
Amazon ElastiCache

ElastiCache for Memcached User Guide

API Version 2015-02-02



Amazon ElastiCache: ElastiCache for Memcached User Guide

Copyright © 2018 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
Use Cases and How ElastiCache Can Help	3
In-Memory Data Store	3
ElastiCache Customer Testimonials	4
ElastiCache for Memcached Resources	5
ElastiCache for Memcached Components and Features	7
Nodes	7
Clusters	8
AWS Regions and Availability Zones	9
Endpoints	9
Parameter Groups	9
Security	9
Security Groups	10
Subnet Groups	10
Events	10
Tools for Managing Your Implementation	11
Using the AWS Management Console	11
Using the AWS CLI	11
Using the AWS SDK	11
Using the ElastiCache API	11
See also	11
Comparing Memcached and Redis	12
Getting Started with ElastiCache for Memcached	16
Setting Up	16
Create Your AWS Account	16
Set Up Your Permissions (New ElastiCache Users Only)	16
Deploy a Memcached Cluster	17
Determine Your Cluster's Requirements	17
Step 1: Launch a Memcached Cluster	18
Step 2: Authorize Access	20
Step 3: Connect to a Cluster's Node	21
Step 4: Delete Your Cluster (Avoid Unnecessary Charges)	22
Where Do I Go From Here?	23
Tutorials and Videos	24
Videos	25
Introductory Videos	25
Advanced Videos	25
Caching Strategies and Best Practices	27
Caching Strategies	27
Lazy Loading	27
Write Through	29
Adding TTL	30
Related Topics	31
Mitigating Failures	32
Mitigating Failures when Running Memcached	32
Recommendations	33
Configuring Your ElastiCache Client for Efficient Load Balancing	34
Consistent Hashing Using Java	34
Consistent Hashing Using PHP	34
Consistent Hashing Using .NET	35
Managing Your ElastiCache for Memcached Implementation	36
Engine Versions and Upgrading	36
Supported Memcached Versions	37
Upgrading Engine Versions	39

Choosing Regions and Availability Zones	39
Supported Regions & Endpoints	41
Locating Your Nodes	44
Managing Maintenance	44
Managing Nodes	45
Connecting to Nodes	46
Reserved Nodes	48
Supported Node Types	49
Replacing Nodes	52
Managing Your ElastiCache Clusters	53
Supported Memcached Versions	53
Other ElastiCache Cluster Operations	53
Creating a Cluster	54
Viewing a Cluster's Details	87
Modifying a Cluster	92
Rebooting a Cluster	94
Adding Nodes to a Cluster	95
Removing Nodes from a Cluster	100
Canceling Pending Add or Delete Node Operations	105
Deleting a Cluster	106
Accessing Your Cluster	108
Determine the Cluster's Platform	108
Grant Access to Your Cluster	110
Finding Connection Endpoints	114
Finding Endpoints (Console)	116
Finding Endpoints (AWS CLI)	118
Finding Endpoints (ElastiCache API)	120
Scaling ElastiCache for Memcached Clusters	120
Scaling Memcached Horizontally	121
Scaling Memcached Vertically	121
Configuring Engine Parameters Using Parameter Groups	123
Parameter Management	124
Parameter Group Tiers	125
Creating a Parameter Group	125
Listing Parameter Groups by Name	129
Listing a Parameter Group's Values	133
Modifying a Parameter Group	134
Deleting a Parameter Group	137
Memcached Specific Parameters	139
Securing Network Access	149
Amazon VPCs and ElastiCache Security	149
Subnets and Subnet Groups	164
Security Groups: EC2-Classic	172
Authentication and Access Control	180
Authentication	181
Access Control	182
Overview of Managing Access	183
Using Identity-Based Policies (IAM Policies)	187
Using Service-Linked Roles	191
ElastiCache API Permissions Reference	198
Monitoring Usage, Events, and Costs	201
Monitoring Use	202
Dimensions for ElastiCache Metrics	202
Host-Level Metrics	202
Metrics for Memcached	203
Which Metrics Should I Monitor?	206
Choosing Metric Statistics and Periods	207

Monitoring CloudWatch Cluster and Node Metrics	207
Monitoring Events	210
Managing ElastiCache Amazon SNS Notifications	210
Viewing ElastiCache Events	213
Event Notifications and Amazon SNS	215
Monitoring Costs with Tags	219
Managing Tags Using the Console	220
Managing Tags Using the AWS CLI	221
Managing Tags Using the ElastiCache API	223
Managing Costs with Reserved Nodes	226
Understanding Utilization Levels	226
Getting Info About Reserved Node Offerings	228
Purchasing a Reserved Node	231
Getting Info About Your Reserved Nodes	234
Reference	236
Using the ElastiCache API	236
Using the Query API	236
Available Libraries	238
Troubleshooting Applications	239
Logging ElastiCache API Calls with AWS CloudTrail	239
Set Up the AWS CLI for ElastiCache	243
Prerequisites	243
Getting the Command Line Tools	244
Setting Up the Tools	244
Providing Credentials for the Tools	245
Environmental Variables	246
Error Messages	246
Notifications	247
General ElastiCache Notifications	247
ElastiCache for Memcached Notifications	247
ElastiCache for Memcached Documentation History	249
AWS Glossary	255

What Is Amazon ElastiCache for Memcached?

Welcome to the *Amazon ElastiCache for Memcached User Guide*. 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, while removing the complexity associated with deploying and managing a distributed cache environment.

Guide Notice

The Amazon ElastiCache documentation has been reorganized so that the information for the Memcached and Redis engines is in separate guides. Use the guide for the engine that you're interested in.

Note

If you're unsure which engine you want to use, see [Comparing Memcached and Redis \(p. 12\)](#) in this guide.



PDF links for ElastiCache for Memcached

- ElastiCache for Memcached User Guide – This guide
- [ElastiCache section of the AWS CLI Reference](#)
- [ElastiCache API Reference](#)



PDF links for ElastiCache for Redis

- [ElastiCache for Redis User Guide](#)
- [ElastiCache section of the AWS CLI Reference](#)
- [ElastiCache API Reference](#)

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

- [Use Cases and How ElastiCache Can Help \(p. 3\)](#)

- [ElastiCache for Memcached Resources \(p. 5\)](#)
- [ElastiCache for Memcached Components and Features \(p. 7\)](#)
- [Tools for Managing Your Implementation \(p. 11\)](#)

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 significantly affected by the speed at which you deliver content. According to the New York Times in 2012, in "[For Impatient Web Users, an Eye Blink Is Just Too Long to Wait](#)," users can register a 250-millisecond (1/4 second) difference between competing sites. They tend to opt out of the slower site in favor of the faster site. Tests done at Amazon in 2007, 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, whether for a webpage or a report that drives business decisions, you can deliver that data much faster if it's cached. Can your business afford to not cache your webpages so as to deliver them with the shortest latency possible?

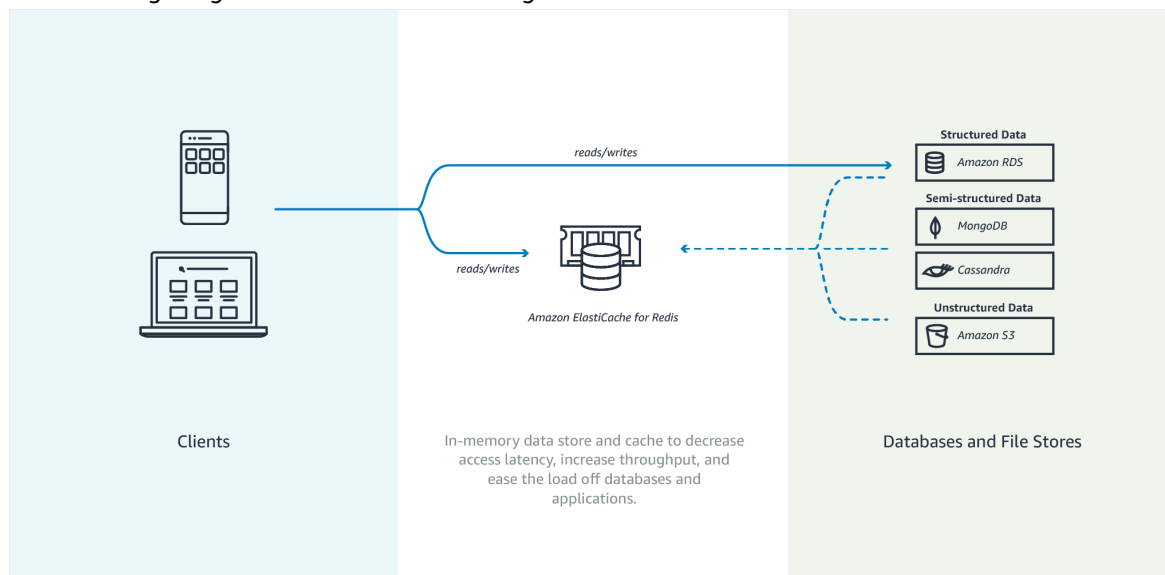
It might be intuitively obvious that you want to cache your most heavily requested items. But why not cache your less frequently requested items? Even the most optimized database query or remote API call is going to be noticeably slower than retrieving a flat key from an in-memory cache. *Noticeably 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, for example, queries that involve joins across multiple tables or queries with intensive calculations. By caching such query results, you pay the price of the query once and then are able to quickly retrieve the data multiple times without having to re-execute the query.

The following image shows ElastiCache caching.



What Should I Cache?

When deciding what data to cache, consider these factors:

Speed and Expense – It's always slower and more expensive to acquire 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 significantly slower and more expensive than simple, single table queries. If the interesting data requires a slow and expensive query to acquire, it's a candidate for caching. If acquiring 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 is rapidly changing or is seldom accessed. For caching to provide a meaningful benefit, the data should be relatively static and frequently accessed, such as a personal profile on a social media site. Conversely, 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 the results of a search because such queries and results are almost always 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. In determining whether your data is a candidate for caching, you need to 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, when serving a publicly traded stock price on a website, staleness might be acceptable, with a disclaimer that prices might be up to *n* minutes delayed. But when serving up the price for the same stock to a broker making a sale or purchase, you want real-time data.

In summary, consider caching your data if the following is true:

- It is slow or expensive to acquire when compared to cache retrieval.
- It is accessed with sufficient frequency.
- It is relatively static, or if rapidly changing, staleness is not a significant issue.

For more information, see the following:

- <https://docs.aws.amazon.com/AmazonElastiCache/latest/mem-ug/Strategies.html> in the ElastiCache for Memcached User Guide

ElastiCache Customer Testimonials

To learn about how businesses like Airbnb, PBS, Esri, and others are using Amazon ElastiCache to grow their businesses with improved customer experience, see [How Others Use Amazon ElastiCache](#).

You can also watch the [ElastiCache Videos \(p. 25\)](#) 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 \(p. 25\)](#) 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 \(p. 16\)](#) 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 \(p. 81\)](#)

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 \(p. 27\)](#)

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 \(p. 236\)](#)

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 \(p. 7\)](#)
- [ElastiCache for Memcached Clusters \(p. 8\)](#)
- [AWS Regions and Availability Zones \(p. 9\)](#)
- [ElastiCache for Memcached Endpoints \(p. 9\)](#)
- [ElastiCache Parameter Groups \(p. 9\)](#)
- [ElastiCache Security \(p. 9\)](#)
- [ElastiCache Security Groups \(p. 10\)](#)
- [ElastiCache Subnet Groups \(p. 10\)](#)
- [ElastiCache for Memcached Events \(p. 10\)](#)

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 \(p. 120\)](#).

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 \(p. 49\)](#).

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

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 Memcached Cluster \(p. 56\)](#).

For more information on nodes, see [Managing Nodes \(p. 45\)](#).

ElastiCache for Memcached Clusters

A Memcached *cluster* is a logical grouping of one or more [ElastiCache Nodes \(p. 7\)](#). 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 Your ElastiCache Clusters \(p. 53\)](#) and [Managing Nodes \(p. 45\)](#)

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 \(p. 32\)](#)

Information about improving the fault tolerance of your clusters.

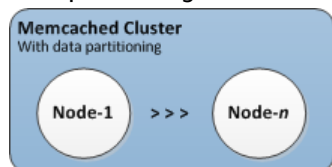
Typical Cluster Configurations

Memcached supports up to 100 nodes per customer for each AWS Region with each cluster having 1–20 nodes. You partition your data across the nodes in a Memcached cluster.

When you run the Memcached engine, clusters can be made up of 1–20 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 Memcached Cluster \(p. 56\)](#).

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 \(p. 32\)](#).

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 \(p. 34\)](#). In the following diagram, you can see examples of single node and multiple node Memcached clusters.

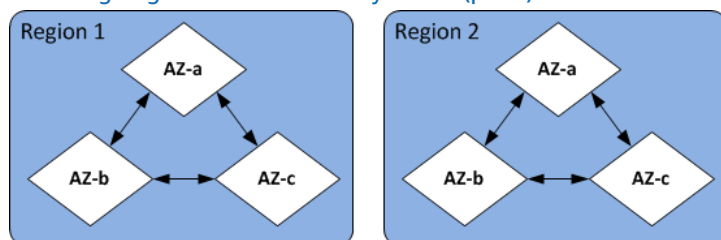


AWS Regions and Availability Zones

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 \(p. 39\)](#).



For information on AWS Regions supported by ElastiCache and their endpoints, see [Supported Regions & Endpoints \(p. 41\)](#).

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 Memcached Cluster \(p. 56\)](#).

For more information, see [Finding Connection Endpoints \(p. 114\)](#).

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 \(p. 123\)](#).

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 subnet groups or 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. 149).

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. 172).

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. 149), [Step 2: Authorize Access](#) (p. 20), and [Subnets and Subnet Groups](#) (p. 164).

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. 210).

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. 236).

See also

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

- [Managing Your ElastiCache for Memcached Implementation](#) (p. 36)
- [Securing Network Access](#) (p. 149)
- [Monitoring Usage, Events, and Costs](#) (p. 201)

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, such as a database.

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

• 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:

- [Redis Version 3.2.10 \(Enhanced\)](#)
- Online resharding best practices for Redis, for more information, see:
 - [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 compliance certification. For more information, see [HIPAA Compliance 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 15 node groups (clustered mode only).
- You need geospatial indexing (clustered mode or non-clustered mode).
- You don't need to support multiple databases.

Important

Redis (cluster mode enabled) has the following limitations:

- No scale-up to larger node types
- No changing the number of replicas in a node group (partition)

Redis version 3.2.4 (cluster mode enabled) has the following limitations:

- No dynamic scale-up to larger node types.
- No dynamic changing the number of replicas in a node group (partition).

- **ElastiCache for Redis (non-clustered mode) 2.8x and 3.2.4 (Enhanced)**

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

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

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

	Memcached	Redis (cluster mode disabled)	Redis (cluster mode enabled)
Engine versions	1.4.x	2.8.x and later	3.2.x and later
Data types	Simple	2.8.x - Complex *	3.2.x and later - Complex
		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
Compliance certifications			
Compliance Certification			
FedRAMP	No	3.2.6, 4.0.10 and later	3.2.6, 4.0.10 and later
HIPAA	No	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	No
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	Yes
		3.2.x and later - Yes	
Notes:			
string, objects (like databases)			
* string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog			
string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog, geospatial indexes			

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. 16)
- [Deploy a Memcached Cluster](#) (p. 17)
- [Where Do I Go From Here?](#) (p. 23)

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. 16)
- [Set Up Your Permissions \(New ElastiCache Users Only\)](#) (p. 16)

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://aws.amazon.com/>, and then choose **Create an AWS Account**.

Note

If you previously signed in to the AWS Management Console using AWS account root user credentials, choose **Sign in to a different account**. If you previously signed in to the console using IAM credentials, choose **Sign-in using root account credentials**. Then choose **Create a new AWS account**.

2. Follow the online instructions.

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

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 \(p. 188\)](#)
- [Using Service-Linked Roles for ElastiCache \(p. 191\)](#)

Deploy a Memcached Cluster

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

Topics

- [Determine Your Cluster's Requirements \(p. 17\)](#)
- [Step 1: Launch a Memcached Cluster \(p. 18\)](#)
- [Step 2: Authorize Access \(p. 20\)](#)
- [Step 3: Connect to a Cluster's Node \(p. 21\)](#)
- [Step 4: Delete Your Cluster \(Avoid Unnecessary Charges\) \(p. 22\)](#)

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 [Determine Your Requirements \(p. 54\)](#).

Step 1: Launch a Memcached 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 \(p. 165\)](#).

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. Choose **Get Started Now**.

If you already have an available cluster, choose **Launch Cluster**.

3. From the list in the upper-right corner, choose the AWS Region you want to launch this cluster in.
4. For **Cluster engine**, choose **Memcached**.
5. Complete the **Memcached settings** section as follows:
 - a. In **Name**, type a name for your cluster.

Cluster naming constraints

- Must contain from 1 to 20 alphanumeric characters or hyphens.
 - Must begin with a letter.
 - Cannot contain two consecutive hyphens.
 - Cannot end with a hyphen.
- b. From the **Engine version compatibility** list, choose the Memcached engine version you want to run on this cluster. Unless you have a specific reason to run an older version, we recommend that you choose the latest version.
 - c. In **Port**, accept the default port, 11211. If you have a reason to use a different port, type in the port number.
 - d. From **Parameter group**, 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 \(p. 125\)](#).

- e. 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 \(p. 81\)](#).

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**.

- f. From the **Number of nodes** list, choose the number of nodes (partitions) you want provisioned for this cluster.
6. Choose **Advanced Memcached settings** and complete the section as follows:
 - a. From the **Subnet group** list, choose the subnet you want to apply to this cluster. For this exercise, accept the default subnet group.

For more information, see [Subnets and Subnet Groups \(p. 164\)](#).
 - b. For **Availability zone(s)**, 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 \(p. 39\)](#).
 - c. From the **Security groups** list, choose the security groups that you want to use for this cluster. For this exercise, accept the default security group.

For more information, see [Amazon VPCs and ElastiCache Security \(p. 149\)](#).
 - d. 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.
After the cluster is created, you can modify it to specify a difference maintenance window. For more information, see [Managing Maintenance \(p. 44\)](#).
 - e. For **Notifications**, leave it as *Disable notifications*.
7. Choose **Create** to launch your cluster, or **Cancel** to cancel the operation.

Step 2: Authorize Access

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:

- [Access Patterns for Accessing an ElastiCache Cluster in an Amazon VPC \(p. 154\)](#)
- [Accessing ElastiCache Resources from Outside AWS \(p. 112\)](#)

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 \(p. 108\)](#).

Step 3: Connect to a Cluster's Node

Before you continue, be sure you have completed [Step 2: Authorize Access \(p. 20\)](#).

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 2: Authorize Access \(p. 20\)](#).

Step 3.1: Find Your Node Endpoints

When your cluster is in the *available* state and you've authorized access to it ([Step 2: Authorize Access \(p. 20\)](#)), 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 Step 3.2.

- [Finding Connection Endpoints \(p. 114\)](#)
- [Finding a Cluster's Endpoints \(Console\) \(p. 116\)](#)—You need the cluster's Configuration endpoint.
- [Finding Endpoints \(AWS CLI\) \(p. 118\)](#)
- [Finding Endpoints \(ElastiCache API\) \(p. 120\)](#)

Step 3.2: Connect to a Memcached Cluster

Once your cluster is in the *available* state and you've authorized access to it ([Step 2: Authorize Access \(p. 20\)](#)), 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 \(p. 46\)](#).

Step 4: 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 1: Launch a Memcached 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.

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 \(p. 81\)](#)

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 \(p. 27\)](#)

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

ElastiCache Tutorials and Videos

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

- [ElastiCache Videos \(p. 25\)](#)
- [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 \(p. 25\)](#)
- [Advanced Videos \(p. 25\)](#)

Introductory Videos

The following videos introduce you to Amazon ElastiCache.

Topics

- [DAT204—Building Scalable Applications on AWS NoSQL Services \(re:Invent 2015\) \(p. 25\)](#)
- [DAT207—Accelerating Application Performance with Amazon ElastiCache \(AWS re:Invent 2013\) \(p. 25\)](#)

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

- [DAT305—Amazon ElastiCache Deep Dive \(re:Invent 2017\) \(p. 26\)](#)
- [DAT306—Amazon ElastiCache Deep Dive \(re:Invent 2016\) \(p. 26\)](#)
- [DAT407—Amazon ElastiCache Deep Dive \(re:Invent 2015\) \(p. 26\)](#)

- [SDD402—Amazon ElastiCache Deep Dive \(re:Invent 2014\)](#) (p. 26)
- [DAT307—Deep Dive into Amazon ElastiCache Architecture and Design Patterns \(re:Invent 2013\)](#) (p. 26)

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

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 \(p. 27\)](#)
- [Mitigating Failures \(p. 32\)](#)
- [Configuring Your ElastiCache Client for Efficient Load Balancing \(p. 34\)](#)

Caching Strategies

This topic covers strategies for populating and maintaining your cache.

The strategy or strategies you want to implement for populating and maintaining your cache depend upon what data you are caching and the access patterns to that data. For example, you likely would not want to use the same strategy for both a Top-10 leaderboard on a gaming site, Facebook posts, and trending news stories. In the remainder of this section we discuss common cache maintenance strategies, their advantages, and their disadvantages.

Topics

- [Lazy Loading \(p. 27\)](#)
- [Write Through \(p. 29\)](#)
- [Adding TTL \(p. 30\)](#)
- [Related Topics \(p. 31\)](#)

Lazy Loading

As the name implies, lazy loading is a caching strategy that loads data into the cache only when necessary.

How Lazy Loading Works

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 does not exist in the cache, or the data in the cache has expired, your application requests the data from your data store which returns the data to your application. Your application then writes the data received from the store to the cache so it can be more quickly retrieved next time it is requested.

Scenario 1: Cache Hit

When data is in the cache and isn't expired

1. Application requests data from the cache.

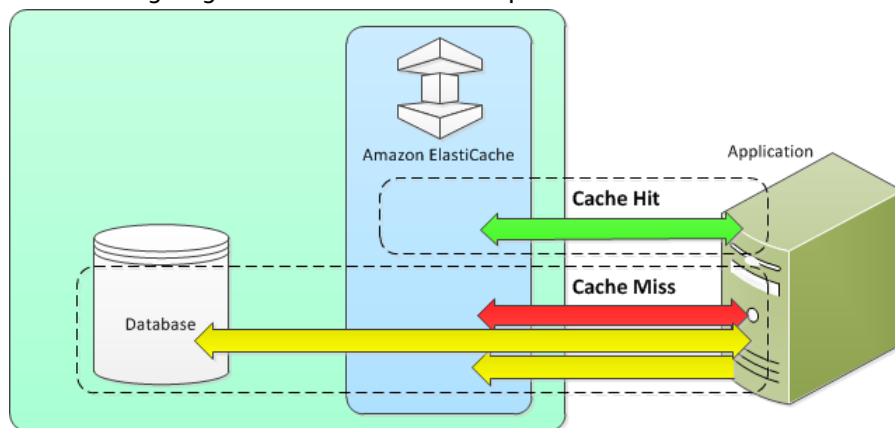
2. Cache returns the data to the application.

Scenario 2: Cache Miss

When data isn't in the cache or is expired

1. Application requests data from the cache.
2. Cache doesn't have the requested data, so returns a null.
3. Application requests and receives the data from the database.
4. Application updates the cache with the new data.

The following diagram illustrates both these processes.



Advantages and Disadvantages of Lazy Loading

Advantages of Lazy Loading

- Only requested data is cached.

Since most data is never requested, lazy loading avoids filling up the cache with data that isn't requested.

- Node failures are not fatal.

When a node fails and is replaced by a new, empty node the 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 and adding the data copy to the cache so that subsequent requests are retrieved from the cache.

Disadvantages of Lazy Loading

- There is a cache miss penalty.

Each cache miss results in 3 trips,

1. Initial request for data from the cache
2. Query of the database for the data
3. Writing the data to the cache

which can cause a noticeable delay in data getting to the application.

- Stale data.

If data is only written to the cache when there is a cache miss, data in the cache can become stale since there are no updates to the cache when data is changed in the database. This issue is addressed by the [Write Through \(p. 29\)](#) and [Adding TTL \(p. 30\)](#) strategies.

Lazy Loading Code

The following code is a pseudo code 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
```

The application code that retrieves the data would be:

```
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

Advantages of Write Through

- Data in the cache is never stale.

Since the data in the cache is updated every time it is 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.

Disadvantages of Write Through

- Missing data.

In the case of spinning up a new node, whether due to a node failure or scaling out, there is missing data which continues to be missing until it is added or updated on the database. This can be minimized by implementing [Lazy Loading \(p. 27\)](#) in conjunction with Write Through.

- Cache churn.

Since most data is never read, there can be a lot of data in the cluster that is never read. This is a waste of resources. By [Adding TTL \(p. 30\)](#) you can minimize wasted space.

Write Through Code

The following code is a pseudo code 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
```

The application code that updates the data would be:

```
save_customer(12345,{"address":"123 Main"})
```

Adding TTL

Lazy loading allows for stale data, but won't fail with empty nodes. Write through ensures that data is always fresh, but may fail with empty nodes and may populate the cache with superfluous data. By adding a time to live (TTL) value to each write, we are able to enjoy the advantages of each strategy and largely avoid cluttering up the cache with superfluous data.

What is TTL?

Time to live (TTL) is an integer value that specifies the number of seconds until the key expires. When an application attempts to read an expired key, it is treated as though the key is not found, meaning that the database is queried for the key and the cache is updated. This does not guarantee that a value is not stale, but 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](#).

Code Example

The following code is a pseudo code 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 code is a pseudo code 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
```

The application code would be:

```
save_customer(12345, {"address": "123 Main"})
```

```
customer_record = get_customer(12345)
```

Related Topics

- [In-Memory Data Store \(p. 3\)](#)
- [Choosing an Engine and Version](#)
- [Scaling ElastiCache for Memcached Clusters \(p. 120\)](#)

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 \(p. 32\)](#)
- [Recommendations \(p. 33\)](#)

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 20 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 20 nodes in a Memcached cluster, or more than 100 nodes total in an AWS Region, please 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 \(Console\) \(p. 83\)](#).

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 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 \(Console\) \(p. 83\)](#).

For more information on regions and Availability Zones, see [Choosing Regions and Availability Zones](#) (p. 39).

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

To minimize the impact of a node failure, we recommend that your implementation use multiple nodes in each shard and distribute the nodes across multiple availability zones.

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.

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 <http://dustin.sallings.org/java-memcached-client/apidocs/net/spy/memcached/KetamaConnectionFactory.html>.

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>.

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, upgrading your engine, scaling your implementation to meet changing business needs, monitoring your use and costs, and securing your data.

Topics

- [Engine Versions and Upgrading \(p. 36\)](#)
- [Choosing Regions and Availability Zones \(p. 39\)](#)
- [Managing Maintenance \(p. 44\)](#)
- [Managing Nodes \(p. 45\)](#)
- [Managing Your ElastiCache Clusters \(p. 53\)](#)
- [Accessing Your Cluster \(p. 108\)](#)
- [Finding Connection Endpoints \(p. 114\)](#)
- [Scaling ElastiCache for Memcached Clusters \(p. 120\)](#)
- [Configuring Engine Parameters Using Parameter Groups \(p. 123\)](#)
- [Securing Network Access \(p. 149\)](#)
- [Authentication and Access Control for Amazon ElastiCache \(p. 180\)](#)

Engine Versions and Upgrading

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

Topics

- [Supported ElastiCache for Memcached Versions \(p. 37\)](#)
- [Upgrading Engine Versions \(p. 39\)](#)

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 1.5.10](#) (p. 37)
- [Memcached Version 1.4.34](#) (p. 37)
- [Memcached Version 1.4.33](#) (p. 37)
- [Memcached Version 1.4.24](#) (p. 38)
- [Memcached Version 1.4.14](#) (p. 38)
- [Memcached Version 1.4.5](#) (p. 38)

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. 139).

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 \(p. 140\)](#).

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 \(p. 142\)](#).
- 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 \(p. 92\)](#).

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 \(p. 146\)](#).

Important

- You can upgrade to a newer engine version, but you can't downgrade to an older engine version. If you want to use an older engine version, you must delete the existing cluster and create it anew with the older engine version.
- Although engine version management functionality is intended to give you as much control as possible over how patching occurs, ElastiCache reserves the right to patch your cluster on your behalf in the unlikely event of a critical security vulnerability in the system or cache software.
- Because the Memcached engine does not support persistence, Memcached engine version upgrades are always a disruptive process which clears all cache data in the cluster.

How to Upgrade Engine Versions

You initiate version upgrades to your cluster by modifying it using the ElastiCache console, the AWS CLI, or the ElastiCache API and specifying a newer engine version. For more information, see the following topics.

- **Using the AWS Management Console** – [Using the AWS Management Console \(p. 92\)](#)
- **Using the AWS CLI** – [Using the AWS CLI \(p. 92\)](#)
- **Using the ElastiCache API** – [Using the ElastiCache API \(p. 93\)](#)

Choosing Regions and Availability Zones

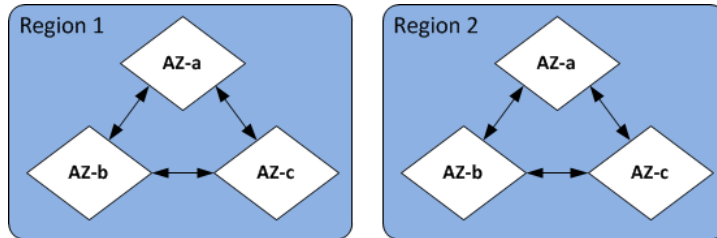
AWS cloud computing resources are housed in highly available data center facilities. To provide additional scalability and reliability, these data center facilities are located in different physical locations. These locations are categorized by *regions* and *Availability Zones*.

Regions are large and widely dispersed into separate geographic locations. Availability Zones are distinct locations within a region that are engineered to be isolated from failures in other Availability Zones and provide inexpensive, low latency network connectivity to other Availability Zones in the same 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 \(p. 41\)](#).



Regions and Availability Zones

Topics

- [Supported Regions & Endpoints \(p. 41\)](#)
- [Locating Your Nodes \(p. 44\)](#)

Supported Regions & Endpoints

Amazon ElastiCache is available in multiple regions so that you can launch ElastiCache clusters in locations that meet your requirements, such as launching in the 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 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 \(p. 39\)](#) 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 (Mumbai) Region ap-south-1	elasticache.ap-south-1.amazonaws.com	HTTPS	
Asia Pacific (Tokyo) Region ap-northeast-1	elasticache.ap-northeast-1.amazonaws.com	HTTPS	
Asia Pacific (Seoul) Region ap-northeast-2	elasticache.ap-northeast-2.amazonaws.com	HTTPS	
Asia Pacific (Osaka-Local) Region * ap-northeast-3	elasticache.ap-northeast-3.amazonaws.com	HTTPS	

Amazon ElastiCache ElastiCache
for Memcached User Guide
Supported Regions & Endpoints

Region Name/Region	Endpoint	Protocol	
Asia Pacific (Singapore) Region ap-southeast-1	elasticache.ap-southeast-1.amazonaws.com	HTTPS	
Asia Pacific (Sydney) Region ap-southeast-2	elasticache.ap-southeast-2.amazonaws.com	HTTPS	
EU (Frankfurt) Region eu-central-1	elasticache.eu-central-1.amazonaws.com	HTTPS	
EU (Ireland) Region eu-west-1	elasticache.eu-west-1.amazonaws.com	HTTPS	
EU (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	
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	
AWS GovCloud (US-West) us-gov-west-1	elasticache.us-gov-west-1.amazonaws.com	HTTPS	
For information on using the AWS GovCloud (US) with ElastiCache, see Services in the AWS GovCloud (US) region: ElastiCache .			
Notes: The Asia Pacific (Osaka-Local) Region is a local region that is available to select AWS customers who request access. Customers wishing to use the Asia Pacific (Osaka-Local) Region should speak with their sales representative. The Asia Pacific (Osaka-Local) Region supports a single Availability Zone.			

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 \(p. 50\)](#).

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:

- [Creating a Cluster \(p. 54\)](#)
- [Modifying an ElastiCache Cluster \(p. 92\)](#)
- [Adding Nodes to a Cluster \(p. 95\)](#)

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-Local) Region	12:00–20:00 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
ap-southeast-2	Asia Pacific (Sydney) Region	12:00–20:00 UTC
cn-north-1	China (Beijing) region	14:00–22:00 UTC
eu-central-1	EU (Frankfurt) Region	23:00–07:00 UTC
eu-west-1	EU (Ireland) Region	22:00–06:00 UTC
eu-west-2	EU (London) Region	23:00–07:00 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

Region Code	Region Name	Region Maintenance Window
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.

More information

For information on your maintenance window and node replacement, see:

- [ElastiCache Maintenance](#)—FAQ on maintenance and node replacement
- [Replacing Nodes \(p. 52\)](#)—Managing node replacement
- [Modifying an ElastiCache Cluster \(p. 92\)](#)—Changing a cluster's maintenance window

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.

Generally speaking, due to its support for sharding, Memcached deployments will have a number of smaller nodes. See [Choosing Your Memcached Node Size \(p. 81\)](#) for a more detailed discussion of which node size to use.

Topics

- [Connecting to Nodes \(p. 46\)](#)
- [ElastiCache Reserved Nodes \(p. 48\)](#)
- [Supported Node Types \(p. 49\)](#)
- [Replacing Nodes \(p. 52\)](#)

Other ElastiCache Node Operations

Additional operations involving nodes:

- [Adding Nodes to a Cluster \(p. 95\)](#)
- [Removing Nodes from a Cluster \(p. 100\)](#)
- [Scaling ElastiCache for Memcached Clusters \(p. 120\)](#)
- [Finding Connection Endpoints \(p. 114\)](#)
- [Automatically Identify Nodes in your Memcached Cluster \(p. 56\)](#)

Connecting to Nodes

Before attempting to connect to your Memcached cluster, you must have the endpoints for the nodes. To find the endpoints, see:

- [Finding a Cluster's Endpoints \(Console\) \(p. 116\)](#)
- [Finding Endpoints \(AWS CLI\) \(p. 118\)](#)
- [Finding Endpoints \(ElastiCache API\) \(p. 120\)](#)

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  
>
```

ElastiCache 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 – 1 or 3 years. In addition to the up front charge there is an hourly usage charge which is significantly less than the hourly usage charge you incur with On-Demand nodes. To determine whether reserved nodes would be a cost savings for your use cases, determine the node size and number of nodes you need, estimate the usage of the node, then 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](#).

For more information, see [Managing Costs with Reserved Nodes \(p. 226\)](#).

Supported Node Types

The following node types are supported by ElastiCache. Generally speaking, the current generation types provide more memory and computational power at lower cost when compared to their equivalent previous generation counterparts.

- General purpose:

- Current generation:

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`

T2 node types: `cache.t2.micro`, `cache.t2.small`, `cache.t2.medium`

- Previous generation: (not recommended)

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:

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`

Additional node type info

- All current generation instance types are created in Amazon VPC by default.

Supported engine versions

Supported engine versions vary by region. The latest engine versions are supported in all regions. To find the available engine versions in your region see [Supported ElastiCache for Memcached Versions \(p. 37\)](#).

Supported Node Types by AWS Region

AWS Region	T2	M4	M5	R4	R5
US-east-2 East (Ohio)	Yes	Yes	Yes	Yes	Yes
US-east-1 East (N. Virginia)	Yes	Yes	Yes	Yes	Yes
US-west-1 West (N. California)	Yes	Yes	Yes	Yes	Yes
US-west-2 West (Oregon)	Yes	Yes	Yes	Yes	Yes
Canada-central-1 (Central)	Yes	Yes	Yes	Yes	Yes
Asia-south-1 Pacific (Mumbai)	Yes	Yes	Yes	Yes	
Asia-northeast-1 Pacific (Tokyo)	Yes	Yes	Yes	Yes	Yes
Asia-northeast-2 Pacific (Seoul)	Yes	Yes	Yes	Yes	Yes
Asia-northeast-3 Pacific (Osaka-Local) *	Yes			Yes	
Asia-southeast-1 Pacific (Singapore)	Yes	Yes	Yes	Yes	Yes
Asia-southeast-2 Pacific (Sydney)	Yes	Yes	Yes	Yes	Yes
EU-central-1 (Frankfurt)	Yes	Yes	Yes	Yes	Yes
EU-west-1 (Ireland)	Yes	Yes	Yes	Yes	Yes
EU-west-2 (London)	Yes	Yes	Yes	Yes	Yes

Amazon ElastiCache ElastiCache
for Memcached User Guide
Supported Node Types

AWS Region	T2	M4	M5	R4	R5
EU-west-3 (Paris)	Yes		Yes	Yes	Yes
South-east-1 America (São Paulo)	Yes	Yes	Yes	Yes	
China-north-1 (Beijing)	Yes	Yes		Yes	
China-northwest-1 (Ningxia)	Yes	Yes		Yes	
AWS Gov-west-1 GovCloud (US-West)	Yes		Yes	Yes	Yes
* The Asia Pacific (Osaka-Local) Region is a local region that is available to select AWS customers who request access. Customers wishing to use the Asia Pacific (Osaka-Local) Region should speak with their sales representative. The Asia Pacific (Osaka-Local) Region supports a single Availability Zone.					

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 prior to the scheduled node replacement window. When you manage a replacement yourself, your instance will receive the OS update when you relaunch the node and your scheduled node replacement will be cancelled. You may continue to receive alerts indicating that the node replacement will take place. If you've already manually mitigated the need for the maintenance, you can ignore these alerts.

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 will replace 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 where you receive an email or a notification event from ElastiCache, if you change your maintenance window before the scheduled replacement time, your node will now be replaced at the new time. For more information, see [Modifying an ElastiCache Cluster \(p. 92\)](#).

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, currently it's Thursday, 11/09 at 1500 and the next maintenance window is Friday, 11/10, at 1700. Following are 3 scenarios with their outcomes:

- You change your maintenance window to Fridays at 1600 (after the current datetime and before the next scheduled maintenance window). The node will be replaced on Friday, 11/10, at 1600.
- You change your maintenance window to Saturday at 1600 (after the current datetime and after the next scheduled maintenance window). The node will be replaced on Saturday, 11/11, at 1600.
- You change your maintenance window to Wednesday at 1600 (earlier in the week than the current datetime). The node will be replaced next Wednesday, 11/15, at 1600.
-

For instructions, see [Managing Maintenance \(p. 44\)](#).

- **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 will be redistributed which will cause cache misses.

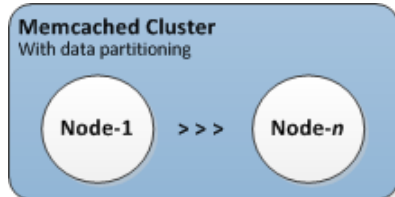
To manually replace a Memcached node

1. Delete the node scheduled for replacement. For instructions, see [Removing Nodes from a Cluster \(p. 100\)](#).
2. Add a new node to the cluster. For instructions, see [Adding Nodes to a Cluster \(p. 95\)](#).
3. If you are not using [Automatically Identify Nodes in your Memcached Cluster \(p. 56\)](#) on this cluster, see your application and replace every instance of the old node's endpoint with the new node's endpoint.

Managing Your ElastiCache 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 that all of the nodes will use.

The following diagram illustrates a typical Memcached cluster. Memcached clusters contain from 1 to 20 nodes across which you horizontally partition your data.



Typical Memcached Cluster

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 always using the your engine's latest version.

ElastiCache clusters are designed to be accessed via an Amazon EC2 instance. If you launch your cluster in an Amazon VPC, you can access it from outside AWS. For more information, see:

- [Step 2: Authorize Access \(p. 20\)](#)
- [Accessing ElastiCache Resources from Outside AWS \(p. 112\)](#)

Supported Memcached Versions

- [Memcached Version 1.4.34 \(p. 37\)](#)
- [Memcached Version 1.4.33 \(p. 37\)](#)
- [Memcached Version 1.4.24 \(p. 38\)](#)
- [Memcached Version 1.4.14 \(p. 38\)](#)
- [Memcached Version 1.4.5 \(p. 38\)](#)

Other ElastiCache Cluster Operations

Additional operations involving clusters:

- [Finding Connection Endpoints \(p. 114\)](#)
- [Accessing ElastiCache Resources from Outside AWS \(p. 112\)](#)

Creating a Cluster

In this section you will find instructions on creating a cluster using the ElastiCache console, the AWS CLI, or the ElastiCache API.

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.

Determine Your Requirements

Topics

- [Memory and Processor Requirements \(p. 55\)](#)
- [Memcached Cluster Configuration \(p. 55\)](#)
- [Scaling Requirements \(p. 55\)](#)
- [Access Requirements \(p. 55\)](#)
- [Region and Availability Zone Requirements \(p. 55\)](#)

Preparation

Knowing the answers to these questions before you begin will expedite creating your cluster.

- Which node instance type do you need?

For guidance on choosing an instance node type, see [Choosing Your Memcached Node Size \(p. 81\)](#).

- Will you launch your cluster in a VPC or an Amazon VPC?

Important

If you're going to launch your cluster in an Amazon VPC, you need to create a subnet group in the same VPC before you start creating a cluster. For more information, see [Subnets and Subnet Groups \(p. 164\)](#).

An advantage of launching in a Amazon VPC is that, though ElastiCache is designed to be accessed from within AWS using Amazon EC2, if your cluster is in an Amazon VPC you can provide access from outside AWS. For more information, see [Accessing ElastiCache Resources from Outside AWS \(p. 112\)](#).

- Do you need to customize any parameter values?

If you do, you need to create a custom Parameter Group. For more information, see [Creating a Parameter Group \(p. 125\)](#).

- Do you need to create your own *Security Group* or *VPC Security Group*?

For more information, see [Security Groups: EC2-Classical \(p. 172\)](#) and [Security in Your VPC](#).

- How do you intend to implement fault tolerance?

For more information, see [Mitigating Failures \(p. 32\)](#).

Topics

- [Memory and Processor Requirements \(p. 55\)](#)
- [Memcached Cluster Configuration \(p. 55\)](#)
- [Scaling Requirements \(p. 55\)](#)
- [Access Requirements \(p. 55\)](#)
- [Region and Availability Zone Requirements \(p. 55\)](#)

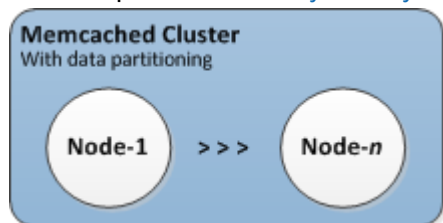
Memory and Processor Requirements

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 20 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* which your application can use 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 Memcached Cluster \(p. 56\)](#).



For more information, see [Managing Your ElastiCache Clusters \(p. 53\)](#).

Scaling Requirements

All clusters can be scaled up by creating a new cluster with the new, larger node type. When scaling up a Memcached cluster the new cluster will start 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 \(p. 120\)](#) 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 2: Authorize Access \(p. 20\)](#) in this guide.

Region and Availability Zone Requirements

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

For more information, see:

- [Choosing Regions and Availability Zones \(p. 39\)](#)
- [Mitigating Failures \(p. 32\)](#)

Automatically Identify Nodes in your Memcached 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.

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 \(p. 57\)](#)
- [How Auto Discovery Works \(p. 58\)](#)
- [Using Auto Discovery \(p. 61\)](#)
- [Connecting to Cache Nodes Manually \(p. 66\)](#)
- [Adding Auto Discovery To Your Client Library \(p. 67\)](#)
- [ElastiCache Clients with Auto Discovery \(p. 68\)](#)

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

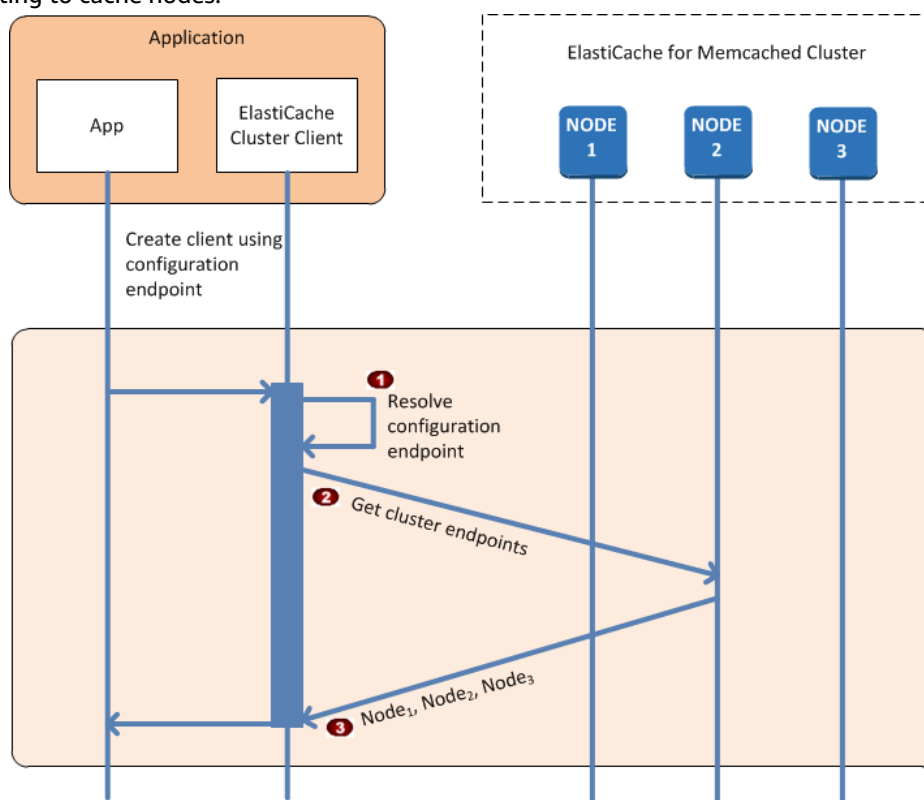
Topics

- [Connecting to Cache Nodes \(p. 58\)](#)
- [Normal Cluster Operations \(p. 59\)](#)
- [Other Operations \(p. 59\)](#)

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

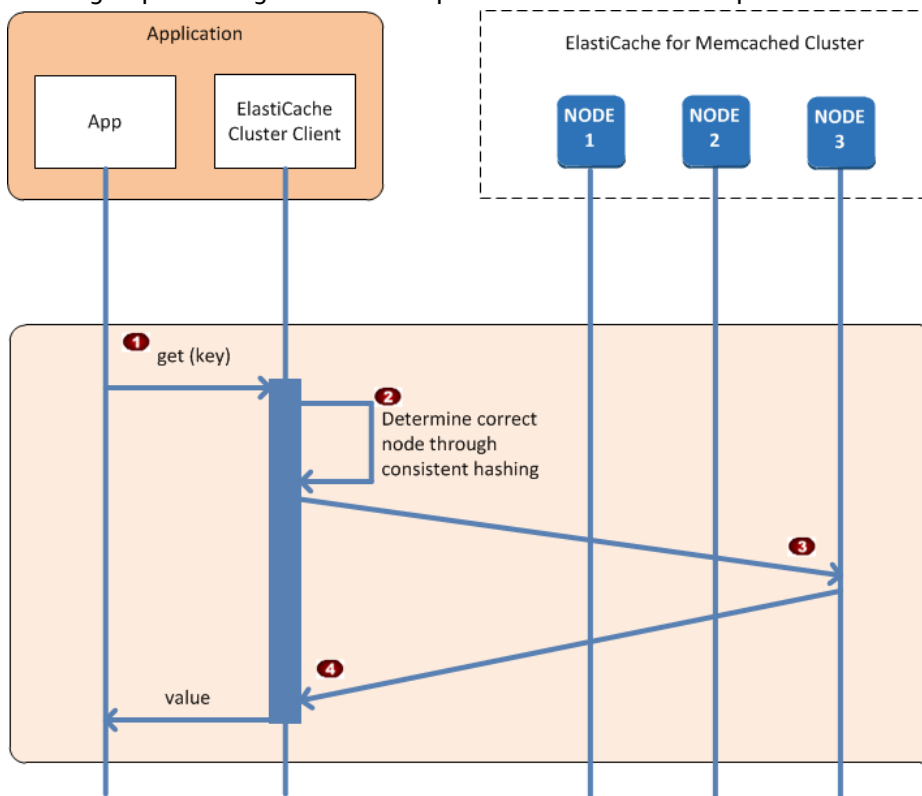
- 1 The application resolves the configuration endpoint's DNS name. Because the configuration endpoint maintains CNAME entries for all of the cache nodes, the DNS name resolves to one of the nodes; the client can then connect to that node.
- 2 The client requests the configuration information for all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

There may arise situations where there is a change in the cluster due to adding an additional node to accommodate additional demand, deleting a node to save money during periods of reduced demand, or replacing 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. Prior to 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 \(p. 61\)](#)
- [Step 2: Download the ElastiCache Cluster Client \(p. 62\)](#)
- [Step 3: Modify Your Application Program \(p. 62\)](#)

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 \(Console\) \(p. 116\)](#)

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.us-east-1e.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.us-east-1e.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.us-east-1.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 \(p. 71\)](#).

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 \(p. 69\)](#).

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.usel.cache.amazonaws.com";
```

Failure to explicitly specify the cluster's configuration endpoint results in configuring to a specific node.

Topics

- [Using the ElastiCache Cluster Client for Java \(p. 63\)](#)
- [Using the ElastiCache Cluster Client for PHP \(p. 63\)](#)
- [Using the ElastiCache Cluster Client for .NET \(p. 64\)](#)

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.usel.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 \(p. 71\)](#)

```
<?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.usel.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');
?>
```

Using the ElastiCache Cluster Client for .NET

.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.usel.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.us-east-1.cache.amazonaws.com:11211",  
        "mycachecluster.fnjyzo.0002.us-east-1.cache.amazonaws.com:11211",  
        "mycachecluster.fnjyzo.0003.us-east-1.cache.amazonaws.com:11211",  
        "mycachecluster.fnjyzo.0004.us-east-1.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 \(p. 68\)](#) 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 0 147\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 \(p. 68\)](#)
- [Configuring ElastiCache Clients \(p. 80\)](#)

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 \(p. 69\)](#)
- [Installing the ElastiCache Cluster Client for PHP \(p. 71\)](#)
- [Compiling the Source Code for the ElastiCache Cluster Client for PHP \(p. 78\)](#)

Installing the ElastiCache Cluster Client for .NET

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 Memcached Cluster \(p. 56\)](#). For sample .NET code to use the client, see [Using the ElastiCache Cluster Client for .NET \(p. 64\)](#).

Topics

- [Installing .NET \(p. 69\)](#)
- [Download the ElastiCache .NET Cluster Client for ElastiCache \(p. 69\)](#)
- [Install AWS Assemblies with NuGet \(p. 69\)](#)

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 Memcached Cluster \(p. 56\)](#). For sample PHP code to use the client, see [Using the ElastiCache Cluster Client for PHP \(p. 63\)](#).

Topics

- [Downloading the Installation Package \(p. 71\)](#)
- [For Users Who Already Have php-memcached Extension Installed \(p. 72\)](#)
- [Installation Steps for New Users \(p. 72\)](#)
- [Removing the PHP Cluster Client \(p. 77\)](#)

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 \(p. 78\)](#).

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 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 \(p. 77\)](#).
2. Install the new ElastiCache *php-memcached* extension as described previously in [Installation Steps for New Users \(p. 72\)](#).

Installation Steps for New Users

Topics

- [Installing PHP 7.x for New Users \(p. 72\)](#)
- [Installing PHP 5.x for New Users \(p. 74\)](#)

Installing PHP 7.x for New Users

Topics

- [To install PHP 7 on a Ubuntu Server 14.04 LTS AMI \(64-bit and 32-bit\) \(p. 72\)](#)
- [To install PHP 7 on an Amazon Linux 201609 AMI \(p. 73\)](#)
- [To install PHP 7 on an SUSE Linux AMI \(p. 73\)](#)

To install PHP 7 on a Ubuntu Server 14.04 LTS AMI (64-bit and 32-bit)

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

```
sudo apt-get update
sudo apt-get install gcc g++
```

3. Install PHP 7.

```
sudo yum install php70
```

4. Download the Amazon ElastiCache Cluster Client.

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.0/latest-64bit
```

5. Extract latest-64bit.

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

6. With root permissions, copy the extracted artifact file `amazon-elasticache-cluster-client.so` into `/usr/lib/php/20151012`.

```
sudo mv artifact/amazon-elasticache-cluster-client.so /usr/lib/php/20151012
```

7. Insert the line `extension=amazon-elasticache-cluster-client.so` into the file `/etc/php/7.0/cli/php.ini`.

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/php/7.0/cli/php.ini
```

8. Start or restart your Apache server.

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

To install PHP 7 on an Amazon Linux 201609 AMI

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

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

3. Install PHP 7.

```
sudo yum install php70
```

4. Download the Amazon ElastiCache Cluster Client.

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.0/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/7.0/modules/`.

```
sudo mv artifact/amazon-elasticache-cluster-client.so /usr/lib64/php/7.0/modules/
```

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

```
echo "extension=amazon-elasticache-cluster-client.so" | sudo tee --append /etc/php-7.0.d/50-memcached.ini
```

8. Start or restart your Apache server.

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

To install PHP 7 on an SUSE Linux AMI

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

```
sudo zypper install gcc
```

3. Install PHP 7.

```
sudo yum install php70
```

4. Download the Amazon ElastiCache Cluster Client.

```
wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.0/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/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
```

Installing PHP 5.x for New Users

Topics

- [To install PHP 5 on an Amazon Linux AMI 2014.03 \(64-bit and 32-bit\) \(p. 74\)](#)
- [To install PHP 5 on a Red Hat Enterprise Linux 7.0 AMI \(64-bit and 32-bit\) \(p. 75\)](#)
- [To install PHP 5 on a Ubuntu Server 14.04 LTS AMI \(64-bit and 32-bit\) \(p. 75\)](#)
- [To install PHP 5 for SUSE Linux Enterprise Server 11 AMI \(64-bit or 32-bit\) \(p. 76\)](#)
- [Other Linux distributions \(p. 77\)](#)

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 \(p. 71\)](#).
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

Please 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 \(p. 71\)](#).
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 \(p. 71\)](#).
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. Please verify the absolute path of the build artifact because it is needed by 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 \(p. 71\)](#).
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, please verify the install path for `amazon-elasticache-cluster-client.so`, and specify the full path of the binary in the extension. Also, verify that the PHP in use is a supported version. We support versions 5.3 through 5.5.

Other Linux distributions

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

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

Removing the PHP Cluster Client

Topics

- [Removing an earlier version of PHP 7 \(p. 77\)](#)
- [Removing an earlier version of PHP 5 \(p. 77\)](#)

Removing an earlier version of PHP 7

To remove an earlier version of PHP 7

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 \(p. 72\)](#).
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 \(p. 78\)](#)
- [Compiling the ElastiCache Memcached Auto Discovery Client for PHP \(p. 78\)](#)

Compiling the libmemcached Library

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

```
sudo yum install gcc gcc-c++ autoconf libevent-devel
```

- On Ubuntu 14.04 AMI

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

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  
mkdir BUILD  
cd BUILD  
../configure --prefix=<libmemcached-install-directory> --with-pic  
make  
sudo make install
```

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 \(p. 78\)](#)
- [Compiling the ElastiCache Memcached Client for PHP 5 \(p. 79\)](#)

Compiling the ElastiCache Memcached Client for PHP 7

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  
git checkout php7  
sudo yum install php70-devel  
phpize  
../configure --with-libmemcached-dir=<libmemcached-install-directory> --disable-memcached-sasl
```

```
make  
make install
```

Note

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

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

Compiling the ElastiCache Memcached Client for PHP 5

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

```
git clone https://github.com/awslabs/aws-elasticache-cluster-client-memcached-for-php.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 \(p. 80\)](#)
- [Connecting for Using Auto Discovery \(p. 81\)](#)
- [DNS Names and Underlying IP \(p. 81\)](#)

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 Memcached Cluster \(p. 56\)](#).

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.

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 back-end database while ElastiCache provisions a replacement for the failed node and it gets repopulated. You can reduce this potential availability impact by spreading your memory and compute capacity over a larger number of nodes, each with smaller capacity, rather than using a fewer number of high capacity nodes.

In a scenario where you want to have 35 GB of cache memory, you can set it up in 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	Cores	Hourly Cost *	Nodes Needed	Total Memory	Total Cores	Monthly Cost
<code>cache.t2.medium</code>	3.22 GB	2	\$ 0.068	11	35.42 GB	22	\$ 538.56
<code>cache.m4.large</code>	6.42 GB	2	\$ 0.156	6	38.52 GB	12	\$ 673.92
<code>cache.m4.xlarge</code>	14.28 GB	4	\$ 0.311	3	42.84 GB	12	\$ 671.76
<code>cache.m5.xlarge</code>	12.93 GB	4	\$ 0.311	3	38.81 GB	12	\$ 671.76
<code>cache.r4.large</code>	12.3 GB	2	\$ 0.228	3	36.9 GB	6	\$ 492.48
<code>cache.r5.large</code>	13.07 GB	2	\$ 0.216	3	39.22 GB	6	\$ 466.56
* Hourly cost per node as of October 22, 2018.							
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 \(p. 146\)](#)

Using multiple nodes will require 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 Memcached Cluster \(p. 56\)](#).

If you're unsure about how much capacity you need, for testing we recommend starting with one `cache.m5.large` node and monitoring the memory usage, CPU utilization, and cache hit rate with the ElastiCache metrics that are published to CloudWatch. For more information on CloudWatch metrics for ElastiCache, see [Monitoring Use with CloudWatch Metrics \(p. 202\)](#). For production and larger workloads, the R5 nodes provide the best performance and RAM cost value.

If your cluster does not have the desired hit rate, you can easily add more nodes, thereby increasing the total available memory in your cluster.

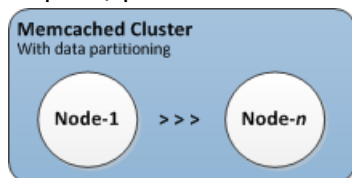
If your cluster turns out to be bound by CPU but it has sufficient hit rate, try setting up a new cluster with a node type that provides more compute power.

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 Memcached Cluster](#) (p. 56).

To create a Memcached cluster using the ElastiCache console:

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at <https://console.aws.amazon.com/elasticache/>.
2. From the list in the upper-right corner, choose the AWS Region you want to launch this cluster in.
3. Choose **Memcached** from the navigation pane.
4. Choose **Create**.
5. For **Cluster engine**, choose *Memcached*. Choosing Memcached will create a Memcached cluster that looks something like this. The number of nodes is determined by the number of nodes you choose in Step 5.f (up to a maximum of 20).



Memcached cluster with data partitioning

6. Complete the **Memcached settings** section.
 - a. In **Name**, type in a name for your cluster.

Cluster naming constraints

- Must contain from 1 to 20 alphanumeric characters or hyphens.
 - Must begin with a letter.
 - Cannot contain two consecutive hyphens.
 - Cannot end with a hyphen.
- b. For **Engine version compatibility**, choose the Memcached engine version you want this cluster to run. Unless you have a specific reason to run an older version, we recommend that you choose the latest version.

Important

You can upgrade to newer engine versions. For more information, see [Upgrading Engine Versions](#) (p. 39). Any change in Memcached engine versions is a disruptive process in which you lose your cluster data.

- c. In **Port**, accept the default port, 11211. If you have a reason to use a different port, type the port number.
- d. For **Parameter group**, choose the default parameter group, 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.

Parameter groups control the run-time parameters of your cluster. For more information on parameter groups, see [Memcached Specific Parameters](#) (p. 139) and [Creating a Parameter Group](#) (p. 125).

- e. For **Node type**, click the down arrow (▼). In the **Change node type** dialog box, choose the *Instance family* of the node type you want, choose the node type you want to use for this cluster, and then choose **Save**.

For more information, see [Choosing Your Memcached Node Size \(p. 81\)](#).

- f. For **Number of nodes**, choose the number of nodes you want for this cluster. You will partition your data across the cluster's nodes.

If you need to change the number of nodes later, scaling horizontally is quite easy with Memcached. For more information, see [Scaling ElastiCache for Memcached Clusters \(p. 120\)](#)

7. Click **Advanced Memcached settings** and complete the section.

- a. For **Subnet group**, choose the subnet you want to apply to this cluster.

For more information, see [Subnets and Subnet Groups \(p. 164\)](#).

- b. For **Availability zone(s)**, you have two options:

- **No preference** – ElastiCache chooses the Availability Zone for each node in your cluster.
- **Specify availability zones** – Specify the Availability Zone for each node in your cluster.

If you chose to specify the Availability Zones, for each node choose an Availability Zone from the list to the right of each node name.

We recommend locating your nodes in multiple Availability Zones for improved fault tolerance. For more information, see [Mitigating Availability Zone Failures \(p. 32\)](#).

For more information, see [Choosing Regions and Availability Zones \(p. 39\)](#).

- c. For **Security groups**, choose the security groups you want to apply to this cluster.

For more information, see [Amazon VPCs and ElastiCache Security \(p. 149\)](#).

- d. The **Maintenance window** is the time, generally an hour in length, each week when ElastiCache schedules system maintenance for your cluster. You can allow ElastiCache choose the day and time for your maintenance window (**No preference**), or you can choose the day, time, and duration yourself (**Specify maintenance window**). If you choose **Specify maintenance window**, choose the **Start day**, **Start time**, and **Duration** (in hours) for your maintenance window. All times are UCT times.

For more information, see [Managing Maintenance \(p. 44\)](#).

- e. For **Notifications**, choose an existing Amazon Simple Notification Service (Amazon SNS) topic, or choose manual ARN input and type in the topic Amazon Resource Name (ARN). Amazon SNS allows you to push notifications to Internet-connected smart devices. The default is to disable notifications. For more information, see <https://aws.amazon.com/sns/>.

8. Review all your entries and choices, then go back and make any needed corrections. When you're ready, choose **Create** to launch your 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 2: Authorize Access \(p. 20\)](#) and [Step 3: Connect to a Cluster's Node \(p. 21\)](#).

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 \(p. 106\)](#).

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 \(p. 106\)](#).

Topics

- [Creating a Memcached Cache Cluster \(AWS CLI\) \(p. 85\)](#)

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. 106).

Topics

- [Creating a Memcached Cache Cluster \(ElastiCache API\)](#) (p. 86)

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=AWS4-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.

Topics

- [Viewing a Cluster's Details \(Console\) \(p. 87\)](#)
- [Viewing a Cluster's Details \(AWS CLI\) \(p. 89\)](#)
- [Viewing a Cluster's Details \(ElastiCache API\) \(p. 90\)](#)

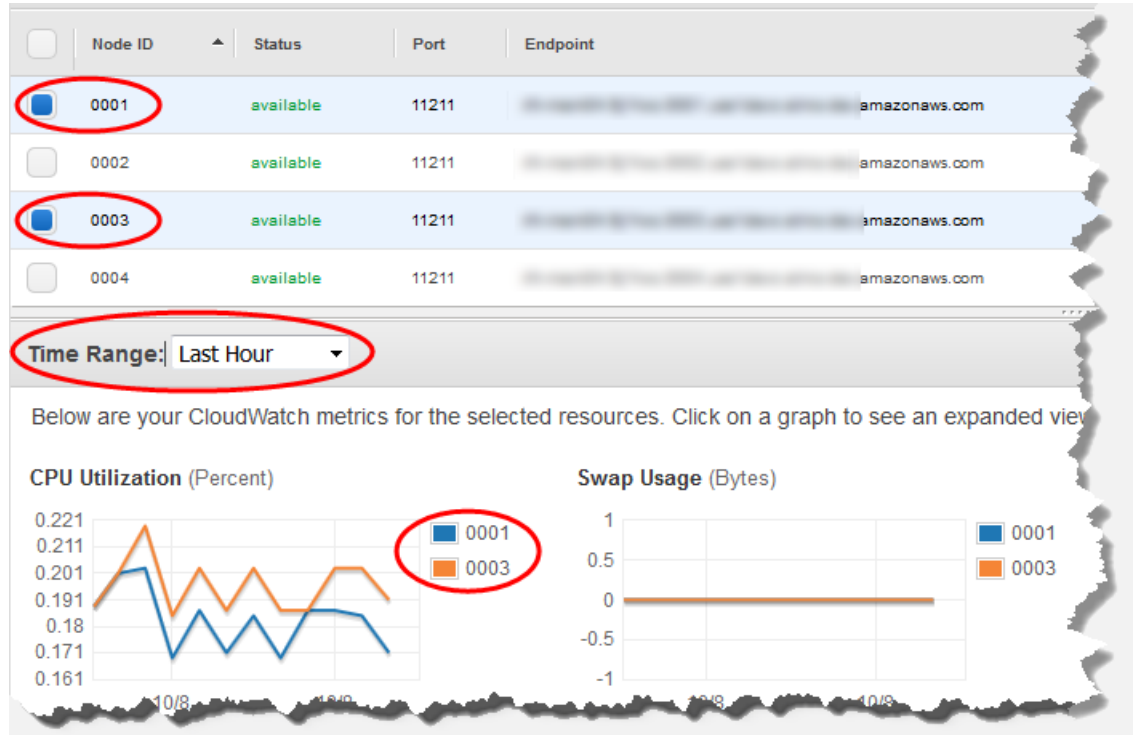
Viewing a Cluster's Details (Console)

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**. This will display a list of all your clusters that are running any version of Memcached.
4. To see details of a cluster, choose the box to the left of the cluster's name.
5. To view node information:
 - a. Choose the cluster's name.
 - b. Choose the **Nodes** tab.
 - c. To view metrics on one or more nodes, choose the box to the left of the Node ID, and then choose the time range for the metrics from the **Time range** list. Selecting multiple nodes will generate overlay graphs.



Metrics over the last hour for two Memcached nodes

Viewing a Cluster's Details (AWS CLI)

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

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

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

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

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

Example

For Linux, macOS, or Unix:

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

For Windows:

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

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

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

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

For more information, see the AWS CLI for ElastiCache topic [describe-cache-clusters](#).

Viewing a Cluster's Details (ElastiCache API)

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

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

```
https://elasticache.us-west-2.amazonaws.com/
?Action=DescribeCacheClusters
&CacheClusterId=my-cluster
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
```

```
&X-Amz-Credential=<credential>
```

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

```
https://elasticache.us-west-2.amazonaws.com/  
?Action=DescribeCacheClusters  
&MaxRecords=25  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T192317Z  
&X-Amz-Credential=<credential>
```

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

Modifying an ElastiCache Cluster

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

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

When you make a change to a cluster's parameters, either by changing the cluster's parameter group or by changing a parameter value in the cluster's parameter group, the changes are applied to the cluster either immediately or after the cluster is restarted. To determine when a particular parameter change is applied, see the **Changes Take Effect** column in the tables for [Memcached Specific Parameters \(p. 139\)](#) and [Redis Specific Parameters \(p. 140\)](#). For information on rebooting a cluster, see [Rebooting a Cluster \(p. 94\)](#).

Using the AWS Management Console

To modify 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 where the cluster you want to modify is located.
3. In the navigation pane, choose the engine running on the cluster you want to modify.

A list of the chosen engine's clusters appears.
4. In the list of clusters, choose the name of the cluster, not the box to the left of the cluster's name, you want to modify.
5. Choose **Modify**.

The **Modify Cluster** window appears.

6. In the **Modify Cluster** window, make the modification(s) you want.

Important

You can upgrade to newer engine versions (see [Upgrading Engine Versions \(p. 39\)](#)), but you cannot downgrade to older engine versions except by deleting the existing cluster and creating it anew.

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 will be 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 (see [Upgrading Engine Versions \(p. 39\)](#)), but you cannot downgrade to older engine versions except by deleting the existing cluster and creating it anew.

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 node type, 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 (see [Upgrading Engine Versions \(p. 39\)](#)), but you cannot downgrade to older engine versions except by deleting the existing cluster and creating it anew.

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=AWS4-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 you want to reboot.

A list of clusters running the chosen engine will appear.

4. Choose the cluster to reboot by choosing on the box to the left of the cluster's name.

The **Reboot** button will become active.

If you choose more than one cluster, the **Reboot** button becomes disabled.

5. Choose **Reboot**.

The reboot cluster confirmation screen appears.

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

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

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

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. 34).

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. 95)

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 you want to add nodes to.

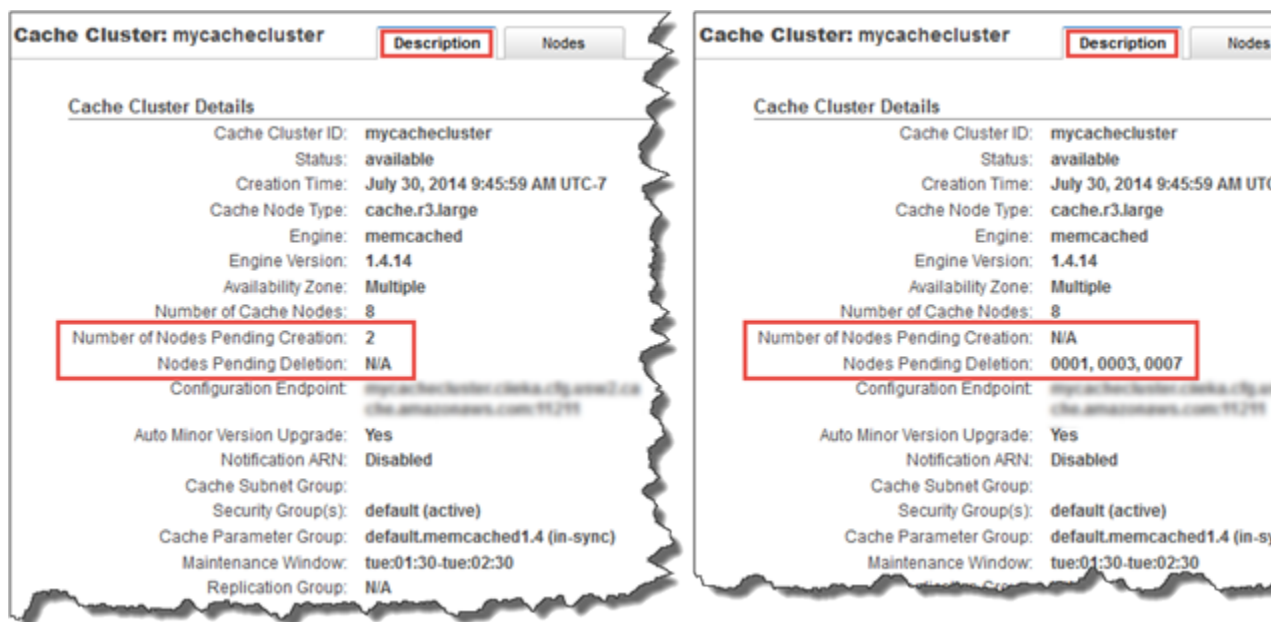
A list of clusters running the chosen engine appears.

3. From the list of clusters, choose the name of the cluster, not the box to the left of the cluster's name, you want to add a node to.
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 you want to add nodes to.
- `--num-cache-nodes` The `--num-cache-nodes` parameter specifies the number of nodes 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 \(p. 102\)](#).
- `--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 you want to add nodes to.
 - `NumCacheNodes` The `NumCachNodes` parameter specifies the number of nodes 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 \(p. 103\)](#)).
 - `ApplyImmediately` Specifies whether to add these nodes immediately or at the next maintenance window.
 - `Region` Specifies the AWS region of the cluster 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=AWS4-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 \(p. 34\)](#).

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

Topics

- [Using the AWS Management Console \(p. 100\)](#)
- [Using the AWS CLI \(p. 102\)](#)
- [Using the ElastiCache API \(p. 103\)](#)

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 you want to remove nodes from.
3. In the navigation pane, choose the engine running on the cluster 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 you want to remove. Using the ElastiCache console, you can only delete one node at a time, so choosing multiple nodes will disable the **Delete node** button.

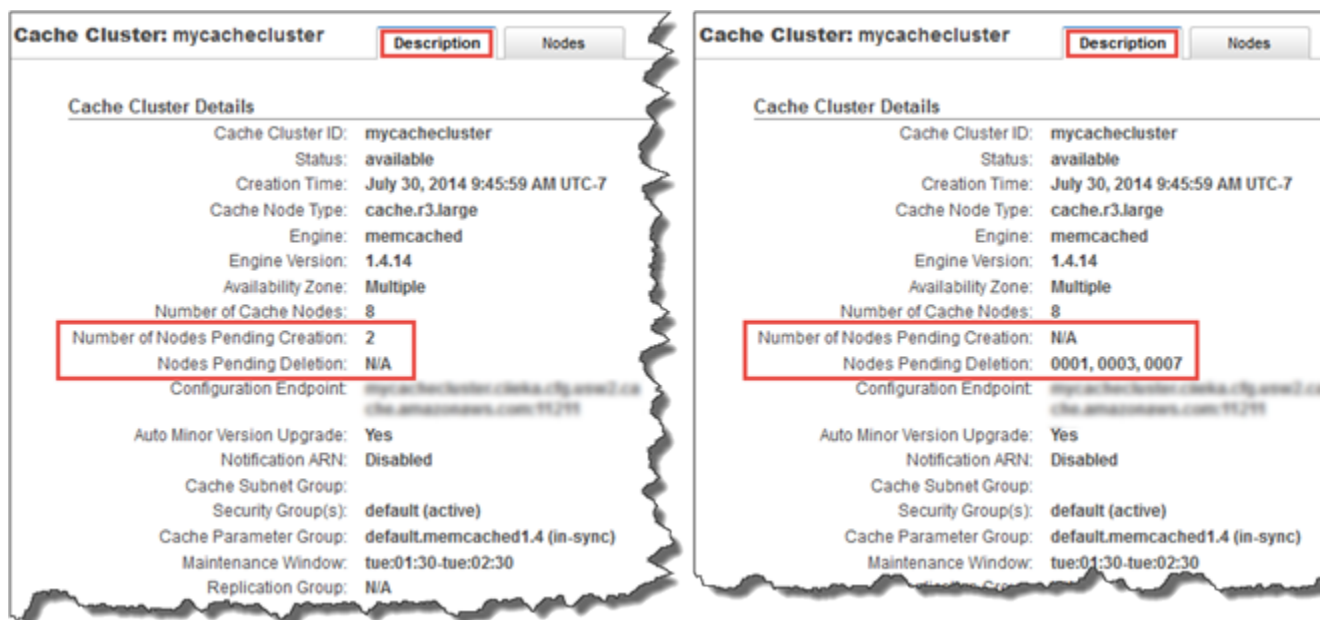
The *Delete Node* dialog appears.
6. To delete the node, complete the **Delete Node** dialog box and choose **Delete Node**. To not delete the node, choose the **Cancel**.

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,

Scenarios	Pending Operation	New Request	Results
			a new node will be created and nodes 0001, 0003, and 0007 will not be deleted.
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.



Using the AWS CLI

1. Identify the IDs of the nodes you want to remove. For more information, see [Viewing a Cluster's Details \(p. 87\)](#).
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 you want to remove nodes from.
- `--num-cache-nodes` The `--num-cache-nodes` parameter specifies the number of nodes you want in this cluster after the modification is applied.
- `--cache-node-ids-to-remove` A list of node IDs 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 region of the cluster 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 you want to remove nodes from.
- `NumCacheNodes` The `NumCacheNodes` parameter specifies the number of nodes 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 region of the cluster 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=AWS4-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](#).

Deleting a Cluster

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

Using the AWS Management Console

The following procedure deletes a single cluster from your deployment. To delete multiple clusters, repeat the procedure for each cluster 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, select the engine the cluster you want to delete is running.

A list of all clusters running the selected engine appears.

3. To select the cluster to delete, select the cluster's name from the list of clusters.

Important

You can only delete one cluster at a time from the ElastiCache console. Selecting multiple clusters disables the delete operation.

4. Select the **Actions** button and then select **Delete** from the list of actions.
5. In the **Delete Cluster** confirmation screen, choose **Delete** to delete the cluster, or select **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 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=AWS4-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 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 \(p. 108\)](#)
- [Grant Access to Your Cluster \(p. 110\)](#)

Determine the Cluster's Platform

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

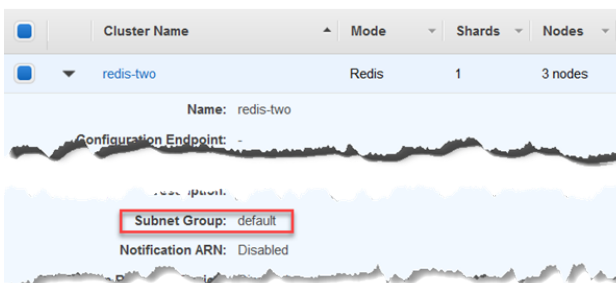
For more information, see [Detecting Your Supported Platforms and Whether You Have a Default VPC](#).

Determining Your Clusters Platform using the ElastiCache Console

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

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 box to the left of the cluster name.
4. Locate **Subnet group**:



- If the **Subnet group** has a name, as shown here, you launched your cluster in EC2-VPC and should continue at [You Launched Your Cluster into EC2-VPC \(p. 110\)](#).
- If there is a dash (-) instead of a **Subnet group** name, you launched your cluster in EC2-Classic and should continue at [You Launched Your Cluster Running in EC2-Classic \(p. 110\)](#).

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-Classic.

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": "3.2.10",
      "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 \(p. 110\)](#).
- 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 \(p. 110\)](#).

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.

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 (Step 4.e.) 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 port 6379, 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 \(p. 110\)](#).

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 \(p. 172\)](#).

Accessing ElastiCache Resources from Outside AWS

Amazon ElastiCache is an AWS service that provides cloud-based in-memory key-value store. The service is designed to be accessed exclusively from within AWS. However, if the ElastiCache cluster is hosted inside a VPC, you can use a Network Address Translation (NAT) instance to provide outside access.

Requirements

The following requirements must be met for you to access your ElastiCache resources from outside AWS:

- The cluster must reside within a VPC and be accessed through a Network Address Translation (NAT) instance. There are no exceptions to this requirement.
- The NAT instance must be launched in the same VPC as the cluster.
- The NAT instance must be launched in a public subnet separate from the cluster.
- An Elastic IP Address (EIP) must be associated with the NAT instance. The port forwarding feature of iptables is used to forward a port on the NAT instance to the cache node port within the VPC.

Considerations

The following considerations should be kept in mind when accessing your ElastiCache resources from outside ElastiCache.

- Clients connect to the EIP and cache port of the NAT instance. Port forwarding on the NAT instance forwards traffic to the appropriate cache cluster node.
- If a cluster node is added or replaced, the iptables rules need to be updated to reflect this change.

Limitations

This approach should be used for testing and development purposes only. It is not recommended for production use due to the following limitations:

- The NAT instance is acting as a proxy between clients and multiple clusters. The addition of a proxy impacts the performance of the cache cluster. The impact increases with number of cache clusters you are accessing through the NAT instance.
- The traffic from clients to the NAT instance is unencrypted. Therefore, you should avoid sending sensitive data via the NAT instance.
- The NAT instance adds the overhead of maintaining another instance.
- The NAT instance serves as a single point of failure. For information about how to set up high availability NAT on VPC, see [High Availability for Amazon VPC NAT Instances: An Example](#).

How to Access ElastiCache Resources from Outside AWS

The following procedure demonstrates how to connect to your ElastiCache resources using a NAT instance.

These steps assume the following:

- You are accessing a Memcached cluster with:
 - IP address – *10.0.1.230*
 - Default Memcached port – *11211*
 - Security group – *sg-bd56b7da*
- Your trusted client has the IP address *198.51.100.27*.
- Your NAT instance has the Elastic IP Address *203.0.113.73*.

- Your NAT instance has security group `sg-ce56b7a9`.

To connect to your ElastiCache resources using a NAT instance

1. Create a NAT instance in the same VPC as your cache cluster but in a public subnet.

By default, the VPC wizard will launch a `cache.m1.small` node type. You should select a node size based on your needs.

For information about creating a NAT instance, see [NAT Instances](#) in the AWS VPC User Guide.

2. Create security group rules for the cache cluster and NAT instance.

The NAT instance security group should have the following rules:

- Two inbound rules
 - One to allow TCP connections from trusted clients to each cache port forwarded from the NAT instance (11211 - 11213).
 - A second to allow SSH access to trusted clients.

NAT Instance Security Group - Inbound Rules

Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	11211-11213	198.51.100.27/32
SSH	TCP	22	203.0.113.73/32

- An outbound rule to allow TCP connections to cache port (11211).

NAT Instance Security Group - Outbound Rule

Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	11211	sg-ce56b7a9 (NAT instance Security Group)

- An inbound rule for the cluster's security group that allows TCP connections from the NAT instance to the cache port (11211).

Cluster Instance Security Group - Inbound Rule

Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	11211	sg-bd56b7da (Cluster Security Group)

3. Validate the rules.
 - Confirm that the trusted client is able to SSH to the NAT instance.
 - Confirm that the trusted client is able to connect to the cluster from the NAT instance.
4. Add an iptables rule to the NAT instance.

An iptables rule must be added to the NAT table for each node in the cluster to forward the cache port from the NAT instance to the cluster node. An example might look like the following:

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

The port number must be unique for each node in the cluster. For example, if working with a three node Memcached cluster using ports 11211 - 11213, the rules would look like the following:

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

5. Confirm that the trusted client is able to connect to the cluster.

The trusted client should connect to the EIP associated with the NAT instance and the cluster port corresponding to the appropriate cluster node. For example, the connection string for PHP might look like the following:

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

A telnet client can also be used to verify the connection. For example:

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

6. Save the iptables configuration.

Save the rules after you test and verify them. If you are using a Redhat-based Linux distribution (like Amazon Linux), run the following command:

```
service iptables save
```

Related topics

The following topics may be of additional interest.

- [Access Patterns for Accessing an ElastiCache Cluster in an Amazon VPC \(p. 154\)](#)
- [Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center \(p. 157\)](#)
- [NAT Instances](#)
- [Configuring ElastiCache Clients](#)
- [High Availability for Amazon VPC NAT Instances: An Example](#)

Finding Connection Endpoints

Your application connects to your cluster using endpoints. An endpoint is a node or cluster's unique address.

Which endpoints to use

- **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.

Topics

- [Finding a Cluster's Endpoints \(Console\) \(p. 116\)](#)
- [Finding Endpoints \(AWS CLI\) \(p. 118\)](#)
- [Finding Endpoints \(ElastiCache API\) \(p. 120\)](#)

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 Memcached Cluster \(p. 56\)](#).

The following procedure demonstrates how to find and copy a cluster's configuration endpoint or any of the node endpoints using the ElastiCache console.

To find and copy the endpoints for a Memcached cluster (console)

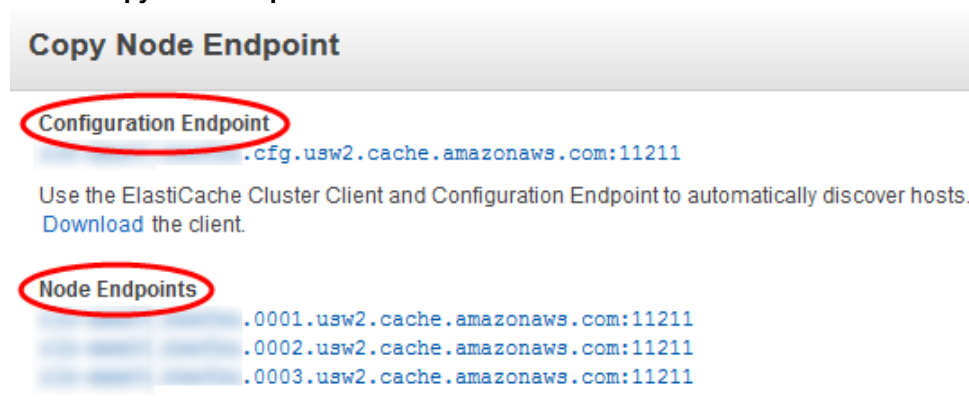
1. Sign in to the AWS Management Console and open the ElastiCache console at <https://console.aws.amazon.com/elasticache/>.
2. From the navigation pane, choose **Memcached**.

The cache clusters screen will appear with a list of Memcached clusters.

3. Find the Memcached cluster you want the endpoints for.

If all you want is the configuration endpoint, you're done. The configuration endpoint is in the **Configuration Endpoint** column and looks something like this, `clusterName.xxxxxx.cfg.usw2.cache.amazonaws.com:port`.

If you want to also see the individual node endpoints or copy any of the endpoints to your clipboard, choose **Copy Node Endpoint**.



Endpoints for a Memcached cluster

4. To copy an endpoint to your clipboard:
 - a. On the **Copy Node Endpoint** screen, highlight the endpoint you want to copy.
 - b. Right-click the highlighted endpoint, and then choose **Copy** from the context menu.

The highlighted endpoint is now copied to your clipboard.

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

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

Important

If you choose to create a CNAME for your Memcached configuration endpoint, in order for your automatic discovery client to recognize the CNAME as a configuration endpoint, you must include `.cfg.` in the CNAME.

Finding Endpoints (AWS CLI)

You can use the AWS CLI for Amazon ElastiCache to discover the endpoints for nodes and clusters.

Topics

- [Finding Endpoints for Nodes and Clusters \(AWS CLI\) \(p. 118\)](#)

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 PHP 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 \(ElastiCache API\) \(p. 120\)](#)

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 PHP 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.

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 you want to perform.

Scaling Memcached Clusters

Action	Topic/Link
Scaling out	Adding Nodes to a Cluster (p. 95)
Scaling in	Removing Nodes from a Cluster (p. 100)
Changing node types	Scaling Memcached Vertically (p. 121)

Memcached clusters are composed of 1 to 20 nodes. Scaling a Memcached cluster out and in is as easy as adding or removing nodes from the cluster.

If you need more than 20 nodes in a Memcached cluster, or more than 100 nodes total in an AWS Region, please 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 \(p. 121\)](#)
- [Scaling Memcached Vertically \(p. 121\)](#)

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

If you need more than 20 nodes in a Memcached cluster, or more than 100 nodes total in an AWS Region, please 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 \(p. 95\)](#)
- [Removing Nodes from a Cluster \(p. 100\)](#)

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

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 Memcached Cluster \(p. 56\)](#). 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 \(p. 81\)](#).

Topics

- [Scaling Memcached Vertically \(Console\) \(p. 122\)](#)
- [Scaling Memcached Vertically \(AWS CLI\) \(p. 122\)](#)
- [Scaling Memcached Vertically \(ElastiCache API\) \(p. 122\)](#)

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. 83).
2. In your application, update the endpoints to the new cluster's endpoints. For more information, see [Finding a Cluster's Endpoints \(Console\)](#) (p. 116).
3. Delete the old cluster. For more information, see [Using the AWS Management Console](#) (p. 106).

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. 85).
2. In your application, update the endpoints to the new cluster's endpoints. For more information, see [Finding Endpoints \(AWS CLI\)](#) (p. 118).
3. Delete the old cache cluster. For more information, see [Using the AWS CLI](#) (p. 106).

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. 86).
2. In your application, update the endpoints to the new cache cluster's endpoints. For more information, see [Finding Endpoints \(ElastiCache API\)](#) (p. 120).
3. Delete the old cache cluster. For more information, see [Using the ElastiCache API](#) (p. 107).

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 \(p. 139\)](#).

As you would expect, some parameter values, such as `max_cache_memory`, are determined by the engine and node type. For a table of these parameter values by node type, see [Memcached Node-Type Specific Parameters \(p. 147\)](#).

Topics

- [Parameter Management \(p. 124\)](#)
- [Cache Parameter Group Tiers \(p. 125\)](#)
- [Creating a Parameter Group \(p. 125\)](#)
- [Listing Parameter Groups by Name \(p. 129\)](#)
- [Listing a Parameter Group's Values \(p. 133\)](#)
- [Modifying a Parameter Group \(p. 134\)](#)
- [Deleting a Parameter Group \(p. 137\)](#)
- [Memcached Specific Parameters \(p. 139\)](#)

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 will 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.

Constraints

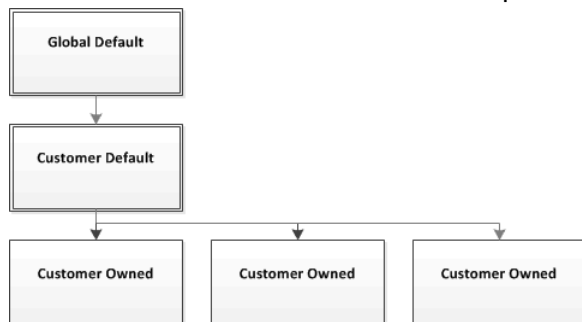
- You cannot modify or delete the default parameter groups. If you need custom parameter values, you must create a custom parameter group.
- The parameter group family and the cluster you're assigning it to must be compatible. For example, if your cluster is running Memcached version 1.4.8, you can only use parameter groups, default or custom, from the Memcached 1.4 family.

The parameter group family and the cluster you're assigning it to must be compatible. For example, if your cluster is running Redis version 3.2.10, you can only use parameter groups, default or custom, from the Redis3.2 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 make a change to a cluster's parameters, either by changing the cluster's parameter group or by changing a parameter value in the cluster's parameter group, the changes are applied to the cluster either immediately or after the cluster is restarted. To determine when a particular parameter change is applied, see the **Changes Take Effect** column in the tables for [Memcached Specific Parameters](#) (p. 139). For information on rebooting a cluster, see [Rebooting a Cluster](#) (p. 94).

Cache Parameter Group Tiers

Amazon ElastiCache has three tiers of cache parameter groups as illustrated here.



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 \(p. 139\)](#).

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 will appear.

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

- Must begin with an ASCII letter.
 - Can only contain ASCII letters, digits, and hyphens.
 - Must be between 1 and 255 characters long.
 - Cannot contain two consecutive hyphens.
 - Cannot 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. 134).

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

- Must begin with an ASCII letter.
- Can only contain ASCII letters, digits, and hyphens.
- Must be between 1 and 255 characters long.
- Cannot contain two consecutive hyphens.

- Cannot 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. 134).

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

- Must begin with an ASCII letter.
- Can only contain ASCII letters, digits, and hyphens.
- Must be between 1 and 255 characters long.
- Cannot contain two consecutive hyphens.
- Cannot 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. 134).

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"  
    }  
  ]  
}
```

The following sample code lists up to 10 parameter groups.

```
aws elasticache describe-cache-parameter-groups --max-records 20
```

The JSON output of this command will look something like this, listing the name, family, and description 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"
    }
  ]
}
```

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, and description 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>
    </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, and description 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>
    </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. 139).

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. Choose **Save Changes**.
7. To find the name of the parameter you changed, see [Memcached Specific Parameters \(p. 139\)](#). If changes to the parameter take place *After restart*, reboot every cluster that uses this parameter group. For more information, see [Rebooting a Cluster \(p. 94\)](#).

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 \(p. 139\)](#)

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](#).

If changes to the parameter take place *After restart*, reboot every cluster that uses this parameter group. For more information, see [Rebooting a Cluster \(p. 94\)](#).

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 \(p. 139\)](#)

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](#).

After updating and saving the parameter, if the change to the parameter take place *After restart*, reboot every cluster that uses this parameter group. For more information, see [Rebooting a Cluster \(p. 94\)](#).

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 cannot 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 \(p. 125\)](#).

Topics

- [Memcached 1.5.10 Parameter Changes \(p. 139\)](#)
- [Memcached 1.4.34 Added Parameters \(p. 140\)](#)
- [Memcached 1.4.33 Added Parameters \(p. 140\)](#)
- [Memcached 1.4.24 Added Parameters \(p. 142\)](#)
- [Memcached 1.4.14 Added Parameters \(p. 143\)](#)
- [Memcached 1.4.5 Supported Parameters \(p. 144\)](#)
- [Memcached Connection Overhead \(p. 146\)](#)
- [Memcached Node-Type Specific Parameters \(p. 147\)](#)

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
<code>no_modern</code>	Default: 0 Type: boolean Modifiable: Yes Allowed_Values: 0,1 Changes Take Effect: At launch	An alias for disabling <code>slab_reassign</code> , <code>slab_automove</code> , <code>lru_crawler</code> , <code>lru_maintainer</code> , <code>maxconns_fast</code> commands. No <code>modern</code> also sets the <code>hash_algorithm</code> to <code>jenkins</code> and allows inlining of ASCII <code>VALUE</code> . Applicable to memcached 1.5 and higher. To revert to <code>modern</code> , which is now the default, you must re-launch.
<code>inline_ascii_resp</code>	Default: 0 Type: boolean Modifiable: Yes Allowed_Values: 0,1 Changes Take Effect: At launch	Stores numbers from <code>VALUE</code> response, inside an item, using up to 24 bytes. Small slowdown for ASCII <code>get</code> , faster sets.

For Memcached 1.5.10, the following parameters are removed.

Name	Details	Description
<code>expirezero_does_not_evict</code>	Default: 0	No longer supported in this version.

Name	Details	Description
	Type: boolean Modifiable: Yes Allowed_Values: 0,1 Changes Take Effect: At launch	
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	Logs fetches, evictions or mutations. When, for example, user turns watch on, they can see logs when get, set, delete, or update occur.

Amazon ElastiCache ElastiCache
for Memcached User Guide
Memcached Specific Parameters

Name	Details	Description
	worker_logbuf_size limits.	
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.
cache_memlimit	Type: integer Modifiable: Yes Changes Take Effect: Immediately	If memory isn't being preallocated, allows dynamically increasing the memory limit of a running system. <code>cache memlimit N</code> where N is a value in megabytes. Value can go up or down. Range: 8 (MB) to the node type's maxmemory.
track_sizes	Default: disabled Type: boolean Modifiable: Yes Changes Take Effect: At Launch	Shows the sizes each slab group has consumed. Enabling <code>track_sizes</code> lets you run <code>stats sizes</code> without the need to run <code>stats sizes_enable</code> .
watcher_logbuf_size	Default: 256 (KB) Type: integer Modifiable: Yes Changes Take Effect: At Launch	The <code>watch</code> command turns on stream logging for Memcached. However <code>watch</code> 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.
worker_logbuf_size	Default: 64 (KB) Type: integer Modifiable: Yes Changes Take Effect: At Launch	The <code>watch</code> command turns on stream logging for Memcached. However <code>watch</code> can drop logs if the rate of evictions, mutations or fetches are high enough to cause logging buffer get full. In such situations, users can increase the buffer size to reduce the chance of log losses.
slab_chunk_max	Default: 524288 (bytes) Type: integer Modifiable: Yes Changes Take Effect: At 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.

Name	Details	Description
<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 (<code>-F</code>) 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 Changes Take Effect: After restart Note You can temporarily enable <code>lru_crawler</code> at runtime from the command line. For more information, see the Description column.	Cleans slab classes of items that have expired. This is a low impact process that runs in the background. Currently requires initiating a crawl using a manual command. To temporarily enable, run <code>lru_crawler enable</code> at the command line. <code>lru_crawler 1,3,5</code> crawls slab classes 1, 3, and 5 looking for expired items to add to the freelist. Values: 0,1 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

Name	Details	Description
		enable permanently, you must modify the parameter value. For more information, see Modifying a Parameter Group (p. 134).
lru_maintainer	Default: 0 (disabled) Type: boolean Modifiable: Yes Changes Take Effect: At launch	A background thread that shuffles items between the LRUs as capacities are reached. Values: 0, 1.
expirezero_does_not_evict	Default: 0 (disabled) Type: boolean Modifiable: Yes Changes Take Effect: At launch	When used with lru_maintainer, makes items with an expiration time of 0 unevictable. Warning This can crowd out memory available for other evictable items. Can be set to disregard lru_maintainer.

Memcached 1.4.14 Added Parameters

For Memcached 1.4.14, the following additional parameters are supported.

Parameter group family: memcached1.4

Parameters added in Memcached 1.4.14

Name	Description
config_max	Default: 16 Maximum number of ElastiCache configuration entries. Type: integer Modifiable: No
config_size_max	Default: 65536 Maximum size of the configuration entries, in bytes. Type: integer Modifiable: No
hashpower_init	Default: 16 Initial size of the ElastiCache hash table, expressed as a power of two. The default is 16 (2^16), or 65536 keys.

Name	Description
	Type: integer Modifiable: No
maxconns_fast	Default: 0 Type: integer Description: 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. Modifiable: Yes Changes Take Effect: After restart
slab_automove	Default: 1 Type: integer Description: Controls 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 integer is not recommended except for testing purposes.) Modifiable: Yes Changes Take Effect: After restart
slab_reassign	Default: 0 Type: Boolean Description: Enable or disable slab reassignment. If this parameter is set to 1, you can use the "slabs reassign" command to manually reassign memory. Modifiable: Yes 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	Default: 1024 Type: integer Modifiable: No	The backlog queue limit.
binding_protocol	Default: auto Type: string	The binding protocol. Permissible values are: ascii and auto.

Name	Details	Description
	Modifiable: Yes Changes Take Effect: After restart	For guidance on modifying the value of <code>binding_protocol</code> , see Modifying a Parameter Group (p. 134).
<code>cas_disabled</code>	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.
<code>chunk_size</code>	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.
<code>chunk_size_growth_factor</code>	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.
<code>error_on_memory_exhaustion</code>	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.
<code>large_memory_pages</code>	Default: 0 (false) Type: Boolean Modifiable: No	If 1 (true), ElastiCache will try to use large memory pages.
<code>lock_down_paged_memory</code>	Default: 0 (false) Type: Boolean Modifiable: No	If 1 (true), ElastiCache will lock down all paged memory.
<code>max_item_size</code>	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.

Name	Details	Description
<code>max_simultaneous_connections</code>	Default: 65000 Type: integer Modifiable: No	The maximum number of simultaneous connections.
<code>maximize_core_file_limit</code>	Default: 0 (false) Type: Boolean Modifiable: Changes Take Effect: No	If 1 (true), ElastiCache will maximize the core file limit.
<code>memcached_connections_overhead</code>	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. 146) .
<code>requests_per_event</code>	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-Specific Parameters

Node Type	max_cache_memory (MiB)	num-threads
cache.t1.micro	213	1
cache.t2.micro	555	1
cache.t2.small	1588	1
cache.t2.medium	3301	2
cache.m1.small	1300	1
cache.m1.medium	3350	1
cache.m1.large	7100	2
cache.m1.xlarge	14600	4
cache.m2.xlarge	16700	2
cache.m2.2xlarge	33800	4
cache.m2.4xlarge	68000	8
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	14618	4
cache.m4.2xlarge	30412	8
cache.m4.4xlarge	62234	16
cache.m4.10xlarge	158355	40
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	242600	32
cache.r4.large	12590	2

Node Type	max_cache_memory (MiB)	num-threads
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

Note

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

Securing Network Access

Amazon ElastiCache uses the following techniques to secure your cache data and protect it from unauthorized access:

Topics

- [Amazon VPCs and ElastiCache Security \(p. 149\)](#)
- [Subnets and Subnet Groups \(p. 164\)](#)
- [Security Groups: EC2-Classical \(p. 172\)](#)

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-Classical.

Important

We have deprecated the use of Amazon EC2-Classical for launching ElastiCache clusters. All current generation nodes are launched in Amazon Virtual Private Cloud only.

Topics

- [Understanding ElastiCache and Amazon VPCs \(p. 150\)](#)
- [Access Patterns for Accessing an ElastiCache Cluster in an Amazon VPC \(p. 154\)](#)
- [Creating a Virtual Private Cloud \(VPC\) \(p. 160\)](#)
- [Creating a Cache Subnet Group \(p. 162\)](#)
- [Creating a Cache Cluster in an Amazon VPC \(p. 163\)](#)
- [Connecting to a Cluster Running in an Amazon VPC \(p. 164\)](#)

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 \(p. 150\)](#)
- [Access Patterns for Accessing an ElastiCache Cluster in an Amazon VPC \(p. 154\)](#)
- [Creating a Virtual Private Cloud \(VPC\) \(p. 160\)](#)
- [Creating a Cache Subnet Group \(p. 162\)](#)
- [Creating a Cache Cluster in an Amazon VPC \(p. 163\)](#)
- [Connecting to a Cluster Running in an Amazon VPC \(p. 164\)](#)

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](#).

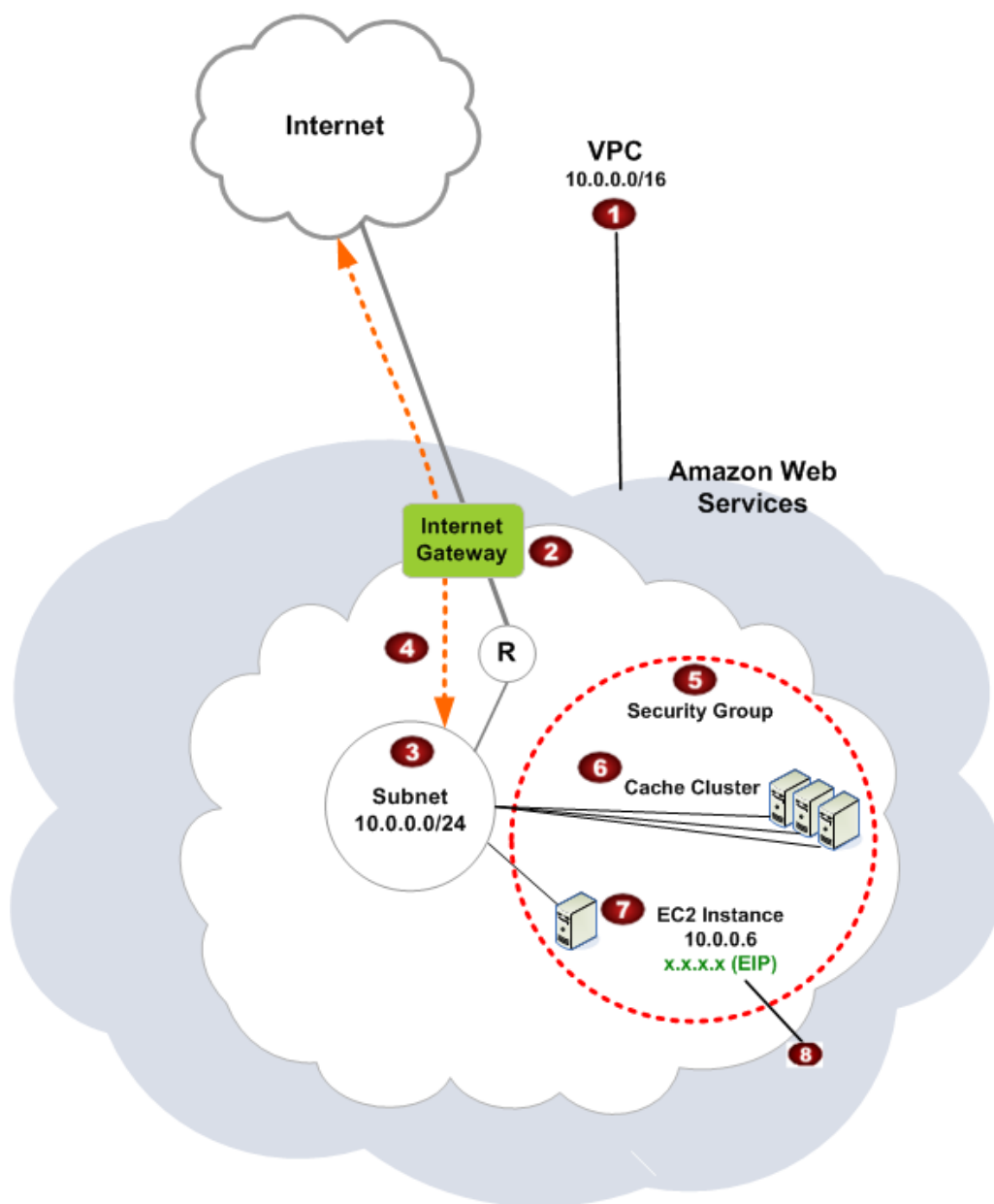
With Amazon Virtual Private Cloud, you can create a virtual network in the AWS cloud that closely resembles a traditional data center. You can configure your Amazon VPC, including selecting its IP address range, creating subnets, and configuring route tables, network gateways, and security settings.

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?

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

In order 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	Amazon VPC Getting Started Guide
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 part of the Amazon EC2 reference)
Complete descriptions of the Amazon VPC API actions, data types, and errors	Amazon EC2 API Reference (the Amazon VPC API actions are part of the Amazon EC2 reference)
Information for the network administrator who needs to configure the gateway at your end of an optional IPsec VPN connection	Amazon VPC Network Administrator Guide

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:

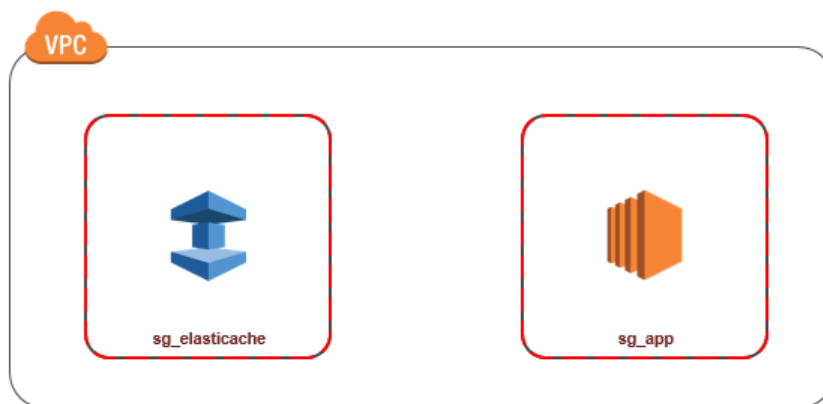
Contents

- [Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in the Same Amazon VPC \(p. 154\)](#)
- [Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs \(p. 155\)](#)
 - [Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs in the Same Region \(p. 156\)](#)
 - [Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs in Different Regions \(p. 157\)](#)
- [Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center \(p. 157\)](#)
 - [Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center Using VPN Connectivity \(p. 158\)](#)
 - [Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center Using Direct Connect \(p. 159\)](#)

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 DB instances 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 DB instances.

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. Choose **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.

Edit inbound rules

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

Add Rule

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

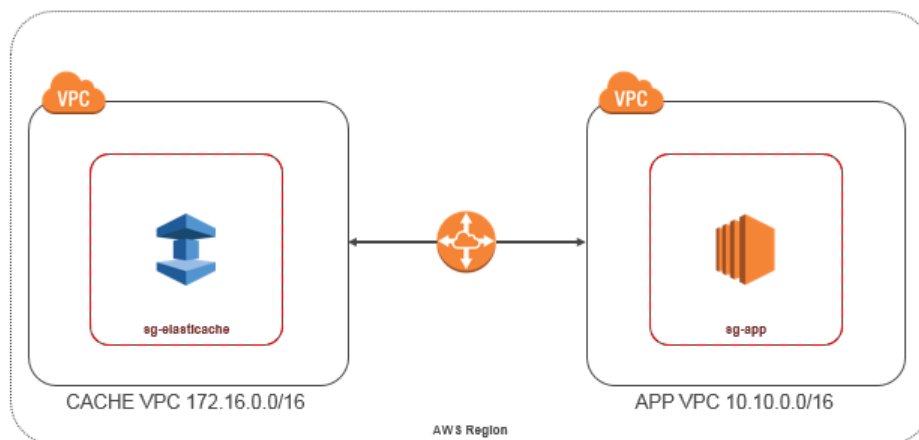
When your Cluster is in a different VPC from the EC2 instance you are using to access it, there are several ways to access the DB instance. If the Cluster and EC2 instance are in different VPCs but in the same region, you can use VPC peering. If the Cluster and the EC2 instance are in different regions, 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 \(p. 156\)](#)
- [Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs in Different Regions \(p. 157\)](#)

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](#).

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 a 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

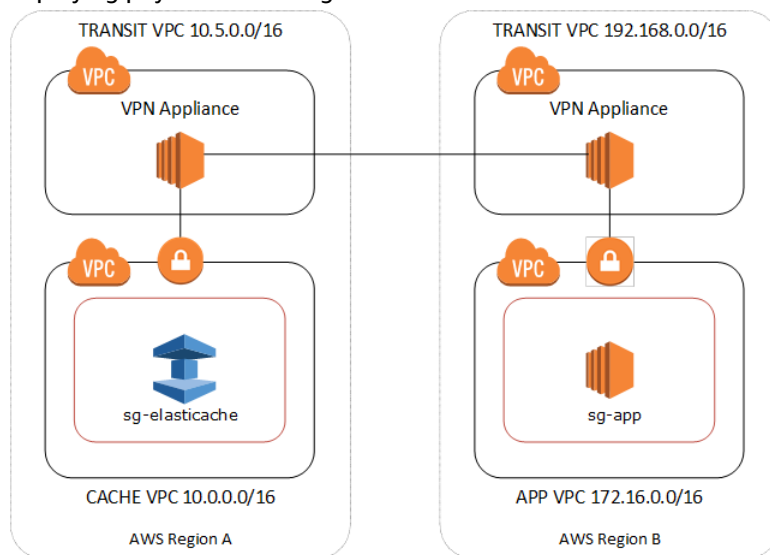
VPC Routing Table

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.

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

One 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 Region

1. Deploy a Transit VPC Solution. For more information, see, [How do I build a global transit network on AWS?](#).
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

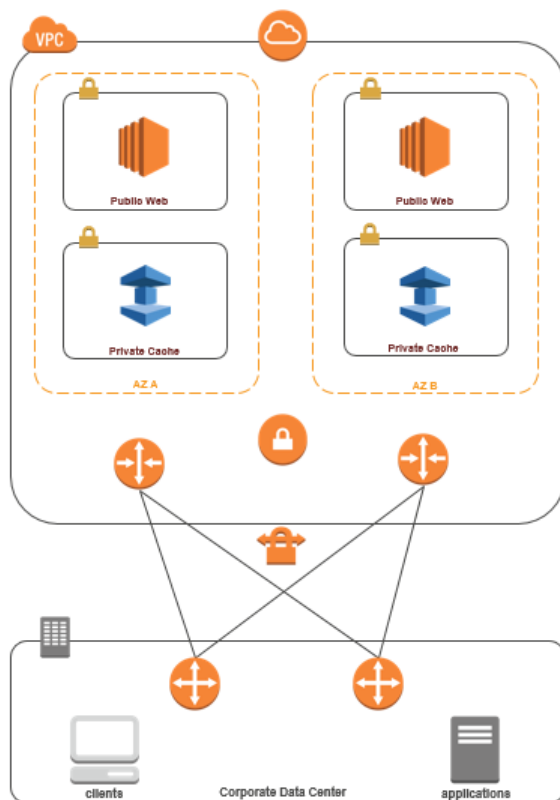
Another possible scenario is a Hybrid architecture where clients or applications in the customer’s data center may need to access an ElastiCache Cluster in the VPC. This scenario is also supported providing there is connectivity between the customers’ VPC and the data center either through VPN or Direct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

Topics

- [Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center Using VPN Connectivity \(p. 158\)](#)
- [Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center Using Direct Connect \(p. 159\)](#)

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

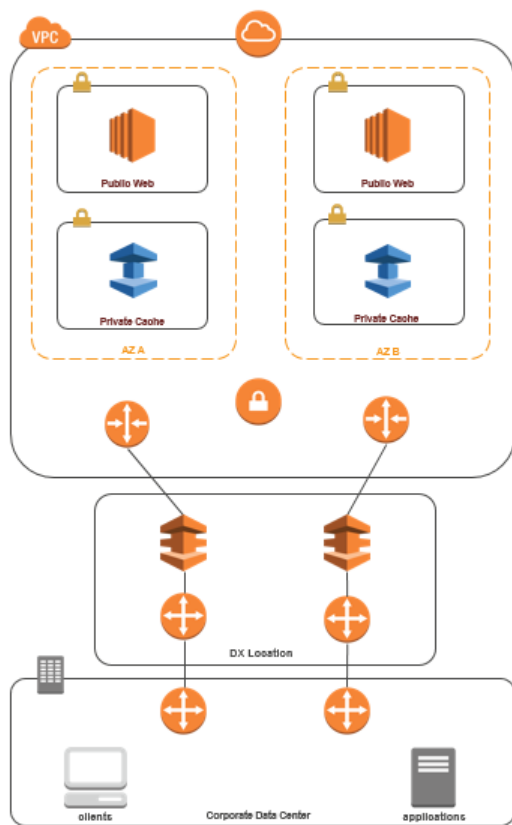
To access a cluster in a VPC from on-prem application over VPN connection

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.

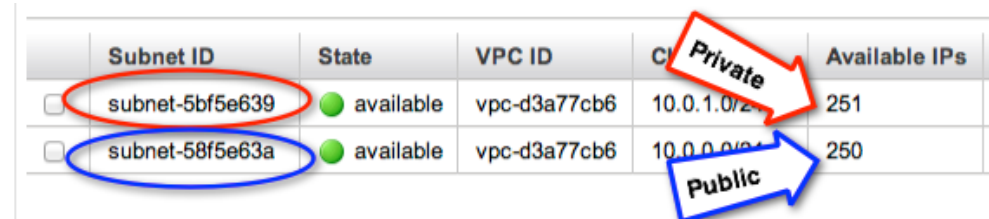
Creating a Virtual Private Cloud (VPC)

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. Create a new Amazon VPC by using the Amazon Virtual Private Cloud wizard:
 - a. In the navigation list, choose **VPC Dashboard**.
 - b. Choose **Start VPC Wizard**.
 - c. In the Amazon VPC wizard, choose **VPC with Public and Private Subnets**, and then choose **Next**.
 - d. On the **VPC with Public and Private Subnets** page, keep the default options, and then choose **Create VPC**.
 - e. In the confirmation message that appears, choose **Close**.
3. Confirm that there are two subnets in your Amazon VPC, a public subnet and a private subnet. These subnets are created automatically.
 - a. In the navigation list, choose **Subnets**.
 - b. In the list of subnets, find the two subnets that are in your Amazon VPC:



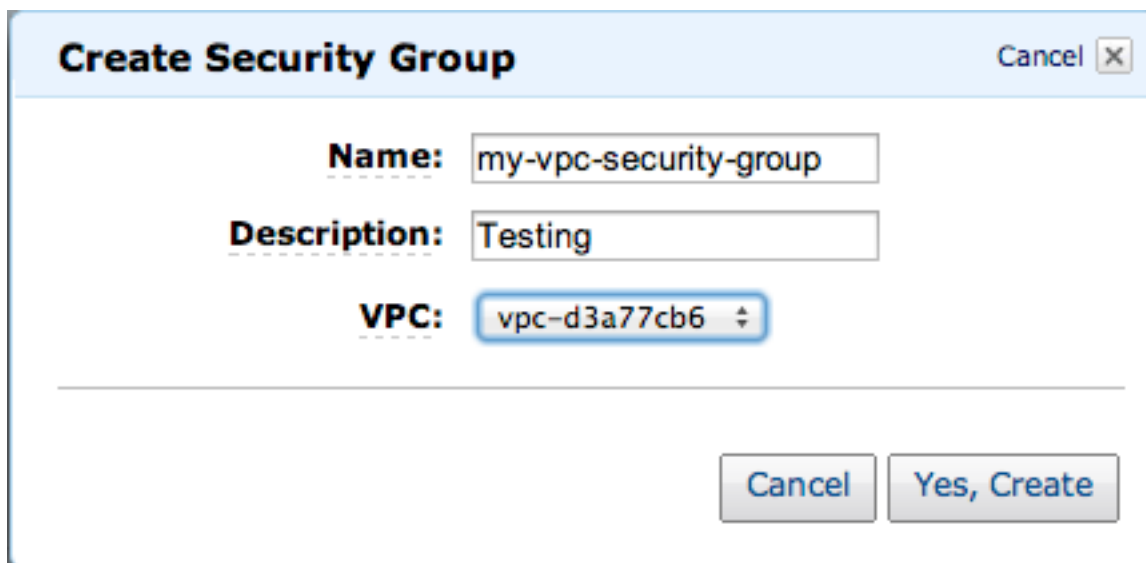
	Subnet ID	State	VPC ID	CIDR Block	Available IPs
<input type="checkbox"/>	subnet-5bf5e639	available	vpc-d3a77cb6	10.0.1.0/24	251
<input type="checkbox"/>	subnet-58f5e63a	available	vpc-d3a77cb6	10.0.0.0/24	250

The public subnet will have one fewer available IP address, because the wizard creates an Amazon EC2 NAT instance and an Elastic IP address (for which Amazon EC2 rates apply) for outbound communication to the Internet from your private subnet.

Tip

Make a note of your two subnet identifiers, and which is public and private. You will need this information later when you launch your cache clusters and add an Amazon EC2 instance to your Amazon VPC.

4. 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**.
5. 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. 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 Cache Subnet Group

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

For guidance on how to create a subnet group using the ElastiCache Management Console, the AWS CLI, or the ElastiCache API, see [Creating a Subnet Group \(p. 165\)](#).

After you create a cache subnet group, you can launch a cache cluster to run in your Amazon VPC. Continue to the next topic [Creating a Cache Cluster in an Amazon VPC \(p. 163\)](#).

Creating a Cache Cluster in an Amazon VPC

In this example, you create a cache cluster in your Amazon VPC.

Creating a Cache Cluster in an Amazon VPC (Console)

To launch a Memcached cache cluster, see [Creating a Memcached Cluster \(Console\)](#) (p. 83). In step 6.c select a VPC subnet group.

You have now launched a cache cluster inside an Amazon VPC. For an example of one way to connect to your new cache cluster running in the Amazon VPC, continue to [Connecting to a Cluster Running in an Amazon VPC](#) (p. 164).

Connecting to a 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.

Note

For information about using Amazon EC2, see the [Amazon EC2 Getting Started Guide](#) in the [Amazon EC2 documentation](#).

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.

For information on connecting to your cluster, see [Step 3: Connect to a Cluster's Node \(p. 21\)](#) in the ElastiCache User Guide.

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.

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 2: Authorize Access \(p. 20\)](#).

Topics

- [Creating a Subnet Group \(p. 165\)](#)
- [Assigning a Subnet Group to a Cluster \(p. 168\)](#)
- [Modifying a Subnet Group \(p. 169\)](#)
- [Deleting a Subnet Group \(p. 171\)](#)

Creating a Subnet Group

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.

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 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 an existing 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	CIDR Block
<input type="text" value="sa-east-1a"/>	<input type="text" value="subnet-5bf5e639"/>	<input type="text" value="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=AWS4-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 one of the following topics.

- **Memcached cluster** – To launch a Memcached cluster, see [Creating a Memcached Cluster \(Console\) \(p. 83\)](#). In step 5.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=AWS4-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.

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=AWS4-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

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. 149)

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. 149).

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. 173)
- [Listing Available Security Groups](#) (p. 175)
- [Viewing a Security Group](#) (p. 177)
- [Authorizing Network Access to an Amazon EC2 Security Group](#) (p. 179)

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

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 ElastiCache console at <https://console.aws.amazon.com/elasticache/>.
2. In the navigation pane, choose **Security Groups**.
3. Choose **Create Security Group**.
4. In **Create Security Group**, type the name of the new security group in **Security Group**.
5. In **Description**, type a description for the new 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=AWS4-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. 149).

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 ElastiCache console at <https://console.aws.amazon.com/elasticache/>.
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=AWS4-HMAC-SHA256
&X-Amz-Date=20150202T220302Z
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20150202T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

Viewing 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. 149).

You can view detailed information about your security group.

The following procedures show you how to view the properties of a security group using the ElastiCache console, AWS CLI, and ElastiCache API.

Viewing a Security 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 pane, choose **Security Groups**.

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

3. Choose a cache security group from the **Security Groups** list.

The list of authorizations defined for the security group appears in the detail section at the bottom of the window.

Viewing a Security Group (AWS CLI)

At the command prompt, use the AWS CLI `describe-cache-security-groups` command with the name of the security group you want to view.

- `--cache-security-group-name` – the name of the security group to return details for.

```
aws elasticache describe-cache-security-groups --cache-security-group-name mysecuritygroup
```

JSON output from this command will look something like this.

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

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

Viewing a Security Group (ElastiCache API)

Using the ElastiCache API, call `DescribeCacheSecurityGroups` with the name of the security group you want to view.

- `CacheSecurityGroupName` – the name of the cache security group to return details for.

Example

Line breaks are added for ease of reading.

```
https://elasticache.amazonaws.com/  
?Action=DescribeCacheSecurityGroups  
&CacheSecurityGroupName=mysecuritygroup  
&Version=2015-02-02  
&SignatureVersion=4  
&SignatureMethod=HmacSHA256  
&Timestamp=20150202T220302Z  
&X-Amz-Algorithm=AWS4-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. 149).

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 ElastiCache console at <https://console.aws.amazon.com/elasticache/>.
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=AWS4-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](#).

Authentication and Access Control for 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. 181)
- [Access Control](#) (p. 182)

Authentication

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

- **AWS account root user** – When you first create an AWS account, you begin with a single 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, even the administrative ones. Instead, adhere to the [best practice of using the root user only to create your first IAM user](#). Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.
- **IAM user** – An [IAM user](#) is an identity within your AWS account that has specific custom permissions (for example, permissions to create a cluster in ElastiCache). You can use an IAM user name and password to sign in to secure AWS webpages like the [AWS Management Console](#), [AWS Discussion Forums](#), or the [AWS Support Center](#).

In addition to a user name and password, you can also generate [access keys](#) for each user. You can use these keys when you access AWS services programmatically, either through [one of the several SDKs](#) or by using the [AWS Command Line Interface \(CLI\)](#). The SDK and CLI tools use the access keys to cryptographically sign your request. If you don't use AWS tools, you must sign the request yourself. ElastiCache supports *Signature Version 4*, a protocol for authenticating inbound API requests. For more information about authenticating requests, see [Signature Version 4 Signing Process](#) in the *AWS General Reference*.

- **IAM role** – An [IAM role](#) is an IAM identity that you can create in your account that has specific permissions. It is similar to an *IAM user*, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:
 - **Federated user access** – Instead of creating an IAM user, you can use existing user identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as *federated users*. AWS assigns a role to a federated user when access is requested through an [identity provider](#). For more information about federated users, see [Federated Users and Roles](#) in the *IAM User Guide*.
 - **AWS service access** – You can use an IAM role in your account to grant an AWS service permissions to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. 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 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*.

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. 183)
- [Using Identity-Based Policies \(IAM Policies\) for Amazon ElastiCache](#) (p. 187)

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), and some services (such as AWS Lambda) also support 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, 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. 183)
- [Understanding Resource Ownership](#) (p. 183)
- [Managing Access to Resources](#) (p. 184)
- [Specifying Policy Elements: Actions, Effects, Resources, and Principals](#) (p. 185)
- [Specifying Conditions in a Policy](#) (p. 185)

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 in the following table.

Resource Type	ARN Format
Cache Cluster	<code>arn:aws:elasticache:<i>region</i>:<i>account-id</i>:cluster:<i>resource-name</i></code>

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* (the root account, an IAM user, or an IAM role) that authenticates the request that creates the resource. The following examples illustrate how this works:

- If you use the root account credentials of your AWS account to create a cache cluster, your AWS account is the owner of the resource (in ElastiCache, the resource is the cache cluster).
- If you create an IAM user in your AWS account and grant permissions to create a cache cluster to that user, the user can create a cache cluster. However, your AWS account, to which the user belongs, owns the cache cluster resource.
- If you create an IAM role in your AWS account with permissions to create a cache cluster, 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) and policies attached to a resource are referred to as *resource-based* policies. Amazon ElastiCache supports only identity-based policies (IAM policies).

Topics

- [Identity-Based Policies \(IAM Policies\)](#) (p. 184)
- [Resource-Based Policies](#) (p. 185)

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 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. The principal in the trust policy can also be an AWS service principal if you want to grant an AWS service permissions to assume the role.

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. In the current implementation, ElastiCache doesn't support identifying specific resources using the resource ARNs (also referred to as resource-level permissions) for any API actions, so you must specify a wildcard character (*).

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "DescribeCacheClusters",
    "Effect": "Allow",
    "Action": [
      "elasticache:DescribeCacheClusters"
    ],
    "Resource": "*"
  }]
```

```
}  
]  
}
```

For more information about using identity-based policies with ElastiCache, see [Using Identity-Based Policies \(IAM Policies\) for Amazon ElastiCache \(p. 187\)](#). For more information about users, groups, roles, and permissions, see [Identities \(Users, Groups, and Roles\)](#) in the *IAM User Guide*.

Resource-Based Policies

Other services, such as Amazon S3, also support resource-based permissions policies. For example, you can attach a policy to an S3 bucket to manage access permissions to that bucket. Amazon ElastiCache doesn't support resource-based policies.

Specifying Policy Elements: Actions, Effects, Resources, and Principals

For each Amazon ElastiCache resource (see [Amazon ElastiCache Resources and Operations \(p. 183\)](#)), 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 snapshot resource, the following actions are defined: `CreateCacheCluster`, `DeleteCacheCluster`, and `DescribeCacheCluster`. Note that, 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 ElastiCache resources, you always use the wildcard character (`*`) in IAM policies. For more information, see [Amazon ElastiCache Resources and Operations \(p. 183\)](#).
- **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, which you might do to make sure that a user cannot access it, 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). ElastiCache doesn't support resource-based policies.

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 \(p. 198\)](#).

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. There are no condition keys specific to Amazon ElastiCache. However, 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 review the introductory topics that explain the basic concepts and options available for you to manage access to your Amazon ElastiCache resources. For more information, see [Overview of Managing Access Permissions to Your ElastiCache Resources](#) (p. 183).

The sections in this topic cover the following:

- [Permissions Required to Use the Amazon ElastiCache Console](#) (p. 188)
- [AWS Managed \(Predefined\) Policies for Amazon ElastiCache](#) (p. 188)
- [Customer Managed Policy Examples](#) (p. 189)

The following shows an example of a permissions policy.

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Sid": "AllowClusterPermissions",
    "Effect": "Allow",
    "Action": [
      "elasticache:CreateCacheCluster",
      "elasticache:CreateReplicationGroup",
      "elasticache:DescribeCacheClusters",
      "elasticache:ModifyCacheCluster",
      "elasticache:RebootCacheCluster"],
    "Resource": "*"
  }]
}
```

The policy has two statements:

- The first statement grants permissions for the Amazon ElastiCache actions (`elasticache:CreateCacheCluster`, `elasticache:DescribeCacheClusters`, `elasticache:ModifyCacheCluster`, and `elasticache:RebootCacheCluster`) on any cache cluster owned by the account. Currently, Amazon ElastiCache doesn't support permissions for actions at the resource-level. Therefore, the policy specifies a wildcard character (*) as the `Resource` value.
- The second statement grants permissions for the IAM action (`iam:PassRole`) on IAM roles. The wildcard character (*) at the end of the `Resource` value means that the statement allows permission for the `iam:PassRole` action on any IAM role. To limit this permission to a specific role, replace the wildcard character (*) in the resource ARN with the specific role name.

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. 198).

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 \(p. 198\)](#).

To use the Amazon ElastiCache console, you need to 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 default policy and choose to use a custom managed policy, please ensure you have either permissions to call `iam:createServiceLinkedRole` (see [Example 5: Allow a User to Call IAM CreateServiceLinkedRole API \(p. 191\)](#)) or you have created the 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 \(p. 188\)](#).

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 \(p. 189\)](#)
- [Example 2: Allow a User Read-Only Access to ElastiCache Resources \(p. 190\)](#)
- [Example 3: Allow a User to Perform Common ElastiCache System Administrator Tasks \(p. 190\)](#)
- [Example 4: Allow a User to Access All ElastiCache API Actions \(p. 190\)](#)
- [Example 5: Allow a User to Call IAM CreateServiceLinkedRole API \(p. 191\)](#)

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"
        }
      }
    }
  ]
}
```

Using Service-Linked Roles for 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 ElastiCache. ElastiCache service-linked roles are predefined by ElastiCache and include all the permissions that the service requires to call AWS services on behalf of your clusters.

A service-linked role makes setting up 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 ElastiCache use cases and have predefined permissions. Only 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 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 ElastiCache \(p. 192\)](#)
- [Creating a Service-Linked Role \(IAM\) \(p. 193\)](#)
 - [Creating a Service-Linked Role \(IAM Console\) \(p. 193\)](#)
 - [Creating a Service-Linked Role \(IAM CLI\) \(p. 193\)](#)
 - [Creating a Service-Linked Role \(IAM API\) \(p. 193\)](#)
- [Editing the Description of a Service-Linked Role for ElastiCache \(p. 194\)](#)
 - [Editing a Service-Linked Role Description \(IAM Console\) \(p. 194\)](#)
 - [Editing a Service-Linked Role Description \(IAM CLI\) \(p. 194\)](#)

- [Editing a Service-Linked Role Description \(IAM API\) \(p. 194\)](#)
- [Deleting a Service-Linked Role for ElastiCache \(p. 195\)](#)
 - [Cleaning Up a Service-Linked Role \(p. 195\)](#)
 - [Deleting a Service-Linked Role \(IAM Console\) \(p. 195\)](#)
 - [Deleting a Service-Linked Role \(IAM CLI\) \(p. 196\)](#)
 - [Deleting a Service-Linked Role \(IAM API\) \(p. 196\)](#)

Service-Linked Role Permissions for ElastiCache

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 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",
      ],
      "Resource": "*"
    }
  ]
}
```

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 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. Expand the **AWS service-linked role** section, and then select the service that you want to allow to assume this new service-linked role.
4. Next to the **AWSServiceRoleForElastiCache** service-linked role, choose **Select**.
5. For **Role name**, type a suffix to add to the service-linked role default name. This suffix helps you identify the purpose of this role. Role names must be unique within your AWS account. They are not distinguished by case. For example, you cannot create roles named both **<service-linked-role-name>_SAMPLE** and **<service-linked-role-name>_sample**. Because various entities might reference the role, you cannot edit the name of the role after it has been created.
6. (Optional) For **Role description**, edit the description for the new service-linked role.
7. 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 with 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 with 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 ElastiCache

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. Type 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 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, you must first confirm that the role has no resources—clusters—associated with the role.

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 ElastiCache resources that require `AWSServiceRoleForElastiCache` (console)

- To delete a cluster, see one of the following:
 - [Using the AWS Management Console \(p. 106\)](#)
 - [Using the AWS CLI \(p. 106\)](#)
 - [Using the ElastiCache API \(p. 107\)](#)

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, type the following operation to list 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 following 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. Type the following to submit a service-linked role deletion request:

```
$ aws iam delete-service-linked-role --role-name role-name
```

3. Type 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 are setting up [Access Control \(p. 182\)](#) and writing permissions policies that you can attach to an IAM identity (identity-based policies), you can 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 wildcard character (*) as the resource value in the policy's `Resource` field.

You can use AWS-wide condition keys in your ElastiCache policies to express conditions. 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`). For all ElastiCache actions, specify the wildcard character (*) as the resource.

Amazon ElastiCache API and Required Permissions for Actions

[AddTagsToResource](#)

Action(s): `elasticache:AddTagsToResource`

Resource: *

[AuthorizeCacheSecurityGroupIngress](#)

Action(s): `elasticache:AuthorizeCacheSecurityGroupIngress`

Resource: *

[CreateCacheCluster](#)

Action(s): `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(s): `elasticache:CreateCacheParameterGroup`

Resource: *

[CreateCacheSecurityGroup](#)

Action(s): `elasticache:CreateCacheSecurityGroup`

Resource: *

[CreateCacheSubnetGroup](#)

Action(s): `elasticache:CreateCacheSubnetGroup`

Resource: *

[DeleteCacheCluster](#)

Action(s): elasticache:DeleteCacheCluster

Resource: *

[DeleteCacheParameterGroup](#)

Action(s): elasticache:DeleteCacheParameterGroup

Resource: *

[DeleteCacheSecurityGroup](#)

Action(s): elasticache:DeleteCacheSecurityGroup

Resource: *

[DeleteCacheSubnetGroup](#)

Action(s): elasticache:DeleteCacheSubnetGroup

Resource: *

[DescribeCacheClusters](#)

Action(s): elasticache:DescribeCacheClusters

Resource: *

[DescribeCacheEngineVersions](#)

Action(s): elasticache:DescribeCacheEngineVersions

Resource: *

[DescribeCacheParameterGroups](#)

Action(s): elasticache:DescribeCacheParameterGroups

Resource: *

[DescribeCacheParameters](#)

Action(s): elasticache:DescribeCacheParameters

Resource: *

[DescribeCacheSecurityGroups](#)

Action(s): elasticache:DescribeCacheSecurityGroups

Resource: *

[DescribeCacheSubnetGroups](#)

Action(s): elasticache:DescribeCacheSubnetGroups

Resource: *

[DescribeEngineDefaultParameters](#)

Action(s): elasticache:DescribeEngineDefaultParameters

Resource: *

[DescribeEvents](#)

Action(s): elasticache:DescribeEvents

Resource: *

[DescribeReservedCacheNodes](#)

Action(s): elasticache:DescribeReservedCacheNodes

Resource: *

[DescribeReservedCacheNodesOfferings](#)

Action(s): elasticache:DescribeReservedCacheNodesOfferings

Resource: *

[ListTagsForResource](#)

Action(s): elasticache:ListTagsForResource

Resource: *

[ModifyCacheCluster](#)

Action(s): elasticache:ModifyCacheCluster

Resource: *

[ModifyCacheParameterGroup](#)

Action(s): elasticache:ModifyCacheParameterGroup

Resource: *

[ModifyCacheSubnetGroup](#)

Action(s): elasticache:ModifyCacheSubnetGroup

Resource: *

[PurchaseReservedCacheNodesOffering](#)

Action(s): elasticache:PurchaseReservedCacheNodesOffering

Resource: *

[RebootCacheCluster](#)

Action(s): elasticache:RebootCacheCluster

Resource: *

[RemoveTagsForResource](#)

Action(s): elasticache:RemoveTagsForResource

Resource: *

[ResetCacheParameterGroup](#)

Action(s): elasticache:ResetCacheParameterGroup

Resource: *

[RevokeCacheSecurityGroupIngress](#)

Action(s): elasticache:RevokeCacheSecurityGroupIngress

Resource: *

Monitoring Usage, Events, and Costs

Knowing how your clusters are performing, the resources they're consuming, the events that are being generated, and the costs of your deployment are important factors in managing your enterprise caching solution. CloudWatch provides metrics for monitoring your cache performance. Cost allocation tags help you monitor and manage costs.

Topics

- [Monitoring Use with CloudWatch Metrics \(p. 202\)](#)
- [Monitoring ElastiCache Events \(p. 210\)](#)
- [Monitoring Costs with Cost Allocation Tags \(p. 219\)](#)
- [Managing Costs with Reserved Nodes \(p. 226\)](#)

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? \(p. 206\)](#) in this guide.

Topics

- [Dimensions for ElastiCache Metrics \(p. 202\)](#)
- [Host-Level Metrics \(p. 202\)](#)
- [Metrics for Memcached \(p. 203\)](#)
- [Which Metrics Should I Monitor? \(p. 206\)](#)
- [Choosing Metric Statistics and Periods \(p. 207\)](#)
- [Monitoring CloudWatch Cluster and Node Metrics \(p. 207\)](#)

Dimensions for ElastiCache Metrics

All ElastiCache metrics use the `AWS/ElastiCache` namespace and provide metrics for a single dimension, the `CacheNodeId`, which is the automatically generated identifier for each cache node in the cache cluster. You can find out what these values are for your cache nodes by using the `DescribeCacheClusters` API or **describe-cache-clusters** command line utility. For more information, see [DescribeCacheClusters](#) in the *Amazon ElastiCache API Reference* and [describe-cache-clusters](#) in the *AWS CLI Command Reference*.

Each metric is published under a single set of dimensions. When retrieving metrics, you must supply both the `CacheClusterId` and `CacheNodeId` dimensions.

Contents

- [Host-Level Metrics \(p. 202\)](#)
- [Metrics for Memcached \(p. 203\)](#)
- [Which Metrics Should I Monitor?](#)

Host-Level Metrics

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

See Also

- [Metrics for Memcached \(p. 203\)](#)

Metric	Description	Unit
<code>CPUUtilization</code>	The percentage of CPU utilization.	Percent
<code>FreeableMemory</code>	The amount of free memory available on the host.	Bytes

Metric	Description	Unit
NetworkBytesIn	The number of bytes the host has read from the network.	Bytes
NetworkBytesOut	The number of bytes the host has written to the network.	Bytes
SwapUsage	The amount of swap used on the host.	Bytes

Metrics for Memcached

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.

For complete documentation of the Memcached `stats` command, go to <https://github.com/memcached/memcached/blob/master/doc/protocol.txt>.

See Also

- [Host-Level Metrics \(p. 202\)](#)

Metric	Description	Unit
BytesReadIntoMemcached	The number of bytes that have been read from the network by the cache node.	Bytes
BytesUsedForCacheItems	The number of bytes used to store cache items.	Bytes
BytesWrittenOutFromMemcached	The number of bytes that have been written to the network by the cache node.	Bytes
CasBadval	The number of CAS (check and set) requests the cache has received where the Cas value did not match the Cas value stored.	Count
CasHits	The number of Cas requests the cache has received where the requested key was found and the Cas value matched.	Count
CasMisses	The number of Cas requests the cache has received where the key requested was not found.	Count
CmdFlush	The number of flush commands the cache has received.	Count
CmdGet	The number of get commands the cache has received.	Count
CmdSet	The number of set commands the cache has received.	Count
CurrConnections	A count of the number of connections connected to the cache at an instant in time. ElastiCache uses two to three of the connections to monitor the cluster in each case.	Count

Metric	Description	Unit
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
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

Metric	Description	Unit
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, <code>UnusedMemory</code> 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. 206)
- [SwapUsage](#) (p. 206)
- [Evictions](#) (p. 206)
- [CurrConnections](#) (p. 206)

CPUUtilization

This is a host-level metric reported as a percent. For more information, see [Host-Level Metrics](#) (p. 202).

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. 202).

This metric should not exceed 50 MB. If it does, 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.

Choosing Metric Statistics and Periods

While CloudWatch will allow you to choose any statistic and period for each metric, not all combinations will be useful. For example, the Average, Minimum, and Maximum statistics for CPUUtilization are useful, but the Sum statistic is not.

All ElastiCache samples are published for a 60 second duration for each individual cache node. For any 60 second period, a cache node metric will only contain a single sample.

For further information on how to retrieve metrics for your cache nodes, see [Monitoring CloudWatch Cluster and Node Metrics](#) (p. 207).

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

- Use the CloudWatch command **mon-get-stats** 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):

For Linux, macOS, or Unix:

```
aws cloudwatch get-metric-statistics \
  --namespace AWS/ElastiCache \
  --metric-name CPUUtilization \
  --dimensions="CacheClusterId=mycacheclass,CacheNodeId=0002" \
  --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="CacheClusterId=mycacheclass,CacheNodeId=0002" ^
  --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=mycacheclass,CacheNodeId=0002`

Example

```
http://monitoring.amazonaws.com/
?Action=GetMetricStatistics
```

Amazon ElastiCache ElastiCache
for Memcached User Guide
Monitoring CloudWatch Cluster and Node Metrics

```
&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=mycacheccluster"
&Dimensions.member.2="CacheNodeId=0002"
&Namespace=AWS/ElastiCache
&MeasureName=CPUUtilization
&Timestamp=2018-07-07T17%3A48%3A21.746Z
&AWSAccessKeyId=<AWS Access Key ID>
&Signature=<Signature>
```

Monitoring ElastiCache Events

When significant events happen on a cluster, such as a failure to add a node, success in adding a node, the modification of a security group and others, ElastiCache sends notification to a specific Amazon SNS topic. 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 \(p. 210\)](#)
- [Viewing ElastiCache Events \(p. 213\)](#)
- [Event Notifications and Amazon SNS \(p. 215\)](#)

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-2:565419523791:ElastiCacheNotifications
```

For Windows:

```
aws elasticache modify-cache-cluster ^  
--cache-cluster-id my-cluster ^  
--notification-topic-arn arn:aws:sns:us-west-2:565419523791: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=AWS4-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=AWS4-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 --duration 1440
```

The output from the `describe-events` command looks something like this.

```
{
  "Events": [
    {
      "Date": "2017-03-29T22:17:37.781Z",
      "Message": "Added cache node 0001 in Availability Zone us-west-2a",
      "SourceIdentifier": "mem01",
      "SourceType": "cache-cluster"
    },
    {
      "Date": "2017-03-29T22:17:37.769Z",
      "Message": "Cache cluster created",
      "SourceIdentifier": "mem01",
      "SourceType": "cache-cluster"
    }
  ]
}
```

```
}  
]  
}
```

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.

Example ElastiCache SNS Notification

The following example shows an ElastiCache Amazon SNS notification for successfully creating a cache cluster.

Example

```
{
  "Date": "2015-12-05T01:02:18.336Z",
  "Message": "Cache cluster created",
  "SourceIdentifier": "memcache-ni",
  "SourceType": "cache-cluster"
}
```

ElastiCache Events

The following ElastiCache events trigger Amazon SNS notifications:

Event Name	Message	Description
ElastiCache:AddCacheNodeComplete	Finished modifying number of nodes from %d to %d"	A cache node has been added to the cache cluster and is ready for use.
ElastiCache:AddCacheNodeFailed due to insufficient free IP addresses	"Failed to modify number of nodes from %d to %d due to insufficient free IP addresses"	A cache node could not be added because there are not enough available IP addresses.
ElastiCache:CacheClusterParametersChanged	"Changed parameter %s to %s" In case of create, also send "Updated to use a CacheParameterGroup %s"	One or more cache cluster parameters have been changed.
ElastiCache:CacheClusterProvisioningComplete	"Cache cluster created"	The provisioning of a cache cluster is completed, and the cache nodes in the cache cluster are ready to use.

Event Name	Message	Description
ElastiCache:CacheClusterProvisioningFailed due to incompatible network state	"Failed to create the cache cluster due to incompatible network state"	An attempt was made to launch a new cache cluster into a nonexistent virtual private cloud (VPC).
ElastiCache:CacheClusterScalingComplete	"Successfully completed applying modification to cache node type to %s."	: Scale up for cache-cluster completed successfully.
ElastiCache:CacheClusterScalingFailed	"Failed applying modification to cache node type to %s."	Scale-up operation on cache-cluster failed.
ElastiCache:CacheClusterSecurityGroupModified	"Successful change to security group"	<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:CacheNodeReplaceComplete	"Finished recovery for cache nodes %s"	<p>ElastiCache has detected that the host running a cache node is degraded or unreachable and has completed 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:CacheNodesRebooted	"Cache node %s restarted"	One or more cache nodes has been rebooted. Message (Memcached): "Cache node %s shutdown" Then a second message: "Cache node %s restarted"
ElastiCache>DeleteCacheClusterComplete	"Cache cluster deleted"	The deletion of a cache cluster and all associated cache nodes has completed.
ElastiCache:FailoverComplete	"Failover to replica node %s completed"	Failover over to a replica node was successful.
ElastiCache:NodeReplacementCanceled	"Node replacement for Cache Cluster ID: %s, Node ID: %s scheduled during the maintenance window from Start Time: %s, End Time: %s has been canceled"	A node in your cluster that was scheduled for replacement is no longer scheduled for replacement.
ElastiCache:NodeReplacementRescheduled	"Node replacement in maintenance window for node with Cache Cluster ID: %s, Node ID: %s has re-scheduled from Previous Start Time: %s, Previous End Time: %s to New Start Time: %s, New End Time: %s"	A node in your cluster previously scheduled for replacement has been rescheduled for replacement during the new window described in the notification. For information on what actions you can take, see Replacing Nodes (p. 52) .
ElastiCache:NodeReplacementScheduled	"Node with Cache Cluster ID: %s, Node ID: %s is scheduled for replacement during the maintenance window from Start Time: %s, End Time: %s"	A node in your cluster is scheduled for replacement during the window described in the notification. For information on what actions you can take, see Replacing Nodes (p. 52) .
ElastiCache:RemoveCacheNodeComplete	"Removed cache nodes %s"	A cache node has been removed from the cache cluster.
ElastiCache:SnapshotComplete	"Snapshot succeeded for snapshot with ID '%s' of cache cluster with ID '%s' "	A cache snapshot has completed successfully.

Event Name	Message	Description
ElastiCache:SnapshotFailed	"Snapshot failed for snapshot with ID '%s' of cache cluster with ID '%s' "	<p>A cache snapshot has failed. See the cluster's cache events for more a detailed cause.</p> <p>If you describe the snapshot, see DescribeSnapshots, the status will be failed.</p>

Related topics

- [Viewing ElastiCache Events \(p. 213\)](#)

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 `EU (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 and Cost Management 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.

You can add, list, modify, or remove tags from an ElastiCache resource by using the ElastiCache management console, AWS CLI, or ElastiCache API.

Topics

- [Managing Your Tags Using the ElastiCache Console \(p. 220\)](#)
- [Managing Your Cost Allocation Tags Using the AWS CLI \(p. 221\)](#)
- [Managing Your Cost Allocation Tags Using the ElastiCache API \(p. 223\)](#)

Managing Your Tags Using the ElastiCache Console

You can use the Amazon ElastiCache console to add, modify, or remove cost allocation tags.

The following procedure walks you through viewing, adding, modifying, or deleting one or more cost allocation tags using the ElastiCache management console.

Managing tags on a Memcached cluster 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. Choose **Memcached**.
3. Choose the box to the left of the cluster's name you want to add tags to.
4. Choose **Manage Tags**, and then use the dialog box to manage your tags.

Manage Tags [X]

Apply tags to your resources to help organize and identify them.
A tag consists of a case-sensitive key-value pair. For example, you could define a tag with key = Name and value = Webserver. [Learn More](#) about tagging your AWS resources.

Applied Tags

Key	Value	Delete
Service	ElastiCache	<input type="checkbox"/>

Add Tags

Key	Value	
Region	AP-Tokyo	<input type="checkbox"/>
Cost-Center	Empty value	<input checked="" type="checkbox"/>
Add key	Empty value	

[Cancel] [Apply Changes]

5. For each tag you want to add, modify, or remove:

To add, modify, or remove tags

- **To add a tag:** In the **Key** column, type a key name in the box that displays *Add key* and an optional value in the box to the right of the key name.
 - **To modify a tag:** In the **Value** column, type a new value or remove the existing value for the tag.
 - **To remove a tag:** Choose the **X** to the right of the tag.
6. When you're finished, choose **Apply Changes**.

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 \(p. 221\)](#)
- [Adding Tags Using the AWS CLI \(p. 222\)](#)
- [Modifying Tags Using the AWS CLI \(p. 222\)](#)
- [Removing Tags Using the AWS CLI \(p. 223\)](#)

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. 223)
- [Adding Tags Using the ElastiCache API](#) (p. 224)
- [Modifying Tags Using the ElastiCache API](#) (p. 224)
- [Removing Tags Using the ElastiCache API](#) (p. 224)

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>
```

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 – 1 or 3 years. In addition to the up front charge there is an hourly usage charge which is significantly less than the hourly usage charge you incur with On-Demand nodes. To determine whether reserved nodes would be a cost savings for your use cases, determine the node size and number of nodes you need, estimate the usage of the node, then 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](#).

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 Cache Nodes](#).

Topics

- [Understanding Utilization Levels \(p. 226\)](#)
- [Getting Info About Reserved Node Offerings \(p. 228\)](#)
- [Purchasing a Reserved Node \(p. 231\)](#)
- [Getting Info About Your Reserved Nodes \(p. 234\)](#)

Understanding Utilization Levels

There are three levels of node reservations – Heavy Utilization, Medium Utilization, and Light Utilization. Nodes can be reserved at any utilization level for either 1 or 3 years. The node type, utilization level, and reservation term will impact your total costs. You should 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, followed by a lower hourly fee for the duration of the term regardless of whether or not your node is running.

Medium Utilization reserved nodes are the best option if you plan to leverage your reserved nodes a substantial amount of the time, but you want either a lower one-time fee or the flexibility 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, and you can save up to 56 percent off of the On-Demand rates over the entire term of your reserved node.

Reserved Cache 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.
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, lowest overall cost if you anticipate you will 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 \(Console\) \(p. 228\)](#)
- [Getting Info About Reserved Node Offerings \(AWS CLI\) \(p. 228\)](#)
- [Getting Info About Reserved Node Offerings \(ElastiCache API\) \(p. 229\)](#)

Getting Info About Reserved Node Offerings (Console)

To get pricing and other information about available reserved cluster offerings using the AWS Management Console, follow the steps in 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 Cache Nodes**.
3. Choose **Purchase Reserved Cache Node**.
4. From the **Product Description** drop down list box, choose Memcached.
5. To determine the available offerings, make selections from the next 3 drop down list boxes:
 - **Cache Node Type**
 - **Term**
 - **Offering Type**

After you make these selections, the cost per node and total cost of your selections is shown in the **Purchase Reserved Cache Nodes** wizard.

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):

```
{
  "ReservedCacheNodesOfferings": [
    {
      "OfferingType": "Heavy Utilization",
      "FixedPrice": 4328.0,
      "ReservedCacheNodesOfferingId": "0192caa9-daf2-4159-b1e5-a79bb1916695",
      "UsagePrice": 0.0,
      "RecurringCharges": [
        {
```

```
        "RecurringChargeAmount": 0.491,  
        "RecurringChargeFrequency": "Hourly"  
    },  
    ],  
    "ProductDescription": "memcached",  
    "Duration": 31536000,  
    "CacheNodeType": "cache.r3.4xlarge"  
},  
  
***** some output omitted for brevity *****  
  
    {  
        "OfferingType": "Heavy Utilization",  
        "FixedPrice": 4132.0,  
        "ReservedCacheNodesOfferingId": "fb766e0a-79d7-4e8f-a780-a2a6ed5ed439",  
        "UsagePrice": 0.0,  
        "RecurringCharges": [  
            {  
                "RecurringChargeAmount": 0.182,  
                "RecurringChargeFrequency": "Hourly"  
            }  
        ],  
        "ProductDescription": "redis",  
        "Duration": 94608000,  
        "CacheNodeType": "cache.r3.2xlarge"  
    }  
]  
}
```

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>
```

This call returns output similar to the following:

```
<DescribeReservedCacheNodesOfferingsResponse xmlns="http://elasticache.us-  
west-2.amazonaws.com/doc/2013-06-15/">  
  <DescribeReservedCacheNodesOfferingsResult>  
    <ReservedCacheNodesOfferings>  
      <ReservedCacheNodesOffering>  
        <Duration>31536000</Duration>  
        <OfferingType>Medium Utilization</OfferingType>  
        <CurrencyCode>USD</CurrencyCode>  
        <RecurringCharges/>  
        <FixedPrice>1820.0</FixedPrice>
```

```
<ProductDescription>memcached</ProductDescription>
<UsagePrice>0.368</UsagePrice>
<ReservedCacheNodesOfferingId>438012d3-4052-4cc7-b2e3-8d3372e0e706</
ReservedCacheNodesOfferingId>
  <CacheNodeType>cache.m1.large</CacheNodeType>
</ReservedCacheNodesOffering>
<ReservedCacheNodesOffering>

  (...some output omitted for brevity...)

</ReservedCacheNodesOffering>
</ReservedCacheNodesOfferings>
</DescribeReservedCacheNodesOfferingsResult>
<ResponseMetadata>
  <RequestId>5e4ec40b-2978-11e1-9e6d-771388d6ed6b</RequestId>
</ResponseMetadata>
</DescribeReservedCacheNodesOfferingsResponse>
```

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 will incur charges on your AWS account that you cannot reverse.

Topics

- [Purchasing a Reserved Node \(Console\)](#) (p. 231)
- [Purchasing a Reserved Node \(AWS CLI\)](#) (p. 231)
- [Purchasing a Reserved Node \(ElastiCache API\)](#) (p. 232)

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 Cache Nodes** link.
3. Choose the **Purchase Reserved Cache Node** button.
4. Choose the node type from the **Product Description** drop-down list box.
5. Choose the node class from the **Cache Node Class** drop-down list box.
6. Choose length of time you want to reserve the node for from the **Term** drop-down list box.
7. Do either one of the following:
 - Choose the offering type from the **Offering Type** drop-down list box.
 - Enter a reserved node ID in the **Reserved Cache Node ID** text box.

Note

The Reserved Cache Node ID is an unique customer-specified identifier to track this reservation. If this box is left blank, ElastiCache automatically generates an identifier for the reservation.

8. Choose the **Next** button.

The **Purchase Reserved Cache Node** dialog box shows a summary of the reserved node attributes that you've chosen and the payment due.

9. Choose the **Yes, Purchase** button to proceed and purchase the reserved node.

Important

When you choose **Yes, Purchase** you incur the charges for the reserved nodes you selected. To avoid incurring these charges, choose **Cancel**.

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=AWS4-HMAC-SHA256  
&X-Amz-Date=20141201T220302Z  
&X-Amz-SignedHeaders=Host  
&X-Amz-Expires=20141201T220302Z  
&X-Amz-Credential=<credential>  
&X-Amz-Signature=<signature>
```

This call returns output similar to the following:

```
<PurchaseReservedCacheNodesOfferingResponse xmlns="http://elasticache.us-  
west-2.amazonaws.com/doc/2013-06-15/">  
  <PurchaseReservedCacheNodesOfferingResult>  
    <ReservedCacheNode>
```

```
<OfferingType>Medium Utilization</OfferingType>
<CurrencyCode>USD</CurrencyCode>
<RecurringCharges/>
<ProductDescription>memcached</ProductDescription>
<ReservedCacheNodesOfferingId>649fd0c8-cf6d-47a0-bfa6-060f8e75e95f</
ReservedCacheNodesOfferingId>
  <State>payment-pending</State>
  <ReservedCacheNodeId>myreservationID</ReservedCacheNodeId>
  <CacheNodeCount>10</CacheNodeCount>
  <StartTime>2013-07-18T23:24:56.577Z</StartTime>
  <Duration>31536000</Duration>
  <FixedPrice>123.0</FixedPrice>
  <UsagePrice>0.123</UsagePrice>
  <CacheNodeType>cache.m1.small</CacheNodeType>
</ReservedCacheNode>
</PurchaseReservedCacheNodesOfferingResult>
<ResponseMetadata>
  <RequestId>7f099901-29cf-11e1-bd06-6fe008f046c3</RequestId>
</ResponseMetadata>
</PurchaseReservedCacheNodesOfferingResponse>
```

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 \(Console\) \(p. 234\)](#)
- [Getting Info About Your Reserved Nodes \(AWS CLI\) \(p. 234\)](#)
- [Getting Info About Your Reserved Nodes \(ElastiCache API\) \(p. 234\)](#)

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 Cache Nodes** link.

The reserved nodes for your account appear in the Reserved Cache 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=AWS4-HMAC-SHA256
&X-Amz-Date=20141201T220302Z
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20141201T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>
```

This call returns output similar to the following:

```
<DescribeReservedCacheNodesResponse xmlns="http://elasticache.us-west-2.amazonaws.com/doc/2013-06-15/">
  <DescribeReservedCacheNodesResult>
    <ReservedCacheNodes>
      <ReservedCacheNode>
        <OfferingType>Medium Utilization</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <ProductDescription>memcached</ProductDescription>
        <ReservedCacheNodesOfferingId>649fd0c8-cf6d-47a0-bfa6-060f8e75e95f</ReservedCacheNodesOfferingId>
        <State>payment-failed</State>
        <ReservedCacheNodeId>myreservationid</ReservedCacheNodeId>
        <CacheNodeCount>1</CacheNodeCount>
        <StartTime>2010-12-15T00:25:14.131Z</StartTime>
        <Duration>31536000</Duration>
        <FixedPrice>227.5</FixedPrice>
        <UsagePrice>0.046</UsagePrice>
        <CacheNodeType>cache.m1.small</CacheNodeType>
      </ReservedCacheNode>
    </ReservedCacheNodes>
    (...some output omitted for brevity...)
  </DescribeReservedCacheNodesResult>
  <ResponseMetadata>
    <RequestId>23400d50-2978-11e1-9e6d-771388d6ed6b</RequestId>
  </ResponseMetadata>
</DescribeReservedCacheNodesResponse>
```

For more information, see [DescribeReservedCacheNodes](#) in the ElastiCache API Reference.

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 \(p. 236\)](#)
- [ElastiCache API Reference](#)
- [ElastiCache section of the AWS CLI Reference](#)
- [Amazon ElastiCache Error Messages \(p. 246\)](#)
- [Notifications \(p. 247\)](#)

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 \(p. 236\)](#)
- [Available Libraries \(p. 238\)](#)
- [Troubleshooting Applications \(p. 239\)](#)
- [Logging ElastiCache API Calls with AWS CloudTrail \(p. 239\)](#)

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=AWS4-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=AWS4-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=2877960fced9040b41b4feaca835fd5cfef9264f768e6a0236c9143f915ffa56
```

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. In order 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/>.

Logging ElastiCache API Calls with AWS CloudTrail

ElastiCache is integrated with AWS CloudTrail, a service that provides a record of actions taken by a user, role, or an AWS service in ElastiCache. CloudTrail captures all API calls for ElastiCache as events, including calls from the ElastiCache console and from code calls to the ElastiCache APIs. If you create a trail, you can enable continuous delivery of CloudTrail events to an Amazon S3 bucket, including events for 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 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](#).

ElastiCache Information in CloudTrail

CloudTrail is enabled on your AWS account when you create the account. When activity occurs in 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 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:

- [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 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 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": "Amazon 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 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": "Amazon CLI/ElastiCache 1.10 API 2014-12-01",
  "requestParameters": {
  },
  "responseElements": null
}
```

```
"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":"Amazon 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.us1prod.cache.amazonaws.com",
      "port":11211
    },
    "pendingModifiedValues":{
      "numCacheNodes":3
    }
  },
}
```

```
}  
  "requestID": "807f4bc3-354c-11e4-9376-c1d979ba565a",  
  "eventID": "e9163565-376f-4223-96e9-9f50528da645"
```

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 \(p. 243\)](#)
- [Getting the Command Line Tools \(p. 244\)](#)
- [Setting Up the Tools \(p. 244\)](#)
- [Providing Credentials for the Tools \(p. 245\)](#)
- [Environmental Variables \(p. 246\)](#)

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 \(p. 246\)](#)
- [Customer's node quota exceeded \(p. 247\)](#)
- [Insufficient cache cluster capacity \(p. 247\)](#)

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. 120).
- Try purchasing Reserved Nodes, which are a long-term capacity reservation. For more information, see [ElastiCache Reserved Nodes](#) (p. 48).

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. 247)

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:

ElastiCache for Memcached Documentation History

- **API version:** 2015-02-02
- **Latest documentation update:** April 20, 2018

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

update-history-change	update-history-description	update-history-date
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 .	December 14, 2018
Support for ElastiCache for Redis 5.0.0	ElastiCache for Redis now supports Redis 5.0.0, including Redis streams. For more information, see ElastiCache for Redis Version 5.0.0 (Enhanced) . It has also added a new metric, <code>StreamBasedCmds</code> , which reports the sum of all of the commands that act upon one or more streams data type. For more information, see Metrics for Redis .	December 9, 2018
ElastiCache for Redis support for M5 and R5 nodes	ElastiCache for Redis now supports M5 and R5 nodes, our AWS Nitro System based general-purpose and memory-optimized instance types. For more information, see Supported Node Types .	October 23, 2018
Support for dynamically changing the number of read replicas	ElastiCache for Redis has added support for adding and removing read replicas from any cluster with no cluster downtime. For more information about these and other changes in this release, see Changing the Number of Replicas in the <i>ElastiCache for Redis User Guide</i>	September 17, 2018

	with DecreaseReplicaCount and IncreaseReplicaCount in the <i>ElastiCache API Reference</i> .	
FedRAMP compliance certification	ElastiCache for Redis is now certified for FedRAMP compliance. For more information, see ElastiCache for Redis FedRAMP Compliance .	August 30, 2018
Redis (cluster mode enabled) engine upgrades	Amazon ElastiCache for Redis has added support for upgrading Redis (cluster mode enabled) engine versions. For more information, see Upgrading Engine Versions .	August 20, 2018
PCI DSS compliance certification	ElastiCache for Redis is now certified for PCI DSS compliance. For more information, see ElastiCache for Redis PCI DSS Compliance .	July 5, 2018
Support for ElastiCache for Redis 4.0.10	ElastiCache for Redis now supports Redis 4.0.10, including both encryption and online cluster resizing in a single version. For more information, see ElastiCache for Redis Version 4.0.10 (Enhanced) .	June 14, 2018
User Guide restructure (p. 249)	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
Support for EngineCPUUtilization metric	ElastiCache for Redis added a new metric, <code>EngineCPUUtilization</code> , which reports the percentage of your CPU's capacity that is currently being used. For more information, see Metrics for Redis .	April 9, 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).	ElastiCache added support for the Asia Pacific (Osaka-Local) region. The Asia Pacific (Osaka-Local) region currently supports a single availability zone and is by invitation only. For more information, see: <ul style="list-style-type: none"> • Supported Regions & Endpoints (p. 41) • Supported Node Types (p. 49) 	February 12, 2018
Support for EU (Paris).	ElastiCache added support for the EU (Paris) region. For more information, see: <ul style="list-style-type: none"> • Supported Regions & Endpoints (p. 41) • Supported Node Types (p. 49) 	December 18, 2017
Support for China (Ningxia) Region	Amazon ElastiCache added support for China (Ningxia) Region. For more information, see: <ul style="list-style-type: none"> • Supported Regions & Endpoints (p. 41) • Supported Node Types (p. 49) 	December 11, 2017
Support for Service Linked Roles	This release of ElastiCache added support for Service Linked Roles (SLR). For more information, see: <ul style="list-style-type: none"> • Using Service-Linked Roles for ElastiCache (p. 191) • Set Up Your Permissions (New ElastiCache Users Only) (p. 16) 	December 7, 2017
Support for R4 node types	This release of ElastiCache added support R4 node types in all regions supported by ElastiCache. You can purchase R4 node types as On-Demand or as Reserved Cache Nodes. For more information, see: <ul style="list-style-type: none"> • Supported Node Types (p. 49) • Memcached Node-Type Specific Parameters (p. 147) 	November 20, 2017
Connection patterns topic	ElastiCache documentation adds a topic covering various patterns for accessing an ElastiCache cluster in an Amazon VPC. For more information, see Access Patterns for Accessing an ElastiCache Cluster in an Amazon VPC (p. 154) in the <i>ElastiCache User Guide</i> .	April 24, 2017
Support for Memcached 1.4.34	ElastiCache adds support Memcached version 1.4.34, which incorporates a number of fixes to earlier Memcached versions. For more information, see Memcached 1.4.34 Release Notes at Memcached on GitHub.	April 10, 2017
Support for Memcached 1.4.33	ElastiCache adds support for Memcached version 1.4.33. For more information, see: <ul style="list-style-type: none"> • Memcached Version 1.4.33 (p. 37) 	December 20, 2016

Change	Description	Date Changed
	<ul style="list-style-type: none"> • Memcached 1.4.33 Added Parameters (p. 140) 	
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:</p> <ul style="list-style-type: none"> • Supported Regions & Endpoints (p. 41) • Supported Node Types (p. 49) 	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 region. For more information, see:</p> <ul style="list-style-type: none"> • Supported Regions & Endpoints (p. 41) • Supported Node Types (p. 49) 	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:</p> <ul style="list-style-type: none"> • Supported Regions & Endpoints (p. 41) • Supported Node Types (p. 49) 	November 1, 2016
US East 2 (Ohio) Region support	<p>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:</p> <ul style="list-style-type: none"> • Supported Regions & Endpoints (p. 41) • Supported Node Types (p. 49) 	October 17, 2016
M4 node type support	<p>ElastiCache adds support for the M4 family of node types in most regions supported by ElastiCache. You can purchase M4 node types as On-Demand or as Reserved Cache Nodes. For more information, see:</p> <ul style="list-style-type: none"> • Supported Node Types (p. 49) • Memcached Node-Type Specific Parameters (p. 147) 	August 3, 2016
Mumbai Region support	<p>ElastiCache adds support for the Asia Pacific (Mumbai) Region. For more information, see:</p> <ul style="list-style-type: none"> • Supported Node Types (p. 49) • Memcached Node-Type Specific Parameters (p. 147) 	June 27, 2016
Support for R3 node types	<p>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. 49).</p>	March 16, 2016
Accessing ElastiCache using a Lambda function	<p>Added a tutorial on configuring a Lambda function to access ElastiCache in an Amazon VPC. For more information, see ElastiCache Tutorials and Videos (p. 24).</p>	February 12, 2016

Change	Description	Date Changed
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: <ul style="list-style-type: none"> Memcached release notes Memcached Version 1.4.24 (p. 38) in the <i>ElastiCache for Memcached User Guide</i> 	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. 78) .	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. 112) .	July 9, 2015
Node replacement messages added	ElastiCache adds three messages pertaining to scheduled node replacement, ElastiCache:NodeReplacementScheduled, ElastiCache:NodeReplacementRescheduled, and ElastiCache:NodeReplacementCanceled. For more information and actions you can take when a node is scheduled for replacement, see ElastiCache's Event Notifications and Amazon SNS (p. 215) .	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. 219) .	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 EU (Frankfurt) Region	ElastiCache adds support for the EU (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 ElastiCache API Calls with AWS CloudTrail (p. 239) .	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. 123) .	September 11, 2014

Change	Description	Date Changed
Flexible node placement supported for Memcached	ElastiCache adds support for creating Memcached nodes across multiple Availability Zones. For more information, see Step 1: Launch a Memcached 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. 123) .	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. 71) .	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. 149) .	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
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. 149) .	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. 123) .	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*.