
Amazon ElastiCache

ElastiCache for Memcached User Guide

API Version 2015-02-02



Amazon ElastiCache: ElastiCache for Memcached User Guide

Copyright © 2022 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

What is ElastiCache for Memcached?	1
Common ElastiCache Use Cases and How ElastiCache Can Help	2
In-Memory Data Store	2
ElastiCache Customer Testimonials	3
ElastiCache for Memcached resources	4
ElastiCache for Memcached components and features	6
Nodes	6
Clusters	7
AWS Regions and availability zones	8
Endpoints	8
Parameter groups	8
Security	8
Security groups	9
Subnet groups	9
Events	9
Tools for managing your implementation	10
Using the AWS Management Console	10
Using the AWS CLI	10
Using the AWS SDK	10
Using the ElastiCache API	10
See also	10
Comparing Memcached and Redis	11
Getting started with ElastiCache for Memcached	15
Setting up	15
Create your AWS account	15
Getting an AWS Access Key	16
Configuring Your Credentials	16
Downloading and Configuring the AWS CLI	17
Set up your permissions (new ElastiCache users only)	17
Create and connect to a Memcached cluster	17
Step 1: Determine your cluster's requirements	17
Step 2: Create a cluster	18
Step 3: Authorize access to the cluster	21
Step 4: Connect to the cluster's node	22
Step 5: Delete your cluster (avoid unnecessary charges)	23
Tutorial: Configuring a Lambda function to access amazon ElastiCache in an Amazon VPC	23
Step 1: Create an ElastiCache cluster	24
Step 2: Create a Lambda function	24
Step 3: Test the Lambda function	27
Tutorials and videos	28
Videos	29
Where do I go from here?	33
Choosing regions and availability zones	34
Availability Zone considerations	34
Supported regions & endpoints	36
Locating your nodes	39
Using Local zones	39
Enabling a local zone	39
Using Outposts	40
Using Outposts with the Memcached console	40
Using Outposts with the AWS CLI	41
Managing nodes	43
Connecting to nodes	44
Supported node types	46

Supported node types by AWS Region	48
Replacing nodes	49
Reserved nodes	51
Managing costs with reserved nodes	51
Migrating previous generation nodes	59
Migrating nodes on a Memcached cluster	59
Managing clusters	61
Choosing a network type	62
Configuring subnets for network type	62
Using dual-stack	62
TLS enabled dual stack ElastiCache clusters	62
Using the AWS Management Console	62
Using the CLI	63
Preparing a cluster	63
Determining your requirements	63
Choosing your node size	65
Creating a cluster	68
Creating a cluster (CLI)	68
Creating a cluster (API)	69
Viewing a cluster's details	70
Modifying a cluster	72
Rebooting a cluster	75
Adding nodes to a cluster	76
Removing nodes from a cluster	80
Canceling pending add or delete node operations	84
Deleting a cluster	85
Accessing your cluster	87
Determine the cluster's platform	87
Grant access to your cluster	90
Finding connection endpoints	96
Automatically identify nodes in your cluster	100
Benefits of Auto Discovery	102
How Auto Discovery Works	103
Using Auto Discovery	106
Connecting to Cache Nodes Manually	111
Adding Auto Discovery To Your Client Library	112
Auto discovery clients	113
Managing your ElastiCache for Memcached implementation	130
Engine versions and upgrading	130
Supported Memcached versions	131
Upgrading engine versions	134
Tagging your ElastiCache resources	134
Monitoring costs with tags	139
Managing tags using the AWS CLI	140
Managing tags using the ElastiCache API	142
Caching strategies and best practices	144
Caching strategies	144
Configuring your ElastiCache client for efficient load balancing	149
IPv6 Client Examples	150
Managing maintenance	153
Scaling ElastiCache for Memcached clusters	154
Scaling Memcached Horizontally	155
Scaling Memcached vertically	155
Configuring engine parameters using parameter groups	157
Parameter management	158
Parameter group tiers	159
Creating a parameter group	159

Listing parameter groups by name	163
Listing a parameter group's values	167
Modifying a parameter group	168
Deleting a parameter group	171
Memcached specific parameters	173
Security	184
Data protection	184
Data security in Amazon ElastiCache	185
Internetwork traffic privacy	189
Amazon VPCs and ElastiCache security	189
Amazon ElastiCache API and interface VPC endpoints (AWS PrivateLink)	208
Subnets and subnet groups	210
Security groups: EC2-Classic	218
Identity and access management	225
Authentication	226
How Amazon ElastiCache works with IAM	227
Access control	231
Overview of managing access	233
Monitoring usage, events, and costs	258
Monitoring use	259
Logging Amazon ElastiCache API calls with AWS CloudTrail	267
Monitoring events	270
Compliance Validation	281
Self-Service Security Updates for Compliance	281
HIPAA Eligibility	282
Resilience	282
Mitigating Failures	283
Infrastructure security	284
Service updates	284
Managing the service updates	285
Quotas	287
Reference	288
Using the ElastiCache API	288
Using the query API	288
Available libraries	290
Troubleshooting applications	291
Set up the AWS CLI for ElastiCache	291
Prerequisites	292
Getting the command line tools	293
Setting up the tools	293
Providing credentials for the tools	294
Environmental variables	294
Error messages	295
Notifications	296
General ElastiCache notifications	296
ElastiCache for Memcached notifications	296
Documentation history	297
AWS glossary	305

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.

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 \(p. 11\)](#) in this guide.

Existing applications that use Memcached can use ElastiCache with almost no modification. Your applications simply need information about the host names and port numbers of the ElastiCache nodes that you have deployed. The ElastiCache Auto Discovery feature for Memcached lets your applications identify all of the nodes in a cache cluster and connect to them. This means that you don't have to maintain a list of available host names and port numbers. In this way, your applications are effectively insulated from changes to node membership in a cluster.

ElastiCache for Memcached has multiple features to enhance reliability for critical production deployments:

- Automatic detection and recovery from cache node failures.
- Automatic discovery of nodes within a cluster enabled for automatic discovery, so that no changes need to be made to your application when you add or remove nodes.
- Flexible Availability Zone placement of nodes and clusters.
- Integration with other AWS services such as Amazon EC2, Amazon CloudWatch, AWS CloudTrail, and Amazon SNS to provide a secure, high-performance, managed in-memory caching solution.

Topics

- [Common ElastiCache Use Cases and How ElastiCache Can Help \(p. 2\)](#)
- [ElastiCache for Memcached resources \(p. 4\)](#)
- [ElastiCache for Memcached components and features \(p. 6\)](#)
- [Tools for managing your implementation \(p. 10\)](#)

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 [ElastiCache Videos \(p. 29\)](#) 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 \(p. 29\)](#) 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 \(p. 15\)](#) section includes an example 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.

After you complete the preceding sections, read these sections:

- [Choosing your Memcached node size \(p. 65\)](#)

You want your nodes to be large enough to accommodate all the data you want to cache. At the same time, you don't want to pay for more cache than you need. You can use this topic to help select the best node size.

- [Caching strategies and best practices \(p. 144\)](#)

Identify and address issues that can impact the efficiency of your cluster.

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 \(p. 288\)](#)

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.

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 \(p. 6\)](#)
- [ElastiCache for Memcached clusters \(p. 7\)](#)
- [AWS Regions and availability zones \(p. 8\)](#)
- [ElastiCache for Memcached endpoints \(p. 8\)](#)
- [ElastiCache parameter groups \(p. 8\)](#)
- [ElastiCache security \(p. 8\)](#)
- [ElastiCache security groups \(p. 9\)](#)
- [ElastiCache subnet groups \(p. 9\)](#)
- [ElastiCache for Memcached events \(p. 9\)](#)

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 clusters \(p. 154\)](#).

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 \(p. 46\)](#).

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 \(p. 51\)](#).

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 on Auto Discovery, see [Automatically identify nodes in your cluster \(p. 100\)](#).

For more information on nodes, see [Managing nodes \(p. 43\)](#).

ElastiCache for Memcached clusters

A Memcached *cluster* is a logical grouping of one or more [ElastiCache nodes \(p. 6\)](#). 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 \(p. 61\)](#) and [Managing nodes \(p. 43\)](#)

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 \(p. 283\)](#)

Information about improving the fault tolerance of your clusters.

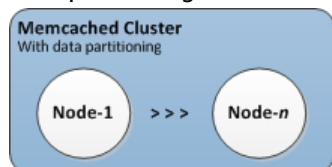
Typical cluster configurations

Memcached supports up to 300 nodes per customer for each AWS Region with each cluster having 1–40 nodes. You partition your data across the nodes in a Memcached cluster.

When you run the Memcached engine, clusters can be made up of 1–40 nodes. You partition your database across the nodes. Your application reads and writes to each node's endpoint. For more information, see [Automatically identify nodes in your cluster \(p. 100\)](#).

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 \(p. 283\)](#).

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 \(p. 149\)](#). In the following diagram, you can see examples of single node and multiple node Memcached clusters.

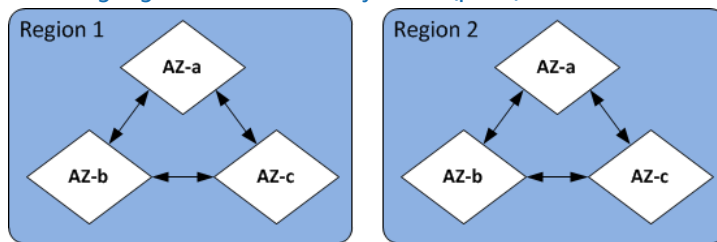


AWS Regions and availability zones

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 \(p. 34\)](#).



For information on AWS Regions supported by ElastiCache and their endpoints, see [Supported regions & endpoints \(p. 36\)](#).

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 [Automatically identify nodes in your cluster \(p. 100\)](#).

For more information, see [Finding connection endpoints \(p. 96\)](#).

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 \(p. 157\)](#).

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 security groups

Note

ElastiCache security groups are only applicable to clusters that are not running in an Amazon Virtual Private Cloud (Amazon VPC) environment. If you are running your ElastiCache nodes in an Amazon VPC, you control access to your cache clusters with Amazon VPC security groups, which are different from ElastiCache security groups.

For more information on using ElastiCache in an Amazon VPC, see [Amazon VPCs and ElastiCache security](#) (p. 189).

ElastiCache allows you to control access to your clusters using security groups. A security group acts like a firewall, controlling network access to your cluster. By default, network access to your clusters is turned off. If you want your applications to access your cluster, you must explicitly enable access from hosts in specific Amazon EC2 security groups. After ingress rules are configured, the same rules apply to all clusters associated with that security group.

To allow network access to your cluster, first create a security group. Then use the [AuthorizeCacheSecurityGroupIngress](#) API action or the [authorize-cache-security-group-ingress](#) AWS CLI command to authorize the desired Amazon EC2 security group. Doing this in turn specifies the Amazon EC2 instances allowed. You can associate the security group with your cluster at the time of creation, or by using the ElastiCache management console or the [ModifyCacheCluster](#) or ([modify-cache-cluster](#)) AWS CLI for ElastiCache command.

Important

IP-range based access control is currently not enabled for clusters. All clients to a cluster must be within the Amazon EC2 network, and authorized via security groups as described previously.

For more information about security groups, see [Security groups: EC2-Classical](#) (p. 218).

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](#) (p. 189), [Step 3: Authorize access to the cluster](#) (p. 21), and [Subnets and subnet groups](#) (p. 210).

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 [Monitoring ElastiCache events](#) (p. 270).

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](#) (p. 288).

See also

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

- [Managing your ElastiCache for Memcached implementation](#) (p. 130)
- [Internetwork traffic privacy](#) (p. 189)
- [Monitoring usage, events, and costs](#) (p. 258)

Comparing Memcached and Redis

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 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 \(Enhanced\)](#).

• 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](#).
- HIPAA eligibility certification. For more information, see [HIPAA Eligibility for Amazon ElastiCache for Redis](#).

- **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	1.6.6	2.8.x and later	3.2.x 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	No	3.2.6, 4.0.10 and later	3.2.6, 4.0.10 and later
Data tiering	No	6.2 and later	6.2 and later
Compliance certifications			
Compliance Certification			
FedRAMP	Yes	3.2.6, 4.0.10 and later	3.2.6, 4.0.10 and later
HIPAA	Yes	3.2.6, 4.0.10 and later	3.2.6, 4.0.10 and later
PCI DSS	No	3.2.6, 4.0.10 and later	3.2.6, 4.0.10 and later
Multi-threaded	Yes	No	No
Node type upgrade	No	Yes	Yes
Engine upgrading	Yes	Yes	Yes
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	2.8.x - No 3.2.x and later - Yes	Yes
Notes:			
string, objects (like databases)			
* string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog			
string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog, geospatial indexes			

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 cluster using the ElastiCache console.

Topics

- [Setting up](#) (p. 15)
- [Create and connect to a Memcached cluster](#) (p. 17)
- [Tutorial: Configuring a Lambda function to access amazon ElastiCache in an Amazon VPC](#) (p. 23)
- [ElastiCache tutorials and videos](#) (p. 28)
- [Where do I go from here?](#) (p. 33)

Setting up

Following, you can find topics that describe the one-time actions you must take to start using ElastiCache.

Topics

- [Create your AWS account](#) (p. 15)
- [Getting an AWS Access Key](#) (p. 16)
- [Configuring Your Credentials](#) (p. 16)
- [Downloading and Configuring the AWS CLI](#) (p. 17)
- [Set up your permissions \(new ElastiCache users only\)](#) (p. 17)

Create your AWS account

To use Amazon ElastiCache, you must have an active AWS account and permissions to access ElastiCache and other AWS resources.

If you don't already have an AWS account, create one now. AWS accounts are free. You are not charged for signing up for an AWS service, only for using AWS services.

To create 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 an administrative user](#), and use only the root user to perform [tasks that require root user access](#).

Getting an AWS Access Key

Before you can access ElastiCache programmatically or through the AWS Command Line Interface (AWS CLI), you must have an AWS access key. You don't need an access key if you plan to use the ElastiCache console only. Access keys consist of an access key ID and secret access key, which are used to sign programmatic requests that you make to AWS. If you don't have access keys, you can create them from the AWS Management Console. As a best practice, do not use the AWS account root user access keys for any task where it's not required. Instead, create a new administrator IAM user with access keys for yourself. The only time that you can view or download the secret access key is when you create the keys. You cannot recover them later. However, you can create new access keys at any time. You must also have permissions to perform the required IAM actions. For more information, see [Permissions Required to Access IAM Resources](#) in the *IAM User Guide*.

To create access keys for an IAM user

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, choose **Users**.
3. Choose the name of the user whose access keys you want to create, and then choose the **Security credentials** tab.
4. In the **Access keys** section, choose **Create access key**.
5. To view the new access key pair, choose **Show**. You will not have access to the secret access key again after this dialog box closes. Your credentials will look something like this:
 - Access key ID: AKIAIOSFODNN7EXAMPLE
 - Secret access key: wJalrXUtnFEMI/K7MDENG/bPxRfCYEXAMPLEKEY
6. To download the key pair, choose **Download .csv file**. Store the keys in a secure location. You will not have access to the secret access key again after this dialog box closes.
7. Keep the keys confidential in order to protect your AWS account and never email them. Do not share them outside your organization, even if an inquiry appears to come from Amazon or Amazon.com. No one who legitimately represents Amazon will ever ask you for your secret key.
8. After you download the .csv file, choose **Close**. When you create an access key, the key pair is active by default, and you can use the pair right away.

Related topics:

- [What is IAM](#) in the *IAM User Guide*.
- [AWS Security Credentials](#) in *AWS General Reference*.

Configuring Your Credentials

Before you can access ElastiCache programmatically or through the AWS CLI, you must configure your credentials to enable authorization for your applications.

There are several ways to do this. For example, you can manually create the credentials file to store your access key ID and secret access key. You also can use the `aws configure` command of the AWS CLI to automatically create the file. Alternatively, you can use environment variables. For more information about configuring your credentials, see the programming-specific AWS SDK developer guide at [Tools to Build on AWS](#).

Downloading and Configuring the AWS CLI

The AWS CLI is available at <http://aws.amazon.com/cli>. It runs on Windows, MacOS and Linux. After you download the AWS CLI, follow these steps to install and configure it:

1. Go to the [AWS Command Line Interface User Guide](#).
2. Follow the instructions for [Installing the AWS CLI](#) and [Configuring the AWS CLI](#).

Set up your permissions (new ElastiCache users only)

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 \(predefined\) policies for Amazon ElastiCache \(p. 238\)](#)
- [Using Service-Linked Roles for Amazon ElastiCache \(p. 249\)](#)

Create and connect to a Memcached cluster

Now that you have your AWS account and permissions set up, you can try out ElastiCache for Memcached by creating a cluster. The following sections show you how to do this.

Topics

- [Step 1: Determine your cluster's requirements \(p. 17\)](#)
- [Step 2: Create a cluster \(p. 18\)](#)
- [Step 3: Authorize access to the cluster \(p. 21\)](#)
- [Step 4: Connect to the cluster's node \(p. 22\)](#)
- [Step 5: Delete your cluster \(avoid unnecessary charges\) \(p. 23\)](#)

Step 1: Determine your cluster's requirements

Before you create a Memcached cluster, you should always determine the requirements for the cluster so that when you create the cluster it will meet your business needs and not need to be redone. Because in this exercise we will largely accept default values for the cluster, we will dispense with determining requirements. For more information, see [Determining your requirements \(p. 63\)](#).

Step 2: Create a cluster

The cluster you're about to launch will be live, and not running in a sandbox. You incur the standard ElastiCache usage fees for the instance until you delete it. The total charges are minimal (typically less than a dollar) if you complete the exercise described here in one sitting and delete your cluster when you are finished. For more information about ElastiCache usage rates, see <https://aws.amazon.com/elasticache/>.

Important

Your cluster is launched in an Amazon VPC. Before you start creating your cluster, you need to create a subnet group. For more information, see [Creating a subnet group \(p. 211\)](#).

To create an ElastiCache for Memcached 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 you want to launch this cluster in.
3. Choose **Get started**.
4. Choose **Create VPC** and follow the steps outlined at [Creating a Virtual Private Cloud \(VPC\)](#).
5. On the ElastiCache dashboard page, choose **Create cluster** and then choose **Create Memcached cluster**.
6. Under **Location**:

AWS Cloud

1. Under **Cluster info**
 - a. Enter a value for **Name**.
 - b. (Optional) Enter a value for **Description**.
2. Under **Cluster settings**
 - a. For **Engine version**, choose an available version.
 - b. For **Port**, accept the default port, 11211. If you have a reason to use a different port, type in the port number.
 - c. For **Parameter groups**, choose the parameter group you want to use with this cluster, or choose "Create new" to create a new parameter group to use with this cluster. For this exercise, accept the *default* parameter group.

For more information, see [Creating a parameter group \(p. 159\)](#).
 - d. For **Node type**, choose the node type that you want to use for this cluster. For this exercise, you can accept the default node type or select another node type.

For more information, see [Choosing your Memcached node size \(p. 65\)](#).

To select another node type:

- i. Choose the down-arrow to the right of the default node type.
- ii. Choose the **Instance family** you want for the nodes in this cluster. Since this is just an exercise, to save costs, choose **t2**.
- iii. From the available node types, choose the box to the left of the node type you want for this cluster. Since this is just an exercise, to save costs, choose **cache.t2.small**.
- iv. Choose **Save**.
- e. From the **Number of nodes** list, choose the number of nodes (partitions) you want provisioned for this cluster.

f. Under **Connectivity**

- i. For **Network type**, choose the IP version(s) this cluster will support.
- ii. For **Subnet groups**, choose the subnet that you want to apply to this cluster. 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.

When creating a new subnet group, enter the **VPC ID** to which it belongs.

For more information, see:

- [Choosing a network type \(p. 62\)](#).
- [Create a subnet in your VPC](#).

If you are [Using local zones with ElastiCache \(p. 39\)](#), you must create or choose a subnet that is in the local zone.

For more information, see [Subnets and subnet groups \(p. 210\)](#).

g. For **Availability zone placements**, you have two options.

- **No preference** – ElastiCache chooses each node's Availability Zone for you.
- **Specify availability zones** – You specify the Availability Zone for each node.

For this exercise, choose **Specify availability zones** and then choose an Availability Zone for each node.

For more information, see [Choosing regions and availability zones \(p. 34\)](#).

3. Under **Advanced settings**

a. For **Security**:

- i. To encrypt your data, you have the option to enable **Encryption in-transit**. This enables encryption of data on the wire. For more information, see [Encryption in transit](#).
- ii. For **Selected security groups** list, choose the security groups that you want to use for this cluster or create a new one. For this exercise, accept the default security group.

For more information, see [Amazon VPCs and ElastiCache security \(p. 189\)](#).

- b. i. The **Maintenance window** is the time, generally an hour, each week where ElastiCache schedules system maintenance on your cluster. You have two options.
 - **No preference**—ElastiCache chooses the day of the week and time of day for the cluster's maintenance window.
 - **Specify maintenance window**—You choose the day of the week, start time, and duration for the cluster's maintenance window.
- ii. After the cluster is created, you can modify it to specify a different maintenance window. For more information, see [Managing maintenance \(p. 153\)](#).
- c. For **Tags** to help you manage your clusters and other ElastiCache resources, you can assign your own metadata to each resource in the form of tags. For more information, see [Tagging your ElastiCache resources](#).
- d. Choose **Next**.

- e. Review your configuration and when satisfied, choose **Create**.

On premises

To finish creating the cluster, follow the steps at [Using Outposts](#).

Step 3: Authorize access to the cluster

This section assumes that you are familiar with launching and connecting to Amazon EC2 instances. For more information, see the [Amazon EC2 Getting Started Guide](#).

All ElastiCache clusters are designed to be accessed from an Amazon EC2 instance. The most common scenario is to access an ElastiCache cluster from an Amazon EC2 instance in the same Amazon Virtual Private Cloud (Amazon VPC). This is the scenario covered in this topic. For information on accessing your ElastiCache cluster from a different Amazon VPC, a different region, or even your corporate network, see the following:

- [Access Patterns for Accessing an ElastiCache Cluster in an Amazon VPC \(p. 195\)](#)
- [Accessing ElastiCache resources from outside AWS \(p. 92\)](#)

By default, network access to your cluster is limited to the user account that was used to launch it. Before you can connect to a cluster from an EC2 instance, you must authorize the EC2 instance to access the cluster. The steps required depend upon whether you launched your cluster into EC2-VPC or EC2-Classic.

For the steps to authorize access to your cluster, see [Accessing your cluster \(p. 87\)](#).

Step 4: Connect to the cluster's node

Before you continue, be sure you have completed [Step 3: Authorize access to the cluster \(p. 21\)](#).

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](#).

An Amazon EC2 instance can connect to a cluster node only if you have authorized it to do so. For more information, see [Step 3: Authorize access to the cluster \(p. 21\)](#).

Find your node endpoints

When your cluster is in the *available* state and you've authorized access to it ([Step 3: Authorize access to the cluster \(p. 21\)](#)), you can log in to an Amazon EC2 instance and connect to the cluster. To do so, you must first determine the endpoint.

To find your endpoints, see the relevant topic from the following list. When you find the endpoint you need, copy it to your clipboard for use in the next section.

- [Finding connection endpoints \(p. 96\)](#)
- [Finding a Cluster's Endpoints \(Console\) \(p. 97\)](#)—You need the cluster's Configuration endpoint.
- [Finding Endpoints \(AWS CLI\) \(p. 98\)](#)
- [Finding Endpoints \(ElastiCache API\) \(p. 100\)](#)

Connect to a Memcached cluster

Once your cluster is in the *available* state and you've authorized access to it ([Step 3: Authorize access to the cluster \(p. 21\)](#)), you can log in to an Amazon EC2 instance and connect to the cluster.

For instructions on connecting to a Memcached cluster, see [Connecting to nodes \(p. 44\)](#).

Step 5: Delete your cluster (avoid unnecessary charges)

Important

It is almost always a good idea to delete clusters that you are not actively using. Until a cluster's status is *deleted*, you continue to incur charges for it.

Before you continue, be sure you have completed at least as far as [Step 2: Create a cluster \(p. 18\)](#).

To delete a Memcached cluster

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at <https://console.aws.amazon.com/elasticache/>.
2. To see a list of all your clusters running Memcached, in the navigation pane, choose Memcached.
3. To select the cluster to delete, select the cluster's name from the list of clusters.

Tip

You can only delete one cluster at a time from the ElastiCache console. Selecting multiple clusters disables the delete operation. To delete multiple clusters, repeat this process for each cluster.

4. For **Actions**, choose **Delete**.
5. In the **Delete Cluster** confirmation screen, choose **Delete** to delete the cluster, or **Cancel** to keep the cluster.

If you choose **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.

Now you have successfully launched, authorized access to, connected to, viewed, and deleted your first ElastiCache for Memcached cluster.

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

In this tutorial, you do the following:

- Create an Amazon ElastiCache cluster in your default Amazon Virtual Private Cloud (Amazon VPC) in the us-east-1 region.
- Create a Lambda function to access the ElastiCache cluster. 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 cluster 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

- [Step 1: Create an ElastiCache cluster \(p. 24\)](#)
- [Step 2: Create a Lambda function \(p. 24\)](#)
- [Step 3: Test the Lambda function \(p. 27\)](#)

Get Started

[Step 1: Create an ElastiCache cluster \(p. 24\)](#)

Step 1: Create an ElastiCache cluster

In the step you create an Amazon ElastiCache cluster 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 clusters using the ElastiCache console or API, see [Creating a cluster \(p. 68\)](#) in the *ElastiCache User Guide*.

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

For Linux, macOS, or Unix:

```
aws elasticache create-cache-cluster \  
  --cache-cluster-id CacheClusterForLambda \  
  --cache-node-type cache.m3.medium \  
  --engine memcached \  
  --num-cache-nodes 1
```

For Windows:

```
aws elasticache create-cache-cluster ^  
  --cache-cluster-id CacheClusterForLambda ^  
  --cache-node-type cache.m3.medium ^  
  --engine memcached ^  
  --num-cache-nodes 1
```

You can look up the default VPC security group in the VPC console under **Security Groups**. Your example Lambda function will add and retrieve an item from this cluster.

2. Write down the configuration endpoint for the cache cluster that you launched. You can get this from the ElastiCache console (see [Finding connection endpoints \(p. 96\)](#)). You will specify this value in your Lambda function code in the next section.

Next Step:

[Step 2: Create a Lambda function \(p. 24\)](#)

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 \(p. 25\)](#)

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 #ELC; cluster. Copy the code and save it into a file called app.py.

```
from __future__ import print_function
import time
import uuid
import sys
import socket
import elasticache_auto_discovery
from pymemcache.client.hash import HashClient

# ElastiCache settings
elasticache_config_endpoint = "your-elasticache-cluster-endpoint:port"
nodes = elasticache_auto_discovery.discover(elasticache_config_endpoint)
nodes = map(lambda x: (x[1], int(x[2])), nodes)
memcache_client = HashClient(nodes)

#####
# This function puts into memcache and get from it.
# Memcached is hosted using elasticache
#####
def 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.
    memcache_client.set('uuid', uuid_in)

    # Get the item (UUID) from the cache.
    uuid_out = memcache_client.get('uuid')

    # Print the results
    if uuid_out == uuid_in:
        # this print should see the CloudWatch Logs and Lambda console.
        print "Success: Inserted: %s. Fetched %s from memcache." %(uuid_in, uuid_out)
    else:
        raise Exception("Bad value retrieved :(. Expected %s got %s." %(uuid_in,
        uuid_out))

    return "Fetched value from Memcached"
```

The preceding code depends upon the pymemcache and elasticache-auto-discovery libraries. Install these libraries using pip.

- **pymemcache**—The Lambda function code uses this library (see [pymemcache](#)) to create a HashClient object which sets and gets items from Memcached.
- **elasticache-auto-discovery**—The Lambda function uses this library (see [elasticache-auto-discovery](#)) to get the nodes in your Amazon ElastiCache cluster.

Save the preceding Python code in a file named `app.py`. Then zip all of these files into a file named `app.zip` to create your deployment package. For step-by-step instructions, see [Creating a Deployment Package \(Python\)](#).

Next Step

[Step 2.2: Create the IAM role \(execution role\) \(p. 26\)](#)

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 choose **Lambda** under **Common use cases**. This grants the AWS Lambda service permissions to assume the role. Choose **Next**.
 - Under **Add permissions**, enter **lambda-vpc-execution-role** and then 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. Write down 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 \(create the Lambda function\) \(p. 26\)](#)

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.

At the command prompt, run the following Lambda CLI `create-function` command using the **adminuser** profile.

You need to update the following `create-function` command by providing the `.zip` file path and the execution role ARN. The `--runtime` parameter value can be `python2.7`, `nodejs`, and `java8`, depending on the language you used to author your code.

For Linux, macOS, or Unix:

```
$ aws lambda create-function \
--function-name AccessMemCached \
--region us-east-1 \
--zip-file fileb://path-to/app.zip \
--role execution-role-arn \
--handler app.handler \
--runtime python3.8 \
--timeout 30 \
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-security-group-id \
--memory-size 1024
```

For Windows:

```
$ aws lambda create-function ^
--function-name AccessMemCached ^
--region us-east-1 ^
--zip-file fileb://path-to/app.zip ^
--role execution-role-arn ^
--handler app.handler ^
--runtime python3.8 ^
--timeout 30 ^
--vpc-config SubnetIds=comma-separated-vpc-subnet-ids,SecurityGroupIds=default-security-group-id ^
--memory-size 1024
```

You can find the subnet IDs and the default security group ID of your VPC from the VPC console.

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 \(p. 27\)](#)

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 \
--profile adminuser \
output.txt
```

For Windows:


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

2. Verify that the Lambda function executed successfully as follows:
 - Review the output.txt file.
 - Review the results in the AWS Lambda console.
 - Verify the results in CloudWatch Logs.

ElastiCache tutorials and videos

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

- [ElastiCache Videos \(p. 29\)](#)
- [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 \(p. 29\)](#)
- [Advanced Videos \(p. 30\)](#)

Introductory Videos

The following videos introduce you to Amazon ElastiCache.

Topics

- [AWS re:Invent 2020: What's new in Amazon ElastiCache \(p. 29\)](#)
- [AWS re:Invent 2019: What's new in Amazon ElastiCache \(p. 29\)](#)
- [AWS re:Invent 2017: What's new in Amazon ElastiCache \(p. 29\)](#)
- [DAT204—Building Scalable Applications on AWS NoSQL Services \(re:Invent 2015\) \(p. 29\)](#)
- [DAT207—Accelerating Application Performance with Amazon ElastiCache \(AWS re:Invent 2013\) \(p. 29\)](#)

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 \(re:Invent 2020\)](#) (p. 30)
- [Supercharge your real-time apps with Amazon ElastiCache \(re:Invent 2019\)](#) (p. 30)
- [Best practices: migrating Redis clusters from Amazon EC2 to ElastiCache \(re:Invent 2019\)](#) (p. 30)
- [Scaling a Fantasy Sports Platform with Amazon ElastiCache & Amazon Aurora STP11 \(re:Invent 2018\)](#) (p. 31)
- [Reliable & Scalable Redis in the Cloud with Amazon ElastiCache \(re:Invent 2018\)](#) (p. 31)
- [ElastiCache Deep Dive: Design Patterns for In-Memory Data Stores \(re:Invent 2018\)](#) (p. 31)
- [DAT305—Amazon ElastiCache Deep Dive \(re:Invent 2017\)](#) (p. 31)
- [DAT306—Amazon ElastiCache Deep Dive \(re:Invent 2016\)](#) (p. 31)
- [DAT407—Amazon ElastiCache Deep Dive \(re:Invent 2015\)](#) (p. 32)
- [SDD402—Amazon ElastiCache Deep Dive \(re:Invent 2014\)](#) (p. 32)
- [DAT307—Deep Dive into Amazon ElastiCache Architecture and Design Patterns \(re:Invent 2013\)](#) (p. 32)

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 \(AWS re:Invent 2013\).](#)

Where do I go from here?

Now that you have tried the Getting Started exercise, you can explore the following sections to learn more about ElastiCache and available tools:

- [Getting started with AWS](#)
- [Tools for Amazon Web Services](#)
- [AWS Command Line Interface](#)
- [Amazon ElastiCache API reference](#)

After you complete the Getting Started exercise, you can read these sections to learn more about ElastiCache administration:

- [Choosing your Memcached node size \(p. 65\)](#)

You want your cache to be large enough to accommodate all the data you want to cache. At the same time, you don't want to pay for more cache than you need. Use this topic to help you choose the best node size.

- [Caching strategies and best practices \(p. 144\)](#)

Identify and address issues that can affect the efficiency of your cluster.

Choosing regions and availability zones

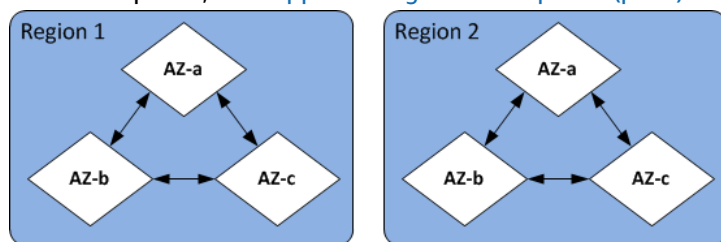
AWS Cloud computing resources are housed in highly available data center facilities. To provide additional scalability and reliability, these data center facilities are located in different physical locations. These locations are categorized by *regions* and *Availability Zones*.

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 \(p. 36\)](#).



Regions and Availability Zones

Topics

- [Availability Zone considerations \(p. 34\)](#)
- [Supported regions & endpoints \(p. 36\)](#)
- [Locating your nodes \(p. 39\)](#)
- [Using local zones with ElastiCache \(p. 39\)](#)
- [Using Outposts \(p. 40\)](#)

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.

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.

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

Each Region is designed to be completely isolated from the other Regions. Within each Region are multiple Availability Zones (AZ). By launching your nodes in different AZs you are able to achieve the greatest possible fault tolerance. For more information on Regions and Availability Zones, see [Choosing regions and availability zones \(p. 34\)](#) 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	
Canada (Central) Region ca-central-1	elasticache.ca-central-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	

Amazon ElastiCache ElastiCache
for Memcached User Guide
Supported regions & endpoints

Region Name/Region	Endpoint	Protocol	
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	
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	
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	

Region Name/Region	Endpoint	Protocol	
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	
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	
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 \(p. 48\)](#).

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. Testing has demonstrated that there is no significant latency difference between locating all nodes in one AZ or spreading them across multiple AZs.

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 \(p. 68\)](#)
- [Modifying an ElastiCache cluster \(p. 72\)](#)
- [Adding nodes to a cluster \(p. 76\)](#)

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 \(console\)](#) (p. 68).

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 compatibility:** 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 \  
  
```

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 \(p. 65\)](#).

Topics

- [Connecting to nodes \(p. 44\)](#)
- [Supported node types \(p. 46\)](#)
- [Replacing nodes \(p. 49\)](#)
- [ElastiCache reserved nodes \(p. 51\)](#)
- [Migrating previous generation nodes \(p. 59\)](#)

Some important operations involving nodes are the following:

- [Adding nodes to a cluster \(p. 76\)](#)
- [Removing nodes from a cluster \(p. 80\)](#)
- [Scaling ElastiCache for Memcached clusters \(p. 154\)](#)
- [Finding connection endpoints \(p. 96\)](#)
- [Automatically identify nodes in your cluster \(p. 100\)](#)

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 \(Console\) \(p. 97\)](#)
- [Finding Endpoints \(AWS CLI\) \(p. 98\)](#)
- [Finding Endpoints \(ElastiCache API\) \(p. 100\)](#)

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 \(p. 65\)](#).

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 \(p. 48\)](#).

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 \(p. 48\)](#).

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`

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
t12g .micro	288	2	10%	0.5	Up to 5 Gigabit
t24g .small	576	2	20%	1.37	Up to 5 Gigabit
t24g .medium	576	2	20%	3.09	Up to 5 Gigabit
t32g .micro	288	2	10%	0.5	Up to 5 Gigabit
t34g .small	576	2	20%	1.37	Up to 5 Gigabit
t34g .medium	576	2	20%	3.09	Up to 5 Gigabit
t2 .micro	144	1	10%	0.5	Low to moderate
t2 .small	288	1	20%	1.55	Low to moderate

CPU credits earned per hour	Maximum earned credits that can be accrued*	vCPUs	Baseline performance per vCPU	Memory (GiB)	Network performance
t2.medium	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.

Note

Supported engine versions vary by AWS Region. The latest engine versions are supported in all AWS Regions. To find the available engine versions in your AWS Region, see [Supported ElastiCache for Memcached versions \(p. 131\)](#).

Supported node types by AWS Region

See [Amazon ElastiCache pricing](#).

For a complete list of node types and specifications, see the following:

- [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 \(p. 72\)](#).

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 \(p. 153\)](#).

- **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 \(p. 80\)](#).
2. Add a new node to the cluster. For instructions, see [Adding nodes to a cluster \(p. 76\)](#).
3. If you are not using [Automatically identify nodes in your cluster \(p. 100\)](#) 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

- [Standard reserved node offerings \(p. 51\)](#)
- [Legacy reserved node offerings \(p. 52\)](#)
- [Getting info about reserved node offerings \(p. 54\)](#)
- [Purchasing a reserved node \(p. 56\)](#)
- [Getting info about your reserved nodes \(p. 58\)](#)

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 ElastiCache 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.
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.

Offering	Up-front cost	Usage fee	Advantage
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 \(Console\) \(p. 54\)](#)
- [Getting info about reserved node offerings \(AWS CLI\) \(p. 54\)](#)
- [Getting info about reserved node offerings \(ElastiCache API\) \(p. 55\)](#)

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.m6g.large",
  "Duration": 94608000,
  "FixedPrice": 1758.0,
  "UsagePrice": 0.0,
  "ProductDescription": "memcached",
  "OfferingType": "All Upfront",
  "RecurringCharges": [
    {
```

```
        "RecurringChargeAmount": 0.0,  
        "RecurringChargeFrequency": "Hourly"  
      }  
    ],  
  },  
  {  
    "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",  
    "CacheNodeType": "cache.m6g.xlarge",  
    "Duration": 94608000,  
    "FixedPrice": 1870.0,  
    "UsagePrice": 0.0,  
    "ProductDescription": "memcached",  
    "OfferingType": "Partial Upfront",  
    "RecurringCharges": [  
      {  
        "RecurringChargeAmount": 0.072,  
        "RecurringChargeFrequency": "Hourly"  
      }  
    ],  
  },  
  {  
    "ReservedCacheNodesOfferingId": "0xxxxxxxx-xxeb-44ex-xx3c-xxxxxxxx072",  
    "CacheNodeType": "cache.r5.12xlarge",  
    "Duration": 31536000,  
    "FixedPrice": 0.0,  
    "UsagePrice": 0.0,  
    "ProductDescription": "memcached",  
    "OfferingType": "No Upfront",  
    "RecurringCharges": [  
      {  
        "RecurringChargeAmount": 3.538,  
        "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 \(Console\) \(p. 56\)](#)
- [Purchasing a reserved node \(AWS CLI\) \(p. 56\)](#)
- [Purchasing a reserved node \(ElastiCache API\) \(p. 57\)](#)

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.m1.small	2013-12-19T00:30:23.247Z	1y	455.00
USD	0.092 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 \(Console\) \(p. 58\)](#)
- [Getting info about your reserved nodes \(AWS CLI\) \(p. 58\)](#)
- [Getting info about your reserved nodes \(ElastiCache API\) \(p. 58\)](#)

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.m1.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. If you have existing clusters, you can continue to use them or create new clusters using the previous generation 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](#) (p. 46).

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 \(console\)](#) (p. 68).
- In your application, update the endpoints to the new cluster's endpoints. For more information, see [Finding a Cluster's Endpoints \(Console\)](#) (p. 97)
- Delete the old cluster. For more information, see [Deleting a cluster](#) (p. 85)

If nodes are running on EC2-Classic

We
are
retiring
EC2-
Classic
on
August
15,
2022.

We recommend that you migrate from EC2-Classic to a VPC. For more information, see [Migrating an EC2-Classic cluster into a VPC \(p. 201\)](#) and the [blog EC2-Classic Networking is Retiring – Here's How to Prepare](#).

If your Amazon EC2 nodes are running on EC2-Classic platform, connect to your Memcached cluster from the Amazon EC2 instance using [ClassicLink](#).

Follow these steps to verify if your EC2 is running in EC2-Classic platform (console):

1. Sign in to the AWS Management Console and open the Amazon EC2 console at <https://console.aws.amazon.com/ec2/>.
2. Choose **Instances** in the navigation pane.
3. Choose your instance and go to **Description**.
4. If **VPC ID** is blank, the instance is running on the EC2-Classic platform.

For information on migrating your EC2-Classic instance to a VPC, see [Migrating an EC2-Classic cluster into a VPC \(p. 201\)](#).

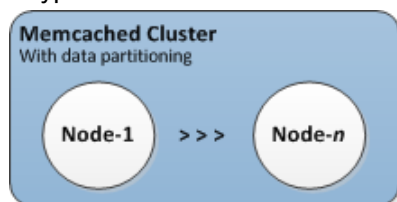
Managing clusters

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 40 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 [Accessing ElastiCache resources from outside AWS \(p. 92\)](#).

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 to the Memcached client. For more information, see [Automatically identify nodes in your cluster](#) (p. 100).

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](#) (p. 151).

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 \(console\) \(p. 68\)](#).

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
```

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 \(p. 63\)](#)
- [Choosing your node size \(p. 65\)](#)

Determining your requirements

Preparation

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

- Which node instance type do you need?

For guidance on choosing an instance node type, see [Choosing your Memcached node size \(p. 65\)](#).

- 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 \(p. 210\)](#).

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 [Accessing ElastiCache resources from outside AWS \(p. 92\)](#).

- Do you need to customize any parameter values?

If you do, create a custom parameter group. For more information, see [Creating a parameter group \(p. 159\)](#).

- Do you need to create your own *security group* or *VPC security group*?

For more information, see [Security groups: EC2-Classical \(p. 218\)](#) and [Security in Your VPC](#).

- How do you intend to implement fault tolerance?

For more information, see [Mitigating Failures \(p. 283\)](#).

Topics

- [Memory and processor requirements \(p. 64\)](#)
- [Memcached cluster configuration \(p. 64\)](#)
- [Scaling requirements \(p. 65\)](#)
- [Access requirements \(p. 65\)](#)
- [Region, Availability Zone and Local Zone requirements \(p. 65\)](#)

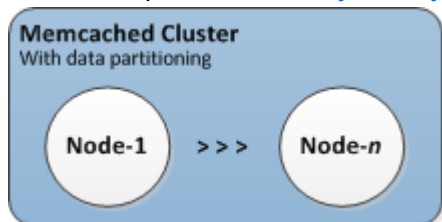
Memory and processor requirements

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 40 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 \(p. 100\)](#).



For more information, see [Managing clusters \(p. 61\)](#).

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 clusters \(p. 154\)](#) 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 user 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 [Step 3: Authorize access to the cluster \(p. 21\)](#) 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:

- [Choosing regions and availability zones \(p. 34\)](#)
- [Using local zones with ElastiCache \(p. 39\)](#)
- [Mitigating Failures \(p. 283\)](#)

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
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 \(p. 180\)](#)

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 \(p. 100\)](#).

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 \(p. 259\)](#). 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](#) (p. 100).

To create a Memcached cluster, follow the steps at [Create a cluster](#)

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 [Step 3: Authorize access to the cluster](#) (p. 21) and [Step 4: Connect to the cluster's node](#) (p. 22).

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](#) (p. 85).

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](#) (p. 85).

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 \(p. 85\)](#).

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, choose the **Nodes** tab, which displays information on node(s) status and endpoint.
6. To view metrics, choose the **Metrics** tab, which displays the relevant metrics for all nodes in the cluster. For more information, see [Monitoring use with CloudWatch metrics \(p. 259\)](#)
7. Choose the **Network and security** tab to view details on the cluster's network connectivity and subnet group configuration and the VPC security group. For more information, see [Subnets and subnet groups \(p. 210\)](#).
8. Choose the **Maintenance** tab to view details on the cluster's maintenance settings. For more information, see [Managing maintenance \(p. 153\)](#).
9. Choose the **Tags** tab to view details on any tags applied to cluster resources. For more information, see [Tagging your ElastiCache resources \(p. 134\)](#).

Viewing a cluster's details (AWS CLI)

The following code lists the details for *my-cluster*:

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

Replace *my-cluster* with the name of your cluster in a case where the cluster is created with 1 cache node and 0 shards using the `create-cache-cluster` command.

In a case where the cluster is created using the AWS Management Console (cluster node enabled or disabled with 1 or more shards), use the following command to describe the cluster's details (replace *my-cluster* with the name of the replication group (name of your cluster):

```
aws elasticache describe-replication-groups --replication-group-id my-cluster
```

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** column in the tables for [Memcached specific parameters \(p. 173\)](#) and [. For information on rebooting a cluster's nodes, see \[Rebooting a cluster \\(p. 75\\)\]\(#\).](#)

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
 - Network type

Note

If you are switching from IPv4 to IPv6, you must select or create subnet groups compatible with IPv6. For more information, see [Choosing a network type \(p. 62\)](#).

- 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 [Upgrading engine versions \(p. 134\)](#). 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 [Upgrading engine versions \(p. 134\)](#). 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 on the box to the left of the cluster's name.

The **Reboot** button becomes active.

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

5. Choose **Reboot**.

The reboot cluster confirmation screen appears.

6. To reboot the cluster, choose **Reboot**. The status of the cluster changes to *rebooting cluster nodes*.

To not reboot the cluster, choose **Cancel**.

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](#) (p. 149).

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 \(console\)](#) (p. 76)

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	<p>The new delete request, pending or immediate, replaces the pending delete request.</p> <p>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.</p>
Scenario 2	Delete	Create	<p>The new create request, pending or immediate, replaces the pending delete request.</p> <p>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.</p>
Scenario 3	Create	Delete	<p>The new delete request, pending or immediate, replaces the pending create request.</p> <p>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.</p>
Scenario 4	Create	Create	<p>The new create request is added to the pending create request.</p> <p>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.</p> <p>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.</p>

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 \(p. 81\)](#).
- `--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 \(p. 82\)](#)).
 - `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 \(p. 149\)](#).

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. 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.

Scenarios	Pending Operation	New Request	Results
			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.
Scenario 4	Create	Create	<p>The new create request is added to the pending create request.</p> <p>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.</p> <p>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.</p>

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 \(p. 70\)](#).
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",  
    "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.

Warning

When you delete an ElastiCache for Redis cluster, your manual snapshots are retained. You can also create a final snapshot before the cluster is deleted. Automatic cache snapshots are not retained.

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

- [Determine the cluster's platform \(p. 87\)](#)
- [Grant access to your cluster \(p. 90\)](#)

Determine the cluster's platform

Before you continue, determine whether you launched your cluster into EC2-VPC or EC2-Classic.

For more information, see [Detecting Your Supported Platforms and Whether You Have a Default VPC](#).

We
are
retiring
EC2-
Classic
on
August
15,
2022.
We
recommend
that
you
migrate
from
EC2-
Classic
to
a
VPC.
For
more
information,
see
[Migrating
an
EC2-
Classic
cluster
into
a
VPC \(p. 201\)](#)

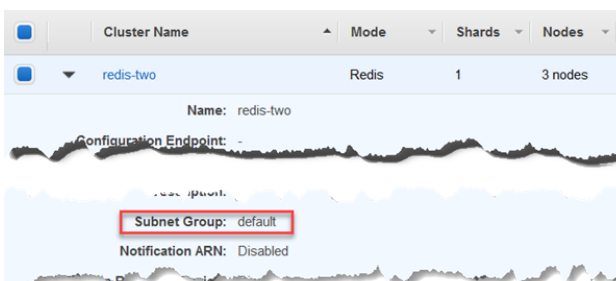
and
the
blog
[EC2-
Classic
Networking](#)
is
[Retiring](#)
–
[Here's
How
to
Prepare.](#)

Determining Your Cluster's Platform using the ElastiCache Console

The following procedure uses the ElastiCache console to determine whether you launched your cluster into EC2-VPC or EC2-Classical.

To determine a cluster's platform 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 your clusters running the Memcached engine, in the left navigation pane, choose **Memcached**.
3. In the list of clusters, expand the cluster you want to authorize access to by choosing the cluster name (not the button to its left).
4. Choose the **Network and security tab**.
5. Locate **Subnet group**:



- If the **Subnet group** has a name and a **VPC ID**, as shown here, you launched your cluster in EC2-VPC and should continue at [You launched your cluster into EC2-VPC \(p. 90\)](#).
- If there is a dash (-) instead of a **Subnet group** name, you launched your cluster in EC2-Classical and should continue at [You launched your cluster running in EC2-Classical \(p. 90\)](#).

For more information, see [Detecting Your Supported Platforms and Whether You Have a Default VPC](#).

Determining Your Clusters Platform using the AWS CLI

The following procedure uses the AWS CLI to determine whether you launched your cluster into EC2-VPC or EC2-Classical.

To determine a cluster's platform using the AWS CLI

1. Open a command window.
2. At the command prompt, run the following command.

For Linux, macOS, or Unix:

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

For Windows:

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

JSON output from this command will look something like this. Some of the output is omitted to save space.

```
{
  "CacheClusters": [
    {
      "Engine": "memcached",
      "AuthTokenEnabled": false,
      "CacheParameterGroup": {
        "CacheNodeIdsToReboot": [],
        "CacheParameterGroupName": "default.memcached1.4",
        "ParameterApplyStatus": "in-sync"
      },
      "CacheClusterId": "my-cluster-001",
      "CacheSecurityGroups": [],
      "NumCacheNodes": 1,
      "AtRestEncryptionEnabled": false,
      "CacheClusterCreateTime": "2018-01-16T20:09:34.449Z",
      "ReplicationGroupId": "my-cluster",
      "AutoMinorVersionUpgrade": true,
      "CacheClusterStatus": "available",
      "PreferredAvailabilityZone": "us-east-2a",
      "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
      "SecurityGroups": [
        {
          "Status": "active",
          "SecurityGroupId": "sg-e8c03081"
        }
      ],
      "TransitEncryptionEnabled": false,
      "CacheSubnetGroupName": "default",
      "EngineVersion": "6.0",
      "PendingModifiedValues": {},
      "PreferredMaintenanceWindow": "sat:05:30-sat:06:30",
      "CacheNodeType": "cache.t2.medium",
    }
  ]
}
```

- If there is a value for `CacheSubnetGroupName`, you launched your cluster in EC2-VPC and should continue at [You launched your cluster into EC2-VPC \(p. 90\)](#).
- If there is no value for `CacheSubnetGroupName`, you launched your cluster in EC2-Classic and should continue at [You launched your cluster running in EC2-Classic \(p. 90\)](#).

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 you 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.

You launched your cluster running in EC2-Classic

If you launched your cluster into EC2-Classic, to allow an Amazon EC2 instance to access your cluster you will need to grant the Amazon EC2 security group associated with the instance access to your cache security group.

To grant an Amazon EC2 security group access to a cluster

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 security groups, from the left navigation pane, choose **Security Groups**.

Important

If **Security Groups** is not listed in the navigation pane, you launched your cluster in EC2-VPC rather than EC2-Classical and should follow the instructions at [You launched your cluster into EC2-VPC \(p. 90\)](#).

3. Choose the box to the left of **default** security group.
4. From the list at the bottom of the screen, choose the **EC2 Security Group Name** you want to authorize.
5. To authorize access, choose **Add**.

Amazon EC2 instances that are associated with the security group are now authorized to connect to your ElastiCache cluster.

To revoke a security group's access, locate the security group in the list of authorized security groups, and then choose **Remove**.

For more information on ElastiCache Security Groups, see [Security groups: EC2-Classical \(p. 218\)](#).

Accessing ElastiCache resources from outside AWS

ElastiCache is a service designed to be used internally to your VPC. External access is discouraged due to the latency of Internet traffic and security concerns. However, if external access to ElastiCache is required for test or development purposes, it can be done through a VPN.

Using the AWS Client VPN, you allow external access to your ElastiCache nodes with the following benefits:

- Restricted access to approved users or authentication keys;
- Encrypted traffic between the VPN Client and the AWS VPN endpoint;
- Limited access to specific subnets or nodes;
- Easy revocation of access from users or authentication keys;
- Audit connections;

The following procedures demonstrate how to:

Topics

- [Create a certificate authority \(p. 92\)](#)
- [Configuring AWS client VPN components \(p. 93\)](#)
- [Configure the VPN client \(p. 95\)](#)

Create a certificate authority

It is possible to create a Certificate Authority (CA) using different techniques or tools. We suggest the easy-rsa utility, provided by the [OpenVPN](#) project. Regardless of the option you choose, make sure to keep the keys secure. The following procedure downloads the easy-rsa scripts, creates the Certificate Authority and the keys to authenticate the first VPN client:

- To create the initial certificates, open a terminal and do the following:
 - `git clone https://github.com/OpenVPN/easy-rsa`
 - `cd easy-rsa`
 - `./easyrsa3/easyrsa init-pki`
 - `./easyrsa3/easyrsa build-ca nopass`
 - `./easyrsa3/easyrsa build-server-full server nopass`
 - `./easyrsa3/easyrsa build-client-full client1.domain.tld nopass`

A **pki** subdirectory containing the certificates will be created under **easy-rsa**.

- Submit the server certificate to the AWS Certificate manager (ACM):
 - On the ACM console, select **Certificate Manager**.
 - Select **Import Certificate**.
 - Enter the public key certificate available in the `easy-rsa/pki/issued/server.crt` file in the **Certificate body** field.
 - Paste the private key available in the `easy-rsa/pki/private/server.key` in the **Certificate private key** field. Make sure to select all the lines between `BEGIN` AND `END PRIVATE KEY` (including the `BEGIN` and `END` lines).
 - Paste the CA public key available on the `easy-rsa/pki/ca.crt` file in the **Certificate chain** field.
 - Select **Review and import**.
- Select **Import**.

To submit the server's certificates to ACM using the AWS CLI, run the following command: `aws acm import-certificate --certificate fileb://easy-rsa/pki/issued/server.crt --private-key fileb://easy-rsa/pki/private/server.key --certificate-chain fileb://easy-rsa/pki/ca.crt --region region`

Note the Certificate ARN for future use.

Configuring AWS client VPN components

Using the AWS Console

On the AWS console, select **Services** and then **VPC**.

Under **Virtual Private Network**, select **Client VPN Endpoints** and do the following:

Configuring AWS Client VPN components

- Select **Create Client VPN Endpoint**.
- Specify the following options:
 - **Client IPv4 CIDR**: use a private network with a netmask of at least /22 range. Make sure that the selected subnet does not conflict with the VPC networks' addresses. Example: 10.0.0.0/22.
 - In **Server certificate ARN**, select the ARN of the certificate previously imported.
 - Select **Use mutual authentication**.
 - In **Client certificate ARN**, select the ARN of the certificate previously imported.
 - Select **Create Client VPN Endpoint**.

Using the AWS CLI

Run the following command:

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

Example output:

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

Associate the target networks to the VPN endpoint

- Select the new VPN endpoint, and then select the **Associations** tab.
- Select **Associate** and specify the following options.
 - **VPC**: Select the ElastiCache Cluster's VPC.
 - Select one of the ElastiCache cluster's networks. If in doubt, review the networks in the **Subnet Groups** on the ElastiCache dashboard.
 - Select **Associate**. If necessary, repeat the steps for the remaining networks.

Using the AWS CLI

Run the following command:

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

Example output:

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

Review the VPN security group

The VPN Endpoint will automatically adopt the VPC's default security group. Check the inbound and outbound rules and confirm if the security group allows the traffic from the VPN network (defined on the VPN Endpoint settings) to the ElastiCache networks on the service ports (by default, 6379 for Redis and 11211 for Memcached).

If you need to change the security group assigned to the VPN Endpoint, proceed as follows:

- Select the current security group.
- Select **Apply Security Group**.
- Select the new Security Group.

Using the AWS CLI

Run the following command:

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

Example output:

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

Note

The ElastiCache security group also needs to allow traffic coming from the VPN clients. The clients' addresses will be masked with the VPN Endpoint address, according to the VPC Network. Therefore, consider the VPC network (not the VPN Clients' network) when creating the inbound rule on the ElastiCache security group.

Authorize the VPN access to the destination networks

On the **Authorization** tab, select **Authorize Ingress** and specify the following:

- Destination network to enable access: Either use 0.0.0.0/0 to allow access to any network (including the Internet) or restrict the the ElastiCache networks/hosts.
- Under **Grant access to**, select **Allow access to all users**.
- Select **Add Authorization Rules**.

Using the AWS CLI

Run the following command:

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

Example output:

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

Allowing access to the Internet from the VPN clients

If you need to browse the Internet through the VPN, you need to create an additional route. Select the **Route Table** tab and then select **Create Route**:

- Route destination: 0.0.0.0/0
- **Target VPC Subnet ID**: Select one of the associated subnets with access to the Internet.
- Select **Create Route**.

Using the AWS CLI

Run the following command:

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

Example output:

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

Configure the VPN client

On the AWS Client VPN Dashboard, select the VPN endpoint recently created and select **Download Client Configuration**. Copy the configuration file, and the files `easy-rsa/pki/issued/client1.domain.tld.crt` and `easy-rsa/pki/private/client1.domain.tld.key`. Edit the configuration file and change or add the following parameters:

- `cert`: add a new line with the parameter `cert` pointing to the `client1.domain.tld.crt` file. Use the full path to the file. Example: `cert /home/user/.cert/client1.domain.tld.crt`
- `cert: key`: add a new line with the parameter `key` pointing to the `client1.domain.tld.key` file. Use the full path to the file. Example: `key /home/user/.cert/client1.domain.tld.key`

Establish the VPN connection with the command: `sudo openvpn --config downloaded-client-config.ovpn`

Revoking access

If you need to invalidate the access from a particular client key, the key needs to be revoked in the CA. Then submit the revocation list to AWS Client VPN.

Revoking the key with `easy-rsa`:

- `cd easy-rsa`
 - `./easyrsa3/easyrsa revoke client1.domain.tld`
 - Enter "yes" to continue, or any other input to abort.
- Continue with revocation: ``yes` ... * `./easyrsa3/easyrsa gen-crl``
- An updated CRL has been created. CRL file: `/home/user/easy-rsa/pki/crl.pem`

Importing the revocation list to the AWS Client VPN:

- On the AWS Management Console, select **Services** and then **VPC**.
- Select **Client VPN Endpoints**.

- Select the Client VPN Endpoint and then select **Actions** -> **Import Client Certificate CRL**.
- Paste the contents of the `crl.pem` file.

Using the AWS CLI

Run the following command:

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

Example output:

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

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

- **Memcached cluster**, If you use Automatic Discovery, 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. For additional information on Automatic Discovery, see [Automatically identify nodes in your cluster \(p. 100\)](#).

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 \(p. 44\)](#).

Configuration and node endpoints look very similar. The differences are highlighted with **bold** following.

```
myclustername. .... .cfg.usw2.cache.amazonaws.com:port    # configuration endpoint contains  
"cfg"  
myclustername. .... .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 \(AWS CLI\) \(p. 98\)](#)

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.1abc4d.0001.usw2.cache.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.1abc4d.0002.usw2.cache.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.1abc4d.0003.usw2.cache.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.1abc4d.cfg.usw2.cache.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 \(ElastiCache API\) \(p. 100\)](#)

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.

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 \(p. 62\)](#).

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 \(p. 102\)](#)
- [How Auto Discovery Works \(p. 103\)](#)
- [Using Auto Discovery \(p. 106\)](#)
- [Connecting to Cache Nodes Manually \(p. 111\)](#)
- [Adding Auto Discovery To Your Client Library \(p. 112\)](#)
- [ElastiCache clients with auto discovery \(p. 113\)](#)

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

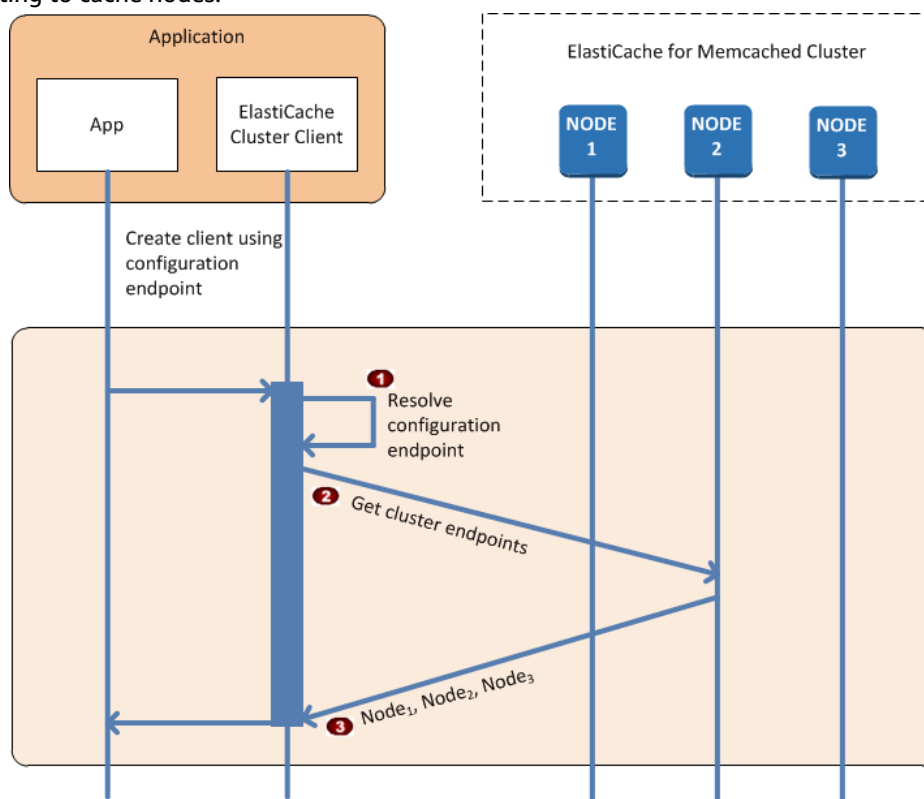
Topics

- [Connecting to Cache Nodes \(p. 103\)](#)
- [Normal Cluster Operations \(p. 104\)](#)
- [Other Operations \(p. 104\)](#)

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.



Process of Connecting to Cache Nodes

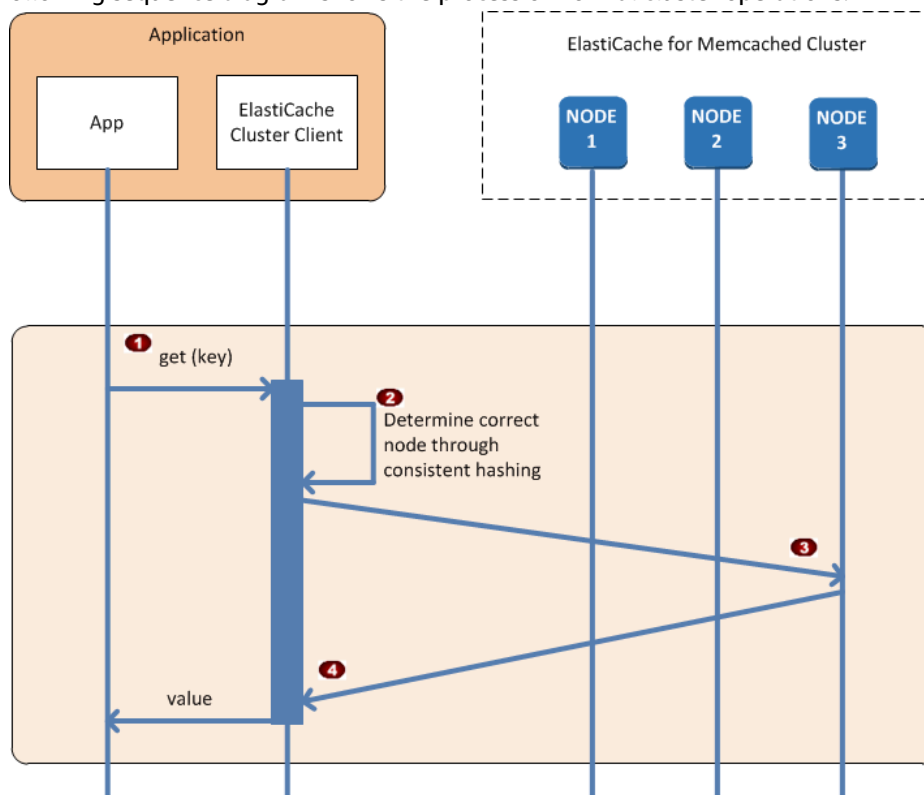
- 1 The application resolves the configuration endpoint's DNS name. Because the configuration endpoint maintains CNAME entries for all of the cache nodes, the DNS name resolves to one of the nodes; the client can then connect to that node.
- 2 The client requests the configuration information for all 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.
- 3 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.



Process of Normal Cluster Operations

1	The application issues a <i>get</i> request for a particular data item, identified by its key.
2	The client uses a hashing algorithm against the key to determine which cache node contains the data item.
3	The data item is requested from the appropriate node.
4	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 \(p. 106\)](#)
- [Step 2: Download the ElastiCache Cluster Client \(p. 107\)](#)
- [Step 3: Modify Your Application Program \(p. 108\)](#)

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 \(Console\) \(p. 97\)](#)

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.usel.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 \(p. 116\)](#). 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 \(p. 114\)](#).

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 \(p. 116\)](#)

```
<?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 [Creating a TLS Memcached client using PHP \(p. 188\)](#).

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 ElastiCache 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 \(p. 113\)](#) 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 <code>get</code> .	Yes
key	The key under which the cluster configuration is stored. For Auto Discovery, this key is named <code>cluster</code> .	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 \(p. 113\)](#)
- [Configuring ElastiCache clients \(p. 128\)](#)

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 \(p. 114\)](#)
- [Installing the ElastiCache cluster client for PHP \(p. 116\)](#)
- [Compiling the source code for the ElastiCache cluster client for PHP \(p. 125\)](#)

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 [Automatically identify nodes in your cluster \(p. 100\)](#). For sample .NET code to use the client, see [Using the ElastiCache Cluster Client for .NET \(p. 109\)](#).

Topics

- [Installing .NET \(p. 114\)](#)
- [Download the ElastiCache .NET cluster client for ElastiCache \(p. 114\)](#)
- [Install AWS assemblies with NuGet \(p. 114\)](#)

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 \(p. 100\)](#). For sample PHP code to use the client, see [Using the ElastiCache Cluster Client for PHP \(p. 108\)](#).

Topics

- [Downloading the installation package \(p. 116\)](#)
- [For users who already have php-memcached extension installed \(p. 117\)](#)
- [Installation steps for new users \(p. 117\)](#)
- [Removing the PHP cluster client \(p. 124\)](#)

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 \(p. 125\)](#).

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](#) (p. 124).
2. Install the new ElastiCache php-memcached extension as described previously in [Installation steps for new users](#) (p. 117).

Installation steps for new users

Topics

- [Installing PHP 7.x - 8.x for new users](#) (p. 117)
- [Installing PHP 5.x for new users](#) (p. 121)

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

Topics

- [To install PHP 7.x - 8.x on an Amazon Linux 2 AMI](#) (p. 117)
- [To install PHP 7.x - 8.x on an Amazon Linux 201609 AMI](#) (p. 119)
- [To install PHP 7.x - 8.x on an SUSE Linux 15 AMI](#) (p. 119)
- [To install PHP 7.x - 8.x on an Ubuntu 22.04 AMI](#) (p. 120)

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 OpenSSL1.1. 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 \(64-bit and 32-bit\)](#) (p. 121)
- [To install PHP 5 on a Red Hat Enterprise Linux 7.0 AMI \(64-bit and 32-bit\)](#) (p. 122)
- [To install PHP 5 on a Ubuntu server 14.04 LTS AMI \(64-bit and 32-bit\)](#) (p. 122)
- [To install PHP 5 for SUSE Linux enterprise server 11 AMI \(64-bit or 32-bit\)](#) (p. 123)
- [Other Linux distributions](#) (p. 123)

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](#) (p. 116).
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 \(p. 116\)](#).
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 \(p. 116\)](#).
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 \(p. 116\)](#).
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, `libsas12.so.3` has replaced `libsas12.so.2`. On those systems, when you load the ElastiCache cluster client,

it attempts and fails to find and load `libsas12.so.2`. To resolve this issue, create a symbolic link to `libsas12.so.3` so that when the client attempts to load `libsas12.so.2`, it is redirected to `libsas12.so.3`. The following code creates this symbolic link.

```
cd /usr/lib64
$ sudo ln libsas12.so.3 libsas12.so.2
```

Removing the PHP cluster client

Topics

- [Removing an earlier version of PHP 7 or higher \(p. 124\)](#)
- [Removing an earlier version of PHP 5 \(p. 124\)](#)

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 \(p. 117\)](#).
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 \(p. 125\)](#)
- [Compiling the ElastiCache Memcached auto discovery client for PHP \(p. 126\)](#)

Compiling the libmemcached library

Prerequisite 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/aws-labs/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 \(p. 126\)](#)
- [Compiling the ElastiCache Memcached client for PHP 5 \(p. 127\)](#)

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/aws-labs/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/aws-labs/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 \(p. 128\)](#)
- [Connecting for using auto discovery \(p. 129\)](#)
- [DNS names and underlying IP \(p. 129\)](#)

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=AWS4-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 \(p. 100\)](#).

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.

Managing your ElastiCache for Memcached implementation

In this section, you can find details about how to manage the various components of your ElastiCache implementation. These include tasks such as creating, updating, and deleting nodes or clusters, and many more.

Topics

- [Engine versions and upgrading \(p. 130\)](#)
- [Tagging your ElastiCache resources \(p. 134\)](#)
- [Caching strategies and best practices \(p. 144\)](#)
- [Managing maintenance \(p. 153\)](#)
- [Scaling ElastiCache for Memcached clusters \(p. 154\)](#)
- [Configuring engine parameters using parameter groups \(p. 157\)](#)

Engine versions and upgrading

This section covers the supported Memcached engine versions and how to upgrade.

Topics

- [Supported ElastiCache for Memcached versions \(p. 131\)](#)
- [Upgrading engine versions \(p. 134\)](#)

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 1.6.12 \(p. 131\)](#)
- [Memcached version 1.6.6 \(p. 131\)](#)
- [Memcached version 1.5.16 \(p. 131\)](#)
- [Memcached version 1.5.10 \(p. 131\)](#)
- [Memcached version 1.4.34 \(p. 132\)](#)
- [Memcached version 1.4.33 \(p. 132\)](#)
- [Memcached version 1.4.24 \(p. 132\)](#)
- [Memcached version 1.4.14 \(p. 132\)](#)
- [Memcached version 1.4.5 \(p. 133\)](#)

Memcached version 1.6.12

ElastiCache for Memcached adds support for Memcached version 1.6.12. It includes no new features, but does include 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 \(p. 173\)](#).

Memcached improvements added since ElastiCache for Memcached version 1.4.34 include the following:

- Cumulative fixes, such as ASCII multiget, 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 \(p. 175\)](#).

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 \(p. 176\)](#).
- 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.

Upgrading engine versions

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 an ElastiCache cluster \(p. 72\)](#).

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 \(p. 180\)](#).

Note the following:

- 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.
- Beginning with Redis 6.0, ElastiCache for Redis will offer a single version for each Redis OSS minor release, rather than offering multiple patch versions.
- 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 – [Using the AWS Management Console \(p. 72\)](#).
- To use the AWS CLI, see [Using the AWS CLI \(p. 73\)](#).
- To use the ElastiCache API, see [Using the ElastiCache API \(p. 73\)](#).

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 \(p. 241\)](#).

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:

- 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
ParameterGroup	Yes

Supports tags	Supports tagging on creation
Securitygroup	Yes
Subnetgroup	Yes
Cluster	Yes
Reserved-instance	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 \(p. 136\)](#).

For more information about tagging your resources for billing, see [Monitoring costs with cost allocation tags \(p. 139\)](#).

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 \(p. 135\)](#).

Tagging resources examples

- 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 redis \  
--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 more information, see [Using condition keys \(p. 242\)](#).

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

- [Listing tags using the AWS CLI \(p. 140\)](#)
- [Adding tags using the AWS CLI \(p. 141\)](#)
- [Modifying tags using the AWS CLI \(p. 142\)](#)
- [Removing tags using the AWS CLI \(p. 142\)](#)

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 \(p. 143\)](#)
- [Adding tags using the ElastiCache API \(p. 143\)](#)
- [Modifying tags using the ElastiCache API \(p. 143\)](#)

- [Removing tags using the ElastiCache API \(p. 144\)](#)

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>
```

Caching strategies and best practices

Following, you can find recommended best practices for Amazon ElastiCache. Following these improves your cluster's performance and reliability.

Topics

- [Caching strategies \(p. 144\)](#)
- [Configuring your ElastiCache client for efficient load balancing \(p. 149\)](#)
- [IPv6 Client Examples \(p. 150\)](#)

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 \(p. 144\)](#)
- [Write-through \(p. 146\)](#)
- [Adding TTL \(p. 147\)](#)
- [Related topics \(p. 148\)](#)

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 \(p. 146\)](#) and [Adding TTL \(p. 147\)](#) 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 \(p. 144\)](#) with write-through.

- Cache churn.

Most data is never read, which is a waste of resources. By [adding a time to live \(TTL\) value \(p. 147\)](#), 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 \(p. 2\)](#)
- [Choosing an Engine and Version](#)
- [Scaling ElastiCache for Memcached clusters \(p. 154\)](#)

Configuring your ElastiCache client for efficient load balancing

Note

This section applies to 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 - $\text{hash}(\text{key}) \bmod n$. 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

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

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:

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.toSet());

    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("xxx", 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 172.31.0.0/16, 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 172.31.0.0/16 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
```

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
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

Region Code	Region Name	Region Maintenance Window
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 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 \(p. 49\)](#)—Managing node replacement
- [Modifying an ElastiCache cluster \(p. 72\)](#)—Changing a cluster's maintenance window

Scaling ElastiCache for Memcached 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 (p. 76)
Scaling in	Removing nodes from a cluster (p. 80)
Changing node types	Scaling Memcached vertically (p. 155)

Memcached clusters are composed of 1 to 40 nodes. Scaling a Memcached cluster out and in is as easy as adding or removing nodes from the cluster.

If you need more than 40 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 \(p. 155\)](#)
- [Scaling Memcached vertically \(p. 155\)](#)

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 40 nodes. To horizontally scale your Memcached cluster, merely add or remove nodes.

If you need more than 40 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 \(p. 76\)](#)
- [Removing nodes from a cluster \(p. 80\)](#)

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 \(p. 149\)](#).

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 \(p. 100\)](#). 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 [Choosing your Memcached node size \(p. 65\)](#).

Topics

- [Scaling Memcached vertically \(Console\) \(p. 156\)](#)
- [Scaling Memcached vertically \(AWS CLI\) \(p. 156\)](#)
- [Scaling Memcached vertically \(ElastiCache API\) \(p. 156\)](#)

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 \(console\)](#) (p. 68).
2. In your application, update the endpoints to the new cluster's endpoints. For more information, see [Finding a Cluster's Endpoints \(Console\)](#) (p. 97).
3. Delete the old cluster. For more information, see [Using the AWS Management Console](#) (p. 85).

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 a cluster \(AWS CLI\)](#) (p. 68).
2. In your application, update the endpoints to the new cluster's endpoints. For more information, see [Finding Endpoints \(AWS CLI\)](#) (p. 98).
3. Delete the old cache cluster. For more information, see [Using the AWS CLI](#) (p. 85).

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 \(ElastiCache API\)](#) (p. 69).
2. In your application, update the endpoints to the new cache cluster's endpoints. For more information, see [Finding Endpoints \(ElastiCache API\)](#) (p. 100).
3. Delete the old cache cluster. For more information, see [Using the ElastiCache API](#) (p. 86).

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 \(p. 173\)](#).

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 \(p. 181\)](#).

Note

For a list of Memcached-specific parameters, see [Memcached Specific Parameters](#).

Topics

- [Parameter management \(p. 158\)](#)
- [Cache parameter group tiers \(p. 159\)](#)
- [Creating a parameter group \(p. 159\)](#)
- [Listing parameter groups by name \(p. 163\)](#)
- [Listing a parameter group's values \(p. 167\)](#)
- [Modifying a parameter group \(p. 168\)](#)
- [Deleting a parameter group \(p. 171\)](#)
- [Memcached specific parameters \(p. 173\)](#)

Parameter management

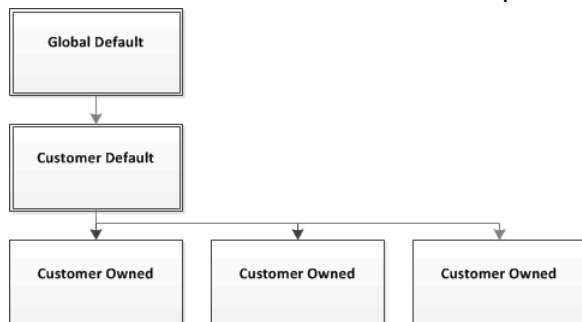
Parameters are grouped together into named parameter groups for easier parameter management. A parameter group represents a combination of specific values for the parameters that are passed to the engine software during startup. These values determine how the engine processes on each node behave at runtime. 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 \(p. 173\)](#). For information on rebooting a cluster's nodes, see [Rebooting a cluster \(p. 75\)](#).

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 \(p. 173\)](#).

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](#) (p. 168).

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](#) (p. 168).

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](#) (p. 168).

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>
      </CacheParameterGroup>
    </CacheParameterGroups>
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>
```

```
<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>
        </CacheNodeTypeSpecificValues>
      </CacheNodeTypeSpecificParameter>
    </CacheClusterClassSpecificParameters>
  </DescribeCacheParametersResult>
  <ResponseMetadata>
    <RequestId>6d355589-af49-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParametersResponse>

...output omitted...
```

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 \(p. 173\)](#).

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 \(p. 173\)](#). If changes to the parameter take place *After restart*, reboot every cluster that uses this parameter group. For more information, see [Rebooting a cluster \(p. 75\)](#).

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 \(p. 173\)](#)

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](#) (p. 173)

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 \(p. 159\)](#).

Topics

- [Memcached 1.6.6 added parameters \(p. 173\)](#)
- [Memcached 1.5.10 parameter changes \(p. 173\)](#)
- [Memcached 1.4.34 added parameters \(p. 174\)](#)
- [Memcached 1.4.33 added parameters \(p. 175\)](#)
- [Memcached 1.4.24 added parameters \(p. 176\)](#)
- [Memcached 1.4.14 added parameters \(p. 177\)](#)
- [Memcached 1.4.5 supported parameters \(p. 179\)](#)
- [Memcached connection overhead \(p. 180\)](#)
- [Memcached node-type specific parameters \(p. 181\)](#)

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, slab_automove, lru_crawler, lru_maintainer, maxconns_fast commands. No modern also sets the hash_algorithm to jenkins and allows inlining of ASCII VALUE. Applicable to memcached 1.5 and higher. To revert to modern, you must disable this parameter and re-launch, which will automatically enable slab_reassign, slab_automove, lru_crawler, lru_maintainer, and maxconns_fast. Note The default configuration value for this parameter has been changed from 0 to 1 as of August 20, 2021. The

Name	Details	Description
		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.
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, 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: Immediately 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.
idle_timeout	Default: 0 (disabled) Type: integer Modifiable: Yes Changes Take Effect: At Launch	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 Launch	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 Launch	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 losses.

Name	Details	Description
<code>worker_logbuf_size</code>	Default: 64 (KB) Type: integer Modifiable: Yes Changes Take Effect: At Launch	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.
<code>slab_chunk_max</code>	Default: 524288 (bytes) Type: integer Modifiable: Yes Changes Take Effect: At Launch	Specifies the maximum size of a slab. Setting smaller slab size uses memory more efficiently. Items larger than <code>slab_chunk_max</code> are split over multiple slabs.
<code>lru_crawler metadump [all 1 2 3]</code>	Default: disabled Type: boolean Modifiable: Yes Changes Take Effect: Immediately	if <code>lru_crawler</code> is enabled this command dumps all keys. <code>all 1 2 3</code> - 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
<code>disable_flush_all</code>	Default: 0 (disabled) Type: boolean Modifiable: Yes Changes Take Effect: At launch	Add parameter (-F) to disable <code>flush_all</code> . Useful if you never want to be able to run a full flush on production instances. Values: 0, 1 (user can do a <code>flush_all</code> when the value is 0).
<code>hash_algorithm</code>	Default: jenkins Type: string Modifiable: Yes Changes Take Effect: At launch	The hash algorithm to be used. Permitted values: murmur3 and jenkins.
<code>lru_crawler</code>	Default: 0 (disabled) Type: boolean Modifiable: Yes	Cleans slab classes of items that have expired. This is a low impact process that runs in the background.

Name	Details	Description
	<p>Changes Take Effect: After restart</p> <p>Note You can temporarily enable <code>lru_crawler</code> at runtime from the command line. For more information, see the Description column.</p>	<p>Constantly requires initiating a crawl using a manual command.</p> <p>To temporarily enable, run <code>lru_crawler enable</code> at the command line.</p> <p><code>lru_crawler 1,3,5</code> crawls slab classes 1, 3, and 5 looking for expired items to add to the freelist.</p> <p>Values: 0,1</p> <p>Note Enabling <code>lru_crawler</code> 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 (p. 168).</p>
<code>lru_maintainer</code>	<p>Default: 0 (disabled)</p> <p>Type: boolean</p> <p>Modifiable: Yes</p> <p>Changes Take Effect: At launch</p>	<p>A background thread that shuffles items between the LRUs as capacities are reached. Values: 0, 1.</p>
<code>expirezero_does_not_evict</code>	<p>Default: 0 (disabled)</p> <p>Type: boolean</p> <p>Modifiable: Yes</p> <p>Changes Take Effect: At launch</p>	<p>When used with <code>lru_maintainer</code>, makes items with an expiration time of 0 unevictable.</p> <p>Warning This can crowd out memory available for other evictable items.</p> <p>Can be set to disregard <code>lru_maintainer</code>.</p>

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	<p>Default: 16</p> <p>Maximum number of ElastiCache configuration entries.</p> <p>Type: integer</p> <p>Modifiable: No</p>
config_size_max	<p>Default: 65536</p> <p>Maximum size of the configuration entries, in bytes.</p> <p>Type: integer</p> <p>Modifiable: No</p>
hashpower_init	<p>Default: 16</p> <p>Initial size of the ElastiCache hash table, expressed as a power of two. The default is 16 (2^16), or 65536 keys.</p> <p>Type: integer</p> <p>Modifiable: No</p>
maxconns_fast	<p>Default: 0 (false)</p> <p>Controls the way in which new connections requests are handled when 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.</p> <p>Type: Boolean</p> <p>Modifiable: Yes</p> <p>Changes Take Effect: After restart</p>
slab_automove	<p>Default: 0 (false)</p> <p>Enables 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 is not recommended except for testing purposes.)</p> <p>Type: integer</p> <p>Modifiable: Yes</p> <p>Changes Take Effect: After restart</p>
slab_reassign	<p>Default: 0 (false)</p> <p>Enables or disable slab reassignment. If this parameter is set to 1, you can use the "slabs reassign" command to manually reassign memory.</p> <p>Type: Boolean</p> <p>Modifiable: Yes</p>

Name	Description
	Changes Take Effect: After restart

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_size	Default: 1024 Type: integer Modifiable: No	The backlog queue limit.
binding_protocol	Default: auto Type: string Modifiable: Yes Changes Take Effect: After restart	The binding protocol. Permissible values are: ascii and auto. For guidance on modifying the value of <code>binding_protocol</code> , see Modifying a parameter group (p. 168) .
cas_disabled	Default: 0 (false) Type: Boolean Modifiable: Yes Changes Take Effect: After restart	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 restart	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 restart	The growth factor that controls the size of each successive Memcached chunk; each chunk will be <code>chunk_size_growth_factor</code> times larger than the previous chunk.
error_on_memory_exhaustion	Default: 0 (false) Type: Boolean Modifiable: Yes Changes Take Effect: After restart	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_defrag	Default: 0 (false) Type: Boolean Modifiable: No	If 1 (true), ElastiCache will try to use large memory pages.
lock_down_page_defrag	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 restart	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 restart	If 1 (true), ElastiCache will maximize the core file limit.
memcached_connections_overhead	Default: 100 Type: integer Modifiable: Yes Changes Take Effect: After restart	The amount of memory to be reserved for Memcached connections and other miscellaneous overhead. For information about this parameter, see Memcached connection overhead (p. 180) .
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 your 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
cache.m2.xlarge	33800	2

Node type	max_cache_memory (in megabytes)	num_threads
cache.m2.2xlarge	30412	4
cache.m2.4xlarge	68000	16
cache.m3.medium	2850	1
cache.m3.large	6200	2
cache.m3.xlarge	13600	4
cache.m3.2xlarge	28600	8
cache.m4.large	6573	2
cache.m4.xlarge	11496	4
cache.m4.2xlarge	30412	8
cache.m4.4xlarge	62234	16
cache.m4.10xlarge	158355	40
cache.m5.large	6537	2
cache.m5.xlarge	13248	4
cache.m5.2xlarge	26671	8
cache.m5.4xlarge	53516	16
cache.m5.12xlarge	160900	48
cache.m5.24xlarge	321865	96
cache.m6g.large	6537	2
cache.m6g.xlarge	13248	4
cache.m6g.2xlarge	26671	8
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

Node type	max_cache_memory (in megabytes)	num_threads
cache.r4.large	12590	2
cache.r4.xlarge	25652	4
cache.r4.2xlarge	51686	8
cache.r4.4xlarge	103815	16
cache.r4.8xlarge	208144	32
cache.r4.16xlarge	416776	64
cache.r5.large	13387	2
cache.r5.xlarge	26953	4
cache.r5.2xlarge	54084	8
cache.r5.4xlarge	108347	16
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

Note

All T2 instances are created in an Amazon Virtual Private Cloud (Amazon VPC).

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 \(p. 184\)](#)
- [Internetwork traffic privacy \(p. 189\)](#)
- [Identity and access management in Amazon ElastiCache \(p. 225\)](#)
- [Monitoring usage, events, and costs \(p. 258\)](#)
- [Compliance Validation for Amazon ElastiCache \(p. 281\)](#)
- [Resilience in Amazon ElastiCache \(p. 282\)](#)
- [Infrastructure security in AWS ElastiCache \(p. 284\)](#)
- [Service updates in ElastiCache for Memcached \(p. 284\)](#)

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 user 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 \(p. 185\)](#)

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 optional encryption features for data on clusters 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 cluster and your application.

If you want to enable in-transit, you must meet the following conditions.

- Your cluster must be running Memcached 1.6.12 or later.
- Your cluster must be created in a VPC based on Amazon VPC.

Topics

- [ElastiCache in-transit encryption \(TLS\) \(p. 185\)](#)

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.

The parameters `TransitEncryptionEnabled` (CLI: `--transit-encryption-enabled`) are only available when using the `CreateCacheCluster` (CLI: `create-cache-cluster`) operation.

Topics

- [In-transit encryption overview \(p. 186\)](#)
- [In-transit encryption conditions \(p. 186\)](#)
- [In-transit encryption best practices \(p. 186\)](#)
- [Enabling in-transit encryption \(p. 186\)](#)
- [Connecting to nodes enabled with in-transit encryption using Openssl \(p. 187\)](#)
- [Creating a TLS Memcached client using Java \(p. 187\)](#)

- [Creating a TLS Memcached client using PHP \(p. 188\)](#)

In-transit encryption overview

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. 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 connections**—both the server and client connections are Secure Socket Layer (SSL) 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 implementation:

- In-transit encryption is supported on clusters running Memcached versions 1.6.12 and later.
- In-transit encryption supports TLS versions 1.2 and 1.3.
- In-transit encryption is supported only for clusters running in an Amazon VPC.
- In-transit encryption is only supported for clusters running the following node types.
 - R6g, R5, R4
 - M6g, M5, M4
 - T4g, T3

For more information, see [Supported node types \(p. 46\)](#).

- 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.
- To connect to an in-transit encryption enabled cluster, it must be enabled for transport layer security (TLS). To connect to a cluster that is not in-transit encryption enabled, the cluster cannot be TLS-enabled.

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 -crlf
```

An example of the output:

```
set hello 0 0 5
world
STORED
get hello
VALUE hello 0 5
world
END
```

For information on connecting to clusters without in-transit encryption enabled, see [Connecting to nodes \(p. 44\)](#).

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");
    }
}
```

For more information on using the Java client, see [Using the ElastiCache Cluster Client for Java \(p. 108\)](#).

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:

See [Using the ElastiCache Cluster Client for PHP \(p. 108\)](#) for more information about Auto Discovery and persistent-id.

```
<?php

/**
 * Sample PHP code to show how to create a TLS Memcached client. In this example we
 * will use the Amazon ElastiCache Auto Discovery 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->getLastError_message(), "\n";
    exit(1);
}

/* Set your TLS context configurations values.
 * See MemcachedTLSSContextConfig in memcached-api.php for all configurations */
$tls_config = new MemcachedTLSSContextConfig();
$tls_config->cert_file = '/path/to/memc.crt';
$tls_config->key_file = '/path/to/memc.key';
$tls_config->ca_cert_file = '/path/to/ca.crt';
$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->getLastError(), "\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->getLastError(), "\n";  
    exit(1);  
}
```

For more information on using the PHP client, see [Installing the ElastiCache cluster client for PHP \(p. 116\)](#).

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 \(p. 189\)](#) explains the type of security group you need for your installation.
- [Identity and access management in Amazon ElastiCache \(p. 225\)](#) 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 \(p. 190\)](#)
- [Access Patterns for Accessing an ElastiCache Cluster in an Amazon VPC \(p. 195\)](#)
- [Migrating an EC2-Classic cluster into a VPC \(p. 201\)](#)
- [Creating a Virtual Private Cloud \(VPC\) \(p. 203\)](#)
- [Connecting to a cache cluster running in an Amazon VPC \(p. 205\)](#)

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](#).

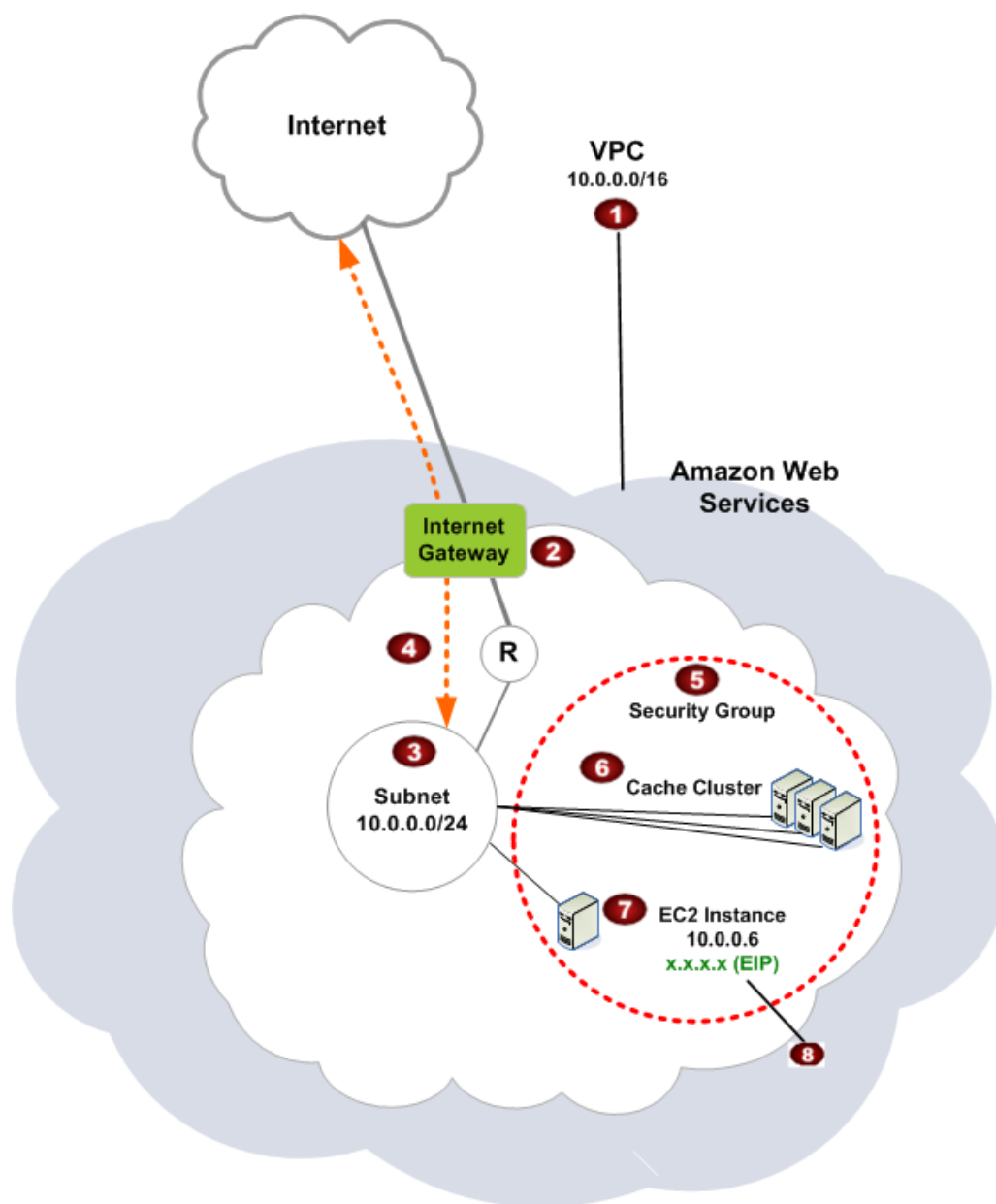
With Amazon Virtual Private Cloud, you can create a virtual network in the AWS cloud that closely resembles a traditional data center. You can configure your Amazon VPC, including selecting its IP address range, creating subnets, and configuring route tables, network gateways, and security settings.

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.

Why use the Amazon VPC instead of EC2 classic with your ElastiCache deployment?

We are retiring EC2-Classic on August 15, 2022. We recommend that you migrate from EC2-Classic to a VPC. For more information, see [Migrating an EC2-Classic cluster into a VPC \(p. 201\)](#) and the [blog EC2-Classic Networking is](#)

Retiring

–

Here's
How
to
Prepare.

Launching your instances into an Amazon VPC allows you to:

- Assign static private IP addresses to your instances that persist across starts and stops.
- Assign multiple IP addresses to your instances.
- Define network interfaces, and attach one or more network interfaces to your instances.
- Change security group membership for your instances while they're running.
- Control the outbound traffic from your instances (egress filtering) in addition to controlling the inbound traffic to them (ingress filtering).
- Add an additional layer of access control to your instances in the form of network access control lists (ACL).
- Run your instances on single-tenant hardware.

For a comparison of Amazon EC2 Classic, Default VPC, and Non-default VPC, see [Differences Between EC2-Classic and EC2-VPC](#).

The Amazon VPC must allow non-dedicated Amazon EC2 instances. You cannot use ElastiCache in an Amazon VPC that is configured for dedicated instance tenancy.

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 cache nodes.
- A cache security group must be defined for your Amazon VPC, or you can use the default provided.
- 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 if you reboot the cache node.

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)
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 Cluster in an Amazon VPC

Amazon ElastiCache supports the following scenarios for accessing a cluster in an Amazon VPC:

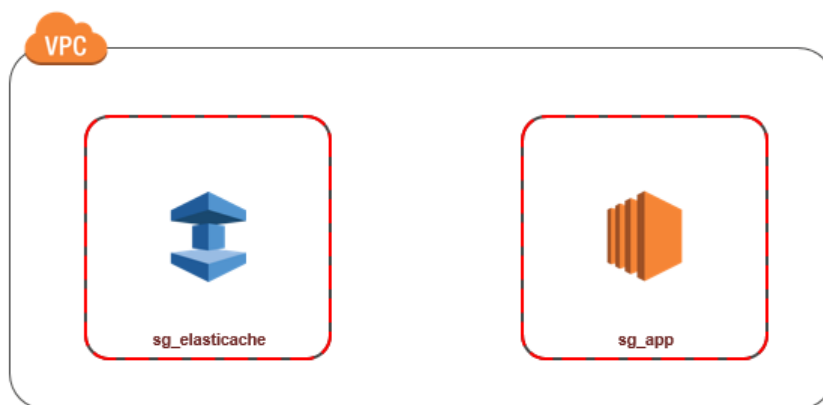
Contents

- [Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in the Same Amazon VPC \(p. 195\)](#)
- [Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs \(p. 196\)](#)
 - [Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs in the Same Region \(p. 197\)](#)
 - [Using Transit Gateway \(p. 198\)](#)
 - [Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs in Different Regions \(p. 198\)](#)
 - [Using Transit VPC \(p. 198\)](#)
- [Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center \(p. 199\)](#)
 - [Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center Using VPN Connectivity \(p. 200\)](#)
 - [Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center Using Direct Connect \(p. 200\)](#)

Accessing an ElastiCache Cluster 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 Cluster in the same VPC.

The following diagram illustrates this scenario



The simplest way to manage access between EC2 instances and clusters in the same VPC is to do the following:

1. Create a VPC security group for your cluster. This security group can be used to restrict access to the cluster instances. For example, you can create a custom rule for this security group that allows TCP access using the port you assigned to the cluster when you created it and an IP address you will use to access the cluster.

The default port for Memcached clusters 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 Cluster 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 clusters.

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 Cluster instances. 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 cluster.

The default port for Memcached clusters 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.

Type ⓘ	Protocol ⓘ	Port Range ⓘ	Source ⓘ
Custom TCP Rule ▼	TCP	6379	Custom ▼ sg_app

Add Rule

sg-99fc5c

Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs

When your Cluster is in a different VPC from the EC2 instance you are using to access it, there are several ways to access the cluster. If the Cluster and EC2 instance are in different VPCs but in the same region, you can use VPC peering. If the Cluster and the EC2 instance are in different regions, you can create VPN connectivity between regions.

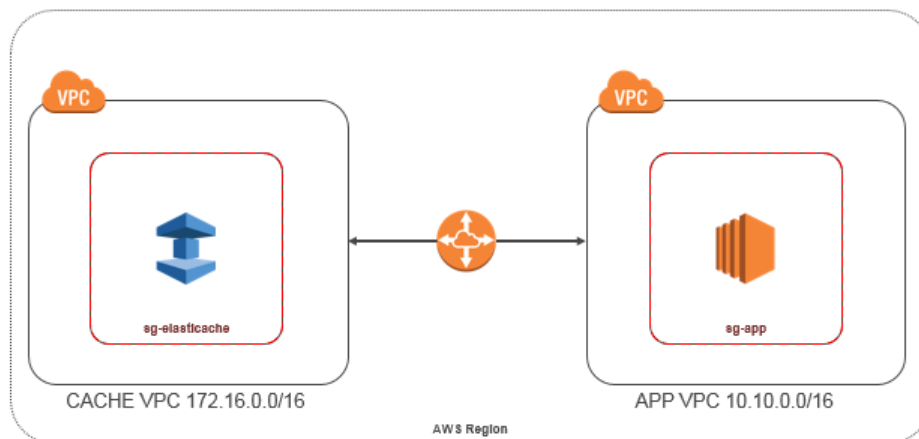
Topics

- [Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs in the Same Region \(p. 197\)](#)

- [Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs in Different Regions \(p. 198\)](#)

Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs in the Same Region

The following diagram illustrates accessing a cluster by an Amazon EC2 instance in a different Amazon VPC in the same region using an Amazon VPC peering connection.



Cluster 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 cluster 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	Destination	Target
172.16.0.0/16	local	10.10.0.0/16	local
10.10.0.0/16	pcx-a894f1c1	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 cluster to allow inbound connection from the Application security group in the peered VPC. For more information, see [Reference Peer VPC Security Groups](#).

Accessing a cluster 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.

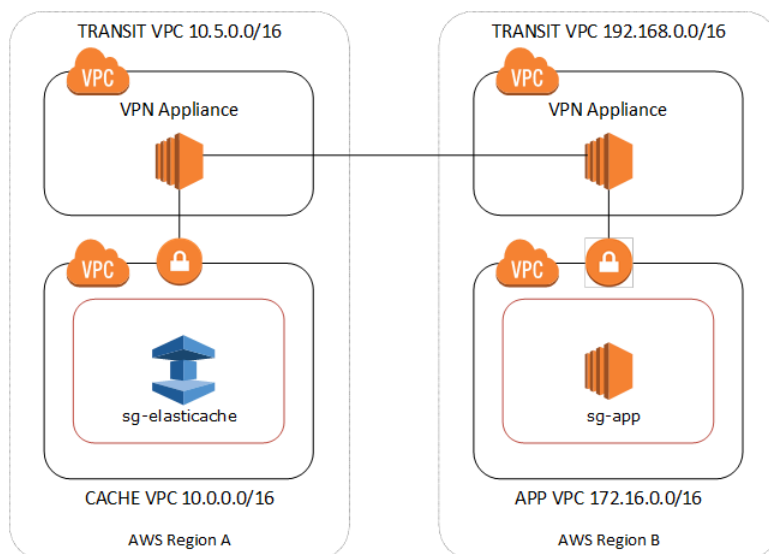
You can also create a peering connection attachment between transit gateways in different AWS Regions. This enables you to route traffic between the transit gateways' attachments across different Regions.

For more information, see [Transit gateways](#).

Accessing an ElastiCache Cluster 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 cluster in a “spoke” VPC within another region.

To access a cluster 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 cluster 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 cluster across regions will introduce networking latencies and additional cross-region data transfer costs.

Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center

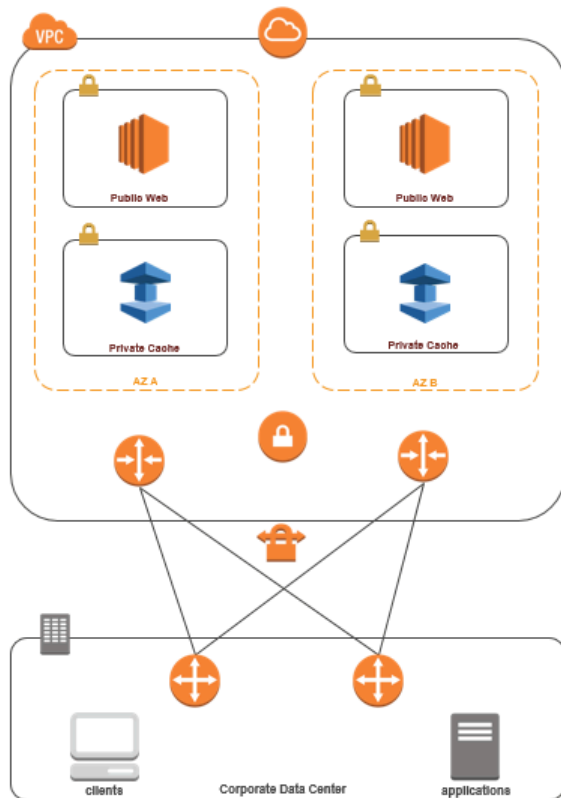
Another possible scenario is a Hybrid architecture where clients or applications in the customer's data center may need to access an ElastiCache Cluster in the VPC. This scenario is also supported providing there is connectivity between the customers' VPC and the data center either through VPN or Direct Connect.

Topics

- [Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center Using VPN Connectivity \(p. 200\)](#)
- [Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center Using Direct Connect \(p. 200\)](#)

Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center Using VPN Connectivity

The following diagram illustrates accessing an ElastiCache cluster from an application running in your corporate network using VPN connections.



Connecting to ElastiCache from your data center via a VPN

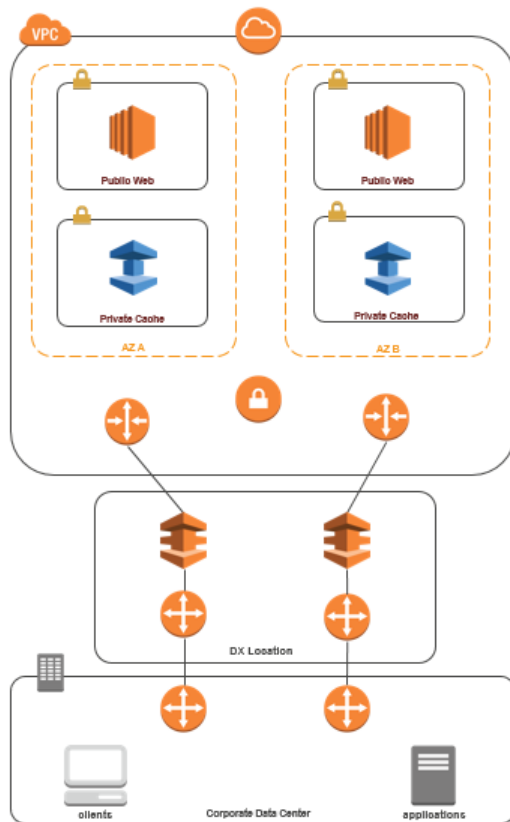
To access a cluster in a VPC from on-prem application over VPN connection

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 cluster 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 cluster to allow inbound connection from the on-premises application servers.

Accessing a cluster over a VPN connection will introduce networking latencies and additional data transfer costs.

Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center Using Direct Connect

The following diagram illustrates accessing an ElastiCache cluster 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 cluster 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 cluster to allow inbound connection from the on-premises application servers.

Accessing a cluster over DX connection may introduce networking latencies and additional data transfer charges.

Migrating an EC2-Classic cluster into a VPC

Note

We are retiring EC2-Classic on August 15, 2022.

Some legacy ElastiCache clusters on the EC2-Classic platform are not in an Amazon VPC. You can use the AWS Management Console to move it into an Amazon VPC using the following steps:

Migrating an EC2-Classic cluster into a VPC

Note

To determine if your cluster is on the EC2-Classic platform, see [Determine the cluster's platform \(p. 87\)](#).

1. You first need to create a Virtual Private Cloud (VPC). Follow the steps to create a VPC at [Creating a Virtual Private Cloud \(VPC\)](#).
2. Once you have created the VPC, follow the steps to create a subnet group at [Subnets and Subnet Groups](#).
3. Your VPC comes with a default security group. You can choose to use it or you can create one by following the steps at [Security groups for your VPC](#) in the Amazon VPC documentation.
4. Create a Memcached cluster. Under **Advanced Memcached settings, Subnet group**, choose the subnet group that belongs to the VPC you created previously. For more information on creating a Memcached cluster in a VPC, see [Create a Memcached cluster](#).
5. To link an EC2-Classic instance to an ElastiCache for Memcached cluster in a VPC, you must first enable the VPC for ClassicLink. You cannot enable a VPC for ClassicLink if the VPC has routing that conflicts with the EC2-Classic private IP address range. For more information, see [Routing for ClassicLink](#).

To enable a VPC for ClassicLink

- Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
 - In the navigation pane, choose **Your VPCs**.
 - Select the VPC.
 - Choose **Actions, Enable ClassicLink**.
 - When prompted for confirmation, choose **Enable ClassicLink**.
 - (Optional) If you want the public DNS hostname to resolve to the private IP address, enable ClassicLink DNS support for the VPC before you link any instances. For more information, see [Enable ClassicLink DNS support](#).
6. Finally, you connect to an ElastiCache cluster from your EC2 instance following the steps at [Connect to a cluster's node](#).

Note

Previous generation node types might not be supported on the VPC platform. For a full list of VPC supported node types, see [Supported node types by AWS Region \(p. 48\)](#).

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)

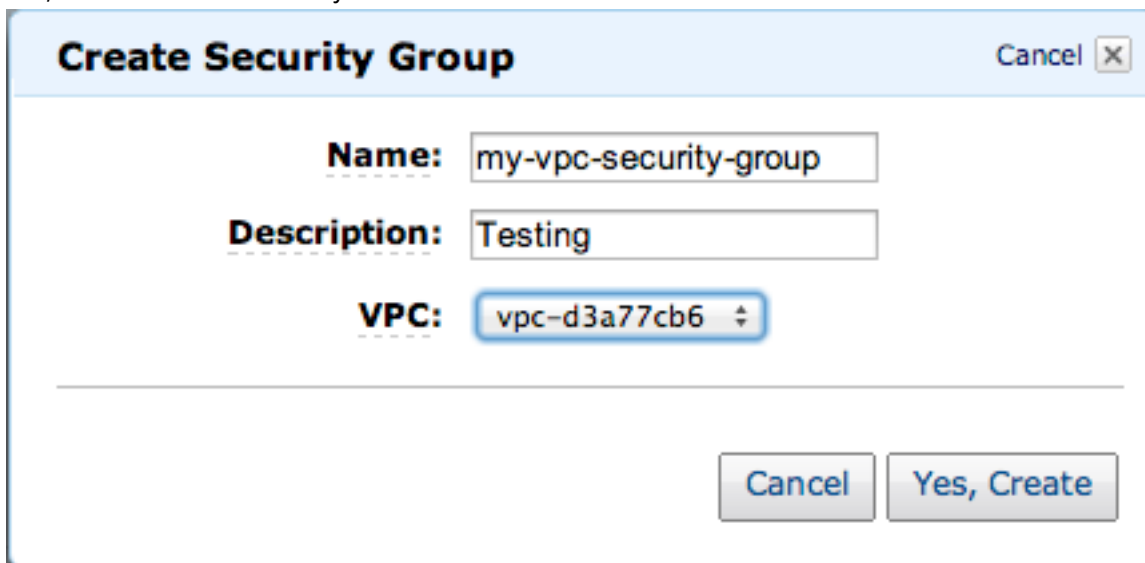
To create an ElastiCache cluster inside an Amazon Virtual Private Cloud

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 Cancel

Name: my-vpc-security-group

Description: Testing

VPC: vpc-d3a77cb6

Cancel Yes, Create

- 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 \(p. 211\)](#)
- [Creating a Memcached cluster \(console\) \(p. 68\)](#).

Connecting to a cache cluster 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 cluster that is running in the Amazon VPC.

Connecting to a cache cluster 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.

1. Choose AMI 2. Choose Instance Type **3. Configure Instance** 4. Add Storage 5. Tag Instance 6. Configure Security Groups

Step 3: Configure Instance Details

Configure the instance to suit your requirements. You can launch multiple instances from the same AMI, request pricing, assign an access management role to the instance, and more.

Number of instances ⓘ

Purchasing option ⓘ ☐ Request Spot Instances

Network ⓘ

Subnet ⓘ ⓘ
250 IP Addresses available

Public IP ⓘ ☐ Automatically assign a public IP address to your instance

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](#).



Choose the name of your Amazon VPC security group, and then choose **Review and Launch**.

- On the **Review Instance and Launch** page, choose **Launch**.
- 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](#).

- 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)

- Open the Amazon VPC console at <https://console.aws.amazon.com/vpc/>.
- In the navigation list, choose **Elastic IPs**.
- Choose **Allocate Elastic IP address**.
- In the **Allocate Elastic IP address** dialog box, accept the default **Network Border Group** and choose **Allocate**.
- Choose the Elastic IP address that you just allocated from the list and choose **Associate Address**.
- 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",
    "Principal": "AWS:",
    "Resource": "*"
  }, {
    "Principal": "AWS:123456789012",
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*"
  }]
}
```

```
"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 clusters running in an Amazon Virtual Private Cloud (VPC) environment.

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 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.

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 [Step 3: Authorize access to the cluster \(p. 21\)](#).

Topics

- [Creating a subnet group \(p. 211\)](#)
- [Assigning a subnet group to a cluster \(p. 214\)](#)
- [Modifying a subnet group \(p. 215\)](#)
- [Deleting a subnet group \(p. 217\)](#)

Creating a subnet group

A *cache subnet group* is a collection of subnets that you may want to designate for your cache clusters in a VPC. When launching a cache cluster 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 cluster.

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 the 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 [Choosing a network type \(p. 62\)](#). 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 **Yes, 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 the Amazon VPC that you created.
 - d. In the **Availability Zone** and **Subnet ID** lists, choose the Availability Zone or [Local Zone](#) and ID of your private subnet, and then choose **Add**.

Create Cache Subnet Group

To create a new Subnet Group give it a name, description, and select an existing VPC below. Once you select a VPC you will be able to add subnets related to that VPC.

Name* ⓘ

Description* ⓘ

VPC ID ⓘ

Add Subnet(s) to this Subnet Group. You may add subnets one at a time below or [add all the subnets](#) related to this VPC. You can make additions/edits after this group is created.

Availability Zone	Subnet ID	Availability Zone	Subnet ID	CIDR Block
<input type="text" value="sa-east-1a"/>	<input type="text" value="subnet-5bf5e639"/>	sa-east-1a	subnet-5bf5e639	10.0.1.0/24

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](#).

Creating a subnet group (ElastiCache API)

Using the ElastiCache API, call `CreateCacheSubnetGroup` with the following parameters:

- `CacheSubnetGroupName`=*mysubnetgroup*
- `CacheSubnetGroupDescription`=*Testing*
- `SubnetIds.member.1`=*subnet-53df9c3a*

Example

```
https://elasticache.us-west-2.amazonaws.com/
?Action=CreateCacheSubnetGroup
&CacheSubnetGroupDescription=Testing
&CacheSubnetGroupName=mysubnetgroup
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&SubnetIds.member.1=subnet-53df9c3a
&Timestamp=20141201T220302Z
&Version=2014-12-01
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256
&X-Amz-Credential=<credential>
&X-Amz-Date=20141201T220302Z
&X-Amz-Expires=20141201T220302Z
&X-Amz-Signature=<signature>
&X-Amz-SignedHeaders=Host
```

Assigning a subnet group to a cluster

After you have created a subnet group, you can launch a cluster in an Amazon VPC. For more information, see the following.

- **Memcached cluster** – To launch a Memcached cluster, see [Creating a Memcached cluster \(console\)](#) (p. 68). 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 cluster 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, choose the one you want to modify.
4. In the lower portion of the ElastiCache console, make any changes to the description or the list of subnet IDs for the subnet group. To save your changes, choose **Save**.

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](#).

Modifying subnet groups (ElastiCache API)

Using the ElastiCache API, call `ModifyCacheSubnetGroup` with the following parameters:

- `CacheSubnetGroupName`=*mysubnetgroup*
- Any other parameters whose values you want to change. This example uses `CacheSubnetGroupDescription`=*New%20description* to change the description of the subnet group.

Example

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=ModifyCacheSubnetGroup  
&CacheSubnetGroupDescription=New%20description  
&CacheSubnetGroupName=mysubnetgroup  
&SubnetIds.member.1=subnet-42df9c3a  
&SubnetIds.member.2=subnet-48fc21a9  
&SignatureMethod=HmacSHA256  
&SignatureVersion=4  
&Timestamp=20141201T220302Z  
&Version=2014-12-01  
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256  
&X-Amz-Credential=<credential>  
&X-Amz-Date=20141201T220302Z  
&X-Amz-Expires=20141201T220302Z  
&X-Amz-Signature=<signature>  
&X-Amz-SignedHeaders=Host
```

Note

When you create a new subnet group, take 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 the 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.

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 cluster. You also cannot delete a subnet group on a cluster with Multi-AZ enabled if doing so leaves that cluster with fewer than two subnets. You must first disable **Multi-AZ** and then delete the subnet.

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, choose **Yes, 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](#).

Deleting a subnet group (ElastiCache API)

Using the ElastiCache API, call `DeleteCacheSubnetGroup` with the following parameter:

- `CacheSubnetGroupName`=*mysubnetgroup*

Example

Line breaks are added for ease of reading.

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DeleteCacheSubnetGroup  
&CacheSubnetGroupName=mysubnetgroup
```

```
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Timestamp=20141201T220302Z
&Version=2014-12-01
&X-Amz-Algorithm=&AWS;4-HMAC-SHA256
&X-Amz-Credential=<credential>
&X-Amz-Date=20141201T220302Z
&X-Amz-Expires=20141201T220302Z
&X-Amz-Signature=<signature>
&X-Amz-SignedHeaders=Host
```

This command produces no output.

For more information, see the ElastiCache API topic [DeleteCacheSubnetGroup](#).

Security groups: EC2-Classic

We
are
retiring
EC2-
Classic
on
August
15,
2022.
We
recommend
that
you
migrate
from
EC2-
Classic
to
a
VPC.
For
more
information,
see
[Migrating
an
EC2-
Classic
cluster
into
a
VPC \(p. 201\)](#)
and
the
blog
[EC2-
Classic
Networking
is
Retiring](#)

–
[Here's
How
to
Prepare.](#)

Important

Amazon ElastiCache security groups are only applicable to clusters that are *not* running in an Amazon Virtual Private Cloud environment (VPC). If you are running in an Amazon Virtual Private Cloud, **Security Groups** is not available in the console navigation pane.

If you are running your ElastiCache nodes in an Amazon VPC, you control access to your clusters with Amazon VPC security groups, which are different from ElastiCache security groups. For more information about using ElastiCache in an Amazon VPC, see [Amazon VPCs and ElastiCache security \(p. 189\)](#)

Amazon ElastiCache allows you to control access to your clusters using ElastiCache security groups. An ElastiCache security group acts like a firewall, controlling network access to your cluster. By default, network access is turned off to your clusters. If you want your applications to access your cluster, you must explicitly enable access from hosts in specific Amazon EC2 security groups. Once ingress rules are configured, the same rules apply to all clusters associated with that security group.

To allow network access to your cluster, create a security group and use the `AuthorizeCacheSecurityGroupIngress` API operation (CLI: `authorize-cache-security-group-ingress`) to authorize the desired Amazon EC2 security group (which in turn specifies the Amazon EC2 instances allowed). The security group can be associated with your cluster at the time of creation, or using the `ModifyCacheCluster` API operation (CLI: `modify-cache-cluster`).

Important

Access control based on IP range is currently not enabled at the individual cluster level. All clients to a cluster must be within the EC2 network, and authorized via security groups as described previously.

For more information about using ElastiCache with Amazon VPCs, see [Amazon VPCs and ElastiCache security \(p. 189\)](#).

Note that Amazon EC2 instances running in an Amazon VPC can't connect to ElastiCache clusters in EC2-Classic.

Topics

- [Creating a security group \(p. 220\)](#)
- [Listing available security groups \(p. 222\)](#)
- [Authorizing network access to an amazon EC2 security group \(p. 224\)](#)

Creating a security group

This topic is relevant to you only if you are not running in an Amazon VPC. If you are running in an Amazon VPC, see [Amazon VPCs and ElastiCache security](#) (p. 189).

To create a security group, you need to provide a name and a description.

The following procedures show you how to create a new security group.

Creating a security group (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 navigation pane, choose **Security Groups**.
3. Choose **Create Security Group**.
4. In **Basic Details**, enter a descriptive name and brief description for the security group. The name and description can be up to 255 characters long, and they can include a-z, A-Z, 0-9, spaces and `._-:/()#,@[]+=&{}!$*`. For **VPC**, choose the VPC in which to create the security group. The security group can only be used in the VPC in which it is created.
5. You can add security group rules now, or you can add them at any time after you have created the security group. For more information about adding security group rules, see [Adding rules to a security group](#).
6. Choose **Create**.

Creating a security group (AWS CLI)

At a command prompt, use the `create-cache-security-group` command with the following parameters:

- `--cache-security-group-name` – The name of the security group you are creating.

Example: `mysecuritygroup`

- `--description` – A description for this security group.

Example: `"My new security group"`

For Linux, macOS, or Unix:

```
aws elasticache create-cache-security-group \
  --cache-security-group-name mysecuritygroup \
  --description "My new security group"
```

For Windows:

```
aws elasticache create-cache-security-group ^
  --cache-security-group-name mysecuritygroup ^
  --description "My new security group"
```

For more information, see [create-cache-security-group](#).

Creating a security group (ElastiCache API)

Using the ElastiCache API operation `CreateCacheSecurityGroup` with the following parameters:

- CacheSecurityGroupName – The name of the security group you are creating.

Example: mysecuritygroup

- Description – A URL encoded description for this security group.

Example: My%20security%20group

Example

Line breaks are added for ease of reading.

```
https://elasticache.us-west-2.amazonaws.com /  
  ?Action=CreateCacheSecurityGroup  
  &CacheSecurityGroupName=mysecuritygroup  
  &Description=My%20security%20group  
  &Version=2015-02-02  
  &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>
```

Listing available security groups

This topic is relevant to you only if you are not running in an Amazon VPC. If you are running in an Amazon VPC, see [Amazon VPCs and ElastiCache security](#) (p. 189).

You can list which security groups have been created for your AWS account.

The following procedures show you how to list the available security groups for your AWS account.

Listing available security groups (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 navigation pane, choose **Security Groups**.

The available security groups appear in the **Security Groups** list.

Listing available security groups (AWS CLI)

At a command prompt, use the `describe-cache-security-groups` command to list all available security groups for your AWS account.

```
aws elasticache describe-cache-security-groups
```

JSON output from this command will look something like this.

```
{
  "Marker": "Marker",
  "CacheSecurityGroups": [
    {
      "OwnerId": "OwnerId",
      "CacheSecurityGroupName": "CacheSecurityGroupName",
      "Description": "Description",
      "EC2SecurityGroups": [
        {
          "Status": "Status",
          "EC2SecurityGroupName": "EC2SecurityGroupName",
          "EC2SecurityGroupOwnerId": "EC2SecurityGroupOwnerId"
        }
      ]
    }
  ]
}
```

For more information, see [describe-cache-security-groups](#).

Listing available security groups (ElastiCache API)

Using the ElastiCache API, call `DescribeCacheSecurityGroups`.

Example

Line breaks are added for ease of reading.

```
https://elasticache.us-west-2.amazonaws.com/
?Action=DescribeCacheSecurityGroups
&MaxRecords=100
&Version=2015-02-02
```

```
&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>
```

Authorizing network access to an amazon EC2 security group

This topic is relevant to you only if you are not running in an Amazon VPC. If you are running in an Amazon VPC, see [Amazon VPCs and ElastiCache security](#) (p. 189).

If you want to access your cluster from an Amazon EC2 instance, you must grant access to the Amazon EC2 security group that the EC2 instance belongs to. The following procedures show you how to grant access to an Amazon EC2 Security Group.

Important

- Authorizing an Amazon EC2 security group only grants access to your clusters from all EC2 instances belonging to the Amazon EC2 security group.
- It takes approximately one minute for changes to access permissions to take effect.

Authorizing network access to an amazon EC2 security group (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 navigation pane, choose **Security Groups**.
3. In the **Security Groups** list, choose the box to the left of the security group that you want to grant access to.
4. At the bottom of the window, in the **EC2 Security Group Name** list, choose your Amazon EC2 security group.
5. Choose **Add**.

Authorizing network access to an amazon EC2 security group (AWS CLI)

At a command prompt, use the `authorize-cache-security-group-ingress` command to grant access to an Amazon EC2 security group with the following parameters.

- `--cache-security-group-name` – the name of the security group you are granting Amazon EC2 access to.
- `--ec2-security-group-name` – the name of the Amazon EC2 security group that the Amazon EC2 instance belongs to.
- `--ec2-security-group-owner-id` – the id of the owner of the Amazon EC2 security group.

Example

For Linux, macOS, or Unix:

```
aws elasticache authorize-cache-security-group-ingress \
  --cache-security-group-name default \
  --ec2-security-group-name myec2group \
  --ec2-security-group-owner-id 987654321021
```

For Windows:

```
aws elasticache authorize-cache-security-group-ingress ^
  --cache-security-group-name default ^
  --ec2-security-group-name myec2group ^
  --ec2-security-group-owner-id 987654321021
```

The command should produce output similar to the following:

```
{
  "CacheSecurityGroup": {
    "OwnerId": "OwnerId",
    "CacheSecurityGroupName": "CacheSecurityGroupName",
    "Description": "Description",
    "EC2SecurityGroups": [
      {
        "Status": "available",
        "EC2SecurityGroupName": "EC2SecurityGroupName",
        "EC2SecurityGroupOwnerId": "EC2SecurityGroupOwnerId"
      }
    ]
  }
}
```

For more information, see [authorize-cache-security-group-ingress](#).

Authorizing network access to an amazon EC2 security group (ElastiCache API)

Using the ElastiCache API, call `AuthorizeCacheSecurityGroupIngress` with the following parameters:

- `CacheSecurityGroupName` – the name of the security group you are granting Amazon EC2 access to.
- `EC2SecurityGroupName` – the name of the Amazon EC2 security group that the Amazon EC2 instance belongs to.
- `EC2SecurityGroupOwnerId` – the id of the owner of the Amazon EC2 security group.

Example

```
https://elasticache.us-west-2.amazonaws.com/
?Action=AuthorizeCacheSecurityGroupIngress
&EC2SecurityGroupOwnerId=987654321021
&EC2SecurityGroupName=myec2group
&Version=2015-02-02
&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>
```

For more information, see [AuthorizeCacheSecurityGroupIngress](#).

Identity and access management in Amazon ElastiCache

Access to Amazon ElastiCache requires credentials that AWS can use to authenticate your requests. Those credentials must have permissions to access AWS resources, such as an ElastiCache cache cluster or an Amazon Elastic Compute Cloud (Amazon EC2) instance. The following sections provide details on how you can use [AWS Identity and Access Management \(IAM\)](#) and ElastiCache to help secure your resources by controlling who can access them.

- [Authentication \(p. 226\)](#)

- [Access control \(p. 231\)](#)

Authentication

You can access AWS as any of the following types of identities:

- **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 do not 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 *AWS General Reference*.

- **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 role**

An *IAM role* is an IAM identity that you can create in your account that has specific permissions. An IAM role is similar to an IAM user in that it is an AWS identity with permissions policies that determine what the identity can and cannot do in AWS. However, instead of being uniquely associated with one person, a role is intended to be assumable by anyone who needs it. Also, a role does not have standard long-term credentials such as a password or access keys associated with it. Instead, when you assume a role, it provides you with temporary security credentials for your role session. 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 permission sets, see [Permission sets](#) in the *AWS IAM Identity Center (successor to AWS Single Sign-On) User Guide*.
- **AWS service access** – 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*.
- **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*.

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 (p. 227)	Yes
Resource-based policies (p. 228)	No
Policy actions (p. 228)	Yes
Policy resources (p. 229)	Yes
Policy condition keys (service-specific) (p. 229)	Yes
ACLs (p. 230)	No
ABAC (tags in policies) (p. 230)	Yes
Temporary credentials (p. 230)	Yes
Principal permissions (p. 231)	Yes
Service roles (p. 231)	Yes
Service-linked roles (p. 231)	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

You can refer to [Using identity-based policies \(IAM policies\) for Amazon ElastiCache](#), for examples.

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.

Policy actions in ElastiCache use the following prefix before the action:

```
elasticache
```

For example, to grant someone permission to create a cluster with Amazon ElastiCache CreateCacheCluster API operation, you include the elasticache:CreateCacheCluster action in their policy. Policy statements must include either an Action or NotAction element. ElastiCache defines its own set of actions that describe tasks that you can perform with this service.

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 Create, include the following action.

```
"Action": "elasticache:Create*"
```

To see a list of ElastiCache actions, see [Actions Defined by Amazon ElastiCache](#) in the Service Authorization Reference.

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 in the *Service Authorization Reference*. To learn with which actions you can specify the ARN of each resource, see .

To see a list of ElastiCache resource types and their ARNs, see [ElastiCache Resource ARN Format examples](#).

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](#). To learn with which actions and resources you can use a condition key, see [Actions Defined by Amazon ElastiCache](#).

ACLs in ElastiCache

Supports ACLs	No
---------------	----

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.

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 principal permissions	Yes
--------------------------------	-----

When you use an IAM user or role to perform actions in AWS, you are considered a principal. Policies grant permissions to a principal. When you use some services, you might perform an action that then triggers another action in a different service. In this case, you must have permissions to perform both actions. To see whether an action requires additional dependent actions in a policy, see in the *Service Authorization Reference*.

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 IAM 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.

For details about using ElastiCache service-linked roles, see [Using service-linked roles for Amazon ElastiCache](#).

Access control

You can have valid credentials to authenticate your requests, but unless you have permissions you cannot create or access Amazon ElastiCache resources. For example, you must have permissions to create an ElastiCache cache cluster.

The following sections describe how to manage permissions for Amazon ElastiCache. We recommend that you read the overview first.

- [Overview of managing access permissions to your ElastiCache resources \(p. 233\)](#)
- [Using identity-based policies \(IAM policies\) for Amazon ElastiCache \(p. 237\)](#)

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*.

When granting permissions, you decide who is getting the permissions. You also decide the resources they get permissions for and the specific actions that you want to allow on those resources.

Topics

- [Amazon ElastiCache resources and operations](#) (p. 233)
- [Understanding resource ownership](#) (p. 234)
- [Managing access to resources](#) (p. 234)
- [Using identity-based policies \(IAM policies\) for Amazon ElastiCache](#) (p. 237)
- [Resource-level permissions](#) (p. 241)
- [Using condition keys](#) (p. 242)
- [Using Service-Linked Roles for Amazon ElastiCache](#) (p. 249)
- [ElastiCache API permissions: Actions, resources, and conditions reference](#) (p. 255)

Amazon ElastiCache resources and operations

In Amazon ElastiCache, the primary resource is a *cache cluster*.

These resources have unique Amazon Resource Names (ARNs) associated with them as shown following.

Note

For resource-level permissions to be effective, the resource name on the ARN string should be lower case.

Resource type	ARN format
Cluster	arn:aws:elasticache:us-east-2:123456789012:cluster:my-cluster
Parameter group	arn:aws:elasticache:us-east-2:123456789012:parametergroup:my-parameter-group
Subnet group	arn:aws:elasticache:us-east-2:123456789012:subnetgroup:my-subnet-group
Reserved instance	arn:aws:elasticache:us-east-2:123456789012:reserved-instance:my-reserved-instance

ElastiCache provides a set of operations to work with ElastiCache resources. For a list of available operations, see Amazon ElastiCache [Actions](#).

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 \(IAM policies\) \(p. 234\)](#)
- [Specifying policy elements: Actions, effects, resources, and principals \(p. 235\)](#)
- [Specifying conditions in a policy \(p. 236\)](#)

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 \(p. 237\)](#). 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 \(p. 233\)](#)), 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 \(p. 233\)](#).
- **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 \(p. 255\)](#).

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 \(p. 242\)](#). 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*.

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 \(p. 233\)](#).

The sections in this topic cover the following:

- [Permissions required to use the Amazon ElastiCache console \(p. 238\)](#)
- [AWS-managed \(predefined\) policies for Amazon ElastiCache \(p. 238\)](#)
- [Customer-managed policy examples \(p. 239\)](#)

The following shows an example of a permissions policy.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AllowClusterPermissions",
    "Effect": "Allow",
    "Action": [
      "elasticache:CreateCacheCluster",
      "elasticache:DescribeCacheClusters",
      "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:CreateCacheCluster`, `elasticache:DescribeCacheClusters`, `elasticache:ModifyCacheCluster`)
- 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 \(p. 255\)](#).

Permissions required to use the Amazon ElastiCache console

The permissions reference table lists the Amazon ElastiCache API operations and shows the required permissions for each operation. For more information about ElastiCache API operations, see [ElastiCache API permissions: Actions, resources, and conditions reference](#) (p. 255).

To use the Amazon ElastiCache console, first grant permissions for additional actions as shown in the following permissions policy.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "MinPermsForECConsole",
    "Effect": "Allow",
    "Action": [
      "elasticache:Describe*",
      "elasticache:List*",
      "ec2:DescribeAvailabilityZones",
      "ec2:DescribeVpcs",
      "ec2:DescribeAccountAttributes",
      "ec2:DescribeSecurityGroups",
      "cloudwatch:GetMetricStatistics",
      "cloudwatch:DescribeAlarms",
      "s3:ListAllMyBuckets",
      "sns:ListTopics",
      "sns:ListSubscriptions" ],
    "Resource": "*"
  }]
}
```

The ElastiCache console needs these additional permissions for the following reasons:

- Permissions for the ElastiCache actions enable the console to display ElastiCache resources in the account.
- The console needs permissions for the ec2 actions to query Amazon EC2 so it can display Availability Zones, VPCs, security groups, and account attributes.
- The permissions for cloudwatch actions enable the console to retrieve Amazon CloudWatch metrics and alarms, and display them in the console.
- The permissions for sns actions enable the console to retrieve Amazon Simple Notification Service (Amazon SNS) topics and subscriptions, and display them in the console.

AWS-managed (predefined) policies for Amazon ElastiCache

AWS addresses many common use cases by providing standalone IAM policies that are created and administered by AWS. Managed policies grant necessary permissions for common use cases so you can avoid having to investigate what permissions are needed. For more information, see [AWS Managed Policies](#) in the *IAM User Guide*.

The following AWS managed policies, which you can attach to users in your account, are specific to ElastiCache:

- **AmazonElastiCacheReadOnlyAccess** - Grants read-only access to Amazon ElastiCache resources.
- **AmazonElastiCacheFullAccess** - Grants full access to Amazon ElastiCache resources.

Note

You can review these permissions policies by signing in to the IAM console and searching for specific policies there.

You can also create your own custom IAM policies to allow permissions for Amazon ElastiCache API actions. You can attach these custom policies to the IAM users or groups that require those permissions.

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 5: Allow a user to call IAM CreateServiceLinkedRole API \(p. 241\)](#)). 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 more information about what permissions are needed to use the ElastiCache console, see [Permissions required to use the Amazon ElastiCache console \(p. 238\)](#).

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 to create and manage security groups \(p. 239\)](#)
- [Example 2: Allow a user read-only access to ElastiCache resources \(p. 239\)](#)
- [Example 3: Allow a user to perform common ElastiCache system administrator tasks \(p. 240\)](#)
- [Example 4: Allow a user to access all ElastiCache API actions \(p. 240\)](#)
- [Example 5: Allow a user to call IAM CreateServiceLinkedRole API \(p. 241\)](#)
- [Example 6: Allow a user to connect to replication group using IAM authentication \(p. 241\)](#)

Example 1: Allow a user to create and manage security groups

The following policy grants permissions for the security group's specific ElastiCache actions. Typically, you attach this type of permissions policy to the system administrators group.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "SecGrpAllows",
    "Effect": "Allow",
    "Action": [
      "elasticache:CreateCacheSecurityGroup",
      "elasticache>DeleteCacheSecurityGroup",
      "elasticache:DescribeCacheSecurityGroup",
      "elasticache:AuthorizeCacheSecurityGroupIngress",
      "elasticache:RevokeCacheSecurityGroupIngress"
    ],
    "Resource": "*"
  }]
}
```

Example 2: 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": "ECUnrestricted",
    "Effect": "Allow",
    "Action": [
      "elasticache:Describe*",
      "elasticache:List*"
    ],
    "Resource": "*"
  }]
}
```

Example 3: Allow a user to perform common ElastiCache system administrator tasks

Common system administrator tasks include modifying cache clusters, parameters, and parameter groups. 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": "ECAAllowSpecific",
    "Effect": "Allow",
    "Action": [
      "elasticache:ModifyCacheCluster",
      "elasticache:RebootCacheCluster",
      "elasticache:DescribeCacheClusters",
      "elasticache:DescribeEvents",
      "elasticache:ModifyCacheParameterGroup",
      "elasticache:DescribeCacheParameterGroups",
      "elasticache:DescribeCacheParameters",
      "elasticache:ResetCacheParameterGroup",
      "elasticache:DescribeEngineDefaultParameters"
    ],
    "Resource": "*"
  }]
}
```

Example 4: 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": "ECAAllowSpecific",
    "Effect": "Allow",
    "Action": [
      "elasticache:*"
    ],
    "Resource": "*"
  }]
}
```

Example 5: 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"
        }
      }
    }
  ]
}
```

Example 6: Allow a user to connect to replication group using IAM authentication

The following policy allows any user to connect to any replication group, using IAM authentication, from the IP address 123.45.167.89.

```
{
  "Version" : "2012-10-17",
  "Statement" :
  [
    {
      "Effect" : "Allow",
      "Action" : ["elasticache:Connect"],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:replicationgroup:*",
      ],
      "Condition": {
        "IpAddress": {
          "aws:SourceIp": "123.45.167.89"
        }
      }
    },
    {
      "Effect" : "Allow",
      "Action" : ["elasticache:Connect"],
      "Resource" : [
        "arn:aws:elasticache:us-east-1:123456789012:user:*"
      ]
    }
  ]
}
```

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.

ElastiCache Resource ARN Format

Note

For resource-level permissions to be effective, the resource name on the ARN string should be lower case.

- Cluster – arn:aws:elasticache:us-east-2:123456789012:cluster:my-cluster
- Parameter group – arn:aws:elasticache:us-east-2:123456789012:parametergroup:my-parameter-group
- Security group – arn:aws:elasticache:us-east-2:123456789012:securitygroup:my-security-group
- Subnet group – arn:aws:elasticache:us-east-2:123456789012:subnetgroup:my-subnet-group
- Reserved instance – arn:aws:elasticache:us-east-2:123456789012:reservedinstance:my-reserved-instance

Examples

- [Example 1: Allow a user full access to specific ElastiCache resource types \(p. 242\)](#)

Example 1: Allow a user full access to specific ElastiCache resource types

The following policy explicitly allows all resources of type subnet group, security group and replication group.

```
{
  "Sid": "Example1",
  "Effect": "Allow",
  "Action": "elasticache:*",
  "Resource": [
    "arn:aws:elasticache:us-east-1:account-id:subnetgroup:*",
    "arn:aws:elasticache:us-east-1:account-id:securitygroup:*"
  ]
}
```

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](#). 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 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"
                ]
            }
        }
    }
]
```

The following table shows the service-specific condition keys that apply to ElastiCache and the actions that use them.

Datatype	Used by	Affected resource type (* If marked with an asterisk, only this resource will be affected if present in the request.)
elasticache:CacheNodeType	CreateCacheCluster	Cluster
	ModifyCacheClusters	Cluster

Datatype	Used by	Affected resource type (* If marked with an asterisk, only this resource will be affected if present in the request.)
cluster creation or scaling operations.		
ElastiCache:EngineVersion Filtering access by the engineVersion parameter present in creation or cluster modification requests.	CreateCacheCluster	Cluster
	ModifyCacheClusters	Cluster
ElastiCache:EngineType Filtering access by the engine type present in creation requests. For replication group creations, default engine "redis" is used as key if parameter is not present.	CreateCacheCluster	Cluster

Datatype	Used by	Affected resource type (* If marked with an asterisk, only this resource will be affected if present in the request.)
elasticache:AuthTokenEnabled ElastiCache access by the presence of non empty AuthToken parameter in the request.	CreateCacheCluster	Cluster
	ModifyCacheClusters	Cluster
elasticache:CacheParameterGroupName ElastiCache access by the the CacheParameterGroupName parameter in the request.	CreateCacheCluster	Cluster, ReplicationGroup*
	ModifyCacheClusters	Cluster
	CreateCacheParameterGroup	ParameterGroup
	ModifyCacheParameterGroup	ParameterGroup
	DeleteCacheParameterGroup	ParameterGroup
	ResetCacheParameterGroup	ParameterGroup

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 \(p. 255\)](#).

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: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"
    "elasticache:CreateReplicationGroup"
  ],
  "Resource": [
    "arn:aws:elasticache:*:*:cluster:*"
    "arn:aws:elasticache:*:*:replicationgroup:*"
  ],
  "Condition": {
    "StringEquals": {
      "elasticache:CacheNodeType": [
        "cache.t2.micro",
        "cache.t2.medium"
      ]
    }
  }
}
```

2. **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"
        "elasticache:CreateReplicationGroup"
      ],
      "Resource": [
        "arn:aws:elasticache:*:*:cluster:*"
        "arn:aws:elasticache:*:*:replicationgroup:*"
      ],
      "Condition": {
        "StringEquals": {
          "elasticache:EngineVersion": "1.6.6"
        }
      }
    }
  ]
}
```

3. **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:CreateReplicationGroup"
        ],
        "Resource": [
            "arn:aws:elasticache:*:*:cluster:*"
        ],
        "Condition": {
            "StringEquals": {
                "elasticache:CacheParameterGroupName": "my-org-param-group"
            }
        }
    }
]
```

4. **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"
                ]
            }
        }
    }
}
```

5. **elasticache:createcacheclasser**: 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 policies 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

- [Service-Linked Role Permissions for Amazon ElastiCache \(p. 249\)](#)
- [Creating a Service-Linked Role \(IAM\) \(p. 250\)](#)
 - [Creating a Service-Linked Role \(IAM Console\) \(p. 251\)](#)
 - [Creating a Service-Linked Role \(IAM CLI\) \(p. 251\)](#)
 - [Creating a Service-Linked Role \(IAM API\) \(p. 251\)](#)
- [Editing the Description of a Service-Linked Role for Amazon ElastiCache \(p. 251\)](#)
 - [Editing a Service-Linked Role Description \(IAM Console\) \(p. 251\)](#)
 - [Editing a Service-Linked Role Description \(IAM CLI\) \(p. 252\)](#)
 - [Editing a Service-Linked Role Description \(IAM API\) \(p. 252\)](#)
- [Deleting a Service-Linked Role for Amazon ElastiCache \(p. 253\)](#)
 - [Cleaning Up a Service-Linked Role \(p. 253\)](#)
 - [Deleting a Service-Linked Role \(IAM Console\) \(p. 253\)](#)
 - [Deleting a Service-Linked Role \(IAM CLI\) \(p. 254\)](#)
 - [Deleting a Service-Linked Role \(IAM API\) \(p. 254\)](#)

Service-Linked Role Permissions for Amazon ElastiCache

Amazon ElastiCache uses the service-linked role named **AWSServiceRoleForElastiCache** – This policy allows ElastiCache to manage AWS resources on your behalf as necessary for managing your cache.

The AWSServiceRoleForElastiCache service-linked role permissions policy allows Amazon ElastiCache to complete the following actions on the specified resources:

```
Permission policy:
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:CreateNetworkInterface",
        "ec2:CreateSecurityGroup",
        "ec2>DeleteNetworkInterface",
        "ec2>DeleteSecurityGroup",
        "ec2:DescribeAvailabilityZones",
```

```
"ec2:DescribeNetworkInterfaces",
"ec2:DescribeSecurityGroups",
"ec2:DescribeSubnets",
"ec2:DescribeVpcs",
"ec2:ModifyNetworkInterfaceAttribute",
"ec2:RevokeSecurityGroupIngress",
"cloudwatch:PutMetricData",
"outposts:GetOutpost",
"outposts:GetOutpostInstanceTypes",
"outposts:ListOutposts",
"outposts:ListSites",
"logs:DeleteLogDelivery",
"application-autoscaling:DeregisterScalableTarget",
"application-autoscaling:DescribeScalableTargets"
],
"Resource": "*"
},
{
  "Effect": "Allow",
  "Action": [
    "ec2:AssignIpv6Addresses",
    "ec2:UnassignIpv6Addresses"
  ],
  "Resource": "arn:aws:ec2:*:*:network-interface/eni-*"
}
]
```

To allow an IAM entity to create `AWSServiceRoleForElastiCache` service-linked roles

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 roles

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` (console)

- To delete a cluster, see the following:
 - [Using the AWS Management Console \(p. 85\)](#)
 - [Using the AWS CLI \(p. 85\)](#)
 - [Using the ElastiCache API \(p. 86\)](#)

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](#) (p. 231) and write permissions policies to attach to an IAM policy (either identity-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](#) (p. 242). For a complete list of AWS-wide keys, see [Available Keys for Conditions](#) in the *IAM User Guide*.

Note

To specify an action, use the `elasticache:` prefix followed by the API operation name (for example, `elasticache:DescribeCacheClusters`).

Amazon ElastiCache API and required permissions for actions

[AddTagsToResource](#)

Action: `elasticache:AddTagsToResource`

Resource: *

[AuthorizeCacheSecurityGroupIngress](#)

Action: `elasticache:AuthorizeCacheSecurityGroupIngress`

Resource: *

[CreateCacheCluster](#)

Actions: `elasticache:CreateCacheCluster`

`s3:GetObject`

Note

If you use the `SnapshotArns` parameter, each member of the `SnapshotArns` list requires its own `s3:GetObject` permission with the S3 ARN as its resource.

Resource: *

`arn:aws:s3:::my_bucket/snapshot1.rdb`

Where *my_bucket/snapshot1* is an S3 bucket and snapshot that you want to create the cache cluster from.

[CreateCacheParameterGroup](#)

Action: `elasticache:CreateCacheParameterGroup`

Resource: *

[CreateCacheSecurityGroup](#)

Action: `elasticache:CreateCacheSecurityGroup`

Resource: *

CreateCacheSubnetGroup

Action: elasticache:CreateCacheSubnetGroup

Resource: *

DeleteCacheCluster

Action: elasticache>DeleteCacheCluster

Resource: *

DeleteCacheParameterGroup

Action: elasticache>DeleteCacheParameterGroup

Resource: *

DeleteCacheSecurityGroup

Action: elasticache>DeleteCacheSecurityGroup

Resource: *

DeleteCacheSubnetGroup

Action: elasticache>DeleteCacheSubnetGroup

Resource: *

DeleteGlobalReplicationGroup

Action: elasticache>DeleteGlobalReplicationGroup

Resource: *

DescribeCacheClusters

Action: elasticache:DescribeCacheClusters

Resource: *

DescribeCacheEngineVersions

Actions: elasticache:DescribeCacheEngineVersions

Resource: *

DescribeCacheParameterGroups

Action: elasticache:DescribeCacheParameterGroups

Resource: *

DescribeCacheParameters

Action: elasticache:DescribeCacheParameters

Resource: *

DescribeCacheSecurityGroups

Action: elasticache:DescribeCacheSecurityGroups

Resource: *

DescribeCacheSubnetGroups

Action: elasticache:DescribeCacheSubnetGroups

Resource: *

[DescribeEngineDefaultParameters](#)

Action: elasticache:DescribeEngineDefaultParameters

Resource: *

[DescribeEvents](#)

Action: elasticache:DescribeEvents

Resource: *

[DescribeReservedCacheNodes](#)

Action: elasticache:DescribeReservedCacheNodes

Resource: *

[DescribeReservedCacheNodesOfferings](#)

Action: elasticache:DescribeReservedCacheNodesOfferings

Resource: *

[ListTagsForResource](#)

Action: elasticache:ListTagsForResource

Resource: *

[ModifyCacheCluster](#)

Action: elasticache:ModifyCacheCluster

Resource: *

[ModifyCacheParameterGroup](#)

Action: elasticache:ModifyCacheParameterGroup

Resource: *

[ModifyCacheSubnetGroup](#)

Action: elasticache:ModifyCacheSubnetGroup

Resource: *

[PurchaseReservedCacheNodesOffering](#)

Action: elasticache:PurchaseReservedCacheNodesOffering

Resource: *

[RebootCacheCluster](#)

Action: elasticache:RebootCacheCluster

Resource: *

[RemoveTagsFromResource](#)

Action: elasticache:RemoveTagsFromResource

Resource: *

[ResetCacheParameterGroup](#)

Action: elasticache:ResetCacheParameterGroup

Resource: *

[RevokeCacheSecurityGroupIngress](#)

Action: elasticache:RevokeCacheSecurityGroupIngress

Resource: *

[StartMigration](#)

Action: elasticache:StartMigration

Resource: *

Monitoring usage, events, and costs

To manage your enterprise caching solution, it's important that you know how your clusters are performing and the resources they're consuming. It's also important that you know the events that are being generated and the costs of your deployment.

Amazon CloudWatch provides metrics for monitoring your cache performance. In addition, cost allocation tags help you monitor and manage costs.

Topics

- [Monitoring use with CloudWatch metrics \(p. 259\)](#)
- [Logging Amazon ElastiCache API calls with AWS CloudTrail \(p. 267\)](#)
- [Monitoring ElastiCache events \(p. 270\)](#)

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? \(p. 264\)](#) in this guide.

Topics

- [Host-Level Metrics \(p. 259\)](#)
- [Metrics for Memcached \(p. 261\)](#)
- [Which metrics should I monitor? \(p. 264\)](#)
- [Monitoring CloudWatch Cluster and Node Metrics \(p. 265\)](#)

Host-Level Metrics

The AWS/ElastiCache namespace includes the following host-level metrics for individual cache nodes.

See Also

- [Metrics for Memcached \(p. 261\)](#)

Metric	Description	Unit
CPUtilization	The percentage of CPU utilization for the entire host.	Percent
CPUCreditBalance	<p>The number of earned CPU credits that an instance has accrued since it was launched or started. For T2 Standard, the CPUCreditBalance also includes the number of launch credits that have been accrued.</p> <p>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.</p> <p>The credits in the CPUCreditBalance are available for the instance to spend to burst beyond its baseline CPU utilization.</p> <p>When an instance is running, credits in the CPUCreditBalance do not expire. When a T3 or T3a instance stops, the CPUCreditBalance value persists for seven days. Thereafter, all accrued</p>	Credits (vCPU-minutes)

Metric	Description	Unit
	<p>credits are lost. When a T2 instance stops, the CPUCreditBalance value does not persist, and all accrued credits are lost.</p> <p>CPU credit metrics are available at a five-minute frequency only.</p>	
CPUCreditUsage	<p>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).</p> <p>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.</p>	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
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
NetworkLinkLocalAllowanceExceeded	The number of packets shaped because the PPS of the traffic to local proxy services exceeded the maximum for the network interface. This impacts traffic to the DNS service, the Instance Metadata Service, and the Amazon Time Sync Service.	Count

Metric	Description	Unit
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
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 \(p. 259\)](#)

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
CmdGets	The number of get commands the cache has received.	Count
CmdSet	The number of set commands the cache has received.	Count
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.	Count

Metric	Description	Unit
	<p>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 seen in the <i>Node type Specific Parameters</i> of the applicable Parameter Group.</p> <p>The total connections is the sum of client connections, the connections for monitoring and the internal connections mentioned above.</p>	
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
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

Metric	Description	Unit
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
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 <code>total_connections</code> statistic by recording the change in <code>total_connections</code> 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 <code>total_items</code> statistic by recording the change in <code>total_items</code> across a period of time.	Count
UnusedMemory	<p>The amount of memory not used by data. This is derived from the Memcached statistics <code>limit_maxbytes</code> and <code>bytes</code> by subtracting <code>bytes</code> from <code>limit_maxbytes</code>.</p> <p>Because Memcached overhead uses memory in addition to that used by data, UnusedMemory 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.</p> <p>For more detailed information, see Memcached item memory usage.</p>	Bytes

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](#) (p. 264)
- [SwapUsage](#) (p. 264)
- [Evictions](#) (p. 264)
- [CurrConnections](#) (p. 264)

CPUUtilization

This is a host-level metric reported as a percent. For more information, see [Host-Level Metrics](#) (p. 259).

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](#) (p. 259).

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>
```

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"
}
```

Monitoring 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 \(p. 270\)](#)
- [Viewing ElastiCache events \(p. 274\)](#)
- [Event Notifications and Amazon SNS \(p. 277\)](#)

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-2:123456789xx: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.amazonaws.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 \(p. 274\)](#).

Event Name	Message	Description
ElastiCache:AddCacheNodeComplete	ElastiCache:AddCacheNodeComplete <i>cache-cluster</i>	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 <i>cluster-name</i>	A cache node could not be added because there are not enough available IP addresses.
ElastiCache:CacheClusterParameterGroupChanged	ElastiCache:CacheClusterParameterGroupChanged <i>cluster-name</i>	One or more cache cluster parameters have been changed.
ElastiCache:CacheClusterProvisioningComplete	ElastiCache:CacheClusterProvisioningComplete <i>cluster-name-0001-005</i>	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:CacheClusterProvisioningFailed <i>cluster-name</i>	An attempt was made to launch a new cache cluster into a nonexistent virtual private cloud (VPC).
ElastiCache:CacheClusterScalingComplete	ElastiCache:CacheClusterScalingComplete <i>cluster-name</i>	Scaling for cache-cluster completed successfully.
ElastiCache:CacheClusterScalingFailed	ElastiCache:CacheClusterScalingFailed <i>cluster-name</i>	Scale-up operation on cache-cluster failed.

Event Name	Message	Description
ElastiCache:CacheClusterSecurityGroupsModified	ElastiCache:CacheClusterSecurityGroupsModified: <i>cluster-name</i>	<p>One of the following events has occurred:</p> <ul style="list-style-type: none"> 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.
ElastiCache:CacheNodeReplaceStarted	ElastiCache:CacheNodeReplaceStarted: <i>cluster-name</i>	<p>ElastiCache has detected that the host running a cache node is degraded or unreachable and has started replacing the cache node.</p> <p>Note The DNS entry for the replaced cache node is not changed.</p> <p>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.</p>

Event Name	Message	Description
ElastiCache:CacheNodeReplaceComplete	ElastiCache:CacheNodeReplaceComplete <i>cluster-name</i>	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 <i>cluster-name</i>	One or more cache nodes has been rebooted. Message (Memcached): "Cache node %s shutdown" Then a second message: "Cache node %s restarted"
ElastiCache:CertificateRenewalComplete (Redis only)	ElastiCache:CertificateRenewalComplete	The Amazon ElastiCache certificate was successfully renewed.
ElastiCache:CreateReplicationGroupComplete	ElastiCache:CreateReplicationGroupComplete <i>cluster-name</i>	The replication group was successfully created.
ElastiCache>DeleteCacheClusterComplete	ElastiCache>DeleteCacheClusterComplete <i>cluster-name</i>	The deletion of a cache cluster and all associated cache nodes has completed.
ElastiCache:FailoverComplete (Redis only)	ElastiCache:FailoverComplete <i>mycluster</i>	Failover over to a replica node was successful.
ElastiCache:ReplicationGroupIncreaseReplicasComplete	ElastiCache:ReplicationGroupIncreaseReplicasComplete <i>cluster-name-0001-005</i>	The number of replicas in the cluster has been increased.
ElastiCache:ReplicationGroupIncreaseReplicasStarted	ElastiCache:ReplicationGroupIncreaseReplicasStarted <i>cluster-name-0003-004</i>	The process of adding replicas to your cluster has begun.
ElastiCache:NodeReplacementCancel	ElastiCache:NodeReplacementCancel <i>cluster-name</i>	A node in your cluster that was scheduled for replacement is no longer scheduled for replacement.

Event Name	Message	Description
ElastiCache:NodeReplacementRescheduled	Failed ElastiCache:NodeReplacementRescheduled : <i>cluster-name</i>	Rescheduled : 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 nodes (p. 49) .
ElastiCache:NodeReplacementScheduled	Failed ElastiCache:NodeReplacementScheduled : <i>cluster-name</i>	Scheduled : cluster is scheduled for replacement during the window described in the notification. For information on what actions you can take, see Replacing nodes (p. 49) .
ElastiCache:RemoveCacheNodeComplete	Failed ElastiCache:RemoveCacheNodeComplete : <i>cluster-name</i>	Complete : node has been removed from the cache cluster.
ElastiCache:ReplicationGroupScalingComplete	Failed ElastiCache:ReplicationGroupScalingComplete : <i>cluster-name</i>	Scale-up complete : replication group completed successfully.
ElastiCache:ReplicationGroupScalingFailed	Failed applying modification to cache node type to %s."	Scale-up operation on replication group failed.
ElastiCache:ServiceUpdateAvailable	Failed ElastiCache:ServiceUpdateAvailable : <i>cluster-name</i>	A self-service update is available for the node.
ElastiCache:SnapshotComplete (Redis only)	ElastiCache:SnapshotComplete : <i>cluster-name</i>	A cache snapshot has completed successfully.
ElastiCache:SnapshotFailed (Redis only)	SnapshotFailed : <i>cluster-name</i>	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 \(p. 274\)](#)

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.
- [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 that helps you check your compliance with security industry standards and best practices.
- [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.

Self-Service Security Updates for Compliance

ElastiCache offers a self-service software update feature called **Service Updates** via the Console, API and CLI. Using this feature, you can manage security updates on your clusters on-demand and in real-time. This feature allows you to control when you update clusters with the latest required security fixes, minimizing the impact on your business.

Security updates are released via the **Service Updates** feature. They are specified by the **Update Type** field of value **security update**. The Service Update has corresponding **Severity** and **Recommended Apply by Date** fields. In order to maintain compliance of your clusters, you must apply the available updates by the **Recommended Apply by Date**. The field **SLA Met** reflects your cluster's compliance status.

Note

If you do not apply the Service Update by the recommended date or when the Service Update expires, ElastiCache will not take any action to apply the update on your behalf. You will be notified of the Service Updates applicable to your Redis clusters via an announcement on the Redis console, email, Amazon SNS, CloudWatch events and AWS Health Dashboard. For more information on Self-Service Maintenance see [Service updates in ElastiCache for Memcached \(p. 284\)](#).

CloudWatch events and AWS Health Dashboard are not supported in the following regions:

- us-gov-west-1
- us-gov-east-1
- cn-north-1
- cn-northwest-1

HIPAA Eligibility

The AWS HIPAA Compliance program includes Amazon ElastiCache for Memcached as a HIPAA eligible service.

To use ElastiCache for Memcached in compliance with HIPAA, you need to set up a Business Associate Agreement (BAA) with AWS. In addition, your cluster and the nodes within your cluster must satisfy the requirements for engine version, node type, and data security listed following.

Requirements

To enable HIPAA support on your ElastiCache for Memcached cluster, your cluster and nodes within the cluster must satisfy the following requirements.

- **Engine version requirements** – Your cluster must be running engine version 1.16.12 to qualify for HIPAA eligibility.
- **Node type requirements** – Your cluster must be running a current-generation node type. For more information, see the following:
 - [Supported node types \(p. 46\)](#)
 - [Choosing your Memcached node size \(p. 65\)](#)
- **Data security requirements** – Your cluster must enable in-transit encryption. For more information, see [ElastiCache in-transit encryption \(TLS\) \(p. 185\)](#),
- **Security Updates Requirement** – You must update your Memcached cluster with the latest Service Updates of type **security** by the **Recommended Apply by Date**. You can update the cluster in real-time and on-demand to ensure no impact to your business. For more information, see [Service updates in ElastiCache for Memcached \(p. 284\)](#)

By implementing these requirements, ElastiCache for Memcached can be used to store, process, and access Protected Health Information (PHI) in compliance with HIPAA.

For general information about AWS Cloud and HIPAA eligibility, see the following:

- [HIPAA Compliance](#)
- [Architecting for HIPAA Security and Compliance on Amazon Web Services](#)
- **Security Updates Requirement** – You must regularly update your Redis cluster by the **Recommended Apply by Date**. You can update the cluster in real-time and on-demand to ensure no impact to your business. For more information, see [Service updates in ElastiCache for Memcached \(p. 284\)](#).

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 \(p. 283\)](#)

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 \(p. 283\)](#)
- [Recommendations \(p. 284\)](#)

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

To mitigate the impact of a node failure, spread your cached data over more nodes. Because Memcached does 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 40 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 40 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 \(console\) \(p. 68\)](#).

Mitigating Availability Zone Failures

To mitigate the impact of an Availability Zone failure, 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 cached 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 \(console\)](#) (p. 68).

For more information on regions and Availability Zones, see [Choosing regions and availability zones](#) (p. 34).

Recommendations

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 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.

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) 1.0 or later. We recommend TLS 1.2 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 for Memcached

ElastiCache for Memcached automatically monitors your fleet of clusters and nodes to apply service updates as they become available. Typically, you set up a predefined maintenance window so that ElastiCache for Memcached 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. You can also monitor the progress of these updates to your selected ElastiCache for Memcached cluster in real time.

Managing the service updates

ElastiCache for Memcached service updates are released on a regular basis. If you have one or more qualifying 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 for Memcached console. By using this dashboard, you can view all the service updates and their status for your ElastiCache for Memcached fleet.

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 for Memcached are always up-to-date with current security patches.

The following sections explore these options in detail.

Topics

- [Applying the service updates \(p. 285\)](#)

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 for Memcached will schedule to apply the update during your clusters' maintenance window 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 for Memcached clusters, see [Applying the service updates using the console \(p. 285\)](#).

When a new service update is available for one or more of your ElastiCache for Memcached clusters, you can use the ElastiCache for Memcached 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 Memcached fleet.
 - **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 Memcached 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 --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-update-cluster --service-update  
ServiceUpdateNameToApply=sample-service-update --cluster-names cluster-1  
cluster2
```

For more information, see [batch-apply-update-action](#).

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
Nodes per Region	300
Nodes per cluster	40
Parameter groups per Region	150
Security groups per Region	50
Subnet groups per Region	150
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 \(p. 288\)](#)
- [ElastiCache API Reference](#)
- [ElastiCache section of the AWS CLI Reference](#)
- [Amazon ElastiCache error messages \(p. 295\)](#)
- [Notifications \(p. 296\)](#)

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 \(p. 288\)](#)
- [Available libraries \(p. 290\)](#)
- [Troubleshooting applications \(p. 291\)](#)

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
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 \(p. 292\)](#)
- [Getting the command line tools \(p. 293\)](#)
- [Setting up the tools \(p. 293\)](#)
- [Providing credentials for the tools \(p. 294\)](#)
- [Environmental variables \(p. 294\)](#)

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 \(p. 295\)](#)
- [Customer's node quota exceeded \(p. 295\)](#)
- [Insufficient cache cluster capacity \(p. 295\)](#)

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 clusters](#) (p. 154).

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](#) (p. 296)

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 7, 2022

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
ElastiCache for Memcached now supports IPV6 (p. 297)	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 (p. 297)	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 (p. 297)	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 (p. 297)	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. (p. 297)	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 (p. 297)	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 (p. 297)	A <i>Local Zone</i> 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 (p. 297)	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 (p. 297)	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 (p. 297)	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 (p. 297)	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. (p. 297)	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 <code>no_modern</code> and <code>inline_ascii_resp</code> parameters. For more information, see Memcached Version 1.5.10 .	November 14, 2018
User Guide restructure (p. 297)	The single <i>ElastiCache User Guide</i> 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: <ul style="list-style-type: none"> Supported regions & endpoints (p. 36) Supported node types (p. 46) 	February 12, 2018
Support for EU (Paris).	ElastiCache added support for the EU (Paris) Region. For more information, see the following: <ul style="list-style-type: none"> Supported regions & endpoints (p. 36) Supported node types (p. 46) 	December 18, 2017
Support for China (Ningxia) Region	Amazon ElastiCache added support for China (Ningxia) Region. For more information, see the following: <ul style="list-style-type: none"> Supported regions & endpoints (p. 36) Supported node types (p. 46) 	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: <ul style="list-style-type: none"> Using Service-Linked Roles for Amazon ElastiCache (p. 249) 	December 7, 2017

Change	Description	Date Changed
	<ul style="list-style-type: none"> • Set up your permissions (new ElastiCache users only) (p. 17) 	
Support for R4 node types	<p>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:</p> <ul style="list-style-type: none"> • Supported node types (p. 46) • Memcached node-type specific parameters (p. 181) 	November 20, 2017
Connection patterns topic	<p>ElastiCache documentation adds a topic covering various patterns for accessing an ElastiCache cluster in an Amazon VPC.</p> <p>For more information, see Access Patterns for Accessing an ElastiCache Cluster in an Amazon VPC (p. 195) in the <i>ElastiCache User Guide</i>.</p>	April 24, 2017
Support for Memcached 1.4.34	<p>ElastiCache adds support Memcached version 1.4.34, which incorporates a number of fixes to earlier Memcached versions.</p> <p>For more information, see Memcached 1.4.34 Release Notes at Memcached on GitHub.</p>	April 10, 2017
Support for Memcached 1.4.33	<p>ElastiCache adds support for Memcached version 1.4.33. For more information, see the following:</p> <ul style="list-style-type: none"> • Memcached version 1.4.33 (p. 132) • Memcached 1.4.33 added parameters (p. 175) 	December 20, 2016
Support for EU West (London) Region	<p>ElastiCache adds support for EU (London) Region. Only node types T2 and M4 are currently supported. For more information, see the following:</p> <ul style="list-style-type: none"> • Supported regions & endpoints (p. 36) • Supported node types (p. 46) 	December 13, 2016
Support for Canada (Montreal) Region	<p>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:</p> <ul style="list-style-type: none"> • Supported regions & endpoints (p. 36) • Supported node types (p. 46) 	December 8, 2016
Support for M4 and R3 node types	<p>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:</p> <ul style="list-style-type: none"> • Supported regions & endpoints (p. 36) • Supported node types (p. 46) 	November 1, 2016

Change	Description	Date Changed
US East 2 (Ohio) Region support	ElastiCache adds support for the US East (Ohio) Region (<i>us-east-2</i>) with M4, T2, and R3 node types. For more information, see the following: <ul style="list-style-type: none"> Supported regions & endpoints (p. 36) Supported node types (p. 46) 	October 17, 2016
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: <ul style="list-style-type: none"> Supported node types (p. 46) Memcached node-type specific parameters (p. 181) 	August 3, 2016
Mumbai Region support	ElastiCache adds support for the Asia Pacific (Mumbai) Region. For more information, see the following: <ul style="list-style-type: none"> Supported node types (p. 46) Memcached node-type specific parameters (p. 181) 	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 node types (p. 46) .	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 (p. 28) .	February 12, 2016
Support for Asia Pacific (Seoul) Region	ElastiCache adds support for the Asia Pacific (Seoul) (<i>ap-northeast-2</i>) Region with t2, m3, and r3 node types.	January 6, 2016
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 <i>jenkins</i> or <i>murmur3</i> 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 (p. 125) .	July 29, 2015
New topic: Accessing ElastiCache from outside AWS	Added new topic on how to access ElastiCache resources from outside AWS. For more information, see Accessing ElastiCache resources from outside AWS (p. 92) .	July 9, 2015

Change	Description	Date Changed
Node replacement messages added	ElastiCache adds three messages pertaining to scheduled node replacement, <code>ElastiCache:NodeReplacementScheduled</code> , <code>ElastiCache:NodeReplacementRescheduled</code> , and <code>ElastiCache:NodeReplacementCanceled</code> . For more information and actions you can take when a node is scheduled for replacement, see ElastiCache's Event Notifications and Amazon SNS (p. 277).	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 (p. 139).	February 9, 2015
Support for AWS GovCloud (US-West) Region	ElastiCache adds support for the AWS GovCloud (US-West) (<i>us-gov-west-1</i>) Region.	January 29, 2015
Support for Europe (Frankfurt) Region	ElastiCache adds support for the Europe (Frankfurt) (<i>eu-central-1</i>) 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 (p. 267).	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 (p. 157).	September 11, 2014
Flexible node placement supported for Memcached	ElastiCache adds support for creating Memcached nodes across multiple Availability Zones. For more information, see Step 2: Create a cluster (p. 18).	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 (p. 157).	July 1, 2014
PHP auto discovery	Added support for PHP version 5.5 auto discovery. For more information, see Installing the ElastiCache cluster client for PHP (p. 116).	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 (p. 189).	January 8, 2013
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

Change	Description	Date Changed
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 (p. 189) .	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, and 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 scalability improvements, and several bug fixes. There are several new cache parameters that can be configured. For more information, see Configuring engine parameters using parameter groups (p. 157) .	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 <i>Amazon ElastiCache User Guide</i> .	August 22, 2011

AWS glossary

For the latest AWS terminology, see the [AWS glossary](#) in the *AWS General Reference*.