（单位：GiB）	内核	小时成本*	需要的节点	总内存（单位：GiB）	核心总数	月度成本
cache.t2.medium	3.22	2	$ 0.068	11	35.42	22	$ 538.56
cache.m4.large	6.42	2	$ 0.156	6	38.52	12	$ 673.92
cache.m4.xlarge	14.28	4	$ 0.311	3	42.84	12	$ 671.76
cache.m5.xlarge	12.93	4	$ 0.311	3	38.81	12	$ 671.76
cache.m6g.large	6.85	2	$ 0.147	6	41.1	12	$ 635
cache.r4.large	12.3	2	$ 0.228	3	36.9	6	$ 492.48
cache.r5.large	13.07	2	$ 0.216	3	39.22	6	$ 466.56
cache.r6g.large	13.07	2	$ 0.205	3	42.12	6	$ 442

* 自 2020 年 10 月 8 日起的每节点小时成本。

30 天（720 小时）的 100% 使用量的月度成本。

这些选项都提供了类似的内存容量，但提供了不同的计算容量和成本。要比较特定选项的成本，请参阅 Amazon ElastiCache 定价。

对于运行 Memcached 的集群，每个节点上的部分可用内存都会用于连接开销。有关更多信息，请参阅 Memcached 连接开销

使用多个节点需要跨这些节点分布密钥。每个节点都有自己的终端节点。为了便于管理端点，您可以使用 ElastiCache（Auto Discovery 功能）以使客户端程序能够自动标识集群中的所有节点。有关更多信息，请参阅自动识别集群中的节点。

在某些情况下，您可能无法确定您需要多少容量。如果是这样，对于测试，我们建议从 `cache.m5.large` 节点开始。然后，使用发布到 Amazon CloudWatch 的 ElastiCache 指标监控内存使用率、CPU 利用率和缓存命中率。有关 ElastiCache 的 CloudWatch 指标的更多信息，请参阅 使用 CloudWatch 指标监控使用情况。对于生产和大型工作负载，R5 节点提供了最佳性能和 RAM 成本价值。

如果您的集群没有所需的命中率，您可以轻松地添加更多的节点，从而增加您的集群的可用内存总量。

如果集群受到 CPU 的约束，但具有足够的命中率，请使用提供更强计算能力的节点类型来设置新集群。

创建集群

以下示例说明如何使用 AWS Management Console、 AWS CLI 和 ElastiCache API 创建集群。

创建 Memcached 集群（控制台）

当您使用 Memcached 引擎时，Amazon ElastiCache 支持在多个节点上水平分区您的数据。Memcached 支持 Auto Discovery，因此您无需跟踪每个节点的终端节点。Memcached 跟踪每个节点的终端节点，并在添加和删除节点时更新终端节点列表。您的应用程序需要与集群进行的所有交互都在配置终端节点上进行。有关 Auto Discovery 的更多信息，请参阅自动识别集群中的节点。

要创建 Memcached 集群，请按照 第 1 步：创建缓存 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 访问您的集群 和 手动连接至缓存节点。

> **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 Memcached 缓存群集 (AWS CLI)

下面的 CLI 代码创建一个具有 3 个节点的 Memcached 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 memcached \
--engine-version 1.4.24 \
--cache-parameter-group default.memcached1.4 \
--num-cache-nodes 3
```

对于 Windows：

```
aws elasticache create-cache-cluster ^
--cache-cluster-id my-cluster ^
--cache-node-type cache.r4.large ^
--engine memcached ^
--engine-version 1.4.24 ^
--cache-parameter-group default.memcached1.4 ^
--num-cache-nodes 3
```

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。

创建 Memcached 缓存集群 (ElastiCache API)

以下代码创建了一个包含 3 个节点 (ElastiCache API) 的 Memcached 集群。

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
    &Engine=memcached
    &NumCacheNodes=3
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &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>
```

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。

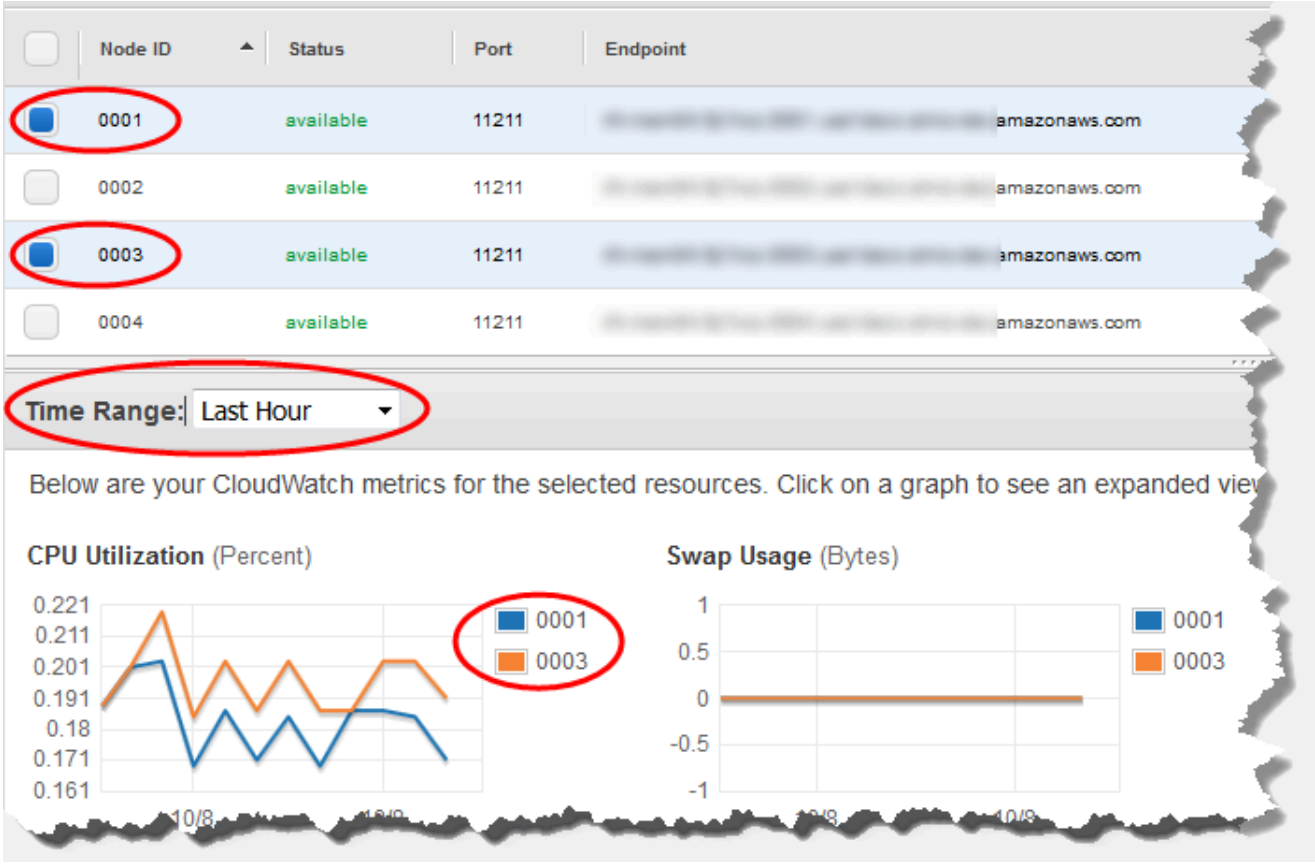
查看集群的详细信息（控制台）

您可以使用 ElastiCache 控制台、AWS CLI for ElastiCache，或 ElastiCache API 查看 Memcached 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 Memcached 集群的详细信息。

查看 Memcached 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 Memcached。这将显示运行任意 Memcached 版本的集群列表。
4. 要查看集群的详细信息，请选择集群名称左侧的复选框。
5. 查看节点信息：
    a. 选择集群的名称。
    b. 选择 Nodes（节点）选项卡。
    c. 要查看一个或多个节点的指标，请选择节点 ID 左侧的复选框，然后在 Time range 列表中选择指标的时间范围。选择多个节点会生成叠加图形。



两个 Memcached 节点过去一小时的指标

查看集群的详细信息 (AWS CLI)

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

以下代码列出了 `my-cluster` 的详细信息。

```
aws elasticache describe-cache-clusters --cache-cluster-id my-cluster
```

以下代码列出了最多 25 个集群的详细信息。

```
aws elasticache describe-cache-clusters --max-items 25
```

Example

对于 Linux、macOS 或 Unix：

```
aws elasticache describe-cache-clusters \
    --cache-cluster-id my-cluster \
    --show-cache-node-info
```

对于 Windows：

```
aws elasticache describe-cache-clusters ^
    --cache-cluster-id my-cluster ^
    --show-cache-node-info
```

此操作将生成类似于以下内容的输出（JSON 格式）：

```
{
    "CacheClusters": [
        {
            "Engine": "memcached",
            "CacheNodes": [
                {
                    "CacheNodeId": "0001",
                    "Endpoint": {
                        "Port": 11211,
                        "Address": "my-cluster.7ef-
example.0001.usw2.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-09-21T16:28:28.973Z",
                    "CustomerAvailabilityZone": "us-west-2b"
                },
                {
                    "CacheNodeId": "0002",
                    "Endpoint": {
                        "Port": 11211,
                        "Address": "my-cluster.7ef-
example.0002.usw2.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-09-21T16:28:28.973Z",
```

```
                    "CustomerAvailabilityZone": "us-west-2b"
                },
                {
                    "CacheNodeId": "0003",
                    "Endpoint": {
                        "Port": 11211,
                        "Address": "my-cluster.7ef-
example.0003.usw2.cache.amazonaws.com"
                    },
                    "CacheNodeStatus": "available",
                    "ParameterGroupStatus": "in-sync",
                    "CacheNodeCreateTime": "2016-09-21T16:28:28.973Z",
                    "CustomerAvailabilityZone": "us-west-2b"
                }
            ],
            "CacheParameterGroup": {
                "CacheNodeIdsToReboot": [],
                "CacheParameterGroupName": "default.memcached1.4",
                "ParameterApplyStatus": "in-sync"
            },
            "CacheClusterId": "my-cluster",
            "PreferredAvailabilityZone": "us-west-2b",
            "ConfigurationEndpoint": {
                "Port": 11211,
                "Address": "my-cluster.7ef-example.cfg.usw2.cache.amazonaws.com"
            },
            "CacheSecurityGroups": [],
            "CacheClusterCreateTime": "2016-09-21T16:28:28.973Z",
            "AutoMinorVersionUpgrade": true,
            "CacheClusterStatus": "available",
            "NumCacheNodes": 3,
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "SecurityGroups": [
                {
                    "Status": "active",
                    "SecurityGroupId": "sg-dbe93fa2"
                }
            ],
            "CacheSubnetGroupName": "default",
            "EngineVersion": "1.4.24",
            "PendingModifiedValues": {},
            "PreferredMaintenanceWindow": "sat:09:00-sat:10:00",
            "CacheNodeType": "cache.m3.medium"
```

```
        }
    ]
}
```

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参数更改，请参阅表格中的更改生效列以了解 Memcached 特定的参数。有关重启集群的信息，请参阅重新引导集群。

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：

   - 引擎版本兼容性
   - VPC 安全组
   - 参数组
   - 维护时段
   - 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)。

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。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。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。

重新引导集群

一些更改需要重启集群才能应用。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。

当您重启集群时，集群将刷新其所有数据并重新启动其引擎。在此过程中，您无法访问集群。由于集群已刷新其所有数据，因此当集群再次可用时，将从空集群开始。

您可以使用 ElastiCache 控制台、AWS CLI 或 ElastiCache API 重启集群。无论您是使用 ElastiCache 控制台、AWS CLI 还是 ElastiCache API，都只能发起对单个集群的重启。要重启多个集群，您必须对过程或操作进行迭代。

使用 AWS Management Console

您可以使用 ElastiCache 控制台重启集群。

重启一个集群（控制台）

1. 登录 AWS Management Console 并打开 ElastiCache 控制台 (https://console.aws.amazon.com/elasticache/)。
2. 从右上角的列表中，选择您感兴趣的 AWS 区域。
3. 在导航窗格中，选择在您要重启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. 通过选择集群名称左侧的按钮来选择要重启的集群。

   选择操作，然后选择重启。

   如果您选择多个集群，则重启按钮不可用。

要重启多个集群，请对要重启的每个集群重复步骤 2 到步骤 5。在重启一个集群之前，您无需等待另一个集群完成重启。

要重启特定节点，请选择相应节点，然后选择 Reboot（重启）。

使用 AWS CLI

要重启集群 (AWS CLI)，请使用 `reboot-cache-cluster` CLI 操作。

要重启集群中的特定节点，请使用 `--cache-node-ids-to-reboot` 列出要重启的特定集群。以下命令重启 my-cluster 的节点 0001、0002 和 0004。

对于 Linux、macOS 或 Unix：

```
aws elasticache reboot-cache-cluster \
    --cache-cluster-id my-cluster \
    --cache-node-ids-to-reboot 0001 0002 0004
```

对于 Windows：

```
aws elasticache reboot-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --cache-node-ids-to-reboot 0001 0002 0004
```

要重启集群中的所有节点，请使用 `--cache-node-ids-to-reboot` 参数并列出该集群的所有节点 ID。有关更多信息，请参阅 reboot-cache-cluster。

使用 ElastiCache API

要使用 ElastiCache API 重启集群，请使用 `RebootCacheCluster` 操作。

要重启集群中的特定节点，请使用 `CacheNodeIdsToReboot` 列出要重启的特定集群。以下命令重启 my-cluster 的节点 0001、0002 和 0004。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=RebootCacheCluster
   &CacheClusterId=my-cluster
   &CacheNodeIdsToReboot.member.1=0001
   &CacheNodeIdsToReboot.member.2=0002
   &CacheNodeIdsToReboot.member.3=0004
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

要重启集群中的所有节点，请使用 `CacheNodeIdsToReboot` 参数并列出该集群的所有节点 ID。有关更多信息，请参阅 RebootCacheCluster。

向集群添加节点

向 Memcached 集群添加节点会增加您集群的分区数量。更改集群中的分区数量时，需要重新映射一些键空间，以将其映射到正确的节点。重新映射键空间会暂时增加集群上的缓存未命中次数。有关更多信息，请参阅配置 ElastiCache 客户端以实现高效负载均衡。

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

主题

- 向集群添加节点（控制台）

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。
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。

场景	待处理的操作	新建请求	结果
			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。

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

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 `modify-cache-cluster`：

- `--cache-cluster-id` 要将节点添加到的缓存集群的 ID。
- `--num-cache-nodes` `--num-cache-nodes` 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，`--num-cache-nodes` 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，则 ElastiCache 需要参数 `cache-node-ids-to-remove` 以及要从集群中移除的节点的列表。有关更多信息，请参阅使用 AWS CLI。
- `--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 modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --num-cache-nodes 5 \
    --apply-immediately
```

对于 Windows：

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --num-cache-nodes 5 ^
    --apply-immediately
```

此操作将生成类似于以下内容的输出（JSON 格式）：

```
{
    "CacheCluster": {
        "Engine": "memcached",
        "CacheParameterGroup": {
            "CacheNodeIdsToReboot": [],
            "CacheParameterGroupName": "default.memcached1.4",
            "ParameterApplyStatus": "in-sync"
        },
        "CacheClusterId": "my-cluster",
        "PreferredAvailabilityZone": "us-west-2b",
        "ConfigurationEndpoint": {
```

```
            "Port": 11211,
            "Address": "rlh-mem000.7alc7bf-example.cfg.usw2.cache.amazonaws.com"
        },
        "CacheSecurityGroups": [],
        "CacheClusterCreateTime": "2016-09-21T16:28:28.973Z",
        "AutoMinorVersionUpgrade": true,
        "CacheClusterStatus": "modifying",
        "NumCacheNodes": 2,
        "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
        "SecurityGroups": [
            {
                "Status": "active",
                "SecurityGroupId": "sg-dbe93fa2"
            }
        ],
        "CacheSubnetGroupName": "default",
        "EngineVersion": "1.4.24",
        "PendingModifiedValues": {
            "NumCacheNodes": 5
        },
        "PreferredMaintenanceWindow": "sat:09:00-sat:10:00",
        "CacheNodeType": "cache.m3.medium",

    }
}
```

有关更多信息，请参阅 AWS CLI 主题 modify-cache-cluster。

使用 ElastiCache API

向集群添加节点 (ElastiCache API)

- 按照以下参数调用 ModifyCacheCluster API 操作：
  - CacheClusterId 要将节点添加到的集群的 ID。
  - NumCacheNodes NumCachNodes 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，NumCacheNodes 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，则 ElastiCache 需要参数 CacheNodeIdsToRemove 以及要从集群中移除的节点的列表（请参阅 使用 ElastiCache API）。
  - 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=ModifyCacheCluster
    &ApplyImmediately=true
    &NumCacheNodes=5
 &CacheClusterId=my-cluster
 &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 ModifyCacheCluster。

从集群中移除节点

每次更改您的 Memcached 集群中的节点数时，您必须至少重新映射部分键空间，以便它映射到正确的节点。有关对 Memcached 集群进行负载均衡的更多详细信息，请参阅[配置 ElastiCache 客户端以实现高效负载均衡]。

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。

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。

场景	待处理的操作	新建请求	结果
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 `modify-cache-cluster` CLI 操作与要删除的节点列表一起使用，如下例所示。

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

- `--cache-cluster-id` 要从其中删除节点的缓存集群的 ID。
- `--num-cache-nodes` `--num-cache-nodes` 参数指定应用修改后此集群中应有的节点的数量。
- `--cache-node-ids-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。

以下示例从集群 my-cluster 中立即删除节点 0001。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --num-cache-nodes 2 \
    --cache-node-ids-to-remove 0001 \
    --region us-east-2 \
    --apply-immediately
```

对于 Windows：

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --num-cache-nodes 2 ^
    --cache-node-ids-to-remove 0001 ^
    --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）：

```
{
    "CacheCluster": {
        "Engine": "memcached",
        "CacheParameterGroup": {
            "CacheNodeIdsToReboot": [],
            "CacheParameterGroupName": "default.memcached1.4",
```

```
            "ParameterApplyStatus": "in-sync"
        },
        "CacheClusterId": "my-cluster",
        "PreferredAvailabilityZone": "us-east-2b",
        "ConfigurationEndpoint": {
            "Port": 11211,
            "Address": "rlh-mem000.7ef-example.cfg.usw2.cache.amazonaws.com"
        },
        "CacheSecurityGroups": [],
        "CacheClusterCreateTime": "2016-09-21T16:28:28.973Z", 9dcv5r
        "AutoMinorVersionUpgrade": true,
        "CacheClusterStatus": "modifying",
        "NumCacheNodes": 3,
        "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
        "SecurityGroups": [
            {
                "Status": "active",
                "SecurityGroupId": "sg-dbe93fa2"
            }
        ],
        "CacheSubnetGroupName": "default",
        "EngineVersion": "1.4.24",
        "PendingModifiedValues": {
            "NumCacheNodes": 2,
            "CacheNodeIdsToRemove": [
                "0001"
            ]
        },
        "PreferredMaintenanceWindow": "sat:09:00-sat:10:00",
        "CacheNodeType": "cache.m3.medium",

    }
}
```

有关更多信息，请参阅 AWS CLI 主题 [describe-cache-cluster](#) 和 [modify-cache-cluster](#)。

使用 ElastiCache API

要使用 ElastiCache API 删除节点，请使用缓存集群 ID 和要删除节点的列表调用 `ModifyCacheCluster` API 操作，如下所示：

- `CacheClusterId` 要从其中删除节点的缓存集群的 ID。

- NumCacheNodes NumCacheNodes 参数指定应用修改后此集群中应有的节点的数量。
- CacheNodeIdsToRemove.member.n 要从集群中移除的节点 ID 的列表。
  - CacheNodeIdsToRemove.member.1=0004
  - CacheNodeIdsToRemove.member.1=0005
- 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=ModifyCacheCluster
    &CacheClusterId=my-cluster
    &ApplyImmediately=true
    &CacheNodeIdsToRemove.member.1=0004
    &CacheNodeIdsToRemove.member.2=0005
    &NumCacheNodes=3
    &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 ModifyCacheCluster。

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。

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. 对于操作，选择删除。
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

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。Memcached 的默认端口是 **11211** 。
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 Memcached 集群：
  - IP 地址 – 10.0.1.230
  - 默认 Memcached 端口 – 11211
  - 安全组：*10\.0\.0\.55*
- 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 (11211 - 11213) 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	11211-11213	198.51.100.27/32
SSH	TCP	22	198.51.100.27/32

- 一个允许与缓存端口 (11211) 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	11211	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 (11211) 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	11211	sg-bd56b7da（NAT 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 11211 -j DNAT --to
 10.0.1.230:11211
```

集群中每个节点的端口号必须唯一。例如，如果处理使用端口 11211 - 11213 的三个节点的 Memcached 集群，则规则将如下所示：

```
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 11211 -j DNAT --to
 10.0.1.230:11211
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 11212 -j DNAT --to
 10.0.1.231:11211
iptables -t nat -A PREROUTING -i eth0 -p tcp --dport 11213 -j DNAT --to
 10.0.1.232:11211
```

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：

   ```
   $memcached->connect( '203.0.113.73', 11211 );
   $memcached->connect( '203.0.113.73', 11212 );
   $memcached->connect( '203.0.113.73', 11213 );
   ```

   也可以使用 telnet 客户端来验证连接。例如：

   ```
   telnet 203.0.113.73 11211
   telnet 203.0.113.73 11212
   telnet 203.0.113.73 11213
   ```

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。

- 用于访问 Amazon VPC 中的 ElastiCache 缓存的访问模式
- 从运行于客户数据中心中的应用程序访问 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。

该使用哪些端点

对于使用 Memcached 的 ElastiCache 无服务器缓存，只需从控制台获取集群端点 DNS 和端口。

从 AWS CLI 中，使用 `describe-serverless-caches` 命令获取端点信息。

Linux

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

Windows

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

上述操作的输出类似于以下内容（JSON 格式）：

```
{
    "ServerlessCaches": [
        {
            "ServerlessCacheName": "serverless-memcached",
            "Description": "test",
            "CreateTime": 1697659642.136,
            "Status": "available",
            "Engine": "memcached",
            "MajorEngineVersion": "1.6",
            "FullEngineVersion": "21",
            "SecurityGroupIds": [
                "sg-083eda453e1e51310"
            ],
            "Endpoint": {
                "Address": "serverless-memcached-01.amazonaws.com",
                "Port":11211
            },
            "ARN": "<the ARN>",
```

```
            "SubnetIds": [
                "subnet-0cf759df15bd4dc65",
                "subnet-09e1307e8f1560d17"
            ],
            "SnapshotRetentionLimit": 0,
            "DailySnapshotTime": "03:00"
        }
    ]
}
```

对基于 Memcached 集群的实例：如果您使用自动发现，则可以使用集群的配置端点来配置您的 Memcached 客户端。这意味着您必须使用支持 Automatic Discovery 的客户端。

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。

以下部分将引导您发现正在运行的引擎所需的端点。

查找集群的端点（控制台）

所有 Memcached 端点都是读写端点。要连接到 Memcached 集群中的节点，您的应用程序可以使用每个节点的端点或将集群的配置端点与 Automatic Discovery 结合使用。要使用 Automatic Discovery，您必须使用支持 Automatic Discovery 的客户端。

在使用 Automatic Discovery 时，您的客户端应用程序将使用配置端点连接到 Memcached 集群。当您通过添加或移除节点来扩展集群，应用程序将自动“获知”集群中的所有节点并能够连接到其中任一节点。如果没有 Automatic Discovery，应用程序将必须执行此操作，否则，您在每次添加或移除节点时，必须手动更新应用程序中的端点。

要复制端点，请直接选择端点地址前面的复制图标。有关使用端点连接到节点的信息，请参阅连接到节点。

配置端点和节点端点看上非常相似。不同之处以粗体 形式突出显示。

```
myclustername.xxxxxx.cfg.usw2.cache.amazonaws.com:port   # configuration endpoint
 contains "cfg"
myclustername.xxxxxx.0001.usw2.cache.amazonaws.com:port  # node endpoint for node 0001
```

> **Important**
>
> 如果您选择为 Memcached 配置端点创建别名记录，以使自动发现客户端可以识别别名记录作为配置端点，则必须在别名记录中包含 `.cfg.`。

查找端点（AWS CLI）

您可以使用 AWS CLI for Amazon ElastiCache 来搜索节点和集群的端点。

主题

- 查找节点和集群的端点（AWS CLI）

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 Memcached 集群，此命令将返回配置端点。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 Memcached 集群 mycluster 的配置端点（`ConfigurationEndpoint`）和单个节点端点（`Endpoint`）。

对于 Linux、macOS 或 Unix：

```
aws elasticache describe-cache-clusters \
    --cache-cluster-id mycluster \
    --show-cache-node-info
```

对于 Windows：

```
aws elasticache describe-cache-clusters ^
    --cache-cluster-id mycluster ^
    --show-cache-node-info
```

上面的操作输出类似以下的内容（JSON 格式）。

```
{
   "CacheClusters": [
   {
       "Engine": "memcached",
       "CacheNodes": [
          {
             "CacheNodeId": "0001",
             "Endpoint": {
                "Port": 11211,
                "Address": "mycluster.amazonaws.com"
```

```
            },
               "CacheNodeStatus": "available",
               "ParameterGroupStatus": "in-sync",
               "CacheNodeCreateTime": "2016-09-22T21:30:29.967Z",
               "CustomerAvailabilityZone": "us-west-2b"
         },
         {
            "CacheNodeId": "0002",
            "Endpoint": {
               "Port": 11211,
               "Address": "mycluster.amazonaws.com"
            },
               "CacheNodeStatus": "available",
               "ParameterGroupStatus": "in-sync",
               "CacheNodeCreateTime": "2016-09-22T21:30:29.967Z",
               "CustomerAvailabilityZone": "us-west-2b"
         },
         {
               "CacheNodeId": "0003",
               "Endpoint": {
                  "Port": 11211,
                  "Address": "mycluster.amazonaws.com"
               },
                  "CacheNodeStatus": "available",
                  "ParameterGroupStatus": "in-sync",
                  "CacheNodeCreateTime": "2016-09-22T21:30:29.967Z",
                  "CustomerAvailabilityZone": "us-west-2b"
         }
      ],
      "CacheParameterGroup": {
      "CacheNodeIdsToReboot": [],
      "CacheParameterGroupName": "default.memcached1.4",
      "ParameterApplyStatus": "in-sync"
           },
           "CacheClusterId": "mycluster",
           "PreferredAvailabilityZone": "us-west-2b",
           "ConfigurationEndpoint": {
               "Port": 11211,
               "Address": "mycluster.amazonaws.com"
           },
           "CacheSecurityGroups": [],
           "CacheClusterCreateTime": "2016-09-22T21:30:29.967Z",
           "AutoMinorVersionUpgrade": true,
           "CacheClusterStatus": "available",
```

```
            "NumCacheNodes": 3,
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/
home#client-download:",
            "CacheSubnetGroupName": "default",
            "EngineVersion": "1.4.24",
            "PendingModifiedValues": {},
            "PreferredMaintenanceWindow": "mon:09:00-mon:10:00",
            "CacheNodeType": "cache.m4.large",

        }
    ]
}
```

Important

如果您选择为 Memcached 配置端点创建别名记录，以使自动发现客户端可以识别别名记录作为配置端点，则必须在别名记录中包含 `.cfg.`。例如，php.ini 文件中的 `session.save_path` 参数的 `mycluster.cfg.local`。

有关更多信息，请参阅主题 describe-cache-clusters。

查找端点（ElastiCache API）

您可以使用 Amazon ElastiCache API 来搜索节点和集群的端点。

主题

- 查找节点和集群的端点（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 Memcached 集群，此命令将返回配置端点。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

以下命令会检索 Memcached 集群 mycluster 的配置端点（`ConfigurationEndpoint`）和单个节点端点（`Endpoint`）。

```
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>
```

Important

如果您选择为 Memcached 配置端点创建别名记录，以使自动发现客户端可以识别别名记录作为配置端点，则必须在别名记录中包含 `.cfg.`。例如，php.ini 文件中的 `session.save_path` 参数的 `mycluster.cfg.local`。

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集群或最后一次修改集群时选择的引擎。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。

有关要使用的节点大小的更详细讨论，请参阅选择 Memcached 节点大小。

主题

- 查看 ElastiCache 节点状态
- 连接到节点
- 受支持的节点类型
- 替换节点
- ElastiCache 预留节点
- 迁移上一代节点

涉及节点的一些重要操作如下：

- 向集群添加节点
- 从集群中移除节点
- 扩展 Mem ElastiCache cached
- 查找连接端点

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。

类型	已计费	描述
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
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。

类型	已计费	描述
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。

连接到节点

在尝试与 Memcached 集群连接之前，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 telnet 实用工具连接到运行 Memcached 的节点。

> **Note**
>
> 有关 Memcached 以及可用 Memcached 命令的更多信息，请参阅 Memcached 网站。

使用 telnet 连接到节点

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. 在 Amazon EC2 实例下载并安装 telnet 实用工具。在 Amazon EC2 实例的命令提示符处，键入以下命令，然后键入 y。

   ```
   sudo yum install telnet
   ```

   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: 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!
```

3. 在 Amazon EC2 实例的命令提示符处，键入以下命令，并使用节点的端点替换此示例中所示的相应节点端点。

```
telnet mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com 11211
```

此时会显示类似以下内容的输出。

```
Trying 128.0.0.1...
Connected to mycachecluster.eaogs8.0001.usw2.cache.amazonaws.com.
Escape character is '^]'.
>
```

4. 运行 Memcached 命令测试连接。

您现已连接到一个节点，可以运行 Memcached 命令了。以下是示例。

```
set a 0 0 5      // Set key "a" with no expiration and 5 byte value
hello            // Set value as "hello"
STORED
get a            // Get value for key "a"
VALUE a 0 5
hello
END
get b            // Get value for key "b" results in miss
END
>
```

受支持的节点类型

有关要使用的节点大小的信息，请参阅 选择 Memcached 节点大小。

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。

- 通用型:
  - 最新一代：

    M6g 节点类型（仅适用于 Memcached 引擎 1.5.16 以上版本）。

    `cache.m6g.large`, `cache.m6g.xlarge`, `cache.m6g.2xlarge`, `cache.m6g.4xlarge`, `cache.m6g.8xlarge`, `cache.m6g.12xlarge`, `cache.m6g.16xlarge`

    > **Note**
    >
    > 有关区域可用性，请参阅 AWS 区域支持的节点类型。

    M5 节点类型：`cache.m5.large`、`cache.m5.xlarge`、`cache.m5.2xlarge`、`cache.m5.4xlarge`、`cache.`

    M4 节点类型：`cache.m4.large`、`cache.m4.xlarge`、`cache.m4.2xlarge`、`cache.m4.4xlarge`、`cache.`

    T4g 节点类型（仅适用于 Memcached 引擎 1.5.16 以上版本）。

    `cache.t4g.micro`, `cache.t4g.small`, `cache.t4g.medium`

    T3 节点类型：`cache.t3.micro`、`cache.t3.small`、`cache.t3.medium`

    T2 节点类型：`cache.t2.micro`、`cache.t2.small`、`cache.t2.medium`
  - 上一代：（不推荐。这些类型仍支持现有集群，但不支持创建新集群。）

    T1 节点类型：`cache.t1.micro`

    M1 节点类型：`cache.m1.small`、`cache.m1.medium`、`cache.m1.large`、`cache.m1.xlarge`

M3 节点类
型：cache.m3.medium、cache.m3.large、cache.m3.xlarge、cache.m3.2xlarge

- 计算优化:
  - 上一代：（不推荐）

    C1 节点类型：cache.c1.xlarge
- 内存优化:
  - 最新一代：

    （R6g 节点类型仅适用于 Memcached 引擎 1.5.16 以上版本。）

    R6g 节点类
    型：cache.r6g.large、cache.r6g.xlarge、cache.r6g.2xlarge、cache.r6g.4xlarge、ca

    > **Note**
    > 有关区域可用性，请参阅 AWS 区域支持的节点类型。

    R5 节点类
    型：cache.r5.large、cache.r5.xlarge、cache.r5.2xlarge、cache.r5.4xlarge、cache.

    R4 节点类
    型：cache.r4.large、cache.r4.xlarge、cache.r4.2xlarge、cache.r4.4xlarge、cache.
  - 上一代：（不推荐）

    M2 节点类型：cache.m2.xlarge、cache.m2.2xlarge、cache.m2.4xlarge

    R3 节点类
    型：cache.r3.large、cache.r3.xlarge、cache.r3.2xlarge、cache.r3.4xlarge、cache.
- 网络优化：
  - 最新一代：

    （C7gn 节点类型仅适用于 Memcached 引擎 1.6.6 以上版本。）

    C7gn 节点类
    型：cache.c7gn.large、cache.c7gn.xlarge、cache.c7gn.2xlarge、cache.c7gn.4xlarge

最新一代

下表显示了实例类型的基准带宽和突增带宽，这些实例类型通过网络 I/O 积分机制将其基准带宽突增到基准以上。

> **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 Memcached 版本	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.m7g.large		0.937	12.5
cache.m7g.xlarge		1.876	12.5
cache.m7g.2xlarge		3.75	15
cache.m7g.4xlarge		7.5	15
cache.m7g.8xlarge		15	不适用
cache.m7g.12xlarge		22.5	不适用
cache.m7g.16xlarge		30	不适用
cache.m6g.large	1.5.16	0.75	10.0
cache.m6g.xlarge	1.5.16	1.25	10.0
cache.m6g.2xlarge	1.5.16	2.5	10.0
cache.m6g.4xlarge	1.5.16	5.0	10.0
cache.m6g.8xlarge	1.5.16	12	不适用
cache.m6g.12xlarge	1.5.16	20	不适用

实例类型	支持的最低 Memcached 版本	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.m6g.16xlarge	1.5.16	25	不适用
cache.m5.large	1.5.16	0.75	10.0
cache.m5.xlarge	1.5.16	1.25	10.0
cache.m5.2xlarge	1.5.16	2.5	10.0
cache.m5.4xlarge	1.5.16	5.0	10.0
cache.m5.12xlarge	1.5.16	不适用	不适用
cache.m5.24xlarge	1.5.16	不适用	不适用
cache.m4.large	1.5.16	0.45	1.2
cache.m4.xlarge	1.5.16	0.75	2.8
cache.m4.2xlarge	1.5.16	1.0	10.0
cache.m4.4xlarge	1.5.16	2.0	10.0
cache.m4.10xlarge	1.5.16	5.0	10.0
cache.t4g.micro	1.5.16	0.064	5.0
cache.t4g.small	1.5.16	0.128	5.0
cache.t4g.medium	1.5.16	0.256	5.0
cache.t3.micro	1.5.16	0.064	5.0
cache.t3.small	1.5.16	0.128	5.0
cache.t3.medium	1.5.16	0.256	5.0
cache.t2.micro	1.5.16	0.064	1.024
cache.t2.small	1.5.16	0.128	1.024

实例类型	支持的最低 Memcached 版本	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.t2.medium	1.5.16	0.256	1.024

内存优化

实例类型	支持的最低版本	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.r7g.large		0.937	12.5
cache.r7g.xlarge		1.876	12.5
cache.r7g.2xlarge		3.75	15
cache.r7g.4xlarge		7.5	15
cache.r7g.8xlarge		15	不适用
cache.r7g.12xlarge		22.5	不适用
cache.r7g.16xlarge		30	不适用
cache.r6g.large	1.5.16	0.75	10.0
cache.r6g.xlarge	1.5.16	1.25	10.0
cache.r6g.2xlarge	1.5.16	2.5	10.0
cache.r6g.4xlarge	1.5.16	5.0	10.0
cache.r6g.8xlarge	1.5.16	12	不适用
cache.r6g.12xlarge	1.5.16	20	不适用
cache.r6g.16xlarge	1.5.16	25	不适用
cache.r5.large	1.5.16	0.75	10.0
cache.r5.xlarge	1.5.16	1.25	10.0

实例类型	支持的最低版本	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.r5.2xlarge	1.5.16	2.5	10.0
cache.r5.4xlarge	1.5.16	5.0	10.0
cache.r5.12xlarge	1.5.16	20	不适用
cache.r5.24xlarge	1.5.16	25	不适用
cache.r4.large	1.5.16	0.75	10.0
cache.r4.xlarge	1.5.16	1.25	10.0
cache.r4.2xlarge	1.5.16	2.5	10.0
cache.r4.4xlarge	1.5.16	5.0	10.0
cache.r4.8xlarge	1.5.16	12	不适用
cache.r4.16xlarge	1.5.16	25	不适用

网络优化

实例类型	支持的最低版本	基准带宽 (Gbps)	突增带宽 (Gbps)
cache.c7gn.large	1.6.6	6.25	30
cache.c7gn.xlarge	1.6.6	12.5	40
cache.c7gn.2xlarge	1.6.6	25	50
cache.c7gn.4xlarge	1.6.6	50	不适用
cache.c7gn.8xlarge	1.6.6	100	不适用
cache.c7gn.12xlarge	1.6.6	150	不适用
cache.c7gn.16xlarge	1.6.6	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。

相关信息

- Amazon ElastiCache 产品功能与详细信息
- Memcached 节点类型特定参数

替换节点

Amazon ElastiCache for Memcached 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 Memcached 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 Memcached 节点时可执行的操作。

- 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。在 ElastiCache 自动使用新节点替换节点时，新节点最初是空的。
- 更改维护时段 – 对于计划的维护事件，您将收到来自 ElastiCache 的电子邮件或通知事件。在这种情况下，如果在计划替换时间之前更改维护时段，则现在将在新时间替换您的节点。有关更多信息，请参阅修改集 ElastiCache 群。

  > **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。

- 手动替换节点 – 如果您需要在下一维护时段之前替换节点，请手动替换节点。

  如果手动替换节点，则会重新分配密钥。此重新分配会导致缓存丢失。

  手动替换 Memcached 节点

  1. 删除计划替换的节点。有关说明，请参阅 从集群中移除节点。
  2. 向集群添加一个新节点。有关说明，请参阅 向集群添加节点。
  3. 如果您未在此集群上使用自动发现，请参阅您的应用程序，并使用新节点的端点替换旧节点的端点的每个实例。

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 Memcached。
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": "memcached",
          "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": "memcached",
          "OfferingType": "Partial Upfront",
          "RecurringCharges": [
              {
                  "RecurringChargeAmount": X.XXXX,
                  "RecurringChargeFrequency": "Hourly"
              }
          ]
    },
    {
          "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",
          "CacheNodeType": "cache.xx.12xlarge",
          "Duration": 31536000,
          "FixedPrice": X.X,
          "UsagePrice": X.X,
          "ProductDescription": "memcached",
          "OfferingType": "No Upfront",
          "RecurringCharges": [
```

```
                {
                    "RecurringChargeAmount": X.XXXX,
                    "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 Memcached。
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",

    "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 Memcached 集群上的节点

要将 ElastiCache for Memcached 迁移到其他节点类型，您必须创建一个新的集群，该集群始终以空开始，以便您的应用程序可以填充。

要使用 ElastiCache 控制台迁移 ElastiCache for Memcached 集群节点类型：

- 利用新的节点类型创建新集群。有关更多信息，请参阅创建 Memcached 集群（控制台）。
- 在您的应用程序中，将终端节点更新为新集群的终端节点。有关更多信息，请参阅查找集群的端点（控制台）
- 删除旧的集群。有关更多信息，请参阅 删除集群

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

引擎版本和升级

本部分介绍支持的 Memcached 引擎版本以及如何升级。

主题

支持的 ElastiCache for Memcached 版本

ElastiCache 支持以下 Memcached 版本，也支持升级到更新的版本。如果升级到更新的版本，请关注那些会导致升级失败的先决条件。

ElastiCache for Memcached 版本

- Memcached 版本 1.6.22
- Memcached 版本 1.6.17
- Memcached 版本 1.6.12
- Memcached 版本 1.6.6
- Memcached 版本 1.5.16
- Memcached 版本 1.5.10
- Memcached 版本 1.4.34
- Memcached 版本 1.4.33
- Memcached 版本 1.4.24
- Memcached 版本 1.4.14
- Memcached 版本 1.4.5

Memcached 版本 1.6.22

ElastiCache for Memcached 添加了对 Memcached 版本 1.6.22 的支持。该版本未包含任何新功能，但包含对 Memcached 1.6.18 的错误修复和累积更新。

有关更多信息，请在 GitHub 上参阅有关 Memcached 的 ReleaseNotes1622。

Memcached 版本 1.6.17

ElastiCache for Memcached 添加了对 Memcached 版本 1.6.17 的支持。该版本未包含任何新功能，但包含对 Memcached 1.6.17 的错误修复和累积更新。

有关更多信息，请在 GitHub 上参阅 Memcached 中的 ReleaseNotes1617。

Memcached 版本 1.6.12

ElastiCache for Memcached 添加了对 Memcached 版本 1.6.12 和传输中加密的支持。它包括 Memcached 1.6.6 中的错误修复和累积更新。

有关更多信息，请在 GitHub 上参阅 Memcached 中的 ReleaseNotes1612。

Memcached 版本 1.6.6

ElastiCache for Memcached 添加了对 Memcached 版本 1.6.6 的支持。该版本未新增任何新功能，但包含对 Memcached 1.5.16 的错误修复和累积更新。ElastiCache for Memcached 不包含对 Extstore 的支持。

有关更多信息，请参阅 GitHub 上 Memcached 中的 ReleaseNotes166。

Memcached 版本 1.5.16

ElastiCache for Memcached 添加了对 Memcached 版本 1.5.16 的支持。它没有包含任何新功能，但包含了 Memcached 1.5.14 和 Memcached 1.5.15 中的错误修复和累积更新。

有关更多信息，请参阅 GitHub 上 Memcached 中的 Memcached 1.5.16 发布说明。

Memcached 版本 1.5.10

ElastiCache for Memcached 版本 1.5.10 支持以下 Memcached 功能：

- 自动 Slab 重新平衡。
- 使用 `murmur3` 算法实现更快的哈希表查找。
- 经过分段的 LRU 算法。
- LRU 爬网程序到后台回收内存。
- `--enable-seccomp`：一个编译时选项。

它还引入了 `no_modern` 和 `inline_ascii_resp` 参数。有关更多信息，请参阅Memcached 1.5.10 参数更改。

自适用于 Memcached 版本 1.4.34 的 ElastiCache 起增加的 Memcached 改进功能包括：

- 累积修复，例如 ASCII multigets、CVE-2017-9951 和 `metadumper` 的限制爬网。
- 通过在达到连接限制时关闭连接来改善连接管理。
- 改进了超过 1MB 的项大小的项大小管理。
- 通过将每个项的内存需求减少几个字节来提高性能和减少内存开销。

有关更多信息，请参阅 GitHub 上 Memcached 中的 Memcached 1.5.10 发布说明。

Memcached 版本 1.4.34

ElastiCache for Memcached 版本 1.4.34 未在版本 1.4.33 的基础上添加任何功能。1.4.34 版是一个错误修复版本，它大于一般的此类版本。

有关更多信息，请参阅 GitHub 上 Memcached 中的 Memcached 1.4.34 发布说明。

Memcached 版本 1.4.33

自版本 1.4.24 起增加的 Memcached 改进功能包括：

- 能够为特定 slab 类、slab 类列表或所有 slab 类转储所有元数据。有关更多信息，请参阅 Memcached 1.4.31 发布说明。
- 改进对超过 1 MB 默认值的大项目的支持。有关更多信息，请参阅 Memcached 1.4.29 发布说明。
- 能够指定在要求关闭客户端之前，客户端可以保持空闲的时间长度。

  能够动态增加可供 Memcached 使用的内存量而无需重新启动集群。有关更多信息，请参阅 Memcached 1.4.27 发布说明。
- 现在支持 `fetchers`, `mutations` 和 `evictions` 的日志记录。有关更多信息，请参阅 Memcached 1.4.26 发布说明。
- 释放的内存可回收到全局池中并重新分配到新的 slab 类。有关更多信息，请参阅 Memcached 1.4.25 发布说明。
- 修复了几个 Bug。
- 一些新的命令和参数。有关列表，请参阅Memcached 1.4.33 增加的参数。

Memcached 版本 1.4.24

自版本 1.4.14 起增加的 Memcached 改进功能包括：

- 使用后台进程的最近最少使用（LRU）的管理。
- 增加了使用 jenkins 或 murmur3 作为哈希算法的选项。
- 一些新的命令和参数。有关列表，请参阅Memcached 1.4.24 增加的参数。
- 修复了几个 Bug。

Memcached 版本 1.4.14

自版本 1.4.5 起增加的 Memcached 改进功能包括：

- 增强了 Slab 重新平衡功能。
- 性能和可扩展性方面的改进。
- 引入了 touch 命令，可在不获取的情况下更新现有项目的过期时间。
- Auto Discovery – 客户端程序功能，可以自动确定集群中的所有缓存节点，以及启动和维护与所有这些节点的连接。

Memcached 版本 1.4.5

Amazon ElastiCache for Memcached 初始支持的引擎和版本是 Memcached 版本 1.4.5。

引擎版本和升级

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。

要升级到更新的 Memcached 版本，请修改缓存群集，同时指定要使用的新引擎版本。升级到更新的 Memcached 版本是一个破坏性过程 – 您会丢失数据并以冷缓存开始。有关更多信息，请参阅修改集群。

从旧版本的 Memcached 升级到 1.4.33 版本或更新版本的 Memcached 时，您应注意以下要求。`CreateCacheCluster` 和 `ModifyCacheCluster` 在下列情况下将失败：

- 如果 `slab_chunk_max > max_item_size`。
- 如果 `max_item_size modulo slab_chunk_max != 0`。
- 如果 `max_item_size > ((max_cache_memory - memcached_connections_overhead) / 4)`。

  `(max_cache_memory - memcached_connections_overhead)` 值是可用于数据的节点内存。有关更多信息，请参阅Memcached 连接开销。

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。
- 由于 Memcached 引擎不支持持久化，因此 Memcached 引擎版本升级始终是一个中断性过程，将清除集群中的所有缓存数据。

如何升级引擎版本

要启动对集群的版本升级，请对其进行修改并指定较新的引擎版本。您可以使用 ElastiCache 控制台、AWS CLI 或 ElastiCache API 执行此步骤：

- 要使用 AWS Management Console，请参阅通过控制台修改集群。
- 要使用 AWS CLI，请参阅通过 CLI 修改集群。
- 要使用 ElastiCache API，请参阅通过 API 修改集群。

如何升级引擎版本

要启动对集群的版本升级，请对其进行修改并指定较新的引擎版本。您可以使用 ElastiCache 控制台、AWS CLI 或 ElastiCache API 执行此步骤：

- 要使用 AWS Management Console，请参阅 – 使用 AWS Management Console。
- 要使用 AWS CLI，请参阅使用 AWS CLI。
- 要使用 ElastiCache API，请参阅 使用 ElastiCache API。

ElastiCache 最佳实践和缓存策略

您可以在下面找到建议的 Amazon ElastiCache 最佳实践。遵循这些最佳实践可提高您的缓存的性能和可靠性。

主题

- 有关 Memcached 客户端的最佳实践
- 支持的 Memcached 命令
- 缓存策略

有关 Memcached 客户端的最佳实践

要详细了解有关使用常用开源 Memcached 客户端库与 ElastiCache 资源进行交互的最佳实践，请参阅以下主题。

主题

- 配置 ElastiCache 客户端以实现高效负载均衡
- IPv6 客户端示例

配置 ElastiCache 客户端以实现高效负载均衡

> **Note**
>
> 此部分适用于自行设计的多节点 Memcached 集群。

为了有效使用多个 ElastiCache Memcached 节点，您需要能够跨节点分布缓存键。对具有 n 个节点的集群进行负载均衡的一个简单方法是，计算该对象的密钥的哈希并通过 n - hash(key) mod n 修改结果。所得的值（0 到 n–1）是您放置对象的节点的编号。

只要节点数 (n) 恒定，这种方法就简便可行。但是，每当您向集群添加节点或从集群删除节点时，需要移动的键数都是 (n - 1)/n（其中，n 是新的节点数）。因此，这种方法会造成移动大量键，并转变为很大的初始缓存未命中数，尤其是当节点数变大时。从 1 个节点扩展为 2 个节点会导致 (2-1)/2 (50%) 的键移动，这是最好的情况。从 9 个节点扩展为 10 个节点会导致 (10–1)/10 (90%) 的键移动。如果因为峰值流量而纵向扩展，您不会希望缓存未命中数变得很大。缓存未命中数很大会造成数据库命中，而数据库已经因峰值流量而过载。

这种两难困境的解决方法是使用一致性哈希。一致性哈希使用这样的算法：每当向集群添加节点或从集群删除节点时，必须移动的键的数目大致为 1/n（其中，n 是新的节点数）。从 1 个节点扩展为 2 个节点会造成移动 1/2 (50%) 的键，这是最差的情况。从 9 个节点扩展为 10 个节点会造成移动 1/10 (10%) 的键。

作为用户，您可以控制对多节点集群使用哪种哈希算法。建议您将客户端配置为使用一致性哈希。幸运的是，有很多用最常见语言编写的 Memcached 客户端库可以实现一致性哈希。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。

如果使用的是 Java、PHP 或 .NET，建议您使用 Amazon ElastiCache 客户端库之一。

使用 Java 实现一致性哈希

ElastiCache Memcached Java 客户端基于开源 Spymemcached Java 客户端，内置有一致性哈希功能。该库包含一个实现一致性哈希的 KetamaConnectionFactory 类。默认情况下，spymemcached 中禁用一致性哈希。

有关更多信息，请参阅 KetamaConnectionFactory 中的 KetamaConnectionFactory 文档。

使用 PHP 实现一致性哈希

ElastiCache Memcached PHP 客户端是一个围绕内置 Memcached PHP 库的包装程序。默认情况下，Memcached PHP 库禁用一致性哈希。

使用以下代码可以启用一致性哈希。

```
$m = new Memcached();
$m->setOption(Memcached::OPT_DISTRIBUTION, Memcached::DISTRIBUTION_CONSISTENT);
```

除了上述代码之外，我们还建议您在 php.ini 文件中启用 `memcached.sess_consistent_hash`。

有关更多信息，请参阅 Memcached PHP 的运行时配置文档，网址为 http://php.net/manual/en/memcached.configuration.php。请特别注意 `memcached.sess_consistent_hash` 参数。

使用 .NET 实现一致性哈希

ElastiCache Memcached .NET 客户端是一个围绕 Enyim Memcached 的包装程序。默认情况下，Enyim Memcached 客户端已启用一致性哈希。

有关更多信息，请参阅 `memcached/locator` 文档 https://github.com/enyim/EnyimMemcached/wiki/MemcachedClient-Configuration#user-content-memcachedlocator。

IPv6 客户端示例

> **Note**
>
> 此部分适用于自行设计的 Memcached 集群。

ElastiCache 与开源 Memcached 兼容。这意味着支持 IPv6 连接的 Memcached 开源客户端应该能够连接到为 Memcached 集群启用 ElastiCache 的 IPv6。此外，以下客户端经过专门验证，可与所有支持的网络类型配置配合使用：

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

经过验证的客户端：

- AWS ElastiCache 适用于 Php 的集群客户端 Memcach ed — 版本 *3.6.2
- AWS ElastiCache 适用于 Java 的集群客户端 Memcached — Github

为双堆栈集群配置首选协议

对于 Memcached 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。

IP 发现参数控制 config get 集群输出中使用的 IP 协议。这反过来又将决定支持自动发现 Memcached 集群的客户端使用 ElastiCache 的 IP 协议。

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。

监视适用于 Java 和 Php 的 `getAvailableNodeEndPoints` 的输出，并监视 `getServerList` 的输出。一旦这些函数的输出报告了集群中使用更新协议的所有节点的 IP，更改即已完成传播。

Java 示例：

```
MemcachedClient client = new MemcachedClient(new InetSocketAddress("xxxx", 11211));

Class targetProtocolType = Inet6Address.class; // Or Inet4Address.class if you're
 switching to IPv4

Set<String> nodes;

do {
    nodes =
 client.getAvailableNodeEndPoints().stream().map(NodeEndPoint::getIpAddress).collect(Collectors

    Thread.sleep(1000);
} while (!nodes.stream().allMatch(node -> {
            try {
                return finalTargetProtocolType.isInstance(InetAddress.getByName(node));
            } catch (UnknownHostException ignored) {}
            return false;
        }));
```

Php 示例：

```
$client = new Memcached;
$client->setOption(Memcached::OPT_CLIENT_MODE, Memcached::DYNAMIC_CLIENT_MODE);
$client->addServer("xxxx", 11211);

$nodes = [];
$target_ips_count = 0;
do {
    # The PHP memcached client only updates the server list if the polling interval has
 expired and a
    # command is sent
    $client->get('test');
```

```
    $nodes = $client->getServerList();

    sleep(1);
    $target_ips_count = 0;

    // For IPv4 use FILTER_FLAG_IPV4
    $target_ips_count = count(array_filter($nodes, function($node) { return
 filter_var($node["ipaddress"], FILTER_VALIDATE_IP, FILTER_FLAG_IPV6); }));

} while (count($nodes) !== $target_ips_count);
```

在 IP 发现更新之前创建的任何现有客户端连接仍将使用旧协议进行连接。一旦在集群发现命令的输出中检测到更改，所有经过验证的客户端都将使用新的 IP 协议自动重新连接到集群。但是，这取决于客户端的实现。

启用 TLS 的双堆栈 ElastiCache 集群

为 ElastiCache 集群启用 TLS 后，集群发现功能将`config get cluster`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。

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 CIDR`netsh 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
```

支持的 Memcached 命令

ElastiCache for Memcached 无服务器支持开源 Memcached 1.6 中的所有 memcached 命令，但以下命令除外：

- 客户端连接需要 TLS，因此不支持 UDP 协议。
- 不支持二进制协议，因为在 memcached 1.6 中已正式弃用该协议。
- GET/GETS 命令限制为 16KB，以避免可能通过提取大量键来对服务器进行 DoS 攻击。
- 延迟的 `flush_all` 命令将被拒绝并返回 `CLIENT_ERROR`。
- 配置引擎的命令，或者显示有关引擎状态或日志的内部信息的命令均不受支持，例如：
  - 对于 STATS 命令，仅支持 `stats` 和 `stats reset`。其他变体将返回 `ERROR`
  - `lru / lru_crawler`：修改 LRU 和 LRU 爬网程序设置
  - `watch`：监视 memcached 服务器日志
  - `verbosity`：配置服务器日志级别

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。Memcached 指定此值（以秒为单位）。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 Memcached `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
- 扩展 Mem ElastiCache cached

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

- 管理维护
- 使用参数组配置引擎参数

管理维护

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

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

区域代码	区域名称	区域维护时段
ap-northeast-1	亚太（东京）区域	13:00–21:00 UTC

区域代码	区域名称	区域维护时段
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
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

区域代码	区域名称	区域维护时段
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

更改您的集群的维护时段

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

> **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
- 修改集 ElastiCache 群 – 更改集群的维护时段

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Memcached 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 Memcached 节点类型的参数。

> **Note**
>
> 有关 Memcached 特定的参数的列表，请参阅 Memcached 特定的参数。

主题

参数管理

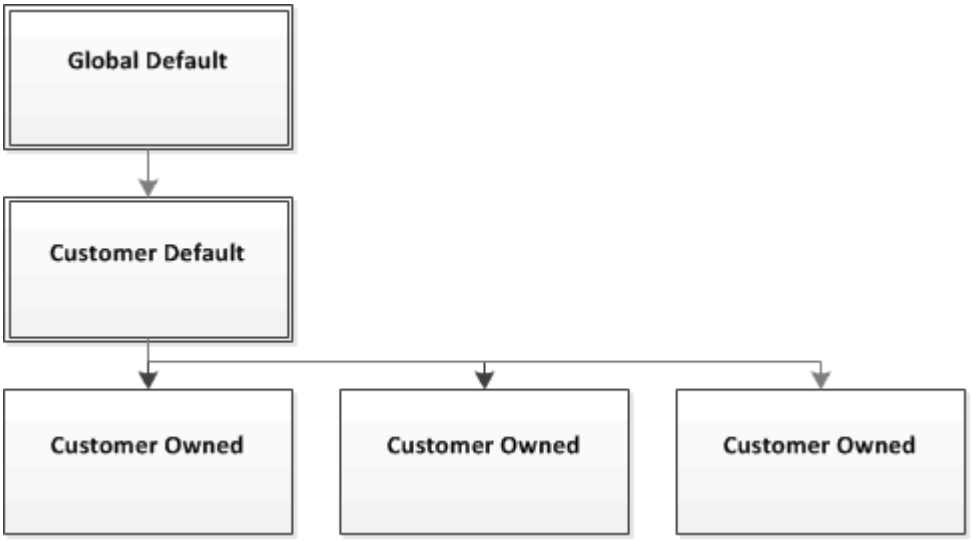
参数已分组到指定的参数组中，以便更轻松地管理参数。参数组表示在启动期间传递给引擎软件的参数的特定值组合。这些值确定每个节点上的引擎进程在运行时的行为方式。特定参数组中的参数值应用于与该组关联的所有节点（不论这些节点属于哪个集群）。

要优化集群的性能，您可以修改某些参数值或更改集群的参数组。

- 您无法修改或删除默认参数组。如果您需要自定义参数值，则必须创建自定义参数组。
- 参数组系列与您分配给参数组的集群必须兼容。例如，如果您的集群运行 Memcached 版本 1.4.8，您只能使用 Memcached 1.4 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，更改将立即应用于集群。无论是更改集群的参数组本身还是更改集群参数组中的参数值，都是如此。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 Memcached 特定的参数。有关重启集群节点的信息，请参阅重启集群。

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Memcached 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 memcached1.4 ）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 memcached1.4 系列作为模板来创建名为 myMem14 的参数组。

对于 Linux、macOS 或 Unix：

```
aws elasticache create-cache-parameter-group \
    --cache-parameter-group-name myMem14  \
    --cache-parameter-group-family memcached1.4 \
    --description "My first parameter group"
```

对于 Windows：

```
aws elasticache create-cache-parameter-group ^
    --cache-parameter-group-name myMem14  ^
    --cache-parameter-group-family memcached1.4 ^
    --description "My first parameter group"
```

该命令的输出内容应类似如下所示。

```
{
    "CacheParameterGroup": {
        "CacheParameterGroupName": "myMem14",
        "CacheParameterGroupFamily": "memcached1.4",
        "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，memcached1.4。
- Description – 用户提供的参数组描述。

Example

以下示例使用 memcached1.4 系列作为模板来创建名为 myMem14 的参数组。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateCacheParameterGroup
   &CacheParameterGroupFamily=memcached1.4
   &CacheParameterGroupName=myMem14
   &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>myMem14</CacheParameterGroupName>
      <CacheParameterGroupFamily>memcached1.4</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 myMem14。

对于 Linux、macOS 或 Unix：

```
aws elasticache describe-cache-parameter-groups \
    --cache-parameter-group-name myMem14
```

对于 Windows：

```
aws elasticache describe-cache-parameter-groups ^
    --cache-parameter-group-name myMem14
```

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": "myMem14",
```

```
         "CacheParameterGroupFamily": "memcached1.4",
         "Description": "My first parameter group"
     }
    ]
}
```

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

以下示例代码列出了参数组 myMem14。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameterGroups
   &CacheParameterGroupName=myMem14
   &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，列出每个参数组的名称、系列和描述。

```
<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
```

```
      <CacheParameterGroup>
        <CacheParameterGroupName>myMem14</CacheParameterGroupName>
        <CacheParameterGroupFamily>memcached1.4</CacheParameterGroupFamily>
        <Description>My custom Memcached 1.4 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>
```

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 myMem14 的所有参数及其值。

对于 Linux、macOS 或 Unix：

```
aws elasticache describe-cache-parameters \
    --cache-parameter-group-name myMem14
```

对于 Windows：

```
aws elasticache describe-cache-parameters ^
    --cache-parameter-group-name myMem14
```

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 myMem14 的所有参数。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameters
   &CacheParameterGroupName=myMem14
   &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Memcached 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 `binding_protocol` 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. 在 Edit Parameter Group:（编辑参数组：）屏幕上，使用左箭头和右箭头滚动找到 `binding_protocol` 参数，然后在 Value（值）列中键入 `ascii`。
6. 在 Edit Parameter Group:（编辑参数组：）屏幕上，使用左箭头和右箭头滚动找到 `cluster-enabled` 参数，然后在 Value（值）列中键入 `yes`。

7. 选择 Save Changes。
8. 要查找您更改的参数名称，请参阅Memcached 特定的参数。如果参数更改在重新启动 生效，则重启使用此参数组的所有集群。有关更多信息，请参阅重启集群。

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Memcached 特定的参数

以下示例代码演示设置两个参数的值：参数组 myMem14 的 chunk_size 和 chunk_size_growth_fact。

对于 Linux、macOS 或 Unix：

```
aws elasticache modify-cache-parameter-group \
    --cache-parameter-group-name myMem14 \
    --parameter-name-values \
        ParameterName=chunk_size,ParameterValue=96 \
        ParameterName=chunk_size_growth_fact,ParameterValue=1.5
```

对于 Windows：

```
aws elasticache modify-cache-parameter-group ^
    --cache-parameter-group-name myMem14 ^
    --parameter-name-values ^
        ParameterName=chunk_size,ParameterValue=96 ^
        ParameterName=chunk_size_growth_fact,ParameterValue=1.5
```

此命令的输出如下所示。

```
{
    "CacheParameterGroupName": "myMem14"
}
```

有关更多信息，请参阅modify-cache-parameter-group。

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Memcached 特定的参数

以下示例代码演示设置两个参数的值：参数组 myMem14 的 chunk_size 和 chunk_size_growth_fact。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyCacheParameterGroup
   &CacheParameterGroupName=myMem14
   &ParameterNameValues.member.1.ParameterName=chunk_size
   &ParameterNameValues.member.1.ParameterValue=96
   &ParameterNameValues.member.2.ParameterName=chunk_size_growth_fact
   &ParameterNameValues.member.2.ParameterValue=1.5
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

有关更多信息，请参阅ModifyCacheParameterGroup。

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 myMem14
```

对于 Windows：

```
aws elasticache delete-cache-parameter-group ^
    --cache-parameter-group-name myMem14
```

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 myMem14 参数组。

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DeleteCacheParameterGroup
   &CacheParameterGroupName=myMem14
   &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。

扩展 Mem ElastiCache cached

扩展 Mem ElastiCache cached

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并且缓存数据存储空间达到最大值时， ElastiCache 将开始使用 LRU 逻辑逐出缓存中的数据。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 Memcached 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 Mem ElastiCache cached 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 Memcached 集群

操作	主题
横向扩展	向集群添加节点
缩减	从集群中删除节点
更改节点类型	纵向扩展 Memcached

Memcached 集群由 1 到 60 个节点组成。横向扩展和收缩 Memcached 集群很简单，只需在集群中添加或删除节点即可。

如果您在 Memcached 集群中需要超过 60 个节点，或者一个 AWS 区域中总共需要超过 300 个节点，请填写 https://aws.amazon.com/contact-us/elasticache-node-limit-request/上 ElastiCache 限申请表。

由于您可以将数据分区到 Memcached 集群中的所有节点，因此几乎不需要扩展到具有更大内存的节点类型。但是，由于 Memcached 引擎不保存数据，如果您扩展到不同类型的节点，除非您的应用程序填充该集群，否则它在启动时将为空。

主题

- 横向扩展 Memcached
- 纵向扩展 Memcached

横向扩展 Memcached

Memcached 引擎支持将您的数据分区到多个节点。因此，可以轻松地横向扩展 Memcached 集群。一个 Memcached 集群可以有 1 到 60 个节点。要横向扩展您的 Memcached 集群，只需添加或移除节点。

如果您在 Memcached 集群中需要超过 60 个节点，或者一个 AWS 区域中总共需要超过 300 个节点，请填写 https://aws.amazon.com/contact-us/elasticache-node-limit-request/上 ElastiCache 限申请表。

以下主题详细介绍了如何通过添加或移除节点来横向扩展或收缩 Memcached 集群。

- 向集群添加节点
- 从集群中删除节点

每次更改您的 Memcached 集群中的节点数时，您必须至少重新映射部分密钥空间，以便它映射到正确的节点。有关对 Memcached 集群进行负载均衡的更多详细信息，请参阅配置 ElastiCache 客户端以实现高效负载均衡。

如果您在 Memcached 集群上使用 Auto Discovery，则您在添加或移除节点时无需更改应用程序中的终端节点。有关自动发现的更多信息，请参阅自动识别集群中的节点。如果您不使用自动发现，则每次更改 Memcached 集群中的节点数后，您都必须更新应用程序中的端点。

纵向扩展 Memcached

在纵向扩展或缩减 Memcached 集群时，必须创建新的集群。除非您的应用程序填充 Memcached 集群，否则它在启动时始终为空。

> **Important**
>
> 如果您要缩减到较小的节点类型，请确保较小的节点类型足以满足您的数据和开销。有关更多信息，请参阅选择缓存节点大小。

主题

- 纵向扩展 Memcached（控制台）
- 纵向扩展 Memcached (AWS CLI)
- 垂直缩放 Memcached (ElastiCache API)

纵向扩展 Memcached（控制台）

以下过程将引导您使用 ElastiCache 控制台垂直扩展集群。

纵向扩展 Memcached 集群（控制台）

1. 利用新的节点类型创建新集群。有关更多信息，请参阅创建 Memcached 集群（控制台）。
2. 在您的应用程序中，将终端节点更新为新集群的终端节点。有关更多信息，请参阅查找集群的端点（控制台）。
3. 删除旧的集群。有关更多信息，请参阅删除 Memcached 中的新节点。

纵向扩展 Memcached (AWS CLI)

以下过程演示了如何使用 AWS CLI 垂直扩展 Memcached 缓存群集。

纵向扩展 Memcached 缓存群集 (AWS CLI)

1. 利用新的节点类型创建新的缓存群集。有关更多信息，请参阅使用 CLI 创建集群。
2. 在您的应用程序中，将终端节点更新为新集群的终端节点。有关更多信息，请参阅查找端点（AWS CLI）。
3. 删除旧缓存群集。有关更多信息，请参阅使用 AWS CLI。

垂直缩放 Memcached (ElastiCache API)

以下过程将引导您使用 ElastiCache API 垂直扩展 Memcached 缓存集群。

垂直扩展 Memcached 缓存集群 (ElastiCache API)

1. 利用新的节点类型创建新的缓存群集。有关更多信息，请参阅创建集群 (ElastiCache API)。
2. 在您的应用程序中，将终端节点更新为新缓存群集的终端节点。有关更多信息，请参阅查找端点（ElastiCache API）。
3. 删除旧缓存群集。有关更多信息，请参阅 使用 ElastiCache API。

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`
- `CreateCacheParameterGroup`
- `CreateCacheSecurityGroup`
- `CreateCacheSubnetGroup`
- `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

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 memcached
    --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"
```

- 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 memcached \
--tags Key="project",Value="XYZ" Key="Elasticache",Value="Service"
```

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. 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 Memcached 集群添加 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集群。

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>`

  Memcached：标签仅应用到集群。

  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。

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`

示例 arn：`arn:aws:elasticache:us-west-2:1234567890:cluster:my-cluster`

主题

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 Memcached 集群 `my-cluster` 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
```

对于 Windows：

```
aws elasticache list-tags-for-resource ^
  --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster
```

此操作的输出类似于下文，即列出资源上的所有标签。

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

如果资源上没有任何标签，则输出空标签列表。

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

有关更多信息，请参阅适用于 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 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 \
 --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 ^
 --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 Memcached 集群上的标签。

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 操作删除现有 ElastiCache for Memcached 集群的标签。

下面的代码使用 AWS CLI 移除了 us-west-2 区域中集群 `my-cluster` 的 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 \
 --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 ^
```

```
  --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` 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
   &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` 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
   &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 Memcached 集群的标签。

以下代码使用 ElastiCache API 从 us-west-2 区域中集群 `my-cluster` 中的 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
   &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 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？
- REL 2：您如何使用 ElastiCache 实现恢复点目标（RPO）？
- REL 3：您如何支持灾难恢复（DR）要求？
- REL 4：如何有效地规划失效转移？

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？
- PE 2：如何在 ElastiCache 集群节点间分配工作？
- 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？
- PE 4：您的工作负载如何优化网络资源和连接的使用？
- PE 5：如何管理键删除和/或驱逐？
- PE 6：如何在 ElastiCache 中对数据建模以及与数据交互？
- PE 7：如何在 Amazon ElastiCache 集群中记录运行缓慢的命令？
- PE8：自动扩缩如何帮助提高 ElastiCache 集群的性能？

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] 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 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]：

- [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]：

  - 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
  - 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、rpr
  - 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）。

Amazon ElastiCache 中的安全性

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

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

- 云的安全性 – AWS 负责保护在 AWS 云中运行 AWS 服务的基础设施。AWS 还向您提供可安全使用的服务。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 Amazon ElastiCache 的合规性计划，请参阅合规性计划范围内的 AWS 服务。
- 云中的安全性：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，包括数据的敏感性、公司的要求以及适用的法律法规。

该文档帮助您了解如何在使用 Amazon ElastiCache 时应用责任共担模式。以下主题说明如何配置 Amazon ElastiCache 以实现您的安全性和合规性目标。您还会了解如何使用帮助您监控和保护 Amazon ElastiCache 资源的其他 AWS 服务。

主题

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 Memcached 还为运行 Memcached 版本 1.6.12 或更高版本的缓存中的数据提供了加密功能。

- 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。

主题

ElastiCache 传输中加密（TLS）

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。通过传输中加密功能，ElastiCache 为您提供了在不同位置之间移动数据时用来保护数据的工具。

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，在使用 `CreateCacheCluster`（CLI：`create-cache-cluster`）操作创建缓存群集时，您可以将参数 `TransitEncryptionEnabled` 设置为 `true`（CLI：`--transit-encryption-enabled`），从而在缓存群集上启用传输中加密。

主题

传输中加密概览

Amazon ElastiCache 传输中加密功能可让您在数据最脆弱时候（从一个位置传输到另一个位置时）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 传输中加密可实现以下功能：

- 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。

传输中加密的条件

在规划您的自行设计集群的实施时，应注意有关 Amazon ElastiCache 传输中加密的以下限制：

- 在运行 Memcached 1.6.12 及更高版本的集群上支持传输中加密。

- 传输中加密支持传输层安全性协议（TLS）版本 1.2 和 1.3。
- 只有在 Amazon VPC 中运行的集群才支持传输中加密。
- 只有运行以下节点类型的集群才支持传输中加密。
  - R6g、R5、R4
  - M6g、M5、M4
  - T4g、T3

  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。
- 只有在创建集群时，才能在集群上启用传输中加密。无法通过修改集群来开启和关闭传输中加密。
- 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。

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。

启用传输中加密

要在使用 AWS 管理控制台创建 Memcached 集群时启用传输中加密，请进行以下选择：

- 选择 Memcached 作为引擎。
- 选择 1.6.12 或更高的引擎版本。
- 在 Encryption in transit（传输中加密）下，选择 Enable（启用）。

有关此分步过程，请参阅创建 Memcached 集群（控制台）。

使用 Openssl 连接到启用了传输中加密的节点

要从启用了传输中加密的 ElastiCache for Memcached 节点中访问数据，您需要使用利用安全套接字层（SSL）的客户端。您也可以在 Amazon Linux 和 Amazon Linux 2 上使用 Openssl s_client。

使用 Openssl s_client 连接到 Amazon Linux 2 或 Amazon Linux 上启用了传输中加密的 Memcached 集群：

```
/usr/bin/openssl s_client -connect memcached-node-endpoint:memcached-port
```

使用 Java 创建 TLS Memcached 客户端

要在 TLS 模式下创建客户端，请执行以下操作以利用适当的 SSLContext 初始化该客户端：

```
import java.security.KeyStore;
import javax.net.ssl.SSLContext;
import javax.net.ssl.TrustManagerFactory;
import net.spy.memcached.AddrUtil;
import net.spy.memcached.ConnectionFactoryBuilder;
import net.spy.memcached.MemcachedClient;
public class TLSDemo {
    public static void main(String[] args) throws Exception {
        ConnectionFactoryBuilder connectionFactoryBuilder = new
 ConnectionFactoryBuilder();
        // Build SSLContext
        TrustManagerFactory tmf =
 TrustManagerFactory.getInstance(TrustManagerFactory.getDefaultAlgorithm());
        tmf.init((KeyStore) null);
        SSLContext sslContext = SSLContext.getInstance("TLS");
        sslContext.init(null, tmf.getTrustManagers(), null);
        // Create the client in TLS mode
        connectionFactoryBuilder.setSSLContext(sslContext);
        MemcachedClient client = new MemcachedClient(connectionFactoryBuilder.build(),
 AddrUtil.getAddresses("mycluster.fnjyzo.cfg.use1.cache.amazonaws.com:11211"));

        // Store a data item for an hour.
        client.set("theKey", 3600, "This is the data value");
    }
}
```

使用 PHP 创建 TLS Memcached 客户端

要在 TLS 模式下创建客户端，请执行以下操作以利用适当的 SSLContext 初始化该客户端：

```
<?php

/**
 * Sample PHP code to show how to create a TLS Memcached client. In this example we
 * will use the Amazon ElastiCache Auto Descovery feature, but TLS can also be
 * used with a Static mode client.
```

```
 * See Using the ElastiCache Cluster Client for PHP (https://docs.aws.amazon.com/
AmazonElastiCache/latest/mem-ug/AutoDiscovery.Using.ModifyApp.PHP.html) for more
 information
 * about Auto Discovery and persistent-id.
 */

/* Configuration endpoint to use to initialize memcached client.
 * this is only an example */
$server_endpoint = "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com";

/* Port for connecting to the cluster.
 * This is only an example     */
$server_port = 11211;

/* Initialize a persistent Memcached client and configure it with the Dynamic client
 mode  */
$tls_client =  new Memcached('persistent-id');
$tls_client->setOption(Memcached::OPT_CLIENT_MODE, Memcached::DYNAMIC_CLIENT_MODE);

/* Add the memcached's cluster server/s */
$tls_client->addServer($server_endpoint, $server_port);

/* Configure the client to use TLS */
if(!$tls_client->setOption(Memcached::OPT_USE_TLS, 1)) {
    echo $tls_client->getLastErrorMessage(), "\n";
    exit(1);
}

/* Set your TLS context configurations values.
 * See MemcachedTLSContextConfig in memcached-api.php for all configurations */
$tls_config = new MemcachedTLSContextConfig();
$tls_config->hostname = '*.mycluster.fnjyzo.use1.cache.amazonaws.com';
$tls_config->skip_cert_verify = false;
$tls_config->skip_hostname_verify = false;

/* Use the created TLS context configuration object to create OpenSSL's SSL_CTX and set
 it to your client.
 * Note:  These TLS context configurations will be applied to all the servers connected
 to this client. */
$tls_client->createAndSetTLSContext((array)$tls_config);

/* test the TLS connection with set-get scenario: */

 /* store the data for 60 seconds in the cluster.
```

```
 * The client will decide which cache host will store this item.
 */
if($tls_client->set('key', 'value', 60)) {
    print "Successfully stored key\n";
} else {
    echo "Failed to set key: ", $tls_client->getLastErrorMessage(), "\n";
    exit(1);
}

/* retrieve the key */
if ($tls_client->get('key') === 'value') {
    print "Successfully retrieved key\n";
} else {
    echo "Failed to get key: ", $tls_client->getLastErrorMessage(), "\n";
    exit(1);
}
```

有关使用 PHP 客户端的更多信息，请参阅 安装适用于 PHP 的 ElastiCache Cluster Client。

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 Amazon ElastiCache 的 Identity and Access Management：

- 同步和交换操作期间的磁盘

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。

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：

- 只有无服务器缓存上支持静态加密。
- 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。

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。

- 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。
- 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。

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

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。

另请参阅

- Amazon VPC 和 ElastiCache 安全性
- 适用于 Amazon ElastiCache 的 Identity and Access Management

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 Amazon ElastiCache 的 Identity and Access Management 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
- 用于访问 Amazon VPC 中的 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。

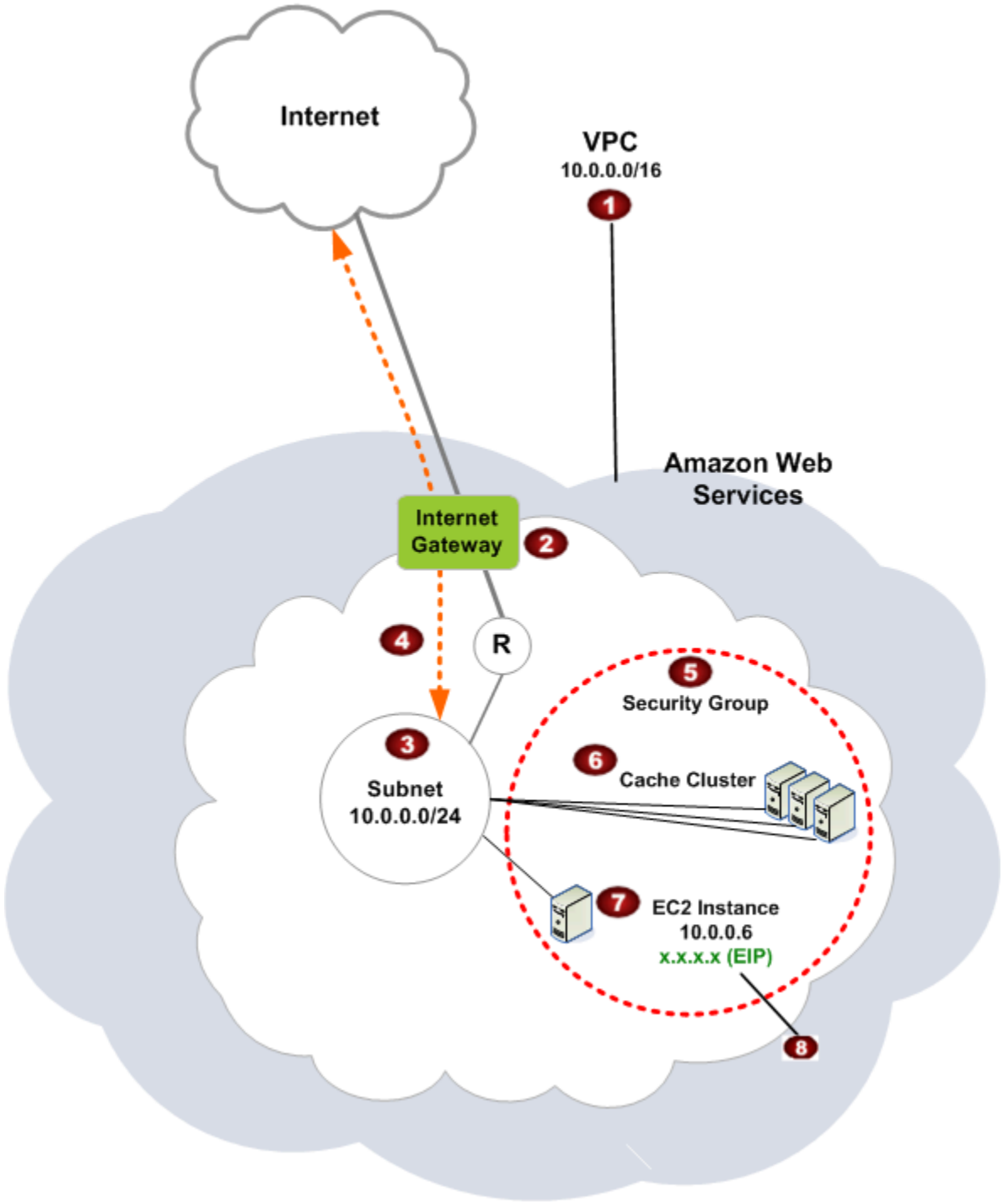
使用 Amazon Virtual Private Cloud，您可以在非常类似于传统数据中心的 AWS 云中创建虚拟网络。您可以配置您的 Amazon VPC，包括选择其 IP 地址范围、创建子网以及配置路由表、网关和安全设置。

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。

用于访问 Amazon VPC 中的 ElastiCache 缓存的访问模式

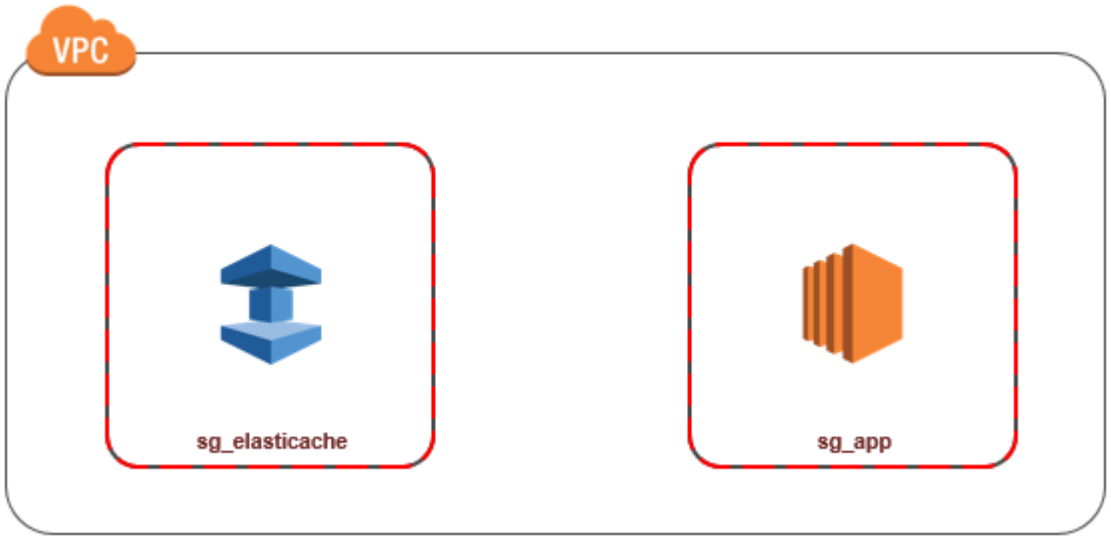
Amazon ElastiCache 支持以下访问 Amazon VPC 中的缓存的场景：

目录

当 ElastiCache 缓存和 Amazon EC2 实例位于同一 Amazon VPC 中时访问该缓存

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。

   Memcached 缓存的默认端口为 11211。
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。

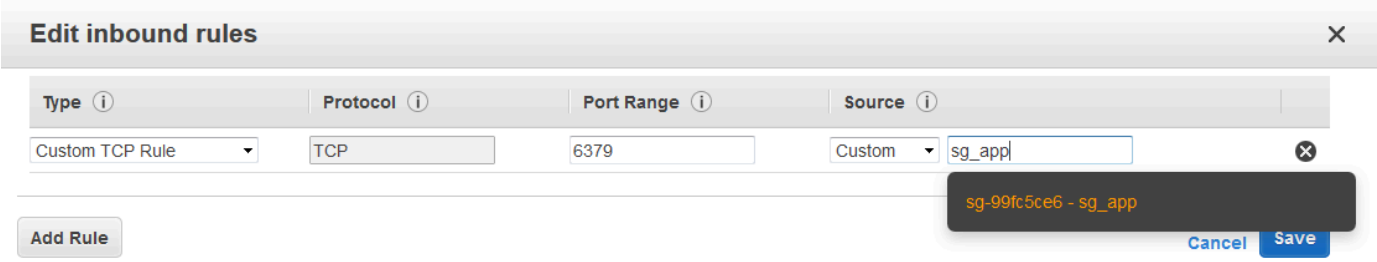
在 VPC 安全组中创建允许从另一安全组连接的规则

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。

      Memcached 缓存的默认端口为 11211。

   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 中时访问该缓存

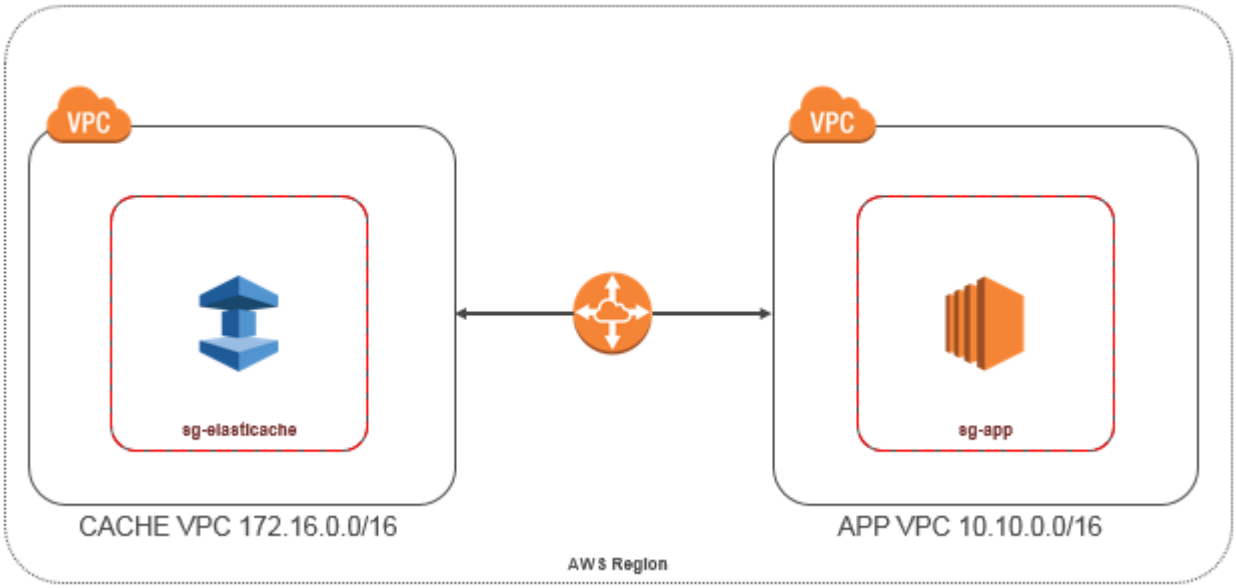
当您的缓存与用来访问它的 EC2 实例位于不同的 VPC 中时，可通过多种方式访问缓存。如果缓存和 EC2 实例位于同一区域的不同的 VPC 中，可以使用 VPC 对等连接。如果缓存和 EC2 实例位于不同的区域中，您可以在两个区域之间创建 VPN 连接。

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
- 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

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。您可以在您自己的 Amazon VPC 之间创建 VPC 对等连接，也可以在它们与同一区域内其他 AWS 账户中的 Amazon VPC 之间创建该连接。要了解有关 Amazon VPC 对等连接的更多信息，请参阅 VPC 文档。

> **Note**
>
> 对等 VPC 的 DNS 名称解析可能会失败，具体取决于应用于 ElastiCache VPC 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

通过中转网关，您可以连接同一 AWS 区域中的 VPC 与 VPN 连接并在它们之间路由流量。中转网关可跨 AWS 账户发挥作用，您可以使用 AWS Resource Access Manager 与其他账户共享您的中转网

关。在您与另一个 AWS 账户共享中转网关后，账户拥有者可以将其 VPC 挂载到您的中转网关。任一账户的用户都可以随时删除此挂载。

您可以在中转网关上启用多播，然后创建一个中转网关多播域，允许通过与域关联的 VPC 挂载，将多播流量从多播源发送到多播组成员。

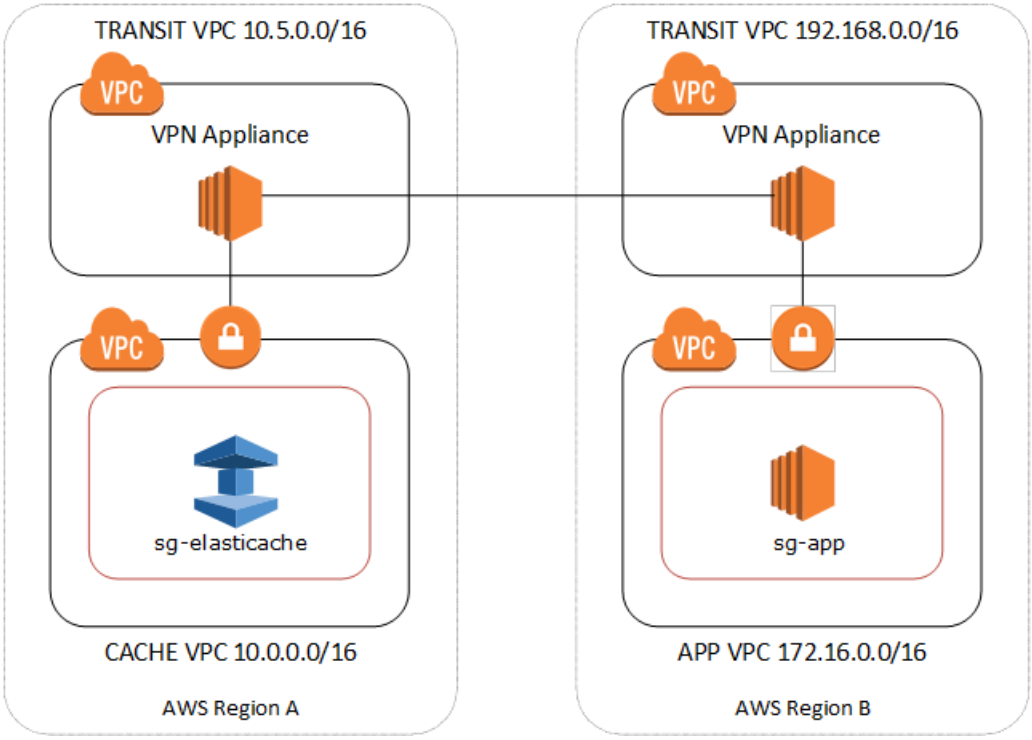
您还可以在不同 AWS 区域的中转网关之间创建对等连接挂载。这使您能够跨不同区域在中转网关的挂载之间路由流量。

有关更多信息，请参阅中转网关。

当 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，在一个区域中的“辐射型”VPC 中部署的应用程序可以连接到另一个区域中的“辐射型”VPC 中的 ElastiCache 缓存。

访问在不同 AWS 区域的不同 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，以允许来自该应用程序实例 IP 范围的入站连接。请注意，这种情况下，无法引用该应用程序服务器安全组。

跨区域访问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。

从运行于客户数据中心中的应用程序访问 ElastiCache 缓存

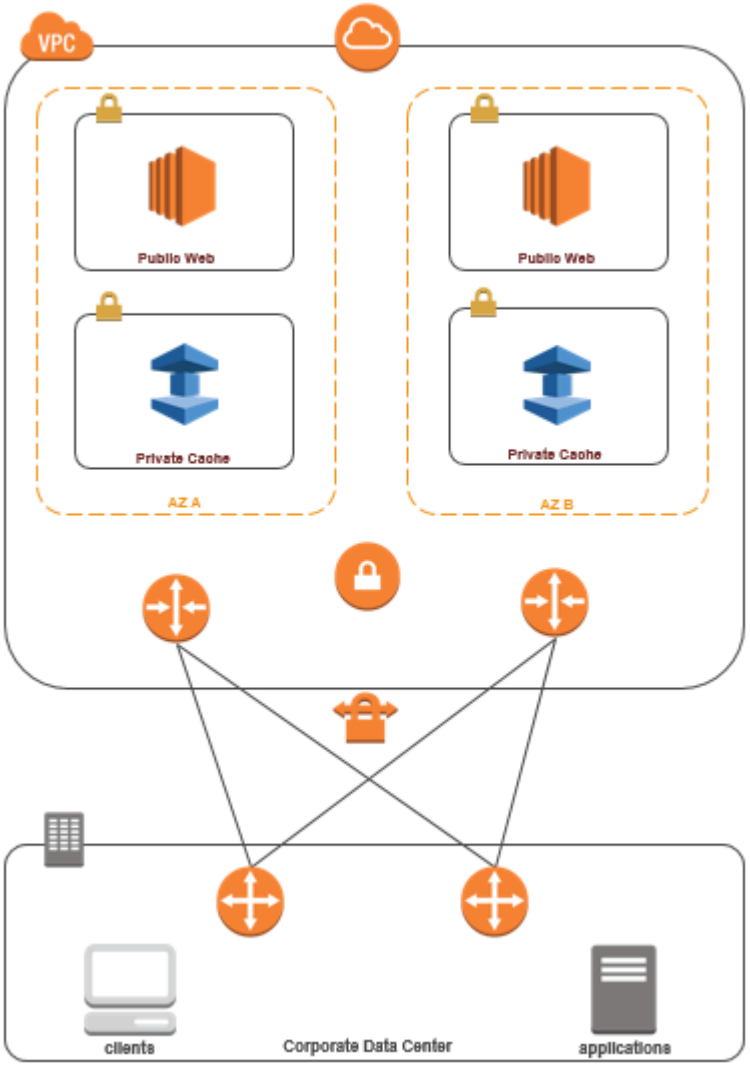
还有一种可能的场景是混合架构，客户数据中心内的客户端或应用程序可能需要访问 VPC 中的 ElastiCache 缓存。此方案也受支持，前提是客户的 VPC 和数据中心之间已通过 VPN 或 Direct Connect 建立连接。

主题

- 使用 VPN 连接从运行于客户数据中心的应用程序访问 ElastiCache 缓存
- 使用 Direct Connect 从运行于客户数据中心中的应用程序访问 ElastiCache 缓存

使用 VPN 连接从运行于客户数据中心的应用程序访问 ElastiCache 缓存

下图说明如何使用 VPN 连接，从运行于企业网络中的应用程序访问 ElastiCache 缓存。



从数据中心或通过 VPN 连接到 ElastiCache

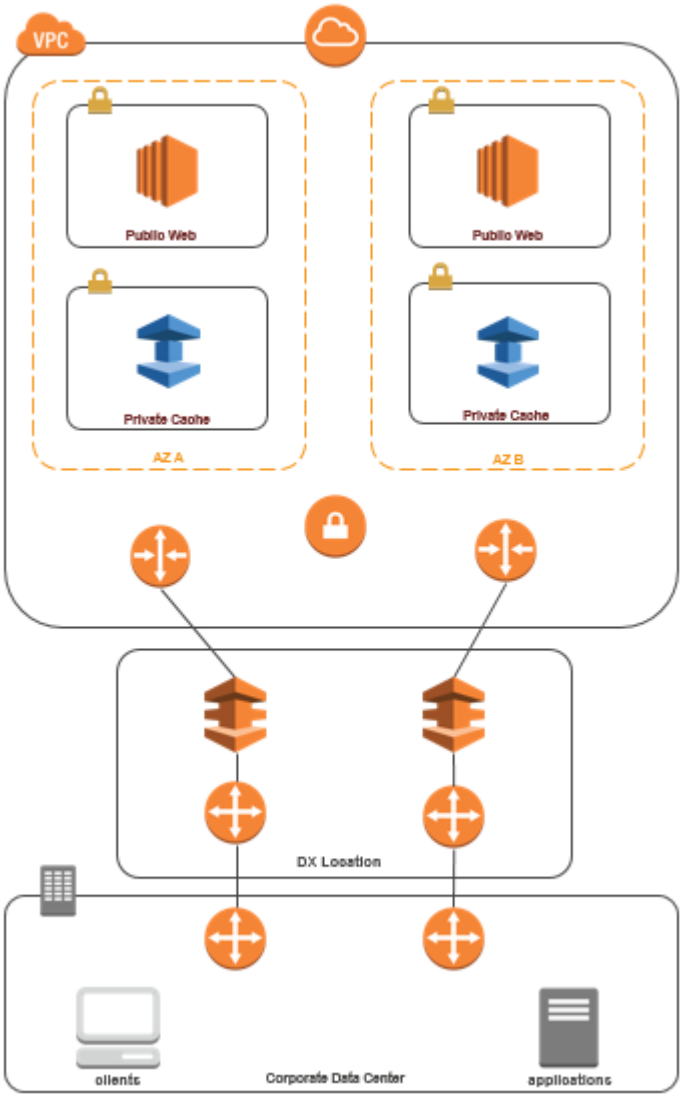
通过 VPN 连接从本地应用程序访问 VPC 中的缓存

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 缓存时所在子网的 VPC 路由表，以允许来自本地应用程序服务器的流量。对于使用 BGP 的动态路由，可自动传播您的路由。
3. 修改 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 从运行于客户数据中心中的应用程序访问 ElastiCache 缓存

下图说明如何使用 Direct Connect，从运行于企业网络中的应用程序访问 ElastiCache 缓存。



从数据中心或通过 Direct Connect 连接到 ElastiCache

使用 Direct Connect 从在您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 AWS Direct 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。

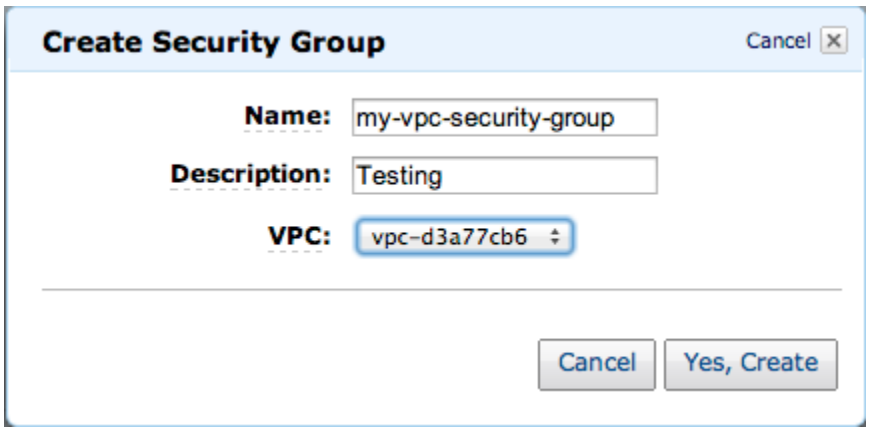
   > **Tip**
   >
   > 记录您的子网标识符，以及哪个是公有的，哪个是私有的。稍后，当您启动集群以及向 Amazon VPC 添加 Amazon EC2 实例时，您将需要此类信息。

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 Memcached 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。

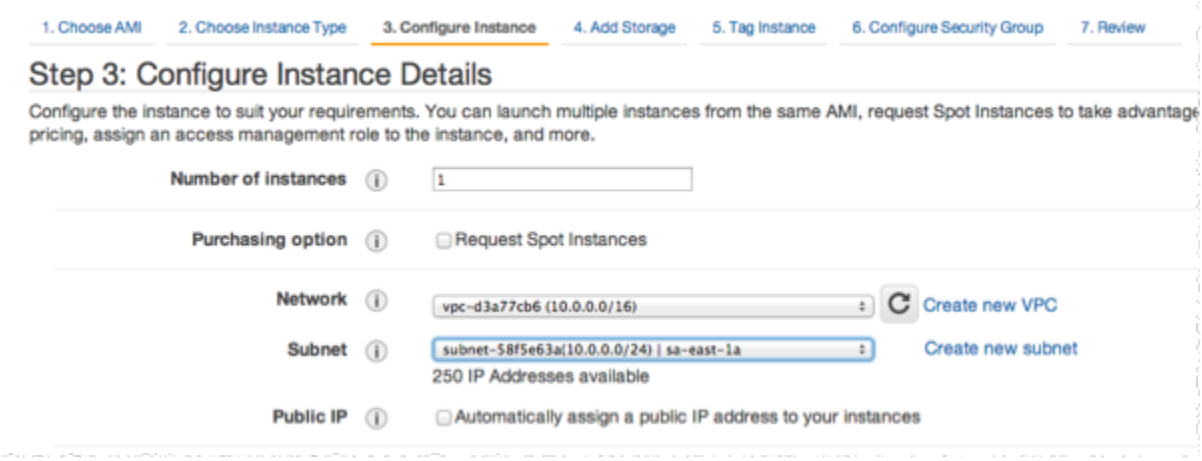
> **Note**
>
> 有关使用 Amazon EC2 的信息，请参阅 Amazon EC2 文档中的 Amazon EC2 入门指南。

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：

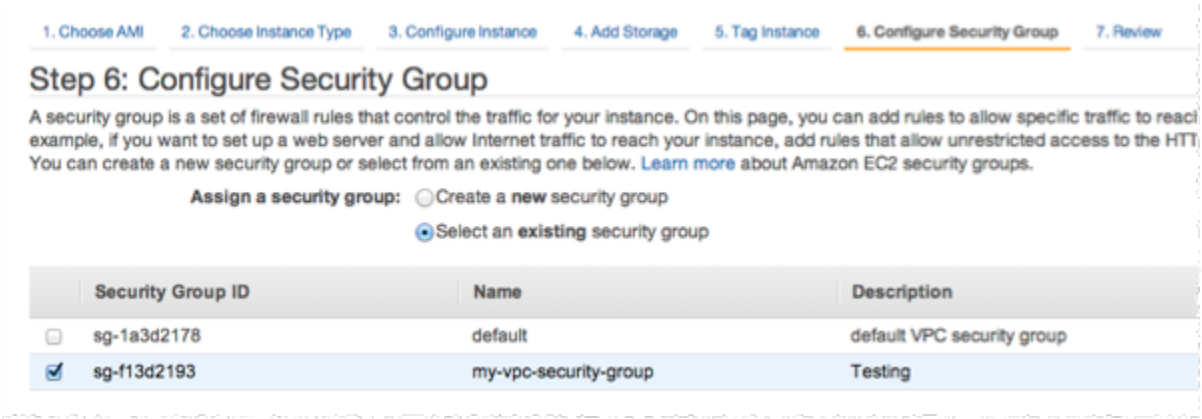
   a. 在 Network（网络）列表中，选择您的 Amazon VPC。

   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. 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 Memcached 的查找连接端点。
3. 使用 telnet 通过端口 11211 连接至您的缓存节点终端节点。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 11211
```

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：

```
add mykey 0 3600 28
This is the value for mykey
```

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：

```
VALUE mykey 0 28
This is the value for my key
END
```

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"
  ],
  "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。

主题

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。

- Memcached 集群 – 要启动 Memcached 集群，请参阅 创建 Memcached 集群（控制台）。在步骤 7.a [Advanced Memcached Settings（高级 Memcached 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 Amazon ElastiCache 的 Identity and Access Management

AWS Identity and Access Management (IAM) 是一项 AWS 服务，可以帮助管理员安全地控制对 AWS 资源的访问。IAM 管理员控制谁可以通过身份验证（登录）和获得授权（具有权限）以使用 ElastiCache 资源。IAM 是一项无需额外费用即可使用的 AWS 服务。

主题

受众

使用 AWS Identity and Access Management（IAM）的方式因您可以在 ElastiCache 中执行的操作而异。

服务用户 - 如果您使用 ElastiCache 服务来完成任务，您的管理员会为您提供所需的凭证和权限。当您使用更多 ElastiCache 功能来完成工作时，您可能需要其他权限。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 ElastiCache 中的功能，请参阅Amazon ElastiCache 身份和访问问题排查。

服务管理员 – 如果您在贵公司负责管理 ElastiCache 资源，您可能具有 ElastiCache 的完全访问权限。您有责任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。要了解有关贵公司如何将 IAM 与 ElastiCache 结合使用的更多信息，请参阅 Amazon ElastiCache 如何与 IAM 配合使用。

IAM 管理员 – 如果您是 IAM 管理员，您可能希望了解如何编写策略以管理对 ElastiCache 的访问的详细信息。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

身份验证是使用身份凭证登录 AWS 的方法。您必须作为 AWS 账户根用户、IAM 用户或通过担任 IAM 角色进行身份验证（登录到 AWS）。

您可以使用通过身份源提供的凭证以联合身份登录到 AWS。AWS IAM Identity Center(IAM Identity Center) 用户、您公司的单点登录身份验证以及您的 Google 或 Facebook 凭证都是联合身份的示例。当您以联合身份登录时，管理员以前使用 IAM 角色设置了身份联合验证。当您使用联合身份验证访问 AWS 时，您就是在间接代入角色。

根据用户类型，您可以登录 AWS Management Console 或 AWS 访问门户。有关登录到 AWS 的更多信息，请参阅《AWS 登录 用户指南》中的如何登录到您的 AWS 账户。

如果您以编程方式访问 AWS，则 AWS 将提供软件开发工具包 (SDK) 和命令行界面 (CLI)，以便使用您的凭证以加密方式签署您的请求。如果您不使用 AWS 工具，则必须自行对请求签名。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，您都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 根用户

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

联合身份

作为最佳实操，要求人类用户（包括需要管理员访问权限的用户）结合使用联合身份验证和身份提供程序，以使用临时凭证来访问 AWS 服务。

联合身份是来自企业用户目录、网络身份提供程序、AWS Directory Service、Identity Center 目录的用户，或任何使用通过身份源提供的凭证来访问 AWS 服务 的用户。当联合身份访问 AWS 账户时，他们担任角色，而角色提供临时凭证。

要集中管理访问权限，我们建议您使用 AWS IAM Identity Center。您可以在 IAM Identity Center 中创建用户和组，也可以连接并同步到自己的身份源中的一组用户和组以跨所有 AWS 账户 和应用程序使用。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 用户和组

IAM 用户是 AWS 账户 内对某个人员或应用程序具有特定权限的一个身份。在可能的情况下，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担任。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的何时创建 IAM 用户（而不是角色）。

IAM 角色

IAM 角色是 AWS 账户 中具有特定权限的身份。它类似于 IAM 用户，但与特定人员不关联。您可以通过切换角色，在 AWS Management Console 中暂时担任 IAM 角色。您可以调用 AWS CLI 或 AWS API 操作或使用自定义网址以担任角色。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 用户或角色可担任 IAM 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。角色是授予跨账户存取权限的主要方式。但是，对于某些 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座席）。要了解用于跨账户存取的角色和基于资源的策略之间的差别，请参阅《IAM 用户指南》中的 IAM 角色与基于资源的策略有何不同。
- 跨服务访问 – 某些 AWS 服务 使用其他 AWS 服务 中的特征。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 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：当您使用 IAM 用户或角色在 AWS 中执行操作时，您将被视为主体。使用某些服务时，您可能会执行一个操作，此操作然后在不同服务中启动另一个操作。FAS 使用主体调用 AWS 服务 的权限，结合请求的 AWS 服务，向下游服务发出请求。只有在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 请求时的策略详细信息，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担任的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。
  - 服务相关角色 - 服务相关角色是与 AWS 服务关联的一种服务角色。服务可以担任代表您执行操作的角色。服务相关角色显示在您的 AWS 账户中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务相关角色的权限。
- 在 Amazon EC2 上运行的应用程序 - 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 AWS CLI 或 AWS API 请求的应用程序的临时凭证。这优先于在 EC2 实例中存储访问密钥。要将 AWS 角色分配给 EC2 实例并使其对该实例的所有应用程序可用，您可以创建一个附加到实例的实例配置文件。实例配置文件包含角色，并使 EC2 实例上运行的程序能够获得临时凭证。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

要了解是使用 IAM 角色还是 IAM 用户，请参阅《IAM 用户指南》中的何时创建 IAM 角色（而不是用户）。

使用策略管理访问

您将创建策略并将其附加到 AWS 身份或资源，以控制 AWS 中的访问。策略是 AWS 中的对象；在与身份或资源相关联时，策略定义它们的权限。在主体（用户、根用户或角色会话）发出请求时，AWS 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 AWS 中存储为 JSON 文档。有关 JSON 策略文档的结构和内容的更多信息，请参阅《IAM 用户指南》中的 JSON 策略概览。

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

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 policy。然后，管理员可以向角色添加 IAM 策略，并且用户可以代入角色。

IAM policy 定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设有一个允许 `iam:GetRole` 操作的策略。具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 策略。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管式策略是可以附加到AWS 账户中的多个用户、组和角色的独立策略。托管式策略包括 AWS 托管式策略和客户管理型策略。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。主体可以包括账户、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来自 IAM 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、AWS WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的访问控制列表 (ACL) 概览。

其他策略类型

AWS 支持额外的、不太常用的策略类型。这些策略类型可以设置更常用的策略类型授予的最大权限。

- 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。您可以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` 字段中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅《IAM 用户指南》中的 IAM 实体的权限边界。
- 服务控制策略 (SCP) - SCP 是 JSON 策略，指定了组织或组织单位 (OU) 在 AWS Organizations 中的最大权限。AWS Organizations 服务可以分组和集中管理您的企业拥有的多个 AWS 账户 账户。

如果在组织内启用了所有特征，则可对任意或全部账户应用服务控制策略 (SCP)。SCP 限制成员账户中实体（包括每个 AWS 账户根用户）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 AWS 如何确定在涉及多种策略类型时是否允许请求，请参阅《IAM 用户指南》中的策略评估逻辑。

Amazon ElastiCache 如何与 IAM 配合使用

在使用 IAM 管理对 ElastiCache 的访问权限之前，您应该了解哪些 IAM 功能可与 ElastiCache 配合使用。

可与 Amazon ElastiCache 配合使用的 IAM 功能

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

要大致了解 Amazon Cognito 和其他 AWS 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中的与 IAM 一起使用的 AWS 服务。

适用于 ElastiCache 的基于身份的策略

支持基于身份的策略	是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。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。主体可以包括账户、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。当主体和资源处于不同的 AWS 账户 中时，则信任账户中的 IAM 管理员还必须授予主体实体（用户或角色）对资源的访问权限。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 IAM 角色与基于资源的策略有何不同。

适用于 ElastiCache 的策略操作

支持策略操作	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与相关联的 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 操作的列表，请参阅《服务授权参考》中的 Amazon ElastiCache 定义的操作。

ElastiCache 中的策略操作在操作前使用以下前缀：

```
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"
         ]
```

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。

ElastiCache 的策略资源

支持策略资源	是

管理员可以使用 AWS JSON 策略来指定谁有权访问什么内容。也就是说，哪个主体 可以对什么资源执行操作，以及在什么条件 下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon 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": "*"
```

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。

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内容。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 Condition 元素（或 Condition 块）中，您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 元素中指定多个密钥，则 AWS 使用逻辑 AND 运算评估它们。如果您要为单个条件键指定多个值，则 AWS 使用逻辑 OR 运算来评估条件。在授予语句的权限之前必须满足所有的条件。

您也可以在指定条件时使用占位符变量。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 策略元素：变量和标签。

AWS 支持全局条件键和特定于服务的条件键。要查看所有 AWS 全局条件键，请参阅《IAM 用户指南》中的 AWS 全局条件上下文键。

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。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon ElastiCache 定义的操作。

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 中的访问控制列表（ACL）

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。

使用 ElastiCache 的基于属性的访问权限控制（ABAC）

支持 ABAC（策略中的标签）	是

基于属性的访问权限控制 (ABAC) 是一种授权策略，该策略基于属性来定义权限。在 AWS 中，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）以及 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，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)。

将临时凭证用于 ElastiCache

支持临时凭证	是

某些 AWS 服务 在使用临时凭证登录时无法正常工作。有关更多信息，包括 AWS 服务 与临时凭证配合使用，请参阅《IAM 用户指南》中的使用 IAM 的 AWS 服务。

如果您不使用用户名和密码而用其他方法登录到 AWS Management Console，则使用临时凭证。例如，当您使用贵公司的单点登录（SSO）链接访问 AWS 时，该过程将自动创建临时凭证。当您以用户身份登录控制台，然后切换角色时，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 或者 AWS API 创建临时凭证。之后，您可以使用这些临时凭证访问 AWS。AWS 建议您动态生成临时凭证，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 AWS 中执行操作时，您将被视为主体。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 使用主体调用 AWS 服务 的权限，结合请求的 AWS 服务，向下游服务发出请求。只有在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 请求时的策略详细信息，请参阅转发访问会话。

ElastiCache 的服务角色

支持服务角色	是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 AWS 服务委派权限的角色。

> **Warning**
>
> 更改服务角色的权限可能会破坏 ElastiCache 的功能。仅当 ElastiCache 提供相关指导时才编辑服务角色。

ElastiCache 的服务相关角色

支持服务相关角色	是

服务相关角色是一种与 AWS 服务 相关的服务角色。服务可以担任代表您执行操作的角色。服务相关角色显示在您的 AWS 账户中，并由该服务拥有。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 的服务。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": "*"
        }
    ]
}
```

Amazon ElastiCache 身份和访问问题排查

使用以下信息可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 ElastiCache 中执行操作

如果 AWS Management Console 告诉您，您无权执行某个操作，则必须联系您的管理员寻求帮助。您的管理员是指为您提供用户名和密码的那个人。

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

如果收到表明您无权执行 `iam:PassRole` 操作的错误，则必须更新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到该服务，而不是创建新服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 ElastiCache 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 AWS 管理员。管理员是向您提供登录凭证的人。

我想要允许我的 AWS 账户之外的用户访问我的 ElastiCache 资源

您可以创建一个角色，以便其它账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以担任角色。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 ElastiCache 是否支持这些功能，请参阅Amazon ElastiCache 如何与 IAM 配合使用。
- 要了解如何为您拥有的 AWS 账户 中的资源提供访问权限，请参阅 IAM 用户指南中的为您拥有的另一个 AWS 账户 中的 IAM 用户提供访问权限。
- 要了解如何为第三方AWS 账户提供您的资源的访问权限，请参阅 IAM 用户指南中的为第三方拥有的 AWS 账户提供访问权限。
- 要了解如何通过联合身份验证提供访问权限，请参阅 IAM 用户指南中的为经过外部身份验证的用户（联合身份验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅 IAM 用户指南中的 IAM 角色与基于资源的策略有何不同。

访问控制

您可能具有有效的凭证来对您的请求进行身份验证，但除非您拥有权限，否则您无法创建或访问 ElastiCache 资源。例如，您必须拥有创建 ElastiCache 集群的权限。

下面几节介绍如何管理 ElastiCache 的权限。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:DescribeCacheClusters",
            "elasticache:ModifyServerlessCache",
            "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
- 示例 2：允许用户执行常见的 ElastiCache 系统管理员任务
- 示例 3：允许用户访问所有 ElastiCache API 操作
- 示例 4：允许用户调用 IAM CreateServiceLinkedRole API

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"
        }
      }
    }
  ]
}
```

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"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
```

```
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "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"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
```

```
                    "elasticache:CacheNodeType": [
                        "cache.t2.micro",
                        "cache.t2.medium"
                    ]
                }
            }
        }
    ]
}
```

4. elasticache:EngineVersion：指定引擎版本 1.6.6 的使用情况

```
{
    "Version": "2012-10-17",
    "Statement": [
     {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:EngineVersion": "1.6.6"
                }
            }
        }
    ]
}
```

5. elasticache: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"
            }
        }
    }
  ]
}
```

6. elasticache: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"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:parametergroup:*",
                "arn:aws:elasticache:*:*:subnetgroup:*"
            ]
        },

        {
            "Effect": "Allow",
            "Action": [
                "elasticache:CreateCacheCluster"
            ],
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
                    "elasticache:CacheParameterGroupName": "my-org-param-group"
                }
            }
        }
    ]
}
```

7. 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"
                    ]
                }
            }
        }
    ]
}
```

8. elasticache: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"
      ],
      "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），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

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。此表列出每个 Amazon ElastiCache API 操作及您可授予执行该操作的权限的对应操作。您可以在策略的 `Action` 字段中指定这些操作，并在策略的 `Resource` 字段中指定资源值。除非另有说明，否则需要该资源。某些字段同时包含必需资源和可选资源。如果没有资源 ARN，则策略中的资源为通配符 (*)。

您可以在 ElastiCache 策略中使用条件键来表达条件。要查看特定于 ElastiCache 的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 范围的键的完整列表，请参阅《IAM 用户指南》中的 AWS 全局条件上下文键。

> **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 操作的列表，请参阅《服务授权参考》中的 Amazon ElastiCache 定义的操作。

Amazon ElastiCache 的合规性验证

作为多个AWS合规性计划的一部分，第三方审核员将评估AWS服务的安全性与合规性，例如SOC、PCI、FedRAMP 和 HIPAA。

要了解某个 AWS 服务 是否在特定合规性计划范围内，请参阅合规性计划范围内的 AWS 服务 ，然后选择您感兴趣的合规性计划。有关常规信息，请参阅 AWS 合规性计划。

您可以使用 AWS Artifact 下载第三方审计报告。有关更多信息，请参阅在 AWS Artifact 中下载报告、。

您使用 AWS 服务 的合规性责任取决于数据的敏感性、贵公司的合规性目标以及适用的法律法规。AWS 提供以下资源来帮助满足合规性：

- 安全性与合规性快速入门指南 - 这些部署指南讨论了架构注意事项，并提供了在 AWS 上部署以安全性和合规性为重点的基准环境的步骤。
- Amazon Web Services 上的 HIPAA 安全性和合规性架构设计 - 该白皮书介绍了公司如何使用 AWS 创建符合 HIPAA 标准的应用程序。

> **Note**
>
> 并非所有 AWS 服务 都符合 HIPAA 要求。有关更多信息，请参阅符合 HIPAA 要求的服务参考。

- AWS 合规性资源 - 此业务手册和指南集合可能适用于您的行业和位置。
- 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 AWS 服务 的最佳实践，并将指南映射到跨多个框架的安全控制，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）。
- AWS Config 开发人员指南中的使用规则评估资源 - 此 AWS Config 服务评测您的资源配置对内部实践、行业指南和法规的遵循情况。
- AWS Security Hub - 此 AWS 服务 向您提供 AWS 中安全状态的全面视图。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操。有关受支持服务及控制的列表，请参阅 Security Hub 控制参考。
- AWS Audit Manager - 此 AWS 服务 可帮助您持续审计您的 AWS 使用情况，以简化管理风险以及与相关法规和行业标准的合规性的方式。

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 合规性计划

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 Memcached 时发生的故障

运行 Memcached 引擎时，您有以下选择来最大程度地减小故障的影响。故障缓解计划中有两类需要解决的故障：节点故障和可用区故障。

缓解节点故障

无服务器缓存使用复制的多可用区架构，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。在自行设计的集群中，要缓解节点故障的影响，请将缓存数据分布到多个节点。由于自行设计的集群不支持复制，因此节点故障始终会导致集群中的一些数据丢失。

在创建 Memcached 集群时，您可以创建包含 1 到 60 个节点的集群，也可以根据特殊要求创建更多节点。跨更多节点对数据进行分区意味着，在节点出现故障时丢失的数据更少。例如，如果您跨 10 个节点对数据进行分区，则任一节点均可存储您的约 10% 的缓存数据。在此情况下，在创建和预配置替换节点时，节点故障会导致丢失约 10% 的需替换的缓存数据。如果在 3 个较大的节点中对相同的数据进行缓存，则节点故障将导致丢失约 33% 的缓存数据。

如果您在 Memcached 集群中需要超过 60 个节点，或者一个 AWS 区域中总共需要超过 300 个节点，请填写 https://aws.amazon.com/contact-us/elasticache-node-limit-request/上 ElastiCache 限申请表。

有关指定 Memcached 集群中的节点数的信息，请参阅创建 Memcached 集群（控制台）。

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，请将节点置于尽可能多的可用区中。在出现极少发生的可用区故障时，您将丢失已在该可用区中缓存的数据，而不会丢失在其他可用区中缓存的数据。

为何要使用如此多的节点？

如果我的区域只有 3 个可用区，既然在可用区出现故障时，我会丢失约三分之一的数据，那么为何我需要 3 个以上的节点？

这个问题问得很好。请记住，我们正在尝试缓解两种不同类型的故障，即节点故障和可用区故障。您说得对，如果您的数据跨可用区分布且其中一个区域发生故障，则无论您拥有多少个节点，都只会丢失该可用区中缓存的数据。但是，如果节点出现故障，则拥有更多节点将减少丢失的数据的比例。

没有用于确定集群中拥有的节点数的“神奇公式”。您必须权衡数据丢失的影响、发生故障的可能性与成本，并得出您自己的结论。

有关指定 Memcached 集群中的节点数的信息，请参阅创建 Memcached 集群（控制台）。

有关区域和可用区的更多信息，请参阅区域和可用区。

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

如果您正在运行 Memcached 并且正在跨节点对数据进行分区，在任一节点出现故障的情况下，您使用的节点越多，丢失的数据就越少。

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。

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
- 使用 AWS 控制台验证是否应用了最新的服务更新

- 停止服务更新

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
```

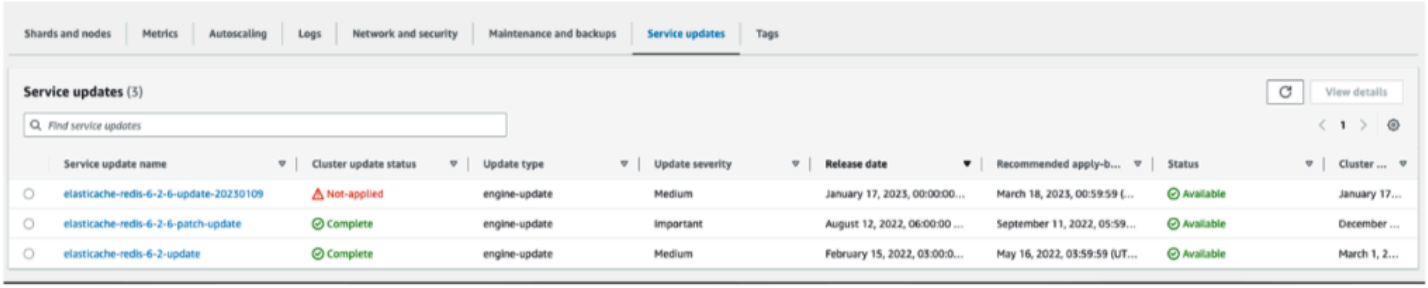
有关更多信息，请参阅batch-apply-update-action。

使用 AWS 控制台验证是否应用了最新的服务更新

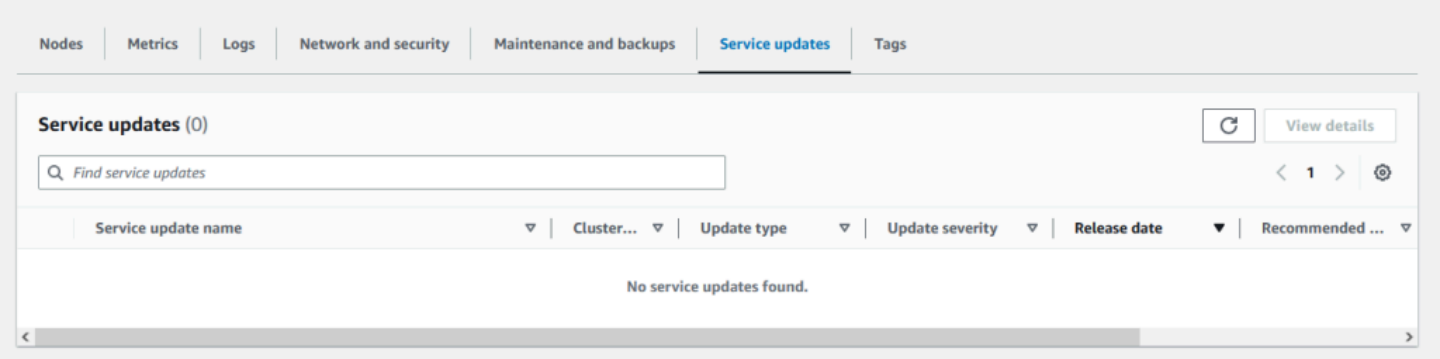
您可以按照以下步骤验证您 ElastiCache 的 For Redis 集群是否正在运行最新的服务更新：

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。

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

- 使用 AWS CloudTrail 记录 Amazon ElastiCache API 调用

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 Memcached 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

指标	描述	单位
SuccessfulReadRequestLatency	成功读取请求的延迟。	微秒
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
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
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
Reclaimed	由缓存使其失效的键数量	计数

命令级指标

ElastiCache 还会发出以下 Memcached 命令级别的指标

指标	描述	单位
CmdGet	缓存已收到的 get 命令数。	计数

指标	描述	单位
CmdSet	缓存已收到的 set 命令数。	计数
CmdTouch	缓存已收到的 touch 命令数。	计数
GetHits	找到了所请求密钥的情况下，缓存收到的 get 请求数。	计数
GetMisses	未找到所请求密钥的情况下，缓存收到的 get 请求数。	计数
IncrHits	找到了所请求密钥的情况下，缓存收到的增量请求数。	计数
IncrMisses	未找到所请求密钥的情况下，缓存收到的增量请求数。	计数
DecrHits	找到了所请求密钥的情况下，缓存已收到的减量请求数。	计数
DecrMisses	未找到所请求密钥的情况下，缓存已收到的减量请求数。	计数
DeleteHits	找到了所请求密钥的情况下，缓存已收到的删除请求数。	计数
DeleteMisses	未找到所请求密钥的情况下，缓存已收到的删除请求数。	计数
TouchHits	已被触动并赋予新的过期时间的密钥数。	计数
TouchMisses	已接触但未找到的键数量。	计数
CasHits	找到了请求的键并且 cas 值匹配的情况下，缓存已收到的 cas 请求数。	计数

指标	描述	单位
CasMisses	未找到所请求键的情况下，缓存已收到的 cas 请求数。	计数
CasBadval	cas 值与已存储 cas 值不匹配的情况下，缓存已收到的 cas 请求数。	计数
CmdFlush	缓存已收到的 flush 命令数。	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

Detail-Type	描述	单位	来源	消息
				败。您无权创建 VPC 端点。

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 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。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

Detail-Type	资源列表	类别	来源	消息
				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。
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。

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
- 自行设计集群的事件
- 使用 CloudWatch 指标监控使用情况
- Amazon SNS 监控 ElastiCache 事件

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 Memcached 的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 Memcached 指标的更多信息，请参阅 Memcached 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。
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:CacheClusterProvisioning	尝试将新缓存群集启动到不存在的 Virtual Private Cloud (VPC) 中。

事件名称	消息	描述
	`Failed : cluster-name`	
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 Memcached 的缓存节点。

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 Memcached 的缓存节点。
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`。

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。

另请参阅

- Memcached 的指标

指标	描述	单位
CPUUtilization	整个主机的 CPU 使用率百分比。	百分比
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	积分（vCPU 分钟）

指标	描述	单位
	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。	
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。	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
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

指标	描述	单位
NetworkBandwidthIn AllowanceExceeded	因入站聚合带宽超过实例的最大值而形成的数据包的数量。	计数
NetworkConntrackAllowanceExceeded	由于连接跟踪超过实例的最大值且无法建立新连接而形成的数据包的数量。这可能会导致进出实例的流量丢失数据包。	计数
NetworkBandwidthOutAllowanceExceeded	因出站聚合带宽超过实例的最大值而形成的数据包的数量。	计数
Network Packets Per Second Allowance Exceeded	因双向每秒数据包数量超过实例的最大值而形成的数据包数量。	计数
NetworkMaxBytesIn	每分钟内接收的最大突增字节数。	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

Memcached 的指标

AWS/ElastiCache 命名空间包括以下 Memcached 指标。

AWS/ElastiCache 命名空间包括以下源自 Memcached stats 命令的指标。每项指标均是按照缓存节点级计算的。

另请参阅

- 主机级指标

指标	描述	单位
BytesReadIntoMemcached	缓存节点已从网络读取的字节数。	字节
BytesUsedForCacheItems	用来存储缓存项的字节数。	字节
BytesWrittenOutFromMemcached	缓存节点已写入网络的字节数。	字节
CasBadval	Cas（检查和设置）值不匹配已存储 Cas 值的情况下，缓存已收到的 Cas 请求数。	计数
CasHits	找到了请求的密钥并且 Cas 值匹配的情况下，缓存已收到的 Cas 请求数。	计数
CasMisses	未找到所请求密钥的情况下，缓存已收到的 Cas 请求数。	计数
CmdFlush	缓存已收到的 flush 命令数。	计数
CmdGet	缓存已收到的 get 命令数。	计数
CmdSet	缓存已收到的 set 命令数。	计数
CurrConnections	在某个瞬间连接到缓存的连接计数。ElastiCache 使用两到三个连接来监控集群。 除了上述内容之外，Memcached 还创建了一些相当于用于节点类型所用线程两倍的内部连接。各种节点类型的线程计数可以在适用参数组的 Nodetype Specific Parameters 中查看。 总连接是客户端连接、用于监控的连接和上述内部连接的总和。	计数
CurrItems	当前存储在缓存中的项目的计数。	计数

指标	描述	单位
DecrHits	找到了所请求密钥的情况下，缓存已收到的减量请求数。	计数
DecrMisses	未找到所请求密钥的情况下，缓存已收到的减量请求数。	计数
DeleteHits	找到了所请求密钥的情况下，缓存已收到的删除请求数。	计数
DeleteMisses	未找到所请求密钥的情况下，缓存已收到的删除请求数。	计数
Evictions	缓存为给新的写入留出空间而移除的未过期项目数。	计数
GetHits	找到了所请求密钥的情况下，缓存收到的 get 请求数。	计数
GetMisses	未找到所请求密钥的情况下，缓存收到的 get 请求数。	计数
IncrHits	找到了所请求密钥的情况下，缓存收到的增量请求数。	计数
IncrMisses	未找到所请求密钥的情况下，缓存收到的增量请求数。	计数
Reclaimed	缓存为给新的写入留出空间而移除的过期项目数。	计数

对于 Memcached 1.4.14，还提供下列额外指标。

指标	描述	单位
BytesUsedForHash	散列表当前使用的字节数。	字节
CmdConfigGet	config get 请求的累计数量。	计数

指标	描述	单位
CmdConfigSet	config set 请求的累计数量。	计数
CmdTouch	touch 请求的累计数量。	计数
CurrConfig	当前存储的配置数。	计数
EvictedUnfetched	从最近最少使用的缓存 (LRU) 移出、设置后从未被触动的有效项目数。	计数
ExpiredUnfetched	从 LRU 移出、设置后从未被触动的过期项目数。	计数
SlabsMoved	已移动的 Slab 页面总数。	计数
TouchHits	已被触动并赋予新的过期时间的密钥数。	计数
TouchMisses	已被触动但未找到的项目数。	计数

AWS/ElastiCache 命名空间包括以下计算出的缓存级别指标。

指标	描述	单位
NewConnections	缓存已收到的新连接数。这一数字通过记录一段时间内 total_connections 的变化，从 Memcached total_connections 统计数据得出。由于连接是为 a 保留的，因此该值将始终至少为 1 ElastiCache。	计数
NewItems	缓存存储的新项目数。这一数字通过记录一段时间内 total_items 的变化，从 Memcached total_items 统计数据得出。	计数
UnusedMemory	数据未使用的内存的数量。这一数量通过从 limit_maxbytes 扣减字节数，从 Memcached 统计数据 limit_maxbytes 和字节数得出。	字节

指标	描述	单位
	由于 Memcached 开销除了数据使用的内存之外还使用内存，因此 UnusedMemory 不应将其视为可用于存储额外数据的内存量。即使您仍然拥有一些未使用的内存，也可能会出现移出情况。 有关更多详细信息，请参阅 Memcached 项目内存使用。	

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
- SwapUsage
- 移出
- 当前连接

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。

因为 Memcached 是多线程的，所以此指标可能高达 90%。如果超过此阈值，请使用更大的缓存节点类型扩展缓存集群，或通过添加更多缓存节点进行扩展。

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，我们建议您增大 ConnectionOverhead 参数值。

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。

如果超过所选阈值，请使用更大的节点类型纵向扩展集群，或通过添加更多节点来横向扩展。

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。

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。

> **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 Memcached 的集群的列表，请在导航窗格中选择 Memcached。
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 Memcached 的缓存节点。

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 Memcached 的缓存节点。
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
每个区域的节点数	300
每个群集的节点	40
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

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 Memcached 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：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，请参阅扩展 Mem ElastiCache cached。

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 Memcached 通知

以下 ElastiCache 通知特定于 Memcached 引擎。

ElastiCache for Memcached 特定通知

- 提示：Memcached LRU 爬网程序导致分段故障

提示：Memcached LRU 爬网程序导致分段故障

⚠ 警报日期：2017 年 2 月 28 日
在某些情况下，您的集群可能会显示不稳定情况，出现 Memcached LRU 爬网程序中的分段故障。这是 Memcached 引擎中存在已有一段时间的问题。在 Memcached 1.4.33 中默认情况下启用了 LRU 爬网程序时，此问题变得明显。
如果您遇到此问题，我们建议您禁用 LRU 爬网程序，直至有修复程序可用。为此，请在命令行上使用 `lru_crawler disable` 或者修改 `lru_crawler` 参数值 (首选)。
解决日期：
解析:

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 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。

Mem ElastiCache cached 最新更新

变更	说明	日期
对 Memcach ElastiCache ed 1.6.22 的支持	ElastiCache 适用于 Memcached 现在支持 Memcached 1.6.22。它包括从版本 1.6.18 开始的累积错误修复。有关更多信息，请参阅 Memcached 版本 1.6.22。	2024 年 1 月 10 日
ElastiCache 适用于 Memcached 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

对 Memcach ElastiCache ed 1.6.17 的支持	ElastiCache 适用于 Memcached 现在支持 Memcached 1.6.17。它包括对版本 Memcached 1.6.12 的累积错误修复。有关更多信息，请参阅 Memcached 版本 1.6.17。	2023 年 1 月 18 日
ElastiCache memcached 版现在支持 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 Memcached 引擎版本 1.6.6 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 Memcached 现在支持传输中加密	传输中加密是一项可选功能，它允许您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。此功能在 Memcached 版本 1.6.12 及更高版本上受支持。有关更多信息，请参阅ElastiCache 传输中加密 (TLS)。	2022 年 5 月 26 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
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
对 Memcached 1.6.6 的支持。	ElastiCache memcached 版现在支持 Memcached 1.6.6。它包括 Memcached 1.5.16 版本中的累积错误修复。有关更多信息，请参阅 Memcached 版本 1.6.6。	2020 年 11 月 12 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 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 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 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 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

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
ElastiCache for Memcached 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。这将最大程度地减少服务中断，特别是在高峰业务流程期间，并有助于确保在安全更新时保持合规性。有关更多信息，请参阅 Amazon 中的自助服务更新 ElastiCache。	2019 年 10 月 9 日
对 Memcach ElastiCache ed 1.5.16 的支持	ElastiCache 适用于 Memcached 现在支持 Memcached 1.5.16。它包括 Memcached 1.5.14 和 Memcached 1.5.15 版本中的累积错误修复。有关更多信息，请参阅 Memcached 版本 1.5.16。	2019 年 9 月 6 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

对 Memcach ElastiCache ed 1.5.10 的支持	ElastiCache 适用于 Memcached 现在支持 Memcached 1.5.10。它包括对 no_modern 和 inline_ascii_resp 参数的支持。有关更多信息，请参阅 Memcached 版本 1.5.10。	2018 年 11 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

下表描述了 2018 年 3 月之前对《ElastiCache 适用于 Redis 的 Memcached 用户指南的重要更改。

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：	2017 年 12 月 18 日

更改	描述	更改日期
	• 支持的区域 • 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 Memcached 节点类型的参数	2017 年 11 月 20 日
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 用于访问 Amazon VPC 中的 ElastiCache 缓存的访问模式。	2017 年 4 月 24 日

更改	描述	更改日期
对 Memcached 1.4.34 的支持	ElastiCache 添加了对 Memcached 1.4.34 版本的支持，该版本包含了对早期 Memcached 版本的许多修复。 有关更多信息，请参阅 Memcached 上的 Memcached 1.4.34 版本说明。GitHub	2017 年 4 月 10 日
对 Memcached 1.4.33 的支持	ElastiCache 增加了对 Memcached 版本 1.4.33 的支持。有关更多信息，请参阅下列内容： • Memcached 版本 1.4.33 • Memcached 1.4.33 增加的参数	2016 年 12 月 20 日
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

更改	描述	更改日期
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
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 Memcached 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 Memcached 节点类型的参数	2016 年 6 月 27 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
对 Memcached 1.4.28 的支持。	ElastiCache 增加了对 Memcached 版本 1.4.24 和 Memcached 自 1.4.14 版本以来的改进的支持。此版本增加了对以下项的支持：将最近最少使用的（LRU）缓存管理作为后台任务、选择 jenkins 或 murmur3 作为哈希算法、新命令以及多项错误修复。有关更多信息，请参阅 Memcached 发布说明 .	2015 年 8 月 27 日
对使用 PHP 5.6 的 Memcached Auto Discovery 的支持	此版本的 Amazon ElastiCache 增加了对 PHP 版本 5.6 的 Memcached 自动发现客户端的支持。有关更多信息，请参阅编译适用于 PHP 的 ElastiCache Cluster Client 的源代码。	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
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
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
Memcached 支持的灵活节点放置	ElastiCache 增加了对跨多个可用区创建 Memcached 节点的支持。	2014 年 7 月 23 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
PHP Auto Discovery	增加了对 PHP 版本 5.5 Auto Discovery 的支持。	2014 年 5 月 13 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
缓存节点 Auto Discovery 的 PHP 支持	初次发布的缓存节点 Auto Discovery 为 Java 程序提供支持。在此版本中，为 PHP ElastiCache 引入了缓存节点自动发现支持。	2013 年 1 月 2 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

更改	描述	更改日期
缓存节点 Auto Discovery 和新缓存引擎版本	ElastiCache 提供缓存节点 auto discovery，即客户端程序能够自动确定集群中的所有缓存节点，并启动和维护与所有这些节点的连接。 此版本还提供新的缓存引擎版本：Memcached 版本 1.4.14。这个新缓存引擎提供增强的 Slab 重新平衡功能、显著的性能和可扩展性改进以及多个缺陷修复。有几个可以配置的新缓存参数。有关更多信息，请参阅使用参数组配置引擎参数。	2012 年 11 月 28 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。