ElastiCache for Memcached User Guide

Amazon ElastiCache

API Version 2015-02-02

Amazon ElastiCache: ElastiCache for Memcached User Guide

Table of Contents

What is Amazon ElastiCache for Memcached?

Welcome to the *Amazon ElastiCache for Memcached User Guide*. Amazon ElastiCache is a web service that makes it easy to set up, manage, and scale a distributed in-memory data store or cache environment in the cloud. It provides a high-performance, scalable, and cost-effective caching solution. At the same time, it helps remove the complexity associated with deploying and managing a distributed cache environment.

You can operate Amazon ElastiCache in two formats. You can get started with a serverless cache or choose to design your own cache cluster.

> **Note**
>
> Amazon ElastiCache works with both the Redis and Memcached engines. Use the guide for the engine that you're interested in. If you're unsure which engine you want to use, see Comparing Memcached and Redis self-designed caches in this guide.

Serverless caching

ElastiCache for Memcached offers serverless caching, which simplifies adding and operating a Memcached based cache for your application. ElastiCache for Memcached Serverless enables you to create a highly available cache in under a minute, and eliminates the need to provision instances or configure nodes or clusters. Developers can create a Serverless cache by specifying the cache name using the ElastiCache console, SDK or CLI.

ElastiCache Serverless also removes the need to plan and manage caching capacity. ElastiCache constantly monitors the cache's memory and compute used by your application, and automatically scales capacity to meet the needs of your application. ElastiCache offers a simple endpoint experience for developers, by abstracting the underlying cache infrastructure and software. ElastiCache manages hardware provisioning, monitoring, node replacements, and software patching automatically and transparently, so that you can focus application development, rather than operating the cache.

ElastiCache for Memcached Serverless is compatible with Memcached 1.6 and above.

Designing your own ElastiCache for Memcached cluster

If you need fine-grained control over your ElastiCache for Memcached cluster, you can choose to design your own Memcached cluster with ElastiCache. ElastiCache enables you to operate a node-based cluster, by choosing the node-type, number of nodes, and node placement across AWS Availability Zones for your cluster. Since, ElastiCache is a fully-managed service, it automatically manages hardware provisioning, monitoring, node replacements, and software patching for your cluster.

Designing your own ElastiCache for Memcached cluster offers greater flexibility and control over your clusters. For example, you can choose to operate a cluster with single-AZ availability or cross-AZ availability depending on your needs. When designing your own clusters, you are responsible for choosing the type and number of nodes correctly to ensure that your cache has enough capacity as required by your application. You can also choose when to apply new software patches to your Memcached cluster.

When designing your own ElastiCache for Memcached, you can choose to run Memcached 1.4 and above.

How it works

Here you can find an overview of the major components of an ElastiCache for Memcached deployment.

Cache and caching engines

A cache is an in-memory data store that you can use to store cached data. Typically, your application will cache frequently accessed data in a cache to optimize response times. ElastiCache for Memcached offers two deployment options: Serverless and self-designed clusters. See Choosing between deployment options

> **Note**
>
> Amazon ElastiCache works with both the Redis and Memcached engines. Use the guide for the engine that you're interested in. If you're unsure which engine you want to use, see Comparing Memcached and Redis self-designed caches in this guide.

Topics

- How ElastiCache for Memcached works
- Pricing dimensions

How ElastiCache for Memcached works

ElastiCache for Memcached Serverless

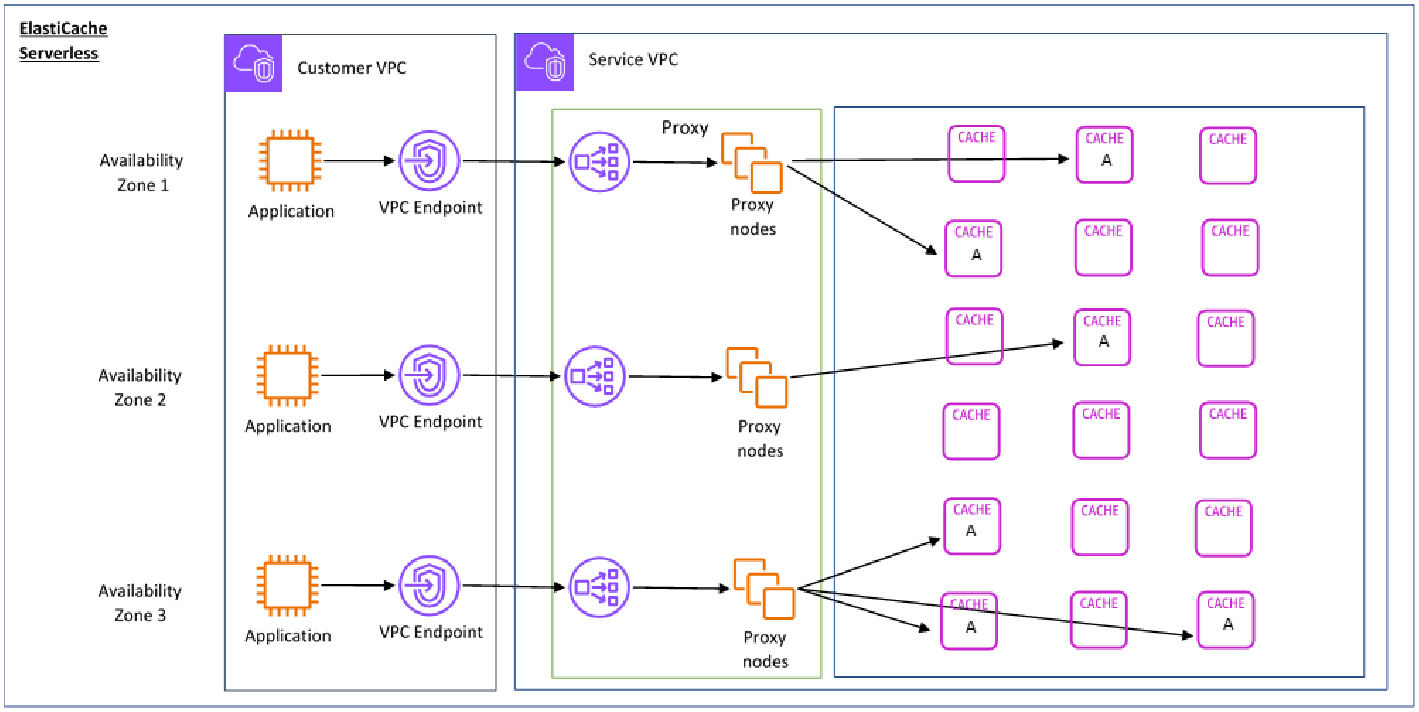
ElastiCache for Memcached Serverless enables you to create a cache without worrying about capacity planning, hardware management, or cluster design. You simply provide a name for your cache and you receive a single endpoint that you can configure in your Memcached client to begin accessing your cache.

> **Note**
>
> ElastiCache for Memcached Serverless is only compatible with Memcached clients that support TLS.

Key Benefits

- **No capacity planning:** ElastiCache Serverless removes the need for you to plan for capacity. ElastiCache Serverless continuously monitors the memory, compute, and network bandwidth utilization of your cache and scales both vertically and horizontally. It allows a cache node to grow in size, while in parallel initiating a scale-out operation to ensure that the cache can scale to meet your application requirements at all times.
- **Pay-per-use:** With ElastiCache Serverless, you pay for the data stored and compute utilized by your workload on the cache. See Pricing dimensions.
- **High-availability:** ElastiCache Serverless automatically replicates your data across multiple Availability Zones (AZ) for high-availability. It automatically monitors the underlying cache nodes and replaces them in case of failures. It offers a 99.99% availability SLA for every cache.
- **Automatic software upgrades:** ElastiCache Serverless automatically upgrades your cache to the latest minor and patch software version without any availability impact to your application. When a new Memcached major version is available, ElastiCache will send you a notification.
- **Security:** Serverless always encrypts data in transit and at rest. You can use a service managed key or use your own Customer Managed Key to encrypt data at rest.

The following diagram illustrates how ElastiCache Serverless works.



When you create a new serverless cache, ElastiCache creates a Virtual Private Cloud (VPC) Endpoint in the subnets of your choice in your VPC. Your application can connect to the cache through these VPC Endpoints.

With ElastiCache Serverless you receive a single DNS endpoint that your application connects to. When you request a new connection to the endpoint, ElastiCache Serverless handles all cache connections through a proxy layer. The proxy layer helps reduce complex client configuration, because the client does not need to rediscover cluster topology in case of changes to the underlying cluster. The proxy layer is a set of proxy nodes that handle connections using a network load balancer. When your application creates a new cache connection, the request is sent to a proxy node by the network load balancer. When your application executes cache commands, the proxy node that is connected to your application executes the requests on a cache node in your cache. The proxy layer abstracts the cache cluster topology and nodes from your client. This enables ElastiCache to intelligently load balance, scale out and add new cache nodes, replace cache nodes when they fail, and update software on the cache nodes, all without availability impact to your application or having to reset connections.

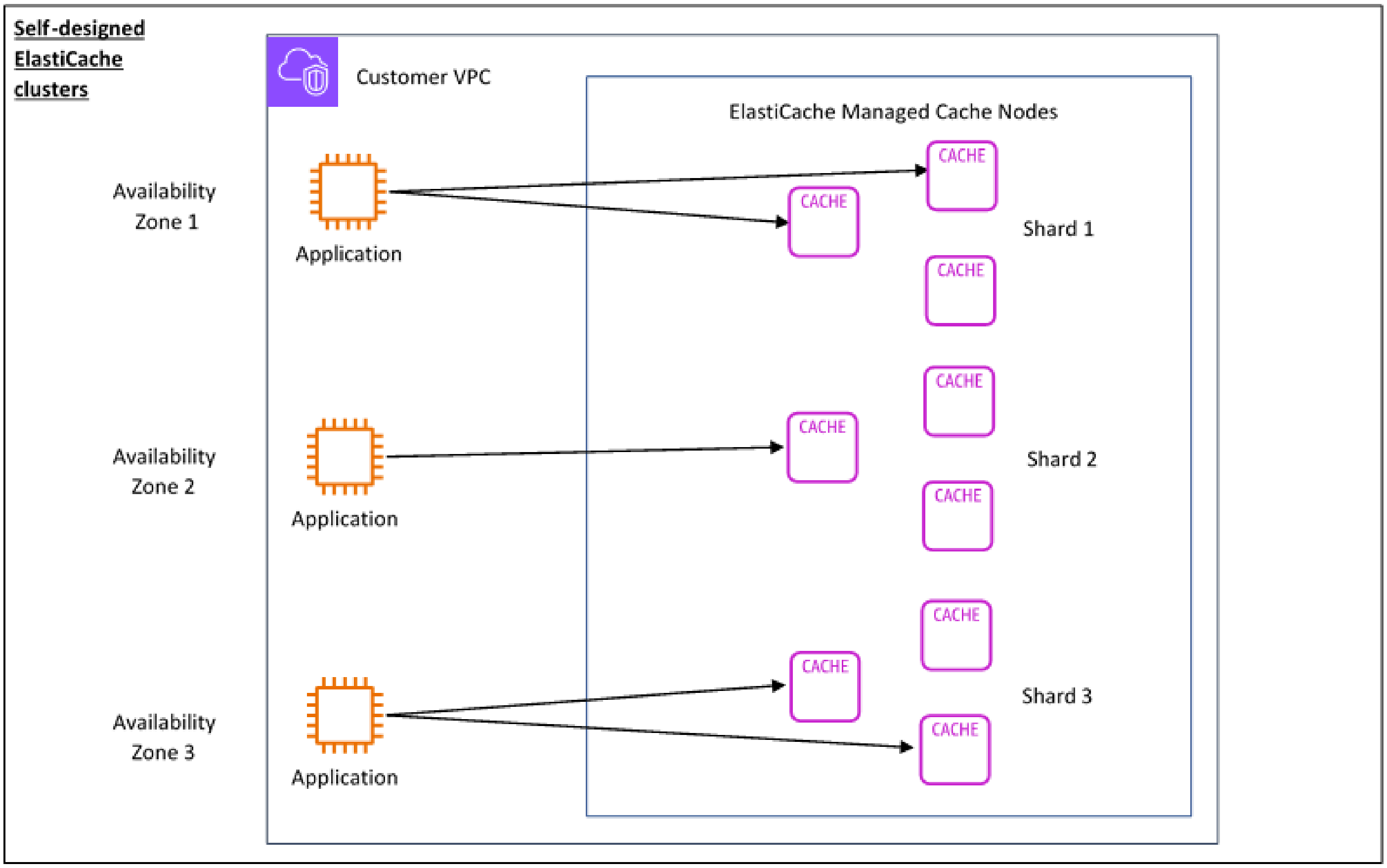
Self-designed ElastiCache clusters

You can choose to design your own ElastiCache clusters by choosing an cache node family, size, and number of nodes for your cluster. Designing your own cluster gives you finer grained control over the configuration and scaling of the cluster.

Key Benefits

- **Design your own cluster:** With ElastiCache, you can design your own cluster and choose where you want to place your cache nodes. For example, if you have an application that wants to trade-off high-availability for low latency, you can choose to deploy your cache nodes in a single AZ. Alternatively, you can design your cluster with nodes across multiple AZs to achieve high-availability.
- **Fine-grained control:** When designing our own cluster, you have more control over fine-tuning the settings on your cache. For example, you can use Configuring engine parameters using parameter groups to configure the cache engine.
- **Scale vertically and horizontally:** You can choose to manually scale your cluster by increasing or decreasing the cache node size when needed. You can also scale horizontally by adding nodes.

The following diagram illustrates how ElastiCache self-designed clusters work.



Pricing dimensions

You can deploy ElastiCache in two deployment options. When deploying ElastiCache Serverless, you pay for usage for data stored in GB-hours and compute in ElastiCache Processing Units (ECPU). When choosing to design your own ElastiCache for Memcached clusters, you pay per hour of the cache node usage. See pricing details here.

Data storage

You pay for data stored in ElastiCache Serverless billed in gigabyte-hours (GB-hrs). ElastiCache Serverless continuously monitors the data stored in your cache, sampling multiple times per minute, and calculates an hourly average to determine the cache's data storage usage in GB-hrs. Each ElastiCache Serverless cache is metered for a minimum of 1 GB of data stored.

ElastiCache Processing Units (ECPUs)

You pay for the requests your application executes on ElastiCache Serverless in ElastiCache Processing Units (ECPUs), a unit that includes both vCPU time and data transferred.

- Simple reads and writes require 1 ECPU for each kilobyte (KB) of data transferred. For example, a GET command that transfers up to 1 KB of data consumes 1 ECPU. A SET request that transfers 3.2 KB of data will consume 3.2 ECPUs.
- Commands that operate on multiple items will consume proportionally more ECPUs. For example, if your application performs a multiget on 3 items, it will consume 3 ECPUs.
- Commands that operate on more items and transfer more data consume ECPUs based on the higher of the two dimensions. For example, if your application uses the GET command, retrieves 3 items, and transfers 3.2 KB of data, it will consume 3.2 ECPU. Alternatively, if it transfers only 2 KB of data, it will consume 3 ECPUs.

ElastiCache Serverless emits a new metric called `ElastiCacheProcessingUnits` that helps you understand the ECPUs consumed by your workload.

Node hours

You can choose to design your own cache cluster by choosing the EC2 node family, size, number of nodes, and placement across Availability Zones. When self-designing your cluster, you pay per hour for each cache node.

Common ElastiCache Use Cases and How ElastiCache Can Help

Whether serving the latest news or a product catalog, or selling tickets to an event, speed is the name of the game. The success of your website and business is greatly affected by the speed at which you deliver content.

In "For Impatient Web Users, an Eye Blink Is Just Too Long to Wait," the New York Times noted that users can register a 250-millisecond (1/4 second) difference between competing sites. Users tend to opt out of the slower site in favor of the faster site. Tests done at Amazon, cited in How Webpage Load Time Is Related to Visitor Loss, revealed that for every 100-ms (1/10 second) increase in load time, sales decrease 1 percent.

If someone wants data, you can deliver that data much faster if it's cached. That's true whether it's for a webpage or a report that drives business decisions. Can your business afford to not cache your webpages so as to deliver them with the shortest latency possible?

It might seem intuitively obvious that you want to cache your most heavily requested items. But why not cache your less frequently requested items? Even the most optimized database query or remote API call is noticeably slower than retrieving a flat key from an in-memory cache. *Noticeably slower* tends to send customers elsewhere.

The following examples illustrate some of the ways using ElastiCache can improve overall performance of your application.

In-Memory Data Store

The primary purpose of an in-memory key-value store is to provide ultrafast (submillisecond latency) and inexpensive access to copies of data. Most data stores have areas of data that are frequently accessed but seldom updated. Additionally, querying a database is always slower and more expensive than locating a key in a key-value pair cache. Some database queries are especially expensive to perform. An example is queries that involve joins across multiple tables or queries with intensive calculations. By caching such query results, you pay the price of the query only once. Then you can quickly retrieve the data multiple times without having to re-execute the query.

What Should I Cache?

When deciding what data to cache, consider these factors:

Speed and expense – It's always slower and more expensive to get data from a database than from a cache. Some database queries are inherently slower and more expensive than others. For example, queries that perform joins on multiple tables are much slower and more expensive than simple, single table queries. If the interesting data requires a slow and expensive query to get, it's a candidate for caching. If getting the data requires a relatively quick and simple query, it might still be a candidate for caching, depending on other factors.

Data and access pattern – Determining what to cache also involves understanding the data itself and its access patterns. For example, it doesn't make sense to cache data that changes quickly or is seldom accessed. For caching to provide a real benefit, the data should be relatively static and frequently accessed. An example is a personal profile on a social media site. On the other hand, you don't want to cache data if caching it provides no speed or cost advantage. For example, it doesn't make sense to cache webpages that return search results because the queries and results are usually unique.

Staleness – By definition, cached data is stale data. Even if in certain circumstances it isn't stale, it should always be considered and treated as stale. To tell whether your data is a candidate for caching, determine your application's tolerance for stale data.

Your application might be able to tolerate stale data in one context, but not another. For example, suppose that your site serves a publicly traded stock price. Your customers might accept some staleness with a disclaimer that prices might be *n* minutes delayed. But if you serve that stock price to a broker making a sale or purchase, you want real-time data.

Consider caching your data if the following is true:

- Your data is slow or expensive to get when compared to cache retrieval.
- Users access your data often.
- Your data stays relatively the same, or if it changes quickly staleness is not a large issue.

For more information, see the following:

- Caching Strategies in the *ElastiCache for Memcached User Guide*

ElastiCache Customer Testimonials

To learn about how businesses like Airbnb, PBS, Esri, and others use Amazon ElastiCache to grow their businesses with improved customer experience, see How Others Use Amazon ElastiCache.

You can also watch the Tutorial videos for additional ElastiCache customer use cases.

ElastiCache for Memcached resources

We recommend that you begin by reading the following sections, and refer to them as you need them:

- **Service Highlights and Pricing** – The product detail page provides a general product overview of ElastiCache, service highlights, and pricing.
- **ElastiCache Videos** – The ElastiCache Videos section has videos that introduce you to Amazon ElastiCache for Memcached, cover common use cases, and demo how to use ElastiCache for Memcached to reduce latency and improve throughput of your applications.
- **Getting Started** – The Getting started with Amazon ElastiCache for Memcached section includes an exame that walks you through the process of creating a cache cluster, authorizing access to the cache cluster, connecting to a cache node, and deleting the cache cluster.
- **Performance at Scale** – The Performance at scale with Amazon ElastiCache white paper addresses caching strategies that enable your application to perform well at scale.

If you want to use the AWS Command Line Interface (AWS CLI), you can use these documents to help you get started:

- AWS Command Line Interface documentation

  This section provides information on downloading the AWS CLI, getting the AWS CLI working on your system, and providing your AWS credentials.
- AWS CLI documentation for ElastiCache

  This separate document covers all of the AWS CLI for ElastiCache commands, including syntax and examples.

You can write application programs to use the ElastiCache API with a variety of popular programming languages. Here are some resources:

- Tools for Amazon Web Services

  Amazon Web Services provides a number of software development kits (SDKs) with support for ElastiCache for Memcached. You can code for ElastiCache using Java, .NET, PHP, Ruby, and other languages. These SDKs can greatly simplify your application development by formatting your requests to ElastiCache, parsing responses, and providing retry logic and error handling.

- Using the ElastiCache API

  If you don't want to use the AWS SDKs, you can interact with ElastiCache directly using the Query API. You can find troubleshooting tips and information on creating and authenticating requests and handling responses in this section.
- Amazon ElastiCache API Reference

  This separate document covers all of the ElastiCache API operations, including syntax and examples.

Tools for managing your implementation

Once you have granted your Amazon EC2 instance access to your ElastiCache cluster, you have four means by which you can manage your ElastiCache cluster: the AWS Management Console, the AWS CLI for ElastiCache, the AWS SDK for ElastiCache, and the ElastiCache API.

Using the AWS Management Console

The AWS Management Console is the easiest way to manage Amazon ElastiCache for Memcached. The console lets you create cache clusters, add and remove cache nodes, and perform other administrative tasks without having to write any code. The console also provides cache node performance graphs from CloudWatch, showing cache engine activity, memory and CPU utilization, as well as other metrics. For more information, see specific topics in this *User Guide*.

Using the AWS CLI

You can also use the AWS Command Line Interface (AWS CLI) for ElastiCache. The AWS CLI makes it easy to perform one-at-a-time operations, such as starting or stopping your cache cluster. You can also invoke AWS CLI for ElastiCache commands from a scripting language of your choice, letting you automate repeating tasks. For more information about the AWS CLI, see the *User Guide* and the AWS CLI Command Reference.

Using the AWS SDK

If you want to access ElastiCache from an application, you can use one of the AWS software development kits (SDKs). The SDKs wrap the ElastiCache API calls, and insulate your application from the low-level details of the ElastiCache API. You provide your credentials, and the SDK libraries take care of authentication and request signing. For more information about using the AWS SDKs, see Tools for Amazon Web Services.

Using the ElastiCache API

You can also write application code directly against the ElastiCache web service API. When using the API, you must write the necessary code to construct and authenticate your HTTP requests, parse the results from ElastiCache, and handle any errors. For more information about the API, see Using the ElastiCache API.

See also

For more detailed information on managing your Amazon ElastiCache for Memcached deployment, see the following:

- Working with ElastiCache
- Internetwork traffic privacy
- Logging and monitoring in Amazon ElastiCache

Choosing between deployment options

Amazon ElastiCache has two deployment options:

- Serverless caching
- Design your own cluster

Serverless caching

Amazon ElastiCache Serverless simplifies cache creation and instantly scales to support customers' most demanding applications. With ElastiCache Serverless, you can create a highly-available and scalable cache in less than a minute, eliminating the need to provision, plan for, and manage cache cluster capacity. ElastiCache Serverless automatically stores data redundantly across three Availability Zones and provides a 99.99% availability Service Level Agreement (SLA). ElastiCache provides automatic data replication across AZs eliminating the need to manually manage replicas and custom software to keep them in sync.

Self-designed clusters

If you need fine-grained control over your ElastiCache for Memcached cluster, you can choose to design your own Memcached cluster with ElastiCache. ElastiCache enables you to operate a node-based cluster, by choosing the node-type, number of nodes, and node placement across AWS Availability Zones for your cluster. Since ElastiCache is a fully-managed service, it automatically manages hardware provisioning, monitoring, node replacements, and software patching for your cluster.

Choosing between deployment options

Choose Serverless caching if:

- You are creating a new cache for new or unknown workloads
- You have unpredictable application traffic
- You want the easiest way to get started with a cache

Choose to design your own ElastiCache cluster if:

- You are already running ElastiCache Serverless, and want finer grained control over the type of node running Memcached, number of nodes, and placement of nodes.
- You don't expect your application traffic to fluctuate much or you can accurately predict your application traffic peaks and troughs.
- You can forecast your capacity requirements to control costs.

Related topics:

- Designing and managing your own ElastiCache cluster for Memcached implementation

Comparing Memcached and Redis self-designed caches

Amazon ElastiCache supports the Memcached and Redis cache engines. Each engine provides some advantages. Use the information in this topic to help you choose the engine and version that best meets your requirements.

> ⚠ **Important**
>
> After you create a cache cluster or replication group, you can upgrade to a newer engine version, but you cannot downgrade to an older engine version. If you want to use an older engine version, you must delete the existing cache cluster or replication group and create it again with the earlier engine version.

On the surface, the engines look similar. Each of them is an in-memory key-value store. However, in practice there are significant differences.

Choose Memcached if the following apply for you:

- You need the simplest model possible.

- You need to run large nodes with multiple cores or threads.
- You need the ability to scale out and in, adding and removing nodes as demand on your system increases and decreases.
- You need to cache objects.

Choose Redis with a version of ElastiCache for Redis if the following apply for you:

- **ElastiCache for Redis version 7.0 (Enhanced)**

  You want to use Redis Functions, Sharded Pub/Sub, or Redis ACL improvements. For more information, see Redis Version 7.0 (Enhanced).
- **ElastiCache for Redis version 6.2 (Enhanced)**

  You want the ability to tier data between memory and SSD using the r6gd node type. For more information, see Data tiering.
- **ElastiCache for Redis version 6.0 (Enhanced)**

  You want to authenticate users with role-based access control.

  For more information, see Redis Version 6.0 (Enhanced).
- **ElastiCache for Redis version 5.0.0 (Enhanced)**

  You want to use Redis streams, a log data structure that allows producers to append new items in real time and also allows consumers to consume messages either in a blocking or non-blocking fashion.

  For more information, see Redis Version 5.0.0 (Enhanced).
- **ElastiCache for Redis version 4.0.10 (Enhanced)**

  Supports both encryption and dynamically adding or removing shards from your Redis (cluster mode enabled) cluster.

  For more information, see Redis Version 4.0.10 (Enhanced).

The following versions are deprecated, have reached or soon to reach end of life.

- **ElastiCache for Redis version 3.2.10 (Enhanced)**

Supports the ability to dynamically add or remove shards from your Redis (cluster mode enabled) cluster.

> ⚠ **Important**
>
> Currently ElastiCache for Redis 3.2.10 doesn't support encryption.

For more information, see the following:

- Redis Version 3.2.10 (Enhanced)
- Online resharding best practices for Redis, For more information, see the following:
    - Best Practices: Online Resharding
    - Online Resharding and Shard Rebalancing for Redis (Cluster Mode Enabled)
- For more information on scaling Redis clusters, see Scaling.

- **ElastiCache for Redis version 3.2.6 (Enhanced)**

  If you need the functionality of earlier Redis versions plus the following features, choose ElastiCache for Redis 3.2.6:

  - In-transit encryption. For more information, see Amazon ElastiCache for Redis In-Transit Encryption.
  - At-rest encryption. For more information, see Amazon ElastiCache for Redis At-Rest Encryption.
- **ElastiCache for Redis (Cluster mode enabled) version 3.2.4**

  If you need the functionality of Redis 2.8.x plus the following features, choose Redis 3.2.4 (clustered mode):

  - You need to partition your data across two to 500 node groups (clustered mode only).
  - You need geospatial indexing (clustered mode or non-clustered mode).
  - You don't need to support multiple databases.
- **ElastiCache for Redis (non-clustered mode) 2.8.x and 3.2.4 (Enhanced)**

  If the following apply for you, choose Redis 2.8.x or Redis 3.2.4 (non-clustered mode):

  - You need complex data types, such as strings, hashes, lists, sets, sorted sets, and bitmaps.

- You need to sort or rank in-memory datasets.
- You need persistence of your key store.
- You need to replicate your data from the primary to one or more read replicas for read intensive applications.
- You need automatic failover if your primary node fails.
- You need publish and subscribe (pub/sub) capabilities—to inform clients about events on the server.
- You need backup and restore capabilities.
- You need to support multiple databases.

Comparison summary of Memcached, Redis (cluster mode disabled), and Redis (cluster mode enabled)

	Memcached	**Redis (cluster mode disabled)**	**Redis (cluster mode enabled)**
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
Compliance certifications			
Compliance Certification			
FedRAMP	Yes - 1.6.12 and later	4.0.10 and later	4.0.10 and later
HIPAA	Yes - 1.6.12 and later	4.0.10 and later	4.0.10 and later
PCI DSS	Yes	4.0.10 and later	4.0.10 and later
Multi-threaded	Yes	No	No
Node type upgrade	No	Yes	Yes
Engine upgrading	Yes	Yes	Yes

	Memcached	**Redis (cluster mode disabled)**	**Redis (cluster mode enabled)**
High availability (replication)	No	Yes	Yes
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

Notes:

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.

After you choose the engine for your cluster, we recommend that you use the most recent version of that engine. For more information, see Supported ElastiCache for Memcached Versions or Supported ElastiCache for Redis Versions.

Getting started with Amazon ElastiCache for Memcached

The topics in this section walk you through the process of creating, granting access to, connecting to, and finally deleting a Memcached serverless cache using the ElastiCache console.

Topics

Setting up

To set up ElastiCache:

Topics

Sign up for an AWS account

If you do not have an AWS account, complete the following steps to create one.

To sign up for an AWS account

1. Open https://portal.aws.amazon.com/billing/signup.

2. Follow the online instructions.

   Part of the sign-up procedure involves receiving a phone call and entering a verification code on the phone keypad.

   When you sign up for an AWS account, an *AWS account root user* is created. The root user has access to all AWS services and resources in the account. As a security best practice, assign administrative access to a user, and use only the root user to perform tasks that require root user access.

AWS sends you a confirmation email after the sign-up process is complete. At any time, you can view your current account activity and manage your account by going to https://aws.amazon.com/ and choosing **My Account**.

Create a user with administrative access

After you sign up for an AWS account, secure your AWS account root user, enable AWS IAM Identity Center, and create an administrative user so that you don't use the root user for everyday tasks.

Secure your AWS account root user

1. Sign in to the AWS Management Console as the account owner by choosing **Root user** and entering your AWS account email address. On the next page, enter your password.

   For help signing in by using root user, see Signing in as the root user in the *AWS Sign-In User Guide*.

2. Turn on multi-factor authentication (MFA) for your root user.

   For instructions, see Enable a virtual MFA device for your AWS account root user (console) in the *IAM User Guide*.

Create a user with administrative access

1. Enable IAM Identity Center.

   For instructions, see Enabling AWS IAM Identity Center in the *AWS IAM Identity Center User Guide*.

2. In IAM Identity Center, grant administrative access to a user.

For a tutorial about using the IAM Identity Center directory as your identity source, see Configure user access with the default IAM Identity Center directory in the *AWS IAM Identity Center User Guide*.

Sign in as the user with administrative access

- To sign in with your IAM Identity Center user, use the sign-in URL that was sent to your email address when you created the IAM Identity Center user.

  For help signing in using an IAM Identity Center user, see Signing in to the AWS access portal in the *AWS Sign-In User Guide*.

Assign access to additional users

1. In IAM Identity Center, create a permission set that follows the best practice of applying least-privilege permissions.

   For instructions, see Create a permission set in the *AWS IAM Identity Center User Guide*.
2. Assign users to a group, and then assign single sign-on access to the group.

   For instructions, see Add groups in the *AWS IAM Identity Center User Guide*.

Grant programmatic access

Users need programmatic access if they want to interact with AWS outside of the AWS Management Console. The way to grant programmatic access depends on the type of user that's accessing AWS.

To grant users programmatic access, choose one of the following options.

Which user needs programmatic access?	To	By
Workforce identity (Users managed in IAM Identity Center)	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use.

Which user needs programmatic access?	To	By
		- For the AWS CLI, see Configuring the AWS CLI to use AWS IAM Identity Center in the *AWS Command Line Interface User Guide*. - For AWS SDKs, tools, and AWS APIs, see IAM Identity Center authentication in the *AWS SDKs and Tools Reference Guide*.
IAM	Use temporary credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions in Using temporary credentials with AWS resources in the *IAM User Guide*.

Which user needs programmatic access?	To	By
IAM	(Not recommended) Use long-term credentials to sign programmatic requests to the AWS CLI, AWS SDKs, or AWS APIs.	Following the instructions for the interface that you want to use. • For the AWS CLI, see Authenticating using IAM user credentials in the *AWS Command Line Interface User Guide*. • For AWS SDKs and tools, see Authenticate using long-term credentials in the *AWS SDKs and Tools Reference Guide*. • For AWS APIs, see Managing access keys for IAM users in the *IAM User Guide*.

Related topics:

- What is IAM in the *IAM User Guide*.
- AWS Security Credentials in *AWS General Reference*.

Set up your permissions (new ElastiCache users only)

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.
- Users managed in IAM through an identity provider:

Create a role for identity federation. Follow the instructions in Creating a role for a third-party identity provider (federation) in the *IAM User Guide*.

- IAM users:
  - Create a role that your user can assume. Follow the instructions in Creating a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

Amazon ElastiCache creates and uses service-linked roles to provision resources and access other AWS resources and services on your behalf. For ElastiCache to create a service-linked role for you, use the AWS-managed policy named `AmazonElastiCacheFullAccess`. This role comes preprovisioned with permission that the service requires to create a service-linked role on your behalf.

You might decide not to use the default policy and instead to use a custom-managed policy. In this case, make sure that you have either permissions to call `iam:createServiceLinkedRole` or that you have created the ElastiCache service-linked role.

For more information, see the following:

- Creating a New Policy (IAM)
- AWS managed policies for Amazon ElastiCache
- Using Service-Linked Roles for Amazon ElastiCache

Set up EC2

You will need to setup an EC2 instance from which you will connect to your cache.

- If you don't already have an EC2 instance, learn how to setup an EC2 instance here: Getting started with EC2.
- Your EC2 instance must be in the same VPC and have the same security group settings as your cache. By default, Amazon ElastiCache creates a cache in your default VPC and uses the default security group. To follow this tutorial, ensure that your EC2 instance is in the default VPC and has the default security group.

Grant network access from an Amazon VPC security group to your cache

ElastiCache for Memcached uses the 11211 and 11212 ports to accept Memcached commands. In order to successfully connect and execute Memcached commands from your EC2 instance, your security group must allow access to these ports.

1. Sign in to the AWS Command Line Interface and open the Amazon EC2 console.
2. In the navigation pane, under **Network & Security**, choose **Security Groups**.
3. From the list of security groups, choose the security group for your Amazon VPC. Unless you created a security group for ElastiCache use, this security group will be named *default*.
4. Choose the Inbound tab, and then:

   a. Choose **Edit**.

   b. Choose **Add rule**.

   c. In the Type column, choose **Custom TCP rule**.

   d. In the **Port range** box, type 11211.

   e. In the **Source** box, choose **Anywhere** which has the port range (0.0.0.0/0) so that any Amazon EC2 instance that you launch within your Amazon VPC can connect to your cache.

   f. If you are using ElastiCache serverless, add another rule by choosing **Add rule**.

   g. In the **Type** column, choose **Custom TCP** rule.

   h. In the **Port range**box, type 11212.

   i. In the **Source** box, choose **Anywhere** which has the port range (0.0.0.0/0) so that any Amazon EC2 instance that you launch within your Amazon VPC can connect to your cache.

   j. Choose **Save**

Step 1: Create a cache

The cache you're about to launch will be live, and not running in a sandbox. You incur the standard ElastiCache usage fees for the cache until it is deleted. The total charges are minimal (typically less than a dollar) if you complete the exercise described here in one sitting and delete the cache when you are finished. For more information about ElastiCache usage rates, see Amazon ElastiCache.

Create a serverless cache

AWS Management Console

To create a new cache using the ElastiCache console:

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the navigation pane on the left side of the console, choose **Memcached Caches**.
3. On the right side of the console, choose **Create Memcached cache**.
4. In the **Cache settings** enter a **Name**. You can optionally enter a **description** for the cache.
5. Leave the default settings selected.
6. Click **Create** to create the cache.
7. Once the cache is in "ACTIVE" status, you can begin writing and reading data to the cache.

To create a new cache using the AWS CLI

The following AWS CLI example creates a new cache using create-serverless-cache.

Linux

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

Windows

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

Note that the value of the Status field is set to `CREATING`.

To verify that ElastiCache has finished creating the cache, use the `describe-serverless-caches` command.

Linux

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

Windows

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

After creating the new cache, proceed to Step 2: Read and write data to the cache.

Step 2: Read and write data to the cache

This section assumes that you've created an Amazon EC2 instance and can connect to it. For instructions on how to do this, see the Amazon EC2 Getting Started Guide.

By default, ElastiCache creates a cache in your default VPC. Make sure that your EC2 instance is also created in the default VPC, so that it is able to connect to the cache.

Configuration

Before you begin, make sure you have the right ports available for access.

Primary port: 11211

Read-optimized port: 11212

Serverless Memcached caches advertise two ports with the same hostname. The primary port allows writes and reads with same consistency guarantees as OSS Memcached. The read-optimized port allows writes and additionally lower-latency eventually-consistent reads.

Find your cache endpoint

AWS Management Console

To find your cache’s endpoint using the ElastiCache console:

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the navigation pane on the left side of the console, choose **Memcached Caches**.
3. On the right side of the console, click on the name of the cache that you just created.
4. In the **Cache details**, locate and copy the cache endpoint.

AWS CLI

The following AWS CLI example shows to find the endpoint for your new cache using the describe-serverless-caches command. Once you have run the command, look for the "Endpoint" field.

Linux

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

Windows

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

Connect using OpenSSL

For information on how to connect using OpenSSL, see ElastiCache in-transit encryption (TLS)

Connect using Memcached Java client

For information on how to connect using the Memcached Java client, see ElastiCache in-transit encryption (TLS)

Connect using Memcached PHP client

```
<?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);
?>
```

Connect using Memcached Python client (Pymemcache)

See 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"
```

Connect using Memcached NodeJS/TS client (Electrode-IO memcache)

See https://github.com/electrode-io/memcache and https://www.npmjs.com/package/memcache-client

Install via `npm i memcache-client`

In the application, create a memcached TLS client as follows:

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

Connect using Memcached Rust client (rust-memcache)

See https://crates.io/crates/memcache and 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();
```

Connect using Memcached Go client (Gomemcache)

See 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)
```

Connect using Memcached Ruby client (Dalli)

See 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")
```

Connect using Memcached .NET client (EnyimMemcachedCore)

See https://github.com/cnblogs/EnyimMemcachedCore

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

You may now proceed to Step 3: (Optional) Clean up.

Step 3: (Optional) Clean up

Using the AWS Management Console

The following procedure deletes a single cache from your deployment. To delete multiple caches, repeat the procedure for each cache that you want to delete. You do not need to wait for one cache to finish deleting before starting the procedure to delete another cache.

To delete a cache

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the ElastiCache console dashboard, choose the engine the cache that you want to delete is running. A list of all caches running that engine appears.
3. To choose the cache to delete, choose the cache's name from the list of caches.

   > ⚠ **Important**
   >
   > You can only delete one cache at a time from the ElastiCache console. Choosing multiple caches disables the delete operation.

4. For **Actions**, choose **Delete**.
5. In the **Delete Cache** confirmation screen, choose **Delete** to delete the cache, or choose **Cancel** to keep the cluster.
6. If you chose **Delete**, the status of the cache changes to *deleting*.

As soon as your cache moves in to the **DELETING** status, you stop incurring charges for it.

Using the AWS CLI

The following code deletes the cache my-cache.

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

The delete-serverless-cache CLI action only deletes one serverless cache. To delete multiple caches, call delete-serverless-cache for each serverless cache that you want to delete. You do not need to wait for one serverless cache to finish deleting before deleting another.

For Linux, macOS, or Unix:

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

For Windows:

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

For more information, see the AWS CLI for ElastiCache topic delete-serverless-cache.

You may now proceed to Next Steps.

Next Steps

For more information about ElastiCache see:

- Working with ElastiCache
- Scaling ElastiCache for Memcached
- Quotas for ElastiCache
- ElastiCache best practices and caching strategies
- Viewing ElastiCache events

Tutorial: Configuring a Lambda function to access amazon ElastiCache in an Amazon VPC

In this tutorial, you do the following:

- Create an Amazon ElastiCache cache in your default Amazon Virtual Private Cloud (Amazon VPC) in the us-east-1 region.
- Create a Lambda function to access the ElastiCache cache. When you create the Lambda function, you provide subnet IDs in your Amazon VPC and a VPC security group to allow the Lambda function to access resources in your VPC. For illustration in this tutorial, the Lambda function generates a UUID, writes it to the cache, and retrieves it from the cache.
- Invoke the Lambda function manually and verify that it accessed the ElastiCache cache in your VPC.

> ⚠ **Important**
>
> The tutorial uses the default Amazon VPC in the us-east-1 region in your account. For more information about Amazon VPC, see How to Get Started with Amazon VPC in the *Amazon VPC User Guide*.

Topics

Get Started

Step 1: Create an ElastiCache cache

Step 1: Create an ElastiCache cache

In this step you create an Amazon ElastiCache cache in the default Amazon Virtual Private Cloud in the us-east-1 region in your account using the AWS CLI. For information on creating ElastiCache serverless cache using the ElastiCache console or API, see Creating a cluster in the *ElastiCache for Memcached User Guide*.

AWS Management Console

Run the following AWS CLI command to create a new Memcached cluster serverless cache in the default VPC in the us-east-1 region.

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
```

Note that the value of the Status field is set to `CREATING`. It can take a few minutes for ElastiCache to finish creating your cluster.

To verify that ElastiCache has finished creating the cache, use the `describe-serverless-caches` command.

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
```

Copy the Endpoint Address shown in the output. You'll need this address when you create the deployment package for your Lambda function.

After creating the new cache, proceed to Step 2: Create a Lambda function

Next Step:

[Step 2: Create a Lambda function]

Step 2: Create a Lambda function

In this step you do the following:

1. Create a Lambda function deployment package using the sample code provided.
2. Create an IAM role (execution role). At the time you upload the deployment package, you need to specify this role so that Lambda can assume the role and then execute the function on your behalf. The permissions policy grants AWS Lambda permissions to set up elastic network interfaces or ENIs to enable your Lambda function to access resources in the VPC. In this example, your Lambda function accesses an ElastiCache cluster in the VPC.
3. Create the Lambda function by uploading the deployment package.

Next Step

[Step 2.1: Create the deployment package]

Step 2.1: Create the deployment package

Currently the example code for the Lambda function is only supplied in Python.

Python

The following example Python code reads and writes an item to your ElastiCache cluster. Copy the code and save it into a file named `app.py`. Be sure to replace the `elasticache_config_endpoint` value in the code with the endpoint address you copied in step 1.

```
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"
```

This code uses the Python pymemcache library to put items into your cache and retrieve them. To create a deployment package containing pymemcache, carry out the following steps.

1. In your project directory containing the app.py source code file, create a folder package to install the pymemacache library into.

   ```
   mkdir package
   ```

2. Install pymemcache using pip.

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

3. Create a .zip file containing the pymemcache library. In Linux and macOS, run the following command. In Windows, use your preferred zip utility to create a .zip file with the pymemache library at the root.

   ```
   cd package
   ```

```
zip -r ../my_deployment_package.zip .
```

4. Add your function code to the .zip file. In Linux and macOS, run the following command. In Windows, use your preferred zip utility to add `app.py` to the root of your .zip file.

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

Next Step

[Step 2.2: Create the IAM role (execution role)]

Step 2.2: Create the IAM role (execution role)

In this step, you create an AWS Identity and Access Management (IAM) role using the following predefined role type and access policy:

- AWS service role of the type **AWS Lambda** – This role grants AWS Lambda permissions to assume the role.
- **AWSLambdaVPCAccessExecutionRole** – This is the access permissions policy that you attach to the role. The policy grants permission for the EC2 actions that AWS Lambda needs to manage ENIs. You can view this AWS-managed policy in IAM console.

For more information about IAM user roles, see Roles (Delegation and Federation) in the *IAM User Guide*.

Use the following procedure to create the IAM role.

To create an IAM (execution) role

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. Choose **Roles** and then **Create role**.
   - Under **Trusted entity type**, choose **AWS Service**, and then under **Use cases** choose **Lambda**. This grants the AWS Lambda service permissions to assume the role. Choose **Next**.
   - Under **Add permissions**, search for **`AWSLambdaVPCAccessExecutionRole`** and select the check box next to the policy.

   - Choose **Next**.
   - In **Role Name**, use a name that is unique within your AWS account (for example, **lambda-vpc-execution-role**).
   - Choose **Create role**.

3. Copy the role ARN. You will need it in the next step when you create your Lambda function.

Next Step

Step 2.3: Upload the deployment package (create the Lambda function)

Step 2.3: Upload the deployment package (create the Lambda function)

In this step, you create the Lambda function (`AccessMemcached`) using the `create-function` AWS CLI command.

From the project directory containing your deployment package .zip file, run the following Lambda CLI `create-function` command.

For the `role` option, use the ARN of the execution role you created in step 2.2. For the `vpc-config` enter comma separated lists of your default VPC's subnets and your default VPC's security group ID. You can find these values in the Amazon VPC console. To find your default VPC's subnets, choose **Your VPCs**, then choose your AWS account's default VPC. To find the security group for this VPC, under **Security**, choose **Security groups**. Ensure that you have the us-east-1 region selected.

For Linux, macOS, or 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 \
```

For 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 ^
```

Optionally, you can upload the .zip file to an Amazon S3 bucket in the same AWS region, and then specify the bucket and object name in the preceding command. You need to replace the `--zip-file` parameter with the `--code` parameter, as shown following:

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

You can also create the Lambda function using the AWS Lambda console. When creating the function, choose a VPC for the Lambda and then select the subnets and security groups from the provided fields.

Next Step

Step 3: Test the Lambda function

Step 3: Test the Lambda function

In this step, you invoke the Lambda function manually using the `invoke` command. When the Lambda function executes, it generates a UUID and writes it to the ElastiCache cluster that you specified in your Lambda code. The Lambda function then retrieves the item from the cache.

1. Invoke the Lambda function (`AccessMemCache`) using the AWS Lambda `invoke` command.

   For Linux, macOS, or Unix:

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

   For Windows:

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

2. Verify that the Lambda function executed successfully as follows:

   - Review the output.txt file.
   - Verify the results in CloudWatch Logs by opening the CloudWatch console and choosing the log group for your function (`/aws/lambda/AccessMemcached`). The log stream should contain output similar to the following:

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

   - Review the results in the AWS Lambda console.

ElastiCache tutorials and videos

The following tutorials address tasks of interest to the Amazon ElastiCache user.

- ElastiCache Videos
- Tutorial: Configuring a Lambda Function to Access Amazon ElastiCache in an Amazon VPC

ElastiCache Videos

Following, you can find videos to help you learn basic and advanced Amazon ElastiCache concepts. For information about AWS Training, see AWS Training & Certification.

Topics

- Introductory Videos
- Advanced Videos

Introductory Videos

The following videos introduce you to Amazon ElastiCache.

Topics

- AWS re:Invent 2020: What's new in Amazon ElastiCache
- AWS re:Invent 2019: What's new in Amazon ElastiCache
- AWS re:Invent 2017: What's new in Amazon ElastiCache
- DAT204—Building Scalable Applications on AWS NoSQL Services (re:Invent 2015)
- DAT207—Accelerating Application Performance with Amazon ElastiCache (AWS re:Invent 2013)

AWS re:Invent 2020: What's new in Amazon ElastiCache

AWS re:Invent 2020: What's new in Amazon ElastiCache

AWS re:Invent 2019: What's new in Amazon ElastiCache

AWS re:Invent 2019: What's new in Amazon ElastiCache

AWS re:Invent 2017: What's new in Amazon ElastiCache

AWS re:Invent 2017: What's new in Amazon ElastiCache

DAT204—Building Scalable Applications on AWS NoSQL Services (re:Invent 2015)

In this session, we discuss the benefits of NoSQL databases and take a tour of the main NoSQL services offered by AWS—Amazon DynamoDB and Amazon ElastiCache. Then, we hear from two leading customers, Expedia and Mapbox, about their use cases and architectural challenges, and how they addressed them using AWS NoSQL services, including design patterns and best practices. You should come out of this session having a better understanding of NoSQL and its powerful capabilities, ready to tackle your database challenges with confidence.

DAT204—Building Scalable Applications on AWS NoSQL Services (re:Invent 2015)

DAT207—Accelerating Application Performance with Amazon ElastiCache (AWS re:Invent 2013)

In this video, learn how you can use Amazon ElastiCache to easily deploy an in-memory caching system to speed up your application performance. We show you how to use Amazon ElastiCache to improve your application latency and reduce the load on your database servers. We'll also show you how to build a caching layer that is easy to manage and scale as your application grows. During this session, we go over various scenarios and use cases that can benefit by enabling caching, and discuss the features provided by Amazon ElastiCache.

DAT207 - Accelerating Application Performance with Amazon ElastiCache (re:Invent 2013)

Advanced Videos

The following videos cover more advanced Amazon ElastiCache topics.

Topics

- Design for success with Amazon ElastiCache best practices (re:Invent 2020)
- Supercharge your real-time apps with Amazon ElastiCache (re:Invent 2019)
- Best practices: migrating Redis clusters from Amazon EC2 to ElastiCache (re:Invent 2019)
- Scaling a Fantasy Sports Platform with Amazon ElastiCache & Amazon Aurora STP11 (re:Invent 2018)
- Reliable & Scalable Redis in the Cloud with Amazon ElastiCache (re:Invent 2018)
- ElastiCache Deep Dive: Design Patterns for In-Memory Data Stores (re:Invent 2018)
- DAT305—Amazon ElastiCache Deep Dive (re:Invent 2017)
- DAT306—Amazon ElastiCache Deep Dive (re:Invent 2016)
- DAT407—Amazon ElastiCache Deep Dive (re:Invent 2015)
- SDD402—Amazon ElastiCache Deep Dive (re:Invent 2014)
- DAT307—Deep Dive into Amazon ElastiCache Architecture and Design Patterns (re:Invent 2013)

Design for success with Amazon ElastiCache best practices (re:Invent 2020)

With the explosive growth of business-critical, real-time applications built on Redis, availability, scalability, and security have become top considerations. Learn best practices for setting up Amazon ElastiCache for success with online scaling, high availability across Multi-AZ deployments, and security configurations.

Design for success with Amazon ElastiCache best practices (re:Invent 2020)

Supercharge your real-time apps with Amazon ElastiCache (re:Invent 2019)

With the rapid growth in cloud adoption and the new scenarios that it empowers, applications need microsecond latency and high throughput to support millions of requests per second. Developers have traditionally relied on specialized hardware and workarounds, such as disk-based databases combined with data reduction techniques, to manage data for real-time applications. These approaches can be expensive and not scalable. Learn how you can boost the performance of real-time applications by using the fully managed, in-memory Amazon ElastiCache for extreme performance, high scalability, availability, and security.

Supercharge your real-time apps with Amazon ElastiCache (re:Invent 2019:)

Best practices: migrating Redis clusters from Amazon EC2 to ElastiCache (re:Invent 2019)

Managing Redis clusters on your own can be hard. You have to provision hardware, patch software, back up data, and monitor workloads constantly. With the newly released Online Migration feature for Amazon ElastiCache, you can now easily move your data from self-hosted Redis on Amazon EC2 to fully managed Amazon ElastiCache, with cluster mode disabled. In this session, you learn about the new Online Migration tool, see a demo, and, more importantly, learn hands-on best practices for a smooth migration to Amazon ElastiCache.

Best practices: migrating Redis clusters from Amazon EC2 to ElastiCache (re:Invent 2019)

Scaling a Fantasy Sports Platform with Amazon ElastiCache & Amazon Aurora STP11 (re:Invent 2018)

Dream11 is India’s leading sports-tech startup. It has a growing base of 40 million+ users playing multiple sports, including fantasy cricket, football, and basketball, and it currently serves one million concurrent users, who produce three million requests per minute under a 50-millisecond response time. In this talk, Dream11 CTO Amit Sharma explains how the company uses Amazon Aurora and Amazon ElastiCache to handle flash traffic, which can triple within a 30-second response window. Sharma also talks about scaling transactions without locking, and he shares the steps for handling flash traffic—thereby serving five million daily active users. Complete Title: AWS re:Invent 2018: Scaling a Fantasy Sports Platform with Amazon ElastiCache & Amazon Aurora (STP11)

Scaling a Fantasy Sports Platform with Amazon ElastiCache & Amazon Aurora STP11 (re:Invent 2018)

Reliable & Scalable Redis in the Cloud with Amazon ElastiCache (re:Invent 2018)

This session covers the features and enhancements in our Redis-compatible service, Amazon ElastiCache for Redis. We cover key features, such as Redis 5, scalability and performance improvements, security and compliance, and much more. We also discuss upcoming features and customer case studies.

Reliable & Scalable Redis in the Cloud with Amazon ElastiCache (re:Invent 2018)

ElastiCache Deep Dive: Design Patterns for In-Memory Data Stores (re:Invent 2018)

In this session, we provide a behind the scenes peek to learn about the design and architecture of Amazon ElastiCache. See common design patterns with our Redis and Memcached offerings and how customers use them for in-memory data processing to reduce latency and improve application throughput. We review ElastiCache best practices, design patterns, and anti-patterns.

ElastiCache Deep Dive: Design Patterns for In-Memory Data Stores (re:Invent 2018)

DAT305—Amazon ElastiCache Deep Dive (re:Invent 2017)

Look behind the scenes to learn about Amazon ElastiCache's design and architecture. See common design patterns with our Memcached and Redis offerings and how customers have used them for in-memory operations to reduce latency and improve application throughput. During this video, we review ElastiCache best practices, design patterns, and anti-patterns.

The video introduces the following:

- ElastiCache for Redis online resharding
- ElastiCache security and encryption
- ElastiCache for Redis version 3.2.10

DAT305—Amazon ElastiCache Deep Dive (re:Invent 2017)

DAT306—Amazon ElastiCache Deep Dive (re:Invent 2016)

Look behind the scenes to learn about Amazon ElastiCache's design and architecture. See common design patterns with our Memcached and Redis offerings and how customers have used them for in-memory operations to reduce latency and improve application throughput. During this session, we review ElastiCache best practices, design patterns, and anti-patterns.

DAT306—Amazon ElastiCache Deep Dive (re:Invent 2016)

DAT407—Amazon ElastiCache Deep Dive (re:Invent 2015)

Peek behind the scenes to learn about Amazon ElastiCache's design and architecture. See common design patterns of our Memcached and Redis offerings and how customers have used them for in-memory operations and achieved improved latency and throughput for applications. During this session, we review best practices, design patterns, and anti-patterns related to Amazon ElastiCache.

DAT407—Amazon ElastiCache Deep Dive (re:Invent 2015)

SDD402—Amazon ElastiCache Deep Dive (re:Invent 2014)

In this video, we examine common caching use cases, the Memcached and Redis engines, patterns that help you determine which engine is better for your needs, consistent hashing, and more as means to building fast, scalable applications. Frank Wiebe, Principal Scientist at Adobe, details how Adobe uses Amazon ElastiCache to improve customer experience and scale their business.

DAT402—Amazon ElastiCache Deep Dive (re:Invent 2014)

DAT307—Deep Dive into Amazon ElastiCache Architecture and Design Patterns (re:Invent 2013)

In this video, we examine caching, caching strategies, scaling out, monitoring. We also compare the Memcached and Redis engines. During this session, also we review best practices and design patterns related to Amazon ElastiCache.

DAT307 - Deep Dive into Amazon ElastiCache Architecture and Design Patterns (AWS re:Invent 2013).

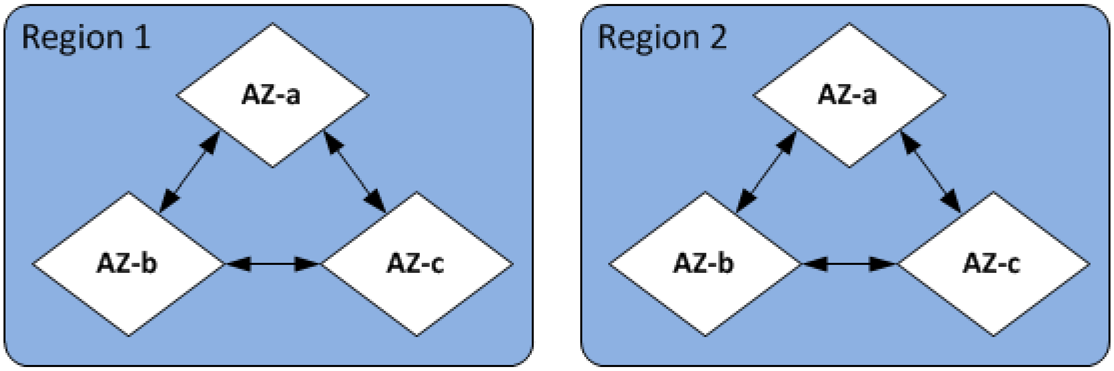
Choosing regions and availability zones

AWS Cloud computing resources are housed in highly available data center facilities. To provide additional scalability and reliability, these data center facilities are located in different physical locations. These locations are categorized by *regions* and *Availability Zones*.

AWS Regions are large and widely dispersed into separate geographic locations. Availability Zones are distinct locations within an AWS Region that are engineered to be isolated from failures in other Availability Zones. They provide inexpensive, low-latency network connectivity to other Availability Zones in the same AWS Region.

> ⚠ **Important**
>
> Each region is completely independent. Any ElastiCache activity you initiate (for example, creating clusters) runs only in your current default region.

To create or work with a cluster in a specific region, use the corresponding regional service endpoint. For service endpoints, see Supported regions & endpoints.



Regions and Availability Zones

Topics

- Availability Zone considerations
- Supported regions & endpoints
- Locating your nodes
- Using local zones with ElastiCache
- Using Outposts

Availability Zone considerations

Distributing your Memcached nodes over multiple Availability Zones within a region helps protect you from the impact of a catastrophic failure, such as a power loss within an Availability Zone.

Serverless Caching

ElastiCache serverless caching creates a highly available cache that spans multiple Availability Zones. You can specify subnets from different availability zones and same VPC as you create your cache or ElastiCache will choose subnets automatically from your default VPC.

Designing your own ElastiCache for Memcached cluster

A Memcached cluster can have up to 300 nodes. When you create or add nodes to your Memcached cluster, you can specify a single Availability Zone for all your nodes, allow ElastiCache to choose a single Availability Zone for all your nodes, specify the Availability Zones for each node, or allow ElastiCache to choose an Availability Zone for each node. New nodes can be created in different Availability Zones as you add them to an existing Memcached cluster. Once a cache node is created, its Availability Zone cannot be modified.

If you want a cluster in a single Availability Zone cluster to have its nodes distributed across multiple Availability Zones, ElastiCache can create new nodes in the various Availability Zones. You can then delete some or all of the original cache nodes. We recommend this approach.

To migrate Memcached nodes from a single Availability Zone to multiple availability zones

1. Modify your cluster by creating new cache nodes in the Availability Zones where you want them. In your request, do the following:

   - Set `AZMode` (CLI: `--az-mode`) to `cross-az`.
   - Set `NumCacheNodes` (CLI: `--num-cache-nodes`) to the number of currently active cache nodes plus the number of new cache nodes you want to create.
   - Set `NewAvailabilityZones` (CLI: `--new-availability-zones`) to a list of the zones you want the new cache nodes created in. To let ElastiCache determine the Availability Zone for each new node, don't specify a list.
   - Set `ApplyImmediately` (CLI: `--apply-immediately`) to true.

Note

If you are not using auto discovery, be sure to update your client application with the new cache node endpoints.

Before moving on to the next step, be sure the Memcached nodes are fully created and available.

2. Modify your cluster by removing the nodes you no longer want in the original Availability Zone. In your request, do the following:

   - Set `NumCacheNodes` (CLI: `--num-cache-nodes`) to the number of active cache nodes you want after this modification is applied.
   - Set `CacheNodeIdsToRemove` (CLI: `--nodes-to-remove`) to a list of the cache nodes you want to remove from the cluster.

     The number of cache node IDs listed must equal the number of currently active nodes minus the value in `NumCacheNodes`.
   - (Optional) Set `ApplyImmediately` (CLI: `--apply-immediately`) to true.

     If you don't set `ApplyImmediately` (CLI: `--apply-immediately`) to true, the node deletions will take place at your next maintenance window.

Supported regions & endpoints

Amazon ElastiCache is available in multiple AWS Regions. This means that you can launch ElastiCache clusters in locations that meet your requirements. For example, you can launch in the AWS Region closest to your customers, or launch in a particular AWS Region to meet certain legal requirements.

Each Region is designed to be completely isolated from the other Regions. Within each Region are multiple Availability Zones (AZ). ElastiCache Serverless caches automatically replicate data across multiple availability zones (except `us-west-1`, where data is replicated in two availability zones) for high availability. When designing your own ElastiCache cluster, you can choose to launch your nodes in different AZs to achieve fault tolerance. For more information on Regions and Availability Zones, see Choosing regions and availability zones at the top of this topic.

Regions where ElastiCache is supported

Region Name/Region	Endpoint	Protocol	
US East (Ohio) Region `us-east-2`	`elasticache.us-east-2.amazonaws.com`	HTTPS	
US East (N. Virginia) Region `us-east-1`	`elasticache.us-east-1.amazonaws.com`	HTTPS	
US West (N. California) Region `us-west-1`	`elasticache.us-west-1.amazonaws.com`	HTTPS	
US West (Oregon) Region `us-west-2`	`elasticache.us-west-2.amazonaws.com`	HTTPS	

Region Name/Region	Endpoint	Protocol
Canada (Central) Region `ca-central-1`	`elasticache.ca-central-1.amazonaws.com`	HTTPS
Canada (West) Region `ca-west-1`	`elasticache.ca-west-1.amazonaws.com`	HTTPS
Asia Pacific (Jakarta) `ap-southeast-3`	`elasticache.ap-southeast-3.amazonaws.com`	HTTPS
Asia Pacific (Mumbai) Region `ap-south-1`	`elasticache.ap-south-1.amazonaws.com`	HTTPS
Asia Pacific (Hyderabad) Region `ap-south-2`	`elasticache.ap-south-2.amazonaws.com`	HTTPS
Asia Pacific (Tokyo) Region `ap-northeast-1`	`elasticache.ap-northeast-1.amazonaws.com`	HTTPS
Asia Pacific (Seoul) Region `ap-northeast-2`	`elasticache.ap-northeast-2.amazonaws.com`	HTTPS
Asia Pacific (Osaka) Region `ap-northeast-3`	`elasticache.ap-northeast-3.amazonaws.com`	HTTPS

Region Name/Region	Endpoint	Protocol
Asia Pacific (Singapore) Region `ap-southeast-1`	`elasticache.ap-southeast-1.amazonaws.com`	HTTPS
Asia Pacific (Sydney) Region `ap-southeast-2`	`elasticache.ap-southeast-2.amazonaws.com`	HTTPS
Europe (Frankfurt) Region `eu-central-1`	`elasticache.eu-central-1.amazonaws.com`	HTTPS
Europe (Zurich) Region `eu-central-2`	`elasticache.eu-central-2.amazonaws.com`	HTTPS
Europe (Stockholm) Region `eu-north-1`	`elasticache.eu-north-1.amazonaws.com`	HTTPS
Middle East (Bahrain) Region `me-south-1`	`elasticache.me-south-1.amazonaws.com`	HTTPS
Middle East (UAE) Region `me-central-1`	`elasticache.me-central-1.amazonaws.com`	HTTPS

Region Name/Region	Endpoint	Protocol	
Europe (Ireland) Region eu-west-1	elasticache.eu-west-1.amazonaws.com	HTTPS	
Europe (London) Region eu-west-2	elasticache.eu-west-2.amazonaws.com	HTTPS	
EU (Paris) Region eu-west-3	elasticache.eu-west-3.amazonaws.com	HTTPS	
Europe (Milan) Region eu-south-1	elasticache.eu-south-1.amazonaws.com	HTTPS	
Europe (Spain) Region eu-south-2	elasticache.eu-south-2.amazonaws.com	HTTPS	
South America (São Paulo) Region sa-east-1	elasticache.sa-east-1.amazonaws.com	HTTPS	
China (Beijing) Region cn-north-1	elasticache.cn-north-1.amazonaws.com.cn	HTTPS	
China (Ningxia) Region cn-northwest-1	elasticache.cn-northwest-1.amazonaws.com.cn	HTTPS	

Region Name/Region	**Endpoint**	**Protocol**
Asia Pacific (Hong Kong) Region `ap-east-1`	`elasticache.ap-east-1.amazonaws.com`	HTTPS
Africa (Cape Town) Region `af-south-1`	`elasticache.af-south-1.amazonaws.com`	HTTPS
Israel (Tel Aviv) Region `il-central-1`	`elasticache.il-central-1.amazonaws.com`	HTTPS
AWS GovCloud (US-West) `us-gov-west-1`	`elasticache.us-gov-west-1.amazonaws.com`	HTTPS
AWS GovCloud (US-East) `us-gov-east-1`	`elasticache.us-gov-east-1.amazonaws.com`	HTTPS
For information on using the AWS GovCloud (US) with ElastiCache, see Services in the AWS GovCloud (US) region: ElastiCache.		

Some regions support a subset of node types. For a table of supported node types by AWS Region, see Supported node types by AWS Region.

For a table of AWS products and services by region, see Products and Services by Region.

Locating your nodes

Amazon ElastiCache supports locating all of a cluster's nodes in a single or multiple Availability Zones (AZs). Further, if you elect to locate your nodes in multiple AZs (recommended), ElastiCache enables you to either choose the AZ for each node, or allow ElastiCache to choose them for you.

By locating the nodes in different AZs, you eliminate the chance that a failure, such as a power outage, in one AZ will cause your entire system to fail.

You can specify an AZ for each node when you create a cluster or by adding nodes when you modify an existing cluster. For more information, see the following:

- Creating a cluster
- Modifying an ElastiCache cluster
- Adding nodes to a cluster

Using local zones with ElastiCache

A *Local Zone* is an extension of an AWS Region that is geographically close to your users. You can extend any virtual private cloud (VPC) from a parent AWS Region into a Local Zones by creating a new subnet and assigning it to the Local Zone. When you create a subnet in a Local Zone, your VPC is extended to that Local Zone. The subnet in the Local Zone operates the same as other subnets in your VPC.

By using Local Zones, you can place resources such as an ElastiCache cluster in multiple locations close to your users.

When you create an ElastiCache cluster, you can choose a subnet in a Local Zone. Local Zones have their own connections to the internet and support AWS Direct Connect. Thus, resources created in a Local Zone can serve local users with very low-latency communications. For more information, see AWS Local Zones.

A Local Zone is represented by an AWS Region code followed by an identifier that indicates the location, for example `us-west-2-lax-1a`.

At this time, the available Local Zones are `us-west-2-lax-1a` and `us-west-2-lax-1b`.

The following limitations apply to ElastiCache for Local Zones:

- The following node types are supported by Local Zones at this time:
  - Current generation:

    M5 node types: `cache.m5.large`, `cache.m5.xlarge`, `cache.m5.2xlarge`, `cache.m5.4xlarge`, `cache.m5.12xlarge`, `cache.m5.24xlarge`

    R5 node types: `cache.r5.large`, `cache.r5.xlarge`, `cache.r5.2xlarge`, `cache.r5.4xlarge`, `cache.r5.12xlarge`, `cache.r5.24xlarge`

    T3 node types: `cache.t3.micro`, `cache.t3.small`, `cache.t3.medium`

Enabling a local zone

1. Enable the Local Zone in the Amazon EC2 console.

   For more information, see Enabling Local Zones in the *Amazon EC2 User Guide*.
2. Create a subnet in the Local Zone.

   For more information, see Creating a subnet in your VPC in the *Amazon VPC User Guide*.
3. Create an ElastiCache subnet group in the Local Zone.

   When you create an ElastiCache subnet group, choose the Availability Zone group for the Local Zone.

   For more information, see Creating a subnet group in the *ElastiCache User Guide*.
4. Create an ElastiCache for Memcached cluster that uses the ElastiCache subnet in the Local Zone.

   For more information, see Creating a Memcached cluster (console).

Using Outposts

AWS Outposts is a fully managed service that extends AWS infrastructure, services, APIs, and tools to customer premises. By providing local access to AWS managed infrastructure, AWS Outposts enables customers to build and run applications on premises using the same programming interfaces as in AWS Regions, while using local compute and storage resources for lower latency and local data processing needs. An Outpost is a pool of AWS compute and storage capacity deployed at a customer site. AWS operates, monitors, and manages this capacity as part of an AWS

Region. You can create subnets on your Outpost and specify them when you create AWS resources such as ElastiCache clusters.

Note

In this version, the following limitations apply:

- ElastiCache for Outposts only supports M5 and R5 node families.
- Multi-AZ (cross Outpost replication is not supported).
- ElastiCache for Outposts is not supported in the following regions: cn-north-1, cn-northwest-1 and ap-northeast-3.

Using Outposts with the Memcached console

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. On the navigation pane, choose **Memcached**.
3. In **Cluster Engine**, select **Memcached**.
4. Under **Location**, select **On-Premises - Create your ElastiCache instances on AWS Outposts**.

Configure on-premises options

You can select either an available Outpost to add your cache cluster or, if there are no available Outposts, create a new one using the following steps:

Under On-Premises options:

1. Under **Memcached settings**:

   a. **Name**: Enter a name for the Memcached cluster

   b. **Description**: Enter a description for the Memcached cluster.

   c. **Engine version compatilbility**: Engine version is based on the AWS Outpost region

   d. **Port**: Accept the default port, 11211. If you have a reason to use a different port, type the port number.

   e. **Parameter group**: Use the drop-down to select a default or custom parameter group.

   f. **Node Type**: Available instances are based on Outposts availability. From the drop down list, select **Outposts** and then select an available node type you want to use for this cluster. Then select **Save**.

   g. **Number of nodes**: Enter the number of nodes you want in your cluster.

2. Under **Advanced Memcached settings**:

   a. **Subnet Group**: From the list, select **Create new**.

      - **Name**: Enter a name for the subnet group
      - **Description**: Enter a description for the subnet group
      - **VPC ID**: The VPC ID should match the Outpost VPC.
      - **Availability Zone or Outpost**: Select the Outpost you are using.
      - **Subnet ID**: Select a subnet ID that is available for the Outpost. If there are no subnet IDs available, you need to create them. For more information, see Create a Subnet.

   b. Select **Create**.

Viewing Outpost cluster details

On the Memcached list page, select a cluster that belongs to an AWS Outpost and note the following when viewing the **Cluster details**:

- **Availability Zone**: This will represent the Outpost, using an ARN (Amazon Resource Name) and the AWS Resource Number.
- **Outpost name**: The name of the AWS Outpost.

Using Outposts with the AWS CLI

You can use the AWS Command Line Interface (AWS CLI) to control multiple AWS services from the command line and automate them through scripts. You can use the AWS CLI for ad hoc (one-time) operations.

Downloading and configuring the AWS CLI

The AWS CLI runs on Windows, macOS, or Linux. Use the following procedure to download and configure it.

To download, install, and configure the CLI

1. Download the AWS CLI on the AWS Command Line Interface webpage.
2. Follow the instructions for Installing the AWS CLI and Configuring the AWS CLI in the *AWS Command Line Interface User Guide*.

Using the AWS CLI with Outposts

Use the following CLI operation to create a cache cluster that uses Outposts:

- create-cache-cluster – Using this operation, the `outpost-mode` parameter accepts a value that specifies whether the nodes in the cache cluster are created in a single Outpost or across multiple Outposts.

> **Note**
>
> At this time, only `single-outpost` mode is supported.

```
aws elasticache create-cache-cluster \
   --cache-cluster-id cache cluster id \
   --outpost-mode single-outpost \
```

Designing and managing your own ElastiCache cluster for Memcached implementation

If you need fine-grained control over your ElastiCache cluster, you can choose to design your own cluster. ElastiCache enables you to operate a node-based cluster by choosing the node-type, number of nodes, and node placement across AWS Availability Zones for your cluster. Since ElastiCache is a fully-managed service, it automatically manages hardware provisioning, monitoring, node replacements, and software patching for your cluster.

For information on setting up see Setting up. For details on managing, updating or deleting nodes or clusters, see Managing nodes. For an overview of the major components of an Amazon ElastiCache deployment when you design your own ElastiCache cluster, see these key concepts.

Topics

ElastiCache for Memcached components and features

Following, you can find an overview of the major components of an Amazon ElastiCache for Memcached deployment.

Topics

ElastiCache nodes

A *node* is the smallest building block of an ElastiCache deployment. A node can exist in isolation from or in some relationship to other nodes.

A node is a fixed-size chunk of secure, network-attached RAM. Each node runs an instance of Memcached. If necessary, you can scale the nodes in a cluster up or down to a different instance type. For more information, see Scaling ElastiCache for Memcached.

Every node within a cluster is the same instance type and runs the same cache engine. Each cache node has its own Domain Name Service (DNS) name and port. Multiple types of cache nodes are supported, each with varying amounts of associated memory. For a list of supported node instance types, see Supported node types.

You can purchase nodes on a pay-as-you-go basis, where you only pay for your use of a node. Or you can purchase reserved nodes at a significantly reduced hourly rate. If your usage rate is high, purchasing reserved nodes can save you money. Suppose that your cluster is almost always in use, and you occasionally add nodes to handle use spikes. In this case, you can purchase a number of reserved nodes to run most of the time and purchase pay-as-you-go nodes for the times you occasionally need to add nodes. For more information on reserved nodes, see ElastiCache reserved nodes.

The Memcached engine supports Auto Discovery. *Auto Discovery* is the ability for client programs to automatically identify all of the nodes in a cache cluster, and to initiate and maintain connections to all of these nodes. With Auto Discovery, your application doesn't need to manually connect to individual nodes. Instead, your application connects to a configuration endpoint. The configuration endpoint DNS entry contains the CNAME entries for each of the cache node endpoints. Thus, by connecting to the configuration endpoint, your application immediately has information about all of the nodes in the cluster and can connect to all of them. You don't need to hard-code the individual cache node endpoints in your application. For more information see Auto Discovery.

For more information on nodes, see Managing nodes.

ElastiCache for Memcached clusters

A Memcached *cluster* is a logical grouping of one or more ElastiCache nodes. Data is partitioned across the nodes in a Memcached cluster.

Many ElastiCache operations are targeted at clusters:

- Creating a cluster
- Modifying a cluster
- Deleting a cluster
- Viewing the elements in a cluster
- Adding or removing cost allocation tags to and from a cluster

For more detailed information, see the following related topics:

- Managing clusters and Managing nodes

  Information about clusters, nodes, and related operations.
- AWS service limits: Amazon ElastiCache

  Information about ElastiCache limits, such as the maximum number of nodes or clusters.

  If you need to exceed these limits, make your request using the Amazon ElastiCache cache node request form.
- Mitigating Failures

  Information about improving the fault tolerance of your clusters.

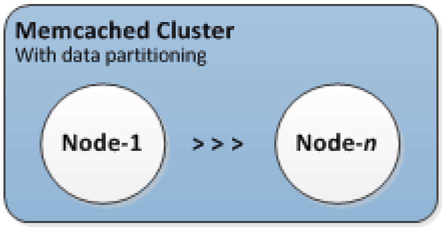
Typical cluster configurations

Memcached supports up to 300 nodes per customer for each AWS Region with each cluster having 1–60 nodes. You partition your data across the nodes in a Memcached cluster.

When you run the Memcached engine, clusters can be made up of 1–60 nodes. You partition your database across the nodes. Your application reads and writes to each node's endpoint. For more information, see Auto Discovery.

For improved fault tolerance, locate your Memcached nodes in various Availability Zones (AZs) within the cluster's AWS Region. That way, a failure in one AZ has minimal impact upon your entire cluster and application. For more information, see Mitigating Failures.

As demand upon your Memcached cluster changes, you can scale out or in by adding or removing nodes, which repartitions your data across the new number of nodes. When you partition your data, we recommend using consistent hashing. For more information about consistent hashing, see Configuring your ElastiCache client for efficient load balancing. In the following diagram, you can see examples of single node and multiple node Memcached clusters.

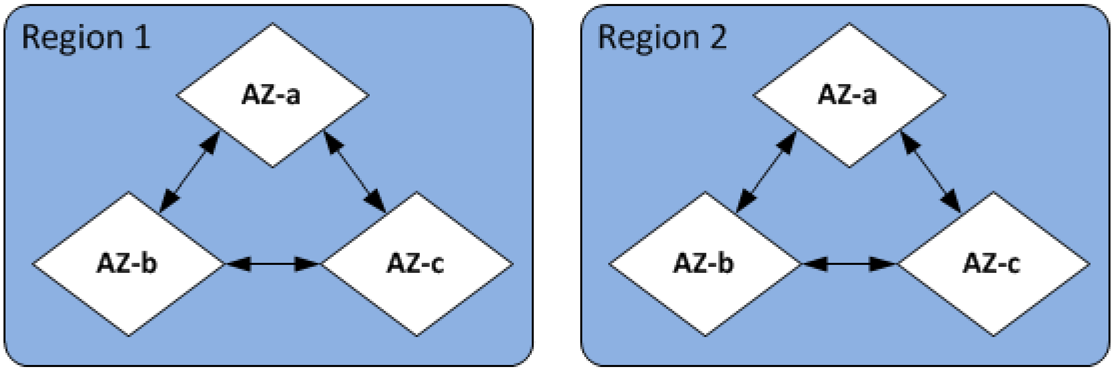


AWS Regions and availability zones

Amazon ElastiCache for Memcached is available in multiple AWS Regions around the world. Thus, you can launch ElastiCache clusters in the locations that meet your business requirements. For example, you can launch in the AWS Region closest to your customers or to meet certain legal requirements.

By default, the AWS SDKs, AWS CLI, ElastiCache API, and ElastiCache console reference the US-West (Oregon) region. As ElastiCache expands availability to new AWS Regions, new endpoints for these AWS Regions are also available to use in your HTTP requests, the AWS SDKs, AWS CLI, and ElastiCache console.

Each AWS Region is designed to be completely isolated from the other AWS Regions. Within each are multiple Availability Zones. By launching your nodes in different Availability Zones, you can achieve the greatest possible fault tolerance. For more information about AWS Regions and Availability Zones, see Choosing regions and availability zones.



For information on AWS Regions supported by ElastiCache and their endpoints, see Supported regions & endpoints.

ElastiCache for Memcached endpoints

An *endpoint* is the unique address your application uses to connect to an ElastiCache node or cluster.

Each node in a Memcached cluster has its own endpoint. The cluster also has an endpoint called the *configuration endpoint*. If you enable Auto Discovery and connect to the configuration endpoint, your application automatically *knows* each node endpoint, even after adding or removing nodes from the cluster. For more information, see Auto Discovery.

For more information, see Endpoints.

ElastiCache parameter groups

Cache parameter groups are an easy way to manage runtime settings for supported engine software. Parameters are used to control memory usage, eviction policies, item sizes, and more. An ElastiCache parameter group is a named collection of engine-specific parameters that you can apply to a cluster. By doing this, you make sure that all of the nodes in that cluster are configured in exactly the same way.

For a list of supported parameters, their default values, and which ones can be modified, see DescribeEngineDefaultParameters (describe-engine-default-parameters).

For more detailed information on ElastiCache parameter groups, see Configuring engine parameters using parameter groups.

ElastiCache security

For enhanced security, ElastiCache node access is restricted to applications running on whitelisted Amazon EC2 instances. You can control the Amazon EC2 instances that can access your cluster by using security groups.

By default, all new ElastiCache clusters are launched in an Amazon Virtual Private Cloud (Amazon VPC) environment. You can use *subnet groups* to grant cluster access from Amazon EC2 instances running on specific subnets. If you choose to run your cluster outside of Amazon VPC, you can create *security groups* to authorize Amazon EC2 instances running within specific Amazon EC2 security groups.

ElastiCache subnet groups

A subnet group is a collection of subnets (typically private) that you can designate for your clusters running in an Amazon Virtual Private Cloud (Amazon VPC) environment.

If you create a cluster in an Amazon VPC, then you must specify a cache subnet group. ElastiCache uses that cache subnet group to choose a subnet and IP addresses within that subnet to associate with your cache nodes.

For more information about cache subnet group usage in an Amazon VPC environment, see Amazon VPCs and ElastiCache security, Authorize access, and Subnets and subnet groups.

ElastiCache for Memcached events

When significant events happen on a cache cluster, ElastiCache sends notification to a specific Amazon SNS topic. Significant events can include such things as a failure to add a node, success in adding a node, the modification of a security group, and others. By monitoring for key events, you can know the current state of your clusters and, depending upon the event, take corrective action.

For more information on ElastiCache events, see Amazon SNS monitoring of ElastiCache events.

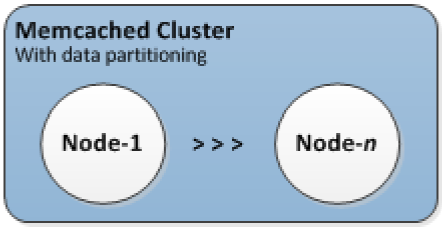
Managing clusters

A *cluster* is a collection of one or more cache nodes, all of which run an instance of the Memcached cache engine software. When you create a cluster, you specify the engine and version for all of the nodes to use.

The following diagram illustrates a typical Memcached cluster. Memcached clusters contain from 1 to 60 nodes across which you horizontally partition your data.

To request a limit increase, see AWS Service Limits and choose the limit type **Nodes per cluster per instance type**.

A typical Memcached cluster looks as follows.



Most ElastiCache operations are performed at the cluster level. You can set up a cluster with a specific number of nodes and a parameter group that controls the properties for each node. All nodes within a cluster are designed to be of the same node type and have the same parameter and security group settings.

Every cluster must have a cluster identifier. The cluster identifier is a customer-supplied name for the cluster. This identifier specifies a particular cluster when interacting with the ElastiCache API and AWS CLI commands. The cluster identifier must be unique for that customer in an AWS Region.

ElastiCache supports multiple engine versions. Unless you have specific reasons, we recommend using the latest version.

ElastiCache clusters are designed to be accessed using an Amazon EC2 instance. If you launch your cluster in a virtual private cloud (VPC) based on the Amazon VPC service, you can access it from outside AWS. For more information, see Access Patterns for Accessing an ElastiCache Cache in an Amazon VPC.

For a list of supported Memcached versions, see Supported ElastiCache for Memcached Versions.

Choosing a network type

ElastiCache supports the Internet Protocol versions 4 and 6 (IPv4 and IPv6), allowing you to configure your cluster to accept:

- only IPv4 connections,
- only IPv6 connections,
- both IPv4 and IPv6 connections (dual-stack)

IPv6 is supported for workloads using Memcached engine version 1.6.6 onward on all instances built on the Nitro system. There are no additional charges for accessing ElastiCache over IPv6.

> **Note**
>
> Migration of clusters created prior to the availability of IPV6 / dual-stack is not supported. Switching between network types on newly created clusters is also not supported.

Configuring subnets for network type

If you create a cluster in an Amazon VPC, you must specify a subnet group. ElastiCache uses that subnet group to choose a subnet and IP addresses within that subnet to associate with your nodes. ElastiCache clusters require a dual-stack subnet with both IPv4 and IPv6 addresses assigned to them to operate in dual-stack mode and an IPv6-only subnet to operate as IPv6-only.

Using dual-stack

When you create a cache cluster and choose dual-stack as the network type, you then need to designate an IP discovery type – either IPv4 or IPv6. ElastiCache will default the network type and IP discovery to IPv6, but that can be changed. If you use Auto Discovery, only the IP addresses of your chosen IP type are returned returned to the Memcached client.

To maintain backwards compatibility with all existing clients, IP discovery is introduced, which allows you to select the IP type (i.e., IPv4 or IPv6) to advertise in the discovery protocol. While this limits auto discovery to only one IP type, dual-stack is still beneficial thanks to Auto Discovery, as it enables migrations (or rollbacks) from an IPv4 to an IPv6 Discovery IP type with no downtime.

TLS enabled dual stack ElastiCache clusters

When TLS is enabled for ElastiCache clusters the cluster discovery functions `config get cluster` return hostnames instead of IPs. The hostnames are then used instead of IPs to connect to the ElastiCache cluster and perform a TLS handshake. This means that clients won't be affected by the IP Discovery parameter. *For TLS enabled clusters the IP Discovery parameter has no effect on the preferred IP protocol.* Instead, the IP protocol used will be determined by which IP protocol the client prefers when resolving DNS hostnames.

For examples on how to configure an IP protocol preference when resolving DNS hostnames, see TLS enabled dual stack ElastiCache clusters.

Using the AWS Management Console

When creating a cache cluster using the AWS Management Console, under **Connectivity**, choose a network type, either **IPv4**, **IPv6** or **Dual stack**. If you choose dual stack, you then must select a **Discovery IP type**, either IPv6 or IPv4.

For more information, see Creating a Memcached cluster (console).

Using the CLI

When creating a cache cluster using the CLI, you use the create-cache-cluster command and specify the `NetworkType` and `IPDiscovery` parameters:

For Linux, macOS, or 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
```

For 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
```

Automatically identify nodes in your cluster

For clusters running the Memcached engine, ElastiCache supports *Auto Discovery*—the ability for client programs to automatically identify all of the nodes in a cache cluster, and to initiate and maintain connections to all of these nodes.

> **Note**
>
> Auto Discovery is added for cache clusters running on Amazon ElastiCache Memcached.

With Auto Discovery, your application does not need to manually connect to individual cache nodes; instead, your application connects to one Memcached node and retrieves the list of nodes. From that list your application is aware of the rest of the nodes in the cluster and can connect to any of them. You do not need to hard code the individual cache node endpoints in your application.

If you are using dual stack network type on your cluster, Auto Discovery will return only IPv4 or IPv6 addresses, depending on which one you select. For more information, see Choosing a network type .

All of the cache nodes in the cluster maintain a list of metadata about all of the other nodes. This metadata is updated whenever nodes are added or removed from the cluster.

Topics

- Benefits of Auto Discovery
- How Auto Discovery Works
- Using Auto Discovery
- Connecting to Cache Nodes Manually
- Adding Auto Discovery To Your Client Library
- ElastiCache clients with auto discovery

Benefits of Auto Discovery

Auto Discovery offers the following benefits:

- When you increase the number of nodes in a cache cluster, the new nodes register themselves with the configuration endpoint and with all of the other nodes. When you remove nodes from the cache cluster, the departing nodes deregister themselves. In both cases, all of the other nodes in the cluster are updated with the latest cache node metadata.
- Cache node failures are automatically detected; failed nodes are automatically replaced.

> **Note**
>
> Until node replacement completes, the node will continue to fail.

- A client program only needs to connect to the configuration endpoint. After that, the Auto Discovery library connects to all of the other nodes in the cluster.
- Client programs poll the cluster once per minute (this interval can be adjusted if necessary). If there are any changes to the cluster configuration, such as new or deleted nodes, the client receives an updated list of metadata. Then the client connects to, or disconnects from, these nodes as needed.

Auto Discovery is enabled on all ElastiCache Memcached cache clusters. You do not need to reboot any of your cache nodes to use this feature.

How Auto Discovery Works

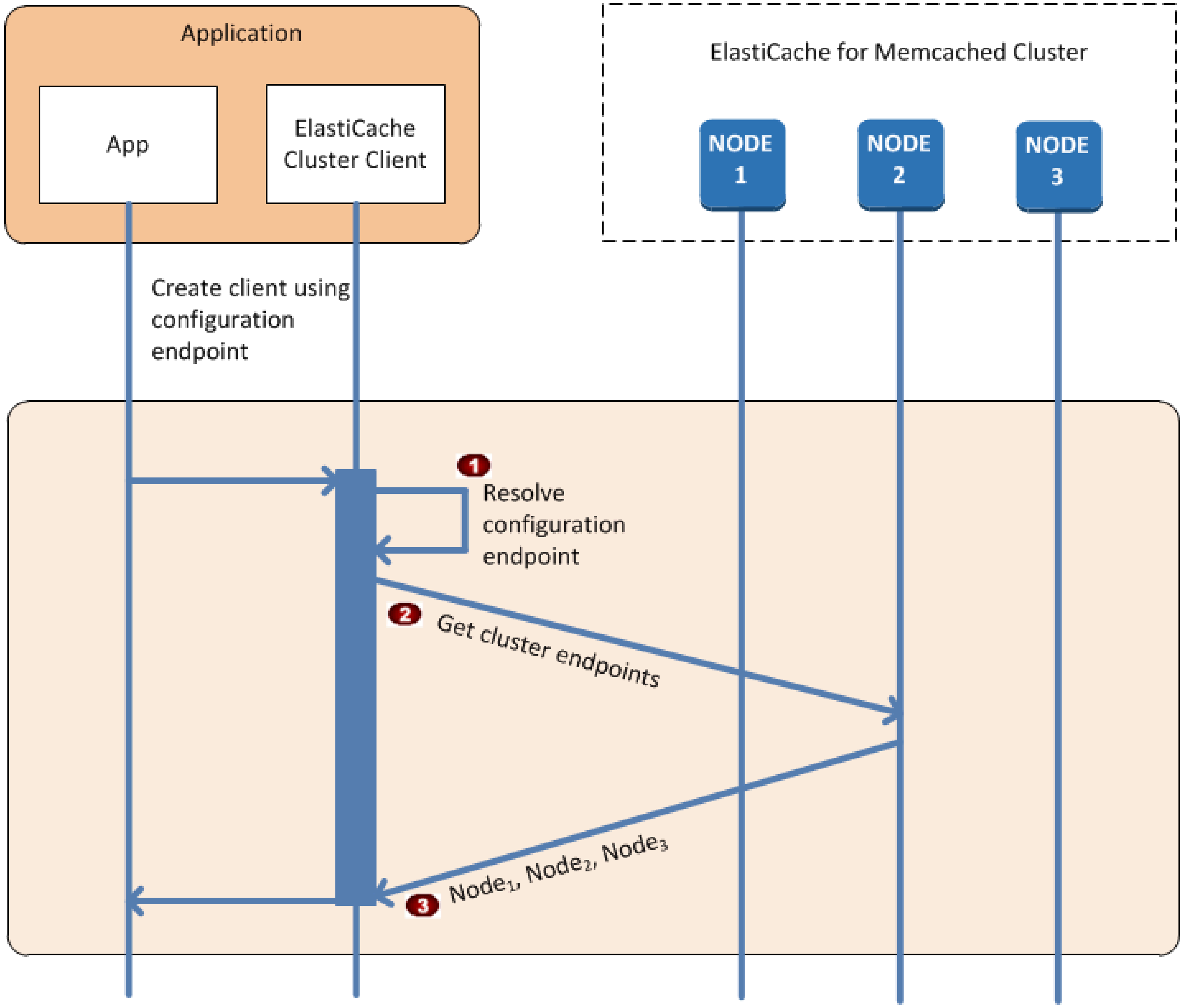
Topics

- Connecting to Cache Nodes
- Normal Cluster Operations
- Other Operations

This section describes how client applications use ElastiCache Cluster Client to manage cache node connections, and interact with data items in the cache.

Connecting to Cache Nodes

From the application's point of view, connecting to the cluster configuration endpoint is no different from connecting directly to an individual cache node. The following sequence diagram shows the process of connecting to cache nodes.


Application
App
ElastiCache Cluster Client
ElastiCache for Memcached Cluster
NODE 1
NODE 2
NODE 3
Create client using configuration endpoint
1
Resolve configuration endpoint
2
Get cluster endpoints
3
Node1, Node2, Node3

Process of Connecting to Cache Nodes

- The application resolves the configuration endpoint's DNS name. Because the configuration endpoint maintains CNAME entries for all of the cache nodes, the DNS name resolves to one of the nodes; the client can then connect to that node.
- The client requests the configuration information for all of the other nodes. Since each node maintains configuration information for all of the nodes in the cluster, any node can pass configuration information to the client upon request.

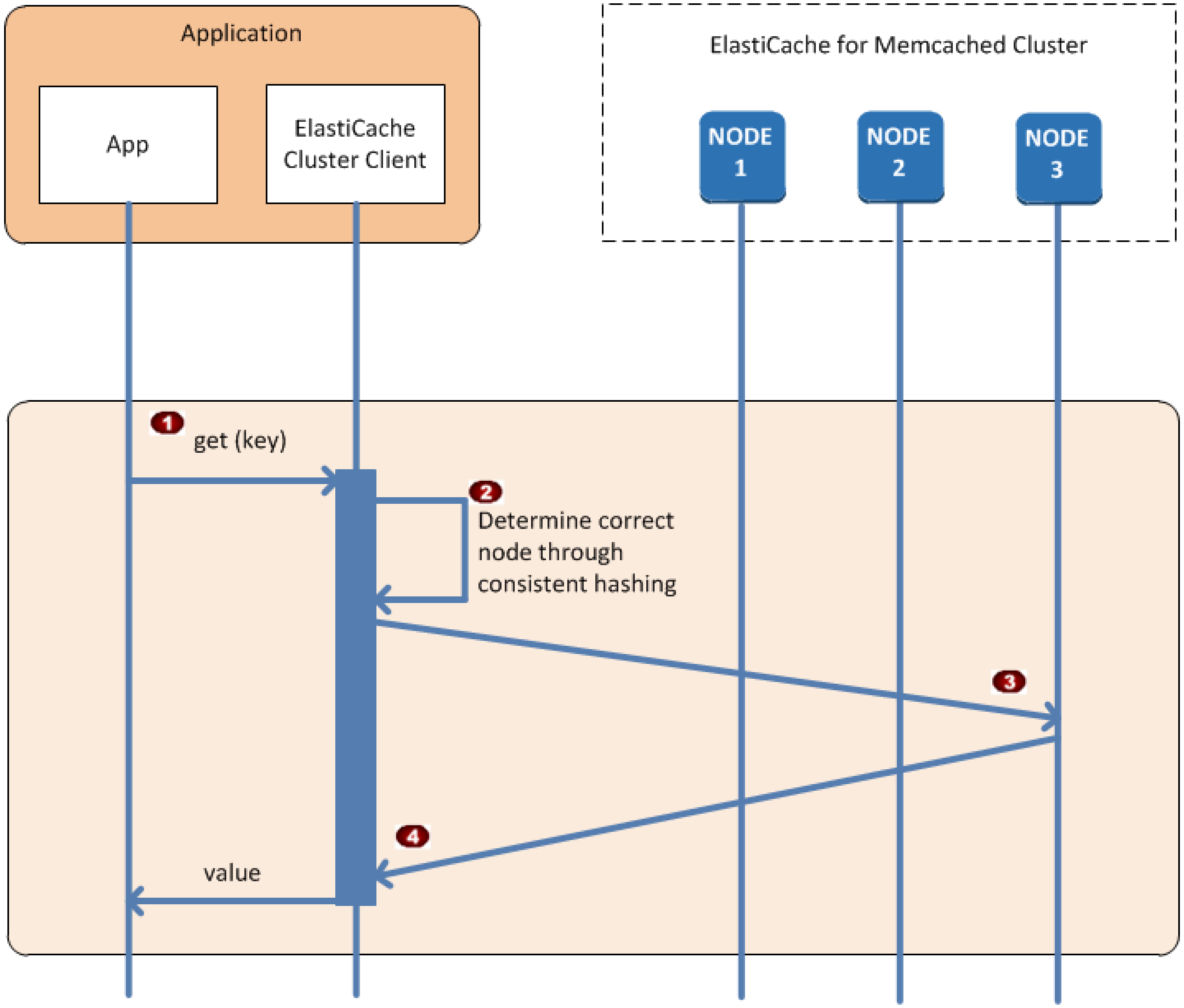
- The client receives the current list of cache node hostnames and IP addresses. It can then connect to all of the other nodes in the cluster.

Note

The client program refreshes its list of cache node hostnames and IP addresses once per minute. This polling interval can be adjusted if necessary.

Normal Cluster Operations

When the application has connected to all of the cache nodes, ElastiCache Cluster Client determines which nodes should store individual data items, and which nodes should be queried for those data items later. The following sequence diagram shows the process of normal cluster operations.


Application
App
ElastiCache Cluster Client
ElastiCache for Memcached Cluster
NODE 1
NODE 2
NODE 3
1
get (key)
2
Determine correct node through consistent hashing
3
4
value

Process of Normal Cluster Operations

- The application issues a *get* request for a particular data item, identified by its key.
- The client uses a hashing algorithm against the key to determine which cache node contains the data item.
- The data item is requested from the appropriate node.
- The data item is returned to the application.

Other Operations

In some situations, you might make a change to a cluster's nodes. For example, you might add an additional node to accommodate additional demand, or delete a node to save money during periods of reduced demand. Or you might replace a node due to a node failure of one sort or another.

When there is a change in the cluster that requires a metadata update to the cluster's endpoints, that change is made to all nodes at the same time. Thus the metadata in any given node is consistent with the metadata in all of the other nodes in the cluster.

In each of these cases, the metadata is consistent among all the nodes at all times since the metadata is updated at the same time for all nodes in the cluster. You should always use the configuration endpoint to obtain the endpoints of the various nodes in the cluster. By using the configuration endpoint, you ensure that you will not be obtaining endpoint data from a node that “disappears” on you.

Adding a Node

During the time that the node is being spun up, its endpoint is not included in the metadata. As soon as the node is available, it is added to the metadata of each of the cluster’s nodes. In this scenario, the metadata is consistent among all the nodes and you will be able to interact with the new node only after it is available. Before the node being available, you will not know about it and will interact with the nodes in your cluster the same as though the new node does not exist.

Deleting a Node

When a node is removed, its endpoint is first removed from the metadata and then the node is removed from the cluster. In this scenario the metadata in all the nodes is consistent and there is no time in which it will contain the endpoint for the node to be removed while the node is not available. During the node removal time it is not reported in the metadata and so your application will only be interacting with the n-1 remaining nodes, as though the node does not exist.

Replacing a Node

If a node fails, ElastiCache takes down that node and spins up a replacement. The replacement process takes a few minutes. During this time the metadata in all the nodes still shows the endpoint for the failed node, but any attempt to interact with the node will fail. Therefore, your logic should always include retry logic.

Using Auto Discovery

To begin using Auto Discovery, follow these steps:

- Step 1: Obtain the Configuration Endpoint
- Step 2: Download the ElastiCache Cluster Client
- Step 3: Modify Your Application Program

Step 1: Obtain the Configuration Endpoint

To connect to a cluster, client programs must know the cluster configuration endpoint. See the topic Finding a Cluster's Endpoints (Console)

You can also use the `aws elasticache describe-cache-clusters` command with the `--show-cache-node-info` parameter:

Whatever method you use to find the cluster's endpoints, the configuration endpoint will always have **.cfg** in its address.

Example Finding endpoints using the AWS CLI for ElastiCache

For Linux, macOS, or Unix:

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

For Windows:

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

This operation produces output similar to the following (JSON format):

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

Step 2: Download the ElastiCache Cluster Client

To take advantage of Auto Discovery, client programs must use the *ElastiCache Cluster Client*. The ElastiCache Cluster Client is available for Java, PHP, and .NET and contains all of the necessary logic for discovering and connecting to all of your cache nodes.

To download the ElastiCache Cluster Client

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. From the ElastiCache console, choose **ElastiCache Cluster Client** then choose **Download**.

The source code for the ElastiCache Cluster Client for Java is available at https://github.com/amazonwebservices/aws-elasticache-cluster-client-memcached-for-java. This library is based on the popular Spymemcached client. The ElastiCache Cluster Client is released under the Amazon Software License https://aws.amazon.com/asl. You are free to modify the source code as you see fit. You can even incorporate the code into other open source Memcached libraries, or into your own client code.

> **Note**
>
> To use the ElastiCache Cluster Client for PHP, you will first need to install it on your Amazon EC2 instance. For more information, see Installing the ElastiCache cluster client for PHP.
> For a TLS supported client download the binary with PHP version 7.4 or higher.
> To use the ElastiCache Cluster Client for .NET, you will first need to install it on your Amazon EC2 instance. For more information, see Installing the ElastiCache cluster client for .NET.

Step 3: Modify Your Application Program

Modify your application program so that it uses Auto Discovery. The following sections show how to use the ElastiCache Cluster Client for Java, PHP, and .NET.

> **Important**
>
> When specifying the cluster's configuration endpoint, be sure that the endpoint has ".cfg" in its address as shown here. Do not use a CNAME or an endpoint without ".cfg" in it.
>
> ```
> "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com";
> ```
>
> Failure to explicitly specify the cluster's configuration endpoint results in configuring to a specific node.

Using the ElastiCache Cluster Client for Java

The program below demonstrates how to use the ElastiCache Cluster Client to connect to a cluster configuration endpoint and add a data item to the cache. Using Auto Discovery, the program connects to all of the nodes in the cluster without any further intervention.

```
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");
    }
```

```
}
```

Using the ElastiCache Cluster Client for PHP

The program below demonstrates how to use the ElastiCache Cluster Client to connect to a cluster configuration endpoint and add a data item to the cache. Using Auto Discovery, the program will connect to all of the nodes in the cluster without any further intervention.

To use the ElastiCache Cluster Client for PHP, you will first need to install it on your Amazon EC2 instance. For more information, see Installing the ElastiCache cluster client for PHP

```
<?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');
  ?>
```

For an example on how to use the ElastiCache Cluster Client with TLS enabled, see Using in transit encryption with PHP and Memcached.

Using the ElastiCache Cluster Client for .NET

> **Note**
>
> The ElastiCache .NET cluster client has been deprecated as of May, 2022.

.NET client for ElastiCache is open source at https://github.com/awslabs/elasticache-cluster-config-net.

.NET applications typically get their configurations from their config file. The following is a sample application config file.

```
<?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>
```

The C# program below demonstrates how to use the ElastiCache Cluster Client to connect to a cluster configuration endpoint and add a data item to the cache. Using Auto Discovery, the program will connect to all of the nodes in the cluster without any further intervention.

```
// *****************
// 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
```

Connecting to Cache Nodes Manually

If your client program does not use Auto Discovery, it can manually connect to each of the cache nodes. This is the default behavior for Memcached clients.

You can obtain a list of cache node hostnames and port numbers from the AWS Management Console. You can also use the AWS CLI `aws elasticache describe-cache-clusters` command with the `--show-cache-node-info` parameter.

Example

The following Java code snippet shows how to connect to all of the nodes in a four-node cache cluster:

```
...

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

If you scale up or scale down your cache cluster by adding or removing nodes, you will need to update the list of nodes in the client code.

Adding Auto Discovery To Your Client Library

The configuration information for Auto Discovery is stored redundantly in each cache cluster node. Client applications can query any cache node and obtain the configuration information for all of the nodes in the cluster.

The way in which an application does this depends upon the cache engine version:

- If the cache engine version is **1.4.14 or higher**, use the `config` command.
- If the cache engine version is **lower than 1.4.14**, use the `get AmazonElastiCache:cluster` command.

The outputs from these two commands are identical, and are described in the Output Format section below.

Cache Engine Version 1.4.14 or Higher

For cache engine version 1.4.14 or higher, use the `config` command. This command has been added to the Memcached ASCII and binary protocols by ElastiCache, and is implemented in the ElastiCache Cluster Client. If you want to use Auto Discovery with another client library, then that library will need to be extended to support the `config` command.

> **Note**
>
> The following documentation pertains to the ASCII protocol; however, the `config` command supports both ASCII and binary. If you want to add Auto Discovery support using the binary protocol, refer to the source code for the ElastiCache Cluster Client.

Syntax

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

Options

Name	Description	Required
`sub-command`	The sub-command used to interact with a cache node. For Auto Discovery, this sub-command is `get`.	Yes

Name	Description	Required
key	The key under which the cluster configuration is stored. For Auto Discovery, this key is named cluster.	Yes

To get the cluster configuration information, use the following command:

```
config get cluster
```

Cache Engine Version Lower Than 1.4.14

To get the cluster configuration information, use the following command:

```
get AmazonElastiCache:cluster
```

> **Note**
>
> Do not tamper with the "AmazonElastiCache:cluster" key, since this is where the cluster configuration information resides. If you do overwrite this key, then the client may be incorrectly configured for a brief period of time (no more than 15 seconds) before ElastiCache automatically and correctly updates the configuration information.

Output Format

Whether you use `config get cluster` or `get AmazonElastiCache:cluster`, the reply consists of two lines:

- The version number of the configuration information. Each time a node is added or removed from the cache cluster, the version number increases by one.
- A list of cache nodes. Each node in the list is represented by a *hostname|ip-address|port* group, and each node is delimited by a space.

A carriage return and a linefeed character (CR + LF) appears at the end of each line. The data line contains a linefeed character (LF) at the end, to which the CR + LF is added. The config version line is terminated by LF without the CR.

A cache cluster containing three nodes would be represented as follows:

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

Each node is shown with both the CNAME and the private IP address. The CNAME will always be present; if the private IP address is not available, it will not be shown; however, the pipe characters "|" will still be printed.

Example

Here is an example of the payload returned when you query the configuration information:

```
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

- The second line indicates that the configuration information has been modified twelve times so far.
- In the third line, the list of nodes is in alphabetical order by hostname. This ordering might be in a different sequence from what you are currently using in your client application.

ElastiCache clients with auto discovery

This section discusses installing and configuring the ElastiCache PHP and .NET clients.

Topics

Installing & compiling cluster clients

This section covers installing, configuring, and compiling the PHP and .NET Amazon ElastiCache auto discovery cluster clients.

Topics

- Installing the ElastiCache cluster client for .NET
- Installing the ElastiCache cluster client for PHP
- Compiling the source code for the ElastiCache cluster client for PHP

Installing the ElastiCache cluster client for .NET

> **Note**
>
> The ElastiCache .NET cluster client has been deprecated as of May, 2022.

You can find the ElastiCache .NET Cluster Client code as open source at https://github.com/awslabs/elasticache-cluster-config-net.

This section describes how to install, update, and remove the .NET components for the ElastiCache Cluster Client on Amazon EC2 instances. For more information about auto discovery, see Auto discovery. For sample .NET code to use the client, see Auto discovery with DotNET.

Topics

- Installing .NET
- Download the ElastiCache .NET cluster client for ElastiCache
- Install AWS assemblies with NuGet

Installing .NET

You must have .NET 3.5 or later installed to use the AWS .NET SDK for ElastiCache. If you don't have .NET 3.5 or later, you can download and install the latest version from http://www.microsoft.com/net.

Download the ElastiCache .NET cluster client for ElastiCache

To download the ElastiCache .NET cluster client

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. On the navigation pane, click **ElastiCache Cluster Client**.
3. In the **Download ElastiCache Memcached Cluster Client** list, select **.NET**, and then click **Download.**

Install AWS assemblies with NuGet

NuGet is a package management system for the .NET platform. NuGet is aware of assembly dependencies and installs all required files automatically. NuGet installed assemblies are stored with your solution, rather than in a central location such as `Program Files`, so you can install versions specific to an application without creating compatibility issues.

Installing NuGet

NuGet can be installed from the Installation Gallery on MSDN; see https://visualstudiogallery.msdn.microsoft.com/27077b70-9dad-4c64-adcf-c7cf6bc9970c. If you are using Visual Studio 2010 or later, NuGet is automatically installed.

You can use NuGet from either **Solution Explorer** or **Package Manager Console**.

Using NuGet from Solution Explorer

To use NuGet from Solution Explorer in Visual Studio 2010

1. From the **Tools** menu, select **Library Package Manager**.
2. Click **Package Manager Console**.

To use NuGet from Solution Explorer in Visual Studio 2012 or Visual Studio 2013

1. From the **Tools** menu, select **NuGet Package Manager**.
2. Click **Package Manager Console**.

From the command line, you can install the assemblies using `Install-Package`, as shown following.

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

To see a page for every package that is available through NuGet, such as the AWSSDK and AWS.Extensions assemblies, see the NuGet website at http://www.nuget.org. The page for each package includes a sample command line for installing the package using the console and a list of the previous versions of the package that are available through NuGet.

For more information on **Package Manager Console** commands, see http://nuget.codeplex.com/wikipage?title=Package%20Manager%20Console%20Command%20Reference%20%28v1.3%29.

Installing the ElastiCache cluster client for PHP

This section describes how to install, update, and remove the PHP components for the ElastiCache Cluster Client on Amazon EC2 instances. For more information about Auto Discovery, see Automatically identify nodes in your cluster. For sample PHP code to use the client. see Using the ElastiCache Cluster Client for PHP.

Topics

- Downloading the installation package
- For users who already have php-memcached extension installed
- Installation steps for new users
- Removing the PHP cluster client

Downloading the installation package

To ensure that you use the correct version of the ElastiCache Cluster Client for PHP, you will need to know what version of PHP is installed on your Amazon EC2 instance. You will also need to know whether your Amazon EC2 instance is running a 64-bit or 32-bit version of Linux.

To determine the PHP version installed on your Amazon EC2 instance

- At the command prompt, run the following command:

  ```
  php -v
  ```

  The PHP version will be shown in the output, as in this example:

  ```
  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**
  >
  > If your PHP and Memcached versions are incompatible, you will get an error message something like the following:
  >
  > ```
  > 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
```

If this happens, you need to compile the module from the source code. For more information, see Compiling the source code for the ElastiCache cluster client for PHP.

To determine your Amazon EC2 AMI architecture (64-bit or 32-bit)

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the **Instances** list, click your Amazon EC2 instance.
3. In the **Description** tab, look for the **AMI:** field. A 64-bit instance should have `x86_64` as part of the description; for a 32-bit instance, look for `i386` or `i686` in this field.

You are now ready to download the ElastiCache Cluster Client.

To download the ElastiCache cluster client for PHP

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. From the ElastiCache console, choose **ElastiCache Cluster Client**.
3. From the **Download ElastiCache Memcached Cluster Client** list, choose the ElastiCache Cluster Client that matches your PHP version and AMI architecture, then choose the **Download** button.

   For a TLS supported client download the binary with PHP version 7.4 or higher.

For users who already have *php-memcached* extension installed

To update the `php-memcached` installation

1. Remove the previous installation of the Memcached extension for PHP as described by the topic Removing the PHP cluster client.
2. Install the new ElastiCache `php-memcached` extension as described previously in Installation steps for new users.

Installation steps for new users

Topics

- Installing PHP 7.x - 8.x for new users
- Installing PHP 5.x for new users

Installing PHP 7.x - 8.x for new users

Topics

- To install PHP 7.x - 8.x on an Amazon Linux 2 AMI
- To install PHP 7.x - 8.x on an Amazon Linux 201609 AMI
- To install PHP 7.x - 8.x on an SUSE Linux 15 AMI
- To install PHP 7.x - 8.x on an Ubuntu 22.04 AMI

To install PHP 7.x - 8.x on an Amazon Linux 2 AMI

> **Note**
>
> If necessary, replace *`PHP-7.x`* with the version you are using.

1. Launch a new instance from the AMI.
2. Run the following command:

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

3. Install PHP 7.x using `amazon-linux-extras`

   With Amazon Linux 2, you can use the *Extras Library* to install application and software updates on your instances. These software updates are known as topics. You can install a specific version of a topic or omit the version information to use the most recent version. For more information, Extras library (Amazon Linux 2).

   To do this, use the following steps;

   a. First, verify if *amazon-linux-extras* is already installed.
   b. If not installed, use the following command to install:

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

c. Confirm that PHP 7.x topic is available in the Amazon Linux 2 machine:

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

d. From the output, review all PHP 7 topics and select the version you want:

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

e. Install PHP packages from the repository. For example:

```
sudo yum clean metadata
```

```
sudo yum install php php-devel
```

4. Download the Amazon ElastiCache Cluster Client.

   - Open the ElastiCache console at https://console.aws.amazon.com/elasticache/.

     Under the ElastiCache dashboard, go to **ElastiCache Cluster Client** and then choose the PHP7 version you want.
   - From command line, replace `PHP-7.X` with the desired PHP version, and replace the `ARCH` with desired architecture (X86 or arm) and for PHP >= 7.4 replace the OpenSSL with the desired OpenSSL version (openssl1.1 or openssl3). If you are using PHP > 7.4, remove OpenSSL suffix.

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

5. Use `tar -zxvf` to extract the downloaded file.

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

6. With root permissions, copy the extracted artifact file `amazon-elasticache-cluster-client.so` into `/usr/lib64/php/modules`.

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

7. Add `extension=amazon-elasticache-cluster-client.so` into file `/etc/php.ini`

8. If you downloaded ElastiCache Cluster Client with PHP 7.4 or higher, install OpenSSL 1.1.x or higher. Installation instructions for OpenSSL 1.1.1:

```
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
```

To install PHP 7.x - 8.x on an Amazon Linux 201609 AMI

> **Note**
>
> If necessary, replace *php7.x* with the version you are using.

1. Launch a new instance from the AMI. For more information, see Step 1: Launch an instance in the *Amazon EC2 User Guide.*

2. Run the following command:

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

3. Install PHP

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

4. Download the Amazon ElastiCache Cluster Client.

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

5. Extract `latest-64bit`.

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

6. With root permission, copy the extracted artifact file `amazon-elasticache-cluster-client.so` into `/usr/lib64/php/7.x/modules/`.

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

7. Create the `50-memcached.ini` file.

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

8. Start or restart your Apache server.

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

To install PHP 7.x - 8.x on an SUSE Linux 15 AMI

Note

If necessary, replace *`php7.x`* with the version you are using.

1. Launch a new instance from the AMI.
2. Run the following command:

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

3. Install PHP

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

   or

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

4. Download the Amazon ElastiCache Cluster Client, replace <ARCH> with desired architecture (X86 or arm). SUSE 15 comes with OpenSSL1.1 built in, so for PHP >= 7.4 choose the client binary with OpenSSL1. If you are using PHP < 7.4, remove OpenSSL suffix.

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

5. Extract `latest-64bit`.

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

6. With root permission, copy the extracted artifact file `amazon-elasticache-cluster-client.so` into `/usr/lib64/php7/extensions/`.

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

7. Insert the line `extension=amazon-elasticache-cluster-client.so` into the file `/etc/php7/cli/php.ini`.

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

8. Start or restart your Apache server.

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

To install PHP 7.x - 8.x on an Ubuntu 22.04 AMI

> **Note**
>
> If necessary, replace *`php7.x`* with the version you are using.

1. Launch a new instance from the AMI.
2. Run the following command:

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

3. Install PHP

   a. Installation instructions for PHP 8.1:

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

   b. Installation instructions for PHP 7.4:

      ```
      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. Download the Amazon ElastiCache Cluster Client, replace <ARCH> with desired architecture (X86 or arm). Ubuntu 22.04 comes with OpenSSL3 built in, so for PHP >= 7.4 choose the client binary with OpenSSL3. If you are using PHP < 7.4, remove OpenSSL suffix.

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

5. Extract latest-64bit.

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

6. With root permission, copy the extracted artifact file `amazon-elasticache-cluster-client.so` into the php extension directory `/usr/lib/php/20190902`. In case this extension dir does not exist, you can find it by running: `php -i | grep extension_dir`
7. Insert the line `extension=amazon-elasticache-cluster-client.so` into the file `/etc/php/7.x/cli/php.ini`.

Installing PHP 5.x for new users

Topics

- To install PHP 5 on an Amazon Linux AMI 2014.03 (64-bit and 32-bit)
- To install PHP 5 on a Red Hat Enterprise Linux 7.0 AMI (64-bit and 32-bit)
- To install PHP 5 on a Ubuntu server 14.04 LTS AMI (64-bit and 32-bit)

- To install PHP 5 for SUSE Linux enterprise server 11 AMI (64-bit or 32-bit)
- Other Linux distributions

To install PHP 5 on an Amazon Linux AMI 2014.03 (64-bit and 32-bit)

1. Launch an Amazon Linux instance (either 64-bit or 32-bit) and log into it.
2. Install PHP dependencies:

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

3. Download the correct `php-memcached` package for your Amazon EC2 instance and PHP version. For more information, see Downloading the installation package.
4. Install `php-memcached`. The URI should be the download path for the installation package:

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

   Here is a sample installation command for PHP 5.4, 64-bit Linux. In this sample, replace *`X.Y.Z`* with the actual version number:

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

   > **Note**
   >
   > Be sure to use the latest version of the install artifact.

5. With root/sudo permission, add a new file named `memcached.ini` in the `/etc/php.d` directory, and insert "extension=amazon-elasticache-cluster-client.so" in the file:

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

6. Start or restart your Apache server.

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

To install PHP 5 on a Red Hat Enterprise Linux 7.0 AMI (64-bit and 32-bit)

1. Launch a Red Hat Enterprise Linux instance (either 64-bit or 32-bit) and log into it.
2. Install PHP dependencies:

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

3. Download the correct `php-memcached` package for your Amazon EC2 instance and PHP version. For more information, see Downloading the installation package.
4. Install `php-memcached`. The URI should be the download path for the installation package:

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

5. With root/sudo permission, add a new file named `memcached.ini` in the `/etc/php.d` directory, and insert `extension=amazon-elasticache-cluster-client.so` in the file.

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

6. Start or restart your Apache server.

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

To install PHP 5 on a Ubuntu server 14.04 LTS AMI (64-bit and 32-bit)

1. Launch an Ubuntu Linux instance (either 64-bit or 32-bit) and log into it.
2. Install PHP dependencies:

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

3. Download the correct `php-memcached` package for your Amazon EC2 instance and PHP version. For more information, see Downloading the installation package.
4. Install `php-memcached`. The URI should be the download path for the installation package.

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

> **Note**
>
> This installation step installs the build artifact `amazon-elasticache-cluster-client.so` into the `/usr/lib/php5/20121212*` directory. Verify the absolute path of the build artifact, because you need it in the next step.

If the previous command doesn't work, you need to manually extract the PHP client artifact `amazon-elasticache-cluster-client.so` from the downloaded `*.tgz` file, and copy it to the `/usr/lib/php5/20121212*` directory.

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

5. With root/sudo permission, add a new file named `memcached.ini` in the `/etc/php5/cli/conf.d` directory, and insert "extension=<absolute path to amazon-elasticache-cluster-client.so>" in the file.

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

6. Start or restart your Apache server.

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

To install PHP 5 for SUSE Linux enterprise server 11 AMI (64-bit or 32-bit)

1. Launch a SUSE Linux instance (either 64-bit or 32-bit) and log into it.
2. Install PHP dependencies:

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

3. Download the correct `php-memcached` package for your Amazon EC2 instance and PHP version. For more information, see Downloading the installation package.
4. Install `php-memcached`. The URI should be the download path for the installation package.

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

5. With root/sudo permission, add a new file named `memcached.ini` in the `/etc/php5/conf.d` directory, and insert **`extension=amazon-elasticache-cluster-client.so`** in the file.

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

6. Start or restart your Apache server.

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

Note

If Step 5 doesn't work for any of the previous platforms, verify the install path for `amazon-elasticache-cluster-client.so`. Also, specify the full path of the binary in the extension. In addition, verify that the PHP in use is a supported version. We support versions 5.3 through 5.5.

Other Linux distributions

On some systems, notably CentOS7 and Red Hat Enterprise Linux (RHEL) 7.1, `libsasl2.so.3` has replaced `libsasl2.so.2`. On those systems, when you load the ElastiCache cluster client, it attempts and fails to find and load `libsasl2.so.2`. To resolve this issue, create a symbolic link to `libsasl2.so.3` so that when the client attempts to load libsasl2.so.2, it is redirected to `libsasl2.so.3`. The following code creates this symbolic link.

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

Removing the PHP cluster client

Topics

- Removing an earlier version of PHP 7 or higher
- Removing an earlier version of PHP 5

Removing an earlier version of PHP 7 or higher

To remove an earlier version of PHP 7 or higher

1. Remove the `amazon-elasticache-cluster-client.so` file from the appropriate PHP lib directory as previously indicated in the installation instructions. See the section for your installation at For users who already have *php-memcached* extension installed.
2. Remove the line `extension=amazon-elasticache-cluster-client.so` from the `php.ini` file.
3. Start or restart your Apache server.

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

Removing an earlier version of PHP 5

To remove an earlier version of PHP 5

1. Remove the `php-memcached` extension:

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

2. Remove the `memcached.ini` file added in the appropriate directory as indicated in the previous installation steps.

Compiling the source code for the ElastiCache cluster client for PHP

This section covers how to obtain and compile the source code for the ElastiCache Cluster Client for PHP.

There are two packages you need to pull from GitHub and compile; aws-elasticache-cluster-client-libmemcached and aws-elasticache-cluster-client-memcached-for-php.

Topics

Compiling the libmemcached library

Prerequesite libraries

- OpenSSL 1.1.0 or higher (unless TLS support is disabled by ./configure --disable-tls).
- SASL (libsasl2, unless SASL support is disabled by `./configure --disable-sasl`).

To compile the aws-elasticache-cluster-client-libmemcached library

1. Launch an Amazon EC2 instance.
2. Install the library dependencies.

   - On Amazon Linux 201509 AMI / Amazon Linux 2 AMI

     ```
     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
     ```

   - On Ubuntu 14.04 AMI (not required for Ubuntu versions that come with OpenSSL >= 1.1)

```
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. Pull the repository and compile the code.

```
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
```

If running `../configure` fails to find `libssl` (OpenSSL library) it may be necessary to tweak the `PKG_CONFIG_PATH` environment variable:

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

Alternately, if you are not using TLS, you can disable it by running:

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

Compiling the ElastiCache Memcached auto discovery client for PHP

The following sections describe how to compile the ElastiCache Memcached Auto Discovery Client

Topics

- Compiling the ElastiCache Memcached client for PHP 7 or higher
- Compiling the ElastiCache Memcached client for PHP 5

Compiling the ElastiCache Memcached client for PHP 7 or higher

Replace PHP-7.x with the version you are using.

Install PHP:

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

Run the following set of commands under the code directory.

```
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
```

If running ../configure fails to find libssl (OpenSSL library) it may be necessary to tweak the `PKG_CONFIG_PATH` environment variable to OpenSSL's .PC file directory:

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

Alternatively, if you are not using TLS, you can disable it by running:

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

Note

You can statically link the libmemcached library into the PHP binary so it can be ported across various Linux platforms. To do that, run the following command before `make`:

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

Compiling the ElastiCache Memcached client for PHP 5

Compile the `aws-elasticache-cluster-client-memcached-for-php` by running the following commands under the `aws-elasticache-cluster-client-memcached-for-php/` folder.

```
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
```

Configuring ElastiCache clients

An ElastiCache cluster is protocol-compliant with Memcached. The code, applications, and most popular tools that you use today with your existing Memcached environment will work seamlessly with the service.

This section discusses specific considerations for connecting to cache nodes in ElastiCache.

Topics

- Finding node endpoints and port numbers
- Connecting for using auto discovery
- DNS names and underlying IP

Finding node endpoints and port numbers

To connect to a cache node, your application needs to know the endpoint and port number for that node.

Finding node endpoints and port numbers (Console)

To determine node endpoints and port numbers

1. Sign in to the Amazon ElastiCache management console and choose the engine running on your cluster.

   A list of all clusters running the chosen engine appears.
2. Continue below for the engine and configuration you're running.
3. Choose the name of the cluster of interest.
4. Locate the **Port** and **Endpoint** columns for the node you're interested in.

Finding cache node endpoints and port numbers (AWS CLI)

To determine cache node endpoints and port numbers, use the command `describe-cache-clusters` with the `--show-cache-node-info` parameter.

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

The fully qualified DNS names and port numbers are in the Endpoint section of the output.

Finding cache node endpoints and port numbers (ElastiCache API)

To determine cache node endpoints and port numbers, use the action `DescribeCacheClusters` with the `ShowCacheNodeInfo=true` parameter.

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
```

Connecting for using auto discovery

If your applications use Auto Discovery, you only need to know the configuration endpoint for the cluster, rather than the individual endpoints for each cache node. For more information, see Automatically identify nodes in your cluster.

> **Note**
>
> At this time, Auto Discovery is only available for cache clusters running Memcached.

DNS names and underlying IP

Clients maintain a server list containing the addresses and ports of the servers holding the cache data. When using ElastiCache, the DescribeCacheClusters API (or the describe-cache-clusters command line utility) returns a fully qualified DNS entry and port number that can be used for the server list.

⚠ **Important**

It is important that client applications are configured to frequently resolve DNS names of cache nodes when they attempt to connect to a cache node endpoint.

VPC Installations

ElastiCache ensures that both the DNS name and the IP address of the cache node remain the same when cache nodes are recovered in case of failure.

Non-VPC Installations

ElastiCache ensures that the DNS name of a cache node is unchanged when cache nodes are recovered in case of failure; however, the underlying IP address of the cache node can change.

Most client libraries support persistent cache node connections by default. We recommend using persistent cache node connections when using ElastiCache. Client-side DNS caching can occur in multiple places, including client libraries, the language runtime, or the client operating system. You should review your application configuration at each layer to ensure that you are frequently resolving IP addresses for your cache nodes.

Preparing a cluster

Following, you can find instructions on creating a cluster using the ElastiCache console, the AWS CLI, or the ElastiCache API.

You can also create an ElastiCache cluster using AWS CloudFormation. For more information, see AWS::ElastiCache::CacheCluster in the *AWS Cloud Formation User Guide*, which includes guidance on how to implement that approach.

Whenever you create a cluster , it is a good idea to do some preparatory work so you won't need to upgrade or make changes right away.

Topics

Determining your requirements

Preparation

Knowing the answers to the following questions helps make creating your cluster go smoother:

Do you want to use ElastiCache serverless or instance based service?

If you want to use serverless cache, simply ensure that you have properly configured your VPC, subnets, and security groups properly. For more details, see Access Patterns for Accessing an ElastiCache Cache in an Amazon VPC. If you want to use instance based ElastiCache, continue reading.

- Which node instance type do you need?

  For guidance on choosing an instance node type, see Choosing your Memcached node size.
- Will you launch your cluster in a virtual private cloud (VPC) based on Amazon VPC?

> ⚠ **Important**
>
> If you're going to launch your cluster in a VPC, make sure to create a subnet group in the same VPC before you start creating a cluster. For more information, see Subnets and subnet groups.
> ElastiCache is designed to be accessed from within AWS using Amazon EC2. However, if you launch in a VPC based on Amazon VPC and your cluster is in an VPC, you can provide access from outside AWS. For more information, see Access Patterns for Accessing an ElastiCache Cache in an Amazon VPC.

- Do you need to customize any parameter values?

  If you do, create a custom parameter group. For more information, see Creating a parameter group.
- Do you need to create your own *VPC security group*?

  For more information, see Security in Your VPC.
- How do you intend to implement fault tolerance?

  For more information, see Mitigating Failures.

Topics

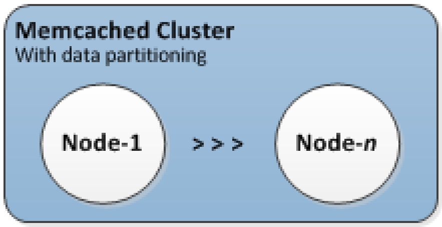
Memory and processor requirements

The basic building block of Amazon ElastiCache is the node. Nodes are configured singularly or in groupings to form clusters. When determining the node type to use for your cluster, take the cluster's node configuration and the amount of data you have to store into consideration.

The Memcached engine is multi-threaded, so a node's number of cores impacts the compute power available to the cluster.

Memcached cluster configuration

ElastiCache for Memcached clusters are comprised of from 1 to 60 nodes. The data in a Memcached cluster is partitioned across the nodes in the cluster. Your application connects with a Memcached cluster using a network address called an Endpoint. Each node in a Memcached cluster has its own endpoint which your application uses to read from or write to the specific node. In addition to the node endpoints, the Memcached cluster itself has an endpoint called the *configuration endpoint*. Your application can use this endpoint to read from or write to the cluster, leaving the determination of which node to read from or write to up to Automatically identify nodes in your cluster.



For more information, see Managing clusters.

Scaling requirements

All clusters can be scaled up by creating a new cluster with the new, larger node type. When you scale up a Memcached cluster. the new cluster starts out empty.

Amazon ElastiCache for Memcached clusters can be scaled out or in. To scale a Memcached cluster out or in you merely add or remove nodes from the cluster. If you have enabled Automatic Discovery and your application is connecting to the cluster's configuration endpoint, you do not need to make any changes in your application when you add or remove nodes.

For more information, see Scaling ElastiCache for Memcached in this guide.

Access requirements

By design, Amazon ElastiCache clusters are accessed from Amazon EC2 instances. Network access to an ElastiCache cluster is limited to the account that created the cluster. Therefore, before you can access a cluster from an Amazon EC2 instance, you must authorize the Amazon EC2 instance to access the cluster. The steps to do this vary, depending upon whether you launched into EC2-VPC or EC2-Classic.

If you launched your cluster into EC2-VPC you need to grant network ingress to the cluster. If you launched your cluster into EC2-Classic you need to grant the Amazon Elastic Compute Cloud security group associated with the instance access to your ElastiCache security group. For detailed instructions, see Accessing your cluster in this guide.

Region, Availability Zone and Local Zone requirements

Amazon ElastiCache supports all AWS regions. By locating your ElastiCache clusters in an AWS Region close to your application you can reduce latency. If your cluster has multiple nodes, locating your nodes in different Availability Zones or in Local Zones can reduce the impact of failures on your cluster.

For more information, see the following:

- Regions and availability zones
- Local zones
- Mitigating Failures

Choosing your node size

The node size you select for your cluster impacts costs, performance, and fault tolerance.

Choosing your Memcached node size

Memcached clusters contain one or more nodes with the cluster's data partitioned across the nodes. Because of this, the memory needs of the cluster and the memory of a node are related, but not the same. You can attain your needed cluster memory capacity by having a few large nodes or several smaller nodes. Further, as your needs change, you can add nodes to or remove nodes from the cluster and thus pay only for what you need.

The total memory capacity of your cluster is calculated by multiplying the number of nodes in the cluster by the RAM capacity of each node after deducting system overhead. The capacity of each node is based on the node type.

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

The number of nodes in the cluster is a key factor in the availability of your cluster running Memcached. The failure of a single node can have an impact on the availability of your application and the load on your backend database. In such a case, ElastiCache provisions a replacement for a failed node and it gets repopulated. To reduce this availability impact, spread your memory and compute capacity over more nodes with smaller capacity, rather than using fewer high-capacity nodes.

In a scenario where you want to have 35 GB of cache memory, you can set up any of the following configurations:

- 11 `cache.t2.medium` nodes with 3.22 GB of memory and 2 threads each = 35.42 GB and 22 threads.
- 6 `cache.m4.large` nodes with 6.42 GB of memory and 2 threads each = 38.52 GB and 12 threads.
- 3 `cache.r4.large` nodes with 12.3 GB of memory and 2 threads each = 36.90 GB and 6 threads.
- 3 `cache.m4.xlarge` nodes with 14.28 GB of memory and 4 threads each = 42.84 GB and 12 threads.

Comparing node options

Node type	Memory (in GiB)	Cores	Hourly cost *	Nodes needed	Total memory (in GiB)	Total cores	Monthly cost
cache.t2.medium	3.22	2	$ 0.068	11	35.42	22	$ 538.56

Node type	Memory (in GiB)	Cores	Hourly cost *	Nodes needed	Total memory (in GiB)	Total cores	Monthly cost
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
* Hourly cost per node as of October 8, 2020.							
Monthly cost at 100% usage for 30 days (720 hours).							

These options each provide similar memory capacity but different computational capacity and cost. To compare the costs of your specific options, see Amazon ElastiCache Pricing.

For clusters running Memcached, some of the available memory on each node is used for connection overhead. For more information, see Memcached connection overhead

Using multiple nodes requires spreading the keys across them. Each node has its own endpoint. For easy endpoint management, you can use the ElastiCache the Auto Discovery feature, which enables client programs to automatically identify all of the nodes in a cluster. For more information, see Automatically identify nodes in your cluster.

In some cases, you might be unsure how much capacity you need. If so, for testing we recommend starting with one `cache.m5.large` node. Then monitor the memory usage, CPU utilization, and cache hit rate with the ElastiCache metrics that are published to Amazon CloudWatch. For more information on CloudWatch metrics for ElastiCache, see Monitoring use with CloudWatch metrics. For production and larger workloads, the R5 nodes provide the best performance and RAM cost value.

If your cluster doesn't have the hit rate that you want, you can easily add more nodes to increase the total available memory in your cluster.

If your cluster is bound by CPU but has sufficient hit rate, set up a new cluster with a node type that provides more compute power.

Creating a cluster

The following examples show how to create a cluster using the AWS Management Console, AWS CLI and ElastiCache API.

Creating a Memcached cluster (console)

When you use the Memcached engine, Amazon ElastiCache supports horizontally partitioning your data over multiple nodes. Memcached enables auto discovery so you don't need to keep track of the endpoints for each node. Memcached tracks each node's endpoint, updating the endpoint list as nodes are added and removed. All your application needs to interact with the cluster is the configuration endpoint. For more information on auto discovery, see Automatically identify nodes in your cluster.

To create a Memcached cluster, follow the steps at Step 1: Create a cache

As soon as your cluster's status is *available*, you can grant Amazon EC2 access to it, connect to it, and begin using it. For more information, see Accessing your cluster and Connecting to Cache Nodes Manually.

⚠ **Important**

As soon as your cluster becomes available, you're billed for each hour or partial hour that the cluster is active, even if you're not actively using it. To stop incurring charges for this cluster, you must delete it. See Deleting a cluster.

Creating a cluster (AWS CLI)

To create a cluster using the AWS CLI, use the `create-cache-cluster` command.

⚠ **Important**

As soon as your cluster becomes available, you're billed for each hour or partial hour that the cluster is active, even if you're not actively using it. To stop incurring charges for this cluster, you must delete it. See Deleting a cluster.

Creating a Memcached Cache Cluster (AWS CLI)

The following CLI code creates a Memcached cache cluster with 3 nodes.

For Linux, macOS, or 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
```

For 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
```

Creating a cluster (ElastiCache API)

To create a cluster using the ElastiCache API, use the `CreateCacheCluster` action.

> ⚠ **Important**
>
> As soon as your cluster becomes available, you're billed for each hour or partial hour that the cluster is active, even if you're not using it. To stop incurring charges for this cluster, you must delete it. See Deleting a cluster.

Creating a Memcached cache cluster (ElastiCache API)

The following code creates a Memcached cluster with 3 nodes (ElastiCache API).

Line breaks are added for ease of reading.

```
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>
```

Viewing a cluster's details

You can view detail information about one or more clusters using the ElastiCache console, AWS CLI, or ElastiCache API.

Viewing a Cluster's Details (Console)

You can view the details of a Memcached cluster using the ElastiCache console, the AWS CLI for ElastiCache, or the ElastiCache API.

The following procedure details how to view the details of a Memcached cluster using the ElastiCache console.

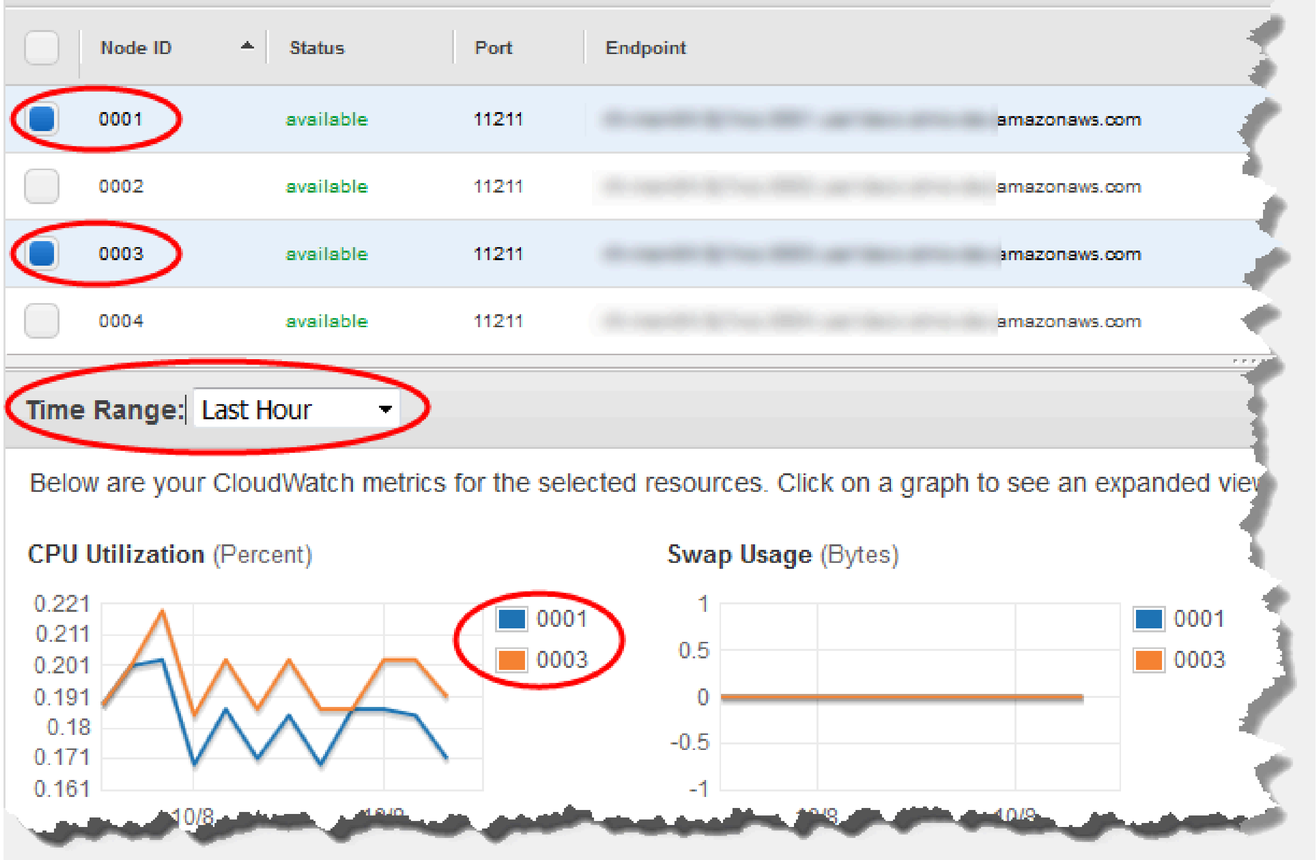
To view a Memcached cluster's details

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. From the list in the upper-right corner, choose the AWS Region you are interested in.
3. In the ElastiCache console dashboard, choose **Memcached**. Doing this displays a list of all your clusters that are running any version of Memcached.
4. To see details of a cluster, choose the box to the left of the cluster's name.
5. To view node information:

   a. Choose the cluster's name.

   b. Choose the **Nodes** tab.

   c. To view metrics on one or more nodes, choose the box to the left of the Node ID, and then choose the time range for the metrics from the **Time range** list. Choosing multiple nodes generates overlay graphs.



□	Node ID	Status	Port	Endpoint
☑	0001	available	11211	[redacted]amazonaws.com
□	0002	available	11211	[redacted]amazonaws.com
☑	0003	available	11211	[redacted]amazonaws.com
□	0004	available	11211	[redacted]amazonaws.com

Time Range: Last Hour

Below are your CloudWatch metrics for the selected resources. Click on a graph to see an expanded viev

Metrics over the last hour for two Memcached nodes

Viewing a cluster's details (AWS CLI)

You can view the details for a cluster using the AWS CLI `describe-cache-clusters` command. If the `--cache-cluster-id` parameter is omitted, details for multiple clusters, up to `--max-items`, are returned. If the `--cache-cluster-id` parameter is included, details for the specified cluster are returned. You can limit the number of records returned with the `--max-items` parameter.

The following code lists the details for `my-cluster`.

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

The following code list the details for up to 25 clusters.

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

Example

For Linux, macOS, or Unix:

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

For Windows:

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

This operation produces output similar to the following (JSON format):

```
{
    "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"
```

```
        }
    ]
}
```

For more information, see the AWS CLI for ElastiCache topic describe-cache-clusters.

Viewing a cluster's details (ElastiCache API)

You can view the details for a cluster using the ElastiCache API `DescribeCacheClusters` action. If the `CacheClusterId` parameter is included, details for the specified cluster are returned. If the `CacheClusterId` parameter is omitted, details for up to `MaxRecords` (default 100) clusters are returned. The value for `MaxRecords` cannot be less than 20 or greater than 100.

The following code lists the details for `my-cluster`.

```
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>
```

The following code list the details for up to 25 clusters.

```
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>
```

For more information, see the ElastiCache API reference topic DescribeCacheClusters.

Modifying an ElastiCache cluster

In addition to adding or removing nodes from a cluster, there can be times where you need to make other changes to an existing cluster, such as, adding a security group, changing the maintenance window or a parameter group.

We recommend that you have your maintenance window fall at the time of lowest usage. Thus it might need modification from time to time.

When you change a cluster's parameters, the change is applied to the cluster either immediately or after the cluster is restarted. This is true whether you change the cluster's parameter group itself or a parameter value within the cluster's parameter group. To determine when a particular parameter change is applied, see the **Changes Take Effect** section of the **Details** column in the tables for Memcached specific parameters . For information on rebooting a cluster, see Rebooting a cluster.

Using the AWS Management Console

To modify a cluster

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. From the list in the upper-right corner, choose the AWS Region where the cluster that you want to modify is located.
3. In the navigation pane, choose the engine running on the cluster that you want to modify.

   A list of the chosen engine's clusters appears.
4. In the list of clusters, for the cluster that you want to modify, choose its name.
5. Choose **Actions** and then choose **Modify**.

   The **Modify Cluster** window appears.
6. In the **Modify Cluster** window, make the modifications that you want. Options include:

   - Engine Version Compatibility
   - VPC Security Group(s)
   - Parameter Group
   - Maintenance Window
   - Topic for SNS Notification

The **Apply Immediately** box applies only to engine version modifications. To apply changes immediately, choose the **Apply Immediately** check box. If this box is not chosen, engine version modifications are applied during the next maintenance window. Other modifications, such as changing the maintenance window, are applied immediately.

7. Choose **Modify**.

Using the AWS CLI

You can modify an existing cluster using the AWS CLI `modify-cache-cluster` operation. To modify a cluster's configuration value, specify the cluster's ID, the parameter to change and the parameter's new value. The following example changes the maintenance window for a cluster named `my-cluster` and applies the change immediately.

> **Important**
>
> You can upgrade to newer engine versions. For more information on doing so, see Engine versions and upgrading . However, you can't downgrade to older engine versions except by deleting the existing cluster and creating it again.

For Linux, macOS, or Unix:

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --preferred-maintenance-window sun:23:00-mon:02:00
```

For Windows:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --preferred-maintenance-window sun:23:00-mon:02:00
```

The `--apply-immediately` parameter applies only to modifications in engine version and changing the number of nodes in a cluster. If you want to apply any of these changes immediately, use the `--apply-immediately` parameter. If you prefer postponing these changes to your next maintenance window, use the `--no-apply-immediately` parameter. Other modifications, such as changing the maintenance window, are applied immediately.

For more information, see the AWS CLI for ElastiCache topic `modify-cache-cluster`.

Using the ElastiCache API

You can modify an existing cluster using the ElastiCache API `ModifyCacheCluster` operation. To modify a cluster's configuration value, specify the cluster's ID, the parameter to change and the parameter's new value. The following example changes the maintenance window for a cluster named `my-cluster` and applies the change immediately.

> **Important**
>
> You can upgrade to newer engine versions. For more information on doing so, see Engine versions and upgrading . However, you can't downgrade to older engine versions except by deleting the existing cluster and creating it again.

Line breaks are added for ease of reading.

```
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>
```

The `ApplyImmediately` parameter applies only to modifications in node type, engine version, and changing the number of nodes in a cluster. If you want to apply any of these changes immediately, set the `ApplyImmediately` parameter to `true`. If you prefer postponing these changes to your next maintenance window, set the `ApplyImmediately` parameter to `false`. Other modifications, such as changing the maintenance window, are applied immediately.

For more information, see the ElastiCache API reference topic `ModifyCacheCluster`.

Rebooting a cluster

Some changes require that the cluster be rebooted for the changes to be applied. For example, for some parameters, changing the parameter value in a parameter group is only applied after a reboot.

When you reboot a cluster, the cluster flushes all its data and restarts its engine. During this process you cannot access the cluster. Because the cluster flushed all its data, when the cluster is available again, you are starting with an empty cluster.

You are able to reboot a cluster using the ElastiCache console, the AWS CLI, or the ElastiCache API. Whether you use the ElastiCache console, the AWS CLI or the ElastiCache API, you can only initiate rebooting a single cluster. To reboot multiple clusters you must iterate on the process or operation.

Using the AWS Management Console

You can reboot a cluster using the ElastiCache console.

To reboot a cluster (console)

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. From the list in the upper-right corner, choose the AWS Region you are interested in.
3. In the navigation pane, choose the engine running on the cluster that you want to reboot.

   A list of clusters running the chosen engine appears.
4. Choose the cluster to reboot by choosing the button to the left of the cluster's name.

   Choose **Actions** and then choose **Reboot**.

   If you choose more than one cluster, the **Reboot** button isn't active.

To reboot multiple clusters, repeat steps 2 through 5 for each cluster that you want to reboot. You do not need to wait for one cluster to finish rebooting to reboot another.

To reboot a specific node, select the node and then choose **Reboot**.

Using the AWS CLI

To reboot a cluster (AWS CLI), use the `reboot-cache-cluster` CLI operation.

To reboot specific nodes in the cluster, use the `--cache-node-ids-to-reboot` to list the specific clusters to reboot. The following command reboots the nodes 0001, 0002, and 0004 of *my-cluster*.

For Linux, macOS, or Unix:

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

For Windows:

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

To reboot all the nodes in the cluster, use the `--cache-node-ids-to-reboot` parameter and list all the cluster's node ids. For more information, see reboot-cache-cluster.

Using the ElastiCache API

To reboot a cluster using the ElastiCache API, use the `RebootCacheCluster` action.

To reboot specific nodes in the cluster, use the `CacheNodeIdsToReboot` to list the specific clusters to reboot. The following command reboots the nodes 0001, 0002, and 0004 of *my-cluster*.

```
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>
```

To reboot all the nodes in the cluster, use the `CacheNodeIdsToReboot` parameter and list all the cluster's node ids. For more information, see RebootCacheCluster.

Adding nodes to a cluster

Adding nodes to a Memcached cluster increases the number of your cluster's partitions. When you change the number of partitions in a cluster some of your key spaces need to be remapped so that they are mapped to the right node. Remapping key spaces temporarily increases the number of cache misses on the cluster. For more information, see Configuring your ElastiCache client for efficient load balancing.

You can use the ElastiCache Management Console, the AWS CLI or ElastiCache API to add nodes to your cluster.

Using the AWS Management Console

Topics

- To add nodes to a cluster (console)

To add nodes to a cluster (console)

The following procedure can be used to add nodes to a cluster.

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the navigation pane, choose the engine running on the cluster that you want to add nodes to.

   A list of clusters running the chosen engine appears.
3. From the list of clusters, for the cluster that you want to add a node to, choose its name.
4. Choose **Add node**.
5. Complete the information requested in the **Add Node** dialog box.
6. Choose the **Apply Immediately - Yes** button to add this node immediately, or choose **No** to add this node during the cluster's next maintenance window.

Impact of New Add and Remove Requests on Pending Requests

Scenarios	Pending Operation	New Request	Results
Scenario 1	Delete	Delete	The new delete request, pending or immediate, replaces the pending delete request. For example, if nodes 0001, 0003, and 0007 are pending deletion and a new request to delete nodes 0002 and 0004 is issued, only nodes 0002 and 0004 will be deleted. Nodes 0001, 0003, and 0007 will not be deleted.
Scenario 2	Delete	Create	The new create request, pending or immediate, replaces the pending delete request. For example, if nodes 0001, 0003, and 0007 are pending deletion and a new request to create a node is issued, a new node will be created and nodes 0001, 0003, and 0007 will not be deleted.
Scenario 3	Create	Delete	The new delete request, pending or immediate, replaces the pending create request. For example, if there is a pending request to create two nodes and a new request is issued to delete node 0003, no new nodes will be created and node 0003 will be deleted.

Scenarios	Pending Operation	New Request	Results
Scenario 4	Create	Create	The new create request is added to the pending create request. For example, if there is a pending request to create two nodes and a new request is issued to create three nodes, the new requests is added to the pending request and five nodes will be created. **Important** If the new create request is set to **Apply Immediately - Yes**, all create requests are performed immediately. If the new create request is set to **Apply Immediately - No**, all create requests are pending.

To determine what operations are pending, choose the **Description** tab and check to see how many pending creations or deletions are shown. You cannot have both pending creations and pending deletions.

7. Choose the **Add** button.

   After a few moments, the new nodes should appear in the nodes list with a status of **creating**. If they don't appear, refresh your browser page. When the node's status changes to *available* the new node is able to be used.

Using the AWS CLI

To add nodes to a cluster using the AWS CLI, use the AWS CLI operation `modify-cache-cluster` with the following parameters:

- `--cache-cluster-id` The ID of the cache cluster that you want to add nodes to.
- `--num-cache-nodes` The `--num-cache-nodes` parameter specifies the number of nodes that you want in this cluster after the modification is applied. To add nodes to this cluster, `--num-`

cache-nodes must be greater than the current number of nodes in this cluster. If this value is less than the current number of nodes, ElastiCache expects the parameter cache-node-ids-to-remove and a list of nodes to remove from the cluster. For more information, see Using the AWS CLI.

- --apply-immediately or --no-apply-immediately which specifies whether to add these nodes immediately or at the next maintenance window.

For Linux, macOS, or Unix:

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

For Windows:

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

This operation produces output similar to the following (JSON format):

```
{
    "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",

    }
}
```

For more information, see the AWS CLI topic `modify-cache-cluster`.

Using the ElastiCache API

To add nodes to a cluster (ElastiCache API)

- Call the `ModifyCacheCluster` API operation with the following parameters:
    - `CacheClusterId` The ID of the cluster that you want to add nodes to.
    - `NumCacheNodes` The `NumCachNodes` parameter specifies the number of nodes that you want in this cluster after the modification is applied. To add nodes to this cluster, `NumCacheNodes` must be greater than the current number of nodes in this cluster. If this value is less than the current number of nodes, ElastiCache expects the parameter `CacheNodeIdsToRemove` with a list of nodes to remove from the cluster (see Using the ElastiCache API).
    - `ApplyImmediately` Specifies whether to add these nodes immediately or at the next maintenance window.
    - `Region` Specifies the AWS Region of the cluster that you want to add nodes to.

  The following example shows a call to add nodes to a cluster.

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>
```

For more information, see ElastiCache API topic `ModifyCacheCluster`.

Removing nodes from a cluster

Each time you change the number of nodes in a Memcached cluster, you must re-map at least some of your keyspace so it maps to the correct node. For more detailed information on load balancing a Memcached cluster, see Configuring your ElastiCache client for efficient load balancing.

You can delete a node from a cluster using the AWS Management Console, the AWS CLI, or the ElastiCache API.

Using the AWS Management Console

To remove nodes from a cluster (console)

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. From the list in the upper-right corner, choose the AWS Region of the cluster that you want to remove nodes from.
3. In the navigation pane, choose the engine running on the cluster that you want to remove a node.

   A list of clusters running the chosen engine appears.
4. From the list of clusters, choose the cluster name from which you want to remove a node.

   A list of the cluster's nodes appears.
5. Choose the box to the left of the node ID for the node that you want to remove. Using the ElastiCache console, you can only delete one node at a time, so choosing multiple nodes means that you can't use the **Delete node** button.

   The *Delete Node* page appears.
6. To delete the node, complete the **Delete Node** page and choose **Delete Node**. To keep the node, choose **Cancel**.

Impact of New Add and Remove Requests on Pending Requests

Scenarios	Pending Operation	New Request	Results
Scenario 1	Delete	Delete	The new delete request, pending or immediate, replaces the pending delete request.

Scenarios	Pending Operation	New Request	Results
			For example, if nodes 0001, 0003, and 0007 are pending deletion and a new request to delete nodes 0002 and 0004 is issued, only nodes 0002 and 0004 will be deleted. Nodes 0001, 0003, and 0007 will not be deleted.
Scenario 2	Delete	Create	The new create request, pending or immediate, replaces the pending delete request. For example, if nodes 0001, 0003, and 0007 are pending deletion and a new request to create a node is issued, a new node will be created and nodes 0001, 0003, and 0007 will not be deleted.
Scenario 3	Create	Delete	The new delete request, pending or immediate, replaces the pending create request. For example, if there is a pending request to create two nodes and a new request is issued to delete node 0003, no new nodes will be created and node 0003 will be deleted.

Scenarios	Pending Operation	New Request	Results
Scenario 4	Create	Create	The new create request is added to the pending create request. For example, if there is a pending request to create two nodes and a new request is issued to create three nodes, the new requests is added to the pending request and five nodes will be created. **Important** If the new create request is set to **Apply Immediately - Yes**, all create requests are performed immediately. If the new create request is set to **Apply Immediately - No**, all create requests are pending.

To determine what operations are pending, choose the **Description** tab and check to see how many pending creations or deletions are shown. You cannot have both pending creations and pending deletions.

Using the AWS CLI

1. Identify the IDs of the nodes that you want to remove. For more information, see Viewing a cluster's details.
2. Use the `modify-cache-cluster` CLI operation with a list of the nodes to remove, as in the following example.

   To remove nodes from a cluster using the command-line interface, use the command `modify-cache-cluster` with the following parameters:

   - `--cache-cluster-id` The ID of the cache cluster that you want to remove nodes from.
   - `--num-cache-nodes` The `--num-cache-nodes` parameter specifies the number of nodes that you want in this cluster after the modification is applied.

- `--cache-node-ids-to-remove` A list of node IDs that you want removed from this cluster.
- `--apply-immediately` or `--no-apply-immediately` Specifies whether to remove these nodes immediately or at the next maintenance window.
- `--region` Specifies the AWS Region of the cluster that you want to remove nodes from.

The following example immediately removes node 0001 from the cluster my-cluster.

For Linux, macOS, or 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
```

For 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
```

This operation produces output similar to the following (JSON format):

```
{
    "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",

    }
}
```

For more information, see the AWS CLI topics `describe-cache-cluster` and `modify-cache-cluster`.

Using the ElastiCache API

To remove nodes using the ElastiCache API, call the `ModifyCacheCluster` API operation with the cache cluster ID and a list of nodes to remove, as shown:

- `CacheClusterId` The ID of the cache cluster that you want to remove nodes from.
- `NumCacheNodes` The `NumCacheNodes` parameter specifies the number of nodes that you want in this cluster after the modification is applied.
- `CacheNodeIdsToRemove.member.n` The list of node IDs to remove from the cluster.

  - CacheNodeIdsToRemove.member.1=0004
  - CacheNodeIdsToRemove.member.1=0005
- ApplyImmediately Specifies whether to remove these nodes immediately or at the next maintenance window.
- Region Specifies the AWS Region of the cluster that you want to remove a node from.

The following example immediately removes nodes 0004 and 0005 from the cluster my-cluster.

```
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>
```

For more information, see ElastiCache API topic ModifyCacheCluster.

Canceling pending add or delete node operations

If you elected to not apply a change immediately, the operation has **pending** status until it is performed at your next maintenance window. You can cancel any pending operation.

To cancel a pending operation

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. From the list in the upper-right corner, choose the AWS Region that you want to cancel a pending add or delete node operation in.
3. In the navigation pane, choose the engine running on the cluster that has pending operations that you want to cancel. A list of clusters running the chosen engine appears.
4. In the list of clusters, choose the name of the cluster, not the box to the left of the cluster's name, that has pending operations that you want to cancel.
5. To determine what operations are pending, choose the **Description** tab and check to see how many pending creations or deletions are shown. You cannot have both pending creations and pending deletions.
6. Choose the **Nodes** tab.
7. To cancel all pending operations, click **Cancel Pending**. The **Cancel Pending** dialog box appears.
8. Confirm that you want to cancel all pending operations by choosing the **Cancel Pending** button, or to keep the operations, choose **Cancel**.

Deleting a cluster

As long as a cluster is in the *available* state, you are being charged for it, whether or not you are actively using it. To stop incurring charges, delete the cluster.

Using the AWS Management Console

The following procedure deletes a single cluster from your deployment. To delete multiple clusters, repeat the procedure for each cluster that you want to delete. You do not need to wait for one cluster to finish deleting before starting the procedure to delete another cluster.

To delete a cluster

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the ElastiCache console dashboard, choose the engine the cluster that you want to delete is running.

   A list of all clusters running that engine appears.
3. To choose the cluster to delete, choose the cluster's name from the list of clusters.

   > ⚠ **Important**
   >
   > You can only delete one cluster at a time from the ElastiCache console. Choosing multiple clusters disables the delete operation.
4. For **Actions**, choose **Delete**.
5. In the **Delete Cluster** confirmation screen, choose **Delete** to delete the cluster, or choose **Cancel** to keep the cluster.

   If you chose **Delete**, the status of the cluster changes to *deleting*.

As soon as your cluster is no longer listed in the list of clusters, you stop incurring charges for it.

Using the AWS CLI

The following code deletes the cache cluster `my-cluster`.

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

The delete-cache-cluster CLI action only deletes one cache cluster. To delete multiple cache clusters, call delete-cache-cluster for each cache cluster that you want to delete. You do not need to wait for one cache cluster to finish deleting before deleting another.

For Linux, macOS, or Unix:

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

For Windows:

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

For more information, see the AWS CLI for ElastiCache topic delete-cache-cluster.

Using the ElastiCache API

The following code deletes the cluster 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>
```

The DeleteCacheCluster API operation only deletes one cache cluster. To delete multiple cache clusters, call DeleteCacheCluster for each cache cluster that you want to delete. You do not need to wait for one cache cluster to finish deleting before deleting another.

For more information, see the ElastiCache API reference topic DeleteCacheCluster.

Accessing your cluster

Your Amazon ElastiCache instances are designed to be accessed through an Amazon EC2 instance.

If you launched your ElastiCache instance in an Amazon Virtual Private Cloud (Amazon VPC), you can access your ElastiCache instance from an Amazon EC2 instance in the same Amazon VPC. Or, by using VPC peering, you can access your ElastiCache instance from an Amazon EC2 in a different Amazon VPC.

If you launched your ElastiCache instance in EC2 Classic, you allow the EC2 instance to access your cluster by granting the Amazon EC2 security group associated with the instance access to your cache security group. By default, access to a cluster is restricted to the account that launched the cluster.

Topics

- Grant access to your cluster

Grant access to your cluster

You launched your cluster into EC2-VPC

If you launched your cluster into an Amazon Virtual Private Cloud (Amazon VPC), you can connect to your ElastiCache cluster only from an Amazon EC2 instance that is running in the same Amazon VPC. In this case, you will need to grant network ingress to the cluster.

> **Note**
>
> If your are using *Local Zones*, make sure you have enabled it. For more information, see Enable Local Zones. By doing so, your VPC is extended to that Local Zone and your VPC will treat the subnet as any subnet in any other Availability Zone and relevant gateways, route tables and other security group considerations. will be automatically adjusted.

To grant network ingress from an Amazon VPC security group to a cluster

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation pane, under **Network & Security**, choose **Security Groups**.

3. From the list of security groups, choose the security group for your Amazon VPC. Unless you created a security group for ElastiCache use, this security group will be named *default*.
4. Choose the **Inbound** tab, and then do the following:

   a. Choose **Edit**.

   b. Choose **Add rule**.

   c. In the **Type** column, choose **Custom TCP rule**.

   d. In the **Port range** box, type the port number for your cluster node. This number must be the same one that you specified when you launched the cluster. The default port for Memcached is **11211** .

   e. In the **Source** box, choose **Anywhere** which has the port range (0.0.0.0/0) so that any Amazon EC2 instance that you launch within your Amazon VPC can connect to your ElastiCache nodes.

   > ⚠ **Important**
   >
   > Opening up the ElastiCache cluster to 0.0.0.0/0 does not expose the cluster to the Internet because it has no public IP address and therefore cannot be accessed from outside the VPC. However, the default security group may be applied to other Amazon EC2 instances in the customer's account, and those instances may have a public IP address. If they happen to be running something on the default port, then that service could be exposed unintentionally. Therefore, we recommend creating a VPC Security Group that will be used exclusively by ElastiCache. For more information, see Custom Security Groups.

   f. Choose **Save**.

When you launch an Amazon EC2 instance into your Amazon VPC, that instance will be able to connect to your ElastiCache cluster.

Accessing ElastiCache resources from outside AWS

Amazon ElastiCache is an AWS service that provides cloud-based in-memory key-value store. The service is designed to be accessed exclusively from within AWS. However, if the ElastiCache cluster is hosted inside a VPC, you can use a Network Address Translation (NAT) instance to provide outside access.

Requirements

The following requirements must be met for you to access your ElastiCache resources from outside AWS:

- The cluster must reside within a VPC and be accessed through a Network Address Translation (NAT) instance. There are no exceptions to this requirement.
- The NAT instance must be launched in the same VPC as the cluster.
- The NAT instance must be launched in a public subnet separate from the cluster.
- An Elastic IP Address (EIP) must be associated with the NAT instance. The port forwarding feature of iptables is used to forward a port on the NAT instance to the cache node port within the VPC.

Considerations

The following considerations should be kept in mind when accessing your ElastiCache resources from outside ElastiCache.

- Clients connect to the EIP and cache port of the NAT instance. Port forwarding on the NAT instance forwards traffic to the appropriate cache cluster node.
- If a cluster node is added or replaced, the iptables rules need to be updated to reflect this change.

Limitations

This approach should be used for testing and development purposes only. It is not recommended for production use due to the following limitations:

- The NAT instance is acting as a proxy between clients and multiple clusters. The addition of a proxy impacts the performance of the cache cluster. The impact increases with number of cache clusters you are accessing through the NAT instance.

- The traffic from clients to the NAT instance is unencrypted. Therefore, you should avoid sending sensitive data via the NAT instance.
- The NAT instance adds the overhead of maintaining another instance.
- The NAT instance serves as a single point of failure. For information about how to set up high availability NAT on VPC, see High Availability for Amazon VPC NAT Instances: An Example.

How to access ElastiCache resources from outside AWS

The following procedure demonstrates how to connect to your ElastiCache resources using a NAT instance.

These steps assume the following:

- `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`

Next you need NAT in the opposite direction:

```
iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to-source 10.0.0.55
```

You also need to enable IP forwarding, which is disabled by default:

```
sudo sed -i 's/net.ipv4.ip_forward=0/net.ipv4.ip_forward=1/g' /etc/
sysctl.conf sudo sysctl --system
```

- You are accessing a Memcached cluster with:
  - IP address – *10.0.1.230*
  - Default Memcached port – *11211*
  - Security group – **10\.0\.0\.55**
- Your trusted client has the IP address *198.51.100.27*.
- Your NAT instance has the Elastic IP Address *203.0.113.73*.
- Your NAT instance has security group *sg-ce56b7a9*.

To connect to your ElastiCache resources using a NAT instance

1. Create a NAT instance in the same VPC as your cache cluster but in a public subnet.

   By default, the VPC wizard will launch a *cache.m1.small* node type. You should select a node size based on your needs. You must use EC2 NAT AMI to be able to access ElastiCache from outside AWS.

   For information about creating a NAT instance, see NAT Instances in the AWS VPC User Guide.

2. Create security group rules for the cache cluster and NAT instance.

   The NAT instance security group and the cluster instance should have the following rules:

   - Two inbound rules
     - One to allow TCP connections from trusted clients to each cache port forwarded from the NAT instance (11211 - 11213).
     - A second to allow SSH access to trusted clients.

   NAT instance security group - inbound rules

Type	Protocol	Port range	Source
Custom TCP Rule	TCP	11211-11213	198.51.100.27/32
SSH	TCP	22	198.51.100.27/32

   - An outbound rule to allow TCP connections to cache port (11211).

   NAT instance security group - outbound rule

Type	Protocol	Port range	Destination
Custom TCP Rule	TCP	11211	sg-ce56b7a9 (Cluster instance Security Group)

   - An inbound rule for the cluster's security group that allows TCP connections from the NAT instance to the cache port (11211).

Cluster instance security group - inbound rule

Type	Protocol	Port range	Source
Custom TCP Rule	TCP	11211	sg-bd56b7da (NAT Security Group)

3. Validate the rules.

   - Confirm that the trusted client is able to SSH to the NAT instance.
   - Confirm that the trusted client is able to connect to the cluster from the NAT instance.

4. Add an iptables rule to the NAT instance.

   An iptables rule must be added to the NAT table for each node in the cluster to forward the cache port from the NAT instance to the cluster node. An example might look like the following:

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

   The port number must be unique for each node in the cluster. For example, if working with a three node Memcached cluster using ports 11211 - 11213, the rules would look like the following:

```
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. Confirm that the trusted client is able to connect to the cluster.

   The trusted client should connect to the EIP associated with the NAT instance and the cluster port corresponding to the appropriate cluster node. For example, the connection string for PHP might look like the following:

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

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

A telnet client can also be used to verify the connection. For example:

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

6. Save the iptables configuration.

   Save the rules after you test and verify them. If you are using a Redhat-based Linux distribution (like Amazon Linux), run the following command:

   ```
   service iptables save
   ```

Related topics

The following topics may be of additional interest.

- Access Patterns for Accessing an ElastiCache Cache in an Amazon VPC
- Accessing an ElastiCache Cache from an Application Running in a Customer's Data Center
- NAT Instances
- Configuring ElastiCache Clients
- High Availability for Amazon VPC NAT Instances: An Example

Finding connection endpoints

Your application connects to your cluster using endpoints. An endpoint is a node or cluster's unique address.

Which endpoints to use

For ElastiCache serverless cache with Memcached, simply acquire the cluster endpoint DNS and port from the console.

From the AWS CLI, use the `describe-serverless-caches` command to acquire the Endpoint information.

Linux

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

Windows

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

The output from the above operation should look something like this (JSON format):

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

For an instance based Memcached cluster, if you use Automatic Discovery then you can use the cluster's *configuration endpoint* to configure your Memcached client. This means you must use a client that supports Automatic Discovery.

If you don't use Automatic Discovery, you must configure your client to use the individual node endpoints for reads and writes. You must also keep track of them as you add and remove nodes.

The following sections guide you through discovering the endpoints you'll need for the engine you're running.

Finding a Cluster's Endpoints (Console)

All Memcached endpoints are read/write endpoints. To connect to nodes in a Memcached cluster your application can use either the endpoints for each node, or the cluster's configuration endpoint along with Automatic Discovery. To use Automatic Discovery you must use a client that supports Automatic Discovery.

When using Automatic Discovery, your client application connects to your Memcached cluster using the configuration endpoint. As you scale your cluster by adding or removing nodes, your application will automatically "know" all the nodes in the cluster and be able to connect to any of them. Without Automatic Discovery your application would have to do this, or you'd have to manually update endpoints in your application each time you added or removed a node.

To copy an endpoint, choose the copy icon directly in front of the endpoint address. For information on using the endpoint to connect to a node, see Connecting to nodes .

Configuration and node endpoints look very similar. The differences are highlighted with **bold** following.

```
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

If you choose to create a CNAME for your Memcached configuration endpoint, in order for your automatic discovery client to recognize the CNAME as a configuration endpoint, you must include `.cfg.` in the CNAME.

Finding Endpoints (AWS CLI)

You can use the AWS CLI for Amazon ElastiCache to discover the endpoints for nodes and clusters.

Topics

- Finding Endpoints for Nodes and Clusters (AWS CLI)

Finding Endpoints for Nodes and Clusters (AWS CLI)

You can use the AWS CLI to discover the endpoints for a cluster and its nodes with the `describe-cache-clusters` command. For Memcached clusters, the command returns the configuration endpoint. If you include the optional parameter `--show-cache-node-info`, the command will also return the endpoints of the individual nodes in the cluster.

Example

The following command retrieves the configuration endpoint (`ConfigurationEndpoint`) and individual node endpoints (`Endpoint`) for the Memcached cluster *mycluster*.

For Linux, macOS, or Unix:

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

For Windows:

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

Output from the above operation should look something like this (JSON format).

```
{
   "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

If you choose to create a CNAME for your Memcached configuration endpoint, in order for your auto discovery client to recognize the CNAME as a configuration endpoint, you must include `.cfg.` in the CNAME. For example, `mycluster`*`.cfg.`*`local` in your php.ini file for the `session.save_path` parameter.

For more information, see the topic describe-cache-clusters.

Finding Endpoints (ElastiCache API)

You can use the Amazon ElastiCache API to discover the endpoints for nodes and clusters.

Topics

- Finding Endpoints for Nodes and Clusters (ElastiCache API)

Finding Endpoints for Nodes and Clusters (ElastiCache API)

You can use the ElastiCache API to discover the endpoints for a cluster and its nodes with the `DescribeCacheClusters` action. For Memcached clusters, the command returns the configuration endpoint. If you include the optional parameter `ShowCacheNodeInfo`, the action also returns the endpoints of the individual nodes in the cluster.

Example

The following command retrieves the configuration endpoint (`ConfigurationEndpoint`) and individual node endpoints (`Endpoint`) for the Memcached cluster *mycluster*.

```
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**
>
> If you choose to create a CNAME for your Memcached configuration endpoint, in order for your auto discovery client to recognize the CNAME as a configuration endpoint, you must include `.cfg.` in the CNAME. For example, `mycluster.cfg.local` in your php.ini file for the `session.save_path` parameter.

Managing nodes

A node is the smallest building block of an Amazon ElastiCache deployment. It is a fixed-size chunk of secure, network-attached RAM. Each node runs the engine that was chosen when the cluster was created or last modified. Each node has its own Domain Name Service (DNS) name and port. Multiple types of ElastiCache nodes are supported, each with varying amounts of associated memory and computational power.

For a more detailed discussion of which node size to use, see Choosing your Memcached node size.

Topics

- Viewing ElastiCache Node Status
- Connecting to nodes
- Supported node types
- Replacing nodes
- ElastiCache reserved nodes
- Migrating previous generation nodes

Some important operations involving nodes are the following:

- Adding nodes to a cluster
- Removing nodes from a cluster
- Scaling ElastiCache for Memcached
- Finding connection endpoints

Viewing ElastiCache Node Status

Using the ElastiCache console, you can quickly access the status of your ElastiCache node. The status of an ElastiCache node indicates the health of the node. You can use the following procedures to view the ElastiCache node status in the Amazon ElastiCache console, the AWS CLI command, or the API operation.

The possible status values for ElastiCache nodes are in the following table. This table also shows if you will be billed for the ElastiCache node.

Type	Billed	Description
`available`	Billed	The ElastiCache node is healthy and available.
`creating`	Not billed	The ElastiCache node is being created. The Node is inaccessible while it is being created.
`deleting`	Not billed	The ElastiCache node is being deleted.
`modifying`	Billed	The ElastiCache node is being modified because of a customer request to modify the node.
`updating`	Billed	An **Updating** state indicates one or more of the following is true of the Amazon ElastiCache node: • The ElastiCache node is being patched as part of the service udpate. For more information on the service updates, refer to the Amazon ElastiCache Managed Maintenance and Service Updates Help Page. • The VPC security groups are updating for the ElastiCache Cluster. • The ElastiCache cluster is being scaled up or scaled down.

Type	Billed	Description
		- The log delivery configurations are being modified for the ElastiCache Cluster. - A delete operation for the ElastiCache node is pending. - The ElastiCache for Redis password is being updated/ rotated using AWS Secrets Manager.
`rebooting cache cluster nodes`	Billed	The ElastiCache node is being rebooted because of a customer request or an Amazon ElastiCache process that requires the rebooting of the node.
`incompatible_parameters`	Not billed	Amazon ElastiCache can't start the node because the parameters specified in the node's parameter group aren't compatible with the node. Either revert the parameter changes or make them compatible with the node to regain access to your node. For more information about the incompatible parameters, check the Events list for the ElastiCache node.

Type	Billed	Description
`incompatible_network`	Not billed	An incompatible-network state indicates one or more of the following is true of the Amazon ElastiCache node: • There are no available IP addresses in the subnet that the ElastiCache node was launched into. • The subnet mentioned in the ElastiCache subnet group no longer exists in the Amazon Virtual Private Cloud (Amazon VPC).

Type	Billed	Description
`restore_failed`	Not billed	A restore-failed state indicates one of the following is true of the Amazon ElastiCache node: • Replacements of node failed due to [Insufficient instance capacity](#) repeatedly. This typically happens when running previous generation nodes that are end-of-life. However, it could also happen with replacement of current generation nodes when AWS does not have enough on-demand capacity to fulfill your request in the specified Availability Zone. For more information on fixing or removing these nodes, see [Migrating previous generation nodes](#). • The specified RDB snapshot failed to restore. • The AWS account for the ElastiCache cluster has been suspended. • The node failed and could not be recovered.
`snapshotting`	Billed	ElastiCache is creating a snapshot of the ElastiCache for Redis node.

Viewing ElastiCache Node Status with the console

To view the status of an ElastiCache Node with the console:

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the navigation pane, choose **Redis Clusters** or **Memcached Clusters**. The **Caches page** appears with the list of ElastiCache Nodes. For each node, the status value is displayed.
3. You can then navigate to the **Service Updates** tab for the cache to display the list of service Updates applicable to the cache.

Viewing ElastiCache Node Status with the AWS CLI

To view ElastiCache node and its status information by using the AWS CLI, use the `describe-cache-cluster` command. For example, the following AWS CLI command displays each ElastiCache node.

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

Viewing ElastiCache Node Status through the API

To view the status of the ElastiCache node using the Amazon ElastiCache API, call the `DescribeCacheClusteroperation` with the `ShowCacheNodeInfo` flag to retrieve information about the individual cache nodes.

Connecting to nodes

Before attempting to connect to your Memcached cluster, you must have the endpoints for the nodes. To find the endpoints, see the following:

- Finding a Cluster's Endpoints (Console)
- Finding Endpoints (AWS CLI)
- Finding Endpoints (ElastiCache API)

In the following example, you use the *telnet* utility to connect to a node that is running Memcached.

> **Note**
>
> For more information about Memcached and available Memcached commands, see the Memcached website.

To connect to a node using *telnet*

1. Connect to your Amazon EC2 instance by using the connection utility of your choice.

   > **Note**
   >
   > For instructions on how to connect to an Amazon EC2 instance, see the Amazon EC2 Getting Started Guide.

2. Download and install the *telnet* utility on your Amazon EC2 instance. At the command prompt of your Amazon EC2 instance, type the following command and type *y* at the command prompt.

   ```
   sudo yum install telnet
   ```

   Output similar to the following appears.

   ```


   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. At the command prompt of your Amazon EC2 instance, type the following command, substituting the endpoint of your node for the one shown in this example.

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

   Output similar to the following appears.

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

4. Test the connection by running Memcached commands.

   You are now connected to a node, and you can run Memcached commands. The following is an example.

   ```
   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
   ```

```
>
```

Supported node types

For information on which node size to use, see Choosing your Memcached node size.

ElastiCache supports the following node types. Generally speaking, the current generation types provide more memory and computational power at lower cost when compared to their equivalent previous generation counterparts.

For more information on performance details for each node type, see Amazon EC2 Instance Types.

- General purpose:
  - Current generation:

    M6g node types (available only for Memcached engine version 1.5.16 onward).

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

    > **Note**
    >
    > For region availability, see Supported node types by AWS Region.

    M5 node types: `cache.m5.large`, `cache.m5.xlarge`, `cache.m5.2xlarge`, `cache.m5.4xlarge`, `cache.m5.12xlarge`, `cache.m5.24xlarge`

    M4 node types: `cache.m4.large`, `cache.m4.xlarge`, `cache.m4.2xlarge`, `cache.m4.4xlarge`, `cache.m4.10xlarge`

    T4g node types (available only for Memcached engine version 1.5.16 onward).

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

    T3 node types: `cache.t3.micro`, `cache.t3.small`, `cache.t3.medium`

    T2 node types: `cache.t2.micro`, `cache.t2.small`, `cache.t2.medium`
  - Previous generation: (not recommended. Existing clusters are still supported but creation of new clusters is not supported for these types.)

    T1 node types: `cache.t1.micro`

M1 node types: `cache.m1.small`, `cache.m1.medium`, `cache.m1.large`, `cache.m1.xlarge`

M3 node types: `cache.m3.medium`, `cache.m3.large`, `cache.m3.xlarge`, `cache.m3.2xlarge`

- Compute optimized:
  - Previous generation: (not recommended)

    C1 node types: `cache.c1.xlarge`
- Memory optimized:
  - Current generation:

    (**R6g node types** are available only for Memcached engine version 1.5.16 onward.)

    R6g node types: `cache.r6g.large`, `cache.r6g.xlarge`, `cache.r6g.2xlarge`, `cache.r6g.4xlarge`, `cache.r6g.8xlarge`, `cache.r6g.12xlarge`, `cache.r6g.16xlarge`

    > **Note**
    >
    > For region availability, see Supported node types by AWS Region.

    R5 node types: `cache.r5.large`, `cache.r5.xlarge`, `cache.r5.2xlarge`, `cache.r5.4xlarge`, `cache.r5.12xlarge`, `cache.r5.24xlarge`

    R4 node types: `cache.r4.large`, `cache.r4.xlarge`, `cache.r4.2xlarge`, `cache.r4.4xlarge`, `cache.r4.8xlarge`, `cache.r4.16xlarge`
  - Previous generation: (not recommended)

    M2 node types: `cache.m2.xlarge`, `cache.m2.2xlarge`, `cache.m2.4xlarge`

    R3 node types: `cache.r3.large`, `cache.r3.xlarge`, `cache.r3.2xlarge`, `cache.r3.4xlarge`, `cache.r3.8xlarge`
- Network optimized:
  - Current generation:

(**C7gn node types** are available only for Memcached engine version 1.6.6 onward.)

C7gn node types: `cache.c7gn.large`, `cache.c7gn.xlarge`, `cache.c7gn.2xlarge`, `cache.c7gn.4xlarge`, `cache.c7gn.8xlarge`, `cache.c7gn.12xlarge`, `cache.c7gn.16xlarge`

Current Generation

The following tables show the baseline and burst bandwidth for instance types that use the network I/O credit mechanism to burst beyond their baseline bandwidth.

> **Note**
>
> Instance types with burstable network performance use a network I/O credit mechanism to burst beyond their baseline bandwidth on a best-effort basis.

General

Instance type	Minimum supported Memcached version	Baseline bandwidth (Gbps)	Burst bandwidth (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	N/A
cache.m7g.12xlarge		22.5	N/A
cache.m7g.16xlarge		30	N/A
cache.m6g.large	1.5.16	0.75	10.0
cache.m6g.xlarge	1.5.16	1.25	10.0

Instance type	Minimum supported Memcached version	Baseline bandwidth (Gbps)	Burst bandwidth (Gbps)
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	N/A
cache.m6g.12xlarge	1.5.16	20	N/A
cache.m6g.16xlarge	1.5.16	25	N/A
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	N/A	N/A
cache.m5.24xlarge	1.5.16	N/A	N/A
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

Instance type	Minimum supported Memcached version	Baseline bandwidth (Gbps)	Burst bandwidth (Gbps)
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
cache.t2.medium	1.5.16	0.256	1.024

Memory optimized

Instance type	Minimum supported version	Baseline bandwidth (Gbps)	Burst bandwidth (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	N/A
cache.r7g.12xlarge		22.5	N/A
cache.r7g.16xlarge		30	N/A
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

Instance type	Minimum supported version	Baseline bandwidth (Gbps)	Burst bandwidth (Gbps)
cache.r6g.8xlarge	1.5.16	12	N/A
cache.r6g.12xlarge	1.5.16	20	N/A
cache.r6g.16xlarge	1.5.16	25	N/A
cache.r5.large	1.5.16	0.75	10.0
cache.r5.xlarge	1.5.16	1.25	10.0
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	N/A
cache.r5.24xlarge	1.5.16	25	N/A
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	N/A
cache.r4.16xlarge	1.5.16	25	N/A

Network optimized

Instance type	Minimum supported version	Baseline bandwidth (Gbps)	Burst bandwidth (Gbps)
cache.c7gn.large	1.6.6	6.25	30

Instance type	Minimum supported version	Baseline bandwidth (Gbps)	Burst bandwidth (Gbps)
cache.c7gn.xlarge	1.6.6	12.5	40
cache.c7gn.2xlarge	1.6.6	25	50
cache.c7gn.4xlarge	1.6.6	50	N/A
cache.c7gn.8xlarge	1.6.6	100	N/A
cache.c7gn.12xlarge	1.6.6	150	N/A
cache.c7gn.16xlarge	1.6.6	200	N/A

Supported node types by AWS Region

Supported node types may vary between AWS Regions. For more details, see Amazon ElastiCache pricing.

Burstable Performance Instances

You can launch general-purpose burstable T4g, T3-Standard and T2-Standard cache nodes in Amazon ElastiCache. These nodes provide a baseline level of CPU performance with the ability to burst CPU usage at any time until the accrued credits are exhausted. A *CPU credit* provides the performance of a full CPU core for one minute.

Amazon ElastiCache's T4g, T3 and T2 nodes are configured as standard and suited for workloads with an average CPU utilization that is consistently below the baseline performance of the instance. To burst above the baseline, the node spends credits that it has accrued in its CPU credit balance. If the node is running low on accrued credits, performance is gradually lowered to the baseline performance level. This gradual lowering ensures the node doesn't experience a sharp performance drop-off when its accrued CPU credit balance is depleted. For more information, see CPU Credits and Baseline Performance for Burstable Performance Instances in the *Amazon EC2 User Guide*.

The following table lists the burstable performance node types, the rate at which CPU credits are earned per hour. It also shows the maximum number of earned CPU credits that a node can accrue and the number of vCPUs per node. In addition, it gives the baseline performance level as a percentage of a full core performance (using a single vCPU).

CPU credits earned per hour	Maximum earned credits that can be accrued*	vCPUs	Baseline performance per vCPU	Memory (GiB)	Network performance
12	288	2	10%	0.5	Up to 5 Gigabit
24	576	2	20%	1.37	Up to 5 Gigabit
24	576	2	20%	3.09	Up to 5 Gigabit
12	288	2	10%	0.5	Up to 5 Gigabit
24	576	2	20%	1.37	Up to 5 Gigabit
24	576	2	20%	3.09	Up to 5 Gigabit
6	144	1	10%	0.5	Low to moderate
12	288	1	20%	1.55	Low to moderate
24	576	2	20%	3.22	Low to moderate

* The number of credits that can be accrued is equivalent to the number of credits that can be earned in a 24-hour period.

** The baseline performance in the table is per vCPU. Some node sizes that have more than one vCPU. For these, calculate the baseline CPU utilization for the node by multiplying the vCPU percentage by the number of vCPUs.

The following CPU credit metrics are available for T3 and T4g burstable performance instances:

> **Note**
>
> These metrics are not available for T2 burstable performance instances.

- `CPUCreditUsage`
- `CPUCreditBalance`

For more information on these metrics, see CPU Credit Metrics.

In addition, be aware of these details:

- All current generation node types are created in a virtual private cloud (VPC) based on Amazon VPC by default.

Related Information

- Amazon ElastiCache Product Features and Details
- Memcached Node-Type Specific Parameters

Replacing nodes

Amazon ElastiCache for Memcached frequently upgrades its fleet with patches and upgrades being applied to instances seamlessly. However, from time to time we need to relaunch your ElastiCache for Memcached nodes to apply mandatory OS updates to the underlying host. These replacements are required to apply upgrades that strengthen security, reliability, and operational performance.

You have the option to manage these replacements yourself at any time before the scheduled node replacement window. When you manage a replacement yourself, your instance receives the OS update when you relaunch the node and your scheduled node replacement is canceled. You might continue to receive alerts indicating that the node replacement takes place. If you already manually mitigated the need for the maintenance, you can ignore these alerts.

> **Note**
>
> Replacement cache nodes automatically generated by Amazon ElastiCache may have different IP addresses. You are responsible for reviewing your application configuration to ensure that your cache nodes are associated with the appropriate IP addresses.

The following list identifies actions you can take when ElastiCache schedules one of your Memcached nodes for replacement.

- **Do nothing** – If you do nothing, ElastiCache replaces the node as scheduled. When ElastiCache automatically replaces the node with a new node, the new node is initially empty.
- **Change your maintenance window** – For scheduled maintenance events, you receive an email or a notification event from ElastiCache. In this case, if you change your maintenance window before the scheduled replacement time, your node now is replaced at the new time. For more information, see Modifying an ElastiCache cluster.

  > **Note**
  >
  > The ability to change your replacement window by moving your maintenance window is only available when the ElastiCache notification includes a maintenance window. If the notification does not include a maintenance window, you cannot change your replacement window.

For example, let's say it's Thursday, November 9, at 15:00 and the next maintenance window is Friday, November 10, at 17:00. Following are three scenarios with their outcomes:

- You change your maintenance window to Fridays at 16:00, after the current date and time and before the next scheduled maintenance window. The node is replaced on Friday, November 10, at 16:00.
- You change your maintenance window to Saturday at 16:00, after the current date and time and after the next scheduled maintenance window. The node is replaced on Saturday, November 11, at 16:00.
- You change your maintenance window to Wednesday at 16:00, earlier in the week than the current date and time). The node is replaced next Wednesday, November 15, at 16:00.

For instructions, see Managing maintenance.

- **Manually replace the node –** If you need to replace the node before the next maintenance window, manually replace the node.

  If you manually replace the node, keys are redistributed. This redistribution causes cache misses.

  To manually replace a Memcached node

  1. Delete the node scheduled for replacement. For instructions, see Removing nodes from a cluster.
  2. Add a new node to the cluster. For instructions, see Adding nodes to a cluster.
  3. If you are not using auto discovery on this cluster, see your application and replace every instance of the old node's endpoint with the new node's endpoint.

ElastiCache reserved nodes

Reserving one or more nodes might be a way for you to reduce costs. Reserved nodes are charged an up front fee that depends upon the node type and the length of reservation— one or three years.

To see if reserved nodes are a cost savings for your use cases, first determine the node size and number of nodes you need. Then estimate the usage of the node, and compare the total cost to you of using On-Demand nodes versus reserved nodes. You can mix and match reserved and On-Demand node usage in your clusters. For pricing information, see Amazon ElastiCache Pricing.

> **Note**
>
> Reserved nodes are not flexible; they only apply to the exact instance type that you reserve.

Managing costs with reserved nodes

Reserving one or more nodes may be a way for you to reduce costs. Reserved nodes are charged an up front fee that depends upon the node type and the length of reservation—one or three years. This charge is much less than the hourly usage charge that you incur with On-Demand nodes.

To see if reserved nodes are a cost savings for your use cases, first determine the node size and number of nodes you need. Then estimate the usage of the node, and compare the total cost to you using On-Demand nodes versus reserved nodes. You can mix and match reserved and On-Demand node usage in your clusters. For pricing information, see Amazon ElastiCache Pricing.

AWS Region, node type and term length must be chosen at purchase, and cannot be changed later.

You can use the AWS Management Console, the AWS CLI, or the ElastiCache API to list and purchase available reserved node offerings.

For more information on reserved nodes, see Amazon ElastiCache Reserved Nodes.

Topics

- Purchasing a reserved node
- Getting info about your reserved nodes

Standard reserved node offerings

When you purchase a standard reserved node instance (RI) in Amazon ElastiCache, you purchase a commitment to getting a discounted rate on a specific node instance type and AWS Region for the duration of the reserved node instance. To use an Amazon ElastiCache reserved node instance, you create a new ElastiCache node instance, just as you would for an on-demand instance.

The new node instance that you create must exactly match the specifications of the reserved node instance. If the specifications of the new node instance match an existing reserved node instance for your account, you are billed at the discounted rate offered for the reserved node instance. Otherwise, the node instance is billed at an on-demand rate. These standard RIs are available from R5 and M5 instance families onwards.

> **Note**
>
> All three offering types discussed next are available in one-year and three-year terms.

Offering Types

No Upfront RI provides access to a reserved ElastiCache instance without requiring an upfront payment. Your *No Upfront* reserved ElastiCache instance bills a discounted hourly rate for every hour within the term, regardless of usage.

Partial Upfront RI requires a part of the reserved ElasticCache instance to be paid upfront. The remaining hours in the term are billed at a discounted hourly rate, regardless of usage. This option is the replacement for the legacy *Heavy Utilization* option, which is explained in the next section.

All Upfront RI requires full payment to be made at the start of the RI term. You incur no other costs for the remainder of the term, regardless of the number of hours used.

Legacy reserved node offerings

There are three levels of legacy node reservations—Heavy Utilization, Medium Utilization, and Light Utilization. Nodes can be reserved at any utilization level for either one or three years. The node type, utilization level, and reservation term affect your total costs. Verify the savings that reserved nodes can provide your business by comparing various models before you purchase reserved nodes.

Nodes purchased at one utilization level or term cannot be converted to a different utilization level or term.

Utilization Levels

Heavy Utilization reserved nodes enable workloads that have a consistent baseline of capacity or run steady-state workloads. Heavy Utilization reserved nodes require a high up-front commitment, but if you plan to run more than 79 percent of the reserved node term you can earn the largest savings (up to 70 percent off of the On-Demand price). With Heavy Utilization reserved nodes, you pay a one-time fee. This is then followed by a lower hourly fee for the duration of the term regardless of whether your node is running.

Medium Utilization reserved nodes are the best option if you plan to use your reserved nodes a large amount of the time and you want either a lower one-time fee or to stop paying for your node when you shut it off. Medium Utilization reserved nodes are a more cost-effective option when you plan to run more than 40 percent of the reserved nodes term. This option can save you up to 64 percent off of the On-Demand price. With Medium Utilization reserved nodes, you pay a slightly higher one-time fee than with Light Utilization reserved nodes, and you receive lower hourly usage rates when you run a node.

Light Utilization reserved nodes are ideal for periodic workloads that run only a couple of hours a day or a few days per week. Using Light Utilization reserved nodes, you pay a one-time fee followed by a discounted hourly usage fee when your node is running. You can start saving when your node is running more than 17 percent of the reserved node term. You can save up to 56 percent off of the On-Demand rates over the entire term of your reserved node.

Legacy reserved node offerings

Offering	Up-front cost	Usage fee	Advantage
Heavy Utilization	Highest	Lowest hourly fee. Applied to the whole term whether or not you're using the reserved node.	Lowest overall cost if you plan to run your reserved nodes more than 79 percent of a three-year term.

Offering	Up-front cost	Usage fee	Advantage
Medium Utilization	Medium	Hourly usage fee charged for each hour the node is running. No hourly charge when the node is not running.	Suitable for elastic workloads or when you expect moderate usage, more than 40 percent of a three-year term.
Light Utilization	Lowest	Hourly usage fee charged for each hour the node is running. No hourly charge when the node is not running. Highest hourly fees of all the offering types, but fees apply only when the reserved node is running.	Highest overall cost if you plan to run all of the time. However, this is the lowest overall cost if you plan to use your reserved node infrequently, more than about 15 percent of a three-year term.
On-Demand Use (No reserved nodes)	None	Highest hourly fee. Applied whenever the node is running.	Highest hourly cost.

For more information, see Amazon ElastiCache Pricing.

Getting info about reserved node offerings

Before you purchase reserved nodes, you can get information about available reserved node offerings.

The following examples show how to get pricing and information about available reserved node offerings using the AWS Management Console, AWS CLI, and ElastiCache API.

Topics

Getting info about reserved node offerings (Console)

To get pricing and other information about available reserved cluster offerings using the AWS Management Console, use the following procedure.

To get information about available reserved node offerings

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the navigation pane, choose **Reserved Nodes**.
3. Choose **Purchase Reserved Node**.
4. For **Engine**, choose Memcached.
5. To determine the available offerings, make selections for the following options:

    - **Node Type**
    - **Term**
    - **Offering Type**

    After you make these selections, the cost per node and total cost of your selections is shown under **Reservation details**.
6. Choose **Cancel** to avoid purchasing these nodes and incurring charges.

Getting info about reserved node offerings (AWS CLI)

To get pricing and other information about available reserved node offerings, type the following command at a command prompt:

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

This operation produces output similar to the following (JSON format):

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

For more information, see describe-reserved-cache-nodes-offerings in the AWS CLI Reference.

Getting info about reserved node offerings (ElastiCache API)

To get pricing and information about available reserved node offerings, call the `DescribeReservedCacheNodesOfferings` action.

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>
```

For more information, see DescribeReservedCacheNodesOfferings in the ElastiCache API Reference.

Purchasing a reserved node

The following examples show how to purchase a reserved node offering using the AWS Management Console, the AWS CLI, and the ElastiCache API.

> ⚠ **Important**
>
> Following the examples in this section incurs charges on your AWS account that you can't reverse.

Topics

Purchasing a reserved node (Console)

This example shows purchasing a specific reserved node offering, *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f*, with a reserved node ID of *myreservationID*.

The following procedure uses the AWS Management Console to purchase the reserved node offering by offering id.

To purchase reserved nodes

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the navigation list, choose the **Reserved Nodes** link.
3. Choose the **Purchase reserved nodes** button.
4. For **Engine**, choose Memcached.
5. To determine the available offerings, make selections for the following options:

   - **Node Type**
   - **Term**
   - **Offering Type**
   - An optional **Reserved node ID**

After you make these selections, the cost per node and total cost of your selections is shown under **Reservation details**.

6. Choose **Purchase**.

Purchasing a reserved node (AWS CLI)

The following example shows purchasing the specific reserved cluster offering, *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f*, with a reserved node ID of *myreservationID*.

Type the following command at a command prompt:

For Linux, macOS, or Unix:

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

For Windows:

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

The command returns output similar to the following:

```
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
```

For more information, see purchase-reserved-cache-nodes-offering in the AWS CLI Reference.

Purchasing a reserved node (ElastiCache API)

The following example shows purchasing the specific reserved node offering, *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f*, with a reserved cluster ID of *myreservationID*.

Call the `PurchaseReservedCacheNodesOffering` operation with the following parameters:

- `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>
```

For more information, see PurchaseReservedCacheNodesOffering in the ElastiCache API Reference.

Getting info about your reserved nodes

You can get information about the reserved nodes you've purchased using the AWS Management Console, the AWS CLI, and the ElastiCache API.

Topics

- Getting info about your reserved nodes (Console)
- Getting info about your reserved nodes (AWS CLI)
- Getting info about your reserved nodes (ElastiCache API)

Getting info about your reserved nodes (Console)

The following procedure describes how to use the AWS Management Console to get information about the reserved nodes you purchased.

To get information about your purchased reserved nodes

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the navigation list, choose the **Reserved nodes** link.

   The reserved nodes for your account appear in the Reserved nodes list. You can choose any of the reserved nodes in the list to see detailed information about the reserved node in the detail pane at the bottom of the console.

Getting info about your reserved nodes (AWS CLI)

To get information about reserved nodes for your AWS account, type the following command at a command prompt:

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

This operation produces output similar to the following (JSON format):

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

For more information, see describe--reserved-cache-nodes in the AWS CLI Reference.

Getting info about your reserved nodes (ElastiCache API)

To get information about reserved nodes for your AWS account, call the `DescribeReservedCacheNodes` operation.

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>
```

For more information, see DescribeReservedCacheNodes in the ElastiCache API Reference.

Migrating previous generation nodes

Previous generation nodes are node types that are being phased out. If you have no existing clusters using a previous generation node type, ElastiCache does not support the creation of new clusters with that node type.

Due to the limited amount of previous generation node types, we cannot guarantee a successful replacement when a node becomes unhealthy in your cluster(s). In such a scenario, your cluster availability may be negatively impacted.

We recommend that you migrate your cluster(s) to a new node type for better availability and performance. For a recommended node type to migrate to, see Upgrade Paths. For a full list of supported node types and previous generation node types in ElastiCache, see Supported node types.

Migrating nodes on a Memcached cluster

To migrate ElastiCache for Memcached to a different node type, you must create a new cluster, which always starts out empty that your application can populate.

To migrate your ElastiCache for Memcached cluster node type using the ElastiCache Console:

- Create a new cluster with the new node type. For more information, see Creating a Memcached cluster (console).
- In your application, update the endpoints to the new cluster's endpoints. For more information, see Finding a Cluster's Endpoints (Console)
- Delete the old cluster. For more information, see Deleting a cluster

Working with ElastiCache

In this section you can find details about how to manage the various components of your ElastiCache implementation.

Topics

- Engine versions and upgrading
- ElastiCache best practices and caching strategies
- Managing your self-designed cluster
- Scaling ElastiCache for Memcached
- Tagging your ElastiCache resources
- Using the Amazon ElastiCache Well-Architected Lens
- Common troubleshooting steps and best practices
- Additional troubleshooting steps

Engine versions and upgrading

This section covers the supported Memcached engine versions and how to upgrade.

Topics

- Supported ElastiCache for Memcached versions
- Engine versions and upgrading
- How to upgrade engine versions

Supported ElastiCache for Memcached versions

ElastiCache supports the following Memcached versions and upgrading to newer versions. When upgrading to a newer version, pay careful attention to the conditions that if not met cause your upgrade to fail.

ElastiCache for Memcached Versions

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

Memcached version 1.6.22

ElastiCache for Memcached adds support for Memcached version 1.6.22. It includes no new features, but does include bug fixes and cumulative updates from Memcached 1.6.18.

For more information, see ReleaseNotes1622 at Memcached on GitHub.

Memcached version 1.6.17

ElastiCache for Memcached adds support for Memcached version 1.6.17. It includes no new features, but does include bug fixes and cumulative updates from Memcached 1.6.17.

For more information, see ReleaseNotes1617 at Memcached on GitHub.

Memcached version 1.6.12

ElastiCache for Memcached adds support for Memcached version 1.6.12 and encryption in-transit. It also includes bug fixes and cumulative updates from Memcached 1.6.6.

For more information, see ReleaseNotes1612 at Memcached on GitHub.

Memcached version 1.6.6

ElastiCache for Memcached adds support for Memcached version 1.6.6. It includes no new features, but does include bug fixes and cumulative updates from Memcached 1.5.16. ElastiCache for Memcached does not include support for Extstore.

For more information, see ReleaseNotes166 at Memcached on GitHub.

Memcached version 1.5.16

ElastiCache for Memcached adds support for Memcached version 1.5.16. It includes no new features, but does include bug fixes and cumulative updates from Memcached 1.5.14 and Memcached 1.5.15.

For more information, see Memcached 1.5.16 Release Notes at Memcached on GitHub.

Memcached version 1.5.10

ElastiCache for Memcached version 1.5.10 supports the following Memcached features:

- Automated slab rebalancing.
- Faster hash table lookups with `murmur3` algorithm.
- Segmented LRU algorithm.
- LRU crawler to background-reclaim memory.
- `--enable-seccomp`: A compile-time option.

It also introduces the `no_modern` and `inline_ascii_resp` parameters. For more information, see Memcached 1.5.10 parameter changes.

Memcached improvements added since ElastiCache for Memcached version 1.4.34 include the following:

- Cumulative fixes, such as ASCII multigets, CVE-2017-9951 and limit crawls for `metadumper`.
- Better connection management by closing connections at the connection limit.
- Improved item-size management for item size above 1MB.
- Better performance and memory-overhead improvements by reducing memory requirements per-item by a few bytes.

For more information, see Memcached 1.5.10 Release Notes at Memcached on GitHub.

Memcached version 1.4.34

ElastiCache for Memcached version 1.4.34 adds no new features to version 1.4.33. Version 1.4.34 is a bug fix release that is larger than the usual such release.

For more information, see Memcached 1.4.34 Release Notes at Memcached on GitHub.

Memcached version 1.4.33

Memcached improvements added since version 1.4.24 include the following:

- Ability to dump all of the metadata for a particular slab class, a list of slab classes, or all slab classes. For more information, see Memcached 1.4.31 Release Notes.
- Improved support for large items over the 1 megabyte default. For more information, see Memcached 1.4.29 Release Notes.
- Ability to specify how long a client can be idle before being asked to close.

  Ability to dynamically increase the amount of memory available to Memcached without having to restart the cluster. For more information, see Memcached 1.4.27 Release Notes.
- Logging of `fetchers`, `mutations`, and `evictions` are now supported. For more information, see Memcached 1.4.26 Release Notes.
- Freed memory can be reclaimed back into a global pool and reassigned to new slab classes. For more information, see Memcached 1.4.25 Release Notes.
- Several bug fixes.
- Some new commands and parameters. For a list, see Memcached 1.4.33 added parameters.

Memcached version 1.4.24

Memcached improvements added since version 1.4.14 include the following:

- Least recently used (LRU) management using a background process.
- Added the option of using *jenkins* or *murmur3* as your hash algorithm.
- Some new commands and parameters. For a list, see Memcached 1.4.24 added parameters.
- Several bug fixes.

Memcached version 1.4.14

Memcached improvements added since version 1.4.5 include the following:

- Enhanced slab rebalancing capability.
- Performance and scalability improvement.
- Introduced the *touch* command to update the expiration time of an existing item without fetching it.
- Auto discovery—the ability for client programs to automatically determine all of the cache nodes in a cluster, and to initiate and maintain connections to all of these nodes.

Memcached version 1.4.5

Memcached version 1.4.5 was the initial engine and version supported by Amazon ElastiCache for Memcached.

Engine versions and upgrading

MAJOR versions are for API incompatible changes and MINOR versions are for new functionality added in a backwards-compatible way. PATCH versions are for backwards-compatible bug fixes and non-functional changes.

Version management for ElastiCache Serverless

ElastiCache Serverless automatically applies the latest MINOR and PATCH software version to your cache, without any impact or downtime to your application. No action is required on your end.

When a new MAJOR version is available, ElastiCache Serverless will send you a notification in the console and an event in EventBridge. You can choose to upgrade your cache to the latest major version by modifying your cache using the Console, CLI, or API and selecting the latest engine version.

Version management for self-designed ElastiCache clusters

When working with self-designed ElastiCache clusters, you can control when the software powering your cache cluster is upgraded to new versions that are supported by ElastiCache . You can control when to upgrade your cache to the latest available MAJOR, MINOR, and PATCH versions. You initiate engine version upgrades to your cluster or replication group by modifying it and specifying a new engine version.

You can control if and when the protocol-compliant software powering your cache cluster is upgraded to new versions that are supported by ElastiCache. This level of control enables you to maintain compatibility with specific versions, test new versions with your application before deploying in production, and perform version upgrades on your own terms and timelines.

Because version upgrades might involve some compatibility risk, they don't occur automatically. You must initiate them.

To upgrade to a newer Memcached version, modify your cache cluster specifying the new engine version you want to use. Upgrading to a newer Memcached version is a destructive process – you lose your data and start with a cold cache. For more information, see Modifying clusters.

You should be aware of the following requirements when upgrading from an older version of Memcached to Memcached version 1.4.33 or newer. `CreateCacheCluster` and `ModifyCacheCluster` fails under the following conditions:

- If `slab_chunk_max > max_item_size`.
- If `max_item_size modulo slab_chunk_max != 0`.
- If `max_item_size > ((max_cache_memory - memcached_connections_overhead) / 4)`.

  The value `(max_cache_memory - memcached_connections_overhead)` is the node's memory useable for data. For more information, see Memcached connection overhead.

Upgrade considerations when working with self-designed clusters

> **Note**
>
> The following considerations only apply when upgrading self-designed clusters. They do not apply to ElastiCache Serverless.

When upgrading a self-designed cluster, consider the follwing

- Engine version management is designed so that you can have as much control as possible over how patching occurs. However, ElastiCache reserves the right to patch your cluster on your behalf in the unlikely event of a critical security vulnerability in the system or cache software.
- Because the Memcached engine does not support persistence, Memcached engine version upgrades are always a disruptive process that clears all cache data in the cluster.

How to upgrade engine versions

To start version upgrades to your cluster, you modify it and specify a newer engine version. You can do this by using the ElastiCache console, the AWS CLI, or the ElastiCache API:

- To use the AWS Management Console, see – Modifying clusters through the console.
- To use the AWS CLI, see Modifying clusters with the CLI.
- To use the ElastiCache API, see Modifying clusters through the API.

How to upgrade engine versions

To start version upgrades to your cluster, you modify it and specify a newer engine version. You can do this by using the ElastiCache console, the AWS CLI, or the ElastiCache API:

- To use the AWS Management Console, see – Using the AWS Management Console.
- To use the AWS CLI, see Using the AWS CLI.
- To use the ElastiCache API, see Using the ElastiCache API.

ElastiCache best practices and caching strategies

Below you can find recommended best practices for Amazon ElastiCache. Following these improves your cache's performance and reliability.

Topics

Best practices with Memcached clients

To learn more about best practices for interacting with ElastiCache resources with commonly used open-source Memcached client libraries, see the topics below.

Topics

Configuring your ElastiCache client for efficient load balancing

> **Note**
>
> This section applies to self-designed multi-node Memcached clusters.

To effectively use multiple ElastiCache Memcached nodes, you need to be able to spread your cache keys across the nodes. A simple way to load balance a cluster with *n* nodes is to calculate the hash of the object's key and mod the result by *n* - `hash(key) mod n`. The resulting value (0 through *n*–1) is the number of the node where you place the object.

This approach is simple and works well as long as the number of nodes (*n*) is constant. However, whenever you add or remove a node from the cluster, the number of keys that need to be moved is *(n - 1) / n* (where *n* is the new number of nodes). Thus, this approach results in a large number of keys being moved, which translates to a large number of initial cache misses, especially as the number of nodes gets large. Scaling from 1 to 2 nodes results in (2-1) / 2 (50 percent) of the keys being moved, the best case. Scaling from 9 to 10 nodes results in (10–1)/10 (90 percent) of the keys being moved. If you're scaling up due to a spike in traffic, you don't want to have a large number of cache misses. A large number of cache misses results in hits to the database, which is already overloaded due to the spike in traffic.

The solution to this dilemma is consistent hashing. Consistent hashing uses an algorithm such that whenever a node is added or removed from a cluster, the number of keys that must be moved is roughly *1 / n* (where *n* is the new number of nodes). Scaling from 1 to 2 nodes results in 1/2 (50 percent) of the keys being moved, the worst case. Scaling from 9 to 10 nodes results in 1/10 (10 percent) of the keys being moved.

As the user, you control which hashing algorithm is used for multi-node clusters. We recommend that you configure your clients to use consistent hashing. Fortunately, there are many Memcached client libraries in most popular languages that implement consistent hashing. Check the documentation for the library you are using to see if it supports consistent hashing and how to implement it.

If you are working in Java, PHP, or .NET, we recommend you use one of the Amazon ElastiCache client libraries.

Consistent Hashing Using Java

The ElastiCache Memcached Java client is based on the open-source spymemcached Java client, which has consistent hashing capabilities built in. The library includes a KetamaConnectionFactory class that implements consistent hashing. By default, consistent hashing is turned off in spymemcached.

For more information, see the KetamaConnectionFactory documentation at KetamaConnectionFactory.

Consistent hashing using PHP

The ElastiCache Memcached PHP client is a wrapper around the built-in Memcached PHP library. By default, consistent hashing is turned off by the Memcached PHP library.

Use the following code to turn on consistent hashing.

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

In addition to the preceding code, we recommend that you also turn `memcached.sess_consistent_hash` on in your php.ini file.

For more information, see the run-time configuration documentation for Memcached PHP at http://php.net/manual/en/memcached.configuration.php. Note specifically the `memcached.sess_consistent_hash` parameter.

Consistent hashing using .NET

The ElastiCache Memcached .NET client is a wrapper around Enyim Memcached. By default, consistent hashing is turned on by the Enyim Memcached client.

For more information, see the `memcached/locator` documentation at https://github.com/enyim/EnyimMemcached/wiki/MemcachedClient-Configuration#user-content-memcachedlocator.

IPv6 client examples

Note

This section applies to self-designed Memcached clusters.

ElastiCache is compatible with open-source Memcached. This means that open source clients for Memcached that support IPv6 connections should be able to connect to IPv6 enabled ElastiCache for Memcached clusters. In addition, the following clients have specifically been validated to work with all supported network type configurations:

Following are best practices for interacting with IPv6 enabled ElastiCache resources with commonly used open-source client libraries. You can view existing best practices for interacting with

ElastiCache for recommendations on configuring clients for ElastiCache resources. However, there are some caveats worth noting when interacting with IPv6 enabled resources.

Validated clients

Validated Clients:

- AWS ElastiCache Cluster Client Memcached for Php – Version *3.6.2
- AWS ElastiCache Cluster Client Memcached for Java – Latest master on Github

Configuring a preferred protocol for dual stack clusters

For Memcached clusters you can control the protocol clients will use to connect to the nodes in the cluster with the IP Discovery parameter. The IP Discovery parameter can be set to either IPv4 or IPv6.

The IP discovery parameter controls the IP protocol used in the config get cluster output. Which in turn will determine the IP protocol used by clients which support auto-discovery for ElastiCache for Memcached clusters.

Changing the IP Discovery will not result in any downtime for connected clients. However, the changes will take some time to propagate.

Monitor the output of `getAvailableNodeEndPoints` for Java and for Php monitor the output of `getServerList`. Once the output of these functions reports IPs for all of the nodes in the cluster that use the updated protocol, the changes have finished propagating.

Java Example:

```
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 Example:

```
$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);
```

Any existing client connections that were created before the IP Discovery was updated will still be connected using the old protocol. All of the validated clients will automatically reconnect to the cluster using the new IP protocol once the changes are detected in the output of the cluster discovery commands. However, this depends on the implementation of the client.

TLS enabled dual stack ElastiCache clusters

When TLS is enabled for ElastiCache clusters the cluster discovery functions `config get cluster` return hostnames instead of IPs. The hostnames are then used instead of IPs to connect to the ElastiCache cluster and perform a TLS handshake. This means that clients won't be affected by the IP Discovery parameter. For TLS enabled clusters the IP Discovery parameter has no effect on the preferred IP protocol. Instead, the IP protocol used will be determined by which IP protocol the client prefers when resolving DNS hostnames.

Java clients

When connecting from a Java environment that supports both IPv4 and IPv6, Java will by default prefer IPv4 over IPv6 for backwards compatibility. However, the IP protocol preference is configurable through the JVM arguments. To prefer IPv4, the JVM accepts `-Djava.net.preferIPv4Stack=true` and to prefer IPv6 set `-Djava.net.preferIPv6Stack=true`. Setting `-Djava.net.preferIPv4Stack=true` means that the JVM will no longer make any IPv6 connections.

Host Level Preferences

In general, if the client or client runtime don't provide configuration options for setting an IP protocol preference, when performing DNS resolution, the IP protocol will depend on the host's configuration. By default, most hosts prefer IPv6 over IPv4 but this preference can be configured at the host level. This will affect all DNS requests from that host, not just those to ElastiCache clusters.

Linux hosts

For Linux, an IP protocol preference can be configured by modifying the `gai.conf` file. The `gai.conf` file can be found under `/etc/gai.conf`. If there is no `gai.conf` specified then an example one should be available under `/usr/share/doc/glibc-common-x.xx/gai.conf` which can be copied to `/etc/gai.conf` and then the default configuration should be un-commented. To update the configuration to prefer IPv4 when connecting to an ElastiCache cluster update the precedence for the CIDR range encompassing the cluster IPs to be above the precedence for default IPv6 connections. By default IPv6 connections have a precedence of 40. For example, assuming the cluster is located in a subnet with the CIDR 172.31.0.0:0/16, the configuration below would cause clients to prefer IPv4 connections to that cluster.

```
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
```

More details on `gai.conf` are available on the Linux main page

Windows hosts

The process for Windows hosts is similar. For Windows hosts you can run `netsh interface ipv6 set prefix CIDR_CONTAINING_CLUSTER_IPS PRECEDENCE LABEL`. This has the same effect as modifying the `gai.conf` file on Linux hosts.

This will update the preference policies to prefer IPv4 connections over IPv6 connections for the specified CIDR range. For example, assuming that the cluster is in a subnet with the 172.31.0.0:0/16 CIDR executing `netsh interface ipv6 set prefix ::ffff:172.31.0.0:0/112 100 15` would result in the following precedence table which would cause clients to prefer IPv4 when connecting to the cluster.

```
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
```

Supported Memcached commands

ElastiCache Serverless for Memcached supports all of the memcached commands in open source memcached 1.6 except for the following:

- Client connections require TLS, as a result UDP protocol is not supported.
- Binary protocol is not supported, as it is officially deprecated in memcached 1.6.
- `GET/GETS` commands are limited to 16KB to avoid potential DoS attack to the server with fetching large number of keys.
- Delayed `flush_all` command will be rejected with `CLIENT_ERROR`.
- Commands that configure the engine or reveal internal information about engine state or logs are not supported, such as:
  - For `STATS` command, only `stats` and `stats reset` are supported. Other variations will return `ERROR`
  - `lru / lru_crawler` - modification for LRU and LRU crawler settings
  - `watch` - watches memcached server logs
  - `verbosity` - configures the server log level

Caching strategies

In the following topic, you can find strategies for populating and maintaining your cache.

What strategies to implement for populating and maintaining your cache depend upon what data you cache and the access patterns to that data. For example, you likely don't want to use the same strategy for both a top-10 leaderboard on a gaming site and trending news stories. In the rest of this section, we discuss common cache maintenance strategies and their advantages and disadvantages.

Topics

Lazy loading

As the name implies, *lazy loading* is a caching strategy that loads data into the cache only when necessary. It works as described following.

Amazon ElastiCache is an in-memory key-value store that sits between your application and the data store (database) that it accesses. Whenever your application requests data, it first makes the request to the ElastiCache cache. If the data exists in the cache and is current, ElastiCache returns the data to your application. If the data doesn't exist in the cache or has expired, your application requests the data from your data store. Your data store then returns the data to your application. Your application next writes the data received from the store to the cache. This way, it can be more quickly retrieved the next time it's requested.

A *cache hit* occurs when data is in the cache and isn't expired:

1. Your application requests data from the cache.
2. The cache returns the data to the application.

A *cache miss* occurs when data isn't in the cache or is expired:

1. Your application requests data from the cache.

2. The cache doesn't have the requested data, so returns a `null`.
3. Your application requests and receives the data from the database.
4. Your application updates the cache with the new data.

Advantages and disadvantages of lazy loading

The advantages of lazy loading are as follows:

- Only requested data is cached.

  Because most data is never requested, lazy loading avoids filling up the cache with data that isn't requested.
- Node failures aren't fatal for your application.

  When a node fails and is replaced by a new, empty node, your application continues to function, though with increased latency. As requests are made to the new node, each cache miss results in a query of the database. At the same time, the data copy is added to the cache so that subsequent requests are retrieved from the cache.

The disadvantages of lazy loading are as follows:

- There is a cache miss penalty. Each cache miss results in three trips:
  1. Initial request for data from the cache
  2. Query of the database for the data
  3. Writing the data to the cache

  These misses can cause a noticeable delay in data getting to the application.
- Stale data.

  If data is written to the cache only when there is a cache miss, data in the cache can become stale. This result occurs because there are no updates to the cache when data is changed in the database. To address this issue, you can use the Write-through and Adding TTL strategies.

Lazy loading pseudocode example

The following is a pseudocode example of lazy loading logic.

```
// *****************************************
// 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
```

For this example, the application code that gets the data is the following.

```
customer_record = get_customer(12345)
```

Write-through

The write-through strategy adds data or updates data in the cache whenever data is written to the database.

Advantages and disadvantages of write-through

The advantages of write-through are as follows:

- Data in the cache is never stale.

  Because the data in the cache is updated every time it's written to the database, the data in the cache is always current.
- Write penalty vs. read penalty.

  Every write involves two trips:

  1. A write to the cache

2. A write to the database

   Which adds latency to the process. That said, end users are generally more tolerant of latency when updating data than when retrieving data. There is an inherent sense that updates are more work and thus take longer.

The disadvantages of write-through are as follows:

- Missing data.

  If you spin up a new node, whether due to a node failure or scaling out, there is missing data. This data continues to be missing until it's added or updated on the database. You can minimize this by implementing lazy loading with write-through.
- Cache churn.

  Most data is never read, which is a waste of resources. By adding a time to live (TTL) value, you can minimize wasted space.

Write-through pseudocode example

The following is a pseudocode example of write-through logic.

```
// *****************************************
// 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
```

For this example, the application code that gets the data is the following.

```
save_customer(12345,{"address":"123 Main"})
```

Adding TTL

Lazy loading allows for stale data but doesn't fail with empty nodes. Write-through ensures that data is always fresh, but can fail with empty nodes and can populate the cache with superfluous data. By adding a time to live (TTL) value to each write, you can have the advantages of each strategy. At the same time, you can and largely avoid cluttering up the cache with extra data.

Time to live (TTL) is an integer value that specifies the number of seconds until the key expires. Memcached specifies this value in seconds. When an application attempts to read an expired key, it is treated as though the key is not found. The database is queried for the key and the cache is updated. This approach doesn't guarantee that a value isn't stale. However, it keeps data from getting too stale and requires that values in the cache are occasionally refreshed from the database.

For more information, see the Memcached `set` command.

TTL pseudocode examples

The following is a pseudocode example of write-through logic with TTL.

```
// *****************************************
// 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
```

The following is a pseudocode example of lazy loading logic with TTL.

```
// *****************************************
// 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
```

For this example, the application code that gets the data is the following.

```
save_customer(12345,{"address":"123 Main"})
```

```
customer_record = get_customer(12345)
```

Related topics

- In-Memory Data Store
- Choosing an engine and version
- Scaling ElastiCache for Memcached

Managing your self-designed cluster

This section contains topics that help you manage your self-designed clusters.

> **Note**
>
> These topics do not apply to ElastiCache Serverless.

Topics

- Managing maintenance
- Configuring engine parameters using parameter groups

Managing maintenance

Every cluster has a weekly maintenance window during which any system changes are applied. If you don't specify a preferred maintenance window when you create or modify a cluster, ElastiCache assigns a 60-minute maintenance window within your region's maintenance window on a randomly chosen day of the week.

The 60-minute maintenance window is chosen at random from an 8-hour block of time per region. The following table lists the time blocks for each region from which the default maintenance windows are assigned. You may choose a preferred maintenance window outside the region's maintenance window block.

Region Code	Region Name	Region Maintenance Window
`ap-northeast-1`	Asia Pacific (Tokyo) Region	13:00–21:00 UTC
`ap-northeast-2`	Asia Pacific (Seoul) Region	12:00–20:00 UTC
`ap-northeast-3`	Asia Pacific (Osaka) Region	12:00–20:00 UTC
`ap-southeast-3`	Asia Pacific (Jakarta) Region	14:00–22:00 UTC
`ap-south-1`	Asia Pacific (Mumbai) Region	17:30–1:30 UTC
`ap-southeast-1`	Asia Pacific (Singapore) Region	14:00–22:00 UTC
`cn-north-1`	China (Beijing) Region	14:00–22:00 UTC
`cn-northwest-1`	China (Ningxia) Region	14:00–22:00 UTC
`ap-east-1`	Asia Pacific (Hong Kong) Region	13:00–21:00 UTC
`ap-southeast-2`	Asia Pacific (Sydney) Region	12:00–20:00 UTC
`eu-west-3`	EU (Paris) Region	23:59–07:29 UTC
`af-south-1`	Africa (Cape Town) Region	13:00–21:00 UTC

Region Code	Region Name	Region Maintenance Window
`eu-central-1`	Europe (Frankfurt) Region	23:00–07:00 UTC
`eu-west-1`	Europe (Ireland) Region	22:00–06:00 UTC
`eu-west-2`	Europe (London) Region	23:00–07:00 UTC
`me-south-1`	Middle East (Bahrain) Region	13:00–21:00 UTC
`me-central-1`	Middle East (UAE) Region	13:00–21:00 UTC
`eu-south-1`	Europe (Milan) Region	21:00–05:00 UTC
`sa-east-1`	South America (São Paulo) Region	01:00–09:00 UTC
`us-east-1`	US East (N. Virginia) Region	03:00–11:00 UTC
`us-east-2`	US East (Ohio) Region	04:00–12:00 UTC
`us-gov-west-1`	AWS GovCloud (US) region	06:00–14:00 UTC
`us-west-1`	US West (N. California) Region	06:00–14:00 UTC
`us-west-2`	US West (Oregon) Region	06:00–14:00 UTC

Changing your Cluster's Maintenance Window

The maintenance window should fall at the time of lowest usage and thus might need modification from time to time. You can modify your cluster to specify a time range of up to 24 hours in duration during which any maintenance activities you have requested should occur. Any deferred or pending cluster modifications you requested occur during this time.

> **Note**
>
> If you want to apply node type modifications and/or engine upgrades immediately using the AWS Management Console select the **Apply Immediately** box. Otherwise these modifications will be applied during your next scheduled maintenance window. To the use the API, see modify-replication-group or modify-cache-cluster.

More information

For information on your maintenance window and node replacement, see the following:

- ElastiCache Maintenance—FAQ on maintenance and node replacement
- Replacing nodes—Managing node replacement
- Modifying an ElastiCache cluster—Changing a cluster's maintenance window

Configuring engine parameters using parameter groups

Amazon ElastiCache uses parameters to control the runtime properties of your nodes and clusters. Generally, newer engine versions include additional parameters to support the newer functionality. For tables of parameters, see Memcached specific parameters.

As you would expect, some parameter values, such as `maxmemory`, are determined by the engine and node type. For a table of these parameter values by node type, see Memcached node-type specific parameters.

> **Note**
>
> For a list of Memcached-specific parameters, see Memcached Specific Parameters.

Topics

- Parameter management
- Cache parameter group tiers
- Creating a parameter group
- Listing parameter groups by name
- Listing a parameter group's values
- Modifying a parameter group
- Deleting a parameter group
- Memcached specific parameters

Parameter management

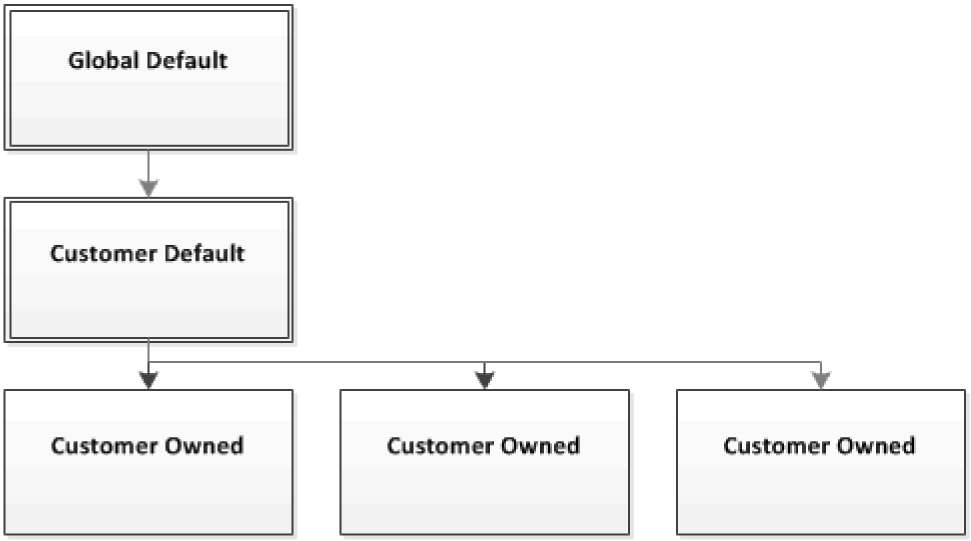
Parameters are grouped together into named parameter groups for easier parameter management. A parameter group represents a combination of specific values for the parameters that are passed to the engine software during startup. These values determine how the engine processes on each node behave at runtime. The parameter values on a specific parameter group apply to all nodes that are associated with the group, regardless of which cluster they belong to.

To fine-tune your cluster's performance, you can modify some parameter values or change the cluster's parameter group.

- You cannot modify or delete the default parameter groups. If you need custom parameter values, you must create a custom parameter group.
- The parameter group family and the cluster you're assigning it to must be compatible. For example, if your cluster is running Memcached version 1.4.8, you can only use parameter groups, default or custom, from the Memcached 1.4 family.
- If you change a cluster's parameter group, the values for any conditionally modifiable parameter must be the same in both the current and new parameter groups.
- When you change a cluster's parameters, the change is applied to the cluster immediately. This is true whether you change the cluster's parameter group itself or a parameter value within the cluster's parameter group. To determine when a particular parameter change is applied, see the **Changes Take Effect** column in the tables for Memcached specific parameters. For information on rebooting a cluster's nodes, see Rebooting clusters.

Cache parameter group tiers

Amazon ElastiCache has three tiers of cache parameter groups as shown following.



Amazon ElastiCache parameter group tiers

Global Default

The top-level root parameter group for all Amazon ElastiCache customers in the region.

The global default cache parameter group:

- Is reserved for ElastiCache and not available to the customer.

Customer Default

A copy of the Global Default cache parameter group which is created for the customer's use.

The Customer Default cache parameter group:

- Is created and owned by ElastiCache.
- Is available to the customer for use as a cache parameter group for any clusters running an engine version supported by this cache parameter group.
- Cannot be edited by the customer.

Customer Owned

A copy of the Customer Default cache parameter group. A Customer Owned cache parameter group is created whenever the customer creates a cache parameter group.

The Customer Owned cache parameter group:

- Is created and owned by the customer.
- Can be assigned to any of the customer's compatible clusters.
- Can be modified by the customer to create a custom cache parameter group.

  Not all parameter values can be modified. For more information, see Memcached specific parameters.

Creating a parameter group

You need to create a new parameter group if there is one or more parameter values that you want changed from the default values. You can create a parameter group using the ElastiCache console, the AWS CLI, or the ElastiCache API.

Creating a parameter group (Console)

The following procedure shows how to create a parameter group using the ElastiCache console.

To create a parameter group using the ElastiCache console

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. To see a list of all available parameter groups, in the left hand navigation pane choose **Parameter Groups**.
3. To create a parameter group, choose **Create Parameter Group**.

   The **Create Parameter Group** screen appears.
4. From the **Family** list, choose the parameter group family that will be the template for your parameter group.

   The parameter group family, such as *memcached1.4* , defines the actual parameters in your parameter group and their initial values. The parameter group family must coincide with the cluster's engine and version.
5. In the **Name** box, type in a unique name for this parameter group.

When creating a cluster or modifying a cluster's parameter group, you will choose the parameter group by its name. Therefore, we recommend that the name be informative and somehow identify the parameter group's family.

Parameter group naming constraints are as follows:

- Must begin with an ASCII letter.
- Can only contain ASCII letters, digits, and hyphens.
- Must be 1–255 characters long.
- Can't contain two consecutive hyphens.
- Can't end with a hyphen.

6. In the **Description** box, type in a description for the parameter group.
7. To create the parameter group, choose **Create**.

   To terminate the process without creating the parameter group, choose **Cancel**.
8. When the parameter group is created, it will have the family's default values. To change the default values you must modify the parameter group. For more information, see Modifying a parameter group.

Creating a parameter group (AWS CLI)

To create a parameter group using the AWS CLI, use the command `create-cache-parameter-group` with these parameters.

- `--cache-parameter-group-name` — The name of the parameter group.

  Parameter group naming constraints are as follows:
  - Must begin with an ASCII letter.
  - Can only contain ASCII letters, digits, and hyphens.
  - Must be 1–255 characters long.
  - Can't contain two consecutive hyphens.
  - Can't end with a hyphen.
- `--cache-parameter-group-family` — The engine and version family for the parameter group.
- `--description` — A user supplied description for the parameter group.

Example

The following example creates a parameter group named *myMem14* using the memcached1.4 family as the template.

For Linux, macOS, or Unix:

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

For Windows:

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

The output from this command should look something like this.

```
{
    "CacheParameterGroup": {
        "CacheParameterGroupName": "myMem14",
        "CacheParameterGroupFamily": "memcached1.4",
        "Description": "My first  parameter group"
    }
}
```

When the parameter group is created, it will have the family's default values. To change the default values you must modify the parameter group. For more information, see Modifying a parameter group.

For more information, see `create-cache-parameter-group`.

Creating a parameter group (ElastiCache API)

To create a parameter group using the ElastiCache API, use the `CreateCacheParameterGroup` action with these parameters.

- `ParameterGroupName` — The name of the parameter group.

  Parameter group naming constraints are as follows:

  - Must begin with an ASCII letter.
  - Can only contain ASCII letters, digits, and hyphens.
  - Must be 1–255 characters long.
  - Can't contain two consecutive hyphens.
  - Can't end with a hyphen.

- `CacheParameterGroupFamily` — The engine and version family for the parameter group. For example, `memcached1.4`.
- `Description` — A user supplied description for the parameter group.

Example

The following example creates a parameter group named *myMem14* using the memcached1.4 family as the template.

```
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>
```

The response from this action should look something like this.

```
<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>
```

When the parameter group is created, it will have the family's default values. To change the default values you must modify the parameter group. For more information, see Modifying a parameter group.

For more information, see `CreateCacheParameterGroup`.

Listing parameter groups by name

You can list the parameter groups using the ElastiCache console, the AWS CLI, or the ElastiCache API.

Listing parameter groups by name (Console)

The following procedure shows how to view a list of the parameter groups using the ElastiCache console.

To list parameter groups using the ElastiCache console

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. To see a list of all available parameter groups, in the left hand navigation pane choose **Parameter Groups**.

Listing parameter groups by name (AWS CLI)

To generate a list of parameter groups using the AWS CLI, use the command `describe-cache-parameter-groups`. If you provide a parameter group's name, only that parameter group will be listed. If you do not provide a parameter group's name, up to `--max-records` parameter groups will be listed. In either case, the parameter group's name, family, and description are listed.

Example

The following sample code lists the parameter group *myMem14*.

For Linux, macOS, or Unix:

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

For Windows:

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

The output of this command will look something like this, listing the name, family, and description for the parameter group.

```
{
    "CacheParameterGroups": [
     {
         "CacheParameterGroupName": "myMem14",
         "CacheParameterGroupFamily": "memcached1.4",
         "Description": "My first parameter group"
     }
    ]
}
```

Example

The following sample code lists up to 10 parameter groups.

```
aws elasticache describe-cache-parameter-groups --max-records 10
```

The JSON output of this command will look something like this, listing the name, family, description and, in the case of redis5.6 whether the parameter group is part of a global datastore (isGlobal), for each parameter group.

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

For more information, see `describe-cache-parameter-groups`.

Listing parameter groups by name (ElastiCache API)

To generate a list of parameter groups using the ElastiCache API, use the `DescribeCacheParameterGroups` action. If you provide a parameter group's name, only that parameter group will be listed. If you do not provide a parameter group's name, up to `MaxRecords` parameter groups will be listed. In either case, the parameter group's name, family, and description are listed.

Example

The following sample code lists the parameter group *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>
```

The response from this action will look something like this, listing the name, family, and description for each parameter group.

```
<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

The following sample code lists up to 10 parameter groups.

```
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>
```

The response from this action will look something like this, listing the name, family, description and, in the case of redis5.6 if the parameter group belongs to a global datastore (isGlobal), for each parameter group.

```
<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>
```

For more information, see `DescribeCacheParameterGroups`.

Listing a parameter group's values

You can list the parameters and their values for a parameter group using the ElastiCache console, the AWS CLI, or the ElastiCache API.

Listing a parameter group's values (Console)

The following procedure shows how to list the parameters and their values for a parameter group using the ElastiCache console.

To list a parameter group's parameters and their values using the ElastiCache console

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. To see a list of all available parameter groups, in the left hand navigation pane choose **Parameter Groups**.
3. Choose the parameter group for which you want to list the parameters and values by choosing the box to the left of the parameter group's name.

   The parameters and their values will be listed at the bottom of the screen. Due to the number of parameters, you may have to scroll up and down to find the parameter you're interested in.

Listing a parameter group's values (AWS CLI)

To list a parameter group's parameters and their values using the AWS CLI, use the command `describe-cache-parameters`.

Example

The following sample code list all the parameters and their values for the parameter group *myMem14*.

For Linux, macOS, or Unix:

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

For Windows:

```
aws elasticache describe-cache-parameters ^
```

```
    --cache-parameter-group-name myMem14
```

For more information, see describe-cache-parameters.

Listing a parameter group's values (ElastiCache API)

To list a parameter group's parameters and their values using the ElastiCache API, use the `DescribeCacheParameters` action.

Example

The following sample code list all the parameters for the parameter group *myMem14*.

```
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>
```

The response from this action will look something like this. This response has been truncated.

```
<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>
```

For more information, see `DescribeCacheParameters`.

Modifying a parameter group

Important

You cannot modify any default parameter group.

You can modify some parameter values in a parameter group. These parameter values are applied to clusters associated with the parameter group. For more information on when a parameter value change is applied to a parameter group, see Memcached specific parameters.

Modifying a parameter group (Console)

The following procedure shows how to change the `binding_protocol` parameter's value using the ElastiCache console. You would use the same procedure to change the value of any parameter.

To change a parameter's value using the ElastiCache console

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.

2. To see a list of all available parameter groups, in the left hand navigation pane choose **Parameter Groups**.
3. Choose the parameter group you want to modify by choosing the box to the left of the parameter group's name.

   The parameter group's parameters will be listed at the bottom of the screen. You may need to page through the list to see all the parameters.
4. To modify one or more parameters, choose **Edit Parameters**.
5. In the **Edit Parameter Group:** screen, scroll using the left and right arrows until you find the `binding_protocol` parameter, then type `ascii` in the **Value** column.
6. In the **Edit Parameter Group:** screen, scroll using the left and right arrows until you find the `cluster-enabled` parameter, then type `yes` in the **Value** column.
7. Choose **Save Changes**.
8. To find the name of the parameter you changed, see Memcached specific parameters. If changes to the parameter take place *After restart*, reboot every cluster that uses this parameter group. For more information, see Rebooting clusters.

Modifying a parameter group (AWS CLI)

To change a parameter's value using the AWS CLI, use the command `modify-cache-parameter-group`.

Example

To find the name and permitted values of the parameter you want to change, see Memcached specific parameters

The following sample code sets the value of two parameters, *chunk_size* and *chunk_size_growth_fact* on the parameter group `myMem14`.

For Linux, macOS, or 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
```

For 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
```

Output from this command will look something like this.

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

For more information, see `modify-cache-parameter-group`.

Modifying a parameter group (ElastiCache API)

To change a parameter group's parameter values using the ElastiCache API, use the `ModifyCacheParameterGroup` action.

Example

To find the name and permitted values of the parameter you want to change, see Memcached specific parameters

The following sample code sets the value of two parameters, *chunk_size* and *chunk_size_growth_fact* on the parameter group `myMem14`.

```
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>
```

For more information, see `ModifyCacheParameterGroup`.

Deleting a parameter group

You can delete a custom parameter group using the ElastiCache console, the AWS CLI, or the ElastiCache API.

You cannot delete a parameter group if it is associated with any clusters. Nor can you delete any of the default parameter groups.

Deleting a parameter group (Console)

The following procedure shows how to delete a parameter group using the ElastiCache console.

To delete a parameter group using the ElastiCache console

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. To see a list of all available parameter groups, in the left hand navigation pane choose **Parameter Groups**.
3. Choose the parameter groups you want to delete by choosing the box to the left of the parameter group's name.

   The **Delete** button will become active.
4. Choose **Delete**.

   The **Delete Parameter Groups** confirmation screen will appear.
5. To delete the parameter groups, on the **Delete Parameter Groups** confirmation screen, choose **Delete**.

   To keep the parameter groups, choose **Cancel**.

Deleting a parameter group (AWS CLI)

To delete a parameter group using the AWS CLI, use the command `delete-cache-parameter-group`. For the parameter group to delete, the parameter group specified by `--cache-parameter-group-name` cannot have any clusters associated with it, nor can it be a default parameter group.

The following sample code deletes the *myMem14* parameter group.

Example

For Linux, macOS, or Unix:

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

For Windows:

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

For more information, see `delete-cache-parameter-group`.

Deleting a parameter group (ElastiCache API)

To delete a parameter group using the ElastiCache API, use the `DeleteCacheParameterGroup` action. For the parameter group to delete, the parameter group specified by `CacheParameterGroupName` cannot have any clusters associated with it, nor can it be a default parameter group.

Example

The following sample code deletes the *myMem14* parameter group.

```
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>
```

For more information, see `DeleteCacheParameterGroup`.

Memcached specific parameters

If you do not specify a parameter group for your Memcached cluster, then a default parameter group appropriate to your engine version will be used. You can't change the values of any parameters in a default parameter group. However, you can create a custom parameter group and assign it to your cluster at any time. For more information, see Creating a parameter group.

Topics

Memcached 1.6.17 changes

From Memcached 1.6.17, we no longer support these administrative commands: `lru_crawler`, `lru`, and `slabs`. With these changes, you will not be able to enable/disable `lru_crawler` at runtime via commands. Please enable/disable `lru_crawler` by modifying your custom parameter group.

Memcached 1.6.6 added parameters

For Memcached 1.6.6, no additional parameters are supported.

Parameter group family: memcached1.6

Memcached 1.5.10 parameter changes

For Memcached 1.5.10, the following additional parameters are supported.

Parameter group family: memcached1.5

Name	Details	Description
`no_modern`	Default: 1 Type: boolean Modifiable: Yes Allowed_Values: 0,1 Changes Take Effect: At launch	An alias for disabling `slab_reassign` , `lru_maintainer_thread` , `lru_segmented` , and`maxconns_fast` commands . When using Memcached 1.5 and higher, `no_modern` also sets the hash_algorithm to `jenkins`. In addition, when using Memcached 1.5.10, `inline_ascii_reponse` is controlled by the parameter `parallelly` . This means that if `no_modern` is disabled then `inline_ascii_reponse` is disabled. From Memcached engine 1.5.16 onward the `inline_ascii_response` parameter no longer applies, so `no_modern` being abled or disabled has no effect on `inline_ascii_reponse` . If `no_modern` is disabled, then `slab_reassign` , `lru_maintainer_thread` , `lru_segmented` , and `maxconns_fast` WILL be enabled. Since `slab_automove` and

Name	Details	Description
		`hash_algorithm` parameters are not SWITCH parameters, their setting is based on the configurations in the parameter group. If you want to disable `no_modern` and revert to `modern`, you must configure a custom parameter group to disable this parameter and then reboot for these changes to take effect. **Note** The default configuration value for this parameter has been changed from 0 to 1 as of August 20, 2021. The updated default value will get automatically picked up by new ElastiCache users for each regions after August 20th, 2021. Existing ElastiCache users in the regions before August 20th, 2021 need to manually modify their custom parameter groups in order to pick up this new change.

Name	Details	Description
`inline_ascii_resp`	Default: 0 Type: boolean Modifiable: Yes Allowed_Values: 0,1 Changes Take Effect: At launch	Stores numbers from `VALUE` response, inside an item, using up to 24 bytes. Small slowdown for ASCII `get`, `faster` sets.

For Memcached 1.5.10, the following parameters are removed.

Name	Details	Description
`expirezero_does_not_evict`	Default: 0 Type: boolean Modifiable: Yes Allowed_Values: 0,1 Changes Take Effect: At launch	No longer supported in this version.
`modern`	Default: 1 Type: boolean Modifiable: Yes (requires re-launch if set to `no_modern`) Allowed_Values: 0,1 Changes Take Effect: At launch	No longer supported in this version. Starting with this version, `no-modern` is enabled by default with every launch or re-launch.

Memcached 1.4.34 added parameters

For Memcached 1.4.34, no additional parameters are supported.

Parameter group family: memcached1.4

Memcached 1.4.33 added parameters

For Memcached 1.4.33, the following additional parameters are supported.

Parameter group family: memcached1.4

Name	Details	Description
`modern`	Default: enabled Type: boolean Modifiable: Yes Changes Take Effect: At launch	An alias to multiple features. Enabling `modern` is equivalent to turning following commands on and using a murmur3 hash algorithm: `slab_reassign`, `slab_automove`, `lru_crawler`, `lru_maintainer`, `maxconns_fast`, and `hash_algorithm=murmur3`.
`watch`	Default: enabled Type: boolean Modifiable: Yes Changes Take Effect: Im ly Logs can get dropped if user hits their `watcher_logbuf_size` and `worker_logbuf_size` limits.	Logs fetches, evictions or mutations. When, for example, user turns `watch` on, they can see logs when `get`, `set`, `delete`, or `update` occur.

Name	Details	Description
`idle_timeout`	Default: 0 (disabled) Type: integer Modifiable: Yes Changes Take Effect: At	The minimum number of seconds a client will be allowed to idle before being asked to close. Range of values: 0 to 86400.
`track_sizes`	Default: disabled Type: boolean Modifiable: Yes Changes Take Effect: At	Shows the sizes each slab group has consumed. Enabling `track_sizes` lets you run `stats sizes` without the need to run `stats sizes_enable`.
`watcher_logbuf_size`	Default: 256 (KB) Type: integer Modifiable: Yes Changes Take Effect: At	The `watch` command turns on stream logging for Memcached. However `watch` can drop logs if the rate of evictions, mutations or fetches are high enough to cause the logging buffer to become full. In such situations, users can increase the buffer size to reduce the chance of log losse s.

Name	Details	Description
`worker_logbuf_size`	Default: 64 (KB) Type: integer Modifiable: Yes Changes Take Effect: At	The `watch` command turns on stream logging for Memcached. However `watch` can drop logs if the rate of evictions, mutations or fetches are high enough to cause logging buffer get full. In such situations, users can increase the buffer size to reduce the chance of log losses.
`slab_chunk_max`	Default: 524288 (bytes) Type: integer Modifiable: Yes Changes Take Effect: At	Specifies the maximum size of a slab. Setting smaller slab size uses memory more efficiently. Items larger than `slab_chunk_max` are split over multiple slabs.
`lru_crawler metadump [all|1|2|3]`	Default: disabled Type: boolean Modifiable: Yes Changes Take Effect: Im ly	if lru_crawler is enabled this command dumps all keys. `all|1|2|3` - all slabs, or specify a particular slab number

Memcached 1.4.24 added parameters

For Memcached 1.4.24, the following additional parameters are supported.

Parameter group family: memcached1.4

Name	Details	Description
`disable_flush_all`	Default: 0 (disabled)	

Name	Details	Description
	Type: boolean Modifiable: Yes Changes Take Effect: At launch	Add parameter (-F) to disable flush_all. Useful if you never want to be able to run a full flush on production instances. Values: 0, 1 (user can do a `flush_all` when the value is 0).
`hash_algorithm`	Default: jenkins Type: string Modifiable: Yes Changes Take Effect: At launch	The hash algorithm to be used. Permitted values: murmur3 and jenkins.

Name	Details	Description
`lru_crawler`	Default: 0 (disabled) Type: boolean Modifiable: Yes Changes Take Effect: Af **Note** You can temporarily enable `lru_crawler` at runtime from the command line. For more information, see the Description column.	Cleans slab classes of items that have expired. This is a low impact process that runs in the background. Currently requires initiating a crawl using a manual command. To temporarily enable, run `lru_crawler enable` at the command line. `lru_crawler 1,3,5` crawls slab classes 1, 3, and 5 looking for expired items to add to the freelist. Values: 0,1 **Note** Enabling `lru_crawler` at the command line enables the crawler until either disabled at the command line or the next reboot. To enable permanently, you must modify the parameter value. For more information, see Modifying a parameter group.

Name	Details	Description
`lru_maintainer`	Default: 0 (disabled) Type: boolean Modifiable: Yes Changes Take Effect: At launch	A background thread that shuffles items between the LRUs as capacities are reached. Values: 0, 1.
`expirezero_does_not_evict`	Default: 0 (disabled) Type: boolean Modifiable: Yes Changes Take Effect: At launch	When used with `lru_maintainer` , makes items with an expiration time of 0 unevictable. ⚠ **Warning** This can crowd out memory available for other evictable items. Can be set to disregard `lru_maintainer` .

Memcached 1.4.14 added parameters

For Memcached 1.4.14, the following additional parameters are supported.

Parameter group family: memcached1.4

Parameters added in Memcached 1.4.14

Name	Description
`config_max`	The maximum number of ElastiCache configuration entries.

Name	Description
`config_size_max`	The maximum size of the configuration entries, in bytes.
`hashpower_init`	The initial size of the ElastiCache hash table, expressed as a power of two. The default is 16 (2^16), or 65536 keys.
`maxconns_fast`	Changes the way in which new connections requests are handled wh en the maximum connection limit is reached. If this parameter is set to 0 (zero), new connections are added to the backlog queue and will wait until other connections are closed. If the parameter is set to 1, ElastiCache sends an error to the client and immediately closes the connection.

Name	Description
slab_automove	Adjusts the slab automove algorithm: If this parameter is set to 0 (zero), the automove algorithm is disabled. If it is set to 1, ElastiCache takes a slow, conservative approach to automatically moving slabs. If it is set to 2, ElastiCache aggressively moves slabs whenever there is an eviction. (This mode is not recommended except for testing purposes.)
slab_reassign	Enable or disable slab reassignment. If this parameter is set to 1, you can use the "slabs reassign" command to manually reassign memory.

Memcached 1.4.5 supported parameters

Parameter group family: memcached1.4

For Memcached 1.4.5, the following parameters are supported.

Parameters added in Memcached 1.4.5

Name	Details	Description
backlog_queue_limit	Default: 1024 Type: integer	The backlog queue limit.

Name	Details	Description
	Modifiable: No	
binding_protocol	Default: auto Type: string Modifiable: Yes Changes Take Effect: After resta	The binding protocol. Permissible values are: `ascii` and `auto`. For guidance on modifying the value of `binding_protocol` , see [Modifying a parameter group]().
cas_disabled	Default: 0 (false) Type: Boolean Modifiable: Yes Changes Take Effect: After resta	If 1 (true), check and set (CAS) operations will be disabled, and items stored will consume 8 fewer bytes than with CAS enabled.
chunk_size	Default: 48 Type: integer Modifiable: Yes Changes Take Effect: After resta	The minimum amount, in bytes, of space to allocate for the smallest item's key, value, and flags.
chunk_size_growth_factor	Default: 1.25 Type: float Modifiable: Yes Changes Take Effect: After resta	The growth factor that controls the size of each successive Memcached chunk; each chunk will be `chunk_size_growth_factor` times larger than the previous chunk.
error_on_memory_exhausted	Default: 0 (false) Type: Boolean Modifiable: Yes Changes Take Effect: After resta	If 1 (true), when there is no more memory to store items, Memcached will return an error rather than evicting items.

Name	Details	Description
large_memory_pages	Default: 0 (false) Type: Boolean Modifiable: No	If 1 (true), ElastiCache will try to use large memory pages.
lock_down_paged_memory	Default: 0 (false) Type: Boolean Modifiable: No	If 1 (true), ElastiCache will lock down all paged memory.
max_item_size	Default: 1048576 Type: integer Modifiable: Yes Changes Take Effect: After resta	The size, in bytes, of the largest item that can be stored in the cluster.
max_simultaneous_connections	Default: 65000 Type: integer Modifiable: No	The maximum number of simultaneous connections.
maximize_core_file_limit	Default: 0 (false) Type: Boolean Modifiable: Changes Take Effect: After resta	If 1 (true), ElastiCache will maximize the core file limit.
memcached_connections_overhead	Default: 100 Type: integer Modifiable: Yes Changes Take Effect: After resta	The amount of memory to be reserved for Memcached connections and other miscellaneous overhead. For information about this parameter, see Memcached connection overhead.

Name	Details	Description
`requests_per_event`	Default: 20 Type: integer Modifiable: No	The maximum number of requests per event for a given connection. This limit is required to prevent resource starvation.

Memcached connection overhead

On each node, the memory made available for storing items is the total available memory on that node (which is stored in the `max_cache_memory` parameter) minus the memory used for connections and other overhead (which is stored in the `memcached_connections_overhead` parameter). For example, a node of type `cache.m1.small` has a `max_cache_memory` of 1300MB. With the default `memcached_connections_overhead` value of 100MB, the Memcached process will have 1200MB available to store items.

The default values for the `memcached_connections_overhead` parameter satisfy most use cases; however, the required amount of allocation for connection overhead can vary depending on multiple factors, including request rate, payload size, and the number of connections.

You can change the value of the `memcached_connections_overhead` to better suit the needs of your application. For example, increasing the value of the `memcached_connections_overhead` parameter will reduce the amount of memory available for storing items and provide a larger buffer for connection overhead. Decreasing the value of the `memcached_connections_overhead` parameter will give you more memory to store items, but can increase your risk of swap usage and degraded performance. If you observe swap usage and degraded performance, try increasing the value of the `memcached_connections_overhead` parameter.

⚠ **Important**

For the `cache.t1.micro` node type, the value for `memcached_connections_overhead` is determined as follows:

- If you cluster is using the default parameter group, ElastiCache will set the value for `memcached_connections_overhead` to 13MB.

- If your cluster is using a parameter group that you have created yourself, you can set the value of `memcached_connections_overhead` to a value of your choice.

Memcached node-type specific parameters

Although most parameters have a single value, some parameters have different values depending on the node type used. The following table shows the default values for the `max_cache_memory` and `num_threads` parameters for each node type. The values on these parameters cannot be modified.

Node type	max_cache_memory (in megabytes)	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

Node type	max_cache_memory (in megabytes)	num_threads
cache.m2.xlarge	33800	2
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

Node type	max_cache_memory (in megabytes)	num_threads
cache.m6g.2xlarge	26671	8
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

Node type	max_cache_memory (in megabytes)	num_threads
cache.r5.4xlarge	108347	16
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

All T2 instances are created in an Amazon Virtual Private Cloud (Amazon VPC).

Scaling ElastiCache for Memcached

Scaling ElastiCache for Memcached

ElastiCache Serverless automatically accommodates your workload traffic as it ramps up or down. For each ElastiCache Serverless cache, ElastiCache continuously tracks the utilization of resources such as CPU, memory, and network. When any of these resources are constrained, ElastiCache Serverless scales out by adding a new shard and redistributing data to the new shard, without any downtime to your application. You can monitor the resources being consumed by your cache in CloudWatch by monitoring the `BytesUsedForCache` metric for cache data storage and `ElastiCacheProcessingUnits` (ECPU) for compute usage.

Setting scaling limits to manage costs

You can choose to configure a maximum usage on both cache data storage and ECPU/second for your cache to control cache costs. Doing so will ensure that your cache usage never exceeds the configured maximum.

If you set a scaling maximum, your application may experience decreased cache performance when the cache hits the maximum. When you set a cache data storage maximum and your cache data storage hits the maximum, ElastiCache will begin evicting data in your cache using LRU logic. When you set an ECPU/second maximum and the compute utilization of your workload exceeds this value, ElastiCache will begin throttling Memcached requests.

If you setup a maximum limit on `BytesUsedForCache` or `ElastiCacheProcessingUnits`, we highly recommend setting up a CloudWatch alarm at a value lower than the maximum limit so that you are notified when your cache is operating close to these limits. We recommend setting an alarm at 75% of the maximum limit you set. See documentation about how to set up CloudWatch alarms.

Pre-scaling with ElastiCache Serverless

ElastiCache Serverless pre-scaling

With pre-scaling, also called pre-warming, you can set minimum supported limits for your ElastiCache cache. You can set these minimums for ElastiCache Processing Units (ECPUs) per second or data storage. This can be useful in preparation for anticipated scaling events. For example, if a gaming company expects a 5x increase in logins within the first minute that their new game launches, they can ready their cache for this significant spike in usage.

You can perform pre-scaling using the ElastiCache console, CLI, or API. ElastiCache Serverless updates the available ECPUs/second on the cache within 60 minutes, and sends an event notification when the minimum limit update is completed.

How pre-scaling works

When the minimum limit for ECPUs/second or data storage is updated via the console, CLI, or API, that new limit is available within 1 hour. ElastiCache Serverless supports 30K ECPUs/second on an empty cache, and up to 90K ECPUs/sec when using the Read from Replica feature. ElastiCache can double ECPUs/second every 10-12 minutes. This scaling speed is sufficient for most workloads. If you anticipate that an upcoming scaling event might exceed this rate, then we recommend setting the minimum ECPUs/second to the peak ECPUs/sec you expect at least 60 minutes before the peak event. Otherwise, the application may experience elevated latency and throttling of requests.

Once the minimum limit update is complete, ElastiCache Serverless will start metering you for the new minimum ECPUs per second or the new minimum storage. This occurs even if your application is not executing requests on the cache, or if your data storage usage is below the minimum. When you lower the minimum limit from its current setting, the update is immediate so ElastiCache Serverless will begin metering at the new minimum limit immediately.

> **Note**
>
> - When you set a minimum usage limit, you are charged for that limit even if your actual usage is lower than the minimum usage limit. ECPU or data storage usage that exceeds the minimum usage limit are charged the regular rate. For example, if you set a minimum usage limit of 100,000 ECPUs/second then you will be charged at least $1.224 per hour (using ECPU prices in us-east-1), even if your usage is lower than that set minimum.
> - ElastiCache Serverless supports the requested minimum scale at an aggregate level on the cache. ElastiCache Serverless also supports a maximum of 30K ECPUs/second per slot (90K ECPUs/second when using Read from Replica using READONLY connections). As a best practice, your application should ensure that key distribution across Redis slots and traffic across keys is as uniform as possible.

Setting scaling limits using the console and AWS CLI

Setting scaling limits using the AWS Console

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the navigation pane, choose the engine running on the cache that you want to modify.
3. A list of caches running the chosen engine appears.
4. Choose the cache to modify by choosing the radio button to the left of the cache's name.
5. Choose **Actions** and then choose **Modify**.
6. Under **Usage limits**, set appropriate **Memory** or **Compute** limits.
7. Click **Preview** changes and then **Save** changes.

Setting scaling limits using the AWS CLI

To change scaling limits using the CLI, use the 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}'
```

Removing scaling limits using the CLI

To remove scaling limits using the CLI, set the Minimum and Maximum limit parameter to 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}'
```

Scaling ElastiCache for Memcached self-designed clusters

The amount of data your application needs to process is seldom static. It increases and decreases as your business grows or experiences normal fluctuations in demand. If you self-manage your cache, you need to provision sufficient hardware for your demand peaks, which can be expensive. By using Amazon ElastiCache you can scale to meet current demand, paying only for what you use. ElastiCache enables you to scale your cache to match demand.

The following helps you find the correct topic for the scaling actions that you want to perform.

Scaling Memcached Clusters

Action	Topic
Scaling out	Adding nodes to a cluster
Scaling in	Deleting nodes from a cluster
Changing node types	Scaling Memcached vertically

Memcached clusters are composed of 1 to 60 nodes. Scaling a Memcached cluster out and in is as easy as adding or removing nodes from the cluster.

If you need more than 60 nodes in a Memcached cluster, or more than 300 nodes total in an AWS Region, fill out the ElastiCache Limit Increase Request form at https://aws.amazon.com/contact-us/elasticache-node-limit-request/.

Because you can partition your data across all the nodes in a Memcached cluster, scaling up to a node type with greater memory is seldom required. However, because the Memcached engine does not persist data, if you do scale to a different node type, your new cluster starts out empty unless your application populates it.

Topics

- Scaling Memcached Horizontally

- Scaling Memcached vertically

Scaling Memcached Horizontally

The Memcached engine supports partitioning your data across multiple nodes. Because of this, Memcached clusters scale horizontally easily. A Memcached cluster can have from 1 to 60 nodes. To horizontally scale your Memcached cluster, merely add or remove nodes.

If you need more than 60 nodes in a Memcached cluster, or more than 300 nodes total in an AWS Region, fill out the ElastiCache Limit Increase Request form at https://aws.amazon.com/contact-us/elasticache-node-limit-request/.

The following topics detail how to scale your Memcached cluster out or in by adding or removing nodes.

- Adding nodes to a cluster
- Deleting nodes from your cluster

Each time you change the number of nodes in your Memcached cluster, you must re-map at least some of your keyspace so it maps to the correct node. For more detailed information on load balancing your Memcached cluster, see Configuring your ElastiCache client for efficient load balancing.

If you use auto discovery on your Memcached cluster, you do not need to change the endpoints in your application as you add or remove nodes. For more information on auto discovery, see Automatically identify nodes in your cluster If you do not use auto discovery, each time you change the number of nodes in your Memcached cluster you must update the endpoints in your application.

Scaling Memcached vertically

When you scale your Memcached cluster up or down, you must create a new cluster. Memcached clusters always start out empty unless your application populates it.

> ⚠ **Important**
>
> If you are scaling down to a smaller node type, be sure that the smaller node type is adequate for your data and overhead. For more information, see Select cache node size.

Topics

Scaling Memcached vertically (Console)

The following procedure walks you through scaling your cluster vertically using the ElastiCache console.

To scale a Memcached cluster vertically (console)

1. Create a new cluster with the new node type. For more information, see Creating a Memcached cluster (console).
2. In your application, update the endpoints to the new cluster's endpoints. For more information, see Finding a Cluster's Endpoints (Console).
3. Delete the old cluster. For more information, see Deleting a new node in Memcached.

Scaling Memcached vertically (AWS CLI)

The following procedure walks you through scaling your Memcached cache cluster vertically using the AWS CLI.

To scale a Memcached cache cluster vertically (AWS CLI)

1. Create a new cache cluster with the new node type. For more information, see Creating clusters with the CLI.
2. In your application, update the endpoints to the new cluster's endpoints. For more information, see Finding Endpoints (AWS CLI).
3. Delete the old cache cluster. For more information, see Using the AWS CLI.

Scaling Memcached vertically (ElastiCache API)

The following procedure walks you through scaling your Memcached cache cluster vertically using the ElastiCache API.

To scale a Memcached cache cluster vertically (ElastiCache API)

1. Create a new cache cluster with the new node type. For more information, see Creating a cluster (ElastiCache API)
2. In your application, update the endpoints to the new cache cluster's endpoints. For more information, see Finding Endpoints (ElastiCache API).
3. Delete the old cache cluster. For more information, see Using the ElastiCache API.

Tagging your ElastiCache resources

To help you manage your clusters and other ElastiCache resources, you can assign your own metadata to each resource in the form of tags. Tags enable you to categorize your AWS resources in different ways, for example, by purpose, owner, or environment. This is useful when you have many resources of the same type—you can quickly identify a specific resource based on the tags that you've assigned to it. This topic describes tags and shows you how to create them.

> ⚠ **Warning**
>
> As a best practice, we recommend that you do not include sensitive data in your tags.

Tag basics

A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value, both of which you define. Tags enable you to categorize your AWS resources in different ways, for example, by purpose or owner. For example, you could define a set of tags for your account's ElastiCache clusters that helps you track each instance's owner and user group.

We recommend that you devise a set of tag keys that meets your needs for each resource type. Using a consistent set of tag keys makes it easier for you to manage your resources. You can search and filter the resources based on the tags you add. For more information about how to implement an effective resource tagging strategy, see the AWS whitepaper Tagging Best Practices.

Tags don't have any semantic meaning to ElastiCache and are interpreted strictly as a string of characters. Also, tags are not automatically assigned to your resources. You can edit tag keys and values, and you can remove tags from a resource at any time. You can set the value of a tag to `null`. If you add a tag that has the same key as an existing tag on that resource, the new value overwrites the old value. If you delete a resource, any tags for the resource are also deleted. Furthermore, if you add or delete tags on a replication group, all nodes in that replication group will also have their tags added or removed.

You can work with tags using the AWS Management Console, the AWS CLI, and the ElastiCache API.

If you're using IAM, you can control which users in your AWS account have permission to create, edit, or delete tags. For more information, see Resource-level permissions.

Resources you can tag

You can tag most ElastiCache resources that already exist in your account. The table below lists the resources that support tagging. If you're using the AWS Management Console, you can apply tags to resources by using the Tag Editor. Some resource screens enable you to specify tags for a resource when you create the resource; for example, a tag with a key of Name and a value that you specify. In most cases, the console applies the tags immediately after the resource is created (rather than during resource creation). The console may organize resources according to the **Name** tag, but this tag doesn't have any semantic meaning to the ElastiCache service.

Additionally, some resource-creating actions enable you to specify tags for a resource when the resource is created. If tags cannot be applied during resource creation, we roll back the resource creation process. This ensures that resources are either created with tags or not created at all, and that no resources are left untagged at any time. By tagging resources at the time of creation, you can eliminate the need to run custom tagging scripts after resource creation.

If you're using the Amazon ElastiCache API, the AWS CLI, or an AWS SDK, you can use the `Tags` parameter on the relevant ElastiCache API action to apply tags. They are:

- `CreateServerlessCache`
- `CreateCacheCluster`
- `CreateCacheParameterGroup`
- `CreateCacheSecurityGroup`
- `CreateCacheSubnetGroup`
- `PurchaseReservedCacheNodesOffering`

The following table describes the ElastiCache resources that can be tagged, and the resources that can be tagged on creation using the ElastiCache API, the AWS CLI, or an AWS SDK.

Tagging support for ElastiCache resources

Supports tags	Supports tagging on creation
Yes	Yes

Supports tags	Supports tagging on creation
Yes	Yes
Yes	Yes
Yes	Yes
Yes	Yes
Yes	Yes

You can apply tag-based resource-level permissions in your IAM policies to the ElastiCache API actions that support tagging on creation to implement granular control over the users and groups that can tag resources on creation. Your resources are properly secured from creation—tags that are applied immediately to your resources. Therefore any tag-based resource-level permissions controlling the use of resources are immediately effective. Your resources can be tracked and reported on more accurately. You can enforce the use of tagging on new resources, and control which tag keys and values are set on your resources.

For more information, see Tagging resources examples.

For more information about tagging your resources for billing, see Monitoring costs with cost allocation tags.

Tag restrictions

The following basic restrictions apply to tags:

- Maximum number of tags per resource – 50
- For each resource, each tag key must be unique, and each tag key can have only one value.
- Maximum key length – 128 Unicode characters in UTF-8.
- Maximum value length – 256 Unicode characters in UTF-8.

- Although ElastiCache allows for any character in its tags, other services can be restrictive. The allowed characters across services are: letters, numbers, and spaces representable in UTF-8, and the following characters: + - = . _ : / @
- Tag keys and values are case-sensitive.
- The `aws:` prefix is reserved for AWS use. If a tag has a tag key with this prefix, then you can't edit or delete the tag's key or value. Tags with the `aws:` prefix do not count against your tags per resource limit.

You can't terminate, stop, or delete a resource based solely on its tags; you must specify the resource identifier. For example, to delete snapshots that you tagged with a tag key called `DeleteMe`, you must use the `DeleteSnapshot` action with the resource identifiers of the snapshots, such as `snap-1234567890abcdef0`.

For more information on ElastiCache resources you can tag, see Resources you can tag.

Tagging resources examples

- Creating a serverless cache using tags

```
aws elasticache create-serverless-cache \
    --serverless-cache-name CacheName \
    --engine memcached
    --tags Key="Cost Center", Value="1110001" Key="project",Value="XYZ"
```

- Adding tags to a serverless cache

```
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"
```

- Creating a Cache Cluster using tags.

```
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"
```

Tag-Based access control policy examples

1. Allowing `AddTagsToResource` action to a cluster only if the cluster has the tag Project=XYZ.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticache:AddTagsToResource",
            "Resource": [
                "arn:aws:elasticache:*:*:cluster:*"
            ],
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Project": "XYZ"
                }
            }
        }
    ]
}
```

2. Allowing to `RemoveTagsFromResource` action from a replication group if it contains the tags Project and Service and keys are different from Project and Service.

```
{
    "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. Allowing `AddTagsToResource` to any resource only if tags are different from Project and Service.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "elasticache:AddTagsToResource",
            "Resource": [
                "arn:aws:elasticache:*:*:*:*"
            ],
            "Condition": {
                "ForAnyValue:StringNotEqualsIgnoreCase": {
                    "aws:TagKeys": [
                        "Service",
                        "Project"
                    ]
                }
            }
        }
    ]
}
```

4. Denying `CreateCacheCluster` action if the request tag `Project` is missing or is not equal to `Dev`, `QA` or `Prod`.

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

For related information on condition keys, see Using condition keys.

Monitoring costs with cost allocation tags

When you add cost allocation tags to your resources in Amazon ElastiCache, you can track costs by grouping expenses on your invoices by resource tag values.

An ElastiCache cost allocation tag is a key-value pair that you define and associate with an ElastiCache resource. The key and value are case-sensitive. You can use a tag key to define a category, and the tag value can be an item in that category. For example, you might define a tag key of `CostCenter` and a tag value of `10010`, indicating that the resource is assigned to the 10010 cost center. You can also use tags to designate resources as being used for test or production by using a key such as `Environment` and values such as `test` or `production`. We recommend that you use a consistent set of tag keys to make it easier to track costs associated with your resources.

Use cost allocation tags to organize your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. Then, to see the cost of combined resources, organize your billing information according to resources with the same tag key values. For example, you can tag several resources with a specific application name, and then organize your billing information to see the total cost of that application across several services.

You can also combine tags to track costs at a greater level of detail. For example, to track your service costs by region you might use the tag keys `Service` and `Region`. On one resource you might have the values `ElastiCache` and `Asia Pacific (Singapore)`, and on another resource the values `ElastiCache` and `Europe (Frankfurt)`. You can then see your total ElastiCache costs broken out by region. For more information, see Use Cost Allocation Tags in the *AWS Billing User Guide*.

You can add ElastiCache cost allocation tags to Memcached clusters. When you add, list, modify, copy, or remove a tag, the operation is applied only to the specified cluster.

Characteristics of ElastiCache cost allocation tags

- Cost allocation tags are applied to ElastiCache resources which are specified in CLI and API operations as an ARN. The resource-type will be a "cluster".

  Sample ARN: `arn:aws:elasticache:`*`<region>`*`:`*`<customer-id>`*`:`*`<resource-type>`*`:`*`<resource-name>`*

  Memcached: Tags are only applied to clusters.

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

- The tag key is the required name of the tag. The key's string value can be from 1 to 128 Unicode characters long and cannot be prefixed with `aws:`. The string can contain only the set of Unicode letters, digits, blank spaces, underscores (_), periods (.), colons (:), backslashes (\ ), equal signs (=), plus signs (+), hyphens (-), or at signs (@).
- The tag value is the optional value of the tag. The value's string value can be from 1 to 256 Unicode characters in length and cannot be prefixed with `aws:`. The string can contain only the set of Unicode letters, digits, blank spaces, underscores (_), periods (.), colons (:), backslashes (\ ), equal signs (=), plus signs (+), hyphens (-), or at signs (@).
- An ElastiCache resource can have a maximum of 50 tags.
- Values do not have to be unique in a tag set. For example, you can have a tag set where the keys `Service` and `Application` both have the value `ElastiCache`.

AWS does not apply any semantic meaning to your tags. Tags are interpreted strictly as character strings. AWS does not automatically set any tags on any ElastiCache resource.

Managing your cost allocation tags using the AWS CLI

You can use the AWS CLI to add, modify, or remove cost allocation tags.

Cost allocation tags are applied to ElastiCache for Memcached clusters. The cluster to be tagged is specified using an ARN (Amazon Resource Name).

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

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

Topics

- Removing tags using the AWS CLI

Listing tags using the AWS CLI

You can use the AWS CLI to list tags on an existing ElastiCache resource by using the list-tags-for-resource operation.

The following code uses the AWS CLI to list the tags on the Memcached cluster `my-cluster` in the us-west-2 region.

For Linux, macOS, or Unix:

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

For Windows:

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

Output from this operation will look something like the following, a list of all the tags on the resource.

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

If there are no tags on the resource, the output will be an empty TagList.

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

For more information, see the AWS CLI for ElastiCache list-tags-for-resource.

Adding tags using the AWS CLI

You can use the AWS CLI to add tags to an existing ElastiCache resource by using the add-tags-to-resource CLI operation. If the tag key does not exist on the resource, the key and value are added to the resource. If the key already exists on the resource, the value associated with that key is updated to the new value.

The following code uses the AWS CLI to add the keys `Service` and `Region` with the values `elasticache` and `us-west-2` respectively to the cluster `my-cluster` in region us-west-2.

For Linux, macOS, or 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
```

For 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
```

Output from this operation will look something like the following, a list of all the tags on the resource following the operation.

```
{
   "TagList": [
      {
         "Value": "elasticache",
         "Key": "Service"
      },
      {
         "Value": "us-west-2",
         "Key": "Region"
      }
```

```
    ]
}
```

For more information, see the AWS CLI for ElastiCache add-tags-to-resource.

You can also use the AWS CLI to add tags to a cluster when you create a new cluster by using the operation create-cache-cluster. You cannot add tags when creating a cluster using the ElastiCache management console. After the cluster is created, you can then use the console to add tags to the cluster.

Modifying tags using the AWS CLI

You can use the AWS CLI to modify the tags on an ElastiCache for Memcached cluster.

To modify tags:

- Use add-tags-to-resource to either add a new tag and value or to change the value associated with an existing tag.
- Use remove-tags-from-resource to remove specified tags from the resource.

Output from either operation will be a list of tags and their values on the specified cluster.

Removing tags using the AWS CLI

You can use the AWS CLI to remove tags from an existing ElastiCache for Memcached cluster by using the remove-tags-from-resource operation.

The following code uses the AWS CLI to remove the tags with the keys `Service` and `Region` from the cluster `my-cluster` in the us-west-2 region.

For Linux, macOS, or Unix:

```
aws elasticache remove-tags-from-resource \
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster \
 --tag-keys PM Service
```

For Windows:

```
aws elasticache remove-tags-from-resource ^
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:my-cluster ^
 --tag-keys PM Service
```

Output from this operation will look something like the following, a list of all the tags on the resource following the operation.

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

For more information, see the AWS CLI for ElastiCache remove-tags-from-resource.

Managing your cost allocation tags using the ElastiCache API

You can use the ElastiCache API to add, modify, or remove cost allocation tags.

Cost allocation tags are applied to ElastiCache for Memcached clusters. The cluster to be tagged is specified using an ARN (Amazon Resource Name).

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

Topics

- Listing tags using the ElastiCache API
- Adding tags using the ElastiCache API
- Modifying tags using the ElastiCache API
- Removing tags using the ElastiCache API

Listing tags using the ElastiCache API

You can use the ElastiCache API to list tags on an existing resource by using the ListTagsForResource operation.

The following code uses the ElastiCache API to list the tags on the resource `my-cluster` in the us-west-2 region.

```
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>
```

Adding tags using the ElastiCache API

You can use the ElastiCache API to add tags to an existing ElastiCache cluster by using the AddTagsToResource operation. If the tag key does not exist on the resource, the key and value are added to the resource. If the key already exists on the resource, the value associated with that key is updated to the new value.

The following code uses the ElastiCache API to add the keys `Service` and `Region` with the values `elasticache` and `us-west-2` respectively to the resource `my-cluster` in the us-west-2 region.

```
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>
```

For more information, see AddTagsToResource in the *Amazon ElastiCache API Reference*.

Modifying tags using the ElastiCache API

You can use the ElastiCache API to modify the tags on an ElastiCache cluster.

To modify the value of a tag:

- Use AddTagsToResource operation to either add a new tag and value or to change the value of an existing tag.
- Use RemoveTagsFromResource to remove tags from the resource.

Output from either operation will be a list of tags and their values on the specified resource.

Use RemoveTagsFromResource to remove tags from the resource.

Removing tags using the ElastiCache API

You can use the ElastiCache API to remove tags from an existing ElastiCache for Memcached cluster by using the RemoveTagsFromResource operation.

The following code uses the ElastiCache API to remove the tags with the keys `Service` and `Region` from the cluster `my-cluster` in region us-west-2.

```
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>
```

Using the Amazon ElastiCache Well-Architected Lens

This section describes the Amazon ElastiCache Well-Architected Lens, a collection of design principles and guidance for designing well-architected ElastiCache workloads.

- The ElastiCache Lens is additive to the AWS Well-Architected Framework.
- Each Pillar has a set of questions to help start the discussion around an ElastiCache Architecture.
  - Each question has a number of leading practices along with their scores for reporting.
    - *Required* - Necessary before going to prod (absent being a high risk)
    - *Best* - Best possible state a customer could be
    - *Good* - What we recommend customers to have (absent being a medium risk)
- Well-Architected terminology
  - Component – Code, configuration and AWS Resources that together deliver against a requirement. Components interact with other components, and often equate to a service in microservice architectures.
  - Workload A set of components that together deliver business value. Examples of workloads are marketing websites, e-commerce websites, the back-ends for a mobile app, analytic platforms, etc.

Topics

Amazon ElastiCache Well-Architected Lens Operational Excellence Pillar

The operational excellence pillar focuses on running and monitoring systems to deliver business value, and continually improving processes and procedures. Key topics include automating changes, responding to events, and defining standards to manage daily operations.

Topics

OE 1: How do you understand and respond to alerts and events triggered by your ElastiCache cluster?

Question-level introduction: When you operate ElastiCache clusters you can optionally receive notifications and alerts when specific events occur. ElastiCache, by default, logs events that relate to your resources, such as a failover, node replacement, scaling operation, scheduled maintenance, and more. Each event includes the date and time, the source name and source type, and a description.

Question-level benefit: Being able to understand and manage the underlying reasons behind the events that trigger alerts generated by your cluster enables you to operate more effectively and respond to events appropriately.

- **[Required]** Review the events generated by ElastiCache on the ElastiCache console (after selecting your region) or using the Amazon Command Line Interface (AWS CLI) describe-events command and the ElastiCache API. Configure ElastiCache to send notifications for important cluster events using Amazon Simple Notification Service (Amazon SNS). Using Amazon SNS with your clusters allows you to programmatically take actions upon ElastiCache events.
  - There are two large categories of events: current and scheduled events. The list of current events includes: resource creation and deletion, scaling operations, failover, node reboot, snapshot created, cluster's parameter modification, CA certificate renewal, failure events (cluster provisioning failure - VPC or ENI-, scaling failures - ENI-, and snapshot failures). The list of scheduled events includes: node scheduled for replacement during the maintenance window and node replacement rescheduled.
  - Although you may not need to react immediately to some of these events, it is critical to first look at all failure events:
    - ElastiCache:AddCacheNodeFailed
    - ElastiCache:CacheClusterProvisioningFailed
    - ElastiCache:CacheClusterScalingFailed
    - ElastiCache:CacheNodesRebooted
    - ElastiCache:SnapshotFailed (Redis only)
  - **[Resources]:**
    - Managing ElastiCache Amazon SNS notifications
    - Event Notifications and Amazon SNS
- **[Best]** To automate responses to events, leverage AWS products and services capabilities such as SNS and Lambda Functions. Follow best practices by making small, frequent, reversible changes, as code to evolve your operations over time. You should use Amazon CloudWatch metrics to monitor your clusters.

  [Resources]: Monitor Amazon ElastiCache for Redis (cluster mode disabled) read replica endpoints using AWS Lambda, Amazon Route 53, and Amazon SNS for a use case that uses Lambda and SNS.

OE 2: When and how do you scale your existing ElastiCache clusters?

Question-level introduction: Right-sizing your ElastiCache cluster is a balancing act that needs to be evaluated every time there are changes to the underlying workload types. Your objective is to operate with the right sized environment for your workload.

Question-level benefit: Over-utilization of your resources may result in elevated latency and overall decreased performance. Under-utilization, on the other hand, may result in over-provisioned resources at non-optimal cost optimization. By right-sizing your environments you can strike a balance between performance efficiency and cost optimization. To remediate over or under utilization of your resources, ElastiCache can scale in two dimensions. You can scale vertically by increasing or decreasing node capacity. You can also scale horizontally by adding and removing nodes.

- **[Required]** CPU and network over-utilization on primary nodes should be addressed by offloading and redirecting the read operations to replica nodes. Use replica nodes for read operations to reduce primary node utilization. This can be configured in your Redis client library by connecting to the ElastiCache reader endpoint for cluster mode disabled, or by using the Redis READONLY command for cluster mode enabled.

  [Resources]:

  - Finding connection endpoints
  - Cluster Right-Sizing
  - Redis READONLY Command
- **[Required]** Monitor the utilization of critical cluster resources such as CPU, memory, and network. The utilization of these specific cluster resources needs to be tracked to inform your decision to scale, and the type of scaling operation. For ElastiCache for Redis cluster mode disabled, primary and replica nodes can scale vertically. Replica nodes can also scale horizontally from 0 to 5 nodes. For cluster mode enabled, the same applies within each shard of your cluster. In addition, you can increase or reduce the number of shards.

  [Resources]:

  - Monitoring best practices with Amazon ElastiCache for Redis using Amazon CloudWatch
  - Scaling ElastiCache for Redis Clusters
  - Scaling ElastiCache for Memcached Clusters
- **[Best]** Monitoring trends over time can help you detect workload changes that would remain unnoticed if monitored at a particular point in time. To detect longer term trends, use

CloudWatch metrics to scan for longer time ranges. The learnings from observing extended periods of CloudWatch metrics should inform your forecast around cluster resources utilization. CloudWatch data points and metrics are available for up to 455 days.

[Resources]:

- Monitoring ElastiCache for Redis with CloudWatch Metrics
- Monitoring Memcached with CloudWatch Metrics
- Monitoring best practices with Amazon ElastiCache for Redis using Amazon CloudWatch

- **[Best]** If your ElastiCache resources are created with CloudFormation it is best practice to perform changes using CloudFormation templates to preserve operational consistency and avoid unmanaged configuration changes and stack drifts.

  [Resources]:

  - ElastiCache resource type reference for CloudFormation

- **[Best]** Automate your scaling operations using cluster operational data and define thresholds in CloudWatch to setup alarms. Use CloudWatch Events and Simple Notification Service (SNS) to trigger Lambda functions and execute an ElastiCache API to scale your clusters automatically. An example would be to add a shard to your cluster when the `EngineCPUUtilization` metric reaches 80% for an extended period of time. Another option would be to use `DatabaseMemoryUsedPercentages` for a memory-based threshold.

  [Resources]:

  - Using Amazon CloudWatch Alarms
  - What are Amazon CloudWatch events?
  - Using AWS Lambda with Amazon Simple Notification Service
  - ElastiCache API Reference

OE 3: How do you manage your ElastiCache cluster resources and maintain your cluster up-to-date?

Question-level introduction: When operating at scale, it is essential that you are able to pinpoint and identify all your ElastiCache resources. When rolling out new application features you need to create cluster version symmetry across all your ElastiCache environment types: dev, testing, and production. Resource attributes allow you to separate environments for different operational objectives, such as when rolling out new features and enabling new security mechanisms.

Question-level benefit: Separating your development, testing, and production environments is best operational practice. It is also best practice that your clusters and nodes across environments have the latest software patches applied using well understood and documented processes. Taking advantage of native ElastiCache features enables your engineering team to focus on meeting business objectives and not on ElastiCache maintenance.

- **[Best]** Run on the latest engine version available and apply the Self-Service Updates as quickly as they become available. ElastiCache automatically updates its underlying infrastructure during your specified maintenance window of the cluster. However, the nodes running in your clusters are updated via Self-Service Updates. These updates can be of two types: security patches or minor software updates. Ensure you understand the difference between types of patches and when they are applied.

  [Resources]:

  - Self-Service Updates in Amazon ElastiCache
  - Amazon ElastiCache Managed Maintenance and Service Updates Help Page
- **[Best]** Organize your ElastiCache resources using tags. Use tags on replication groups and not on individual nodes. You can configure tags to be displayed when you query resources and you can use tags to perform searches and apply filters. You should use Resource Groups to easily create and maintain collections of resources that share common sets of tags.

  [Resources]:

  - Tagging Best Practices
  - ElastiCache resource type reference for CloudFormation
  - Parameter Groups

OE 4: How do you manage clients’ connections to your ElastiCache clusters?

Question-level introduction: When operating at scale you need to understand how your clients connect with the ElastiCache cluster to manage your application operational aspects (such as response times).

Question-level benefit: Choosing the most appropriate connection mechanism ensures that your application does not disconnect due to connectivity errors, such as time-outs.

- **[Required]** Separate read from write operations and connect to the replica nodes to execute read operations. However, be aware when you separate the writes from the reads you will lose the ability to read a key immediately after writing it due to the asynchronous nature of Redis replication. The WAIT command can be leveraged to improve real world data safety and force replicas to acknowledge writes before responding to clients, at an overall performance cost. Using replica nodes for read operations can be configured in your ElastiCache for Redis client library using the ElastiCache reader endpoint for cluster mode disabled. For cluster mode enabled, use the ElastiCache for Redis READONLY command. For many of the ElastiCache for Redis client libraries, ElastiCache for Redis READONLY is implemented by default or via a configuration setting.

[Resources]:

  - Finding connection endpoints
  - READONLY

- **[Required]** Use connection pooling. Establishing a TCP connection has a cost in CPU time on both client and server sides and pooling allows you to reuse the TCP connection.

  To reduce connection overhead, you should use connection pooling. With a pool of connections your application can re-use and release connections ‘at will’, without the cost of establishing the connection. You can implement connection pooling via your ElastiCache for Redis client library (if supported), with a Framework available for your application environment, or build it from the ground up.

- **[Best]** Ensure that the socket timeout of the client is set to at least one second (vs. the typical “none” default in several clients).

  - Setting the timeout value too low can lead to possible timeouts when the server load is high. Setting it too high can result in your application taking a long time to detect connection issues.
  - Control the volume of new connections by implementing connection pooling in your client application. This reduces latency and CPU utilization needed to open and close connections, and perform a TLS handshake if TLS is enabled on the cluster.

  [Resources]: Configure Amazon ElastiCache for Redis for higher availability

- **[Good]** Using pipelining (when your use cases allow it) can significantly boost the performance.

  - With pipelining you reduce the Round-Trip Time (RTT) between your application clients and the cluster and new requests can be processed even if the client has not yet read the previous responses.
  - With pipelining you can send multiple commands to the server without waiting for replies/ack. The downside of pipelining is that when you eventually fetch all the responses in bulk there may have been an error that you will not catch until the end.

- Implement methods to retry requests when an error is returned that omits the bad request.

  [Resources]: Pipelining

OE 5: How do you deploy ElastiCache Components for a Workload?

Question-level introduction: ElastiCache environments can be deployed manually through the AWS Console, or programmatically through APIs, CLI, toolkits, etc. Operational Excellence best practices suggest automating deployments through code whenever possible. Additionally, ElastiCache clusters can either be isolated by workload or combined for cost optimization purposes.

Question-level benefit: Choosing the most appropriate deployment mechanism for your ElastiCache environments can improve Operation Excellence over time. It is recommended to perform operations as code whenever possible to minimize human error and increase repeatability, flexibility, and response time to events.

By understanding the workload isolation requirements, you can choose to have dedicated ElastiCache environments per workload or combine multiple workloads into single clusters, or combinations thereof. Understanding the tradeoffs can help strike a balance between Operational Excellence and Cost Optimization

- **[Required]** Understand the deployment options available to ElastiCache, and automate these procedures whenever possible. Possible avenues of automation include CloudFormation, AWS CLI/SDK, and APIs.

  [Resources]:

  - Amazon ElastiCache resource type reference
  - elasticache
  - Amazon ElastiCache API Reference
- **[Required]** For all workloads determine the level of cluster isolation needed.
  - **[Best]:** High Isolation – a 1:1 workload to cluster mapping. Allows for finest grained control over access, sizing, scaling, and management of ElastiCache resources on a per workload basis.
  - **[Better]:** Medium Isolation – M:1 isolated by purpose but perhaps shared across multiple workloads (for example a cluster dedicated to caching workloads, and another dedicated for messaging).
  - **[Good]:** Low Isolation – M:1 all purpose, fully shared. Recommended for workloads where shared access is acceptable.

OE 6: How do you plan for and mitigate failures?

Question-level introduction: Operational Excellence includes anticipating failures by performing regular "pre-mortem" exercises to identify potential sources of failure so they can be removed or mitigated. ElastiCache offers a Failover API that allows for simulated node failure events, for testing purposes.

Question-level benefit: By testing failure scenarios ahead of time you can learn how they impact your workload. This allows for safe testing of response procedures and their effectiveness, as well as gets your team familiar with their execution.

[Required] Regularly perform failover testing in dev/test accounts. TestFailover

OE 7: How do you troubleshoot Redis engine events?

Question-level introduction: Operational Excellence requires the ability to investigate both service-level and engine-level information to analyze the health and status of your clusters. Amazon ElastiCache for Redis can emit Redis engine logs to both Amazon CloudWatch and Amazon Kinesis Data Firehose.

Question-level benefit: Enabling Redis engine logs on Amazon ElastiCache for Redis clusters provides insight into events that impact the health and performance of clusters. Redis engine logs provide data directly from the Redis engine that is not available through the ElastiCache events mechanism. Through careful observation of both ElastiCache events (see preceding OE-1) and Redis engine logs, it is possible to determine an order of events when troubleshooting from both the ElastiCache service perspective and Redis engine perspective.

- **[Required]** Ensure that Redis engine logging functionality is enabled, which is available as of ElastiCache for Redis 6.2 and newer. This can be performed during cluster creation or by modifying the cluster after creation.
  - Determine whether Amazon CloudWatch Logs or Amazon Kinesis Data Firehose is the appropriate target for Redis engine logs.
  - Select an appropriate target log within either CloudWatch or Kinesis Data Firehose to persist the logs. If you have multiple clusters, consider a different target log for each cluster as this will help isolate data when troubleshooting.

  [Resources]:

  - Log delivery: Log delivery
  - Logging destinations: Amazon CloudWatch Logs

  - Amazon CloudWatch Logs introduction: What is Amazon CloudWatch Logs?
  - Amazon Kinesis Data Firehose introduction: What Is Amazon Kinesis Data Firehose?
- **[Best]** If using Amazon CloudWatch Logs, consider leveraging Amazon CloudWatch Logs Insights to query Redis engine log for important information.

  As an example, create a query against the CloudWatch Log group that contains the Redis engine logs that will return events with a LogLevel of 'WARNING', such as:

  ```
  fields @timestamp, LogLevel, Message
  | sort @timestamp desc
  | filter LogLevel = "WARNING"
  ```

  [Resources]: Analyzing log data with CloudWatch Logs Insights

Amazon ElastiCache Well-Architected Lens Security Pillar

The security pillar focuses on protecting information and systems. Key topics include confidentiality and integrity of data, identifying and managing who can do what with privilege-based management, protecting systems, and establishing controls to detect security events.

Topics

SEC 1: What steps are you taking in controlling authorized access to ElastiCache data?

Question-level introduction: All ElastiCache clusters are designed to be accessed from Amazon Elastic Compute Cloud instances in a VPC, serverless functions (AWS Lambda), or containers

(Amazon Elastic Container Service). The most encountered scenario is to access an ElastiCache cluster from an Amazon Elastic Compute Cloud instance within the same Amazon Virtual Private Cloud (Amazon Virtual Private Cloud). Before you can connect to a cluster from an Amazon EC2 instance, you must authorize the Amazon EC2 instance to access the cluster. To access an ElastiCache cluster running in a VPC, it is necessary to grant network ingress to the cluster.

Question-level benefit: Network ingress into the cluster is controlled via VPC security groups. A security group acts as a virtual firewall for your Amazon EC2 instances to control incoming and outgoing traffic. Inbound rules control the incoming traffic to your instance, and outbound rules control the outgoing traffic from your instance. In the case of ElastiCache, when launching a cluster, it requires associating a security group. This ensures that inbound and outbound traffic rules are in place for all nodes that make up the cluster. Additionally, ElastiCache is configured to deploy on private subnets exclusively such that they are only accessible from via the VPC's private networking.

- **[Required]** The security group associated with your cluster controls network ingress and access to the cluster. By default, a security group will not have any inbound rules defined and, therefore, no ingress path to ElastiCache. To enable this, configure an inbound rule on the security group specifying source IP address/range, TCP type traffic and the port for your ElastiCache cluster (default port 6379 for ElastiCache for Redis for example). While it is possible to allow a very broad set of ingress sources, like all resources within a VPC (0.0.0.0/0), it is advised to be as granular as possible in defining the inbound rules such as authorizing only inbound access to Redis clients running on Amazon Amazon EC2 instances associated with a specific security group.

  [Resources]:

  - Subnets and subnet groups
  - Accessing your cluster or replication group
  - Control traffic to resources using security groups
  - Amazon Elastic Compute Cloud security groups for Linux instances
- **[Required]** AWS Identity and Access Management policies can be assigned to AWS Lambda functions allowing them to access ElastiCache data. To enable this feature, create an IAM execution role with the `AWSLambdaVPCAccessExecutionRole` permission, then assign the role to the AWS Lambda function.

  [Resources]: Configuring a Lambda function to access Amazon ElastiCache in an Amazon VPC: Tutorial: Configuring a Lambda function to access Amazon ElastiCache in an Amazon VPC

SEC 2: Do your applications require additional authorization to ElastiCache over and above networking-based controls?

Question-level introduction: In scenarios where it is necessary to restrict or control access to ElastiCache for Redis clusters at an individual client level, it is recommended to authenticate via the ElastiCache for Redis AUTH command. ElastiCache for Redis authentication tokens, with optional user and user group management, enable ElastiCache for Redis to require a password before allowing clients to run commands and access keys, thereby improving data plane security.

Question-level benefit: To help keep your data secure, ElastiCache for Redis provides mechanisms to safeguard against unauthorized access of your data. This includes enforcing Role-Based Access Control (RBAC) AUTH, or AUTH token (password) be used by clients to connect to ElastiCache before performing authorized commands.

- **[Best]** For ElastiCache for Redis 6.x and higher, define authentication and authorization controls by defining user groups, users, and access strings. Assign users to user groups, then assign user groups to clusters. To utilize RBAC, it must be selected upon cluster creation, and in-transit encryption must be enabled. Ensure you are using a Redis client that supports TLS to be able to leverage RBAC.

  [Resources]:

  - Applying RBAC to a Replication Group for ElastiCache for Redis
  - Specifying Permissions Using an Access String
  - ACL
  - Supported ElastiCache for Redis versions
- **[Best]** For ElastiCache for Redis versions prior to 6.x, in addition to setting strong token/password and maintaining a strict password policy for ElastiCache for Redis AUTH, it is best practice to rotate the password/token. ElastiCache can manage up to two (2) authentication tokens at any given time. You can also modify the cluster to explicitly require the use of authentication tokens.

  [Resources]: Modifying the AUTH token on an existing ElastiCache for Redis cluster

SEC 3: Is there a risk that commands can be executed inadvertently, causing data loss or failure?

Question-level introduction: There are a number of Redis commands that can have adverse impacts on operations if executed by mistake or by malicious actors. These commands can have unintended consequences from a performance and data safety perspective. For example a developer may routinely call the FLUSHALL command in a dev environment, and due to a mistake may inadvertently attempt to call this command on a production system, resulting in accidental data loss.

Question-level benefit: Beginning with ElastiCache for Redis 5.0.3 on ElastiCache, you have the ability to rename certain commands that might be disruptive to your workload. Renaming the commands can help prevent them from being inadvertently executed on the cluster.

- **[Required]**

  [Resources]:

  - ElastiCache for Redis version 5.0.3 (deprecated, use version 5.0.6)
  - Redis 5.0.3 parameter changes
  - Redis security

SEC 4: How do you ensure data encryption at rest with ElastiCache

Question-level introduction: While ElastiCache for Redis is an in-memory data store, it is possible to encrypt any data that may be persisted (on storage) as part of standard operations of the cluster. This includes both scheduled and manual backups written to Amazon S3, as well as data saved to disk storage as a result of sync and swap operations. Instance types in the M6g and R6g families also feature always-on, in-memory encryption.

Question-level benefit: ElastiCache for Redis provides optional encryption at-rest to increase data security.

- **[Required]** At-rest encryption can be enabled on an ElastiCache cluster (replication group) only when it is created. An existing cluster cannot be modified to begin encrypting data at-rest. By default, ElastiCache will provide and manage the keys used in at-rest encryption.

  [Resources]:

  - At-Rest Encryption Conditions

  - Enabling At-Rest Encryption
- **[Best]** Leverage Amazon EC2 instance types that encrypt data while it is in memory (such as M6g or R6g). Where possible, consider managing your own keys for at-rest encryption. For more stringent data security environments, AWS Key Management Service (KMS) can be used to self-manage Customer Master Keys (CMK). Through ElastiCache integration with AWS Key Management Service, you are able to create, own, and manage the keys used for encryption of data at rest for your ElastiCache for Redis cluster.

  [Resources]:

  - Using customer managed keys from AWS Key Management Service
  - AWS Key Management Service
  - AWS KMS concepts

SEC 5: How do you encrypt in-transit data with ElastiCache?

Question-level introduction: It is a common requirement to mitigate against data being compromised while in transit. This represents data within components of a distributed system, as well as between application clients and cluster nodes. ElastiCache for Redis supports this requirement by allowing for encrypting data in-transit between clients and cluster, and between cluster nodes themselves. Instance types in the M6g and R6g families also feature always-on, in-memory encryption.

Question-level benefit: Amazon ElastiCache in-transit encryption is an optional feature that allows you to increase the security of your data at its most vulnerable points, when it is in-transit from one location to another.

- **[Required]** In-transit encryption can only be enabled on an ElastiCache for Redis cluster (replication group) upon creation. Please note that, due to the additional processing required for encrypting/decrypting data, implementing in-transit encryption will have some performance impact. To understand the impact, it is recommended to benchmark your workload before and after enabling encryption-in-transit.

  [Resources]:

  - In-transit encryption overview

SEC 6: How do you restrict access to control plane resources?

Question-level introduction: IAM policies and ARN enable fine grained access controls for ElastiCache for Redis, allowing for tighter control to manage the creation, modification and deletion of ElastiCache for Redis clusters.

Question-level benefit: Management of Amazon ElastiCache resources, such as replication groups, nodes, etc. can be constrained to AWS accounts that have specific permissions based on IAM policies, improving security and reliability of resources.

- **[Required]** Manage access to Amazon ElastiCache resources by assigning specific AWS Identity and Access Managementpolicies to AWS users, allowing finer control over which accounts can perform what actions on clusters.

  [Resources]:

  - Overview of managing access permissions to your ElastiCache resources
  - Using identity-based policies (IAM policies) for Amazon ElastiCache

SEC 7: How do you detect and respond to security events?

Question-level introduction: ElastiCache, when deployed with RBAC enabled, exports CloudWatch metrics to notify users of security events. These metrics help identify failed attempts to authenticate, access keys, or run commands that connecting RBAC users are not authorized for.

Additionally, AWS products and services resources help secure your overall workload by automating deployments and logging all actions and modifications for later review/audit.

Question-level benefit: By monitoring events, you enable your organization to respond according to your requirements, policies, and procedures. Automating the monitoring and responses to these security events hardens your overall security posture.

- **[Required]** Familiarize yourself with the CloudWatch Metrics published that pertain to RBAC authentication and authorization failures.
  - AuthenticationFailures = Failed attempts to authenticate to Redis
  - KeyAuthorizationFailures = Failed attempts by users to access keys without permission
  - CommandAuthorizationFailures = Failed attempts by users to run commands without permission

[Resources]:

- Metrics for Redis

- **[Best]** It is recommended to setup alerts and notifications on these metrics and respond as necessary.

  [Resources]:

  - Using Amazon CloudWatch alarms

- **[Best]** Use the Redis ACL LOG command to gather further details

  [Resources]:

  - ACL LOG

- **[Best]** Familiarize yourself with the AWS products and services capabilities as it pertains to monitoring, logging, and analyzing ElastiCache deployments and events

  [Resources]:

  - Logging Amazon ElastiCache API calls with AWS CloudTrail
  - elasticache-redis-cluster-automatic-backup-check
  - Monitoring use with CloudWatch Metrics

Amazon ElastiCache Well-Architected Lens Reliability Pillar

Topics

REL 1: How are you supporting high availability (HA) architecture deployments?

Question-level introduction: Understanding the high availability architecture of Amazon ElastiCache will enable you to operate in a resilient state during availability events.

Question-level benefit: Architecting your ElastiCache clusters to be resilient to failures ensures higher availability for your ElastiCache deployments.

- **[Required]** Determine the level of reliability you require for your ElastiCache cluster. Different workloads have different resiliency standards, from entirely ephemeral to mission critical workloads. Define needs for each type of environment you operate such as dev, test, and production.

  Caching engine: Memcached vs ElastiCache for Redis

  1. Memcached does not provide any replication mechanism and is used primarily for ephemeral workloads.
  2. ElastiCache for Redis offers HA features discussed below
- **[Best]** For workloads that require HA, use ElastiCache for Redis in cluster mode with a minimum of two replicas per shard, even for small throughput requirement workloads that require only one shard.
  1. For cluster mode enabled, multi-AZ is enabled automatically.

     Multi-AZ minimizes downtime by performing automatic failovers from primary node to replicas, in case of any planned or unplanned maintenance as well as mitigating AZ failure.
  2. For sharded workloads, a minimum of three shards provides faster recovery during failover events as the Redis Cluster Protocol requires a majority of primary nodes be available to achieve quorum.
  3. Set up two or more replicas across Availability.

     Having two replicas provides improved read scalability and also read availability in scenarios where one replica is undergoing maintenance.
  4. Use Graviton2-based node types (default nodes in most regions).

     Amazon ElastiCache for Redis has added optimized performance on these nodes. As a result, you get better replication and synchronization performance, resulting in overall improved availability.
  5. Monitor and right-size to deal with anticipated traffic peaks: under heavy load, the ElastiCache for Redis engine may become unresponsive, which affects availability. `BytesUsedForCache` and `DatabaseMemoryUsagePercentage` are good indicators of your memory usage, whereas `ReplicationLag` is an indicator of your replication health based on your write rate. You can use these metrics to trigger cluster scaling.

6. Ensure client-side resiliency by testing with the Failover API prior to a production failover event.

[Resources]:

- Configure Amazon ElastiCache for Redis for higher availability
- High availability using replication groups

REL 2: How are you meeting your Recovery Point Objectives (RPOs) with ElastiCache?

Question-level introduction: Understand workload RPO to inform decisions on ElastiCache backup and recovery strategies.

Question-level benefit: Having an in-place RPO strategy can improve business continuity in the event of a disaster recovery scenarios. Designing your backup and restore policies can help you meet your Recovery Point Objectives (RPO) for your ElastiCache data. ElastiCache for Redis offers snapshot capabilities which are stored in Amazon S3, along with a configurable retention policy. These snapshots are taken during a defined backup window, and handled by the service automatically. If your workload requires additional backup granularity, you have the option to create up to 20 manual backups per day. Manually created backups do not have a service retention policy and can be kept indefinitely.

- **[Required]** Understand and document the RPO of your ElastiCache deployments.
  - Be aware that Memcached does not offer any backup processes.
  - Review the capabilities of ElastiCache Backup and Restore features.
- **[Best]** Have a well-communicated process in place for backing up your cluster.
  - Initiate manual backups on an as-needed basis.
  - Review retention policies for automatic backups.
  - Note that manual backups will be retained indefinitely.
  - Schedule your automatic backups during periods of low usage.
  - Perform backup operations against read-replicas to ensure you minimize the impact on cluster performance.
- **[Good]** Leverage the scheduled backup feature of ElastiCache to regularly back up your data during a defined window.
  - Periodically test restores from your backups.

- **[Resources]:**
  - Redis
  - Backup and restore for ElastiCache for Redis
  - Making manual backups
  - Scheduling automatic backups
  - Backup and Restore ElastiCache Redis Clusters

REL 3: How do you support disaster recovery (DR) requirements?

Question-level introduction: Disaster recovery is an important aspect of any workload planning. ElastiCache for Redis offers several options to implement disaster recovery based on workload resilience requirements. With Amazon ElastiCache for Redis Global Datastore, you can write to your ElastiCache for Redis cluster in one region and have the data available to be read from two other cross-region replica clusters, thereby enabling low-latency reads and disaster recovery across regions.

Question-level benefit: Understanding and planning for a variety of disaster scenarios can ensure business continuity. DR strategies must be balanced against cost, performance impact, and data loss potential.

- **[Required]** Develop and document DR strategies for all your ElastiCache components based upon workload requirements. ElastiCache is unique in that some use cases are entirely ephemeral and don't require any DR strategy, whereas others are on the opposite end of the spectrum and require an extremely robust DR strategy. All options must be weighed against Cost Optimization – greater resiliency requires larger amounts of instrastructure.

  Understand the DR options available on a regional and multi-region level.

  - Multi-AZ Deployments are recommended to guard against AZ failure. Be sure to deploy with Cluster-Mode enabled in Multi-AZ architectures, with a minimum of 3 AZs available.
  - Global Datastore is recommended to guard against regional failures.
- **[Best]** Enable Global Datastore for workloads that require region level resiliency.
  - Have a plan to failover to secondary region in case of primary degradation.
  - Test multi-region failover process prior to a failover over in production.
  - Monitor `ReplicationLag` metric to understand potential impact of data loss during failover events.

- **[Resources]:**
  - Mitigating Failures
  - Replication across AWS Regions using global datastores
  - Restoring from a backup with optional cluster resizing
  - Minimizing downtime in ElastiCache for Redis with Multi-AZ

REL 4: How do you effectively plan for failovers?

Question-level introduction: Enabling multi-AZ with automatic failovers is an ElastiCache best practice. In certain cases, ElastiCache for Redis replaces primary nodes as part of service operations. Examples include planned maintenance events and the unlikely case of a node failure or availability zone issue. Successful failovers rely on both ElastiCache and your client library configuration.

Question-level benefit: Following best practices for ElastiCache failovers in conjunction with your specific ElastiCache for Redis client library helps you minimize potential downtime during failover events.

- **[Required]** For cluster mode disabled, use timeouts so your clients detect if it needs to disconnect from the old primary node and reconnect to the new primary node, using the updated primary endpoint IP address. For cluster mode enabled, the client library is responsible with detecting changes in the underlying cluster topology. This is accomplished most often by configuration settings in the ElastiCache for Redis client library, which also allow you to configure the frequency and the method of refresh. Each client library offers its own settings and more details are available in their corresponding documentation.

  [Resources]:
  - Minimizing downtime in ElastiCache for Redis with Multi-AZ
  - Review the best practices of your ElastiCache for Redis client library.
- **[Required]** Successful failovers depend on a healthy replication environment between the primary and the replica nodes. Review and understand the asynchronous nature of Redis replication, as well as the available CloudWatch metrics to report on the replication lag between primary and replica nodes. For use cases that require greater data safety, leverage the Redis WAIT command to force replicas to acknowledge writes before responding to connected clients.

  [Resources]:
  - Metrics for Redis

  - Monitoring best practices with Amazon ElastiCache for Redis using Amazon CloudWatch
- **[Best]** Regularly validate the responsiveness of your application during failover using the ElastiCache Test Failover API.

  [Resources]:

  - Testing Automatic Failover to a Read Replica on Amazon ElastiCache for Redis
  - Testing automatic failover

REL 5: Are your ElastiCache components designed to scale?

Question-level introduction: By understanding the scaling capabilities and available deployment topologies, your ElastiCache components can adjust over time to meet changing workload requirements. ElastiCache offers 4-way scaling: in/out (horizontal) as well as up/down (vertical).

Question-level benefit: Following best practices for ElastiCache deployments provides the greatest amount of scaling flexibility, as well as meeting the Well Architected principle of scaling horizontally to minimize the impact of failures.

- **[Required]** Understand the difference between Cluster-mode Enabled and Cluster-mode Disabled topologies. In almost all cases it is recommended to deploy with Cluster-mode enabled as it allow for greater scalability over time. Cluster-mode disabled components are limited in their ability to horizontally scale by adding read replicas.
- **[Required]** Understand when and how to scale.
  - For more READIOPS: add replicas
  - For more WRITEOPS: add shards (scale out)
  - For more network IO – use network optimized instances, scale up
- **[Best]** Deploy your ElastiCache components with Cluster-mode enabled, with a bias toward more, smaller nodes rather than fewer, larger nodes. This effectively limits the blast radius of a node failure.
- **[Best]** Include replicas in your clusters for enhanced responsiveness during scaling events
- **[Good]** For cluster-mode disabled, leverage read replicas to increase overall read capacity. ElastiCache has support for up to 5 read replicas in cluster-mode disabled, as well as vertical scaling.
- **[Resources]:**
  - Scaling ElastiCache for Redis clusters

- Online scaling up
- Scaling ElastiCache for Memcached clusters

Amazon ElastiCache Well-Architected Lens Performance Efficiency Pillar

The performance efficiency pillar focuses on using IT and computing resources efficiently. Key topics include selecting the right resource types and sizes based on workload requirements, monitoring performance, and making informed decisions to maintain efficiency as business needs evolve.

Topics

PE 1: How do you monitor the performance of your Amazon ElastiCache cluster?

Question-level introduction: By understanding the existing monitoring metrics, you can identify current utilization. Proper monitoring can help identify potential bottlenecks impacting the performance of your cluster.

Question-level benefit: Understanding the metrics associated with your cluster can help guide optimization techniques that can lead to reduced latency and increased throughput.

- **[Required]** Baseline performance testing using a subset of your workload.
  - You should monitor performance of the actual workload using mechanisms such as load testing.

  - Monitor the CloudWatch metrics while running these tests to gain an understanding of metrics available, and to establish a performance baseline.
- **[Best]** For ElastiCache for Redis workloads, rename computationally expensive commands, such as KEYS, to limit the ability of users to run blocking commands on production clusters.
  - ElastiCache for Redis workloads running engine 6.x, can leverage role-based access control to restrict certain commands. Access to the commands can be controlled by creating Users and User Groups with the AWS Console or CLI, and associating the User Groups to an ElastiCache for Redis cluster. In Redis 6, when RBAC is enabled, we can use "-@dangerous" and it will disallow expensive commands like KEYS, MONITOR, SORT, etc. for that user.
  - For engine version 5.x, rename commands using the `rename-commands` parameter on the Amazon ElastiCache for Redis cluster parameter group.
- **[Better]** Analyze slow queries and look for optimization techniques.
  - For ElastiCache for Redis workloads, learn more about your queries by analyzing the Slow Log. For example, you can use the following command, `redis-cli slowlog get 10` to show last 10 commands which exceeded latency thresholds (10 seconds by default).
  - Certain queries can be performed more efficiently using complex ElastiCache for Redis data structures. As an example, for numerical style range lookups, an application can implement simple numerical indexes with Sorted Sets. Managing these indexes can reduce scans performed on the data set, and return data with greater performance efficiency.
  - For ElastiCache for Redis workloads, `redis-benchmark` provides a simple interface for testing the performance of different commands using user defined inputs like number of clients, and size of data.
  - Since Memcached only supports simple key level commands, consider building additional keys as indexes to avoid iterating through the key space to serve client queries.
- **[Resources]:**
  - Monitoring use with CloudWatch Metrics
  - Monitoring use with CloudWatch Metrics
  - Using Amazon CloudWatch alarms
  - Redis-specific parameters
  - SLOWLOG
  - Redis benchmark

PE 2: How are you distributing work across your ElastiCache Cluster nodes?

Question-level introduction: The way your application connects to Amazon ElastiCache nodes can impact the performance and scalability of the cluster.

Question-level benefit: Making proper use of the available nodes in the cluster will ensure that work is distributed across the available resources. The following techniques help avoid idle resources as well.

- **[Required]** Have clients connect to the proper ElastiCache endpoint.
  - Amazon ElastiCache for Redis implements different endpoints based on the cluster mode in use. For cluster mode enabled, ElastiCache will provide a configuration endpoint. For cluster mode disabled, ElastiCache provides a primary endpoint, typically used for writes, and a reader endpoint for balancing reads across replicas. Implementing these endpoints correctly will results in better performance, and easier scaling operations. Avoid connecting to individual node endpoints unless there is a specific requirement to do so.
  - For multi-node Memcached clusters, ElastiCache provides a configuration endpoint which enables Auto Discovery. It is recommended to use a hashing algorithm to distribute work evenly across the cache nodes. Many Memcached client libraries implement consistent hashing. Check the documentation for the library you are using to see if it supports consistent hashing and how to implement it. You can find more information on implementing these features here.
- **[Better]** Implement a strategy for identifying and remediating hot keys in your workload.
  - Consider the impact of multi-dimensional Redis data structures such a lists, streams, sets, etc. These data structures are stored in single Redis Keys, which reside on a single node. A very large multi-dimensional key has the potential to utilize more network capacity and memory than other data types and can cause a disproportionate use of that node. If possible, design your workload to spread out data access across many discrete Keys.
  - Hot keys in the workload can impact performance of the node in use. For ElastiCache for Redis workloads, you can detect hot keys using `redis-cli --hotkeys` if an LFU max-memory policy is in place.
  - Consider replicating hot keys across multiple nodes to distribute access to them more evenly. This approach requires the client to write to multiple primary nodes (the Redis node itself will not provide this functionality) and to maintain a list of key names to read from, in addition to the original key name.

  - ElastiCache for Redis version 6 supports server-assisted client-side caching. This enables applications to wait for changes to a key before making network calls back to ElastiCache.
- **[Resources]:**
  - Configure Amazon ElastiCache for Redis for higher availability
  - Finding connection endpoints
  - Load balancing best practices
  - Client-side caching in Redis

PE 3: For caching workloads, how do you track and report the effectiveness and performance of your cache?

Question-level introduction: Caching is a commonly encountered workload on ElastiCache and it is important that you understand how to manage the effectiveness and performance of your cache.

Question-level benefit: Your application may show signs of sluggish performance. Your ability to use cache specific metrics to inform your decision on how to increase app performance is critical for your cache workload.

- **[Required]** Measure and track over time the cache-hits ratio. The efficiency of your cache is determined by its 'cache hits ratio'. The cache hits ratio is defined by the total of key hits divided by the total hits and misses. The closer to 1 the ratio is, the more effective your cache is. A low cache hits ratio is caused by the volume of cache misses. Cache misses occur when the requested key is not found in the cache. A key is not in the cache because it either has been evicted or deleted, has expired, or has never existed. Understand why keys are not in cache and develop appropriate strategies to have them in cache.

  [Resources]:
- **[Required]** Measure and collect your application cache performance in conjunction with latency and CPU utilization values to understand whether you need to make adjustments to your time-to-live or other application components. ElastiCache provides a set of CloudWatch metrics for aggregated latencies for each data structure. These latency metrics are calculated using the commandstats statistic from the ElastiCache for Redis INFO command and do not include the network and I/O time. This is only the time consumed by ElastiCache for Redis to process the operations.

  [Resources]:

  - Monitoring best practices with Amazon ElastiCache for Redis using Amazon CloudWatch
- **[Best]** Choose the right caching strategy for your needs. A low cache hits ratio is caused by the volume of cache misses. If your workload is designed to have low volume of cache misses (such as real time communication), it is best to conduct reviews of your caching strategies and apply the most appropriate resolutions for your workload, such as query instrumentation to measure memory and performance. The actual strategies you use to implement for populating and maintaining your cache depend on what data your clients need to cache and the access patterns to that data. For example, it is unlikely that you will use the same strategy for both personalized recommendations on a streaming application, and for trending news stories.

  [Resources]:

  - Caching strategies
  - Caching Best Practices
  - Performance at Scale with Amazon ElastiCache Whitepaper

PE 4: How does your workload optimize the use of networking resources and connections?

Question-level introduction: ElastiCache for Redis and Memcached are supported by many application clients, and implementations may vary. You need to understand the networking and connection management in place to analyze potential performance impact.

Question-level benefit: Efficient use of networking resources can improve the performance efficiency of your cluster. The following recommendations can reduce networking demands, and improve cluster latency and throughput.

- **[Required]** Proactively manage connections to your ElastiCache cluster.
  - Connection pooling in the application reduces the amount of overhead on the cluster created by opening and closing connections. Monitor connection behavior in Amazon CloudWatch using `CurrConnections` and `NewConnections`.
  - Avoid connection leaking by properly closing client connections where appropriate. Connection management strategies include properly closing connections that are not in use, and setting connection time-outs.
  - For Memcached workloads, there is a configurable amount of memory reserved for handling connections called, `memcached_connections_overhead`.
- **[Better]** Compress large objects to reduce memory, and improve network throughput.

  - Data compression can reduce the amount of network throughput required (Gbps), but increases the amount of work on the application to compress and decompress data.
  - Compression also reduces the amount of memory consumed by keys
  - Based on your application needs, consider the trade-offs between compression ratio and compression speed.
- **[Resources]:**
  - Amazon ElastiCache for Redis - Global Datastore
  - Memcached specific parameters
  - Amazon ElastiCache for Redis 5.0.3 enhances I/O handling to boost performance
  - Configure Amazon ElastiCache for Redis for higher availability

PE 5: How do you manage key deletion and/or eviction?

Question-level introduction: Workloads have different requirements, and expected behavior when a cluster node is approaching memory consumption limits. Amazon ElastiCache for Redis has different policies for handling these situations.

Question-level benefit: Proper management of available memory, and understanding of eviction policies will help ensure awareness of cluster behavior when instance memory limits are exceeded.

- **[Required]** Instrument the data access to evaluate which policy to apply. Identify an appropriate max-memory policy to control if and how evictions are performed on the cluster.
  - Eviction occurs when the max-memory on the cluster is consumed and a policy is in place to allow eviction. The behavior of the cluster in this situation depends on the eviction policy specified. This policy can be managed using the `maxmemory-policy` on the ElastiCache for Redis cluster parameter group.
  - The default policy `volatile-lru` frees up memory by evicting keys with a set expiration time (TTL value). Least frequently used (LFU) and least recently used (LRU) policies remove keys based on usage.
  - For Memcached workloads, there is a default LRU policy in place controlling evictions on each node. The number of evictions on your Amazon ElastiCache cluster can be monitored using the Evictions metric on Amazon CloudWatch.
- **[Better]** Standardize delete behavior to control performance impact on your cluster to avoid unexpected performance bottlenecks.

  - For ElastiCache for Redis workloads, when explicitly removing keys from the cluster, `UNLINK` is like `DEL`: it removes the specified keys. However, the command performs the actual memory reclaiming in a different thread, so it is not blocking, while `DEL` is. The actual removal will happen later asynchronously.
  - For ElastiCache for Redis 6.x workloads, the behavior of the `DEL` command can be modified in the parameter group using `lazyfree-lazy-user-del` parameter.
- **[Resources]:**
  - Configuring engine parameters using parameter groups
  - UNLINK
  - Cloud Financial Management with AWS

PE 6: How do you model and interact with data in ElastiCache?

Question-level introduction: ElastiCache is heavily application dependent on the data structures and the data model used, but it also needs to consider the underlying data store (if present). Understand the ElastiCache for Redis data structures available and ensure you are using the most appropriate data structures for your needs.

Question-level benefit: Data modeling in ElastiCache has several layers, including application use case, data types, and relationships between data elements. Additionally, each ElastiCache for Redis data type and command have their own well documented performance signatures.

- **[Best]** A best practice is to reduce unintentional overwriting of data. Use a naming convention that minimizes overlapping key names. Conventional naming of your data structures uses a hierarchical method such as: `APPNAME:CONTEXT:ID`, such as `ORDER-APP:CUSTOMER:123`.

  [Resources]:

  - Key naming
- **[Best]** ElastiCache for Redis commands have a time complexity defined by the Big O notation. This time complexity of a command is a algorithmic/mathematical representation of its impact. When introducing a new data type in your application you need to carefully review the time complexity of the related commands. Commands with a time complexity of O(1) are constant in time and do not depend on the size of the input however commands with a time complexity of O(N) are linear in time and are subject to the size of the input. Due to the single threaded design of ElastiCache for Redis, large volume of high time complexity operations will result in lower performance and potential operation timeouts.

  [Resources]:

  - Commands
- **[Best]** Use APIs to gain GUI visibility into the data model in your cluster.

  [Resources]:

  - Redis Commander
  - Redis Browser
  - Redsmin

PE 7: How do you log slow running commands in your Amazon ElastiCache cluster?

Question-level introduction: Performance tuning benefits through the capture, aggregation, and notification of long-running commands. By understanding how long it takes for commands to execute, you can determine which commands result in poor performance as well as commands that block the engine from performing optimally. Amazon ElastiCache for Redis also has the capability to forward this information to Amazon CloudWatch or Amazon Kinesis Data Firehose.

Question-level benefit: Logging to a dedicated permanent location and providing notification events for slow commands can help with detailed performance analysis and can be used to trigger automated events.

- **[Required]** Amazon ElastiCache for Redis running engine version 6.0 or newer, properly configured parameter group and SLOWLOG logging enabled on the cluster.
  - The required parameters are only available when engine version compatibility is set to Redis version 6.0 or higher.
  - SLOWLOG logging occurs when the server execution time of a command takes longer than a specified value. The behavior of the cluster depends on the associated Parameter Group parameters which are `slowlog-log-slower-than` and `slowlog-max-len`.
  - Changes take effect immediately.
- **[Best]** Take advantage of CloudWatch or Kinesis Data Firehose capabilities.
  - Use the filtering and alarm capabilities of CloudWatch, CloudWatch Logs Insights and Amazon Simple Notification Services to achieve performance monitoring and event notification.
  - Use the streaming capabilities of Kinesis Data Firehose to archive SLOWLOG logs to permanent storage or to trigger automated cluster parameter tuning.

    - Determine if JSON or plain TEXT format suits your needs best.
    - Provide IAM permissions to publish to CloudWatch or Kinesis Data Firehose.
- **[Better]** Configure `slowlog-log-slower-than` to a value other than the default.
    - This parameter determines how long a command may execute for within the Redis engine before it is logged as a slow running command. The default value is 10,000 microseconds (10 milliseconds). The default value may be too high for some workloads.
    - Determine a value that is more appropriate for your workload based on application needs and testing results; however, a value that is too low may generate excessive data.
- **[Better]** Leave `slowlog-max-len` at the default value.
    - This parameter determines the upper limit for how many slow-running commands are captured in Redis memory at any given time. A value of 0 effectively disables the capture. The higher the value, the more entries will be stored in memory, reducing the chance of important information being evicted before it can be reviewed. The default value is 128.
    - The default value is appropriate for most workloads. If there is a need to analyze data in an expanded time window from the redis-cli via the SLOWLOG command, consider increasing this value. This allows more commands to remain in Redis memory.

      If you are emitting the SLOWLOG data to either CloudWatch Logs or Kinesis Data Firehose, the data will be persisted and can be analyzed outside of the ElastiCache system, reducing the need to store large numbers of slow running commands in Redis memory.
- **[Resources]:**
    - How do I turn on Redis Slow log in an ElastiCache for Redis cache cluster?
    - Log delivery
    - Redis-specific parameters
    - https://aws.amazon.com/cloudwatch/Amazon CloudWatch
    - Amazon Kinesis Data Firehose

PE8: How does Auto Scaling help in increasing the performance of the ElastiCache cluster?

Question-level introduction: By implementing the feature of Redis auto scaling, your ElastiCache components can adjust over time to increase or decrease the desired shards or replicas automatically. This can be done by implementing either the target tracking or scheduled scaling policy.

Question-level benefit: Understanding and planning for the spikes in the workload can ensure enhanced caching performance and business continuity. ElastiCache for Redis Auto Scaling continually monitors your CPU/Memory utilization to make sure your cluster is operating at your desired performance levels.

- **[Required]** When launching a cluster for ElastiCache for Redis:

  1. Ensure that the Cluster mode is enabled
  2. Make sure the instance belongs to a family of certain type and size that support auto scaling
  3. Ensure the cluster is not running in Global datastores, Outposts or Local Zones

  [Resources]:

  - Scaling clusters in Redis (Cluster Mode Enabled)
  - Using Auto Scaling with shards
  - Using Auto Scaling with replicas
- **[Best]** Identify if your workload is read-heavy or write-heavy to define scaling policy. For best performance, use just one tracking metric. It is recommended to avoid multiple policies for each dimension, as auto scaling policies scale out when the target is hit, but scale in only when all target tracking policies are ready to scale in.

  [Resources]:

  - Auto Scaling policies
  - Defining a scaling policy
- **[Best]** Monitoring performance over time can help you detect workload changes that would remain unnoticed if monitored at a particular point in time. You can analyze corresponding CloudWatch metrics for cluster utilization over a four-week period to determine the target value threshold. If you are still not sure of what value to choose, we recommend starting with a minimum supported predefined metric value.

  [Resources]:

  - Monitoring use with CloudWatch Metrics
- **[Better]** We advise testing your application with expected minimum and maximum workloads, to identify the exact number of shards/replicas required for the cluster to develop scaling policies and mitigate availability issues.

  [Resources]:

  - Registering a Scalable Target

- Registering a Scalable Target

Amazon ElastiCache Well-Architected Lens Cost Optimization Pillar

The cost optimization pillar focuses on avoiding unnecessary costs. Key topics include understanding and controlling where money is being spent, selecting the most appropriate node type (use instances that support data tiering based on workload needs), the right number of resource types (how many read replicas) , analyzing spend over time, and scaling to meet business needs without overspending.

Topics

COST 1: How do you identify and track costs associated with your ElastiCache resources? How do you develop mechanisms to enable users to create, manage, and dispose of created resources?

Question-level introduction: Understanding cost metrics requires the participation of and collaboration across multiple teams: software engineering, data management, product owners, finance, and leadership. Identifying key cost drivers requires all involved parties understand service usage control levers and cost management trade-offs and it is frequently the key difference between successful and less successful cost optimization efforts. Ensuring you have processes and tools in place to track resources created from development to production and retirement helps you manage the costs associated with ElastiCache.

Question-level benefit: Continuous tracking of all costs associated with your workload requires a deep understanding of the architecture that includes ElastiCache as one of its components. Additionally, you should have a cost management plan in place to collect and compare usage against your budget.

- **[Required]** Institute a Cloud Center of Excellence (CCoE) with one of its founding charters to own defining, tracking, and taking action on metrics around your organizations' ElastiCache usage. If a CCoE exists and functions, ensure that it knows how to read and track costs associated with ElastiCache. When resources are created, use IAM roles and policies to validate that only specific teams and groups can instantiate resources. This ensures that costs are associated with business outcomes and a clear line of accountability is established, from a cost perspective.

    1. CCoE should identify, define, and publish cost metrics that are updated on a regular -monthly- basis around key ElastiCache usage across categorical data such as:

        a. Types of nodes used and their attributes: standard vs. memory optimized, on-demand vs. reserved instances, regions and availability zones

        b. Types of environments: free, dev, testing, and production

        c. Backup storage and retention strategies

        d. Data transfer within and across regions

        e. Instances running on Amazon Outposts

    2. CCoE consists of a cross-functional team with non-exclusive representation from software engineering, data management, product team, finance, and leadership teams in your organization.

    [Resources]:

    - Create a Cloud Center of Excellence
    - Amazon ElastiCache pricing

- **[Required]** Use cost allocation tags to track costs at a low level of granularity. Use AWS Cost Management to visualize, understand, and manage your AWS costs and usage over time.

    1. Use tags to organize your resources, and cost allocation tags to track your AWS costs on a detailed level. After you activate cost allocation tags, AWS uses the cost allocation tags to organize your resource costs on your cost allocation report, to make it easier for you to categorize and track your AWS costs. AWS provides two types of cost allocation tags, an AWS generated tags and user-defined tags. AWS defines, creates, and applies the AWS generated tags for you, and you define, create, and apply user-defined tags. You must activate both types of tags separately before they can appear in Cost Management or on a cost allocation report.

    2. Use cost allocation tags to organize your AWS bill to reflect your own cost structure. When you add cost allocation tags to your resources in Amazon ElastiCache, you will be able to

track costs by grouping expenses on your invoices by resource tag values. You should consider combining tags to track costs at a greater level of detail.

[Resources]:

- Using AWS cost allocation tags
- Monitoring costs with cost allocation tags
- AWS Cost Explorer

- **[Best]** Connect ElastiCache cost to metrics that reach across the organization.

  1. Consider business metrics as well as operational metrics like latency - what concepts in your business model are understandable across roles? The metrics need to be understandable by as many roles as possible in the organization.
  2. Examples - simultaneous served users, max and average latency per operation and user, user engagement scores, user return rates/week, session length/user, abandonment rate, cache hit rate, and keys tracked

  [Resources]:

  - Monitoring use with CloudWatch Metrics

- **[Good]** Maintain up-to-date architectural and operational visibility on metrics and costs across the entire workload that uses ElastiCache.

  1. Understand your entire solution ecosystem, ElastiCache tends to be part of a full ecosystem of AWS services in their technology set, from clients to API Gateway, Redshift, and QuickSight for reporting tools (for example).
  2. Map components of your solution from clients, connections, security, in-memory operations, storage, resource automation, data access and management, on your architecture diagram. Each layer connects to the entire solution and has its own needs and capabilities that add to and/or help you manage the overall cost.
  3. Your diagram should include the use of compute, networking, storage, lifecycle policies, metrics gathering as well as the operational and functional ElastiCache elements of your application
  4. The requirements of your workload are likely to evolve over time and it is essential that you continue to maintain and document your understanding of the underlying components as well as your primary functional objectives in order to remain proactive in your workload cost management.
  5. Executive support for visibility, accountability, prioritization, and resources is crucial to you having an effective cost management strategy for your ElastiCache.

COST 2: How do you use continuous monitoring tools to help you optimize the costs associated with your ElastiCache resources?

Question-level introduction: You need to aim for a proper balance between your ElastiCache cost and application performance metrics. Amazon CloudWatch provides visibility into key operational metrics that can help you assess whether your ElastiCache resources are over or under utilized, relative to your needs. From a cost optimization perspective, you need to understand when you are overprovisioned and be able to develop appropriate mechanisms to resize your ElastiCache resources while maintaining your operational, availability, resilience, and performance needs.

Question-level benefit: In an ideal state, you will have provisioned sufficient resources to meet your workload operational needs and not have under-utilized resources that can lead to a sub-optimal cost state. You need to be able to both identify and avoid operating oversized ElastiCache resources for long periods of time.

- **[Required]** Use CloudWatch to monitor your ElastiCache clusters and analyze how these metrics relate to your AWS Cost Explorer dashboards.

  1. ElastiCache provides both host-level metrics (for example, CPU usage) and metrics that are specific to the cache engine software (for example, cache gets and cache misses). These metrics are measured and published for each cache node in 60-second intervals.

  2. ElastiCache performance metrics (CPUUtilization, EngineUtilization, SwapUsage, CurrConnections, and Evictions) may indicate that you need to scale up/down (use larger/smaller cache node types) or in/out (add more/less shards). Understand the cost implications of scaling decisions by creating a playbook matrix that estimates the additional cost and the min and max lengths of time required to meet your application performance thresholds.

  [Resources]:

  - Monitoring use with CloudWatch Metrics
  - Which Metrics Should I Monitor?
  - Amazon ElastiCache pricing

- **[Required]** Understand and document your backup strategy and cost implications.

  1. With ElastiCache, the backups are stored in Amazon S3, which provides durable storage. You need to understand the cost implications in relation to your ability to recover from failures.

  2. Enable automatic backups that will delete backup files that are past the retention limit.

  [Resources]:

  - Scheduling automatic backups
  - Amazon Simple Storage Service pricing

- **[Best]** Use Reserved Nodes for your instances as a deliberate strategy to manage costs for workloads that are well understood and documented. Reserved nodes are charged an up front fee that depends upon the node type and the length of reservation—one or three years. This charge is much less than the hourly usage charge that you incur with On-Demand nodes.
  1. You may need to operate your ElastiCache clusters using on-demand nodes until you have gathered sufficient data to estimate the reserved instance requirements. Plan and document the resources needed to meet your needs and compare expected costs across instance types (on-demand vs. reserved)
  2. Regularly evaluate new cache node types available and assess whether it makes sense, from a cost and operational metrics perspective, to migrate your instance fleet to new cache node types

COST 3: Should you use an instance type that support data tiering? What are the advantages of a data tiering? When not to use data tiering instances?

Question-level introduction: Selecting the appropriate instance type can not only have performance and service level impact but also financial impact. Instance types have different cost associated with them. Selecting one or a few large instance types that can accommodate all storage needs in memory might be a natural decision. However, this could have significant cost impact as the project matures. Ensuring that the correct instance type is selected requires periodic examination of ElastiCache object idle time.

Question-level benefit: You should have a clear understanding of how various instance types impact your cost at the present and in the future. Marginal or periodic workload changes should not cause disproportionate costs changes. If the workload permits it, instance types that support data tiering offer a better price per storage available storage. Because of the per instance available SSD storage data tiering instances support a much higher total data per instance capability.

- **[Required]** Understand limitations of data tiering instances
  1. Only available for ElastiCache for Redis clusters.
  2. Only limited instance types support data tiering.
  3. Only ElastiCache for Redis version 6.2 and above is supported
  4. Large items are not swapped out to SSD. Objects over 128 MiB are kept in memory.

[Resources]:

- Data tiering
- Amazon ElastiCache pricing

- **[Required]** Understand what percentage of your database is regularly accessed by your workload.

  1. Data tiering instances are ideal for workloads that often access a small portion of your overall dataset but still requires fast access to the remaining data. In other words, the ration of hot to warm data is about 20:80.
  2. Develop cluster level tacking of object idle time.
  3. Large implementations of over 500 Gb of data are good candidates

- **[Required]** Understand that data tiering instances are not optional for certain workloads.

  1. There is a small performance cost for accessing less frequently used objects as those are swapped out to local SSD. If your application is response time sensitive test the impact on your workload.
  2. Not suitable for caches that store mostly large objects over 128 MiB in size.

  [Resources]:

  - Limitations

- **[Best]** Reserved instance types support data tiering. This assures the lowest cost in terms of amount of data storage per instance.

  1. You may need to operate your ElastiCache clusters using non-data tiering instances until you have a better understanding of your requirements.
  2. Analyze your ElastiCache clusters data usage pattern.
  3. Create an automated job that periodically collects object idle time.
  4. If you notice that a large percentage (about 80%) of objects are idle for a period of time deemed appropriate for your workload document the findings and suggest migrating the cluster to instances that support data tiering.
  5. Regularly evaluate new cache node types available and assess whether it makes sense, from a cost and operational metrics perspective, to migrate your instance fleet to new cache node types.

  [Resources]:

  - OBJECT IDLETIME

- Amazon ElastiCache pricing

Common troubleshooting steps and best practices

Topics

Connection issues

If you are unable to connect to your ElastiCache cache, consider one of the following:

1. **Using TLS:** If you are experiencing a hung connection when trying to connect to your ElastiCache endpoint, you may not be using TLS in your client. If you are using ElastiCache Serverless, encryption in transit is always enabled. Make sure that your client is using TLS to connect to the cache. Learn more about connecting to a TLS enabled cache here.
2. **VPC:** ElastiCache caches are accessible only from within a VPC. Ensure that the EC2 instance from which you are accessing the cache and the ElastiCache cache are created in the same VPC. Alternatively, you must enable VPC peering between the VPC where your EC2 instance resides and the VPC where you are creating your cache.
3. **Security groups:** ElastiCache uses security groups to control access to your cache. Consider the following:
    a. Make sure that the security group used by your ElastiCache cache allows inbound access to it from your EC2 instance. See here to learn how to setup inbound rules in your security group correctly.
    b. Make sure that the security group used by your ElastiCache cache allows access to your cache’s ports (6379 and 6380 for serverless, and 6379 by default for self-designed). ElastiCache uses these ports to accept Redis commands. Learn more about how to setup port access here.

Redis client errors

ElastiCache Serverless is only accessible using Redis clients that support the Redis cluster mode protocol. Self-designed clusters can be accessed from Redis clients in either mode, depending on the cluster configuration.

If you are experiencing Redis errors in your client, consider the following:

1. **Cluster mode:** If you are experiencing CROSSLOT errors or errors with the SELECT Redis command, you may be trying to access a Cluster Mode Enabled cache with a Redis client that does not support the Redis Cluster protocol. ElastiCache Serverless only supports Redis clients that support the Redis cluster protocol. If you want to use Redis in “Cluster Mode Disabled” (CMD), then you must design your own cluster.
2. **CROSSLOT errors:** If you are experiencing the `ERR CROSSLOT Keys in request don't hash to the same slot` error, you may be attempting to access keys that do not belong to the same slot in a Cluster mode cache. As a reminder, ElastiCache Serverless always operates in Cluster Mode. Multi-key operations, transactions, or Lua scripts involving multiple keys are allowed only if all the keys involved are in the same hash slot.

For additional best practices around configuring Redis clients, please review this blog post.

Troubleshooting high latency in ElastiCache Serverless

If your workload appears to experience high latency, you can analyze the CloudWatch `SuccessfulReadRequestLatency` and `SuccessfulWriteRequestLatency` metrics to check if the latency is related to ElastiCache Serverless. These metrics measure latency which is internal to ElastiCache Serverless - client side latency and network trip times between your client and the ElastiCache Serverless endpoint are not included.

Some variability and occasional spikes should not be a cause for concern. However, if the `Average` statistic shows a sharp increase and persists, you should check the AWS Health Dashboard and your Personal Health Dashboard for more information. If necessary, consider opening a support case with AWS Support.

Consider the following best practices and strategies to reduce latency:

- **Enable Read from Replica:** If your application allows it, we recommend enabling the “Read from Replica” feature in your Redis client to scale reads and achieve lower latency. When enabled, ElastiCache Serverless attempts to route your read requests to replica cache nodes that are in the same Availability Zone (AZ) as your client, thus avoiding cross-AZ network latency. Note, that enabling the Read from Replica feature in your client signifies that your application accepts eventual consistency in data. Your application may receive older data for some time if you attempt to read after writing to a key.

- **Ensure your application is deployed in the same AZs as your cache:** You may observe higher client side latency if your application is not deployed in the same AZs as your cache. When you create a serverless cache you can provide the subnets from where your application will access the cache, and ElastiCache Serverless creates VPC Endpoints in those subnets. Ensure that your application is deployed in the same AZs. Otherwise, your application may incur a cross-AZ hop when accessing the cache resulting in higher client side latency.
- **Reuse connections:** ElastiCache Serverless requests are made via a TLS enabled TCP connection using the RESP protocol. Initiating the connection (including authenticating the connection, if configured) takes time so the latency of the first request is higher than typical. Requests over an already initialized connection deliver ElastiCache's consistent low latency. For this reason, you should consider using connection pooling or reusing existing Redis connections.
- **Scaling speed:** ElastiCache Serverless automatically scales as your request rate grows. A sudden large increase in request rate, faster than the speed at which ElastiCache Serverless scales, may result in elevated latency for some time. ElastiCache Serverless can typically increase its supported request rate quickly, taking up to 10-12 minutes to double the request rate.
- **Inspect long running commands:** Some Redis commands, including Lua scripts or commands on large data structures, may run for a long time. To identify these commands, ElastiCache publishes command level metrics. With ElastiCache Serverless you can use the `BasedECPUs` metrics.
- **Throttled Requests:** When requests are throttled in ElastiCache Serverless, you may experience an increase in client side latency in your application. When requests are throttled in ElastiCache Serverless, you should see an increase in the `ThrottledRequests` ElastiCache Serverless metric. Review the section below for troubleshooting throttled requests.
- **Uniform distribution of keys and requests:** In ElastiCache for Redis, an uneven distribution of keys or requests per slot can result in a hot slot which can result in elevated latency. ElastiCache Serverless supports up to 30,000 ECPUs/second (90,000 ECPUs/second when using Read from Replica) on a single slot, in a workload that executes simple SET/GET commands. We recommend evaluating your key and request distribution across slots and ensuring a uniform distribution if your request rate exceeds this limit.

Troubleshooting throttling issues in ElastiCache Serverless

In service-oriented architectures and distributed systems, limiting the rate at which API calls are processed by various service components is called throttling. This smooths spikes, controls for mismatches in component throughput, and allows for more predictable recoveries when there's an unexpected operational event. ElastiCache Serverless is designed for these types of architectures, and most Redis clients have retries built in for throttled requests. Some degree of throttling is not necessarily a problem for your application, but persistent throttling of a latency-sensitive part of your data workflow can negatively impact user experience and reduce the overall efficiency of the system.

When requests are throttled in ElastiCache Serverless, you should see an increase in the `ThrottledRequests` ElastiCache Serverless metric. If you are noticing a high number of throttled requests, consider the following:

- **Scaling speed:** ElastiCache Serverless automatically scales as you ingest more data or grow your request rate.If your application scales faster than the speed at which ElastiCache Serverless scales, then your requests may get throttled while ElastiCache Serverless scales to accommodate your workload. ElastiCache Serverless can typically increase the storage size quickly, taking up to 10-12 minutes to double the storage size in your cache.
- **Uniform distribution of keys and requests:** In ElastiCache for Redis, an uneven distribution of keys or requests per slot can result in a hot slot. A hot slot can result in throttling of requests if the request rate to a single slot exceeds 30,000 ECPUs/second, in a workload that executes simple SET/GET commands.
- **Read from Replica:** If you application allows it, consider using the “Read from Replica“ feature. Most Redis clients can be configured to ”scale reads“ to direct reads to replica nodes. This feature enables you to scale read traffic. In addition ElastiCache Serverless automatically routes read from replica requests to nodes in the same Availability Zone as your application resulting in lower latency. When Read from Replica is enabled, you can achieve up to 90,000 ECPUs/second on a single slot, for workloads with simple SET/GET commands.

Related Topics

- Additional troubleshooting steps
- the section called “Best practices and caching strategies”

Additional troubleshooting steps

The following items must be verified while troubleshooting persistent connectivity issues with ElastiCache:

Topics

Security groups

Security Groups are virtual firewalls protecting your ElastiCache client (EC2 instance, AWS Lambda function, Amazon ECS container, etc.) and ElastiCache cache. Security groups are stateful, meaning that after the incoming or outgoing traffic is allowed, the responses for that traffic will be automatically authorized in the context of that specific security group.

The stateful feature requires the security group to keep track of all authorized connections, and there is a limit for tracked connections. If the limit is reached, new connections will fail. Please refer to the troubleshooting section for help on how to identify if the limits has been hit on the client or ElastiCache side.

You can have a single security groups assigned at the same time to the client and ElastiCache cluster, or individual security groups for each.

For both cases, you need to allow the TCP outbound traffic on the ElastiCache port from the source and the inbound traffic on the same port to ElastiCache. The default port is 11211 for Memcached and 6379 for Redis. By default, security groups allow all outbound traffic. In this case, only the inbound rule in the target security group is required.

For more information, see Access patterns for accessing an ElastiCache cluster in an Amazon VPC.

Network ACLs

Network Access Control Lists (ACLs) are stateless rules. The traffic must be allowed in both directions (Inbound and Outbound) to succeed. Network ACLs are assigned to subnets, not specific resources. It is possible to have the same ACL assigned to ElastiCache and the client resource, especially if they are in the same subnet.

By default, network ACLs allow all trafic. However, it is possible to customize them to deny or allow traffic. Additionally, the evaluation of ACL rules is sequential, meaning that the rule with the lowest number matching the traffic will allow or deny it. The minimum configuration to allow the Redis traffic is:

Client side Network ACL:

- **Inbound Rules:**
- Rule number: preferably lower than any deny rule;
- Type: Custom TCP Rule;
- Protocol: TCP
- Port Range: 1024-65535
- Source: 0.0.0.0/0 (or create individial rules for the ElastiCache cluster subnets)
- Allow/Deny: Allow

- **Outbound Rules:**
- Rule number: preferably lower than any deny rule;
- Type: Custom TCP Rule;
- Protocol: TCP
- Port Range: 6379
- Source: 0.0.0.0/0 (or the ElastiCache cluster subnets. Keep in mind that using specific IPs may create issues in case of failover or scaling the cluster)
- Allow/Deny: Allow

ElastiCache Network ACL:

- **Inbound Rules:**
- Rule number: preferably lower than any deny rule;
- Type: Custom TCP Rule;
- Protocol: TCP
- Port Range: 6379
- Source: 0.0.0.0/0 (or create individial rules for the ElastiCache cluster subnets)
- Allow/Deny: Allow

- **Outbound Rules:**
- Rule number: preferably lower than any deny rule;
- Type: Custom TCP Rule;
- Protocol: TCP
- Port Range: 1024-65535
- Source: 0.0.0.0/0 (or the ElastiCache cluster subnets. Keep in mind that using specific IPs may create issues in case of failover or scaling the cluster)
- Allow/Deny: Allow

For more information, see Network ACLs.

Route tables

Similarly to Network ACLs, each subnet can have different route tables. If clients and the ElastiCache cluster are in different subnets, make sure that their route tables allow them to reach each other.

More complex environments, involving multiple VPCs, dynamic routing, or network firewalls, may become difficult to troubleshoot. See Network connectivity validation to confirm that your network settings are appropriate.

DNS resolution

ElastiCache provides the service endpoints based on DNS names. The endpoints available are `Configuration`, `Primary`, `Reader`, and `Node` endpoints. For more information, see Finding Connection Endpoints.

In case of failover or cluster modification, the address associated to the endpoint name may change and will be automatically updated.

Custom DNS settings (i.e., not using the VPC DNS service) may not be aware of the ElastiCache-provided DNS names. Make sure that your system can successfully resolve the ElastiCache endpoints using system tools like `dig` (as shown following) or `nslookup`.

```
$ dig +short example.xxxxxx.ng.0001.use1.cache.amazonaws.com
example-001.xxxxxx.0001.use1.cache.amazonaws.com.
1.2.3.4
```

You can also force the name resolution through the VPC DNS service:

```
$ 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
```

Identifying issues with server-side diagnostics

CloudWatch metrics and run-time information from the ElastiCache engine are common sources or information to identify potential sources of connection issues. A good analysis commonly starts with the following items:

- CPU usage: Redis is a multi-threaded application. However, execution of each command happens in a single (main) thread. For this reason, ElastiCache provides the metrics `CPUUtilization` and `EngineCPUUtilization`. `EngineCPUUtilization` provides the CPU utilization dedicated to the Redis process, and `CPUUtilization` the usage across all vCPUs. Nodes with more than one vCPU usually have different values for `CPUUtilization` and `EngineCPUUtilization`, the second being commonly higher. High `EngineCPUUtilization` can be caused by an elevated number of requests or complex operations that take a significant amount of CPU time to complete. You can identify both with the following:
  - Elevated number of requests: Check for increases on other metrics matching the `EngineCPUUtilization` pattern. Useful metrics are:

  - `CacheHits` and `CacheMisses`: the number of successful requests or requests that didn't find a valid item in the cache. If the ratio of misses compared to hits is high, the application is wasting time and resources with unfruitful requests.
  - `SetTypeCmds` and `GetTypeCmds`: These metrics correlated with `EngineCPUUtilization` can help to understand if the load is significantly higher for write requests, measured by `SetTypeCmds`, or reads, measured by `GetTypeCmds`. If the load is predominantly reads, using multiple read-replicas can balance the requests across multiple nodes and spare the primary for writes. In cluster mode-disabled clusters, the use of read-replicas can be done by creating an additional connection configuration in the application using the ElastiCache reader endpoint. For more information, see Finding Connection Endpoints. The read operations must be submitted to this additional connection. Write operations will be done through the regular primary endpoint. In cluster mode-enabled, it is advisable to use a library that supports read replicas natively. With the right flags, the library will be able to automatically discover the cluster topology, the replica nodes, enable the read operations through the READONLY Redis command, and submit the read requests to the replicas.
- Elevated number of connections:
  - `CurrConnections` and `NewConnections`: `CurrConnection` is the number of established connections at the moment of the datapoint collection, while `NewConnections` shows how many connections were created in the period.

    Creating and handling connections implies significant CPU overhead. Additionally, the TCP three-way handshake required to create new connections will negatively affect the overall response times.

    An ElastiCache node with thousands of `NewConnections` per minute indicates that a connection is created and used by just a few commands, which is not optimal. Keeping connections established and reusing them for new operations is a best practice. This is possible when the client application supports and properly implements connection pooling or persistent connections. With connection pooling, the number of `currConnections` does not have big variations, and the `NewConnections` should be as low as possible. Redis provides optimal performance with small number of currConnections. Keeping currConnection in the order of tens or hundreds minimizes the usage of resources to support individual connections like client buffers and CPU cycles to serve the connection.
- Network throughput:
  - Determine the bandwidth: ElastiCache nodes have network bandwidth proportional to the node size. Since applications have different characteristics, the results can vary according to

  the workload. As examples, applications with high rate of small requests tend to affect more the CPU usage than the network throughput while bigger keys will cause higher network utilization. For that reason, it is advisable to test the nodes with the actual workload for a better understanding of the limits.

  Simulating the load from the application would provide more accurate results. However, benchmark tools can give a good idea of the limits.

  - For cases where the requests are predominantly reads, using replicas for read operations will alleviate the load on the primary node. If the use-case is predominantly writes, the use of many replicas will amplify the network usage. For every byte written to the primary node, N bytes will be sent to the replicas,being N the number of replicas. The best practice for write intensive workloads are using ElastiCache for Redis with cluster mode-enabled so the writes can be balanced across multiple shards, or scale-up to a node type with more network capabilities.
  - The CloudWatchmetrics `NetworkBytesIn` and `NetworkBytesOut` provide the amount of data coming into or leaving the node, respectively. `ReplicationBytes` is the traffic dedicated to data replication.

  For more information, see Network-related limits.

- Complex commands: Redis commands are served on a single thread, meaning that requests are served sequentially. A single slow command can affect other requests and connections, culminating in time-outs. The use of commands that act upon multiple values, keys, or data types must be done carefully. Connections can be blocked or terminated depending on the number of parameters, or size of its input or output values.

  A notorious example is the KEYS command. It sweeps the entire keyspace searching for a given pattern and blocks the execution of other commands during its execution. Redis uses the “Big O” notation to describe its commands complexity.

  Keys command has O(N) time complexity, N being the number of keys in the database. Therefore, the larger the number of keys, the slower the command will be. KEYS can cause trouble in different ways: If no search pattern is used, the command will return all key names available. In databases with thousand or million of items, a huge output will be created and flood the network buffers.

If a search pattern is used, only the keys matching the pattern will return to the client. However, the engine still sweeps the entire keyspace searching for it, and the time to complete the command will be the same.

An alternative for KEYS is the SCAN command. It iterates over the keyspace and limits the iterations in a specific number of items, avoiding prolonged blocks on the engine.

Scan has the COUNT parameter, used to set the size of the iteration blocks. The default value is 10 (10 items per iteration).

Depending on the number of items in the database, small COUNT values blocks will require more iterations to complete a full scan, and bigger values will keep the engine busy for longer at each iteration. While small count values will make SCAN slower on big databases, larger values can cause the same issues mentioned for KEYS.

As an example, running the SCAN command with count value as 10 will requires 100,000 repetitions on a database with 1 million keys. If the average network round-trip time is 0.5 milliseconds, approximately 50,000 milliseconds (50 seconds) will be spent transferring requests.

On the other hand, if the count value were 100,0000, a single iteration would be required and only 0.5 ms would be spent transferring it. However, the engine would be entirely blocked for other operations until the command finishes sweeping all the keyspace.

Besides KEYS, several other commands are potentially harmful if not used correctly. To see a list of all commands and their respective time complexity, go to https://redis.io/commands.

Examples of potential issues:

- Lua scripts: Redis provides an embedded Lua interpreter, allowing the execution of scripts on the server-side. Lua scripts on Redis are executed on engine level and are atomic by definition, meaning that no other command or script will be allowed to run while a script is in execution. Lua scripts provide the possibility of running multiple commands, decision-making algorithms, data parsing, and others directly on the Redis engine. While the atomicity of scripts and the chance of offloading the application are tempting, scripts must be used with care and for small operations. On ElastiCache, the execution time of Lua scripts is limited to 5 seconds. Scripts that haven't written to the keyspace will be automatically terminated after the 5 seconds period. To avoid data corruption and inconsistencies, the node will failover if the script execution hasn't completed in 5 seconds and had any write during its execution. Transactions are the alternative to guarantee consistency of multiple related key modifications in Redis. A transaction allows the execution of a block of commands, watching existing keys for modifications. If any of the watched keys changes before the completion of the transaction, all modifications are discarded.

- Mass deletion of items: The DEL command accepts multiple parameters, which are the key names to be deleted. Deletion operations are synchronous and will take significant CPU time if the list of parameters is big, or contains a big list, set, sorted set, or hash (data structures holding several sub-items). In other words, even the deletion of a single key can take significant time if it has many elements. The alternative to DEL is UNLINK, which is an asynchronous command available since Redis 4. UNLINK must be preferred over DEL whenever possible. Starting on ElastiCache for Redis 6x, the lazyfree-lazy-user-del parameter makes the DEL command behave like UNLINK when enabled. For more information, see Redis 6.0 Parameter Changes.
- Commands acting upon multiple keys: DEL was mentioned before as a command that accepts multiple arguments and its execution time will be directly proportional to that. However, Redis provides many more commands that work similarly. As examples, MSET and MGET allow the insertion or retrieval of multiple String keys at once. Their usage may be beneficial to reduce the network latency inherent to multiple individual SET or GET commands. However, an extensive list of parameters will affect CPU usage.

  While CPU utilization alone is not the cause for connectivity issues, spending too much time to process a single or few commands over multiple keys may cause failure of other requests and increase overall CPU utilization.

  The number of keys and their size will affect the command complexity and consequently completion time.

  Other examples of commands that can act upon multiple keys: HMGET, HMSET, MSETNX, PFCOUNT, PFMERGE, SDIFF, SDIFFSTORE, SINTER, SINTERSTORE, SUNION, SUNIONSTORE, TOUCH, ZDIFF, ZDIFFSTORE, ZINTER or ZINTERSTORE.
- Commands acting upon multiple data types: Redis also provides commands that act upon one or multiple keys, regardless of their data type. ElastiCache for Redis provides the metric KeyBasedCmds to monitor such commands. This metric sums the execution of the following commands in the selected period:
  - O(N) complexity:
    - KEYS

  - O(1)
    - EXISTS
    - OBJECT
    - PTTL
    - RANDOMKEY
    - TTL
    - TYPE
    - EXPIRE
    - EXPIREAT
    - MOVE
    - PERSIST
    - PEXPIRE
    - PEXPIREAT
    - UNLINK (O(N) to reclaim memory. However the memory-reclaiming task happens in a separated thread and does not block the engine
  - Different complexity times depending on the data type:
    - DEL
    - DUMP
    - RENAME is considered a command with O(1) complexity, but executes DEL internally. The execution time will vary depending on the size of the renamed key.
    - RENAMENX
    - RESTORE
    - SORT
  - Big hashes: Hash is a data type that allows a single key with multiple key-value sub-items. Each hash can store 4.294.967.295 items and operations on big hashes can become expensive. Similarly to KEYS, hashes have the HKEYS command with O(N) time complexity, N being the number of items in the hash. HSCAN must be preferred over HKEYS to avoid long running commands. HDEL, HGETALL, HMGET, HMSET and HVALS are commands that should be used with caution on big hashes.
- Other big data-structures: Besides hashes, other data structures can be CPU intensive. Sets, Lists, Sorted Sets, and Hyperloglogs can also take significant time to be manipulated

depending on their size and commands used. For more information on those commands, see https://redis.io/commands.

Network connectivity validation

After reviewing the network configurations related to DNS resolution, security groups, network ACLs, and route tables, the connectivity can be validated with the VPC Reachability Analyzer and system tools.

Reachability Analyzer will test the network connectivity and confirm if all the requirements and permissions are satisfied. For the tests below you will need the ENI ID (Elastic Network Interface Identification) of one of the ElastiCache nodes available in your VPC. You can find it by doing the following:

1. Go to https://console.aws.amazon.com/ec2/v2/home?#NIC:
2. Filter the interface list by your ElastiCache cluster name or the IP address got from the DNS validations previously.
3. Write down or otherwise save the ENI ID. If multiple interfaces are shown, review the description to confirm that they belong to the right ElastiCache cluster and choose one of them.
4. Proceed to the next step.
5. Create an analyze path at https://console.aws.amazon.com/vpc/home?#ReachabilityAnalyzer and choose the following options:

   - **Source Type**: Choose **instance** if your ElastiCache client runs on an Amazon EC2 instance or **Network Interface** if it uses another service, such as AWS Fargate Amazon ECS with awsvpc network, AWS Lambda, etc), and the respective resource ID (EC2 instance or ENI ID);
   - **Destination Type**: Choose **Network Interface** and select the **Elasticache ENI** from the list.
   - **Destination port**: specify 6379 for ElastiCache for Redis or 11211 for ElastiCache for Memcached. Those are the ports defined with the default configuration and this example assumes that they are not changed.
   - **Protocol**: TCP

Create the analyze path and wait a few moments for the result. If the status is unreachable, open the analysis details and review the **Analysis explorer** for details where the requests were blocked.

If the reachability tests passed, proceed to the verification on the OS level.

To validate the TCP connectivity on the ElastiCache service port: On Amazon Linux, Nping is available in the package nmap and can test the TCP connectivity on the ElastiCache port, as well as providing the network round-trip time to establish the connection. Use this to validate the network connectivity and the current latency to the ElastiCache cluster, as shown following:

```
$ 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
```

By default, nping sends 5 probes with a delay of 1 second between them. You can use the option “-c” to increase the number of probes and “--delay“ to change the time to send a new test.

If the tests with nping fail and the *VPC Reachability Analyzer* tests passed, ask your system administrator to review possible Host-based firewall rules, asymmetric routing rules, or any other possible restriction on the operating system level.

On the ElastiCache console, check if **Encryption in-transit** is enabled in your ElastiCache cluster details. If in-transit encryption is enabled, confirm if the TLS session can be established with the following command:

```
openssl s_client -connect example.xxxxxx.use1.cache.amazonaws.com:6379
```

An extensive output is expected if the connection and TLS negotiation are successful. Check the return code available in the last line, the value must be 0 (ok). If openssl returns something different, check the reason for the error at https://www.openssl.org/docs/man1.0.2/man1/verify.html#DIAGNOSTICS.

If all the infrastructure and operating system tests passed but your application is still unable to connect to ElastiCache, check if the application configurations are compliant with the ElastiCache settings. Common mistakes are:

- Your application does not support ElastiCache cluster mode, and ElastiCache has cluster-mode enabled;
- Your application does not support TLS/SSL, and ElastiCache has in-transit encryption enabled;
- Application supports TLS/SSL but does not have the right configuration flags or trusted certification authorities;

Network-related limits

- Maximum number of connections: There are hard limits for simultaneous connections. Each ElastiCache node allows up to 65,000 simultaneous connections across all clients. This limit can be monitored through the `CurrConnections` metrics on CloudWatch. However, clients also have their limits for outbound connections.On Linux, check the allowed ephemeral port range with the command:

  ```
  # sysctl net.ipv4.ip_local_port_range
  net.ipv4.ip_local_port_range = 32768 60999
  ```

  In the previous example, 28231 connections will be allowed from the same source, to the same destination IP (ElastiCache node) and port. The following command shows how many connections exist for a specific ElastiCache node (IP 1.2.3.4):

  ```
  ss --numeric --tcp state connected "dst 1.2.3.4 and dport == 6379" | grep -vE
   '^State' | wc -l
  ```

  If the number is too high, your system may become overloaded trying to process the connection requests. It is advisable to consider implementing techniques like connection pooling or persistent connections to better handle the connections. Whenever possible, configure the connection pool to limit the maximum number of connections to a few hundred. Also, back-off logic to handle time-outs or other connection exceptions would are advisable to avoid connection churn in case of issues.
- Network traffic limits: Check the following CloudWatch metrics for Redis to identify possible network limits being hit on the ElastiCache node:
  - `NetworkBandwidthInAllowanceExceeded` / `NetworkBandwidthOutAllowanceExceeded`: Network packets shaped because the throughput exceeded the aggregated bandwidth limit.

It is important to note that every byte written to the primary node will be replicated to N replicas, N being the number of replicas. Clusters with small node types, multiple replicas, and intensive write requests may not be able to cope with the replication backlog. For such cases, it's a best practice to scale-up (change node type), scale-out (add shards in cluster-mode enabled clusters), reduce the number of replicas, or minimize the number of writes.

- `NetworkConntrackAllowanceExceeded`: Packets shaped because the maximum number of connections tracked across all security groups assigned to the node has been exceeded. New connections will likely fail during this period.
- `NetworkPackets PerSecondAllowanceExceeded`: Maximum number of packets per second exceeded. Workloads based on a high rate of very small requests may hit this limit before the maximum bandwidth.

The metrics above are the ideal way to confirm nodes hitting their network limits. However, limits are also identifiable by plateaus on network metrics.

If the plateaus are observed for extended periods, they will be likely followed by replication lag, increase on bytes Used for cache, drop on freeable memory, high swap and CPU usage. Amazon EC2 instances also have network limits that can tracked through ENA driver metrics. Linux instances with enhanced networking support and ENA drivers 2.2.10 or newer can review the limit counters with the command:

```
# ethtool -S eth0 | grep "allowance_exceeded"
```

CPU Usage

The CPU usage metric is the starting point of investigation, and the following items can help to narrow down possible issues on the ElastiCache side:

- Redis SlowLogs: The ElastiCache default configuration retains the last 128 commands that took over 10 milliseconds to complete. The history of slow commands is kept during the engine runtime and will be lost in case of failure or restart. If the list reaches 128 entries, old events will be removed to open room for new ones. The size of the list of slow events and the execution time considered slow can by modified via the parameters `slowlog-max-len` and `slowlog-log-slower-than` in a custom parameter group. The slowlogs list can be retrieved by running `SLOWLOG GET 128` on the engine, 128 being the last 128 slow commands reported. Each entry has the following fields:

```
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
```

The event above happened on December 26, at 19:26:07 UTC, took 4.8 seconds (4.823ms) to complete and was caused by the KEYS command requested from the client 1.2.3.4.

On Linux, the timestamp can be converted with the command date:

```
$ date --date='@1609010767'
Sat Dec 26 19:26:07 UTC 2020
```

With Python:

```
>>> from datetime import datetime
>>> datetime.fromtimestamp(1609010767)
datetime.datetime(2020, 12, 26, 19, 26, 7)
```

Or on Windows with 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
```

Many slow commands in a short period of time (same minute or less) is a reason for concern. Review the nature of commands and how they can be optimized (see previous examples). If commands with O(1) time complexity are frequently reported, check the other factors for high CPU usage mentioned before.

- Latency metrics: ElastiCache for Redis provides CloudWatch metrics to monitor the average latency for different classes of commands. The datapoint is calculated by dividing the total number of executions of commands in the category by the total execution time in the period. It is important to understand that latency metric results are an aggregate of multiple commands. A single command can cause unexpected results, like timeouts, without significant impact on the metrics. For such cases, the slowlog events would be a more accurate source of information. The following list contains the latency metrics available and the respective commands that affect them.
  - EvalBasedCmdsLatency: related to Lua Script commands, `eval`, `evalsha`;
  - GeoSpatialBasedCmdsLatency: `geodist`, `geohash`, `geopos`, `georadius`, `georadiusbymember`, `geoadd`;
  - GetTypeCmdsLatency: Read commands, regardless of data type;
  - HashBasedCmdsLatency: `hexists`, `hget`, `hgetall`, `hkeys`, `hlen`, `hmget`, `hvals`, `hstrlen`, `hdel`, `hincrby`, `hincrbyfloat`, `hmset`, `hset`, `hsetnx`;
  - HyperLogLogBasedCmdsLatency: `pfselftest`, `pfcount`, `pfdebug`, `pfadd`, `pfmerge`;
  - KeyBasedCmdsLatency: Commands that can act upon different data types: dump, `exists`, `keys`, `object`, `pttl`, `randomkey`, `ttl`, `type`, `del`, `expire`, `expireat`, move, `persist`, `pexpire`, `pexpireat`, `rename`, `renamenx`, `restoreK`, `sort`, `unlink`;
  - ListBasedCmdsLatency: lindex, llen, lrange, blpop, brpop, brpoplpush, linsert, lpop, lpush, lpushx, lrem, lset, ltrim, rpop, rpoplpush, rpush, rpushx;
  - PubSubBasedCmdsLatency: psubscribe, publish, pubsub, punsubscribe, subscribe, unsubscribe;
  - SetBasedCmdsLatency: `scard`, `sdiff`, `sinter`, `sismember`, `smembers`, `srandmember`, `sunion`, `sadd`, `sdiffstore`, `sinterstore`, `smove`, `spop`, `srem`, `sunionstore`;
  - SetTypeCmdsLatency: Write commands, regardless of data-type;
  - 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 runtime commands:
  - info commandstats: Provides a list of commands executed since the Redis engine started, their cumulative executions number, total execution time, and average execution time per command;
  - client list: Provides a list of currently connected clients and relevant information like buffers usage, last command executed, etc;
- Backup and replication: ElastiCache for Redis versions earlier than 2.8.22 use a forked process to create backups and process full syncs with the replicas. This method may incur in significant memory overhead for write intensive use-cases.

  Starting with ElastiCache Redis 2.8.22, AWS introduced a forkless backup and replication method. The new method may delay writes in order to prevent failures. Both methods can cause periods of higher CPU utilization, lead to higher response times and consequently lead to client timeouts during their execution. Always check if the client failures happen during the backup window or the SaveInProgress metric was 1 in the period. It is advisable to schedule the backup window for periods of low utilization to minimize the possibility of issues with clients or backup failures.

Connections being terminated from the server side

The default ElastiCache for Redis configuration keeps the client connections established indefinitely. However, in some cases connection termination may be desirable. For example:

- Bugs in the client application may cause connections to be forgotten and kept established with an idle state. This is called "connection leak“ and the consequence is a steady increase on the number of established connections observed on the CurrConnections metric. This behavior can result in saturation on the client or ElastiCache side. When an immediate fix is not possible from the client side, some administrators set a ”timeout“ value in their ElastiCache parameter group. The timeout is the time in seconds allowed for idle connections to persist. If the client doesn't submit any request in the period, the Redis engine will terminate the connection as soon as the connection reaches the timeout value. Small timeout values may result in unnecessary disconnections and clients will need handle them properly and reconnect, causing delays.

- The memory used to store keys is shared with client buffers. Slow clients with big requests or responses may demand a significant amount of memory to handle its buffers. The default ElastiCache for Redis configurations does not restrict the size of regular client output buffers. If the `maxmemory` limit is hit, the engine will try to evict items to fulfill the buffer usage. In extreme low memory conditions, ElastiCache for Redis might choose to disconnect clients that consume large client output buffers in order to free memory and retain the cluster's health.

  It is possible to limit the size of client buffers with custom configurations and clients hitting the limit will be disconnected. However, clients should be able to handle unexpected disconnections. The parameters to handle buffers size for regular clients are the following:

  - client-query-buffer-limit: Maximum size of a single input request;
  - client-output-buffer-limit-normal-soft-limit: Soft limit for client connections. The connection will be terminated if stays above the soft limit for more than the time in seconds defined on client-output-buffer-limit-normal-soft-seconds or if it hits the hard limit;
  - client-output-buffer-limit-normal-soft-seconds: Time allowed for the connections exceeding the client-output-buffer-limit-normal-soft-limit;
  - client-output-buffer-limit-normal-hard-limit: A connection hitting this limit will be immediatelly terminated.

  Besides the regular client buffers, the following options control the buffer for replica nodes and Pub/Sub (Publish/Subscribe) clients:

  - 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;

Client-side troubleshooting for Amazon EC2 instances

The load and responsiveness on the client side can also affect the requests to ElastiCache. EC2 instance and operating system limits need to be carefully reviewed while troubleshooting intermittent connectivity or timeout issues. Some key points to observe:

- CPU:
  - EC2 instance CPU usage: Make sure the CPU hasn't been saturated or near to 100 percent. Historical analysis can be done via CloudWatch, however keep in mind that data points granularity is either 1 minute (with detailed monitoring enabled) or 5 minutes;
  - If using burstable EC2 instances, make sure that their CPU credit balance hasn't been depleted. This information is available on the `CPUCreditBalance` CloudWatch metric.
  - Short periods of high CPU usage can cause timeouts without reflecting on 100 percent utilization on CloudWatch. Such cases require real-time monitoring with operating-system tools like `top`, `ps` and `mpstat`.
- Network
  - Check if the Network throughput is under acceptable values according to the instance capabilities. For more information, see Amazon EC2 Instance Types
  - On instances with the `ena` Enhanced Network driver, check the ena statistics for timeouts or exceeded limits. The following statistics are useful to confirm network limits saturation:
    - `bw_in_allowance_exceeded` / `bw_out_allowance_exceeded`: number of packets shaped due to excessive inbound or outbound throughput;
    - `conntrack_allowance_exceeded`: number of packets dropped due to security groups connection tracking limits. New connections will fail when this limit is saturated;
    - `linklocal_allowance_exceeded`: number of packets dropped due to excessive requests to instance meta-data, NTP via VPC DNS. The limit is 1024 packets per second for all the services;
    - `pps_allowance_exceeded`: number of packets dropped due to excessive packets per second ratio. The PPS limit can be hit when the network traffic consists on thousands or millions of very small requests per second. ElastiCache traffic can be optimized to make better use of network packets via pipelines or commands that do multiple operations at once like `MGET` instead of `GET`.

Dissecting the time taken to complete a single request

- On the network: `Tcpdump` and `Wireshark` (tshark on the command line) are handy tools to understand how much time the request took to travel the network, hit the ElastiCache engine and get a return. The following example highlights a single request created with the following command:

  ```
  $ echo ping | nc example.xxxxxx.ng.0001.use1.cache.amazonaws.com 6379
  +PONG
  ```

  In parallel to the command above, tcpdump was in execution and returned:

  ```
  $ 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
```

From the output above we can confirm that the TCP three-way handshake was completed in 222 microseconds (918091 - 917869) and the ping command was submitted and returned in 173 microseconds (918295 - 918122).

It took 438 microseconds (918307 - 917869) from requesting to closing the connection. Those results would confirm that network and engine response times are good and the investigation can focus on other components.

- On the operating system: `Strace` can help identifying time gaps on the OS level. The analysis of actual applications would be way more extensive and specialized application profilers or debuggers are advisable. The following example just shows if the base operating system components are working as expected, otherwise further investigation may be required. Using the same Redis `PING` command with `strace` we get:

```
$ 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 +++
```

In the example above, the command took a little more than 54 milliseconds to complete (752110 - 697712 = 54398 microseconds).

A significant amount of time, approximately 20ms, was taken to instantiate nc and do the name resolution (from 697712 to 717890), after that, 2ms were required to create the TCP socket (745659 to 747858), and 0.4 ms (747858 to 748330) to submit and receive the response for the request.

Security in Amazon ElastiCache

Cloud security at AWS is the highest priority. As an AWS customer, you benefit from a data center and network architecture that is built to meet the requirements of the most security-sensitive organizations.

Security is a shared responsibility between AWS and you. The shared responsibility model describes this as security *of* the cloud and security *in* the cloud:

- **Security of the cloud** – AWS is responsible for protecting the infrastructure that runs AWS services in the AWS Cloud. AWS also provides you with services that you can use securely. Third-party auditors regularly test and verify the effectiveness of our security as part of the AWS compliance programs. To learn about the compliance programs that apply to Amazon ElastiCache, see AWS Services in Scope by Compliance Program.
- **Security in the cloud** – Your responsibility is determined by the AWS service that you use. You are also responsible for other factors including the sensitivity of your data, your company's requirements, and applicable laws and regulations.

This documentation helps you understand how to apply the shared responsibility model when using Amazon ElastiCache. The following topics show you how to configure Amazon ElastiCache to meet your security and compliance objectives. You also learn how to use other AWS services that help you to monitor and secure your Amazon ElastiCache resources.

Topics

Data protection in Amazon ElastiCache

The AWS shared responsibility model applies to data protection in AWS ElastiCache (ElastiCache). As described in this model, AWS is responsible for protecting the global infrastructure that runs all of the AWS Cloud. You are responsible for maintaining control over your content that is hosted on this infrastructure. This content includes the security configuration and management tasks for the AWS services that you use. For more information about data privacy, see the Data privacy FAQ.

For data protection purposes, we recommend that you protect AWS account credentials and set up individual accounts with AWS Identity and Access Management (IAM). That way each user is given only the permissions necessary to fulfill their job duties. We also recommend that you secure your data in the following ways:

- Use multi-factor authentication (MFA) with each account.
- Use TLS to communicate with AWS resources.
- Set up API and user activity logging with AWS CloudTrail.
- Use AWS encryption solutions, along with all default security controls within AWS services.
- Use advanced managed security services such as Amazon Macie, which assists in discovering and securing personal data that is stored in Amazon S3.

We strongly recommend that you never put sensitive identifying information, such as your customers' account numbers, into free-form fields such as a **Name** field. This includes when you work with ElastiCache or other AWS services using the console, API, AWS CLI, or AWS SDKs. Any data that you enter into ElastiCache or other services might get picked up for inclusion in diagnostic logs. When you provide a URL to an external server, don't include credentials information in the URL to validate your request to that server.

Topics

Data security in Amazon ElastiCache

To help keep your data secure, Amazon ElastiCache and Amazon EC2 provide mechanisms to guard against unauthorized access of your data on the server.

Amazon ElastiCache for Memcached also provides encryption features for data on caches running Memcached versions 1.6.12 or later.

- In-transit encryption encrypts your data whenever it is moving from one place to another, such as between nodes in your cluster or between your cache and your application.
- At-rest encryption encrypts your on-disk data during sync and backup operations.

Topics

- ElastiCache in-transit encryption (TLS)
- At-Rest Encryption in ElastiCache

ElastiCache in-transit encryption (TLS)

To help keep your data secure, Amazon ElastiCache and Amazon EC2 provide mechanisms to guard against unauthorized access of your data on the server. By providing in-transit encryption capability, ElastiCache gives you a tool you can use to help protect your data when it is moving from one location to another.

All serverless caches have in-transit encryption enabled. For self-designed clusters, you can enable in-transit encryption on a cache cluster by setting the parameter `TransitEncryptionEnabled` to `true` (CLI: `--transit-encryption-enabled`) when you create the cache cluster using the `CreateCacheCluster` (CLI: `create-cache-cluster`) operation.

Topics

- In-transit encryption overview
- In-transit encryption conditions
- In-transit encryption best practices
- Enabling in-transit encryption
- Connecting to nodes enabled with in-transit encryption using Openssl
- Creating a TLS Memcached client using Java
- Creating a TLS Memcached client using PHP

In-transit encryption overview

Amazon ElastiCache in-transit encryption is a feature that allows you to increase the security of your data at its most vulnerable points—when it is in transit from one location to another. Because there is some processing needed to encrypt and decrypt the data at the endpoints, enabling in-transit encryption can have some performance impact. You should benchmark your data with and without in-transit encryption to determine the performance impact for your use cases.

ElastiCache in-transit encryption implements the following features:

- **Encrypted client connections—**client connections to cache nodes are TLS encrypted.
- **Encrypted server connections—**data moving between nodes in a cluster is encrypted.
- **Server authentication—**clients can authenticate that they are connecting to the right server.

In-transit encryption conditions

The following constraints on Amazon ElastiCache in-transit encryption should be kept in mind when you plan your self-designed cluster implementation:

- In-transit encryption is supported on clusters running Memcached versions 1.6.12 and later.
- In-transit encryption supports Transport Layer Security (TLS) versions 1.2 and 1.3.
- In-transit encryption is supported only for clusters running in an Amazon VPC.
- In-transit encryption is not supported for replication groups running the following node types: M1, M2, M3, R3, T2.

  For more information, see Supported node types.
- In-transit encryption is enabled by explicitly setting the parameter `TransitEncryptionEnabled` to `true`.
- You can enable in-transit encryption on a cluster only when creating the cluster. You cannot toggle in-transit encryption on and off by modifying a cluster.
- Ensure that your caching client supports TLS connectivity and that you have enabled it in client configuration.

In-transit encryption best practices

- Because of the processing required to encrypt and decrypt the data at the endpoints, implementing in-transit encryption can reduce performance. Benchmark in-transit encryption compared to no encryption on your own data to determine its impact on performance for your implementation.
- Because creating new connections can be expensive, you can reduce the performance impact of in-transit encryption by persisting your TLS connections.

Enabling in-transit encryption

To enable in-transit encryption when creating a Memcached cluster using the AWS Management Console, make the following selections:

- Choose Memcached as your engine.
- Choose engine version 1.6.12 or later.
- Under **Encryption in transit**, choose **Enable**.

For the step-by-step process, see Creating a Memcached cluster (console).

Connecting to nodes enabled with in-transit encryption using Openssl

To access data from ElastiCache for Memcached nodes enabled with in-transit encryption, you need to use clients that work with Secure Socket Layer (SSL). You can also use Openssl s_client on Amazon Linux and Amazon Linux 2.

To use Openssl s_client to connect to a Memcached cluster enabled with in-transit encryption on Amazon Linux 2 or Amazon Linux:

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

Creating a TLS Memcached client using Java

To create a client in TLS mode, do the following to initialize the client with the appropriate SSLContext:

```
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");
    }
}
```

Creating a TLS Memcached client using PHP

To create a client in TLS mode, do the following to initialize the client with the appropriate SSLContext:

```
<?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);
}
```

For more information on using the PHP client, see Installing the ElastiCache cluster client for PHP.

At-Rest Encryption in ElastiCache

To help keep your data secure, Amazon ElastiCache and Amazon S3 provide different ways to restrict access to data in your cache. For more information, see Amazon VPCs and ElastiCache security and Identity and Access Management for Amazon ElastiCache.

- Disk during sync and swap operations

ElastiCache offers default (service managed) encryption at rest, as well as ability to use your own symmetric customer managed AWS KMS keys in AWS Key Management Service (KMS). When the cache is backed up, under encryption options, choose whether to use the default encryption key or a customer-managed key. For more information, see Enabling At-Rest Encryption.

> **Note**
>
> The default (service managed) encryption is the only option available in the GovCloud (US) Regions.

At-rest encryption can be enabled on a cache only when it is created. Because there is some processing needed to encrypt and decrypt the data, enabling at-rest encryption can have a performance impact during these operations. You should benchmark your data with and without at-rest encryption to determine the performance impact for your use cases.

Topics

At-Rest Encryption Conditions

The following constraints on ElastiCache at-rest encryption should be kept in mind when you plan your implementation of ElastiCache encryption at-rest:

- At-rest encryption is supported only on serverless caches.

- The option to use customer managed key for encryption at rest is not available in AWS GovCloud (us-gov-east-1 and us-gov-west-1) regions.

Using customer managed keys from AWS KMS

ElastiCache supports symmetric customer managed AWS KMS keys (KMS key) for encryption at rest. Customer-managed KMS keys are encryption keys that you create, own and manage in your AWS account. For more information, see AWS KMS keys in the *AWS Key Management Service Developer Guide*. The keys must be created in AWS KMS before they can be used with ElastiCache.

To learn how to create AWS KMS root keys, see Creating Keys in the *AWS Key Management Service Developer Guide*.

ElastiCache allows you to integrate with AWS KMS. For more information, see Using Grants in the *AWS Key Management Service Developer Guide*. No customer action is needed to enable Amazon ElastiCache integration with AWS KMS.

The `kms:ViaService` condition key limits use of an AWS KMS key (KMS key) to requests from specified AWS services. To use `kms:ViaService` with ElastiCache, include both ViaService names in the condition key value: `elasticache.AWS_region.amazonaws.com` and `dax.AWS_region.amazonaws.com`. For more information, see kms:ViaService.

You can use AWS CloudTrail to track the requests that Amazon ElastiCache sends to AWS Key Management Service on your behalf. All API calls to AWS Key Management Service related to customer managed keys have corresponding CloudTrail logs. You can also see the grants that ElastiCache creates by calling the ListGrants KMS API call.

- If you delete the key or disable the key and revoke grants for the key that you used to encrypt a cache, the cache becomes irrecoverable. In other words, it cannot be modified or recovered after a hardware failure. AWS KMS deletes root keys only after a waiting period of at least seven days. After the key is deleted, you can use a different customer managed key to create a backup for archival purposes.
- Automatic key rotation preserves the properties of your AWS KMS root keys, so the rotation has no effect on your ability to access your ElastiCache data. Encrypted Amazon ElastiCache caches don't support manual key rotation, which involves creating a new root key and updating any references to the old key. To learn more, see Rotating AWS KMS keys in the *AWS Key Management Service Developer Guide*.

- Encrypting an ElastiCache cache using KMS key requires one grant per cache. This grant is used throughout the lifespan of the cache.
- For more information on AWS KMS grants and limits, see Limits in the *AWS Key Management Service Developer Guide*.

Enabling At-Rest Encryption

All serverless caches have at-rest encryption enabled.

You can enable at-rest encryption when you create an ElastiCache cache. You can do so using the AWS Management Console, the AWS CLI, or the ElastiCache API.

When creating a cache, you can pick one of the following options:

- **Default –** This option uses service managed encryption at rest.
- **Customer managed key –** This option allows you to provide the Key ID/ARN from AWS KMS for encryption at rest.

To learn how to create AWS KMS root keys, see Create Keys in the *AWS Key Management Service Developer Guide*

Contents

Enabling At-Rest Encryption Using the AWS Management Console

Enabling At-Rest Encryption on a Serverless Cache (Console)

All serverless caches have at-rest encryption enabled. By default, an AWS-owned KMS key is used to encrypt data. To choose your own AWS KMS key, make the following selections:

- Expand the **Default settings** section.
- Choose **Customize default settings** under **Default settings** section.
- Choose **Customize your security settings** under **Security** section.
- Choose **Customer managed CMK** under **Encryption key** setting.
- Select a key under **AWS KMS key** setting.

See Also

- Amazon VPCs and ElastiCache security
- Identity and Access Management for Amazon ElastiCache

Internetwork traffic privacy

Amazon ElastiCache uses the following techniques to secure your cache data and protect it from unauthorized access:

- **Amazon VPCs and ElastiCache security** explains the type of security group you need for your installation.
- **Identity and Access Management for Amazon ElastiCache** for granting and limiting actions of users, groups, and roles.

Amazon VPCs and ElastiCache security

Because data security is important, ElastiCache provides means for you to control who has access to your data. How you control access to your data is dependent upon whether or not you launched your clusters in an Amazon Virtual Private Cloud (Amazon VPC) or Amazon EC2-Classic.

> ⚠ **Important**
>
> We have deprecated the use of Amazon EC2-Classic for launching ElastiCache clusters. All current generation nodes are launched in Amazon Virtual Private Cloud only.

The Amazon Virtual Private Cloud (Amazon VPC) service defines a virtual network that closely resembles a traditional data center. When you configure your Amazon VPC you can select its IP address range, create subnets, and configure route tables, network gateways, and security settings. You can also add a cache cluster to the virtual network, and control access to the cache cluster by using Amazon VPC security groups.

This section explains how to manually configure an ElastiCache cluster in an Amazon VPC. This information is intended for users who want a deeper understanding of how ElastiCache and Amazon VPC work together.

Topics

- Understanding ElastiCache and Amazon VPCs
- Access Patterns for Accessing an ElastiCache Cache in an Amazon VPC
- Creating a Virtual Private Cloud (VPC)
- Connecting to a cache running in an Amazon VPC

Understanding ElastiCache and Amazon VPCs

ElastiCache is fully integrated with the Amazon Virtual Private Cloud (Amazon VPC). For ElastiCache users, this means the following:

- If your AWS account supports only the EC2-VPC platform, ElastiCache always launches your cluster in an Amazon VPC.
- If you're new to AWS, your clusters will be deployed into an Amazon VPC. A default VPC will be created for you automatically.
- If you have a default VPC and don't specify a subnet when you launch a cluster, the cluster launches into your default Amazon VPC.

For more information, see Detecting Your Supported Platforms and Whether You Have a Default VPC.

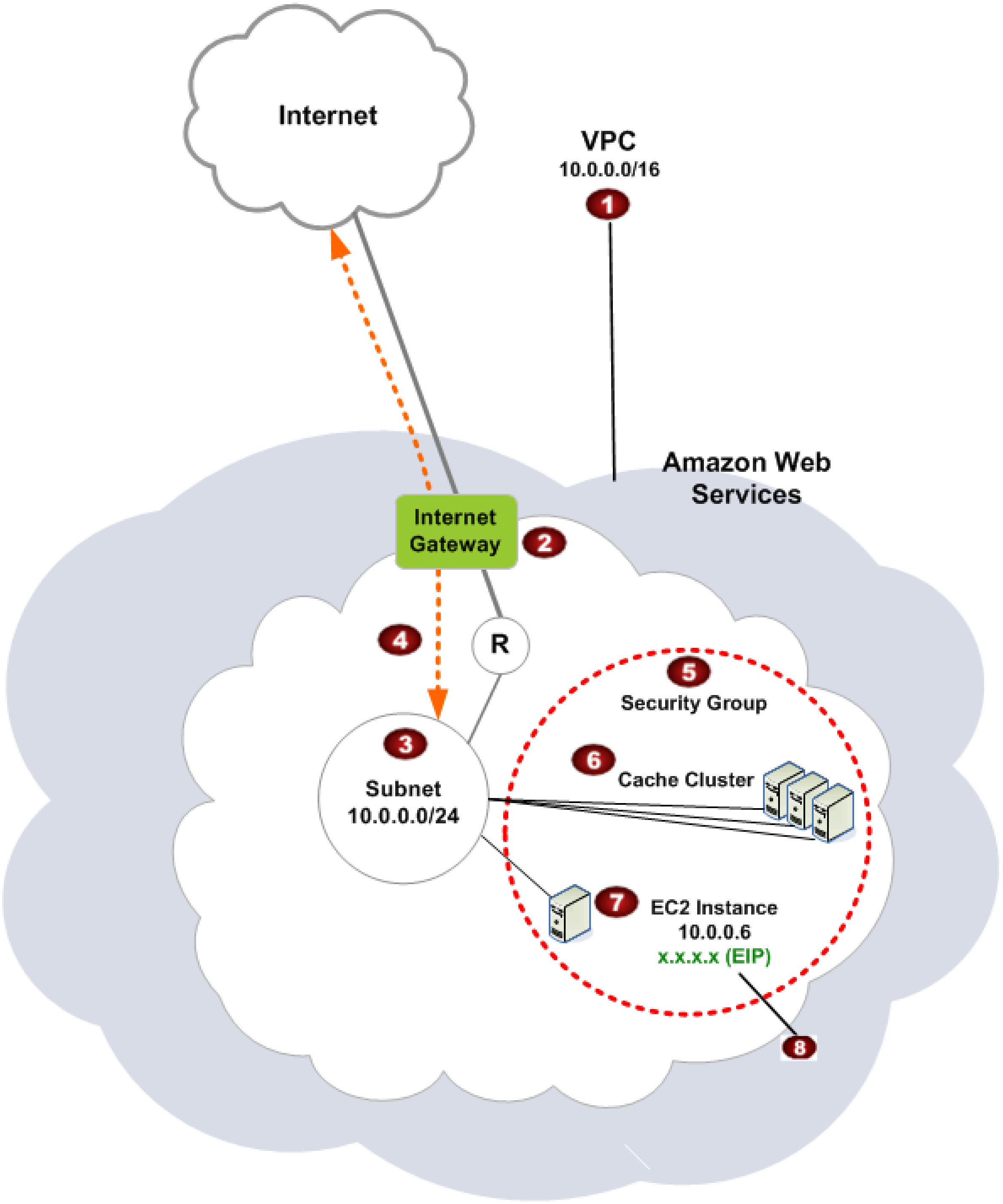
With Amazon Virtual Private Cloud, you can create a virtual network in the AWS cloud that closely resembles a traditional data center. You can configure your Amazon VPC, including selecting its IP address range, creating subnets, and configuring route tables, network gateways, and security settings.

The basic functionality of ElastiCache is the same in a virtual private cloud; ElastiCache manages software upgrades, patching, failure detection and recovery whether your clusters are deployed inside or outside an Amazon VPC.

ElastiCache cache nodes deployed outside an Amazon VPC are assigned an IP address to which the endpoint/DNS name resolves. This provides connectivity from Amazon Elastic Compute Cloud (Amazon EC2) instances. When you launch an ElastiCache cluster into an Amazon VPC private subnet, every cache node is assigned a private IP address within that subnet.

Overview of ElastiCache in an Amazon VPC

The following diagram and table describe the Amazon VPC environment, along with ElastiCache clusters and Amazon EC2 instances that are launched in the Amazon VPC.



1	The Amazon VPC is an isolated portion of the AWS Cloud that is assigned its own block of IP addresses.
2	

	An Internet gateway connects your Amazon VPC directly to the Internet and provides access to other AWS resources such as Amazon Simple Storage Service (Amazon S3) that are running outside your Amazon VPC.
3	An Amazon VPC subnet is a segment of the IP address range of an Amazon VPC where you can isolate AWS resources according to your security and operational needs.
4	A routing table in the Amazon VPC directs network traffic between the subnet and the Internet. The Amazon VPC has an implied router, which is symbolized in this diagram by the circle with the R.
5	An Amazon VPC security group controls inbound and outbound traffic for your ElastiCache clusters and Amazon EC2 instances.
6	You can launch an ElastiCache cluster in the subnet. The cache nodes have private IP addresses from the subnet's range of addresses.
7	You can also launch Amazon EC2 instances in the subnet. Each Amazon EC2 instance has a private IP address from the subnet's range of addresses. The Amazon EC2 instance can connect to any cache node in the same subnet.
8	For an Amazon EC2 instance in your Amazon VPC to be reachable from the Internet, you need to assign a static, public address called an Elastic IP address to the instance.

Prerequisites

To create an ElastiCache cluster within an Amazon VPC, your Amazon VPC must meet the following requirements:

- The Amazon VPC must allow nondedicated Amazon EC2 instances. You cannot use ElastiCache in an Amazon VPC that is configured for dedicated instance tenancy.
- A cache subnet group must be defined for your Amazon VPC. ElastiCache uses that cache subnet group to select a subnet and IP addresses within that subnet to associate with your VPC endpoints or cache nodes.

- CIDR blocks for each subnet must be large enough to provide spare IP addresses for ElastiCache to use during maintenance activities.

Routing and security

You can configure routing in your Amazon VPC to control where traffic flows (for example, to the Internet gateway or virtual private gateway). With an Internet gateway, your Amazon VPC has direct access to other AWS resources that are not running in your Amazon VPC. If you choose to have only a virtual private gateway with a connection to your organization's local network, you can route your Internet-bound traffic over the VPN and use local security policies and firewall to control egress. In that case, you incur additional bandwidth charges when you access AWS resources over the Internet.

You can use Amazon VPC security groups to help secure the ElastiCache clusters and Amazon EC2 instances in your Amazon VPC. Security groups act like a firewall at the instance level, not the subnet level.

> **Note**
>
> We strongly recommend that you use DNS names to connect to your cache nodes, as the underlying IP address can change.

Amazon VPC documentation

Amazon VPC has its own set of documentation to describe how to create and use your Amazon VPC. The following table gives links to the Amazon VPC guides.

Description	Documentation
How to get started using Amazon VPC	Getting started with Amazon VPC
How to use Amazon VPC through the AWS Management Console	Amazon VPC User Guide
Complete descriptions of all the Amazon VPC commands	Amazon EC2 Command Line Reference (the Amazon VPC commands are found in the Amazon EC2 reference)

Description	Documentation
Complete descriptions of the Amazon VPC API operations, data types, and errors	Amazon EC2 API Reference (the Amazon VPC API operations are found in the Amazon EC2 reference)
Information for the network administrator who needs to configure the gateway at your end of an optional IPsec VPN connection	What is AWS Site-to-Site VPN?

For more detailed information about Amazon Virtual Private Cloud, see Amazon Virtual Private Cloud.

Access Patterns for Accessing an ElastiCache Cache in an Amazon VPC

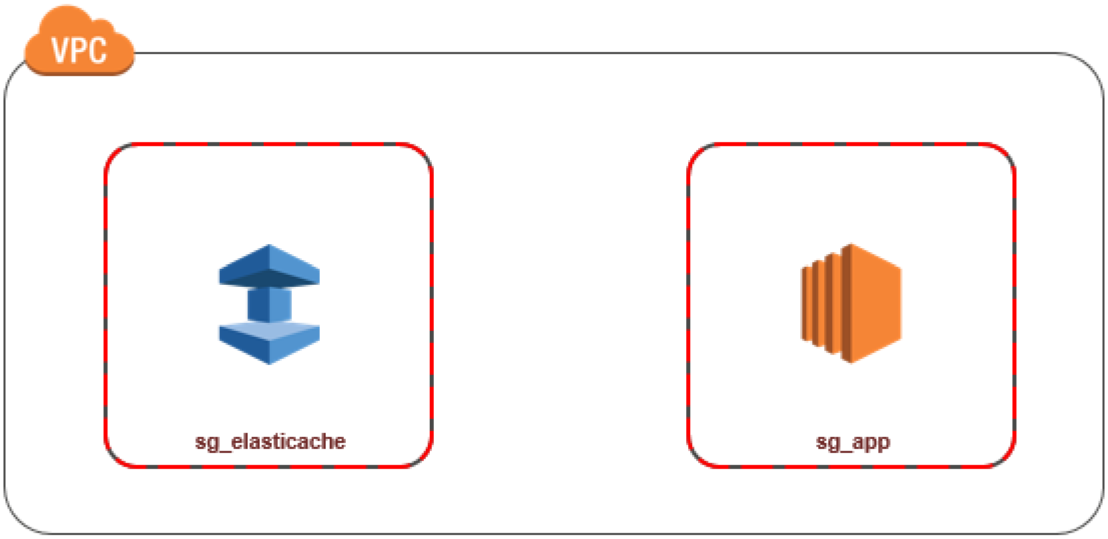
Amazon ElastiCache supports the following scenarios for accessing a cache in an Amazon VPC:

Contents

Accessing an ElastiCache Cache when it and the Amazon EC2 Instance are in the Same Amazon VPC

The most common use case is when an application deployed on an EC2 instance needs to connect to a cache in the same VPC.

The following diagram illustrates this scenario



The simplest way to manage access between EC2 instances and caches in the same VPC is to do the following:

1. Create a VPC security group for your cache. This security group can be used to restrict access to the cache. For example, you can create a custom rule for this security group that allows TCP access using the port you assigned to the cache when you created it and an IP address you will use to access the cache.

   The default port for Memcached caches is 11211.
2. Create a VPC security group for your EC2 instances (web and application servers). This security group can, if needed, allow access to the EC2 instance from the Internet via the VPC's routing table. For example, you can set rules on this security group to allow TCP access to the EC2 instance over port 22.
3. Create custom rules in the security group for your cache that allow connections from the security group you created for your EC2 instances. This would allow any member of the security group to access the caches.

Note

If you are planning to use Local Zones, ensure that you have enabled them. When you create a subnet group in that local zone, your VPC is extended to that Local Zone and your VPC will treat the subnet as any subnet in any other Availability Zone. All relevant gateways and route tables will be automatically adjusted.

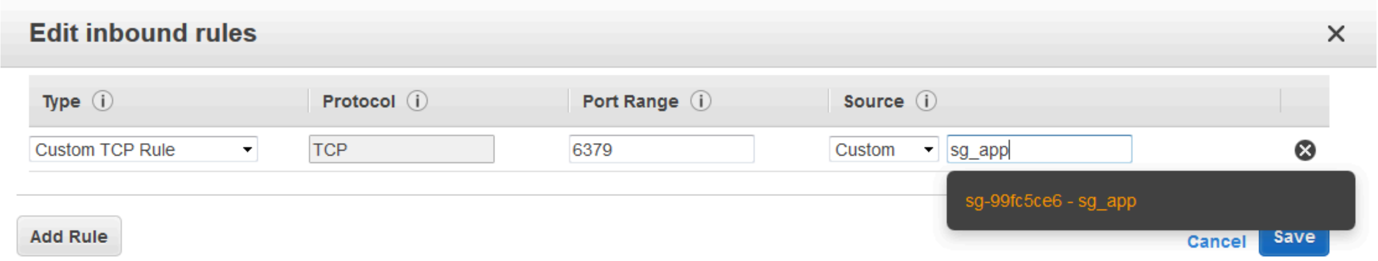
To create a rule in a VPC security group that allows connections from another security group

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc.
2. In the navigation pane, choose **Security Groups**.
3. Select or create a security group that you will use for your cache. Under **Inbound Rules**, select **Edit Inbound Rules** and then select **Add Rule**. This security group will allow access to members of another security group.
4. From **Type** choose **Custom TCP Rule**.

   a. For **Port Range**, specify the port you used when you created your cache.

      The default port for Memcached caches is 11211.

   b. In the **Source** box, start typing the ID of the security group. From the list select the security group you will use for your Amazon EC2 instances.
5. Choose **Save** when you finish.



Edit inbound rules

Type	Protocol	Port Range	Source	
Custom TCP Rule	TCP	6379	Custom	sg_app

sg-99fc5ce6 - sg_app

Accessing an ElastiCache Cache when it and the Amazon EC2 Instance are in Different Amazon VPCs

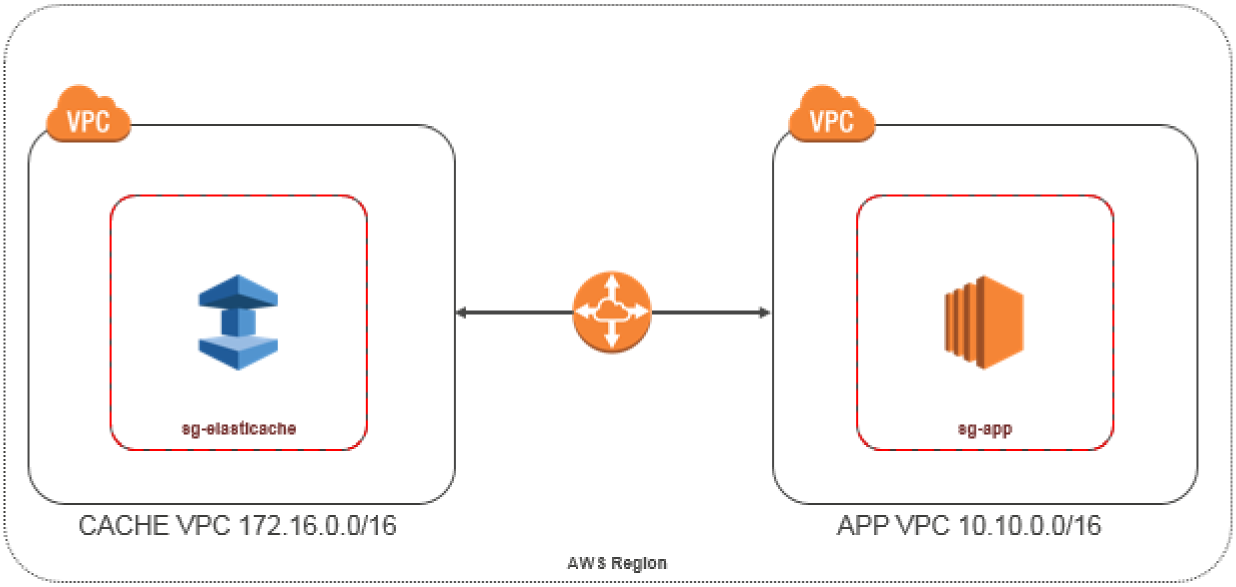
When your cache is in a different VPC from the EC2 instance you are using to access it, there are several ways to access the cache. If the cache and EC2 instance are in different VPCs but in the same region, you can use VPC peering. If the cache and the EC2 instance are in different regions, you can create VPN connectivity between regions.

Topics

- [Accessing an ElastiCache Cache when it and the Amazon EC2 Instance are in Different Amazon VPCs in the Same Region]()
- [Accessing an ElastiCache Cache when it and the Amazon EC2 Instance are in Different Amazon VPCs in Different Regions]()

Accessing an ElastiCache Cache when it and the Amazon EC2 Instance are in Different Amazon VPCs in the Same Region

The following diagram illustrates accessing a cache by an Amazon EC2 instance in a different Amazon VPC in the same region using an Amazon VPC peering connection.



Cache accessed by an Amazon EC2 instance in a different Amazon VPC within the same Region - VPC Peering Connection

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IP addresses. Instances in either VPC can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own Amazon VPCs, or with an Amazon VPC in another AWS account within a single region. To learn more about Amazon VPC peering, see the VPC documentation.

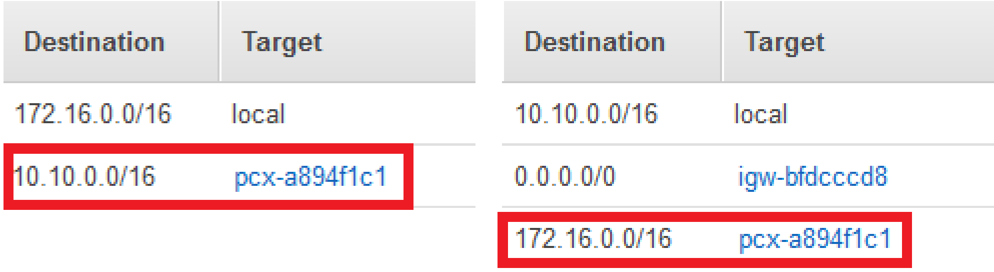
Note

DNS name resolution may fail for peered VPCs, depending on the configurations applied to the ElastiCache VPC. To resolve this, both VPCs must be enabled for DNS hostnames and DNS resolution. For more information, see Enable DNS resolution for a VPC peering connection.

To access a cache in a different Amazon VPC over peering

1. Make sure that the two VPCs do not have an overlapping IP range or you will not be able to peer them.
2. Peer the two VPCs. For more information, see Creating and Accepting an Amazon VPC Peering Connection.
3. Update your routing table. For more information, see Updating Your Route Tables for a VPC Peering Connection

   Following is what the route tables look like for the example in the preceeding diagram. Note that **pcx-a894f1c1** is the peering connection.

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 Routing Table
4. Modify the Security Group of your ElastiCache cache to allow inbound connection from the Application security group in the peered VPC. For more information, see Reference Peer VPC Security Groups.

Accessing a cache over a peering connection will incur additional data transfer costs.

Using Transit Gateway

A transit gateway enables you to attach VPCs and VPN connections in the same AWS Region and route traffic between them. A transit gateway works across AWS accounts, and you can use AWS Resource Access Manager to share your transit gateway with other accounts. After you share a transit gateway with another AWS account, the account owner can attach their VPCs to your transit gateway. A user from either account can delete the attachment at any time.

You can enable multicast on a transit gateway, and then create a transit gateway multicast domain that allows multicast traffic to be sent from your multicast source to multicast group members over VPC attachments that you associate with the domain.

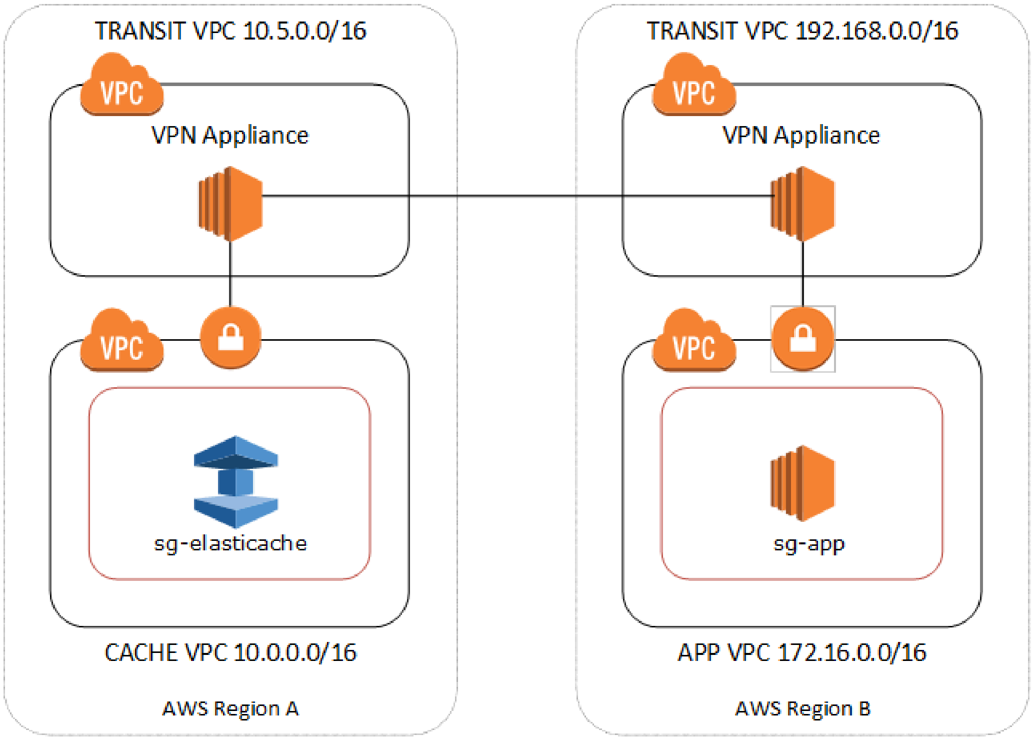
You can also create a peering connection attachment between transit gateways in different AWS Regions. This enables you to route traffic between the transit gateways' attachments across different Regions.

For more information, see Transit gateways.

Accessing an ElastiCache Cache when it and the Amazon EC2 Instance are in Different Amazon VPCs in Different Regions

Using Transit VPC

An alternative to using VPC peering, another common strategy for connecting multiple, geographically disperse VPCs and remote networks is to create a transit VPC that serves as a global network transit center. A transit VPC simplifies network management and minimizes the number of connections required to connect multiple VPCs and remote networks. This design can save time and effort and also reduce costs, as it is implemented virtually without the traditional expense of establishing a physical presence in a colocation transit hub or deploying physical network gear.



Connecting across different VPCs in different regions

Once the Transit Amazon VPC is established, an application deployed in a “spoke” VPC in one region can connect to an ElastiCache cache in a “spoke” VPC within another region.

To access a cache in a different VPC within a different AWS Region

1. Deploy a Transit VPC Solution. For more information, see, AWS Transit Gateway.
2. Update the VPC routing tables in the App and Cache VPCs to route traffic through the VGW (Virtual Private Gateway) and the VPN Appliance. In case of Dynamic Routing with Border Gateway Protocol (BGP) your routes may be automatically propagated.
3. Modify the Security Group of your ElastiCache cache to allow inbound connection from the Application instances IP range. Note that you will not be able to reference the application server Security Group in this scenario.

Accessing a cache across regions will introduce networking latencies and additional cross-region data transfer costs.

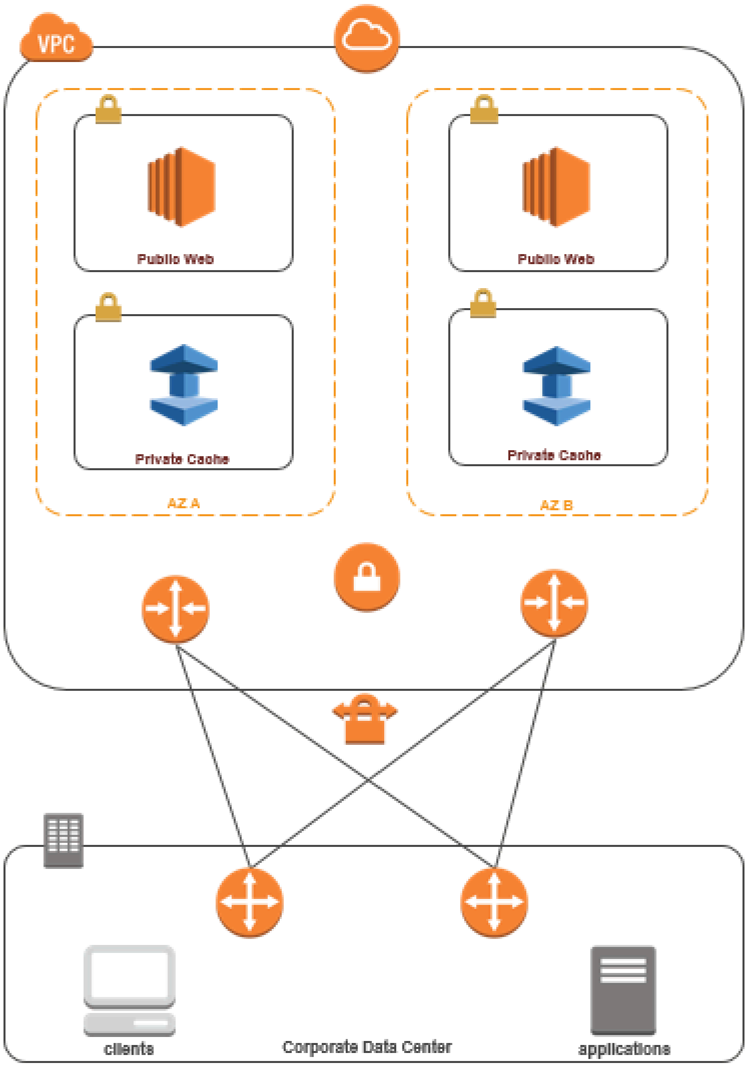
Accessing an ElastiCache Cache from an Application Running in a Customer's Data Center

Another possible scenario is a Hybrid architecture where clients or applications in the customer's data center may need to access an ElastiCache cache in the VPC. This scenario is also supported providing there is connectivity between the customers' VPC and the data center either through VPN or Direct Connect.

Topics

Accessing an ElastiCache Cache from an Application Running in a Customer's Data Center Using VPN Connectivity

The following diagram illustrates accessing an ElastiCache cache from an application running in your corporate network using VPN connections.



Connecting to ElastiCache from your data center via a VPN

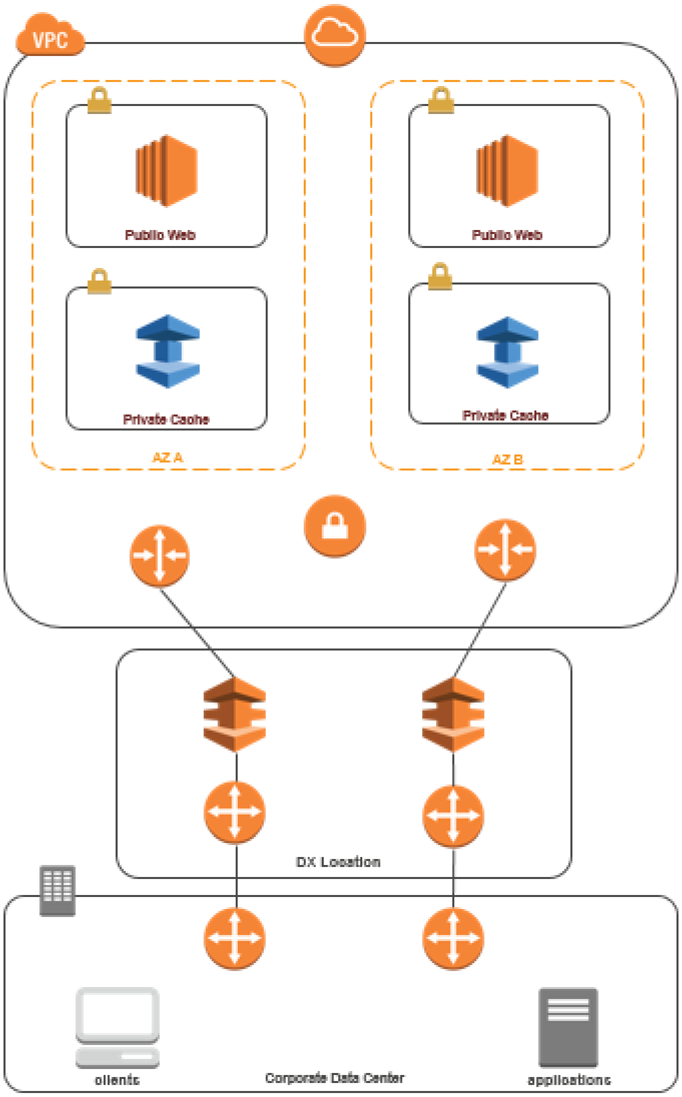
To access a cache in a VPC from on-prem application over VPN connection

1. Establish VPN Connectivity by adding a hardware Virtual Private Gateway to your VPC. For more information, see Adding a Hardware Virtual Private Gateway to Your VPC.
2. Update the VPC routing table for the subnet where your ElastiCache cache is deployed to allow traffic from your on-premises application server. In case of Dynamic Routing with BGP your routes may be automatically propagated.
3. Modify the Security Group of your ElastiCache cache to allow inbound connection from the on-premises application servers.

Accessing a cache over a VPN connection will introduce networking latencies and additional data transfer costs.

Accessing an ElastiCache Cache from an Application Running in a Customer's Data Center Using Direct Connect

The following diagram illustrates accessing an ElastiCache cache from an application running on your corporate network using Direct Connect.



Connecting to ElastiCache from your data center via Direct Connect

To access an ElastiCache cache from an application running in your network using Direct Connect

1. Establish Direct Connect connectivity. For more information, see, Getting Started with AWS Direct Connect.
2. Modify the Security Group of your ElastiCache cache to allow inbound connection from the on-premises application servers.

Accessing a cache over DX connection may introduce networking latencies and additional data transfer charges.

Creating a Virtual Private Cloud (VPC)

In this example, you create an Amazon VPC with a private subnet for each Availability Zone.

Creating an Amazon VPC (Console)

1. Sign in to the AWS Management Console, and open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the VPC dashboard, choose **Create VPC**.
3. Under **Resources** to create, choose **VPC and more**.
4. Under **Number of Availability Zones (AZs)**, choose the number of Availability Zones you want to launch your subnets in.
5. Under **Number of public subnets**, choose the number of public subnets you want to add to your VPC.
6. Under **Number of private subnets**, choose the number of private subnets you want to add to your VPC.

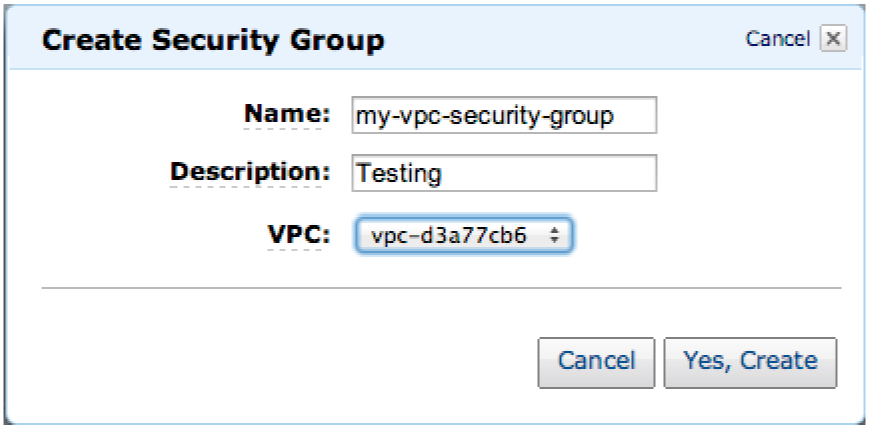
   > **Tip**
   >
   > Make a note of your subnet identifiers, and which are public and private. You will need this information later when you launch your clusters and add an Amazon EC2 instance to your Amazon VPC.

7. Create an Amazon VPC security group. You will use this group for your cache cluster and your Amazon EC2 instance.

   a. In the navigation pane of the Amazon VPC Management console, choose **Security Groups**.

   b. Choose **Create Security Group**.

   c. Type a name and a description for your security group in the corresponding boxes. In the **VPC** box, choose the identifier for your Amazon VPC.



Create Security Group

Name: my-vpc-security-group

Description: Testing

VPC: vpc-d3a77cb6

   d. When the settings are as you want them, choose **Yes, Create**.

8. Define a network ingress rule for your security group. This rule will allow you to connect to your Amazon EC2 instance using Secure Shell (SSH).

   a. In the navigation list, choose **Security Groups**.

   b. Find your security group in the list, and then choose it.

   c. Under **Security Group**, choose the **Inbound** tab. In the **Create a new rule** box, choose **SSH**, and then choose **Add Rule**.

   d. Set the following values for your new inbound rule to allow HTTP access:

      - Type: HTTP
      - Source: 0.0.0.0/0

      Choose **Apply Rule Changes**.

Now you are ready to create a cache subnet group and launch a cache cluster in your Amazon VPC.

- Creating a subnet group
- Creating a Memcached cluster (console).

Connecting to a cache running in an Amazon VPC

This example shows how to launch an Amazon EC2 instance in your Amazon VPC. You can then log in to this instance and access the ElastiCache cache that is running in the Amazon VPC.

Connecting to a cache running in an Amazon VPC (Console)

In this example, you create an Amazon EC2 instance in your Amazon VPC. You can use this Amazon EC2 instance to connect to cache nodes running in the Amazon VPC.

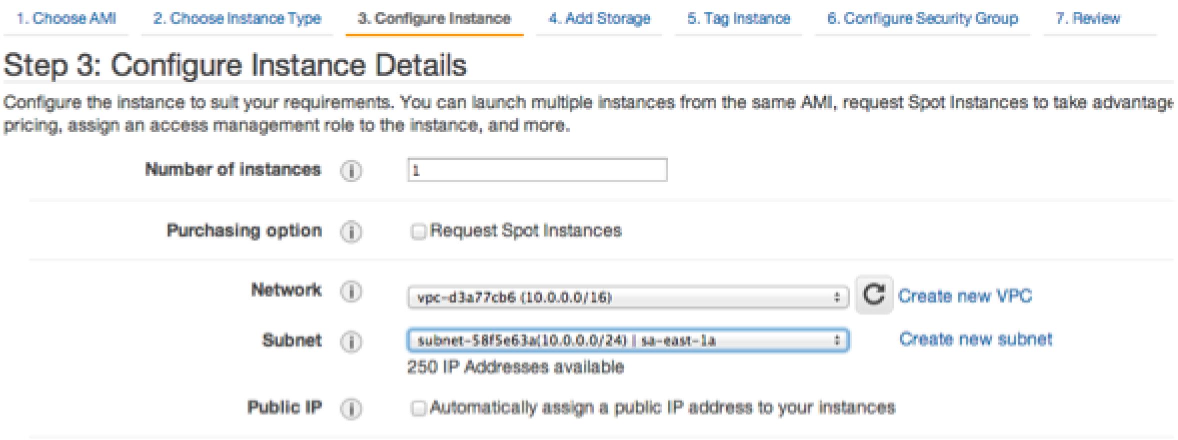
> **Note**
>
> For information about using Amazon EC2, see the Amazon EC2 Getting Started Guide in the Amazon EC2 documentation.

To create an Amazon EC2 instance in your Amazon VPC using the Amazon EC2 console

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the console, choose **Launch Instance** and follow these steps:
3. On the **Choose an Amazon Machine Image (AMI)** page, choose the 64-bit Amazon Linux AMI, and then choose **Select**.
4. On the **Choose an Instance Type** page, choose **3. Configure Instance**.
5. On the **Configure Instance Details** page, make the following selections:

    a. In the **Network** list, choose your Amazon VPC.

    b. In the **Subnet** list, choose your public subnet.



Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request Spot Instances to take advantage pricing, assign an access management role to the instance, and more.

Number of instances: 1

Purchasing option: □ Request Spot Instances

Network: vpc-d3a77cb6 (10.0.0.0/16)

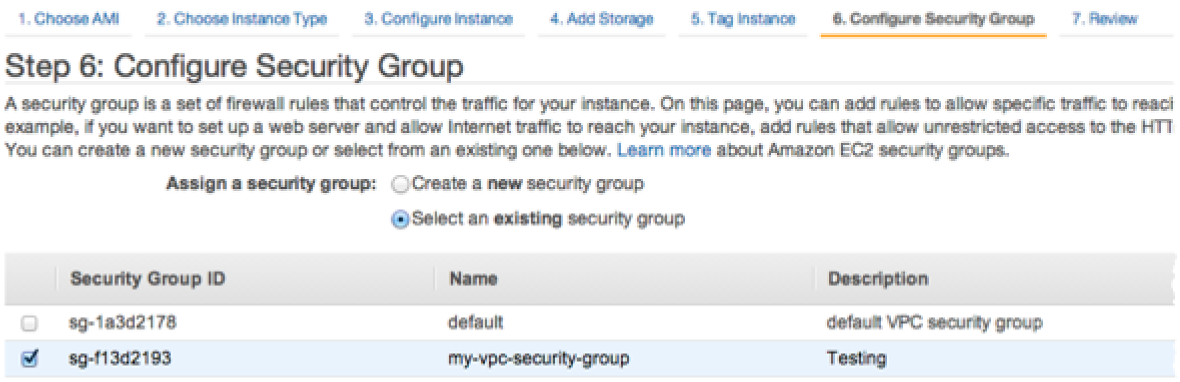
Subnet: subnet-58f5e63a(10.0.0.0/24) | sa-east-1a

250 IP Addresses available

Public IP: □ Automatically assign a public IP address to your instances

When the settings are as you want them, choose **4. Add Storage**.

6. On the **Add Storage** page, choose **5. Tag Instance**.
7. On the **Tag Instance** page, type a name for your Amazon EC2 instance, and then choose **6. Configure Security Group**.
8. On the **Configure Security Group** page, choose **Select an existing security group**. For more information on security groups, see [Amazon EC2 security groups for Linux instances](#).



Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reaci example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTT You can create a new security group or select from an existing one below. Learn more about Amazon EC2 security groups.

Assign a security group: ○ Create a **new** security group

● Select an **existing** security group

	Security Group ID	Name	Description
□	sg-1a3d2178	default	default VPC security group
☑	sg-f13d2193	my-vpc-security-group	Testing

Choose the name of your Amazon VPC security group, and then choose **Review and Launch**.

9. On the **Review Instance and Launch** page, choose **Launch**.
10. In the **Select an existing key pair or create a new key pair** window, specify a key pair that you want to use with this instance.

> **Note**
>
> For information about managing key pairs, see the [Amazon EC2 Getting Started Guide](#).

11. When you are ready to launch your Amazon EC2 instance, choose **Launch**.

You can now assign an Elastic IP address to the Amazon EC2 instance that you just created. You need to use this IP address to connect to the Amazon EC2 instance.

To assign an elastic IP address (Console)

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation list, choose **Elastic IPs**.
3. Choose **Allocate Elastic IP address**.

4. In the **Allocate Elastic IP address** dialog box, accept the default **Network Border Group** and choose **Allocate** .
5. Choose the Elastic IP address that you just allocated from the list and choose **Associate Address**.
6. In the **Associate Address** dialog box, in the **Instance** box, choose the ID of the Amazon EC2 instance that you launched.

   In the **Private IP address** box, select the box to obtain the private IP address and then choose **Associate**.

   You can now use SSH to connect to the Amazon EC2 instance using the Elastic IP address that you created.

To connect to your Amazon EC2 instance

- Open a command window. At the command prompt, issue the following command, replacing *mykeypair.pem* with the name of your key pair file and *54.207.55.251* with your Elastic IP address.

  ```
  ssh -i mykeypair.pem ec2-user@54.207.55.251
  ```

  > ⚠ **Important**
  >
  > Do not log out of your Amazon EC2 instance yet.

You are now ready to interact with your ElastiCache cluster. Before you can do that, if you haven't already done so, you need to install the *telnet* utility.

To install *telnet* and interact with your cache cluster (AWS CLI)

1. Open a command window. At the command prompt, issue the following command. At the confirmation prompt, type *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. Go to the ElastiCache console at https://console.aws.amazon.com/elasticache/ and obtain the endpoint for one of the nodes in your cache cluster. For more information, see Finding connection endpoints for Memcached.
3. Use *telnet* to connect to your cache node endpoint over port 11211. Replace the hostname shown below with the hostname of your cache node.

   ```
   telnet my-cache-cluster.7wufxa.0001.use1.cache.amazonaws.com 11211
   ```

   You are now connected to the cache engine and can issue commands. In this example, you add a data item to the cache and then get it immediately afterward. Finally, you'll disconnect from the cache node.

   To store a key and a value, type the following two lines:

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

   The cache engine responds with the following:

   ```
   OK
   ```

   To retrieve the value for `mykey`, type the following:

   ```
   get mykey
   ```

The cache engine responds with the following:

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

To disconnect from the cache engine, type the following:

```
quit
```

> ⚠ **Important**
>
> To avoid incurring additional charges on your AWS account, be sure to delete any AWS resources you no longer want after trying these examples.

Amazon ElastiCache API and interface VPC endpoints (AWS PrivateLink)

You can establish a private connection between your VPC and Amazon ElastiCache API endpoints by creating an *interface VPC endpoint*. Interface endpoints are powered by AWS PrivateLink. AWS PrivateLink allows you to privately access Amazon ElastiCache API operations without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection.

Instances in your VPC don't need public IP addresses to communicate with Amazon ElastiCache API endpoints. Your instances also don't need public IP addresses to use any of the available ElastiCache API operations. Traffic between your VPC and Amazon ElastiCache doesn't leave the Amazon network. Each interface endpoint is represented by one or more elastic network interfaces in your subnets. For more information on elastic network interfaces, see Elastic network interfaces in the *Amazon EC2 User Guide*.

- For more information about VPC endpoints, see Interface VPC endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*.
- For more information about ElastiCache API operations, see ElastiCache API operations.

After you create an interface VPC endpoint, if you enable private DNS hostnames for the endpoint, the default ElastiCache endpoint (https://elasticache.*Region*.amazonaws.com) resolves to your

VPC endpoint. If you do not enable private DNS hostnames, Amazon VPC provides a DNS endpoint name that you can use in the following format:

```
VPC_Endpoint_ID.elasticache.Region.vpce.amazonaws.com
```

For more information, see Interface VPC Endpoints (AWS PrivateLink) in the *Amazon VPC User Guide*. ElastiCache supports making calls to all of its API Actions inside your VPC.

> **Note**
>
> Private DNS hostnames can be enabled for only one VPC endpoint in the VPC. If you want to create an additional VPC endpoint then private DNS hostname should be disabled for it.

Considerations for VPC endpoints

Before you set up an interface VPC endpoint for Amazon ElastiCache API endpoints, ensure that you review Interface endpoint properties and limitations in the *Amazon VPC User Guide*. All ElastiCache API operations relevant to managing Amazon ElastiCache resources are available from your VPC using AWS PrivateLink.

VPC endpoint policies are supported for ElastiCache API endpoints. By default, full access to ElastiCache API operations is allowed through the endpoint. For more information, see Controlling access to services with VPC endpoints in the *Amazon VPC User Guide*.

Creating an interface VPC endpoint for the ElastiCache API

You can create a VPC endpoint for the Amazon ElastiCache API using either the Amazon VPC console or the AWS CLI. For more information, see Creating an interface endpoint in the *Amazon VPC User Guide*.

After you create an interface VPC endpoint, you can enable private DNS hostnames for the endpoint. When you do, the default Amazon ElastiCache endpoint (https://elasticache.*Region*.amazonaws.com) resolves to your VPC endpoint. For the China (Beijing) and China (Ningxia) AWS Regions, you can make API requests with the VPC endpoint by using `elasticache.cn-north-1.amazonaws.com.cn` for Beijing and `elasticache.cn-northwest-1.amazonaws.com.cn` for Ningxia. For more information, see Accessing a service through an interface endpoint in the *Amazon VPC User Guide*.

Creating a VPC endpoint policy for the Amazon ElastiCache API

You can attach an endpoint policy to your VPC endpoint that controls access to the ElastiCache API. The policy specifies the following:

- The principal that can perform actions.
- The actions that can be performed.
- The resources on which actions can be performed.

For more information, see Controlling access to services with VPC endpoints in the *Amazon VPC User Guide*.

Example VPC endpoint policy for ElastiCache API actions

The following is an example of an endpoint policy for the ElastiCache API. When attached to an endpoint, this policy grants access to the listed ElastiCache API actions for all principals on all resources.

```
{
 "Statement": [{
  "Principal": "*",
  "Effect": "Allow",
  "Action": [
   "elasticache:CreateCacheCluster",
   "elasticache:ModifyCacheCluster"
  ],
  "Resource": "*"
 }]
}
```

Example VPC endpoint policy that denies all access from a specified AWS account

The following VPC endpoint policy denies AWS account *123456789012* all access to resources using the endpoint. The policy allows all actions from other accounts.

```
{
 "Statement": [{
   "Action": "*",
   "Effect": "Allow",
   "Resource": "*",
   "Principal": "*"
  },
  {
   "Action": "*",
   "Effect": "Deny",
   "Resource": "*",
   "Principal": {
    "AWS": [
     "123456789012"
    ]
   }
  }
 ]
}
```

Subnets and subnet groups

A *subnet group* is a collection of subnets (typically private) that you can designate for your self-designed clusters running in an Amazon Virtual Private Cloud (VPC) environment.

If you create a self-designed cluster in an Amazon VPC, you must use a subnet group. ElastiCache uses that subnet group to choose a subnet and IP addresses within that subnet to associate with your nodes.

ElastiCache provides a default IPv4 subnet group or you can choose to create a new one. For IPv6, you need to create a subnet group with an IPv6 CIDR block. If you choose **dual stack**, you then must select a Discovery IP type, either IPv6 or IPv4.

ElastiCache Serverless does not use a subnet group resource, and instead takes a list of subnets directly during creation.

This section covers how to create and leverage subnets and subnet groups to manage access to your ElastiCache resources.

For more information about subnet group usage in an Amazon VPC environment, see Accessing your cluster .

Topics

- Creating a subnet group
- Assigning a subnet group to a cache
- Modifying a subnet group

- Deleting a subnet group

Creating a subnet group

A *cache subnet group* is a collection of subnets that you may want to designate for your caches in a VPC. When launching a cache in a VPC, you need to select a cache subnet group. Then ElastiCache uses that cache subnet group to assign IP addresses within that subnet to each cache node in the cache.

When you create a new subnet group, note the number of available IP addresses. If the subnet has very few free IP addresses, you might be constrained as to how many more nodes you can add to a cluster. To resolve this issue, you can assign one or more subnets to a subnet group so that you have a sufficient number of IP addresses in your cluster's Availability Zone. After that, you can add more nodes to your cluster.

If you choose IPV4 as your network type, a default subnet group will be available or you can choose to create a new one. ElastiCache uses that subnet group to choose a subnet and IP addresses within that subnet to associate with your nodes. If you choose dual-stack or IPV6, you will be directed to create dual-stack or IPV6 subnets. For more information on network types, see Network type. For more information, see Create a subnet in your VPC.

The following procedures show you how to create a subnet group called `mysubnetgroup` (console), the AWS CLI, and the ElastiCache API.

Creating a subnet group (Console)

The following procedure shows how to create a subnet group (console).

To create a subnet group (Console)

1. Sign in to the AWS Management Console, and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the navigation list, choose **Subnet groups**.
3. Choose **Create subnet group**.
4. In the **Create subnet group** wizard, do the following. When all the settings are as you want them, choose **Create**.

   a. In the **Name** box, type a name for your subnet group.

   b. In the **Description** box, type a description for your subnet group.

   c. In the **VPC ID** box, choose your Amazon VPC.

  d. All subnets are chosen by default. In the **Selected subnets** panel, click **Manage** and select the Availability Zones or Local Zones and IDs of your private subnets, and then choose **Choose**.

5. In the confirmation message that appears, choose **Close**.

Your new subnet group appears in the **Subnet Groups** list of the ElastiCache console. At the bottom of the window you can choose the subnet group to see details, such as all of the subnets associated with this group.

Creating a subnet group (AWS CLI)

At a command prompt, use the command `create-cache-subnet-group` to create a subnet group.

For Linux, macOS, or Unix:

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

For Windows:

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

This command should produce output similar to the following:

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

For more information, see the AWS CLI topic create-cache-subnet-group.

Assigning a subnet group to a cache

After you have created a subnet group, you can launch a cache in an Amazon VPC. For more information, see the following.

- **Memcached cluster –** To launch a Memcached cluster, see Creating a Memcached cluster (console). In step 7.a (**Advanced Memcached Settings**), choose a VPC subnet group.

Modifying a subnet group

You can modify a subnet group's description, or modify the list of subnet IDs associated with the subnet group. You cannot delete a subnet ID from a subnet group if a cache is currently using that subnet.

The following procedures show you how to modify a subnet group.

Modifying subnet groups (Console)

To modify a subnet group

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the navigation pane, choose **Subnet groups**.
3. In the list of subnet groups, select the radio button of the one you want to modify and choose **Modify**.
4. In the **Selected subnets** panel, choose **Manage**.
5. Make any changes to the selected subnets and click **Choose**.
6. Click **Save changes** to save your changes.

Modifying subnet groups (AWS CLI)

At a command prompt, use the command `modify-cache-subnet-group` to modify a subnet group.

For Linux, macOS, or Unix:

```
aws elasticache modify-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup \
    --cache-subnet-group-description "New description" \
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

For Windows:

```
aws elasticache modify-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup ^
    --cache-subnet-group-description "New description" ^
    --subnet-ids "subnet-42df9c3a" "subnet-48fc21a9"
```

This command should produce output similar to the following:

```
{
    "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"
    }
}
```

For more information, see the AWS CLI topic modify-cache-subnet-group.

Deleting a subnet group

If you decide that you no longer need your subnet group, you can delete it. You cannot delete a subnet group if it is currently in use by a cache.

The following procedures show you how to delete a subnet group.

Deleting a subnet group (Console)

To delete a subnet group

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the navigation pane, choose **Subnet groups**.
3. In the list of subnet groups, choose the one you want to delete and then choose **Delete**.
4. When you are asked to confirm this operation, type the name of the subnet group in the text input field and choose **Delete**.

Deleting a subnet group (AWS CLI)

Using the AWS CLI, call the command **delete-cache-subnet-group** with the following parameter:

- `--cache-subnet-group-name` *mysubnetgroup*

For Linux, macOS, or Unix:

```
aws elasticache delete-cache-subnet-group \
    --cache-subnet-group-name mysubnetgroup
```

For Windows:

```
aws elasticache delete-cache-subnet-group ^
    --cache-subnet-group-name mysubnetgroup
```

This command produces no output.

For more information, see the AWS CLI topic delete-cache-subnet-group.

Identity and Access Management for Amazon ElastiCache

AWS Identity and Access Management (IAM) is an AWS service that helps an administrator securely control access to AWS resources. IAM administrators control who can be *authenticated* (signed in) and *authorized* (have permissions) to use ElastiCache resources. IAM is an AWS service that you can use with no additional charge.

Topics

Audience

How you use AWS Identity and Access Management (IAM) differs, depending on the work that you do in ElastiCache.

Service user – If you use the ElastiCache service to do your job, then your administrator provides you with the credentials and permissions that you need. As you use more ElastiCache features to do your work, you might need additional permissions. Understanding how access is managed can help you request the right permissions from your administrator. If you cannot access a feature in ElastiCache, see Troubleshooting Amazon ElastiCache identity and access.

Service administrator – If you're in charge of ElastiCache resources at your company, you probably have full access to ElastiCache. It's your job to determine which ElastiCache features and resources your service users should access. You must then submit requests to your IAM administrator to change the permissions of your service users. Review the information on this page to understand the basic concepts of IAM. To learn more about how your company can use IAM with ElastiCache, see How Amazon ElastiCache works with IAM.

IAM administrator – If you're an IAM administrator, you might want to learn details about how you can write policies to manage access to ElastiCache. To view example ElastiCache identity-based policies that you can use in IAM, see Identity-based policy examples for Amazon ElastiCache.

Authenticating with identities

Authentication is how you sign in to AWS using your identity credentials. You must be *authenticated* (signed in to AWS) as the AWS account root user, as an IAM user, or by assuming an IAM role.

You can sign in to AWS as a federated identity by using credentials provided through an identity source. AWS IAM Identity Center (IAM Identity Center) users, your company's single sign-on authentication, and your Google or Facebook credentials are examples of federated identities. When you sign in as a federated identity, your administrator previously set up identity federation using IAM roles. When you access AWS by using federation, you are indirectly assuming a role.

Depending on the type of user you are, you can sign in to the AWS Management Console or the AWS access portal. For more information about signing in to AWS, see How to sign in to your AWS account in the *AWS Sign-In User Guide*.

If you access AWS programmatically, AWS provides a software development kit (SDK) and a command line interface (CLI) to cryptographically sign your requests by using your credentials. If you don't use AWS tools, you must sign requests yourself. For more information about using the recommended method to sign requests yourself, see Signing AWS API requests in the *IAM User Guide*.

Regardless of the authentication method that you use, you might be required to provide additional security information. For example, AWS recommends that you use multi-factor authentication (MFA) to increase the security of your account. To learn more, see Multi-factor authentication in the *AWS IAM Identity Center User Guide* and Using multi-factor authentication (MFA) in AWS in the *IAM User Guide*.

AWS account root user

When you create an AWS account, you begin with one sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account *root user* and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you don't use the root user for your everyday tasks. Safeguard your root user credentials and use them to perform the tasks that only the root user can perform. For the complete list of tasks that require you to sign in as the root user, see Tasks that require root user credentials in the *IAM User Guide*.

Federated identity

As a best practice, require human users, including users that require administrator access, to use federation with an identity provider to access AWS services by using temporary credentials.

A *federated identity* is a user from your enterprise user directory, a web identity provider, the AWS Directory Service, the Identity Center directory, or any user that accesses AWS services by using credentials provided through an identity source. When federated identities access AWS accounts, they assume roles, and the roles provide temporary credentials.

For centralized access management, we recommend that you use AWS IAM Identity Center. You can create users and groups in IAM Identity Center, or you can connect and synchronize to a set of users and groups in your own identity source for use across all your AWS accounts and applications. For information about IAM Identity Center, see What is IAM Identity Center? in the *AWS IAM Identity Center User Guide*.

IAM users and groups

An *IAM user* is an identity within your AWS account that has specific permissions for a single person or application. Where possible, we recommend relying on temporary credentials instead of creating IAM users who have long-term credentials such as passwords and access keys. However, if you have specific use cases that require long-term credentials with IAM users, we recommend that you rotate access keys. For more information, see Rotate access keys regularly for use cases that require long-term credentials in the *IAM User Guide*.

An *IAM group* is an identity that specifies a collection of IAM users. You can't sign in as a group. You can use groups to specify permissions for multiple users at a time. Groups make permissions easier to manage for large sets of users. For example, you could have a group named *IAMAdmins* and give that group permissions to administer IAM resources.

Users are different from roles. A user is uniquely associated with one person or application, but a role is intended to be assumable by anyone who needs it. Users have permanent long-term credentials, but roles provide temporary credentials. To learn more, see When to create an IAM user (instead of a role) in the *IAM User Guide*.

IAM roles

An *IAM role* is an identity within your AWS account that has specific permissions. It is similar to an IAM user, but is not associated with a specific person. You can temporarily assume an IAM role in the AWS Management Console by switching roles. You can assume a role by calling an AWS CLI or AWS API operation or by using a custom URL. For more information about methods for using roles, see Using IAM roles in the *IAM User Guide*.

IAM roles with temporary credentials are useful in the following situations:

- **Federated user access –** To assign permissions to a federated identity, you create a role and define permissions for the role. When a federated identity authenticates, the identity is associated with the role and is granted the permissions that are defined by the role. For information about roles for federation, see Creating a role for a third-party Identity Provider in the *IAM User Guide*. If you use IAM Identity Center, you configure a permission set. To control what your identities can access after they authenticate, IAM Identity Center correlates the permission set to a role in IAM. For information about permissions sets, see Permission sets in the *AWS IAM Identity Center User Guide*.
- **Temporary IAM user permissions –** An IAM user or role can assume an IAM role to temporarily take on different permissions for a specific task.
- **Cross-account access –** You can use an IAM role to allow someone (a trusted principal) in a different account to access resources in your account. Roles are the primary way to grant cross-account access. However, with some AWS services, you can attach a policy directly to a resource (instead of using a role as a proxy). To learn the difference between roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.
- **Cross-service access –** Some AWS services use features in other AWS services. For example, when you make a call in a service, it's common for that service to run applications in Amazon EC2 or store objects in Amazon S3. A service might do this using the calling principal's permissions, using a service role, or using a service-linked role.
  - **Forward access sessions (FAS) –** When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must

have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

  - **Service role –** A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.
  - **Service-linked role –** A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.
- **Applications running on Amazon EC2 –** You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS CLI or AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM role to grant permissions to applications running on Amazon EC2 instances in the *IAM User Guide*.

To learn whether to use IAM roles or IAM users, see When to create an IAM role (instead of a user) in the *IAM User Guide*.

Managing access using policies

You control access in AWS by creating policies and attaching them to AWS identities or resources. A policy is an object in AWS that, when associated with an identity or resource, defines their permissions. AWS evaluates these policies when a principal (user, root user, or role session) makes a request. Permissions in the policies determine whether the request is allowed or denied. Most policies are stored in AWS as JSON documents. For more information about the structure and contents of JSON policy documents, see Overview of JSON policies in the *IAM User Guide*.

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

By default, users and roles have no permissions. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

IAM policies define permissions for an action regardless of the method that you use to perform the operation. For example, suppose that you have a policy that allows the `iam:GetRole` action. A user with that policy can get role information from the AWS Management Console, the AWS CLI, or the AWS API.

Identity-based policies

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

Identity-based policies can be further categorized as *inline policies* or *managed policies*. Inline policies are embedded directly into a single user, group, or role. Managed policies are standalone policies that you can attach to multiple users, groups, and roles in your AWS account. Managed policies include AWS managed policies and customer managed policies. To learn how to choose between a managed policy or an inline policy, see Choosing between managed policies and inline policies in the *IAM User Guide*.

Resource-based policies

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

Resource-based policies are inline policies that are located in that service. You can't use AWS managed policies from IAM in a resource-based policy.

Access control lists (ACLs)

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Amazon S3, AWS WAF, and Amazon VPC are examples of services that support ACLs. To learn more about ACLs, see Access control list (ACL) overview in the *Amazon Simple Storage Service Developer Guide*.

Other policy types

AWS supports additional, less-common policy types. These policy types can set the maximum permissions granted to you by the more common policy types.

- **Permissions boundaries** – A permissions boundary is an advanced feature in which you set the maximum permissions that an identity-based policy can grant to an IAM entity (IAM user or role). You can set a permissions boundary for an entity. The resulting permissions are the intersection of an entity's identity-based policies and its permissions boundaries. Resource-based policies that specify the user or role in the `Principal` field are not limited by the permissions boundary. An explicit deny in any of these policies overrides the allow. For more information about permissions boundaries, see Permissions boundaries for IAM entities in the *IAM User Guide*.
- **Service control policies (SCPs)** – SCPs are JSON policies that specify the maximum permissions for an organization or organizational unit (OU) in AWS Organizations. AWS Organizations is a service for grouping and centrally managing multiple AWS accounts that your business owns. If you enable all features in an organization, then you can apply service control policies (SCPs) to any or all of your accounts. The SCP limits permissions for entities in member accounts, including each AWS account root user. For more information about Organizations and SCPs, see How SCPs work in the *AWS Organizations User Guide*.
- **Session policies** – Session policies are advanced policies that you pass as a parameter when you programmatically create a temporary session for a role or federated user. The resulting session's permissions are the intersection of the user or role's identity-based policies and the session policies. Permissions can also come from a resource-based policy. An explicit deny in any of these policies overrides the allow. For more information, see Session policies in the *IAM User Guide*.

Multiple policy types

When multiple types of policies apply to a request, the resulting permissions are more complicated to understand. To learn how AWS determines whether to allow a request when multiple policy types are involved, see Policy evaluation logic in the *IAM User Guide*.

How Amazon ElastiCache works with IAM

Before you use IAM to manage access to ElastiCache, learn what IAM features are available to use with ElastiCache.

IAM features you can use with Amazon ElastiCache

IAM feature	ElastiCache support
Identity-based policies	Yes
Resource-based policies	No
Policy actions	Yes
Policy resources	Yes
Policy condition keys	Yes
ACLs	Yes
ABAC (tags in policies)	Yes
Temporary credentials	Yes
Principal permissions	Yes
Service roles	Yes
Service-linked roles	Yes

To get a high-level view of how ElastiCache and other AWS services work with most IAM features, see AWS services that work with IAM in the *IAM User Guide*.

Identity-based policies for ElastiCache

Supports identity-based policies	Yes

Identity-based policies are JSON permissions policy documents that you can attach to an identity, such as an IAM user, group of users, or role. These policies control what actions users and roles can perform, on which resources, and under what conditions. To learn how to create an identity-based policy, see Creating IAM policies in the *IAM User Guide*.

With IAM identity-based policies, you can specify allowed or denied actions and resources as well as the conditions under which actions are allowed or denied. You can't specify the principal in an identity-based policy because it applies to the user or role to which it is attached. To learn about all of the elements that you can use in a JSON policy, see IAM JSON policy elements reference in the *IAM User Guide*.

Identity-based policy examples for ElastiCache

To view examples of ElastiCache identity-based policies, see Identity-based policy examples for Amazon ElastiCache.

Resource-based policies within ElastiCache

Supports resource-based policies	No

Resource-based policies are JSON policy documents that you attach to a resource. Examples of resource-based policies are IAM *role trust policies* and Amazon S3 *bucket policies*. In services that support resource-based policies, service administrators can use them to control access to a specific resource. For the resource where the policy is attached, the policy defines what actions a specified principal can perform on that resource and under what conditions. You must specify a principal in a resource-based policy. Principals can include accounts, users, roles, federated users, or AWS services.

To enable cross-account access, you can specify an entire account or IAM entities in another account as the principal in a resource-based policy. Adding a cross-account principal to a resource-based policy is only half of establishing the trust relationship. When the principal and the resource are in different AWS accounts, an IAM administrator in the trusted account must also grant the principal entity (user or role) permission to access the resource. They grant permission by attaching an identity-based policy to the entity. However, if a resource-based policy grants access to a principal in the same account, no additional identity-based policy is required. For more information, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Policy actions for ElastiCache

Supports policy actions	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Action` element of a JSON policy describes the actions that you can use to allow or deny access in a policy. Policy actions usually have the same name as the associated AWS API operation. There are some exceptions, such as *permission-only actions* that don't have a matching API operation. There are also some operations that require multiple actions in a policy. These additional actions are called *dependent actions*.

Include actions in a policy to grant permissions to perform the associated operation.

To see a list of ElastiCache actions, see Actions Defined by Amazon ElastiCache in the *Service Authorization Reference*.

Policy actions in ElastiCache use the following prefix before the action:

```
elasticache
```

To specify multiple actions in a single statement, separate them with commas.

```
"Action": [
      "elasticache:action1",
      "elasticache:action2"
         ]
```

You can specify multiple actions using wildcards (*). For example, to specify all actions that begin with the word `Describe`, include the following action:

```
"Action": "elasticache:Describe*"
```

To view examples of ElastiCache identity-based policies, see Identity-based policy examples for Amazon ElastiCache.

Policy resources for ElastiCache

Supports policy resources	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Resource` JSON policy element specifies the object or objects to which the action applies. Statements must include either a `Resource` or a `NotResource` element. As a best practice, specify a resource using its Amazon Resource Name (ARN). You can do this for actions that support a specific resource type, known as *resource-level permissions*.

For actions that don't support resource-level permissions, such as listing operations, use a wildcard (*) to indicate that the statement applies to all resources.

```
"Resource": "*"
```

To see a list of ElastiCache resource types and their ARNs, see Resources Defined by Amazon ElastiCache in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions Defined by Amazon ElastiCache.

To view examples of ElastiCache identity-based policies, see Identity-based policy examples for Amazon ElastiCache.

Policy condition keys for ElastiCache

Supports service-specific policy condition keys	Yes

Administrators can use AWS JSON policies to specify who has access to what. That is, which **principal** can perform **actions** on what **resources**, and under what **conditions**.

The `Condition` element (or `Condition` *block*) lets you specify conditions in which a statement is in effect. The `Condition` element is optional. You can create conditional expressions that use condition operators, such as equals or less than, to match the condition in the policy with values in the request.

If you specify multiple Condition elements in a statement, or multiple keys in a single Condition element, AWS evaluates them using a logical AND operation. If you specify multiple values for a single condition key, AWS evaluates the condition using a logical OR operation. All of the conditions must be met before the statement's permissions are granted.

You can also use placeholder variables when you specify conditions. For example, you can grant an IAM user permission to access a resource only if it is tagged with their IAM user name. For more information, see IAM policy elements: variables and tags in the *IAM User Guide*.

AWS supports global condition keys and service-specific condition keys. To see all AWS global condition keys, see AWS global condition context keys in the *IAM User Guide*.

To see a list of ElastiCache condition keys, see Condition Keys for Amazon ElastiCache in the *Service Authorization Reference*. To learn with which actions and resources you can use a condition key, see Actions Defined by Amazon ElastiCache.

To view examples of ElastiCache identity-based policies, see Identity-based policy examples for Amazon ElastiCache.

Access control lists (ACLs) in ElastiCache

Supports ACLs	Yes

Access control lists (ACLs) control which principals (account members, users, or roles) have permissions to access a resource. ACLs are similar to resource-based policies, although they do not use the JSON policy document format.

Attribute-based access control (ABAC) with ElastiCache

Supports ABAC (tags in policies)	Yes

Attribute-based access control (ABAC) is an authorization strategy that defines permissions based on attributes. In AWS, these attributes are called *tags*. You can attach tags to IAM entities (users or roles) and to many AWS resources. Tagging entities and resources is the first step of ABAC. Then you design ABAC policies to allow operations when the principal's tag matches the tag on the resource that they are trying to access.

ABAC is helpful in environments that are growing rapidly and helps with situations where policy management becomes cumbersome.

To control access based on tags, you provide tag information in the condition element of a policy using the `aws:ResourceTag/key-name`, `aws:RequestTag/key-name`, or `aws:TagKeys` condition keys.

If a service supports all three condition keys for every resource type, then the value is **Yes** for the service. If a service supports all three condition keys for only some resource types, then the value is **Partial**.

For more information about ABAC, see What is ABAC? in the *IAM User Guide*. To view a tutorial with steps for setting up ABAC, see Use attribute-based access control (ABAC) in the *IAM User Guide*.

Using Temporary credentials with ElastiCache

Supports temporary credentials	Yes

Some AWS services don't work when you sign in using temporary credentials. For additional information, including which AWS services work with temporary credentials, see AWS services that work with IAM in the *IAM User Guide*.

You are using temporary credentials if you sign in to the AWS Management Console using any method except a user name and password. For example, when you access AWS using your company's single sign-on (SSO) link, that process automatically creates temporary credentials. You also automatically create temporary credentials when you sign in to the console as a user and then switch roles. For more information about switching roles, see Switching to a role (console) in the *IAM User Guide*.

You can manually create temporary credentials using the AWS CLI or AWS API. You can then use those temporary credentials to access AWS. AWS recommends that you dynamically generate temporary credentials instead of using long-term access keys. For more information, see Temporary security credentials in IAM.

Cross-service principal permissions for ElastiCache

Supports forward access sessions (FAS)	Yes

When you use an IAM user or role to perform actions in AWS, you are considered a principal. When you use some services, you might perform an action that then initiates another action in a different service. FAS uses the permissions of the principal calling an AWS service, combined with the requesting AWS service to make requests to downstream services. FAS requests are only made when a service receives a request that requires interactions with other AWS services or resources to complete. In this case, you must have permissions to perform both actions. For policy details when making FAS requests, see Forward access sessions.

Service roles for ElastiCache

Supports service roles	Yes

A service role is an IAM role that a service assumes to perform actions on your behalf. An IAM administrator can create, modify, and delete a service role from within IAM. For more information, see Creating a role to delegate permissions to an AWS service in the *IAM User Guide*.

> ⚠ **Warning**
>
> Changing the permissions for a service role might break ElastiCache functionality. Edit service roles only when ElastiCache provides guidance to do so.

Service-linked roles for ElastiCache

Supports service-linked roles	Yes

A service-linked role is a type of service role that is linked to an AWS service. The service can assume the role to perform an action on your behalf. Service-linked roles appear in your AWS account and are owned by the service. An IAM administrator can view, but not edit the permissions for service-linked roles.

For details about creating or managing service-linked roles, see AWS services that work with IAM. Find a service in the table that includes a `Yes` in the **Service-linked role** column. Choose the **Yes** link to view the service-linked role documentation for that service.

Identity-based policy examples for Amazon ElastiCache

By default, users and roles don't have permission to create or modify ElastiCache resources. They also can't perform tasks by using the AWS Management Console, AWS Command Line Interface (AWS CLI), or AWS API. To grant users permission to perform actions on the resources that they need, an IAM administrator can create IAM policies. The administrator can then add the IAM policies to roles, and users can assume the roles.

To learn how to create an IAM identity-based policy by using these example JSON policy documents, see Creating IAM policies in the *IAM User Guide*.

For details about actions and resource types defined by ElastiCache, including the format of the ARNs for each of the resource types, see Actions, Resources, and Condition Keys for Amazon ElastiCache in the *Service Authorization Reference*.

Topics

Policy best practices

Identity-based policies determine whether someone can create, access, or delete ElastiCache resources in your account. These actions can incur costs for your AWS account. When you create or edit identity-based policies, follow these guidelines and recommendations:

- **Get started with AWS managed policies and move toward least-privilege permissions** – To get started granting permissions to your users and workloads, use the *AWS managed policies* that grant permissions for many common use cases. They are available in your AWS account. We recommend that you reduce permissions further by defining AWS customer managed policies that are specific to your use cases. For more information, see AWS managed policies or AWS managed policies for job functions in the *IAM User Guide*.
- **Apply least-privilege permissions** – When you set permissions with IAM policies, grant only the permissions required to perform a task. You do this by defining the actions that can be taken on specific resources under specific conditions, also known as *least-privilege permissions*. For more information about using IAM to apply permissions, see Policies and permissions in IAM in the *IAM User Guide*.

- **Use conditions in IAM policies to further restrict access** – You can add a condition to your policies to limit access to actions and resources. For example, you can write a policy condition to specify that all requests must be sent using SSL. You can also use conditions to grant access to service actions if they are used through a specific AWS service, such as AWS CloudFormation. For more information, see IAM JSON policy elements: Condition in the *IAM User Guide*.
- **Use IAM Access Analyzer to validate your IAM policies to ensure secure and functional permissions** – IAM Access Analyzer validates new and existing policies so that the policies adhere to the IAM policy language (JSON) and IAM best practices. IAM Access Analyzer provides more than 100 policy checks and actionable recommendations to help you author secure and functional policies. For more information, see IAM Access Analyzer policy validation in the *IAM User Guide*.
- **Require multi-factor authentication (MFA)** – If you have a scenario that requires IAM users or a root user in your AWS account, turn on MFA for additional security. To require MFA when API operations are called, add MFA conditions to your policies. For more information, see Configuring MFA-protected API access in the *IAM User Guide*.

For more information about best practices in IAM, see Security best practices in IAM in the *IAM User Guide*.

Using the ElastiCache console

To access the Amazon ElastiCache console, you must have a minimum set of permissions. These permissions must allow you to list and view details about the ElastiCache resources in your AWS account. If you create an identity-based policy that is more restrictive than the minimum required permissions, the console won't function as intended for entities (users or roles) with that policy.

You don't need to allow minimum console permissions for users that are making calls only to the AWS CLI or the AWS API. Instead, allow access to only the actions that match the API operation that they're trying to perform.

To ensure that users and roles can still use the ElastiCache console, also attach the ElastiCache `ConsoleAccess` or `ReadOnly` AWS managed policy to the entities. For more information, see Adding permissions to a user in the *IAM User Guide*.

Allow users to view their own permissions

This example shows how you might create a policy that allows IAM users to view the inline and managed policies that are attached to their user identity. This policy includes permissions to complete this action on the console or programmatically using the AWS CLI or AWS API.

```
{
    "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": "*"
        }
    ]
}
```

Troubleshooting Amazon ElastiCache identity and access

Use the following information to help you diagnose and fix common issues that you might encounter when working with ElastiCache and IAM.

Topics

I am not authorized to perform an action in ElastiCache

If the AWS Management Console tells you that you're not authorized to perform an action, then you must contact your administrator for assistance. Your administrator is the person that provided you with your user name and password.

The following example error occurs when the `mateojackson` user tries to use the console to view details about a fictional *`my-example-widget`* resource but does not have the fictional `elasticache:`*`GetWidget`* permissions.

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 elasticache:GetWidget on resource: my-example-widget
```

In this case, Mateo asks his administrator to update his policies to allow him to access the *`my-example-widget`* resource using the `elasticache:`*`GetWidget`* action.

I am not authorized to perform iam:PassRole

If you receive an error that you're not authorized to perform the `iam:PassRole` action, your policies must be updated to allow you to pass a role to ElastiCache.

Some AWS services allow you to pass an existing role to that service instead of creating a new service role or service-linked role. To do this, you must have permissions to pass the role to the service.

The following example error occurs when an IAM user named `marymajor` tries to use the console to perform an action in ElastiCache. However, the action requires the service to have permissions that are granted by a service role. Mary does not have permissions to pass the role to the service.

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

In this case, Mary's policies must be updated to allow her to perform the `iam:PassRole` action.

If you need help, contact your AWS administrator. Your administrator is the person who provided you with your sign-in credentials.

I want to allow people outside of my AWS account to access my ElastiCache resources

You can create a role that users in other accounts or people outside of your organization can use to access your resources. You can specify who is trusted to assume the role. For services that support resource-based policies or access control lists (ACLs), you can use those policies to grant people access to your resources.

To learn more, consult the following:

- To learn whether ElastiCache supports these features, see How Amazon ElastiCache works with IAM.
- To learn how to provide access to your resources across AWS accounts that you own, see Providing access to an IAM user in another AWS account that you own in the *IAM User Guide*.
- To learn how to provide access to your resources to third-party AWS accounts, see Providing access to AWS accounts owned by third parties in the *IAM User Guide*.
- To learn how to provide access through identity federation, see Providing access to externally authenticated users (identity federation) in the *IAM User Guide*.
- To learn the difference between using roles and resource-based policies for cross-account access, see How IAM roles differ from resource-based policies in the *IAM User Guide*.

Access control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access ElastiCache resources. For example, you must have permissions to create an ElastiCache cluster.

The following sections describe how to manage permissions for ElastiCache. We recommend that you read the overview first.

- Overview of managing access permissions to your ElastiCache resources
- Using identity-based policies (IAM policies) for Amazon ElastiCache

Overview of managing access permissions to your ElastiCache resources

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles). In addition, Amazon ElastiCache also supports attaching permissions policies to resources.

> **Note**
>
> An *account administrator* (or administrator user) is a user with administrator privileges. For more information, see IAM Best Practices in the *IAM User Guide*.

To provide access, add permissions to your users, groups, or roles:

- Users and groups in AWS IAM Identity Center:

  Create a permission set. Follow the instructions in Create a permission set in the *AWS IAM Identity Center User Guide*.
- Users managed in IAM through an identity provider:

  Create a role for identity federation. Follow the instructions in Creating a role for a third-party identity provider (federation) in the *IAM User Guide*.
- IAM users:
  - Create a role that your user can assume. Follow the instructions in Creating a role for an IAM user in the *IAM User Guide*.
  - (Not recommended) Attach a policy directly to a user or add a user to a user group. Follow the instructions in Adding permissions to a user (console) in the *IAM User Guide*.

Topics

- Resource-level permissions
- Using condition keys
- Using Service-Linked Roles for Amazon ElastiCache
- ElastiCache API permissions: Actions, resources, and conditions reference

Amazon ElastiCache resources and operations

To see a list of ElastiCache resource types and their ARNs, see Resources Defined by Amazon ElastiCache in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions Defined by Amazon ElastiCache.

Understanding resource ownership

A *resource owner* is the AWS account that created the resource. That is, the resource owner is the AWS account of the principal entity that authenticates the request that creates the resource. A *principal entity* can be the root account, an IAM user, or an IAM role). The following examples illustrate how this works:

- Suppose that you use the root account credentials of your AWS account to create a cache cluster. In this case, your AWS account is the owner of the resource. In ElastiCache, the resource is the cache cluster.
- Suppose that you create an IAM user in your AWS account and grant permissions to create a cache cluster to that user. In this case, the user can create a cache cluster. However, your AWS account, to which the user belongs, owns the cache cluster resource.
- Suppose that you create an IAM role in your AWS account with permissions to create a cache cluster. In this case, anyone who can assume the role can create a cache cluster. Your AWS account, to which the role belongs, owns the cache cluster resource.

Managing access to resources

A *permissions policy* describes who has access to what. The following section explains the available options for creating permissions policies.

> **Note**
>
> This section discusses using IAM in the context of Amazon ElastiCache. It doesn't provide detailed information about the IAM service. For complete IAM documentation, see What Is

IAM? in the *IAM User Guide*. For information about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the *IAM User Guide*.

Policies attached to an IAM identity are referred to as *identity-based* policies (IAM policies). Policies attached to a resource are referred to as *resource-based* policies.

Topics

Identity-based policies (IAM policies)

You can attach policies to IAM identities. For example, you can do the following:

- **Attach a permissions policy to a user or a group in your account –** An account administrator can use a permissions policy that is associated with a particular user to grant permissions. In this case, the permissions are for that user to create an ElastiCache resource, such as a cache cluster, parameter group, or security group.
- **Attach a permissions policy to a role (grant cross-account permissions) –** You can attach an identity-based permissions policy to an IAM role to grant cross-account permissions. For example, the administrator in Account A can create a role to grant cross-account permissions to another AWS account (for example, Account B) or an AWS service as follows:

  1. Account A administrator creates an IAM role and attaches a permissions policy to the role that grants permissions on resources in Account A.
  2. Account A administrator attaches a trust policy to the role identifying Account B as the principal who can assume the role.
  3. Account B administrator can then delegate permissions to assume the role to any users in Account B. Doing this allows users in Account B to create or access resources in Account A. In some cases, you might want to grant an AWS service permissions to assume the role. To support this approach, the principal in the trust policy can also be an AWS service principal.

  For more information about using IAM to delegate permissions, see Access Management in the *IAM User Guide*.

The following is an example policy that allows a user to perform the `DescribeCacheClusters` action for your AWS account. ElastiCache also supports identifying specific resources using the resource ARNs for API actions. (This approach is also referred to as resource-level permissions).

```
{
   "Version": "2012-10-17",
   "Statement": [{
      "Sid": "DescribeCacheClusters",
      "Effect": "Allow",
      "Action": [
         "elasticache:DescribeCacheClusters"],
      "Resource": resource-arn
      }
   ]
}
```

For more information about using identity-based policies with ElastiCache, see Using identity-based policies (IAM policies) for Amazon ElastiCache. For more information about users, groups, roles, and permissions, see Identities (Users, Groups, and Roles in the *IAM User Guide*.

Specifying policy elements: Actions, effects, resources, and principals

For each Amazon ElastiCache resource (see Amazon ElastiCache resources and operations), the service defines a set of API operations (see Actions). To grant permissions for these API operations, ElastiCache defines a set of actions that you can specify in a policy. For example, for the ElastiCache cluster resource, the following actions are defined: `CreateCacheCluster`, `DeleteCacheCluster`, and `DescribeCacheCluster`. Performing an API operation can require permissions for more than one action.

The following are the most basic policy elements:

- **Resource** – In a policy, you use an Amazon Resource Name (ARN) to identify the resource to which the policy applies. For more information, see Amazon ElastiCache resources and operations.
- **Action** – You use action keywords to identify resource operations that you want to allow or deny. For example, depending on the specified `Effect`, the `elasticache:CreateCacheCluster` permission allows or denies the user permissions to perform the Amazon ElastiCache `CreateCacheCluster` operation.
- **Effect** – You specify the effect when the user requests the specific action—this can be either allow or deny. If you don't explicitly grant access to (allow) a resource, access is implicitly denied.

You can also explicitly deny access to a resource. For example, you might do this to make sure that a user can't access a resource, even if a different policy grants access.

- **Principal** – In identity-based policies (IAM policies), the user that the policy is attached to is the implicit principal. For resource-based policies, you specify the user, account, service, or other entity that you want to receive permissions (applies to resource-based policies only).

To learn more about IAM policy syntax and descriptions, see AWS IAM Policy Reference in the *IAM User Guide*.

For a table showing all of the Amazon ElastiCache API actions, see ElastiCache API permissions: Actions, resources, and conditions reference.

Specifying conditions in a policy

When you grant permissions, you can use the IAM policy language to specify the conditions when a policy should take effect. For example, you might want a policy to be applied only after a specific date. For more information about specifying conditions in a policy language, see Condition in the *IAM User Guide*.

To express conditions, you use predefined condition keys. To use ElastiCache-specific condition keys, see Using condition keys. There are AWS-wide condition keys that you can use as appropriate. For a complete list of AWS-wide keys, see Available Keys for Conditions in the *IAM User Guide*.

AWS managed policies for Amazon ElastiCache

An AWS managed policy is a standalone policy that is created and administered by AWS. AWS managed policies are designed to provide permissions for many common use cases so that you can start assigning permissions to users, groups, and roles.

Keep in mind that AWS managed policies might not grant least-privilege permissions for your specific use cases because they're available for all AWS customers to use. We recommend that you reduce permissions further by defining customer managed policies that are specific to your use cases.

You cannot change the permissions defined in AWS managed policies. If AWS updates the permissions defined in an AWS managed policy, the update affects all principal identities (users, groups, and roles) that the policy is attached to. AWS is most likely to update an AWS managed policy when a new AWS service is launched or new API operations become available for existing services.

For more information, see AWS managed policies in the *IAM User Guide*.

AWS managed policy: ElastiCacheServiceRolePolicy

You can't attach ElastiCacheServiceRolePolicy to your IAM entities. This policy is attached to a service-linked role that allows ElastiCache to perform actions on your behalf.

This policy allows ElastiCache to manage AWS resources on your behalf as necessary for managing your cache:

- `ec2` – Manage EC2 networking resources to attach to cache nodes, including VPC endpoints (for serverless caches), Elastic Network Interfaces (ENIs) (for self-designed clusters), and security groups.
- `cloudwatch` – Emit metric data from the service into CloudWatch.
- `outposts` – Allow creation of cache nodes on AWS Outposts.

You can find the ElastiCacheServiceRolePolicy policy on the IAM console and ElastiCacheServiceRolePolicy in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AmazonElastiCacheFullAccess

You can attach the `AmazonElastiCacheFullAccess` policy to your IAM identities.

This policy allows principals full access to ElastiCache using the AWS Management Console:

- `elasticache` — Access all APIs.
- `iam` — Create service-linked role necessary for service operation.
- `ec2` — Describe dependent EC2 resources necessary for cache creation (VPC, subnet, security group) and allow creation of VPC endpoints (for serverless caches).
- `kms` — Allow usage of customer-managed CMKs for encryption-at-rest.
- `cloudwatch` — Allow access to metrics to display ElastiCache metrics in the console.
- `application-autoscaling` — Allow access to describe autoscaling policies for caches.
- `logs` — Used to populate log streams for log delivery functionality in the console.
- `firehose` — Used to populate delivery streams for log delivery functionality in the console.
- `s3` — Used to populate S3 buckets for snapshot restore functionality in the console.
- `outposts` — Used to populate AWS Outposts for cache creation in the console.
- `sns` — Used to populate SNS topics for notification functionality in the console.

You can find the AmazonElastiCacheFullAccess policy on the IAM console and AmazonElastiCacheFullAccess in the *AWS Managed Policy Reference Guide*.

AWS managed policy: AmazonElastiCacheReadOnlyAccess

You can attach the `AmazonElastiCacheReadOnlyAccess` policy to your IAM identities.

This policy allows principals read-only access to ElastiCache using the AWS Management Console:

- `elasticache` — Access read-only `Describe` APIs.

You can find the AmazonElastiCacheReadOnlyAccess policy on the IAM console and AmazonElastiCacheReadOnlyAccess in the *AWS Managed Policy Reference Guide*.

ElastiCache updates to AWS managed policies

View details about updates to AWS managed policies for ElastiCache since this service began tracking these changes. For automatic alerts about changes to this page, subscribe to the RSS feed on the ElastiCache Document history page.

Change	Description	Date
AmazonElastiCacheFullAccess – Update to an existing policy	ElastiCache added new permissions to allow management of serverless caches, and to enable usage of all service features via the console.	November 27, 2023
ElastiCacheServiceRolePolicy – Update to an existing policy	ElastiCache added new permissions to allow management of VPC endpoints for serverless cache resources.	November 27, 2023
ElastiCache started tracking changes	ElastiCache started tracking changes for its AWS managed policies.	February 07, 2020

Using identity-based policies (IAM policies) for Amazon ElastiCache

This topic provides examples of identity-based policies in which an account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles).

⚠ **Important**

We recommend that you first read the topics that explain the basic concepts and options to manage access to Amazon ElastiCache resources. For more information, see Overview of managing access permissions to your ElastiCache resources.

The sections in this topic cover the following:

- AWS managed policies for Amazon ElastiCache
- Customer-managed policy examples

The following shows an example of a permissions policy.

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

The policy has two statements:

- The first statement grants permissions for the Amazon ElastiCache actions (`elasticache:Create*`, `elasticache:Describe*`, `elasticache:Modify*`)
- The second statement grants permissions for the IAM action (`iam:PassRole`) on the IAM role name specified at the end of the `Resource` value.

The policy doesn't specify the `Principal` element because in an identity-based policy you don't specify the principal who gets the permission. When you attach policy to a user, the user is the implicit principal. When you attach a permissions policy to an IAM role, the principal identified in the role's trust policy gets the permissions.

For a table showing all of the Amazon ElastiCache API actions and the resources that they apply to, see ElastiCache API permissions: Actions, resources, and conditions reference.

Customer-managed policy examples

If you are not using a default policy and choose to use a custom-managed policy, ensure one of two things. Either you should have permissions to call `iam:createServiceLinkedRole` (for more information, see Example 4: Allow a user to call IAM CreateServiceLinkedRole API). Or you should have created an ElastiCache service-linked role.

When combined with the minimum permissions needed to use the Amazon ElastiCache console, the example policies in this section grant additional permissions. The examples are also relevant to the AWS SDKs and the AWS CLI.

For instructions on setting up IAM users and groups, see Creating Your First IAM User and Administrators Group in the *IAM User Guide*.

> ⚠ **Important**
>
> Always test your IAM policies thoroughly before using them in production. Some ElastiCache actions that appear simple can require other actions to support them when you are using the ElastiCache console. For example, `elasticache:CreateCacheCluster` grants permissions to create ElastiCache cache clusters. However, to perform this operation, the ElastiCache console uses a number of `Describe` and `List` actions to populate console lists.

Examples

- Example 1: Allow a user read-only access to ElastiCache resources
- Example 2: Allow a user to perform common ElastiCache system administrator tasks
- Example 3: Allow a user to access all ElastiCache API actions
- Example 4: Allow a user to call IAM CreateServiceLinkedRole API

Example 1: Allow a user read-only access to ElastiCache resources

The following policy grants permissions ElastiCache actions that allow a user to list resources. Typically, you attach this type of permissions policy to a managers group.

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

Example 2: Allow a user to perform common ElastiCache system administrator tasks

Common system administrator tasks include modifying resources. A system administrator may also want to get information about the ElastiCache events. The following policy grants a user permissions to perform ElastiCache actions for these common system administrator tasks. Typically, you attach this type of permissions policy to the system administrators group.

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Sid": "ECAllowMutations",
      "Effect":"Allow",
      "Action":[
          "elasticache:Modify*",
          "elasticache:Describe*",
          "elasticache:ResetCacheParameterGroup"
      ],
      "Resource":"*"
      }
   ]
}
```

Example 3: Allow a user to access all ElastiCache API actions

The following policy allows a user to access all ElastiCache actions. We recommend that you grant this type of permissions policy only to an administrator user.

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

Example 4: Allow a user to call IAM CreateServiceLinkedRole API

The following policy allows user to call the IAM `CreateServiceLinkedRole` API. We recommend that you grant this type of permissions policy to the user who invokes mutative ElastiCache operations.

```
{
  "Version":"2012-10-17",
  "Statement":[
    {
      "Sid":"CreateSLRAllows",
      "Effect":"Allow",
      "Action":[
        "iam:CreateServiceLinkedRole"
      ],
      "Resource":"*",
      "Condition":{
        "StringLike":{
          "iam:AWSServiceName":"elasticache.amazonaws.com"
        }
      }
    }
  ]
}
```

Resource-level permissions

You can restrict the scope of permissions by specifying resources in an IAM policy. Many ElastiCache API actions support a resource type that varies depending on the behavior of the action. Every IAM policy statement grants permission to an action that's performed on a resource. When the action doesn't act on a named resource, or when you grant permission to perform the action on all resources, the value of the resource in the policy is a wildcard (*). For many API actions, you can restrict the resources that a user can modify by specifying the Amazon Resource Name (ARN) of a resource, or an ARN pattern that matches multiple resources. To restrict permissions by resource, specify the resource by ARN.

To see a list of ElastiCache resource types and their ARNs, see Resources Defined by Amazon ElastiCache in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see Actions Defined by Amazon ElastiCache.

Examples

- Example 1: Allow a user full access to specific ElastiCache resource types
- Example 2: Deny a user access to a serverless cache.

Example 1: Allow a user full access to specific ElastiCache resource types

The following policy explicitly allows all resources of type serverless cache.

```
{
        "Sid": "Example1",
        "Effect": "Allow",
        "Action": "elasticache:*",
        "Resource": [
             "arn:aws:elasticache:us-east-1:account-id:serverlesscache:*"
        ]
}
```

Example 2: Deny a user access to a serverless cache.

The following example explicitly denies access to a particular serverless cache.

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

Using condition keys

You can specify conditions that determine how an IAM policy takes effect. In ElastiCache, you can use the `Condition` element of a JSON policy to compare keys in the request context with key values that you specify in your policy. For more information, see IAM JSON policy elements: Condition.

To see a list of ElastiCache condition keys, see Condition Keys for Amazon ElastiCache in the *Service Authorization Reference*.

For a list of global condition keys, see AWS global condition context keys.

Specifying Conditions: Using Condition Keys

To implement fine-grained control, you write an IAM permissions policy that specifies conditions to control a set of individual parameters on certain requests. You then apply the policy to IAM users, groups, or roles that you create using the IAM console.

To apply a condition, you add the condition information to the IAM policy statement. In the following example, you specify the condition that any self-designed cache cluster created will be of node type `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"
                    ]
                }
            }
        }
    ]
}
```

For more information, see Tag-Based access control policy examples.

For more information on using policy condition operators, see ElastiCache API permissions: Actions, resources, and conditions reference.

Example Policies: Using Conditions for Fine-Grained Parameter Control

This section shows example policies for implementing fine-grained access control on the previously listed ElastiCache parameters.

1. **elasticache:MaximumDataStorage**: Specify the maximum data storage of a serverless cache. Using the provided conditions, the customer can not create caches that can store more than a specific amount of data.

   ```
   {
       "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**: Specify the maximum ECPU per second value of a serverless cache. Using the provided conditions, the customer can not create caches that can execute more than a specific number of ECPUs per second.

```
{
    "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**: Specify which NodeType(s) a user can create. Using the provided conditions, the customer can specify a single or a range value for a node type.

```
{
    "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**: Specify usage of engine version 1.6.6

```
{
    "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**: Specify usage of customer managed AWS KMS keys.

```
{
  "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**: Specify a non default parameter group with specific parameters from an organization on your clusters. You could also specify a naming pattern for your parameter groups or block delete on a specific parameter group name. Following is an example constraining usage of only "my-org-param-group".

```
{
    "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**: Denying `CreateCacheCluster` action if the request tag `Project` is missing or is not equal to `Dev`, `QA` or `Prod`.

```
{
    "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**: Allowing `CreateCacheCluster` with `cacheNodeType` cache.r5.large or cache.r6g.4xlarge and tag `Project=XYZ`.

```
{
  "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

When creating polices to enforce tags and other condition keys together, the conditional `IfExists` may be required on condition key elements due to the extra `elasticache:AddTagsToResource` policy requirements for creation requests with the `--tags` parameter.

Using Service-Linked Roles for Amazon ElastiCache

Amazon ElastiCache uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to an AWS service, such as Amazon ElastiCache. Amazon ElastiCache service-linked roles are predefined by Amazon ElastiCache. They include all the permissions that the service requires to call AWS services on behalf of your clusters.

A service-linked role makes setting up Amazon ElastiCache easier because you don’t have to manually add the necessary permissions. The roles already exist within your AWS account but are linked to Amazon ElastiCache use cases and have predefined permissions. Only Amazon ElastiCache can assume these roles, and only these roles can use the predefined permissions policy. You can delete the roles only after first deleting their related resources. This protects your Amazon ElastiCache resources because you can't inadvertently remove necessary permissions to access the resources.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have **Yes** in the **Service-Linked Role** column. Choose a **Yes** with a link to view the service-linked role documentation for that service.

Contents

- Deleting a Service-Linked Role (IAM API)

Service-Linked Role Permissions for Amazon ElastiCache

Permissions to create service-linked role

To allow an IAM entity to create AWSServiceRoleForElastiCache service-linked role

Add the following policy statement to the permissions for that IAM entity:

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

To allow an IAM entity to delete AWSServiceRoleForElastiCache service-linked role

Add the following policy statement to the permissions for that IAM entity:

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

Alternatively, you can use an AWS managed policy to provide full access to Amazon ElastiCache.

Creating a Service-Linked Role (IAM)

You can create a service-linked role using the IAM console, CLI, or API.

Creating a Service-Linked Role (IAM Console)

You can use the IAM console to create a service-linked role.

To create a service-linked role (console)

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane of the IAM console, choose **Roles**. Then choose **Create new role**.
3. Under **Select type of trusted entity** choose **AWS Service**.
4. Under **Or select a service to view its use cases**, choose **ElastiCache**.
5. Choose **Next: Permissions**.
6. Under **Policy name**, note that the `ElastiCacheServiceRolePolicy` is required for this role. Choose **Next:Tags**.
7. Note that tags are not supported for Service-Linked roles. Choose **Next:Review**.
8. (Optional) For **Role description**, edit the description for the new service-linked role.
9. Review the role and then choose **Create role**.

Creating a Service-Linked Role (IAM CLI)

You can use IAM operations from the AWS Command Line Interface to create a service-linked role. This role can include the trust policy and inline policies that the service needs to assume the role.

To create a service-linked role (CLI)

Use the following operation:

```
$ aws iam create-service-linked-role --aws-service-name elasticache.amazonaws.com
```

Creating a Service-Linked Role (IAM API)

You can use the IAM API to create a service-linked role. This role can contain the trust policy and inline policies that the service needs to assume the role.

To create a service-linked role (API)

Use the CreateServiceLinkedRole API call. In the request, specify a service name of `elasticache.amazonaws.com`.

Editing the Description of a Service-Linked Role for Amazon ElastiCache

Amazon ElastiCache does not allow you to edit the AWSServiceRoleForElastiCache service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM.

Editing a Service-Linked Role Description (IAM Console)

You can use the IAM console to edit a service-linked role description.

To edit the description of a service-linked role (console)

1. In the navigation pane of the IAM console, choose **Roles**.
2. Choose the name of the role to modify.
3. To the far right of **Role description**, choose **Edit**.
4. Enter a new description in the box and choose **Save**.

Editing a Service-Linked Role Description (IAM CLI)

You can use IAM operations from the AWS Command Line Interface to edit a service-linked role description.

To change the description of a service-linked role (CLI)

1. (Optional) To view the current description for a role, use the AWS CLI for IAM operation `get-role`.

   Example

   ```
   $ aws iam get-role --role-name AWSServiceRoleForElastiCache
   ```

   Use the role name, not the ARN, to refer to roles with the CLI operations. For example, if a role has the following ARN: `arn:aws:iam::123456789012:role/myrole`, refer to the role as **`myrole`**.

2. To update a service-linked role's description, use the AWS CLI for IAM operation `update-role-description`.

   For Linux, macOS, or Unix:

```
$ aws iam update-role-description \
    --role-name AWSServiceRoleForElastiCache \
    --description "new description"
```

For Windows:

```
$ aws iam update-role-description ^
    --role-name AWSServiceRoleForElastiCache ^
    --description "new description"
```

Editing a Service-Linked Role Description (IAM API)

You can use the IAM API to edit a service-linked role description.

To change the description of a service-linked role (API)

1. (Optional) To view the current description for a role, use the IAM API operation GetRole.

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=GetRole
      &RoleName=AWSServiceRoleForElastiCache
      &Version=2010-05-08
      &AUTHPARAMS
   ```

2. To update a role's description, use the IAM API operation UpdateRoleDescription.

   Example

   ```
   https://iam.amazonaws.com/
      ?Action=UpdateRoleDescription
      &RoleName=AWSServiceRoleForElastiCache
      &Version=2010-05-08
      &Description="New description"
   ```

Deleting a Service-Linked Role for Amazon ElastiCache

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up your service-linked role before you can delete it.

Amazon ElastiCache does not delete the service-linked role for you.

Cleaning Up a Service-Linked Role

Before you can use IAM to delete a service-linked role, first confirm that the role has no resources—clusters—associated with it.

To check whether the service-linked role has an active session in the IAM console

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane of the IAM console, choose **Roles**. Then choose the name (not the check box) of the AWSServiceRoleForElastiCache role.
3. On the **Summary** page for the selected role, choose the **Access Advisor** tab.
4. On the **Access Advisor** tab, review recent activity for the service-linked role.

To delete Amazon ElastiCache resources that require AWSServiceRoleForElastiCache

- To delete a cluster, see the following:
  - Using the AWS Management Console
  - Using the AWS CLI
  - Using the ElastiCache API

Deleting a Service-Linked Role (IAM Console)

You can use the IAM console to delete a service-linked role.

To delete a service-linked role (console)

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.

2. In the navigation pane of the IAM console, choose **Roles**. Then select the check box next to the role name that you want to delete, not the name or row itself.
3. For **Role actions** at the top of the page, choose **Delete role**.
4. In the confirmation dialog box, review the service last accessed data, which shows when each of the selected roles last accessed an AWS service. This helps you to confirm whether the role is currently active. If you want to proceed, choose **Yes, Delete** to submit the service-linked role for deletion.
5. Watch the IAM console notifications to monitor the progress of the service-linked role deletion. Because the IAM service-linked role deletion is asynchronous, after you submit the role for deletion, the deletion task can succeed or fail. If the task fails, you can choose **View details** or **View Resources** from the notifications to learn why the deletion failed.

Deleting a Service-Linked Role (IAM CLI)

You can use IAM operations from the AWS Command Line Interface to delete a service-linked role.

To delete a service-linked role (CLI)

1. If you don't know the name of the service-linked role that you want to delete, enter the following command. This command lists the roles and their Amazon Resource Names (ARNs) in your account.

   ```
   $ aws iam get-role --role-name role-name
   ```

   Use the role name, not the ARN, to refer to roles with the CLI operations. For example, if a role has the ARN `arn:aws:iam::123456789012:role/myrole`, you refer to the role as **`myrole`**.
2. Because a service-linked role cannot be deleted if it is being used or has associated resources, you must submit a deletion request. That request can be denied if these conditions are not met. You must capture the `deletion-task-id` from the response to check the status of the deletion task. Enter the following to submit a service-linked role deletion request.

   ```
   $ aws iam delete-service-linked-role --role-name role-name
   ```
3. Enter the following to check the status of the deletion task.

   ```
   $ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-
   id
   ```

The status of the deletion task can be NOT_STARTED, IN_PROGRESS, SUCCEEDED, or FAILED. If the deletion fails, the call returns the reason that it failed so that you can troubleshoot.

Deleting a Service-Linked Role (IAM API)

You can use the IAM API to delete a service-linked role.

To delete a service-linked role (API)

1. To submit a deletion request for a service-linked roll, call DeleteServiceLinkedRole. In the request, specify a role name.

   Because a service-linked role cannot be deleted if it is being used or has associated resources, you must submit a deletion request. That request can be denied if these conditions are not met. You must capture the DeletionTaskId from the response to check the status of the deletion task.
2. To check the status of the deletion, call GetServiceLinkedRoleDeletionStatus. In the request, specify the DeletionTaskId.

   The status of the deletion task can be NOT_STARTED, IN_PROGRESS, SUCCEEDED, or FAILED. If the deletion fails, the call returns the reason that it failed so that you can troubleshoot.

ElastiCache API permissions: Actions, resources, and conditions reference

When you set up access control and write permissions policies to attach to an IAM policy (either idenity-based or resource-based), use the following table as a reference. The table lists each Amazon ElastiCache API operation and the corresponding actions for which you can grant permissions to perform the action. You specify the actions in the policy's `Action` field, and you specify a resource value in the policy's `Resource` field. Unless indicated otherwise, the resource is required. Some fields include both a required resource and optional resources. When there is no resource ARN, the resource in the policy is a wildcard (*).

You can use condition keys in your ElastiCache policies to express conditions. To see a list of ElastiCache-specific condition keys, along with the actions and resource types to which they apply, see Using condition keys. For a complete list of AWS-wide keys, see AWS global condition context keys in the *IAM User Guide*.

> **Note**
>
> To specify an action, use the `elasticache:` prefix followed by the API operation name (for example, `elasticache:DescribeCacheClusters`).

To see a list of ElastiCache actions, see Actions Defined by Amazon ElastiCache in the *Service Authorization Reference*.

Compliance validation for Amazon ElastiCache

Third-party auditors assess the security and compliance of AWS services as part of multiple AWS compliance programs, such as SOC, PCI, FedRAMP, and HIPAA.

To learn whether an AWS service is within the scope of specific compliance programs, see AWS services in Scope by Compliance Program and choose the compliance program that you are interested in. For general information, see AWS Compliance Programs.

You can download third-party audit reports using AWS Artifact. For more information, see Downloading Reports in AWS Artifact.

Your compliance responsibility when using AWS services is determined by the sensitivity of your data, your company's compliance objectives, and applicable laws and regulations. AWS provides the following resources to help with compliance:

- Security and Compliance Quick Start Guides – These deployment guides discuss architectural considerations and provide steps for deploying baseline environments on AWS that are security and compliance focused.
- Architecting for HIPAA Security and Compliance on Amazon Web Services – This whitepaper describes how companies can use AWS to create HIPAA-eligible applications.

> **Note**
>
> Not all AWS services are HIPAA eligible. For more information, see the HIPAA Eligible Services Reference.

- AWS Compliance Resources – This collection of workbooks and guides might apply to your industry and location.
- AWS Customer Compliance Guides – Understand the shared responsibility model through the lens of compliance. The guides summarize the best practices for securing AWS services and map the guidance to security controls across multiple frameworks (including National Institute of Standards and Technology (NIST), Payment Card Industry Security Standards Council (PCI), and International Organization for Standardization (ISO)).
- Evaluating Resources with Rules in the *AWS Config Developer Guide* – The AWS Config service assesses how well your resource configurations comply with internal practices, industry guidelines, and regulations.
- AWS Security Hub – This AWS service provides a comprehensive view of your security state within AWS. Security Hub uses security controls to evaluate your AWS resources and to check your compliance against security industry standards and best practices. For a list of supported services and controls, see Security Hub controls reference.
- AWS Audit Manager – This AWS service helps you continuously audit your AWS usage to simplify how you manage risk and compliance with regulations and industry standards.

More information

For general information about AWS Cloud compliance, see the following:

- FIPS Endpoints by Service
- Service updates in ElastiCache
- AWS Cloud Compliance

- Shared Responsibility Model
- AWS PCI DSS Compliance Program

Resilience in Amazon ElastiCache

The AWS global infrastructure is built around AWS Regions and Availability Zones. AWS Regions provide multiple physically separated and isolated Availability Zones, which are connected with low-latency, high-throughput, and highly redundant networking. With Availability Zones, you can design and operate applications and databases that automatically fail over between Availability Zones without interruption. Availability Zones are more highly available, fault tolerant, and scalable than traditional single or multiple data center infrastructures.

For more information about AWS Regions and Availability Zones, see AWS Global Infrastructure.

In addition to the AWS global infrastructure, Amazon ElastiCache offers several features to help support your data resiliency and backup needs.

Topics

- Mitigating Failures

Mitigating Failures

When planning your Amazon ElastiCache implementation, you should plan so that failures have a minimal impact upon your application and data. The topics in this section cover approaches you can take to protect your application and data from failures.

Topics

- Mitigating Failures when Running Memcached
- Recommendations

Mitigating Failures when Running Memcached

When running the Memcached engine, you have the following options for minimizing the impact of a failure. There are two types of failures to address in your failure mitigation plans: node failure and Availability Zone failure.

Mitigating Node Failures

Serverless caches automatically mitigate node failures with a replicated Multi-AZ architecture so that node failures are transparent to your application. To mitigate the impact of a node failure in a self-designed cluster, spread your cached data over more nodes. Because self-designed clusters do not support replication, a node failure will always result in some data loss from your cluster.

When you create your Memcached cluster you can create it with 1 to 60 nodes, or more by special request. Partitioning your data across a greater number of nodes means you'll lose less data if a node fails. For example, if you partition your data across 10 nodes, any single node stores approximately 10% of your cached data. In this case, a node failure loses approximately 10% of your cache which needs to be replaced when a replacement node is created and provisioned. If the same data were cached in 3 larger nodes, the failure of a node would lose approximately 33% of your cached data.

If you need more than 60 nodes in a Memcached cluster, or more than 300 nodes total in an AWS Region, fill out the ElastiCache Limit Increase Request form at https://aws.amazon.com/contact-us/elasticache-node-limit-request/.

For information on specifying the number of nodes in a Memcached cluster, see Creating a Memcached cluster (console).

Mitigating Availability Zone Failures

Serverless caches automatically mitigate availability zone failures with a replicated Multi-AZ architecture so that AZ failures are transparent to your application.

To mitigate the impact of an Availability Zone failure in a self-designed cluster, locate your nodes in as many Availability Zones as possible. In the unlikely event of an AZ failure, you will lose the data cached in that AZ, not the data cached in the other AZs.

Why so many nodes?

If my region has only 3 Availability Zones, why do I need more than 3 nodes since if an AZ fails I lose approximately one-third of my data?

This is an excellent question. Remember that we’re attempting to mitigate two distinct types of failures, node and Availability Zone. You’re right, if your data is spread across Availability Zones and one of the zones fails, you will lose only the data cached in that AZ, irrespective of the number of nodes you have. However, if a node fails, having more nodes will reduce the proportion of data lost.

There is no "magic formula" for determining how many nodes to have in your cluster. You must weight the impact of data loss vs. the likelihood of a failure vs. cost, and come to your own conclusion.

For information on specifying the number of nodes in a Memcached cluster, see Creating a Memcached cluster (console).

For more information on regions and Availability Zones, see Regions and Availability Zones.

Recommendations

We recommend creating serverless caches over self-designed clusters, as you automatically obtain better fault tolerance without additional configuration. When creating a self-designed cluster, however, there are two types of failures you need to plan for: individual node failures and broad Availability Zone failures. The best failure mitigation plan will address both kinds of failures.

Minimizing the Impact of Node Failures

When running Memcached and partitioning your data across nodes, the more nodes you use the smaller the data loss if any one node fails.

Minimizing the Impact of Availability Zone Failures

To minimize the impact of an Availability Zone failure, we recommend launching your nodes in as many different Availability Zones as are available. Spreading your nodes evenly across AZs will minimize the impact in the unlikely event of an AZ failure. This is done automatically for serverless caches.

Infrastructure security in AWS ElastiCache

As a managed service, AWS ElastiCache is protected by the AWS global network security procedures that are described in the Security and Compliance section at AWS Architecture Center.

You use AWS published API calls to access ElastiCache through the network. Clients must support Transport Layer Security (TLS) 1.2 or later. We recommend TLS 1.3 or later. Clients must also support cipher suites with perfect forward secrecy (PFS) such as Ephemeral Diffie-Hellman (DHE) or Elliptic Curve Ephemeral Diffie-Hellman (ECDHE). Most modern systems such as Java 7 and later support these modes.

Additionally, requests must be signed by using an access key ID and a secret access key that is associated with an IAM principal. Or you can use the AWS Security Token Service (AWS STS) to generate temporary security credentials to sign requests.

Service updates in ElastiCache

ElastiCache automatically monitors your fleet of caches, clusters, and nodes to apply service updates as they become available. Service updates for serverless caches are automatically and transparently applied. For self-designed clusters, you set up a predefined maintenance window so that ElastiCache can apply these updates. However, in some cases you might find this approach too rigid and likely to constrain your business flows.

With service updates, you control when and which updates are applied to your self-designed clusters. You can also monitor the progress of these updates to your selected ElastiCache cluster in real time.

Managing service updates

ElastiCache service updates for self-designed clusters are released on a regular basis. If you have one or more qualifying self-designed clusters for those service updates, you receive notifications through email, SNS, the Personal Health Dashboard (PHD), and Amazon CloudWatch events when the updates are released. The updates are also displayed on the **Service Updates** page on the ElastiCache console. By using this dashboard, you can view all the service updates and their status for your ElastiCache fleet. Service updates for serverless caches are transparently applied and cannot be managed via **Service Updates**.

You control when to apply an update before an auto-update starts. We strongly recommend that you apply any updates of type **security-update** as soon as possible to ensure that your ElastiCache clusters are always up-to-date with current security patches.

The following sections explore these options in detail.

Topics

Applying the service updates

You can start applying the service updates to your fleet from the time that the updates have an **available** status. Service updates are cumulative. In other words, any updates that you haven't applied yet are included with your latest update.

If a service update has auto-update enabled, you can choose to not take any action when it becomes available. ElastiCache will schedule to apply the update during one of your clusters' upcoming maintenance windows after the **Auto-update start date**. You will receive related notifications for each stage of the update.

> **Note**
>
> You can apply only those service updates that have an **available** or **scheduled** status.

For more information about reviewing and applying any service-specific updates to applicable ElastiCache clusters, see Applying the service updates using the console.

When a new service update is available for one or more of your ElastiCache clusters, you can use the ElastiCache console, API, or AWS CLI to apply the update. The following sections explain the options that you can use to apply updates.

Applying the service updates using the console

To view the list of available service updates, along with other information, go to the **Service Updates** page in the console.

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. On the navigation pane, choose **Service Updates**.
3. Under **Service updates** you can view the following:
    - **Service update name**: The unique name of the service update
    - **Update type**: The type of the service update, which is one of **security-update** or **engine-update**
    - **Update severity**: The priority of applying the update:
        - **critical:** We recommend that you apply this update immediately (within 14 days or less).

  - **important:** We recommend that you apply this update as soon as your business flow allows (within 30 days or less).
  - **medium:** We recommend that you apply this update as soon as you can (within 60 days or less).
  - **low:** We recommend that you apply this update as soon as you can (within 90 days or less).
- **Engine version**: If the update type is engine-update, the engine version that is being updated.
- **Release Date**: When the update is released and available to apply on your clusters.
- **Recommended Apply By Date**: ElastiCache guidance date to apply the updates by.
- **Status**: The status of the update, which is one of the following:
  - **available:** The update is available for requisite clusters.
  - **complete:** The update has been applied.
  - **cancelled:** The update has been canceled and is no longer necessary.
  - **expired:** The update is no longer available to apply.

4. Choose an individual update (not the button to its left) to view details of the service update.

   In the **Cluster update status** section, you can view a list of clusters where the service update has not been applied or has just been applied recently. For each cluster, you can view the following:

   - **Cluster name**: The name of the cluster
   - **Nodes updated:** The ratio of individual nodes within a specific cluster that were updated or remain available for the specific service update.
   - **Update Type**: The type of the service update, which is one of **security-update** or **engine-update**
   - **Status**: The status of the service update on the cluster, which is one of the following:
     - *available*: The update is available for the requisite cluster.
     - *in-progress*: The update is being applied to this cluster.
     - *scheduled*: The update date has been scheduled.
     - *complete*: The update has been successfully applied. Cluster with a complete status will be displayed for 7 days after its completion.

If you chose any or all of the clusters with the **available** or **scheduled** status, and then chose **Apply now**, the update will start being applied on those clusters.

Applying the service updates using the AWS CLI

After you receive notification that service updates are available, you can inspect and apply them using the AWS CLI:

- To retrieve a description of the service updates that are available, run the following command:

  ```
  aws elasticache describe-service-updates --service-update-status
  available
  ```

  For more information, see describe-service-updates.
- To apply a service update on a list of clusters, run the following command:

  ```
  aws elasticache batch-apply-update-action --service-update
  ServiceUpdateNameToApply=sample-service-update --cluster-names cluster-1
  cluster2
  ```

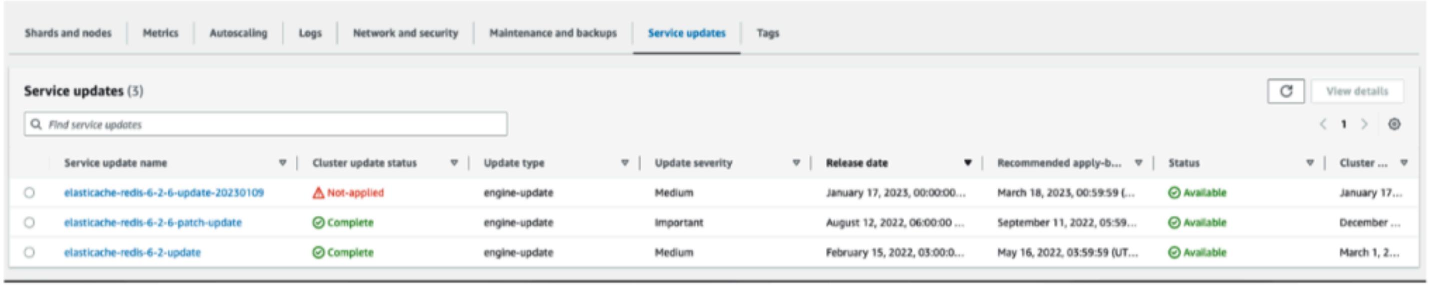
  For more information, see batch-apply-update-action.

Verifying you have the latest Service Update Applied using the AWS console

You can verify your ElastiCache for Redis clusters are running the latest service update by following these steps:

1. Choose an applicable cluster on the **Redis Clusters** page
2. Choose **Service updates** in the navigation pane to see the applicable service updates for that cluster, if any.

If the console displays a list of service updates, you can select the service update and choose **Apply now**.

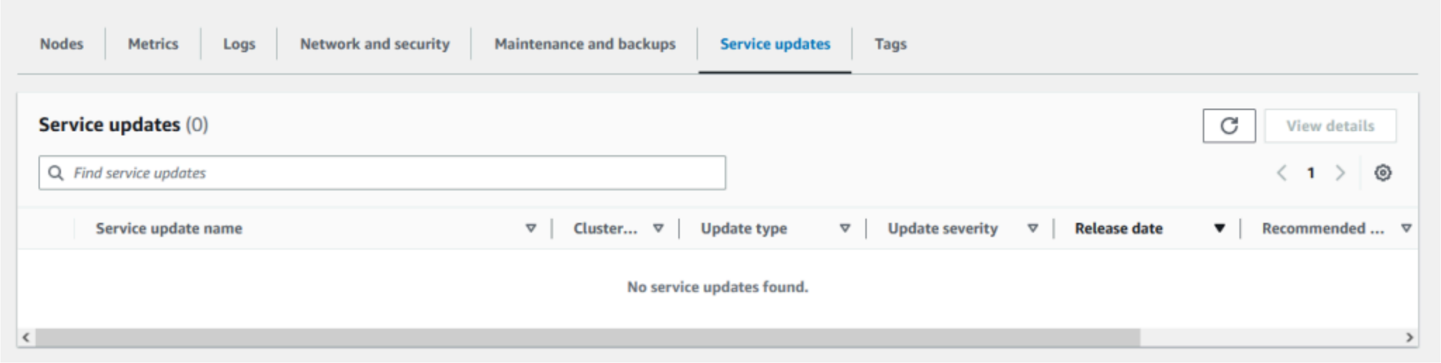


Service updates (3)

Find service updates

	Service update name	Cluster update status	Update type	Update severity	Release date	Recommended apply-b...	Status	Cluster ...
○	elasticache-redis-6-2-6-update-20230109	Not-applied	engine-update	Medium	January 17, 2023, 00:00:00...	March 18, 2023, 00:59:59 (...	Available	January 17...
○	elasticache-redis-6-2-6-patch-update	Complete	engine-update	Important	August 12, 2022, 06:00:00 ...	September 11, 2022, 05:59...	Available	December ...
○	elasticache-redis-6-2-update	Complete	engine-update	Medium	February 15, 2022, 03:00:0...	May 16, 2022, 03:59:59 (UT...	Available	March 1, 2...

If the console displays “No service updates found”, it means the ElastiCache for Redis cluster already has the latest service update applied.



Service updates (0)

Find service updates

Service update name	Cluster...	Update type	Update severity	Release date	Recommended ...

No service updates found.

Stopping the service updates

You can stop updates to clusters if needed. For example, you might want to stop updates if you have an unexpected surge to your clusters that are undergoing updates. Or you might want to stop updates if they're taking too long and interrupting your business flow at a peak time.

The Stopping operation immediately interrupts all updates to those clusters and any nodes that are yet to be updated. It continues to completion any nodes that have an **in progress** status. However, it ceases updates to other nodes in the same cluster that have an **update available** status and reverts them to a **Stopping** status.

When the **Stopping** workflow is complete, the nodes that have a **Stopping** status change to a **Stopped** status. Depending on the workflow of the update, some clusters won't have any nodes updated. Other clusters might include some nodes that are updated and others that still have an **update available** status.

You can return later to finish the update process as your business flows permit. In this case, choose the applicable clusters that you want to complete updates on, and then choose **Apply Now**. For more information, see Applying the service updates.

Using the console

You can interrupt a service update using the ElastiCache console. The following demonstrates how to do this:

- After a service update has progressed on a selected cluster, the ElastiCache console displays the **View/Stop Update** tab at the top of the ElastiCache dashboard.
- To interrupt the update, choose **Stop Update**.
- When you stop the update, choose the cluster and examine the status. It reverts to a **Stopping** status and eventually a **Stopped** status.

Using the AWS CLI

You can interrupt a service update using the AWS CLI. The following code example shows how to do this.

For a replication group, do the following:

```
aws elasticache batch-stop-update-action --service-update-name sample-service-update --replication-group-ids my-replication-group-1 my-replication-group-2
```

For a cache cluster, do the following:

```
aws elasticache batch-stop-update-action --service-update-name sample-service-update --cache-cluster-ids my-cache-cluster-1 my-cache-cluster-2
```

For more information, see BatchStopUpdateAction.

Logging and monitoring in Amazon ElastiCache

To manage your cache, it's important that you know how your caches are performing. ElastiCache generates metrics that are published to Amazon CloudWatch Logs for monitoring your cache performance. In addition, ElastiCache generates events when significant changes happen on your cache resources (e.g. a new cache is created, or a cache is deleted).

Topics

Serverless metrics and events

This section describes the metrics and events that you can monitor when working with serverless caches.

Topics

Serverless cache metrics

The `AWS/ElastiCache` namespace includes the following CloudWatch metrics for your Memcached serverless caches.

Metric	Description	Unit
BytesUsedForCache	The total number of bytes used by the data stored in your cache.	Bytes

Metric	Description	Unit
ElastiCacheProcessingUnits	The total number of ElastiCacheProcessingUnits (ECPUs) consumed by the requests executed on your cache	Count
SuccessfulReadRequestLatency	Latency of successful read requests.	Microseconds
SuccessfulWriteRequestLatency	Latency of successful write requests	Microseconds
TotalCmdsCount	Total count of all commands executed on your cache	Count
CurrConnections	The number of client connections to your cache.	Count
ThrottledCmds	The number of requests that were throttled by ElastiCache because the workload was scaling faster than ElastiCache can scale.	Count
NewConnections	The total number of connections that have been accepted by the server during this period.	Count
CurrItems	The number of items in the cache.	Count
NetworkBytesIn	Total bytes transferred in to cache	Bytes
NetworkBytesOut	Total bytes transferred out of cache	Bytes

Metric	Description	Unit
Evictions	The count of keys evicted by the cache	Count
Reclaimed	The number of keys expired by the cache	Count

Command level metrics

ElastiCache also emits the following Memcached command level metrics

Metric	Description	Unit
CmdGet	The number of get commands the cache has received.	Count
CmdSet	The number of set commands the cache has received.	Count
CmdTouch	The number of touch commands the cache has received.	Count
GetHits	The number of get requests the cache has received where the key requested was found.	Count
GetMisses	The number of get requests the cache has received where the key requested was not found.	Count
IncrHits	The number of increment requests the cache has received where the key requested was found.	Count

Metric	Description	Unit
IncrMisses	The number of increment requests the cache has received where the key requested was not found.	Count
DecrHits	The number of decrement requests the cache has received where the requested key was found.	Count
DecrMisses	The number of decrement requests the cache has received where the requested key was not found.	Count
DeleteHits	The number of delete requests the cache has received where the requested key was found.	Count
DeleteMisses	The number of delete requests the cache has received where the requested key was not found.	Count
TouchHits	The number of keys that have been touched and were given a new expiration time.	Count
TouchMisses	The number of keys that have been touched, but were not found.	Count

Metric	Description	Unit
CasHits	The number of cas requests the cache has received where the requested key was found and the cas value matched.	Count
CasMisses	The number of cas requests the cache has received where the key requested was not found.	Count
CasBadval	The number of cas requests the cache has received where the cas value did not match the cas value stored.	Count
CmdFlush	The number of flush commands the cache has received.	Count

Serverless cache events

ElastiCache logs events that relate to your serverless cache. This information includes the date and time of the event, the source name and source type of the event, and a description of the event. You can easily retrieve events from the log using the ElastiCache console, the AWS CLI describe-events command, or the ElastiCache API action `DescribeEvents`.

You can choose to monitor, ingest, transform, and act on ElastiCache events using Amazon EventBridge. Learn more in the Amazon EventBridge https://docs.aws.amazon.com/eventbridge/latest/userguide/.

Viewing ElastiCache events (Console)

To view events using the ElastiCache console:

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/

2. To see a list of all available events, in the navigation pane, choose **Events**.
3. On the *Events* screen each row of the list represents one event and displays the event source, the event type, the GMT time of the event, and a description of the event. Using the **Filter** you can specify whether you want to see all events, or just events of a specific type in the event list.

Viewing ElastiCache events (AWS CLI)

To generate a list of ElastiCache events using the AWS CLI, use the command describe-events. You can use optional parameters to control the type of events listed, the time frame of the events listed, the maximum number of events to list, and more.

The following code lists up to 40 serverless cache events.

```
aws elasticache describe-events --source-type serverless-cache --max-items 40
```

The following code lists all events for serverless caches for the past 24 hours (1440 minutes).

```
aws elasticache describe-events --source-type serverless-cache --duration 1440
```

Serverless Events

This section documents the different types of events that you may receive for your serverless caches.

Serverless Cache Creation Events

Detail-Type	Description	Unit	Source	Message
Cache created	Cache arn	creation	serverless-cache	Cache <cache-name> is created and ready to use.
Cache creation failed	Cache arn	failure	serverless-cache	Failed to create cache <cache-name>. Insufficient free IP addresses

Detail-Type	Description	Unit	Source	Message
				to create VPC endpoint.
Cache creation failed	Cache arn	failure	serverless-cache	Failed to create cache <cache-name>. Invalid subnets provided in the request.
Cache creation failed	Cache arn	failure	serverless-cache	Failed to create cache <cache-name>. Quota limit reached for creating VPC endpoint.
Cache creation failed	Cache arn	failure	serverless-cache	Failed to create cache <cache-name>. You do not have permissions to create a VPC endpoint.

Serverless Cache Update Events

Detail-Type	Resources list	Category	Source	Message
Cache updated	Cache arn	configuration change	serverless-cache	SecurityGroups updated for the cache <cache-name>.

Detail-Type	Resources list	Category	Source	Message
Cache updated	Cache arn	configuration change	serverless-cache	Tags updated for the cache <cache-name>.
Cache updated failed	Cache arn	configuration change	serverless-cache	An update to the cache <cache-name> failed. SecurityGroups update failed.
Cache updated failed	Cache arn	configuration change	serverless-cache	An update to the cache <cache-name> failed. SecurityGroups update failed because of insufficient permissions.
Cache updated failed	Cache arn	configuration change	serverless-cache	An update to the cache <cache-name> failed. SecurityGroups update failed because the SecurityGroups are invalid.

Serverless Cache Deletion Events

Detail-Type	Resources list	Category	Source	Message
Cache deleted	Cache arn	deletion	serverless-cache	Cache <cache-name> was deleted.

Serverless Cache Usage Limit Events

Detail-Type	Description	Unit	Source	Message
Cache updated	Cache arn	configuration change	serverless-cache	Limits updated for the cache <cache-name>.
Cache updated failed	Cache arn	failure	serverless-cache	A limit update to the cache <cache-name> failed because the cache was deleted.
Cache updated failed	Cache arn	failure	serverless-cache	A limit update to the cache <cache-name> failed due to invalid configura tion.

Self-designed clusters metrics and events

This section describes the metrics, events, and logs that you can expect to see when you work with self-designed clusters.

Topics

- Metrics for self-designed clusters

- Events for self-designed clusters
- Monitoring use with CloudWatch metrics
- Amazon SNS monitoring of ElastiCache events

Metrics for self-designed clusters

When you self-design clusters, ElastiCache emits metrics at each node level, including both host-level metrics and cache metrics.

For more information on host-level metrics for Memcached, see Host-Level Metrics.

For more information on node-level Memcached metrics, see Metrics for Memcached.

Events for self-designed clusters

ElastiCache logs events that relate to your self-designed caches. When working with self-designed clusters, you can view your cluster events in the ElastiCache console, using the AWS CLI, or using Amazon Simple Notification Service (SNS). Self-designed cluster events are not published to Amazon EventBridge.

Self-designed cluster event information includes the date and time of the event, the source name and source type of the event, and a description of the event. You can easily retrieve events from the log using the ElastiCache console, the AWS CLI describe-events command, or the ElastiCache API action DescribeEvents.

Viewing ElastiCache events (Console)

The following procedure displays events using the ElastiCache console.

To view events using the ElastiCache console

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/
2. To see a list of all available events, in the navigation pane, choose Events.
3. On the Events screen each row of the list represents one event and displays the event source, the event type, the GMT time of the event, and a description of the event. Using the Filter you can specify whether you want to see all events, or just events of a specific type in the event list.

Viewing ElastiCache events (AWS CLI)

To generate a list of ElastiCache events using the AWS CLI, use the command describe-events. You can use optional parameters to control the type of events listed, the time frame of the events listed, the maximum number of events to list, and more.

The following code lists up to 40 self-designed cluster events.

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
```

The following code lists all events for self-designed caches for the past 24 hours (1440 minutes).

```
aws elasticache describe-events --source-type cache-cluster --duration 1440
```

Self-designed cluster events

This section contains the list of events you can expect to receive for your self-designed clusters.

The following ElastiCache events trigger Amazon SNS notifications. For information on event details, see Viewing ElastiCache events.

Event Name	Message	Description
ElastiCache:AddCacheNodeComplete	`ElastiCache:AddCacheNodeComplete : ` *`cache-cluster`*	A cache node has been added to the cache cluster and is ready for use.
ElastiCache:AddCacheNodeFailed due to insufficient free IP addresses	`ElastiCache:AddCacheNodeFailed : ` *`cluster-name`*	A cache node could not be added because there are not enough available IP addresses .
ElastiCache:CacheClusterParametersChanged	`ElastiCache:CacheClusterParametersChanged : ` *`cluster-name`*	One or more cache cluster parameters have been changed.
ElastiCache:CacheClusterProvisioningComplete	`ElastiCache:CacheClusterProvisioning`	The provisioning of a cache cluster is completed, and the cache nodes in the cache cluster are ready to use.

Event Name	Message	Description
	`Complete` *`cluster-name-0001-005`*	
ElastiCache:CacheClusterProvisioningFailed due to incompatible network state	`ElastiCache:CacheClusterProvisioningFailed :` *`cluster-name`*	An attempt was made to launch a new cache cluster into a nonexistent virtual private cloud (VPC).
ElastiCache:CacheClusterScalingComplete	`CacheClusterScalingComplete :` *`cluster-name`*	Scaling for cache-cluster completed successfully.
ElastiCache:CacheClusterScalingFailed	ElastiCache:CacheClusterScalingFailed : *`cluster-name`*	Scale-up operation on cache-cluster failed.
ElastiCache:CacheClusterSecurityGroupModified	`ElastiCache:CacheClusterSecurityGroupModified :` *`cluster-name`*	One of the following events has occurred: - The list of cache security groups authorized for the cache cluster has been modified. - One or more new EC2 security groups have been authorized on any of the cache security groups associated with the cache cluster. - One or more EC2 security groups have been revoked from any of the cache security groups associated with the cache cluster.

Event Name	Message	Description
ElastiCache:CacheNodeReplaceStarted	`ElastiCache:CacheNodeReplaceStarted : *cluster-name*`	ElastiCache has detected that the host running a cache node is degraded or unreachable and has started replacing the cache node. **Note** The DNS entry for the replaced cache node is not changed. In most instances, you do not need to refresh the server-list for your clients when this event occurs. However, some cache client libraries may stop using the cache node even after ElastiCache has replaced the cache node; in this case, the application should refresh the server-list when this event occurs.

Event Name	Message	Description
ElastiCache:CacheNodeReplaceComplete	`ElastiCache:CacheNodeReplaceComplete :` *`cluster-name`*	ElastiCache has detected that the host running a cache node is degraded or unreachable and has completed replacing the cache node. **Note** The DNS entry for the replaced cache node is not changed. In most instances, you do not need to refresh the server-list for your clients when this event occurs. However, some cache client libraries may stop using the cache node even after ElastiCache has replaced the cache node; in this case, the application should refresh the server-list when this event occurs.
ElastiCache:CacheNodesRebooted	`ElastiCache:CacheNodesRebooted :` *`cluster-name`*	One or more cache nodes has been rebooted. Message (Memcached): `"Cache node %s shutdown"` Then a second message: `"Cache node %s restarted"`

Event Name	Message	Description
ElastiCache:CertificateRenewalComplete (Redis only)	`ElastiCache:CertificateRenewalComplete`	The Amazon CA certificate was successfully renewed.
ElastiCache:CreateReplicationGroupComplete	`ElastiCache:CreateReplicationGroupComplete :` *`cluster-name`*	The replication group was successfully created.
ElastiCache:DeleteCacheClusterComplete	`ElastiCache:DeleteCacheClusterComplete :` *`cluster-name`*	The deletion of a cache cluster and all associated cache nodes has completed.
ElastiCache:FailoverComplete (Redis only)	`ElastiCache:FailoverComplete :` *`mycluster`*	Failover over to a replica node was successful.
ElastiCache:ReplicationGroupIncreaseReplicaCountFinished	`ElastiCache:ReplicationGroupIncreaseReplicaCountFinished :` *`cluster-name-0001-005`*	The number of replicas in the cluster has been increased.
ElastiCache:ReplicationGroupIncreaseReplicaCountStarted	`ElastiCache:ReplicationGroupIncreaseReplicaCountStarted :` *`cluster-name-0003-004`*	The process of adding replicas to your cluster has begun.
ElastiCache:NodeReplacementCanceled	`ElastiCache:NodeReplacementCanceled :` *`cluster-name`*	A node in your cluster that was scheduled for replacement is no longer scheduled for replacement.

Event Name	Message	Description
ElastiCache:NodeReplacementRescheduled	`ElastiCache:NodeReplacementRescheduled : cluster-name`	A node in your cluster previously scheduled for replacement has been rescheduled for replacement during the new window described in the notification. For information on what actions you can take, see Replacing cache nodes for Memcached.
ElastiCache:NodeReplacementScheduled	`ElastiCache:NodeReplacementScheduled : cluster-name`	A node in your cluster is scheduled for replacement during the window described in the notification. For information on what actions you can take, see Replacing cache nodes for Memcached.
ElastiCache:RemoveCacheNodeComplete	`ElastiCache:RemoveCacheNodeComplete : cluster-name`	A cache node has been removed from the cache cluster.
ElastiCache:ReplicationGroupScalingComplete	`ElastiCache:ReplicationGroupScalingComplete : cluster-name`	Scale-up operation on replication group completed successfully.
ElastiCache:ReplicationGroupScalingFailed	`"Failed applying modification to cache node type to %s."`	Scale-up operation on replication group failed.

Event Name	Message	Description
ElastiCache:ServiceUpdateAvailableForNode	`"Service update is available for cache node %s."`	A self-service update is available for the node.
ElastiCache:SnapshotComplete (Redis only)	`ElastiCache:SnapshotComplete :` *`cluster-name`*	A cache snapshot has completed successfully.
ElastiCache:SnapshotFailed (Redis only)	`SnapshotFailed :` *`cluster-name`*	A cache snapshot has failed. See the cluster's cache events for more a detailed cause. If you describe the snapshot, see `DescribeSnapshots` , the status will be `failed`.

Monitoring use with CloudWatch metrics

ElastiCache provides metrics that enable you to monitor your clusters. You can access these metrics through CloudWatch. For more information on CloudWatch, see the CloudWatch documentation.

ElastiCache provides both host-level metrics (for example, CPU usage) and metrics that are specific to the cache engine software (for example, cache gets and cache misses). These metrics are measured and published for each Cache node in 60-second intervals.

Important

You should consider setting CloudWatch alarms on certain key metrics, so that you will be notified if your cache cluster's performance starts to degrade. For more information, see Which metrics should I monitor? in this guide.

Topics

- Host-Level Metrics
- Metrics for Memcached

- Which metrics should I monitor?
- Monitoring CloudWatch Cluster and Node Metrics

Host-Level Metrics

The `AWS/ElastiCache` namespace includes the following host-level metrics for individual cache nodes.

See Also

- Metrics for Memcached

Metric	Description	Unit
`CPUUtilization`	The percentage of CPU utilization for the entire host.	Percent
`CPUCreditBalance`	The number of earned CPU credits that an instance has accrued since it was launched or started. For T2 Standard, the CPUCredit Balance also includes the number of launch credits that have been accrued. Credits are accrued in the credit balance after they are earned, and removed from the credit balance when they are spent. The credit balance has a maximum limit, determined by the instance size. After the limit is reached, any new credits that are earned are discarded . For T2 Standard, launch credits do not count towards the limit. The credits in the CPUCreditBalance are available for the instance to spend to burst beyond its baseline CPU utilization.	Credits (vCPU-minutes)

Metric	Description	Unit
	CPU credit metrics are available at a five-minute frequency only.	
`CPUCreditUsage`	The number of CPU credits spent by the instance for CPU utilization. One CPU credit equals one vCPU running at 100% utilization for one minute or an equivalent combination of vCPUs, utilization, and time (for example, one vCPU running at 50% utilization for two minutes or two vCPUs running at 25% utilization for two minutes). CPU credit metrics are available at a five-minute frequency only. If you specify a period greater than five minutes, use the Sum statistic instead of the Average statistic.	Credits (vCPU-minutes)
`FreeableMemory`	The amount of free memory available on the host. This is derived from the RAM, buffers, and cache that the OS reports as freeable.	Bytes
`NetworkBytesIn`	The number of bytes the host has read from the network.	Bytes
`NetworkBytesOut`	The number of bytes sent out on all network interfaces by the instance.	Bytes
`NetworkPacketsIn`	The number of packets received on all network interfaces by the instance. This metric identifies the volume of incoming traffic in terms of the number of packets on a single instance.	Count
`NetworkPacketsOut`	The number of packets sent out on all network interfaces by the instance. This metric identifies the volume of outgoing traffic in terms of the number of packets on a single instance.	Count

Metric	Description	Unit
NetworkBandwidthInAllowanceExceeded	The number of packets shaped because the inbound aggregate bandwidth exceeded the maximum for the instance.	Count
NetworkConntrackAllowanceExceeded	The number of packets shaped because connection tracking exceeded the maximum for the instance and new connections could not be established. This can result in packet loss for traffic to or from the instance.	Count
NetworkBandwidthOutAllowanceExceeded	The number of packets shaped because the outbound aggregate bandwidth exceeded the maximum for the instance.	Count
Network Packets Per Second Allowance Exceeded	The number of packets shaped because the bidirectional packets per second exceeded the maximum for the instance.	Count
NetworkMaxBytesIn	The maximum burst of received bytes within each minute.	Bytes
NetworkMaxBytesOut	The maximum burst of transmitted bytes within each minute.	Bytes
NetworkMaxPacketsIn	The maximum burst of received packets within each minute.	Count
NetworkMaxPacketsOut	The maximum burst of transmitted packets within each minute.	Count
SwapUsage	The amount of swap used on the host.	Bytes

Metrics for Memcached

The AWS/ElastiCache namespace includes the following Memcached metrics.

The AWS/ElastiCache namespace includes the following metrics that are derived from the Memcached stats command. Each metric is calculated at the cache node level.

See also

- Host-Level Metrics

Metric	Description	Unit
`BytesReadIntoMemcached`	The number of bytes that have been read from the network by the cache node.	Bytes
`BytesUsedForCacheItems`	The number of bytes used to store cache items.	Bytes
`BytesWrittenOutFromMemcached`	The number of bytes that have been written to the network by the cache node.	Bytes
`CasBadval`	The number of CAS (check and set) requests the cache has received where the Cas value did not match the Cas value stored.	Count
`CasHits`	The number of Cas requests the cache has received where the requested key was found and the Cas value matched.	Count
`CasMisses`	The number of Cas requests the cache has received where the key requested was not found.	Count
`CmdFlush`	The number of flush commands the cache has received.	Count
`CmdGet`	The number of get commands the cache has received.	Count
`CmdSet`	The number of set commands the cache has received.	Count

Metric	Description	Unit
`CurrConnections`	A count of the number of connections connected to the cache at an instant in time. ElastiCache uses two to three of the connections to monitor the cluster. In addition to the above, memcached creates a number of internal connections equal to twice the threads used for the node type. The thread count for the various node types can be see in the `Nodetype Specific Parameters` of the applicable Parameter Group. The total connections is the sum of client connections, the connections for monitoring and the internal connections mentioned above.	Count
`CurrItems`	A count of the number of items currently stored in the cache.	Count
`DecrHits`	The number of decrement requests the cache has received where the requested key was found.	Count
`DecrMisses`	The number of decrement requests the cache has received where the requested key was not found.	Count
`DeleteHits`	The number of delete requests the cache has received where the requested key was found.	Count
`DeleteMisses`	The number of delete requests the cache has received where the requested key was not found.	Count
`Evictions`	The number of non-expired items the cache evicted to allow space for new writes.	Count

Metric	Description	Unit
GetHits	The number of get requests the cache has received where the key requested was found.	Count
GetMisses	The number of get requests the cache has received where the key requested was not found.	Count
IncrHits	The number of increment requests the cache has received where the key requested was found.	Count
IncrMisses	The number of increment requests the cache has received where the key requested was not found.	Count
Reclaimed	The number of expired items the cache evicted to allow space for new writes.	Count

For Memcached 1.4.14, the following additional metrics are provided.

Metric	Description	Unit
BytesUsedForHash	The number of bytes currently used by hash tables.	Bytes
CmdConfigGet	The cumulative number of config get requests.	Count
CmdConfigSet	The cumulative number of config set requests.	Count
CmdTouch	The cumulative number of touch requests.	Count
CurrConfig	The current number of configurations stored.	Count
EvictedUnfetched	The number of valid items evicted from the least recently used cache (LRU) which were never touched after being set.	Count

Metric	Description	Unit
`ExpiredUnfetched`	The number of expired items reclaimed from the LRU which were never touched after being set.	Count
`SlabsMoved`	The total number of slab pages that have been moved.	Count
`TouchHits`	The number of keys that have been touched and were given a new expiration time.	Count
`TouchMisses`	The number of items that have been touched, but were not found.	Count

The AWS/ElastiCache namespace includes the following calculated cache-level metrics.

Metric	Description	Unit
`NewConnections`	The number of new connections the cache has received. This is derived from the memcached total_connections statistic by recording the change in total_connections across a period of time. This will always be at least 1, due to a connection reserved for a ElastiCache.	Count
`NewItems`	The number of new items the cache has stored. This is derived from the memcached total_items statistic by recording the change in total_items across a period of time.	Count
`UnusedMemory`	The amount of memory not used by data. This is derived from the Memcached statistics limit_maxbytes and bytes by subtracting bytes from limit_maxbytes. Because Memcached overhead uses memory in addition to that used by data, UnusedMem	Bytes

Metric	Description	Unit
	ory should not be considered to be the amount of memory available for additional data. You may experience evictions even though you still have some unused memory. For more detailed information, see Memcached item memory usage.	

Which metrics should I monitor?

The following CloudWatch metrics offer good insight into ElastiCache performance. In most cases, we recommend that you set CloudWatch alarms for these metrics so that you can take corrective action before performance issues occur.

Metrics to Monitor

- CPUUtilization
- SwapUsage
- Evictions
- CurrConnections

CPUUtilization

This is a host-level metric reported as a percent. For more information, see Host-Level Metrics.

Because Memcached is multi-threaded, this metric can be as high as 90%. If you exceed this threshold, scale your cache cluster up by using a larger cache node type, or scale out by adding more cache nodes.

SwapUsage

This is a host-level metric reported in bytes. For more information, see Host-Level Metrics.

The `FreeableMemory` CloudWatch metric being close to 0 (i.e., below 100MB) or `SwapUsage` metric greater than the `FreeableMemory` metric indicates a node is under memory pressure. If this happens, we recommend that you increase the *ConnectionOverhead parameter value*.

Evictions

This is a cache engine metric. We recommend that you determine your own alarm threshold for this metric based on your application needs.

If you exceed your chosen threshold, scale your cluster up by using a larger node type, or scale out by adding more nodes.

CurrConnections

This is a cache engine metric. We recommend that you determine your own alarm threshold for this metric based on your application needs.

An increasing number of *CurrConnections* might indicate a problem with your application; you will need to investigate the application behavior to address this issue.

Monitoring CloudWatch Cluster and Node Metrics

ElastiCache and CloudWatch are integrated so you can gather a variety of metrics. You can monitor these metrics using CloudWatch.

> **Note**
>
> The following examples require the CloudWatch command line tools. For more information about CloudWatch and to download the developer tools, see the [CloudWatch product page](#).

The following procedures show you how to use CloudWatch to gather storage space statistics for an cache cluster for the past hour.

> **Note**
>
> The `StartTime` and `EndTime` values supplied in the examples below are for illustrative purposes. You must substitute appropriate start and end time values for your cache nodes.

For information on ElastiCache limits, see AWS Service Limits for ElastiCache.

Monitoring CloudWatch Cluster and Node Metrics (Console)

To gather CPU utilization statistics for a cache cluster

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Select the cache nodes you want to view metrics for.

   > **Note**
   >
   > Selecting more than 20 nodes disables viewing metrics on the console.

   a. On the **Cache Clusters** page of the AWS Management Console, click the name of one or more cache clusters.

The detail page for the cache cluster appears.

b. Click the **Nodes** tab at the top of the window.

c. On the **Nodes** tab of the detail window, select the cache nodes that you want to view metrics for.

   A list of available CloudWatch Metrics appears at the bottom of the console window.

d. Click on the **CPU Utilization** metric.

   The CloudWatch console will open, displaying your selected metrics. You can use the **Statistic** and **Period** drop-down list boxes and **Time Range** tab to change the metrics being displayed.

Monitoring CloudWatch Cluster and Node Metrics using the CloudWatch CLI

To gather CPU utilization statistics for a cache cluster

- For Linux, macOS, or 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
```

  For 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
```

Monitoring CloudWatch Cluster and Node Metrics using the CloudWatch API

To gather CPU utilization statistics for a cache cluster

- Call the CloudWatch API `GetMetricStatistics` with the following parameters (note that the start and end times are shown as examples only; you will need to substitute your own appropriate start and end times):
  - `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 monitoring of ElastiCache events

When significant events happen for a cluster, ElastiCache sends notification to a specific Amazon SNS topic. Examples include a failure to add a node, success in adding a node, the modification of a security group, and others. By monitoring for key events, you can know the current state of your clusters and, depending upon the event, be able to take corrective action.

Topics

- Managing ElastiCache Amazon SNS notifications
- Viewing ElastiCache events
- Event Notifications and Amazon SNS

Managing ElastiCache Amazon SNS notifications

You can configure ElastiCache to send notifications for important cluster events using Amazon Simple Notification Service (Amazon SNS). In these examples, you will configure a cluster with the Amazon Resource Name (ARN) of an Amazon SNS topic to receive notifications.

> **Note**
>
> This topic assumes that you've signed up for Amazon SNS and have set up and subscribed to an Amazon SNS topic. For information on how to do this, see the Amazon Simple Notification Service Developer Guide.

Adding an Amazon SNS topic

The following sections show you how to add an Amazon SNS topic using the AWS Console, the AWS CLI, or the ElastiCache API.

Adding an Amazon SNS topic (Console)

The following procedure shows you how to add an Amazon SNS topic for a cluster.

> **Note**
>
> This process can also be used to modify the Amazon SNS topic.

To add or modify an Amazon SNS topic for a cluster (Console)

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In **Clusters**, choose the cluster for which you want to add or modify an Amazon SNS topic ARN.
3. Choose **Modify**.
4. In **Modify Cluster** under **Topic for SNS Notification**, choose the SNS topic you want to add, or choose **Manual ARN input** and type the ARN of the Amazon SNS topic.
5. Choose **Modify**.

Adding an Amazon SNS topic (AWS CLI)

To add or modify an Amazon SNS topic for a cluster, use the AWS CLI command `modify-cache-cluster`.

The following code example adds an Amazon SNS topic arn to *my-cluster*.

For Linux, macOS, or Unix:

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --notification-topic-arn arn:aws:sns:us-
west-2:123456789xxx:ElastiCacheNotifications
```

For Windows:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --notification-topic-arn arn:aws:sns:us-west-2:123456789xx:ElastiCacheNotifications
```

For more information, see modify-cache-cluster.

Adding an Amazon SNS topic (ElastiCache API)

To add or modify an Amazon SNS topic for a cluster, call the `ModifyCacheCluster` action with the following parameters:

- `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>
```

For more information, see ModifyCacheCluster.

Enabling and disabling Amazon SNS notifications

You can turn notifications on or off for a cluster. The following procedures show you how to disable Amazon SNS notifications.

Enabling and disabling Amazon SNS notifications (Console)

To disable Amazon SNS notifications using the AWS Management Console

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. To see a list of your clusters running Memcached, in the navigation pane choose **Memcached**.
3. Choose the box to the left of the cluster you want to modify notification for.
4. Choose **Modify**.
5. In **Modify Cluster** under **Topic for SNS Notification**, choose *Disable Notifications*.
6. Choose **Modify**.

Enabling and disabling Amazon SNS notifications (AWS CLI)

To disable Amazon SNS notifications, use the command `modify-cache-cluster` with the following parameters:

For Linux, macOS, or Unix:

```
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \
    --notification-topic-status inactive
```

For Windows:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cluster ^
    --notification-topic-status inactive
```

Enabling and disabling Amazon SNS notifications (ElastiCache API)

To disable Amazon SNS notifications, call the `ModifyCacheCluster` action with the following parameters:

- `CacheClusterId=my-cluster`
- `NotificationTopicStatus=inactive`

This call returns output similar to the following:

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>
```

Viewing ElastiCache events

ElastiCache logs events that relate to your cluster instances, security groups, and parameter groups. This information includes the date and time of the event, the source name and source type of the event, and a description of the event. You can easily retrieve events from the log using the ElastiCache console, the AWS CLI `describe-events` command, or the ElastiCache API action `DescribeEvents`.

The following procedures show you how to view all ElastiCache events for the past 24 hours (1440 minutes).

Viewing ElastiCache events (Console)

The following procedure displays events using the ElastiCache console.

To view events using the ElastiCache console

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. To see a list of all available events, in the navigation pane, choose **Events**.

   On the *Events* screen each row of the list represents one event and displays the event source, the event type (cache-cluster, cache-parameter-group, cache-security-group, or cache-subnet-group), the GMT time of the event, and a description of the event.

   Using the **Filter** you can specify whether you want to see all events, or just events of a specific type in the event list.

Viewing ElastiCache events (AWS CLI)

To generate a list of ElastiCache events using the AWS CLI, use the command `describe-events`. You can use optional parameters to control the type of events listed, the time frame of the events listed, the maximum number of events to list, and more.

The following code lists up to 40 cache cluster events.

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
```

The following code lists all events for the past 24 hours (1440 minutes).

```
aws elasticache describe-events --source-type cache-cluster --duration 1440
```

The output from the `describe-events` command looks something like this.

```
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"
        }
    ]
}
```

For more information, such as available parameters and permitted parameter values, see describe-events.

Viewing ElastiCache events (ElastiCache API)

To generate a list of ElastiCache events using the ElastiCache API, use the `DescribeEvents` action. You can use optional parameters to control the type of events listed, the time frame of the events listed, the maximum number of events to list, and more.

The following code lists the 40 most recent cache-cluster events.

```
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>
```

The following code lists the cache-cluster events for the past 24 hours (1440 minutes).

```
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>
```

The above actions should produce output similar to the following.

```
<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>
```

For more information, such as available parameters and permitted parameter values, see `DescribeEvents`.

Event Notifications and Amazon SNS

ElastiCache can publish messages using Amazon Simple Notification Service (SNS) when significant events happen on a cache cluster. This feature can be used to refresh the server-lists on client machines connected to individual cache node endpoints of a cache cluster.

Note

For more information on Amazon Simple Notification Service (SNS), including information on pricing and links to the Amazon SNS documentation, see the Amazon SNS product page.

Notifications are published to a specified Amazon SNS *topic*. The following are requirements for notifications:

- Only one topic can be configured for ElastiCache notifications.
- The AWS account that owns the Amazon SNS topic must be the same account that owns the cache cluster on which notifications are enabled.
- The Amazon SNS topic you are publishing to cannot be encrypted.

Note

It is possible to attach an encrypted (at-rest) Amazon SNS topic to the cluster. However, the status of the topic from the ElastiCache console will show as inactive, which effectively disassociates the topic from the cluster when ElastiCache pushes messages to the topic.

- The Amazon SNS topic has to be in the same Region as the ElastiCache cluster.

ElastiCache Events

The following ElastiCache events trigger Amazon SNS notifications. For information on event details, see Viewing ElastiCache events.

Event Name	Message	Description
ElastiCache:AddCacheNodeComplete	`ElastiCache:AddCacheNodeComplete : ` *`cache-cluster`*	A cache node has been added to the cache cluster and is ready for use.
ElastiCache:AddCacheNodeFailed due to insufficient free IP addresses	`ElastiCache:AddCacheNodeFailed : ` *`cluster-name`*	A cache node could not be added because there are not enough available IP addresses .
ElastiCache:CacheClusterParametersChanged	`ElastiCache:CacheClusterParametersChanged : ` *`cluster-name`*	One or more cache cluster parameters have been changed.
ElastiCache:CacheClusterProvisioningComplete	`ElastiCache:CacheClusterProvisioningComplete ` *`cluster-name-0001-005`*	The provisioning of a cache cluster is completed, and the cache nodes in the cache cluster are ready to use.
ElastiCache:CacheClusterProvisioningFailed due to incompatible network state	`ElastiCache:CacheClusterProvisioning`	An attempt was made to launch a new cache cluster

Event Name	Message	Description
	`Failed : ` *`cluster-name`*	into a nonexistent virtual private cloud (VPC).
ElastiCache:CacheClusterScalingComplete	`CacheClusterScalingComplete : ` *`cluster-name`*	Scaling for cache-cluster completed successfully.
ElastiCache:CacheClusterScalingFailed	ElastiCache:CacheClusterScalingFailed : *`cluster-name`*	Scale-up operation on cache-cluster failed.
ElastiCache:CacheClusterSecurityGroupModified	`ElastiCache:CacheClusterSecurityGroupModified : ` *`cluster-name`*	One of the following events has occurred: - The list of cache security groups authorized for the cache cluster has been modified. - One or more new EC2 security groups have been authorized on any of the cache security groups associated with the cache cluster. - One or more EC2 security groups have been revoked from any of the cache security groups associated with the cache cluster.

Event Name	Message	Description
ElastiCache:CacheNodeReplaceStarted	`ElastiCache:CacheNodeReplaceStarted : *cluster-name*`	ElastiCache has detected that the host running a cache node is degraded or unreachable and has started replacing the cache node. **Note** The DNS entry for the replaced cache node is not changed. In most instances, you do not need to refresh the server-list for your clients when this event occurs. However, some cache client libraries may stop using the cache node even after ElastiCache has replaced the cache node; in this case, the application should refresh the server-list when this event occurs.

Event Name	Message	Description
ElastiCache:CacheNodeReplaceComplete	`ElastiCache:CacheNodeReplaceComplete : ` *`cluster-name`*	ElastiCache has detected that the host running a cache node is degraded or unreachable and has completed replacing the cache node. **Note** The DNS entry for the replaced cache node is not changed. In most instances, you do not need to refresh the server-list for your clients when this event occurs. However, some cache client libraries may stop using the cache node even after ElastiCache has replaced the cache node; in this case, the application should refresh the server-list when this event occurs.
ElastiCache:CacheNodesRebooted	`ElastiCache:CacheNodesRebooted : ` *`cluster-name`*	One or more cache nodes has been rebooted. Message (Memcached): `"Cache node %s shutdown"` Then a second message: `"Cache node %s restarted"`

Event Name	Message	Description
ElastiCache:CertificateRenewalComplete (Redis only)	`ElastiCache:CertificateRenewalComplete`	The Amazon CA certificate was successfully renewed.
ElastiCache:CreateReplicationGroupComplete	`ElastiCache:CreateReplicationGroupComplete :` *cluster-name*	The replication group was successfully created.
ElastiCache:DeleteCacheClusterComplete	`ElastiCache:DeleteCacheClusterComplete :` *cluster-name*	The deletion of a cache cluster and all associated cache nodes has completed.
ElastiCache:FailoverComplete (Redis only)	`ElastiCache:FailoverComplete :` *mycluster*	Failover over to a replica node was successful.
ElastiCache:ReplicationGroupIncreaseReplicaCountFinished	`ElastiCache:ReplicationGroupIncreaseReplicaCountFinished :` *cluster-name-0001-005*	The number of replicas in the cluster has been increased.
ElastiCache:ReplicationGroupIncreaseReplicaCountStarted	`ElastiCache:ReplicationGroupIncreaseReplicaCountStarted :` *cluster-name-0003-004*	The process of adding replicas to your cluster has begun.
ElastiCache:NodeReplacementCanceled	`ElastiCache:NodeReplacementCanceled :` *cluster-name*	A node in your cluster that was scheduled for replacement is no longer scheduled for replacement.

Event Name	Message	Description
ElastiCache:NodeReplacementRescheduled	`ElastiCache:NodeReplacementRescheduled : cluster-name`	A node in your cluster previously scheduled for replacement has been rescheduled for replacement during the new window described in the notification. For information on what actions you can take, see Replacing cache nodes for Memcached.
ElastiCache:NodeReplacementScheduled	`ElastiCache:NodeReplacementScheduled : cluster-name`	A node in your cluster is scheduled for replacement during the window described in the notification. For information on what actions you can take, see Replacing cache nodes for Memcached.
ElastiCache:RemoveCacheNodeComplete	`ElastiCache:RemoveCacheNodeComplete : cluster-name`	A cache node has been removed from the cache cluster.
ElastiCache:ReplicationGroupScalingComplete	`ElastiCache:ReplicationGroupScalingComplete : cluster-name`	Scale-up operation on replication group completed successfully.
ElastiCache:ReplicationGroupScalingFailed	`"Failed applying modification to cache node type to %s."`	Scale-up operation on replication group failed.

Event Name	Message	Description
ElastiCache:ServiceUpdateAvailableForNode	`"Service update is available for cache node %s."`	A self-service update is available for the node.
ElastiCache:SnapshotComplete (Redis only)	`ElastiCache:SnapshotComplete :` *`cluster-name`*	A cache snapshot has completed successfully.
ElastiCache:SnapshotFailed (Redis only)	`SnapshotFailed :` *`cluster-name`*	A cache snapshot has failed. See the cluster's cache events for more a detailed cause. If you describe the snapshot, see `DescribeSnapshots` , the status will be `failed`.

Related topics

- Viewing ElastiCache events

Logging Amazon ElastiCache API calls with AWS CloudTrail

Amazon ElastiCache is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in Amazon ElastiCache. CloudTrail captures all API calls for Amazon ElastiCache as events, including calls from the Amazon ElastiCache console and from code calls to the Amazon ElastiCache API operations. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for Amazon ElastiCache. If you don't configure a trail, you can still view the most recent events in the CloudTrail console in **Event history**. Using the information collected by CloudTrail, you can determine the request that was made to Amazon ElastiCache, the IP address from which the request was made, who made the request, when it was made, and additional details.

To learn more about CloudTrail, see the AWS CloudTrail User Guide.

Amazon ElastiCache information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in Amazon ElastiCache, that activity is recorded in a CloudTrail event along with other AWS service events in **Event history**. You can view, search, and download recent events in your AWS account. For more information, see Viewing Events with CloudTrail Event History.

For an ongoing record of events in your AWS account, including events for Amazon ElastiCache, create a trail. A trail enables CloudTrail to deliver log files to an Amazon S3 bucket. By default, when you create a trail in the console, the trail applies to all regions. The trail logs events from all regions in the AWS partition and delivers the log files to the Amazon S3 bucket that you specify. Additionally, you can configure other AWS services to further analyze and act upon the event data collected in CloudTrail logs. For more information, see the following:

- Overview for Creating a Trail
- CloudTrail Supported Services and Integrations
- Configuring Amazon SNS Notifications for CloudTrail
- Receiving CloudTrail Log Files from Multiple Regions and Receiving CloudTrail Log Files from Multiple Accounts

All Amazon ElastiCache actions are logged by CloudTrail and are documented in the ElastiCache API Reference. For example, calls to the `CreateCacheCluster`, `DescribeCacheCluster` and `ModifyCacheCluster` actions generate entries in the CloudTrail log files.

Every event or log entry contains information about who generated the request. The identity information helps you determine the following:

- Whether the request was made with root or IAM user credentials.
- Whether the request was made with temporary security credentials for a role or federated user.
- Whether the request was made by another AWS service.

For more information, see the CloudTrail userIdentity Element.

Understanding Amazon ElastiCache log file entries

A trail is a configuration that enables delivery of events as log files to an Amazon S3 bucket that you specify. CloudTrail log files contain one or more log entries. An event represents a single request from any source and includes information about the requested action, the date and time of the action, request parameters, and so on. CloudTrail log files are not an ordered stack trace of the public API calls, so they do not appear in any specific order.

The following example shows a CloudTrail log entry that demonstrates the `CreateCacheCluster` action.

```
{ 
    "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"
}
```

The following example shows a CloudTrail log entry that demonstrates the `DescribeCacheCluster` action. Note that for all Amazon ElastiCache Describe calls (`Describe*`), the `ResponseElements` section is removed and appears as `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"
}
```

The following example shows a CloudTrail log entry that records a `ModifyCacheCluster` action.

```
{
    "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"
}
```

Quotas for ElastiCache

Your AWS account has default quotas, formerly referred to as limits, for each AWS service. Unless otherwise noted, each quota is Region-specific. You can request increases for some quotas, and other quotas cannot be increased.

To view the quotas for ElastiCache, open the Service Quotas console. In the navigation pane, choose **AWS services** and select **ElastiCache**.

To request a quota increase, see Requesting a Quota Increase in the *Service Quotas User Guide*. If the quota is not yet available in Service Quotas, use the limit increase form.

Your AWS account has the following quotas related to ElastiCache.

Resource	Default
Serverless caches per region	40
Nodes per Region	300
Nodes per cluster	60
Parameter groups per Region	300
Security groups per Region	50
Subnet groups per Region	300
Subnets per subnet group	20

Reference

The topics in this section cover working with the Amazon ElastiCache API and the ElastiCache section of the AWS CLI. Also included in this section are common error messages and service notifications.

- Using the ElastiCache API
- ElastiCache API Reference
- ElastiCache section of the AWS CLI Reference
- Amazon ElastiCache error messages
- Notifications

Using the ElastiCache API

This section provides task-oriented descriptions of how to use and implement ElastiCache operations. For a complete description of these operations, see the Amazon ElastiCache API Reference

Topics

- Using the query API
- Available libraries
- Troubleshooting applications

Using the query API

Query parameters

HTTP Query-based requests are HTTP requests that use the HTTP verb GET or POST and a Query parameter named `Action`.

Each Query request must include some common parameters to handle authentication and selection of an action.

Some operations take lists of parameters. These lists are specified using the `param.`*n* notation. Values of *n* are integers starting from 1.

Query request authentication

You can only send Query requests over HTTPS and you must include a signature in every Query request. This section describes how to create the signature. The method described in the following procedure is known as *signature version 4*.

The following are the basic steps used to authenticate requests to AWS. This assumes you are registered with AWS and have an Access Key ID and Secret Access Key.

Query authentication process

1. The sender constructs a request to AWS.
2. The sender calculates the request signature, a Keyed-Hashing for Hash-based Message Authentication Code (HMAC) with a SHA-1 hash function, as defined in the next section of this topic.
3. The sender of the request sends the request data, the signature, and Access Key ID (the key-identifier of the Secret Access Key used) to AWS.
4. AWS uses the Access Key ID to look up the Secret Access Key.
5. AWS generates a signature from the request data and the Secret Access Key using the same algorithm used to calculate the signature in the request.
6. If the signatures match, the request is considered to be authentic. If the comparison fails, the request is discarded, and AWS returns an error response.

> **Note**
>
> If a request contains a `Timestamp` parameter, the signature calculated for the request expires 15 minutes after its value.
> If a request contains an `Expires` parameter, the signature expires at the time specified by the `Expires` parameter.

To calculate the request signature

1. Create the canonicalized query string that you need later in this procedure:

   a. Sort the UTF-8 query string components by parameter name with natural byte ordering. The parameters can come from the GET URI or from the POST body (when Content-Type is application/x-www-form-urlencoded).

   b. URL encode the parameter name and values according to the following rules:

      i. Do not URL encode any of the unreserved characters that RFC 3986 defines. These unreserved characters are A-Z, a-z, 0-9, hyphen (-), underscore (_), period (.), and tilde (~).

      ii. Percent encode all other characters with %XY, where X and Y are hex characters 0-9 and uppercase A-F.

      iii. Percent encode extended UTF-8 characters in the form %XY%ZA....

      iv. Percent encode the space character as %20 (and not +, as common encoding schemes do).

   c. Separate the encoded parameter names from their encoded values with the equals sign (=) (ASCII character 61), even if the parameter value is empty.

   d. Separate the name-value pairs with an ampersand (&) (ASCII code 38).

2. Create the string to sign according to the following pseudo-grammar (the "\n" represents an ASCII newline).

   ```
   StringToSign = HTTPVerb + "\n" +
   ValueOfHostHeaderInLowercase + "\n" +
   HTTPRequestURI + "\n" +
   CanonicalizedQueryString <from the preceding step>
   ```

   The HTTPRequestURI component is the HTTP absolute path component of the URI up to, but not including, the query string. If the HTTPRequestURI is empty, use a forward slash (/).

3. Calculate an RFC 2104-compliant HMAC with the string you just created, your Secret Access Key as the key, and SHA256 or SHA1 as the hash algorithm.

   For more information, see https://www.ietf.org/rfc/rfc2104.txt.

4. Convert the resulting value to base64.

5. Include the value as the value of the `Signature` parameter in the request.

For example, the following is a sample request (linebreaks added for clarity).

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeCacheClusters
    &CacheClusterIdentifier=myCacheCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Version=2014-12-01
```

For the preceding query string, you would calculate the HMAC signature over the following string.

```
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:
```

The result is the following signed request.

```
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
```

For detailed information on the signing process and calculating the request signature, see the topic Signature Version 4 Signing Process and its subtopics.

Available libraries

AWS provides software development kits (SDKs) for software developers who prefer to build applications using language-specific APIs instead of the Query API. These SDKs provide basic functions (not included in the APIs), such as request authentication, request retries, and error handling so that it is easier to get started. SDKs and additional resources are available for the following programming languages:

- Java
- Windows and .NET
- PHP
- Python
- Ruby

For information about other languages, see Sample Code & Libraries.

Troubleshooting applications

ElastiCache provides specific and descriptive errors to help you troubleshoot problems while interacting with the ElastiCache API.

Retrieving errors

Typically, you want your application to check whether a request generated an error before you spend any time processing results. The easiest way to find out if an error occurred is to look for an `Error` node in the response from the ElastiCache API.

XPath syntax provides a simple way to search for the presence of an `Error` node, as well as an easy way to retrieve the error code and message. The following code snippet uses Perl and the XML::XPath module to determine if an error occurred during a request. If an error occurred, the code prints the first error code and message in the response.

```
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"; }
```

Troubleshooting tips

We recommend the following processes to diagnose and resolve problems with the ElastiCache API.

- Verify that ElastiCache is running correctly.

  To do this, simply open a browser window and submit a query request to the ElastiCache service (such as https://elasticache.amazonaws.com). A MissingAuthenticationTokenException or 500 Internal Server Error confirms that the service is available and responding to requests.
- Check the structure of your request.

  Each ElastiCache operation has a reference page in the *ElastiCache API Reference*. Double-check that you are using parameters correctly. To give you ideas regarding what might be wrong, look at the sample requests or user scenarios to see if those examples are doing similar operations.
- Check the forum.

  ElastiCache has a discussion forum where you can search for solutions to problems others have experienced along the way. To view the forum, see

  https://forums.aws.amazon.com/ .

Setting up the ElastiCache command line interface

This section describes the prerequisites for running the command line tools, where to get the command line tools, how to set up the tools and their environment, and includes a series of common examples of tool usage.

Follow the instructions in this topic only if you are going to the AWS CLI for ElastiCache.

Important

The Amazon ElastiCache Command Line Interface (CLI) does not support any ElastiCache improvements after API version 2014-09-30. To use newer ElastiCache functionality from the command line, use the AWS Command Line Interface.

Topics

Prerequisites

This document assumes that you can work in a Linux/UNIX or Windows environment. The Amazon ElastiCache command line tools also work on Mac OS X, which is a UNIX-based environment; however, no specific Mac OS X instructions are included in this guide.

As a convention, all command line text is prefixed with a generic **`PROMPT>`** command line prompt. The actual command line prompt on your machine is likely to be different. We also use **`$`** to indicate a Linux/UNIX specific command and **`C:\>`** for a Windows specific command. The example output resulting from the command is shown immediately thereafter without any prefix.

The Java runtime environment

The command line tools used in this guide require Java version 5 or later to run. Either a JRE or JDK installation is acceptable. To view and download JREs for a range of platforms, including Linux/UNIX and Windows, see Java SE Downloads.

Setting the Java home variable

The command line tools depend on an environment variable (`JAVA_HOME`) to locate the Java Runtime. This environment variable should be set to the full path of the directory that contains a subdirectory named `bin` which in turn contains the executable `java` (on Linux and UNIX) or `java.exe` (on Windows) executable.

To set the Java Home variable

1. Set the Java Home variable.

   - On Linux and UNIX, enter the following command:

   ```
   $ export JAVA_HOME=<PATH>
   ```

   - On Windows, enter the following command:

   ```
   C:\> set JAVA_HOME=<PATH>
   ```

2. Confirm the path setting by running **$JAVA_HOME/bin/java -version** and checking the output.

   - On Linux/UNIX, you will see output similar to the following:

   ```
   $ $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)
   ```

   - On Windows, you will see output similar to the following:

   ```
   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)
   ```

Getting the command line tools

The command line tools are available as a ZIP file on the ElastiCache Developer Tools web site. These tools are written in Java, and include shell scripts for Windows 2000/XP/Vista/Windows 7, Linux/UNIX, and Mac OSX. The ZIP file is self-contained and no installation is required; simply download the zip file and unzip it to a directory on your local machine.

Setting up the tools

The command line tools depend on an environment variable (AWS_ELASTICACHE_HOME) to locate supporting libraries. You need to set this environment variable before you can use the tools. Set it to the path of the directory you unzipped the command line tools into. This directory is named ElastiCacheCli-A.B.nnnn (A, B and n are version/release numbers), and contains subdirectories named bin and lib.

To set the AWS_ELASTICACHE_HOME environment variable

- Open a command line window and enter one of the following commands to set the AWS_ELASTICACHE_HOME environment variable.
  - On Linux and UNIX, enter the following command:

    ```
    $ export &AWS;_ELASTICACHE_HOME=<path-to-tools>
    ```
  - On Windows, enter the following command:

    ```
    C:\> set &AWS;_ELASTICACHE_HOME=<path-to-tools>
    ```

To make the tools easier to use, we recommend that you add the tools' BIN directory to your system PATH. The rest of this guide assumes that the BIN directory is in your system path.

To add the tools' BIN directory to your system path

- Enter the following commands to add the tools' BIN directory to your system PATH.
  - On Linux and UNIX, enter the following command:

    ```
    $ export PATH=$PATH:$&AWS;_ELASTICACHE_HOME/bin
    ```
  - On Windows, enter the following command:

    ```
    C:\> set PATH=%PATH%;%&AWS;_ELASTICACHE_HOME%\bin
    ```

Note

The Windows environment variables are reset when you close the command window. You might want to set them permanently. Consult the documentation for your version of Windows for more information.

Note

Paths that contain a space must be wrapped in double quotes, for example: "C:\Program Files\Java"

Providing credentials for the tools

The command line tools need the AWS Access Key and Secret Access Key provided with your AWS account. You can get them using the command line or from a credential file located on your local system.

The deployment includes a template file ${AWS_ELASTICACHE_HOME}/credential-file-path.template that you need to edit with your information. Following are the contents of the template file:

```
AWSAccessKeyId=<Write your AWS access ID>
AWSSecretKey=<Write your AWS secret key>
```

Important

On UNIX, limit permissions to the owner of the credential file:

```
$ chmod 600 <the file created above>
```

With the credentials file setup, you'll need to set the AWS_CREDENTIAL_FILE environment variable so that the ElastiCache tools can find your information.

To set the AWS_CREDENTIAL_FILE environment variable

1. Set the environment variable:

    - On Linux and UNIX, update the variable using the following command:

    ```
    $ export &AWS;_CREDENTIAL_FILE=<the file created above>
    ```

    - On Windows, set the variable using the following command:

    ```
    C:\> set &AWS;_CREDENTIAL_FILE=<the file created above>
    ```

2. Check that your setup works properly, run the following command:

    ```
    elasticache --help
    ```

    You should see the usage page for all ElastiCache commands.

Environmental variables

Environment variables can be useful for scripting, configuring defaults or temporarily overriding them.

In addition to the AWS_CREDENTIAL_FILE environment variable, most API tools included with the ElastiCache Command Line Interface support the following variables:

- **EC2_REGION** — The AWS region to use.
- **AWS_ELASTICACHE_URL** — The URL to use for the service call. Not required to specify a different regional endpoint if EC2_REGION is specified or the --region parameter is passed.

The following examples show how to set the environmental variable EC2_REGION to configure the region used by the API tools:

Linux, OS X, or Unix

```

$ export EC2_REGION=us-west-1
```

Windows

```
$ set EC2_REGION=us-west-1
```

Amazon ElastiCache error messages

The following error messages are returned by Amazon ElastiCache. You may receive other error messages that are returned by ElastiCache, other AWS services, or by Memcached. For descriptions of error messages from sources other than ElastiCache, see the documentation from the source that is generating the error message.

- Cluster node quota exceeded
- Customer's node quota exceeded
- Insufficient cache cluster capacity

Error Message: **Cluster node quota exceeded. Each cluster can have at most *%n* nodes in this region.**

> **Cause:** You attempted to create or modify a cluster with the result that the cluster would have more than *%n* nodes.
>
> **Solution:** Change your request so that the cluster does not have more than *%n* nodes. Or, if you need more than *%n* nodes, make your request using the Amazon ElastiCache Node request form.
>
> For more information, see Amazon ElastiCache Limits in *Amazon Web Services General Reference*.

Error Messages: **Customer node quota exceeded. You can have at most *%n* nodes in this region** Or, **You have already reached your quota of %s nodes in this region.**

> **Cause:** You attempted to create or modify a cluster with the result that your account would have more than *%n* nodes across all clusters in this region.
>
> **Solution:** Change your request so that the total nodes in the region across all clusters for this account does not exceed *%n*. Or, if you need more than *%n* nodes, make your request using the Amazon ElastiCache Node request form.

For more information, see Amazon ElastiCache Limits in *Amazon Web Services General Reference*.

Error Messages: **InsufficientCacheClusterCapacity**

Cause: AWS does not currently have enough available On-Demand capacity to service your request.

Solution:

- Wait a few minutes and then submit your request again; capacity can shift frequently.
- Submit a new request with a reduced number of nodes or shards (node groups). For example, if you're making a single request to launch 15 nodes, try making 3 requests of 5 nodes, or 15 requests for 1 node instead.
- If you're launching a cluster, submit a new request without specifying an Availability Zone.
- If you're launching a cluster, submit a new request using a different node type (which you can scale up at a later stage). For more information, see Scaling ElastiCache for Memcached.

Notifications

This topic covers ElastiCache notifications that you might be interested in. A notification is a situation or event that, in most cases, is temporary, lasting only until a solution is found and implemented. Notifications generally have a start date and a resolution date, after which the notification is no longer relevant. Any one notification might or might not be relevant to you. We recommend an implementation guideline that, if followed, improves the performance of your cluster.

Notifications do not announce new or improved ElastiCache features or functionality.

General ElastiCache notifications

Currently there are no outstanding ElastiCache notifications that are not engine specific.

ElastiCache for Memcached notifications

The following ElastiCache notifications are specific to the Memcached engine.

ElastiCache for Memcached specific notifications

- Alert: Memcached LRU crawler causing segmentation faults

Alert: Memcached LRU crawler causing segmentation faults

> ⚠ Alert Date: February 28, 2017
>
> In some circumstances, your cluster might display instability with a segmentation fault in the Memcached LRU Crawler. This is an issue within the Memcached engine that has existed for some time. The issue became apparent in Memcached 1.4.33 when the LRU Crawler was enabled by default.
>
> If you are experiencing this issue, we recommend that you disable the LRU Crawler until there is a fix. To do so, use `lru_crawler disable` at the command line or modify the `lru_crawler` parameter value (preferred).
>
> Resolved Date:
>
> Resolution:

Documentation history

- **API version:** 2015-02-02
- **Latest documentation update:** November 27, 2023

The following table describes important changes in each release of the *ElastiCache for Memcached User Guide* after March 2018. For notification about updates to this documentation, you can subscribe to the RSS feed.

Recent ElastiCache for Memcached Updates

Change	Description	Date
Support for ElastiCache for Memcached 1.6.22	ElastiCache for Memcached now supports Memcached 1.6.22. It includes cumulative bug fixes from version 1.6.18. For more information, see Memcached Version 1.6.22.	January 10, 2024
ElastiCache for Memcached now supports creation of serverless caches	You can now create serverless caches, which simplify cache management and instantly scale to support the most demanding applications. For more information, see Choosing between deployment options. As part of this feature, new permissions were added to `ElastiCacheServiceRolePolicy` and `AmazonElastiCacheFullAccess` to allow association of serverless caches with managed VPC endpoints. Additionally,	November 27, 2023

	permissions were added to support a revised console experience using `AmazonElastiCacheFullAccess` policy.	
Support for ElastiCache for Memcached 1.6.17	ElastiCache for Memcached now supports Memcached 1.6.17. It includes cumulative bug fixes from version Memcached 1.6.12 For more information, see Memcached Version 1.6.17.	January 18, 2023
ElastiCache for Memcached now supports IPV6	ElastiCache supports the Internet Protocol versions 4 and 6 (IPv4 and IPv6), allowing you to configure your cluster to accept only IPv4 connections, only IPv6 connections or both IPv4 and IPv6 connections (dual-stack). IPv6 is supported for workloads using Memcached engine version 1.6.6 onward on all instances built on the Nitro system. There are no additional charges for accessing ElastiCache over IPv6. For more information, see Choosing a network type.	November 7, 2022

ElastiCache for Memcached now supports in-transit encryption	In-transit encryption is an optional feature that allows you to increase the security of your data at its most vulnerable points – when it is in transit from one location to another. It is supported on Memcached versions 1.6.12 and later. For more information, see ElastiCache in-transit encryption (TLS).	May 26, 2022
ElastiCache now supports PrivateLink	AWS PrivateLink allows you to privately access ElastiCache API operations without an internet gateway, NAT device, VPN connection, or AWS Direct Connect connection. For more information, see Amazon ElastiCache API and interface VPC endpoints (AWS PrivateLink) for Redis or Amazon ElastiCache API and interface VPC endpoints (AWS PrivateLink) for Memcached.	January 24, 2022

Support for tagging resources and condition keys	ElastiCache now supports tagging to help you manage your clusters and other ElastiCache resources. For more information, see Tagging your ElastiCache resources. ElastiCache also introduces support for condition keys. You can specify conditions that determine how an IAM policy takes effect. For more information, see Using condition keys.	April 7, 2021
Support for Memcached 1.6.6.	ElastiCache for Memcached now supports Memcached 1.6.6. It includes cumulative bug fixes from versions Memcached 1.5.16. For more information, see Memcached Version 1.6.6.	November 12, 2020
ElastiCache is now available on AWS Outposts	AWS Outposts bring native AWS services, infrastructure, and operating models to virtually any data center, co-location space, or on-premises facility. You can deploy ElastiCache on Outposts to set up, operate, and use cache on-premises, just as you would in the cloud. For more information, see Using Outposts for Redis or Using Outposts for Memcached.	October 8, 2020

ElastiCache now supports Local Zones	A *Local Zone* is an extension of an AWS Region that is geographically close to your users. You can extend any virtual private cloud (VPC) from a parent AWS Region into Local Zones by creating a new subnet and assigning it to a Local Zone. For more information, see Using Local Zones.	September 25, 2020
ElastiCache now supports resource-level permissions	You can now restrict the scope of a user's permissions by specifying ElastiCache resources in an AWS Identity and Access Management (IAM) policy. For more information, see Resource-level permissions.	August 12, 2020
ElastiCache now supports auto-update of ElastiCache clusters	Amazon ElastiCache now supports auto-update of ElastiCache clusters after the "recommended apply by date" of service update has passed. ElastiCache uses your maintenance window to schedule the automatic update of applicable clusters. For more information, see Self-service updates.	May 13, 2020

Amazon ElastiCache now supports T3-Standard cache nodes	You can now launch the next generation general-purpose burstable T3-Standard cache nodes in Amazon ElastiCache. Amazon EC2’s T3-Standard instances provide a baseline level of CPU performance with the ability to burst CPU usage at any time until the accrued credits are exhausted . For more information, see Supported Node Types.	November 12, 2019
ElastiCache for Memcached now allows users to apply service updates on their own schedule	With this feature, you can choose to apply available service updates at a time of your choosing and not just during maintenance windows. This will minimize service interruptions, particularly during peak business flows, and help ensure you remain compliant with security updates. For more information, see Self-Service Updates in Amazon ElastiCache.	October 9, 2019
Support for ElastiCache for Memcached 1.5.16	ElastiCache for Memcached now supports Memcached 1.5.16. It includes cumulative bug fixes from versions Memcached 1.5.14 and Memcached 1.5.15. For more information, see Memcached Version 1.5.16.	September 6, 2019

ElastiCache Standard Reserved Instance offerings : Partial Upfront, All Upfront and No Upfront.	Reserved Instances give you the flexibility to reserve an Amazon ElastiCache instance for a one- or three-year term based on an instance type and AWS Region. For more information, see Managing Costs with Reserved Nodes.	January 18, 2019
Support for ElastiCache for Memcached 1.5.10	ElastiCache for Memcached now supports Memcached 1.5.10. It includes support for the no_modern and inline_ascii_resp parameters. For more information, see Memcached Version 1.5.10.	November 14, 2018
User Guide restructure	The single *ElastiCache User Guide* is now restructured so that there are separate user guides for Redis (ElastiCache for Redis User Guide) and for Memcached (ElastiCache for Memcached User Guide). The documentation structure in the AWS CLI Command Reference: elasticache section and the Amazon ElastiCache API Reference remain unchanged.	April 20, 2018

The following table describes the important changes to the *ElastiCache for Memcached User Guide* before March 2018.

Change	Description	Date Changed
Support for Asia Pacific (Osaka-local) Region.	ElastiCache added support for the Asia Pacific (Osaka-local) Region. The Asia Pacific (Osaka) Region currently supports a single Availability Zone and is by invitation only. For more information, see the following: - Supported Regions - Supported cache node types	February 12, 2018
Support for EU (Paris).	ElastiCache added support for the EU (Paris) Region. For more information, see the following: - Supported Regions - Supported cache node types	December 18, 2017
Support for China (Ningxia) Region	Amazon ElastiCache added support for China (Ningxia) Region. For more information, see the following: - Supported Regions - Supported cache node types	December 11, 2017
Support for Service Linked Roles	This release of ElastiCache added support for Service Linked Roles (SLR). For more information, see the following: - Using Service-Linked Roles for Amazon ElastiCache - Set up your permissions (new ElastiCache users only)	December 7, 2017

Change	Description	Date Changed
Support for R4 node types	This release of ElastiCache added support R4 node types in all AWS Regions supported by ElastiCache. You can purchase R4 node types as On-Demand or as Reserved Cache Nodes. For more information, see the following: • Supported cache node types • Memcached node-type specific parameters	November 20, 2017
Connection patterns topic	ElastiCache documentation adds a topic covering various patterns for accessing an ElastiCache cluster in an Amazon VPC. For more information, see Access Patterns for Accessing an ElastiCache Cache in an Amazon VPC in the *ElastiCache User Guide*.	April 24, 2017
Support for Memcached 1.4.34	ElastiCache adds support Memcached version 1.4.34, which incorporates a number of fixes to earlier Memcached versions. For more information, see Memcached 1.4.34 Release Notes at Memcached on GitHub.	April 10, 2017
Support for Memcached 1.4.33	ElastiCache adds support for Memcached version 1.4.33. For more information, see the following: • Memcached version 1.4.33 • Memcached 1.4.33 added parameters	December 20, 2016

Change	Description	Date Changed
Support for EU West (London) Region	ElastiCache adds support for EU (London) Region. Only node types T2 and M4 are currently supported. For more information, see the following: • Supported Regions • Supported cache node types	December 13, 2016
Support for Canada (Montreal) Region	ElastiCache adds support for the Canada (Montreal) Region. Only node type M4 and T2 are currently supported in this AWS Region. For more information, see the following: • Supported Regions • Supported cache node types	December 8, 2016
Support for M4 and R3 node types	ElastiCache adds support for R3 and M4 node types in South America (São Paulo) Region and M4 node types in China (Beijing) Region. For more information, see the following: • Supported Regions • Supported cache node types	November 1, 2016
US East 2 (Ohio) Region support	ElastiCache adds support for the US East (Ohio) Region (*us-east-2*) with M4, T2, and R3 node types. For more information, see the following: • Supported Regions • Supported cache node types	October 17, 2016

Change	Description	Date Changed
M4 node type support	ElastiCache adds support for the M4 family of node types in most AWS Regions supported by ElastiCache. You can purchase M4 node types as On-Demand or as Reserved Cache Nodes. For more information, see the following: • Supported cache node types • Memcached node-type specific parameters	August 3, 2016
Mumbai Region support	ElastiCache adds support for the Asia Pacific (Mumbai) Region. For more information, see the following: • Supported cache node types • Memcached node-type specific parameters	June 27, 2016
Support for R3 node types	ElastiCache adds support for R3 node types in the China (Beijing) Region and South America (São Paulo) Region. For more information, see Supported cache node types.	March 16, 2016
Accessing ElastiCache using a Lambda function	Added a tutorial on configuring a Lambda function to access ElastiCache in an Amazon VPC. For more information, see ElastiCache tutorials and videos.	February 12, 2016
Support for Asia Pacific (Seoul) Region	ElastiCache adds support for the Asia Pacific (Seoul) (*ap-northeast-2*) Region with t2, m3, and r3 node types.	January 6, 2016

Change	Description	Date Changed
Support for Memcached 1.4.28.	ElastiCache adds support for Memcached version 1.4.24 and Memcached improvements since version 1.4.14. This release adds support for least recently used (LRU) cache management as a background task, choice of *jenkins* or *murmur3* as your hashing algorithm, new commands, and miscellaneous bug fixes. For more information, see Memcached release notes .	August 27, 2015
Support for Memcached Auto Discovery using PHP 5.6	This release of Amazon ElastiCache adds support for Memcached Auto Discovery client for PHP version 5.6. For more information, see Compiling the source code for the ElastiCache cluster client for PHP.	July 29, 2015
New topic: Accessing ElastiCache from outside AWS	Added new topic on how to access ElastiCache resources from outside AWS. For more informati on, see Accessing ElastiCache from outside AWS.	July 9, 2015
Node replaceme nt messages added	ElastiCache adds three messages pertaining to scheduled node replacement, ElastiCache:Node ReplacementScheduled, ElastiCache:Node ReplacementRescheduled, and ElastiCache:Node ReplacementCanceled. For more information and actions you can take when a node is scheduled for replacement, see ElastiCache's Event Notifications and Amazon SNS.	June 11, 2015
Support for cost allocation tags	ElastiCache adds support for cost allocation tags. For more information, see Monitoring costs with cost allocation tags.	February 9, 2015

Change	Description	Date Changed
Support for AWS GovCloud (US-West) Region	ElastiCache adds support for the AWS GovCloud (US-West) (*us-gov-west-1*) Region.	January 29, 2015
Support for Europe (Frankfurt) Region	ElastiCache adds support for the Europe (Frankfurt) (*eu-central-1*) Region.	January 19, 2015
AWS CloudTrail logging of API calls supported	ElastiCache adds support for using AWS CloudTrail to log all ElastiCache API calls. For more information, see Logging Amazon ElastiCache API calls with AWS CloudTrail.	September 15, 2014
New instance sizes supported	ElastiCache adds support for additional General Purpose (T2) instances. For more information, see Configuring engine parameters using parameter groups.	September 11, 2014
Flexible node placement supported for Memcached	ElastiCache adds support for creating Memcached nodes across multiple Availability Zones.	July 23, 2014
New instance sizes supported	ElastiCache adds support for additional General Purpose (M3) instances and Memory Optimized (R3) instances. For more information, see Configuring engine parameters using parameter groups.	July 1, 2014
PHP auto discovery	Added support for PHP version 5.5 auto discovery.	May 13, 2014
Support for default Amazon Virtual Private Cloud (VPC)	In this release, ElastiCache is fully integrated with Amazon Virtual Private Cloud (VPC). For new customers, cache clusters are created in an Amazon VPC by default. For more information, see Amazon VPCs and ElastiCache security.	January 8, 2013

Change	Description	Date Changed
PHP support for cache node auto discovery	The initial release of cache node auto discovery provided support for Java programs. In this release, ElastiCache brings cache node auto discovery support to PHP.	January 2, 2013
Support for Amazon Virtual Private Cloud (VPC)	In this release, ElastiCache clusters can be launched in Amazon Virtual Private Cloud (VPC). By default, new customers' cache clusters are created in an Amazon VPC automatically; existing customers can migrate to Amazon VPC at their own pace. For more information, see Amazon VPCs and ElastiCache security.	December 20, 2012
Cache node auto discovery and new cache engine version	ElastiCache provides cache node auto discovery—the ability for client programs to automatically determine all of the cache nodes in a cluster, a nd to initiate and maintain connections to all of these nodes. This release also offers a new cache engine version: Memcached version 1.4.14. This new cache engine provides enhanced slab rebalancing capability, significant performance and scalabili ty improvements, and several bug fixes. There are several new cache parameters that can be configured. For more information, see Configuri ng engine parameters using parameter groups.	November 28, 2012
New cache node types	This release provides four additional cache node types.	November 13, 2012
Reserved cache nodes	This release adds support for reserved cache nodes.	April 5, 2012
New guide	This is the first release of *Amazon ElastiCache User Guide*.	August 22, 2011

AWS Glossary

For the latest AWS terminology, see the AWS glossary in the *AWS Glossary Reference*.