Amazon ElastiCache

User Guide

API Version 2015-02-02

Amazon ElastiCache: User Guide

Table of Contents

What Is Amazon ElastiCache?

Welcome to the *Amazon ElastiCache User Guide*. ElastiCache is a web service that makes it easy to set up, manage, and scale a distributed in-memory 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.

With ElastiCache, you can quickly deploy your cache environment, without having to provision hardware or install software. You can choose from Memcached or Redis protocol-compliant cache engine software, and let ElastiCache perform software upgrades and patch management for you. For enhanced security, ElastiCache can be run in the Amazon Virtual Private Cloud (Amazon VPC) environment, giving you complete control over network access to your clusters. With just a few clicks in the AWS Management Console, you can add or remove resources such as nodes, clusters, or read replicas to your ElastiCache environment to meet your business needs and application requirements.

Existing applications that use Memcached or Redis can use ElastiCache with almost no modification. Your applications simply need to know 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, rather than having 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 has multiple features to enhance reliability for critical production deployments:

- Automatic detection and recovery from cache node failures.
- Automatic failover (Multi-AZ) of a failed primary cluster to a read replica in Redis replication groups.
- Flexible Availability Zone placement of nodes and clusters.
- Integration with other AWS services such as Amazon EC2, CloudWatch, CloudTrail, and Amazon SNS to provide a secure, high-performance, managed in-memory caching solution.

See Also

Performance at Scale with Amazon ElastiCache

When Should I Use ElastiCache?

Whether serving up the latest news, a Top-10 leaderboard, 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 impacted by the speed at which you deliver content. According to research reported by the NY Times in 2012, "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 and will 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 100ms (1/10 second) increase in load time, sales would decrease 1%. If someone wants data, whether for a webpage or a report that drives business decisions, you can deliver that data faster if it is cached, much faster. Can your business afford to not cache your web pages so as to deliver them with the shortest latency possible?

It may be intuitively obvious that you want to cache your most heavily requested items. But why would you not want to 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. Remember, *noticeably slower* is what sends customers elsewhere.

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

In-Memory Data Cache

The primary purpose of an in-memory key-value store is to provide ultra-fast (sub-millisecond 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 will always be 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.

What Should I Cache?

When deciding what data to cache you should consider these factors:

Speed and Expense – It is 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 is a candidate for caching. If acquiring the data requires a relatively quick and simple query, it may 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 wouldn't make sense to cache web pages that return the results of a search since 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 may be able to tolerate stale data in one context, but not another. For example, when serving up a publicly traded stock price on a web site, staleness might be quite acceptable, along with a disclaimer that prices may 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:

- 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 Caching Strategies (p. 70).

Gaming Leaderboards (Redis Sorted Lists)

Redis sorted sets move the computational complexity associated with leaderboards from your application to your Redis cluster.

Leaderboards, such as the Top 10 scores for a game, are computationally complex, especially with a large number of concurrent players and continually changing scores. Redis sorted sets guarantee both uniqueness and element ordering. Using Redis sorted sets, each time a new element is added to the sorted set it is re-ranked in real time and added to the set in its appropriate numeric position.

Example - Redis Leaderboard

In this example four gamers and their scores are entered into a sorted list using `ZADD`. The command `ZREVRANGEBYSCORE` lists the players by their score, high to low. Next, `ZADD` is used to update June's score by overwriting the existing entry. Finally `ZREVRANGEBYSCORE` list the players by their score, high to low, showing that June has moved up in the rankings.

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

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

ZADD leaderboard 232 June

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

The following command lets June know where she ranks among all the players. Since ranking is zero-based, *ZREVRANK* returns a 1 for June who is in second position.

```
ZREVRANK leaderboard June
1
```

For more information, see the Redis Documentation on sorted sets.

Messaging (Redis pub/sub)

When you send an email message, you send it to one or more specified recipients. In the pub/sub paradigm, you send a message to a specific channel not knowing who, if anyone, will receive it.

Recipients of the message are those who are subscribed to the channel. For example, suppose you subscribe to the *news.sports.golf* channel. You and all others subscribed to the *news.sports.golf* channel will receive any messages published to *news.sports.golf*.

Redis pub/sub functionality has no relation to any key space. Therefore, it will not interfere on any level.

Subscribing

To receive messages on a channel you must subscribe to the channel. You may subscribe to a single channel, multiple specified channels, or all channels that match a pattern. To cancel a subscription you unsubscribe from the channel specified when you subscribed to it or the same pattern you used if you subscribed using pattern matching.

Example - Subscription to a Single Channel

To subscribe to a single channel, use the SUBSCRIBE command specifying the channel you want to subscribe to. In the following example, a client subscribes to the *news.sports.golf* channel.

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

After a while, the client cancels their subscription to the channel using the UNSUBSCRIBE command specifying the channel to unsubscribe from.

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

Example - Subscriptions to Multiple Specified Channels

To subscribe to multiple specific channels, list the channels with the SUBSCRIBE command. In the following example, a client subscribes to both the *news.sports.golf*, *news.sports.soccer* and *news.sports.skiing* channels.

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

To cancel a subscription to a specific channel, use the UNSUBSCRIBE command specifying the channel to unsubscribe from.

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

To cancel subscriptions to multiple channels, use the UNSUBSCRIBE command specifying the channels to unsubscribe from.

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

To cancel all subscriptions, use UNSUBSCRIBE and specify each channel or UNSUBSCRIBE without specifying any channel.

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

```
UNSUBSCRIBE
```

Example - Subscriptions Using Pattern Matching

Clients can subscribe to all channels that match a pattern by using the PSUBSCRIBE command.

In the following example, a client subscribes to all sports channels. Rather than listing all the sports channels individually, as would be done using SUBSCRIBE, pattern matching is used with the `PSUBSCRIBE` command.

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

To cancel subscriptions to these channels, use the PUNSUBSCRIBE command.

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

Important
The channel string sent to a [P]SUBSCRIBE command and to the [P]UNSUBSCRIBE command must match. You cannot PSUBSCRIBE to *news.** and PUNSUBSCRIBE from *news.sports.** or UNSUBSCRIBE from *news.sports.golf*.

Publishing

To send a message to all subscribers to a channel, use the PUBLISH command, specifying the channel and the message. The following example publishes the message, "It's Saturday and sunny. I'm headed to the links." to the *news.sports.golf* channel.

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

A client cannot publish to a channel to which it is subscribed.

For more information, see Pub/Sub in the Redis documentation.

Recommendation Data (Redis Counters & Hashes)

Redis counters and hashes make compiling recommendations simple. Each time a user "likes" a product, you increment an *item:productID:like* counter. Each time a user "dislikes" a product, you increment an *item:productID:dislike* counter. Using Redis hashes, you can also maintain a list of everyone who has liked or disliked a product.

Example - Likes & Dislikes

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

Other Redis Uses

An article by Salvatore Sanfilippo (How to take advantage of Redis just adding it to your stack) discusses a number of common database uses and how they can be easily solved using Redis, thus removing load from your database and improving performance.

Testimonials

Go to Testimonials to read about how businesses like airbnb, PBS, esri, and others are leveraging Amazon ElastiCache to grow their businesses with improved customer experience.

Amazon ElastiCache 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 Tutorial Videos (p. 7) section has videos that introduce you to Amazon ElastiCache, cover common use cases for ElastiCache, and demo how to use ElastiCache to reduce latency and improve throughput of your applications.
- **Getting Started** – The Getting Started with Amazon ElastiCache (p. 21) 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:

- Engines and Versions (p. 32)

  ElastiCache supports two engines—Memcached and Redis. This topic helps you determine which engine is best for your scenario.
- Selecting Your Node Size (p. 78)

  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. This topic assists you in selecting the best node size.
- Best Practices for Implementing Amazon ElastiCache (p. 60)

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

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

- AWS Command Line Interface Documentation

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

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

You can write application programs to leverage the ElastiCache API using 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. You can code against 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. 383)

  If you don't want to use the AWS SDKs, you can interact with ElastiCache directly using the Query API. This section provides troubleshooting tips and information on creating and authenticating requests and handling responses.

- Amazon ElastiCache API Reference

  This is a separate document with all of the ElastiCache API operations, including syntax and examples.

ElastiCache Tutorial Videos

This section contains tutorial videos to help you learn basic and advanced Amazon ElastiCache concepts. For information about AWS Training, see AWS Training & Certification.

Introductory Video Tutorials

For introductory video tutorials about Amazon ElastiCache, see the following.

Topics

Introduction to Amazon ElastiCache

In this tutorial, you learn about key Amazon ElastiCache concepts, watch a demo of creating and launching an ElastiCache cluster in the Amazon cloud, and then go practice with a free lab at Qwik Labs.

Introduction to Amazon ElastiCache.

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

In this session, we discuss the benefits of NoSQL databases and take a tour of the main NoSQL services offered by AWS—Amazon DynamoDB and Amazon ElastiCache. Then, we hear from two leading customers, Expedia and Mapbox, about their use cases and architectural challenges, and how they addressed them using AWS NoSQL services, including design patterns and best practices. You will walk 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 tutorial, learn how you can use Amazon ElastiCache to easily deploy a Memcached- or Redis-compatible 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 Video Tutorials

For advanced tutorials videos about Amazon ElastiCache, see the following.

Topics

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 tutorial, 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 tutorial, 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).

ElastiCache Terminology

In October 2016, Amazon ElastiCache launched support for Redis 3.2 which, among other things, added support for partitioning your data. In order to preserve compatibility with previous versions, we are extending our current API/CLI (API version 2015-02-02) operations to include the new Redis functionality. In parallel, we are using terminology in the ElastiCache console that is used in this new functionality and common across the industry. This means that at some points, the terminology used in the API/CLI may be different from the terminology used in the console. The following table identifies terms which may differ between the API/CLI and the console.

Cache Cluster/Node vs. Node

Because of the one-to-one relationship between a Node and a Cache Cluster when there are no replica nodes, the ElastiCache console often used the terms interchangeably. Going forward, the console now uses the term Node throughout. The one exception is the button **Create Cluster**, which launches the process to create a cluster with or without replica nodes.

There is no change in the ElastiCache API or AWS CLI terms.

Cache xxx vs. xxx

The console has dropped the word *cache* from terms like *cache node*, *cache cluster*, *cache parameter group*, and so on. The console now refers to these as *node*, *cluster*, *parameter group*, and so on.

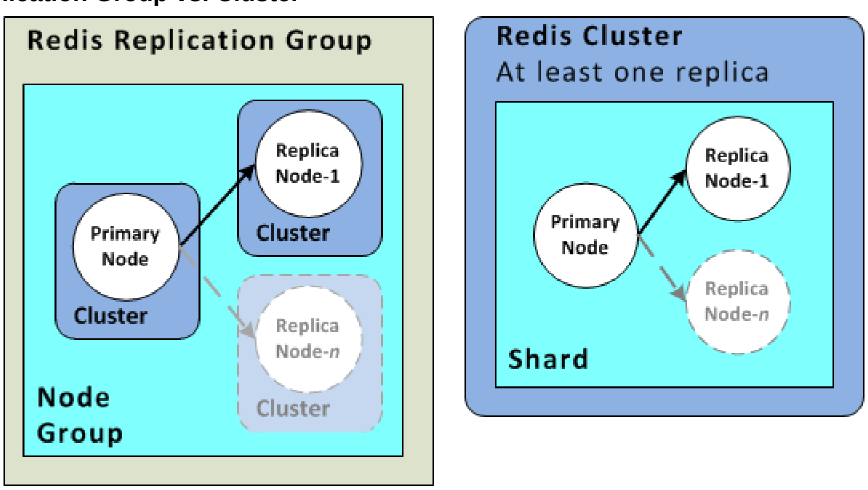
There is no change in the ElastiCache API or AWS CLI terms.

Node Group vs. Shard

Node groups and shards exist only when there are read replica nodes. *Shard* and *Node Group* are similar. Both implement replication, but with slightly different internal structures. A shard is a collection of up to 6 Redis nodes. A node group is a collection of up to 6 Redis single node clusters. In both cases, replication is implemented by having one node/cluster be the read/write primary and all other nodes/clusters read-only replicas. For a visual comparison, see the graphic in the following term, *Replication Group vs. Cluster*. The ElastiCache console no longer uses the term *replication group*, replacing it with *cluster*.

There is no change in the ElastiCache API or AWS CLI terms.

Replication Group vs. Cluster



A Redis cluster (called *replication group* in the API/CLI) is a collection of shards (called *node groups* in the API/CLI) as defined above.

A Redis (cluster mode disabled) cluster never has more than one shard (called *node group* in the API/CLI). A Redis (cluster mode enabled) cluster can partition its data across 1 to 15 shards (called *node groups* in the API/CLI).

There is no change in the ElastiCache API or AWS CLI terms.

ElastiCache Components & Features

The topics in this section are an overview of the major components of an Amazon ElastiCache deployment.

Topics

ElastiCache Nodes

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

A node is a fixed-size chunk of secure, network-attached RAM. Each node runs an instance of either Memcached or Redis, depending on which was selected when you created your cluster. If necessary, you can scale the nodes in a cluster up or down to a different instance type. For more information, see Scaling (p. 169).

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. 90).

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 could save you money. If your cluster is almost always in use and you occasionally add nodes to handle use spikes, 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. 81).

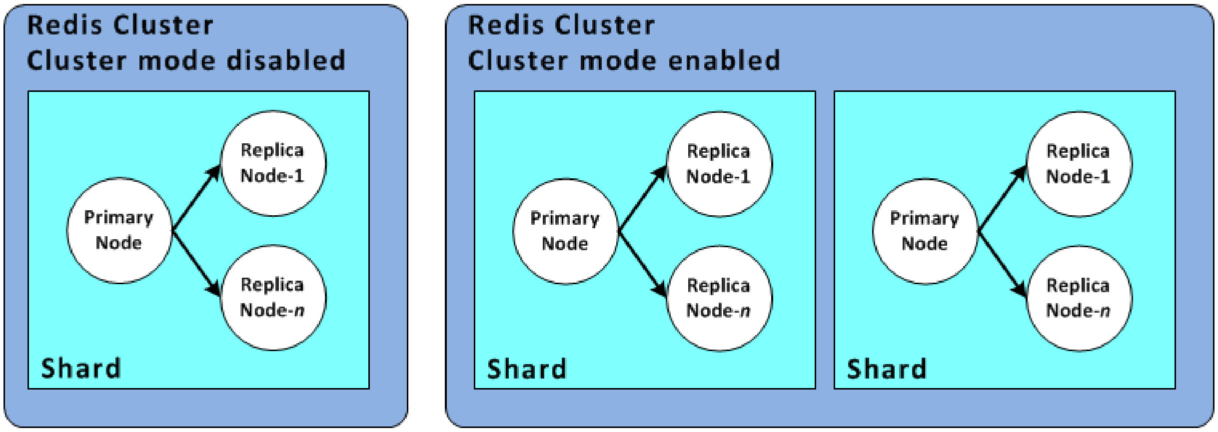
The Memcached engine supports Auto Discovery–the ability for client programs to automatically identify all of the nodes in a cache cluster, and to initiate and maintain connections to all of these nodes. With Auto Discovery, your application does not need to manually connect to individual nodes; instead, 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, you application immediately *knows* about all of the nodes in the cluster and can connect to all of them. You do not need to hard code the individual cache node endpoints in your application. For more information on Auto Discovery, see Node Auto Discovery (Memcached) (p. 95).

For more information on nodes, see ElastiCache Nodes (p. 76).

ElastiCache Shards (Redis)

A Redis shard (called *node group* in the API/CLI) is a grouping of 1 to 6 related nodes. A Redis (cluster mode disabled) cluster always has one shard. A Redis (cluster mode enabled) cluster can have from 1 to 15 shards.

A multiple node shard implements replication by have one read/write primary node and 1 to 5 replica nodes. For more information, see ElastiCache Replication (Redis) (p. 190).



Redis shard configurations

For more information on shards, see Shards (Redis) (p. 125).

ElastiCache Clusters

A *cluster* is a logical grouping of one or more ElastiCache Nodes (p. 11) or ElastiCache Shards (Redis) (p. 12). Data is partitioned across the shards in a Redis (cluster mode enabled) cluster.

Many ElastiCache operations are targeted at clusters.

- Creating a cluster
- Modifying a cluster
- Taking snapshots of a cluster (all versions of Redis)
- Deleting a cluster
- Viewing the elements in a cluster
- Adding or removing cost allocation tags to/from a cluster

For more detailed information, see the following related topics:

- ElastiCache Clusters (p. 126) and ElastiCache Nodes (p. 76)

  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. 62)

Information about improving the fault tolerance of your clusters and replication groups.

Typical Cluster Configurations

Depending on the engine you select, possible cluster configurations differ.

Memcached supports up to 100 nodes per customer per region with each cluster having 1 to 20 nodes. You can partition your data across the nodes in a Memcached cluster.

A Redis cluster contains 1 to 15 shards (API: node groups), each of which is a partition of your data. Redis (cluster mode disabled) always has just one shard.

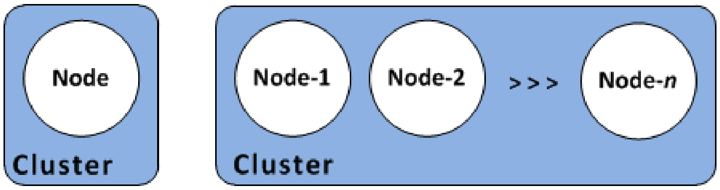
Following are typical cluster configurations for the Memcached and Redis engines.

Memcached Clusters

When you run the Memcached engine, clusters can be made up of 1 to 20 nodes. You can partition your database across the nodes. Your application reads and writes to each node's endpoint. For more information, see Node Auto Discovery (Memcached) (p. 95).

For improved fault tolerance, locate your Memcached nodes in various Availability Zones (AZs) within the cluster's region. That way, a failure in one AZ will have minimal impact upon your entire cluster and application. For more information, see Mitigating Failures (p. 62).

As demand upon your Memcached cluster changes, you can scale out or in by adding or removing nodes and repartitioning 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. 66).



A single node and a multi-node Memcached cluster

Memcached clusters: single node and multiple node clusters

Redis Clusters

A Redis cluster contains 1 to 15 shards (called *node groups* in the API/CLI). Redis (cluster mode disabled) clusters always contain just one shard (called *node group* in the API/CLI). A Redis shard contains 1 to 6 nodes. If there is more than one node in a shard, the shard supports replication with one node being the read/write primary node and the others read-only replica nodes.

For improved fault tolerance, we recommend having at least two nodes in a Redis cluster and enabling Multi-AZ with automatic failover. For more information, see Mitigating Failures (p. 62).

As demand upon your Redis (cluster mode disabled) cluster changes you can scale up or down by moving your cluster to a different node instance type. If your application is read intensive, we recommend adding read-only replicas Redis (cluster mode disabled) cluster so you can spread the reads across a more appropriate number of nodes.

ElastiCache supports changing a Redis (cluster mode disabled) cluster's node type to a larger node type dynamically. For information on scaling up or down, see Scaling Redis (cluster mode disabled) Clusters (p. 173) or Scaling Redis Clusters with Replica Nodes (p. 180).

ElastiCache Replication (Redis)

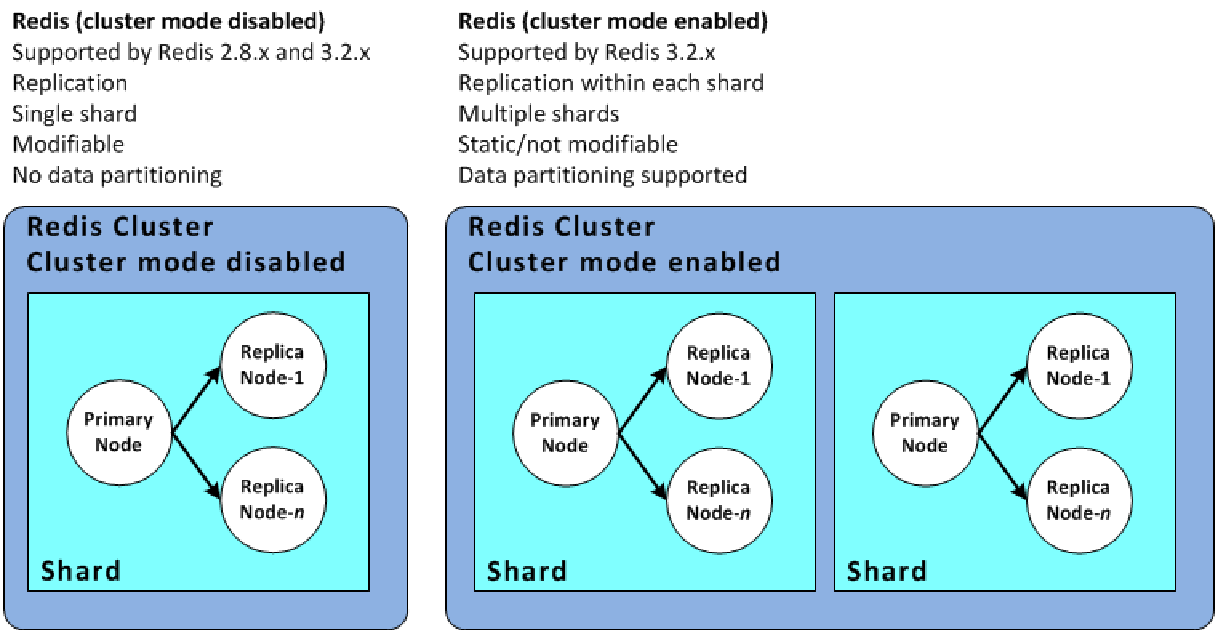
Before you continue reading here, see ElastiCache Terminology (p. 9) to better understand the differences in terminology between the ElastiCache console and the API/CLI.

Replication is implemented by grouping from 2 to 6 nodes in a shard (API/CLI: node group). One of these nodes is the read/write primary node. All the other nodes are read-only replica nodes.

Each replica node maintains a copy of the data from the primary node. Replica nodes use asynchronous replication mechanisms to keep synchronized with the primary node. Applications can read from any node in the cluster but can write only to primary nodes. Read replicas enhance scalability by spreading reads across multiple endpoints. Read replicas also improve fault tolerance by maintaining multiple copies of the data. Locating read replicas in multiple Availability Zones further improves fault tolerance. For more information on fault tolerance, see Mitigating Failures (p. 62).

Redis (cluster mode disabled) clusters support one shard (called *node group* in the API/CLI). Redis (cluster mode enabled) clusters support from 1 to 15 shards (node groups).

The following graphic illustrates replication for Redis (cluster mode disabled) and Redis (cluster mode enabled) clusters using the console's view and terminology.



Redis replication (console view), single shard and multiple shards

Replication from the API/CLI perspective uses different terminology to maintain compatibility with previous versions, but the results are the same. The following diagram shows the API/CLI terms for implementing replication.

Comparing Replication: Redis (cluster mode disabled) & Redis (cluster mode enabled)

The following table compares various features of Redis (cluster mode disabled) and Redis (cluster mode enabled) replication groups.

	Redis (cluster mode disabled)	Redis (cluster mode enabled)
Node groups	1	1 to 15

	Redis (cluster mode disabled)	**Redis (cluster mode enabled)**
Replicas per node group #	0 to 5	0 to 5
Data partitioning	No	Yes
Add/Delete replicas	Yes	No
Add/Delete node groups	No	No
Supports scale up	Yes	No
Supports engine upgrades	Yes	N/A
Promote replica to primary	Yes	No
Multi-AZ with automatic failover	Optional	Required
Backup/Restore	Yes #	Yes #
Notes:		
# If any primary has no replicas and the primary fails, you will lose all that primary's data.		
# Backup/Restore must be to a replication group with the same number of node groups.		

All of the shards (API/CLI: node groups) and nodes must reside in the same region. However, you can provision the individual nodes in multiple Availability Zones within that region.

Read replicas guard against potential data loss because your data is replicated over two or more nodes – the primary and one or more read replicas. For greater reliability and faster recovery, we recommend that you create one or more read replicas in different Availability Zones, and enable Multi-AZ with automatic failover instead of using AOF. AOF is disabled when Multi-AZ with automatic failover is enabled. For more information, see Replication: Multi-AZ with Automatic Failover (Redis) (p. 196).

Replication: Limits & Exclusions

- AOF is not supported on node type `cache.t1.micro`.
- Multi-AZ with automatic failover is only supported on Redis versions 2.6.8 and later.
- Multi-AZ with automatic failover is not supported on node types T1 and T2.

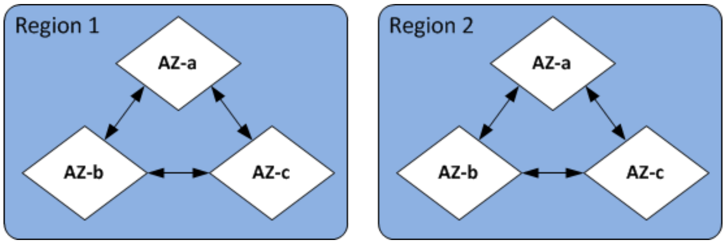
For more information on AOF and Multi-AZ, see Mitigating Failures (p. 62).

Regions & Availability Zones

Amazon ElastiCache is available in multiple regions around the world so that you can launch ElastiCache clusters in locations that meet your business 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 ElastiCache console.

Each region is designed to be completely isolated from the other regions. Within each region are multiple Availability Zones. By launching your nodes in different Availability Zones you are able to achieve the greatest possible fault tolerance. For more information about regions and Availability Zones, see Selecting Regions and Availability Zones (p. 45).



Regions and Availability Zones

For information on regions supported by ElastiCache and their endpoints, see Supported Regions & Endpoints (p. 46).

ElastiCache Endpoints

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

Memcached Endpoints

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 will automatically *know* each node endpoint, even after adding or removing nodes from the cluster. For more information, see Node Auto Discovery (Memcached) (p. 95).

Single Node Redis Cluster Endpoints

The endpoint for a single node Redis cluster is used to connect to the cluster for both reads and writes.

Multi-Node Redis Cluster Endpoints

A multiple node Redis (cluster mode disabled) cluster has two types of endpoints. The primary endpoint always connects to the primary node in the cluster, even if the specific node in the primary role changes. Use the primary endpoint for all writes to the cluster.

The read endpoint in a Redis (cluster mode disabled) cluster always points to a specific node. Whenever you add or remove a read replica, you must update the associated node endpoint in your application.

A Redis (cluster mode enabled) cluster has a single configuration endpoint. By connecting to the configuration endpoint your application is able to discover the primary and read endpoints for each shard in the cluster.

For more information, see Finding Your ElastiCache Endpoints (p. 48).

ElastiCache Parameter Groups

Cache parameter groups are an easy way to manage runtime settings for supported engine software. Memcached and Redis have many parameters to control memory usage, eviction policies, item sizes, and more. An ElastiCache parameter group is a named collection of Memcached- or Redis-specific parameters that you can apply to a cluster, thereby guaranteeing 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 Parameters and Parameter Groups (p. 272).

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 Virtual Private Cloud (Amazon VPC) with ElastiCache (p. 316).

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, create a security group and use the AuthorizeCacheSecurityGroupIngress API or the authorize-cache-security-group-ingress AWS CLI command 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 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-Classic] (p. 263).

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 select 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 Virtual Private Cloud (Amazon VPC) with ElastiCache (p. 316), Step 4: Authorize Access (p. 24), and Subnets and Subnet Groups (p. 307).

ElastiCache Backups/Snapshots (Redis)

A backup is a point-in-time copy of a Redis cluster. Backups can be used to restore an existing cluster or to seed a new cluster. Backups consist of all the data in a cluster plus some metadata. Backups are not supported by the Memcached engine.

During the time Redis is creating a backup, to keep it a point-in-time backup, the process forks and all writes to the cluster are recorded in available memory, apart from the cluster's data that is being backed up. Because of this, you must have sufficient extra memory to accommodate these writes. When selecting a node type, keep this in mind if you're using Redis. For more information on selecting a node type for your Redis deployment, see Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60).

For more information, see ElastiCache Backup & Restore (Redis) (p. 235).

ElastiCache Events

When significant events happen on a cache 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.

For more information on ElastiCache events, see Monitoring ElastiCache Events (p. 363).

Accessing Amazon ElastiCache

Your Amazon ElastiCache instances can only 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, 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.

For more information on granting Amazon EC2 access to your cluster, see Step 4: Authorize Access (p. 24) and Accessing ElastiCache Resources from Outside AWS (p. 348).

Managing ElastiCache

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.

Managing ElastiCache (Console)

The AWS Management Console is the easiest way to manage Amazon ElastiCache. 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*.

Managing ElastiCache (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 Command Line Interface Reference.

Managing ElastiCache (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.

Managing ElastiCache (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. 383).

Getting Started with Amazon ElastiCache

Beginning with creating your own AWS account, the topics in this section walk you through the process of creating, granting access to, connecting to, and finally deleting a standalone Redis (cluster mode disabled) cluster using the ElastiCache console.

Amazon ElastiCache supports high availability through the use of Redis replication groups. For information about Redis replication groups and how to create them, see ElastiCache Replication (Redis) (p. 190).

Beginning with Redis version 3.2, ElastiCache Redis supports partitioning your data across multiple node groups, with each node group implementing a replication group. This exercise creates a standalone Redis cluster.

Topics

Step 1: Create an AWS Account [One time]

To use ElastiCache, you need an AWS account. If you don't already have one, you'll be prompted to create one when you sign up. You're not charged for any AWS services that you sign up for unless you use them.

To create an AWS account

1. Open https://aws.amazon.com/, and then choose **Create an AWS Account**.
2. Follow the online instructions.

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

Step 2: Launch a Cluster

Before you continue, be sure you have completed Step 1: Create an AWS Account [One time] (p. 21).

The cluster you're about to launch will be live, and not running in a sandbox. You will incur the standard ElastiCache usage fees for the instance until you delete it. The total charges will be 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 will be 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. 308).

To create a standalone Redis (cluster mode disabled) cluster

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Select **Get Started Now**.

   If you already have an available cluster, select **Launch Cluster**.
3. For **Cluster engine**, select **Redis**.
4. Make sure **Cluster Mode enabled (Scale Out)** is not selected.
5. Complete the **Redis 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 Redis engine version you want to run on this cluster. Unless you have a specific reason to run an older version, we recommend that you select the latest version.

   c. In **Port**, accept the default port, 6379. If you have a reason to use a different port, enter the port number.

   d. From **Parameter group**, choose the parameter group you want to use with this cluster, or select "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. 273).

   e. For **Node type**, choose the node type that you want to use for this cluster. For this exercise, above the table select the **t2** instance family, choose **cache.t2.small**, and finally choose **Save**.

      For more information, see Selecting Your Node Size (p. 78).

   f. From **Number of replicas**, choose the number of read replicas you want for this cluster. Since in this exercise we're creating a standalone cluster, select *None*.

      When you select *None*, the **Replication group description** field disappears.
6. Choose **Advanced Redis settings** and complete the section as follows:

Note
The **Advanced Redis settings** details are slightly different if you are creating a Redis (cluster mode enabled) replication group. For a step-by-step walk through to create a Redis (cluster mode enabled) replication group, see Creating a Redis (cluster mode enabled) Replication Group from Scratch (p. 215).

a. From the **Subnet group** list, select the subnet you want to apply to this cluster. For this exercise, select *default*.

   For more information, see Subnets and Subnet Groups (p. 307).

b. Select how you want the **Availability zone(s)** selected for this cluster. You have two options.

   - **No preference** – ElastiCache will select the availability zone.
   - **Specify availability zones** – You specify the availability zone for your cluster.

   For this exercise, select **Specify availability zones** and then choose an availability zone from the list below **Primary**.

   For more information, see Selecting Regions and Availability Zones (p. 45).

c. From the **Security groups** list, choose the security groups that you want to use for this cluster. For this exercise, select *default*.

   For more information, see ElastiCache and Security Groups (p. 331).

d. If you are going to seed your cluster with data from a .RDB file, in the **Seed RDB file S3 location** box, enter the Amazon S3 location of the .RDB file.

   For more information, see Using a Backup to Seed a Cluster (p. 257).

e. Because this is not a production cluster, clear the **Enable automatic backups** check box.

   For more information on Redis backup and restore, see ElastiCache Backup & Restore (Redis) (p. 235).

f. The **Maintenance window** is the time, generally an hour, each week where ElastiCache schedules system maintenance on your cluster. You can allow ElastiCache to specify the day and time for your maintenance window (*No preference*), or you can specify the day and time yourself (*Specify maintenance window*. If you choose *Specify maintenance window*, specify the **Start day**, **Start time**, and **Duration** (in hours) for your maintenance window. For this exercise, select *No preference*.

   For more information, see Maintenance Window (p. 43).

g. For **Notifications**, leave it as *Disabled*.

7. Select **Create cluster** to launch your cluster, or **Cancel** to cancel the operation.

Step 3: (Optional) View Cluster Details

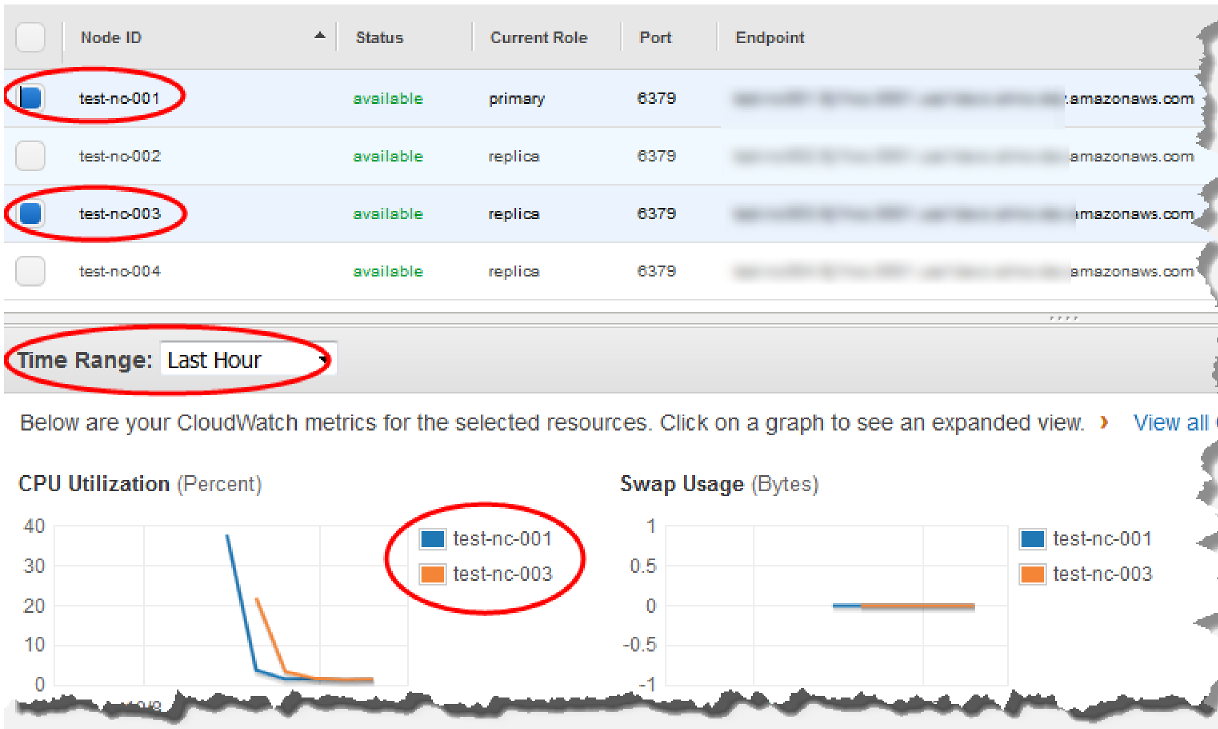
Before you continue, make sure you have completed Step 2: Launch a Cluster (p. 22).

To view a Redis (cluster mode disabled) 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. In the ElastiCache console dashboard, choose **Redis** to display a list of all your clusters that are running any version of Redis.

3. To see details of a cluster, select the check box to the left of the cluster's name. Make sure you select a cluster running the Redis engine, not Clustered Redis. This diplays details about the cluster, including the cluster's primary endpoint.
4. To view node information:

   a. Choose the cluster's name.
   b. Choose the **Nodes** tab. This diplays details about each node, including the node's endpoint which you need to use to read from the cluster.
   c. To view metrics on one or more nodes, select the box to the left of the node ID, then select the time range for the metrics from the **Time range** list. If you select multiple nodes, you can see overlay graphs.



Metrics over the last hour for two Redis nodes

Step 4: Authorize Access

This section assumes that you are familiar with launching and connecting to Amazon EC2 instances. For more information, go to the *Amazon EC2 Getting Started Guide*.

All ElastiCache clusters are designed to be accessed from an Amazon EC2 instance. A cluster and its related EC2 instance must be in the same Amazon Virtual Private Cloud (Amazon VPC). If you must access an ElastiCache cluster from somewhere other than an EC2 instance in the same VPC, as a workaround you can set up one or more EC2 hosts inside the cache's VPC to act as a proxy for the outside world. Setting up a host adds an extra network hop or extra Secure Sockets Layer (SSL) overhead and cost, or both. However, those costs are small for many use cases. You must grant the proxy EC2 instance access to your cluster. For information on accessing your ElastiCache resources from outside AWS, go to Accessing ElastiCache Resources from Outside AWS (p. 348).

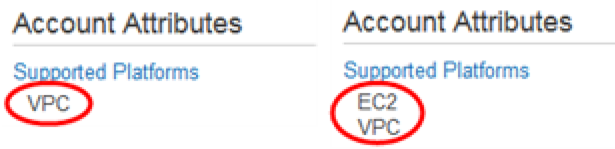
By default, network access to your cluster is limited to the user account that was used to launch it. Before you can connect to a cluster from an EC2 instance, you must authorize the EC2 instance to access the cluster. The steps required depend upon whether you launched your cluster into an Amazon VPC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

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

To determine whether you launched your cluster into EC2-VPC or EC2-Classic using the AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. Locate **Supported Platforms** in the upper-right corner.

   Under **Supported Platforms**, you will see either only **VPC** or both **EC2** and **VPC**.

   If you see only **VPC**, continue at You Launched Your Cluster into EC2-VPC (p. 25).

   If you see both **EC2** and **VPC**, continue at You Launched Your Cluster into EC2-Classic (p. 26).

For more information, see Detecting Your Supported Platforms and Whether You Have a Default VPC.

To determine whether you launched your cluster into EC2-VPC or EC2-Classic using the AWS Command Line Interface (AWS CLI)

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

   ```
   aws ec2 describe-account-attributes
   ```

   If you see only **VPC** in the output, continue at You Launched Your Cluster into EC2-VPC (p. 25).

   If you see both **EC2** and **VPC** in the output, continue at You Launched Your Cluster into EC2-Classic (p. 26).

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 left navigation pane, under **Network & Security**, select **Security Groups**.
3. In the list of security groups, select the security group for your Amazon VPC. If you are a new ElastiCache user, this security group will be named *default*.
4. Select **Inbound** tab, and then do the following:

   a. Select **Edit**.
   b. Select **Add rule**.
   c. In the **Type** column, select **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 ports are as follows:

   - Memcached: port 11211
   - Redis: port 6379

e. In the **Source** box, select **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.

f. Select **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 into 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. From the left navigation pane, select **Cache Security Groups**.

   A list of cache security groups appears.
3. Select the **default** security group.
4. From the list at the bottom of the screen, select the **EC2 Security Group Name** you want to authorize.
5. Select **Add** to authorize access.

   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 select **Remove**.

Step 5: Connect to a Cluster's Node

Before you continue, be sure you have completed Step 4: Authorize Access (p. 24).

This section assumes that you've created an Amazon EC2 instance and can connect to it. For instructions on how to do this, go to 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 4: Authorize Access (p. 24).

Step 5.1: Find your Node Endpoints

Once your cluster is in the `available` state and you've authorized access to it, you can log in to an Amazon EC2 instance and connect to a node in the cluster. To do so, you must first determine the node endpoint.

To find your node's endpoints, see the relevant topic. When you find the endpoint you need, copy it to your clipboard for use in Step 5.2.

- Finding Your ElastiCache Endpoints (p. 48)
- Finding the Endpoints for a Memcached Cluster (Console) (p. 49)
- Finding the Endpoints for a Redis (cluster mode disabled) Cluster (Console) (p. 51)
- Finding the Endpoints for a Redis (cluster mode enabled) Cluster (Console) (p. 51)
- Finding Endpoints (AWS CLI) (p. 54)
- Finding Endpoints (ElastiCache API) (p. 58)

Step 5.2: Connect to a Memcached Node

Now that you have an endpoint, you can log in to an Amazon EC2 instance and connect to the cache node. The procedure depends on the engine that you are using:

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

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. Run 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
>
```

Step 5.2: Connect to a Redis Cluster or Replication Group

Now that you have the endpoint you need, you can log in to an EC2 instance and connect to the cache node. The procedure depends on the engine that you are using:

In the following example, you use the *redis-cli* utility to connect to a cluster that is running Redis.

> **Note**
> For more information about Redis and available Redis commands, see Redis commands webpage.

To connect to a Redis cluster using *redis-cli*

1. Connect to your Amazon EC2 instance 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. Before you can build *redis-cli*, you will need to download and install the GNU Compiler Collection (*gcc*). At the command prompt of your EC2 instance, type the following command and type *y* at the confirmation prompt.

```
sudo yum install gcc

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

```
--> Running transaction check

...(output omitted)...

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

...(output omitted)...

Complete!
```

3. Download and compile the *redis-cli* utility. This utility is included in the Redis software distribution. At the command prompt of your EC2 instance, type the following commands:

   Note
   For Ubuntu systems, prior to running `make`, run `make distclean`.

```
wget http://download.redis.io/redis-stable.tar.gz
tar xvzf redis-stable.tar.gz
cd redis-stable
make
```

4. At the command prompt of your EC2 instance, type the following command, substituting the endpoint of your cluster for the one shown in this example.

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

   A Redis command prompt similar to the following appears.

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

5. Run Redis commands.

   You are now connected to the cluster and can run Redis commands. The following is an example.

```
set a "hello"          // Set key "a" with a string value and no
 expiration
OK
get a                  // Get value for key "a"
"hello"
get b                  // Get value for key "b" results in miss
(nil)
set b "Good-bye" EX 5  // Set key "b" with a string value and a 5 second
 expiration
get b
"Good-bye"
                       // wait 5 seconds
```

```
get b
(nil)                    // key has expired, nothing returned
quit                     // Exit from redis-cli
```

Step 6: Delete Your Cluster [Avoid Unnecessary Charges]

Before you continue, be sure you have completed at least as far as Step 2: Launch a Cluster (p. 22).

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

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, either Memcached or Redis.

   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:

   a. If this is a Redis cluster, specify whether or not a final snapshot should be taken, and, if you want a final snapshot, the name of the snapshot.
   b. 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.

Congratulations! You have successfully launched, authorized access to, connected to, viewed, and deleted a Redis 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
- The AWS Command Line Interface
- Amazon ElastiCache API Reference

If you haven't already read them, here are some ElastiCache topics you should become familiar with.

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

- Engines and Versions (p. 32)

  ElastiCache supports two engines–Memcached and Redis. This topic helps you determine which engine is best for your scenario.
- Selecting Your Node Size (p. 78)

  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. This topic assists you in selecting the best node size.
- Best Practices for Implementing Amazon ElastiCache (p. 60)

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

Engines and Versions

Amazon ElastiCache supports these cache engines: Memcached and Redis. Each engine provides some advantages. Use the information in this topic to help you select 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 (see Upgrading Engine Versions (p. 41)), 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 anew with the earlier engine version.

Topics

Selecting an Engine: Memcached, Redis (cluster mode disabled), or Redis (cluster mode enabled)

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

Select Memcached if the following apply to your situation:

- You need the simplest model possible.
- You need to run large nodes with multiple cores or threads.
- You need the ability to scale out/in, adding and removing nodes as demand on your system increases and decreases.
- You need to partition your data across multiple shards.
- You need to cache objects, such as a database.

Select Redis 2.8.x or Redis 3.2 (non-clustered mode) if the following apply to your situation:

- You need complex data types, such as strings, hashes, lists, sets, sorted sets, and bitmaps.
- You need to sort or rank in-memory data-sets.
- 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.

Select Redis 3.2 (clustered mode) if you require all the functionality of Redis 2.8.x with the following differences:

- You need to partition your data across 2 to 15 node groups. (Cluster mode only.)
- You need geospatial indexing. (clustered mode or non-clustered mode)
- You do not need to support multiple databases.

Important

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

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

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 & 3.2.x	3.2.x
Data types	Simple #	Redis 2.8.x - Complex * Redis 3.2.x - Complex #	Complex #
Multi-threaded	Yes	No	No
Cluster is modifiable	Yes	Yes	No
Node type upgrade	No	Yes	No
Engine upgrading	Yes	Yes	No
Data partitioning	Yes	No	Yes
Persistance of key store	No	Yes	Yes
High availability (Replication)	No	Yes	Yes
Automatic Failover of Primary	No	Optional	Required
Pub/Sub capabilities	No	Yes	Yes
Sorted lists	No	Yes	Yes
Counters & hashes	No	Yes	Yes
Backup/Restore capabilities	No	Yes	Yes
Geospatial indexing	No	Redis 2.8.x - No Redis 3.2.x - Yes	Yes
Notes:			
# string, objects (like databases)			
* string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog			
# string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog, geospacial indexes			

After you select the engine for your cluster, we recommend that you use the most recent version of that engine. For more information see Comparing Memcached Versions (p. 37) or Comparing Redis Versions (p. 38).

Determine Available Engine Versions

Not all versions of an engine are available in every region. Therefore, before you create a cluster or replication group, you should determine which engine versions are supported in your region.

You can determine which engine versions are supported in a region using the ElastiCache console, the AWS CLI, or the ElastiCache API.

Determine Available Engine Versions (Console)

When creating a cluster or replication group you are asked to select an engine version from a list. The engine versions in the list are those available in the current region.

For more information, see Creating a Cluster (p. 128) or Creating a Replication Group from Scratch (p. 209).

Determine Available Engine Versions (AWS CLI)

To determine which engine versions are available in a region, use the `describe-cache-engine-versions` operation.

```
aws elasticache describe-cache-engine-versions
```

The output of this operation should looks something like this (JSON format).

```
{
    "CacheEngineVersions": [
        {
            "Engine": "memcached",
            "CacheEngineDescription": "memcached",
            "CacheEngineVersionDescription": "memcached version 1.4.14",
            "CacheParameterGroupFamily": "memcached1.4",
            "EngineVersion": "1.4.14"
        },
        ... some output omitted for brevity
        {
            "Engine": "redis",
            "CacheEngineDescription": "Redis",
            "CacheEngineVersionDescription": "redis version 2.8.6",
            "CacheParameterGroupFamily": "redis2.8",
            "EngineVersion": "2.8.6"
        }
    ]
}
```

For more information, see describe-cache-engine-versions.

Determine Available Engine Versions (ElastiCache API)

To determine which engine versions are available in a region, use the `DescribeCacheEngineVersions` action.

```
https://elasticache.us-west-2.amazonaws.com/
```

```
?Action=DescribeCacheEngineVersions
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

For more information, see DescribeCacheEngineVersions.

Comparing Memcached Versions

ElastiCache supports these versions of Memcached:

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. 286).
- 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.

Comparing Redis Versions

ElastiCache supports these Redis versions.

Topics

Note

Because the newer Redis versions provide a better and more stable user experience, Redis versions 2.6.13, 2.8.6, and 2.8.19 are deprecated from the ElastiCache console. While we recommend against it, if you must use one of these older Redis versions, you can use the AWS CLI or ElastiCache API.

For more information see the following topics:

	AWS AWS CLI	ElastiCache API
Create Cluster	Creating a Cache Cluster (AWS CLI) (p. 137) #	Creating a Cache Cluster (ElastiCache API) (p. 140) #
Modify Cluster	Modifying a Cache Cluster (AWS CLI) (p. 150) #	Modifying a Cache Cluster (ElastiCache API) (p. 150) #
Create Replication Group	Creating a Redis (cluster mode disabled) Replication Group from Scratch (AWS CLI) (p. 210) Creating a Redis (cluster mode enabled) Replication Group from Scratch (AWS CLI) (p. 215)	Creating a Redis (cluster mode disabled) Replication Group from Scratch (ElastiCache API) (p. 212) Creating a Redis (cluster mode enabled) Replication Group from Scratch (ElastiCache API) (p. 218)
Modify Replication Group	Modifying a Replication Group (AWS CLI) (p. 226) #	Modifying a Replication Group (ElastiCache API) (p. 226) #
# Cannot be used to create a Redis (cluster mode enabled) cluster or replication group.		
# Cannot be used to modify a Redis (cluster mode enabled) cluster or replication group.		

Redis Version 3.2.4 (Enhanced)

Redis version 3.2.4 introduces the next major version of the Redis engine supported by Amazon ElastiCache. Redis 3.2.4 users have all the functionality of earlier Redis versions available to them plus the option to run in *cluster mode* or *non-cluster mode*. The following table summarizes .

Comparing Redis 3.2.4 Cluster Mode and Non-Cluster Mode

	Redis 3.2.4	
Data partitioning	No	Yes
Geospatial indexing	Yes	Yes
Change node type	Yes	No
Replica scaling	Yes	No
Database support	Multiple	Single

Notes:

- **Partitioning** – the ability to split your data across 2 to 15 node groups (shards) with replication support for each node group.
- **Geospatial indexing** – Redis 3.2 introduces support for geospatial indexing via six GEO commands. For more information, see the Redis GEO* command documentation Redis Commands: GEO on the Redis Commands page (filtered for GEO).

For information about additional Redis 3 features, see Redis 3.2 release notes and Redis 3.0 release notes.

Currently ElastiCache managed Redis (cluster mode enabled) does not support the following Redis 3.2 features:

- Replica migration
- Cluster rebalancing
- Lua debugger

ElastiCache disables the following Redis 3.2 management commands:

- `cluster meet`
- `cluster replicate`
- `cluster flushslots`
- `cluster addslots`
- `cluster delslots`
- `cluster setslot`
- `cluster saveconfig`
- `cluster forget`
- `cluster failover`
- `cluster bumpepoch`
- `cluster set-config-epoch`

- `cluster reset`

For information about Redis 3.2.4 parameters, see Redis 3.2.4 Parameter Changes (p. 293).

Redis Version 2.8.24 (Enhanced)

Redis improvements added since version 2.8.23 include bug fixes and logging of bad memory access addresses. For more information, see Redis 2.8 release notes.

Redis Version 2.8.23 (Enhanced)

Redis improvements added since version 2.8.22 include bug fixes. For more information, see Redis 2.8 release notes. This release also includes support for the new parameter *`close-on-slave-write`* which, if enabled, disconnects clients who attempt to write to a read-only replica.

For more information on Redis 2.8.23 parameters, see Redis 2.8.23 (Enhanced) Added Parameters (p. 295) in the ElastiCache User Guide.

Redis Version 2.8.22 (Enhanced)

Redis improvements added since version 2.8.21 include the following:

- Support for forkless backups and synchronizations which allows you to allocate less memory for backup overhead and more for your application. For more information see How Synchronization and Backup are Implemented (p. 203). The forkless process can impact both latency and throughput. In the case of high write throughput, when a replica re-syncs, it may be unreachable for the entire time it is syncing.
- In the event of a failover, replication groups now recover faster as replicas will perform partial syncs with the primary rather than full syncs whenever possible. Additionally, both the primary and replicas no longer use the disk during syncs, providing further speed gains.
- Support for two new CloudWatch metrics.
  - `ReplicationBytes` – The number of bytes a replication group's primary cluster is sending to the read replicas.
  - `SaveInProgress` – A binary value that indicates whether or not there is a background save process running.

  For more information, see Metrics for Redis (p. 356).
- A number of critical bug fixes in replication PSYNC behavior. For more information, see Redis 2.8 release notes.
- To maintain enhanced replication performance in Multi-AZ replication groups and for increased cluster stability, non-ElastiCache replicas are no longer supported.
- To improve data consistency between the primary cluster and replicas in a replication group, the replicas will no longer evict keys independent of the primary cluster.
- Redis configuration variables *`appendonly`* and *`appendfsync`* are not supported on Redis version 2.8.22 and later.
- In low-memory situations, clients with a large output buffer may be disconnected from a replica cluster. If disconnected, the client will need to reconnect. Such situations are most likely to occur for PUBSUB clients.

Redis Version 2.8.21

Redis improvements added since version 2.8.19 include a number of bug fixes. For more information, see Redis 2.8 release notes.

Redis Version 2.8.19

Redis improvements added since version 2.8.6 include the following:

- Support for HyperLogLog. For more information, go to Redis new data structure: HyperLogLog.
- The sorted set data type has now support for lexicographic range queries with the new commands `ZRANGEBYLEX`, `ZLEXCOUNT`, and `ZREMRANGEBYLEX`.
- To prevent a primary node from sending stale data to replica nodes, the master SYNC fails if a background save (`bgsave`) child process is aborted.
- Support for the *HyperLogLogBasedCommands* CloudWatch metric. For more information, see Metrics for Redis (p. 356).

Redis Version 2.8.6

Redis improvements added since version 2.6.13 include the following:

- Improved resiliency and fault tolerance for read replicas.
- Support for partial resynchronization.
- Support for user-defined minimum number of read replicas that must be available at all times.
- Full support for pub/sub—notifying clients of events on the server.
- Automatic detection of a primary node failure and failover of your primary node to a secondary node.

Redis Version 2.6.13

Redis version 2.6.13 was the initial version of Redis supported by Amazon ElastiCache. Multi-AZ with automatic failover is not supported on Redis 2.6.13.

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 Memcached or Redis 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 will not occur automatically and must be initiated by you.

You initiate version upgrades to your cluster or replication group by modifying it and specifying a new engine version. For more information, see Modifying an ElastiCache Cluster (p. 149) or Modifying a Cluster with Replicas (p. 226).

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.
- Redis (cluster mode enabled) does not support changing engine versions.

- ElastiCache does not support switching between cluster enabled and cluster disabled.

Important Notes on Memcached Engine Upgrades

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.

Important Notes on Redis Engine Upgrades

The Amazon ElastiCache engine upgrade process is designed to make a best effort to retain your existing data and requires successful Redis replication.

> **Important**
> If you want to upgrade your engine from Redis 2.x to Redis 3.x you can do so, but you cannot upgrade from Redis (cluster mode disabled) to Redis (cluster mode enabled). To upgrade to Redis (cluster mode enabled), you must create a new Redis (cluster mode enabled) cluster which you can seed using a Redis (cluster mode disabled) snapshot as long as both the old and new clusters have the same number of shards (API/CLI: node groups).

- For single Redis clusters and clusters with Multi-AZ disabled, we recommend that sufficient memory be made available to Redis as described in Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60). Please note that in these cases, the primary will be unavailable to service requests during the upgrade process.

- For Redis clusters with Multi-AZ enabled, in addition to the preceeding, we also recommend scheduling engine upgrades during periods of low incoming write traffic. The primary will continue to be available to service requests during the upgrade process, except for a few minutes when a failover is initiated.

Blocked Redis Engine Upgrades

As shown in the following table, your Redis engine upgrade operation is blocked if you have a pending scale up operation.

Pending Operations	Blocked Operations
Scale up	Immediate engine upgrade
Engine upgrade	Immediate scale up
Scale up and engine upgrade	Immediate scale up
Scale up and engine upgrade	Immediate engine upgrade

To resolve a blocked engine upgrade, do one of the following

- Schedule your Redis engine upgrade operation for the next maintenance window by clearing the **Apply immediately** check box (CLI use: `--no-apply-immediately`, API use: `ApplyImmediately=false`).

- Wait until your next maintenance window (or after) to perform your Redis engine upgrade operation.

- Add the Redis scale up operation to this cluster modification with the **Apply Immediately** check box selected (CLI use: `--apply-immediately`, API use: `ApplyImmediately=true`). (This effectively cancels the engine upgrade during the next maintenance window by performing it immediately.)

How to Upgrade Engine Versions

You initiate version upgrades to your cluster or replication group 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.

Important
Remember, for Redis (cluster mode enabled) you cannot modify clusters or replication groups.

	Clusters	Replication Groups
Using the console	Modifying a Cluster (Console) (p. 149)	Modifying a Redis Cluster (Console) (p. 226)
Using the AWS CLI	Modifying a Cache Cluster (AWS CLI) (p. 150)	Modifying a Replication Group (AWS CLI) (p. 226)
Using the ElastiCache API	Modifying a Cache Cluster (ElastiCache API) (p. 150)	Modifying a Replication Group (ElastiCache API) (p. 226)

Maintenance Window

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 cache cluster, ElastiCache assigns a 60-minute maintenance window within your region's maintenance window on a randomly selected day of the week.

The 60-minute maintenance window is selected 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 select 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-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

Region Code	Region Name	Region Maintenance Window
eu-central-1	EU (Frankfurt) Region	23:00–07:00 UTC
eu-west-1	EU (Ireland) Region	22:00–06:00 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-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

The maintenance window should fall at the time of lowest usage and thus might need modification from time to time. You can 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 have requested occur during this time.

For more information about how to adjust the preferred maintenance window for your cache clusters, see Modifying an ElastiCache Cluster (p. 149) or Modifying a Cluster with Replicas (p. 226).

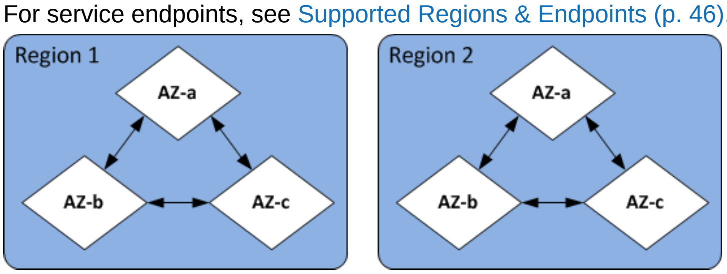
Selecting Regions and Availability Zones

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. 46).



Regions and Availability Zones

Topics

Locating Your Redis Read Replicas and Memcached Nodes

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

By locating the clusters or nodes in different Availability Zones, you eliminate the chance that a failure, such as a power outage, in one Availability Zone will cause your entire system to fail. Testing has demonstrated that there is no significant latency difference between locating all nodes in one Availability Zone or spreading them across multiple Availability Zones.

To specify an Availability Zone for your Memcached nodes, create a Memcached cluster as you normally do. On the **Cluster Details** page of the Launch Cluster wizard, use the **Preferred Zone** list to specify an Availability Zone for this node.

To specify an Availability Zone for your Redis read replica, you first create a replication group and then add from one to five read replicas to the replication group. You can specify a different Availability Zone for each read replica. For more information on creating a Redis read replica in an Availability Zone different from the primary Redis cache cluster, see Creating a Redis Cluster with Replicas (p. 204) and Adding a Read Replica to a Redis Cluster (p. 229).

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, go to Selecting Regions and Availability Zones (p. 45) at the top of this topic.

Regions where ElastiCache is supported

Region Name	Region	Endpoint	Protocol
US East (N. Virginia) Region	`us-east-1`	`elasticache.us-east-1.amazonaws.com`	HTTPS
US East (Ohio) Region	`us-east-2`	`elasticache.us-east-2.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 Only T2 and M4 node types are currently supported in this region.	ca-central-1	`elasticache.ca-central-1.amazonaws.com`	HTTPS
Asia Pacific (Mumbai) Region	`Asia Pacific (Mumbai)`	`elasticache.ap-south-1.amazonaws.com`	HTTPS

Region Name	Region	Endpoint	Protocol
Only T2, R3, and M4 node types are currently supported in this region.			
Asia Pacific (Seoul) Region	`ap-northeast-2`	`elasticache.ap-northeast-2.amazonaws.com`	HTTPS
Asia Pacific (Singapore) Region	`ap-southeast-1`	`elasticache.ap-southeast-1.amazonaws.com`	HTTPS
Asia Pacific (Sydney) Region	`ap-southeast-2`	`elasticache.ap-southeast-2.amazonaws.com`	HTTPS
Asia Pacific (Tokyo) Region	`ap-northeast-1`	`elasticache.ap-northeast-1.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
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
AWS GovCloud (US) M4 node types are currently not supported in this region. For information on using the AWS GovCloud (US) with ElastiCache, see Services in the AWS GovCloud (US) region: ElastiCache.	`us-gov-west-1`	`elasticache.us-gov-west-1.amazonaws.com`	HTTPS

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

Finding Your ElastiCache 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.

- **Redis standalone node**, use the node's endpoint for both read and write operations.

- **Redis (cluster mode disabled) clusters**, use the *Primary Endpoint* for all write operations. Use the individual *Node Endpoints* for read operations (In the API/CLI these are referred to as Read Endpoints).

- **Redis (cluster mode enabled) clusters**, use the cluster's *Configuration Endpoint* for all operations. You must use a client that supports Redis Cluster (Redis 3.2). You can still read from individual node endpoints (In the API/CLI these are referred to as Read Endpoints).

The following sections guide you through discovering the endpoints you'll need for the engine you're running.

Finding the Endpoints for a Memcached Cluster (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 Node Auto Discovery (Memcached) (p. 95).

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

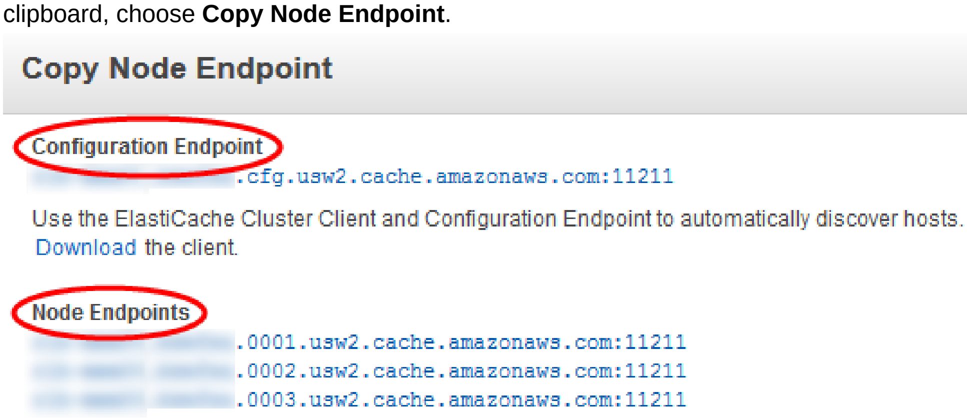
To find and copy the endpoints for a Memcached cluster (console)

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. From the left 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 select **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 the Endpoints for a Redis (cluster mode disabled) Cluster (Console)

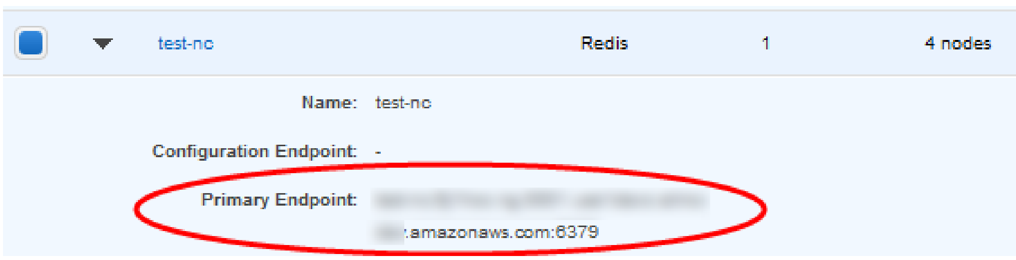
If a Redis (cluster mode disabled) cluster has only one node, the node's endpoint is used for both reads and writes If a Redis (cluster mode disabled) cluster has multiple nodes, there are two types of endpoints, the Primary endpoint which always points to whichever node is serving as Primary, and the node endpoints. The Primary endpoint is used for writes. The node endpoints are used for reads.

To find a Redis (cluster mode disabled) cluster's endpoints

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

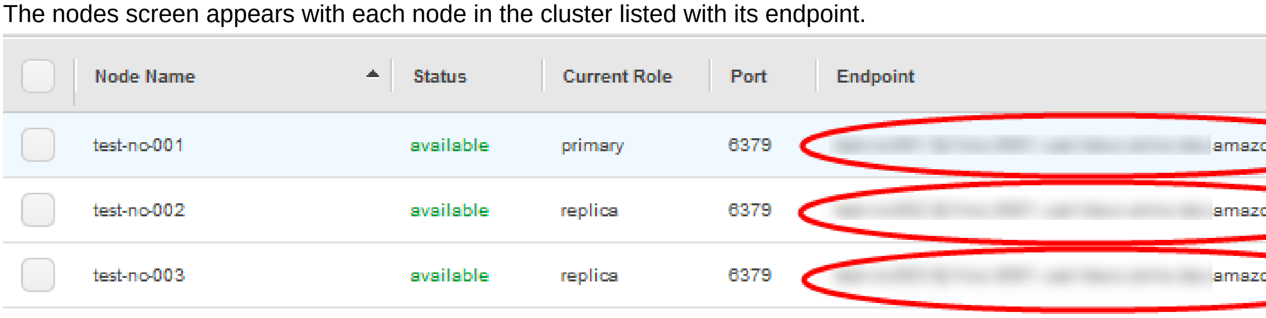
   The clusters screen will appear with a list of Redis (cluster mode disabled) and Redis (cluster mode enabled) clusters.
3. To find the cluster's Primary endpoint, choose the box to the left of cluster's name.

   If there is only one node in the cluster, there is no primary endpoint and you can continue at the next step.

   Primary endpoint for a Redis (cluster mode disabled) cluster
4. To find the node endpoints for the cluster, choose the cluster's name.

   The nodes screen appears with each node in the cluster listed with its endpoint.

   Node endpoints for a Redis (cluster mode disabled) cluster
5. To copy the endpoint to your clipboard:

   a. Find the endpoint (only one at a time) you want to copy and highlight it.
   b. Right-click the highlighted endpoint, and then select **Copy** from the context menu.

   The highlighted endpoint is now copied to your clipboard.

A Redis endpoint looks something like
`*clusterName.xxxxxx*.0001.usw2.cache.amazonaws.com *port*`.

Finding the Endpoints for a Redis (cluster mode enabled) Cluster (Console)

Use the *Configuration Endpoint* for both read and write operations. Redis determines which of the cluster's node to access.

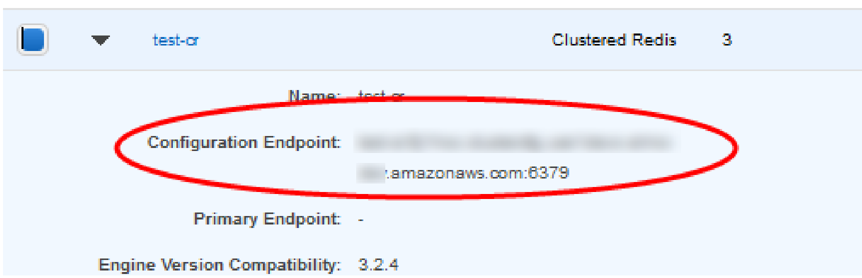
The following procedure demonstrates how to find and copy Redis (cluster mode enabled) cluster endpoints.

To find the configuration endpoint for a Redis (cluster mode enabled) cluster

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. From the left navigation pane, select **Redis**.

   A list of clusters running any version of Redis appears.
3. From the list of clusters, choose the box to the left of a cluster running "Clustered Redis".

   The screen expands showing details about the selected cluster.
4. Locate the *Configuration endpoint*.



Configuration endpoint for a Redis (cluster mode enabled) cluster

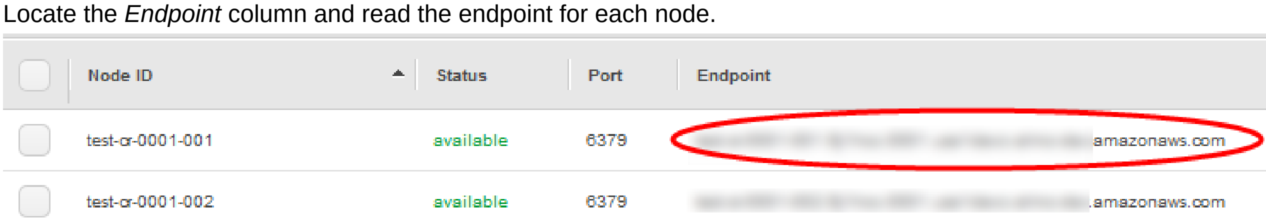
To find the node endpoints for a Redis (cluster mode enabled) cluster

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. From the left navigation pane, select **Redis**.

   A list of clusters running any version of Redis appears.
3. From the list of clusters, choose the cluster name of a cluster running "Clustered Redis".

   The shards page opens.
4. Choose the name of the shard you want node endpoint for.

   A list of the shard's nodes appears with each node's endpoint.
5. Locate the *Endpoint* column and read the endpoint for each node.



Node endpoints for a Redis (cluster mode enabled) cluster

To copy an endpoint to your clipboard

1. Find the endpoint you want to copy using one of the preceeding procedures.
2. Highlight the endpoint that you want to copy.
3. Right-click the highlighted endpoint and select **Copy** from the context menu.

   The highlighted endpoint is now copied to your clipboard.

Finding Endpoints (AWS CLI)

You can use the AWS CLI for Amazon ElastiCache to discover the endpoints for nodes, clusters, and replication groups

Topics

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 Redis clusters, the command returns the cluster endpoint. 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.

The following command retrieves the configuration endpoint (`ConfigurationEndpoint`) and individual node endpoints (`Endpoint`) for the Memcached cluster *mycluster*.

For Linux, OS X, 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, go to the topic describe-cache-clusters.

Finding the Endpoints for Replication Groups (AWS CLI)

You can use the AWS CLI to discover the endpoints for a replication group and its clusters with the `describe-replication-groups` command. The command returns the replication group's primary endpoint and a list of all the clusters in the replication group with their endpoints.

The following operation retrieves the primary endpoint (PrimaryEndpoint) and individual node endpoints (ReadEndpoint) for the replication group `myreplgroup`. Use the primary endpoint for all write operations and the individual node endpoints for all read operations.

For Linux, OS X, or Unix:

```
aws elasticache describe-replication-groups \
    --replication-group-id myreplgroup
```

For Windows:

```
aws elasticache describe-replication-groups ^
    --replication-group-id myreplgroup
```

Output from this operation should look something like this (JSON format).

```
{
   "ReplicationGroups": [
     {
       "Status": "available",
       "Description": "test",
       "NodeGroups": [
         {
            "Status": "available",
               "NodeGroupMembers": [
                  {
                     "CurrentRole": "primary",
                     "PreferredAvailabilityZone": "us-west-2a",
                     "CacheNodeId": "0001",
                     "ReadEndpoint": {
                        "Port": 6379,
                        "Address":
 "myreplgroup-001.1abc4d.0001.usw2.cache.amazonaws.com"
                     },
                     "CacheClusterId": "myreplgroup-001"
                  },
                  {
                     "CurrentRole": "replica",
                     "PreferredAvailabilityZone": "us-west-2b",
                     "CacheNodeId": "0001",
                     "ReadEndpoint": {
                        "Port": 6379,
                        "Address":
 "myreplgroup-002.1abc4d.0001.usw2.cache.amazonaws.com"
                     },
                     "CacheClusterId": "myreplgroup-002"
                  },
                  {
```

```
                     "CurrentRole": "replica",
                     "PreferredAvailabilityZone": "us-west-2c",
                     "CacheNodeId": "0001",
                     "ReadEndpoint": {
                        "Port": 6379,
                        "Address":
 "myreplgroup-003.1abc4d.0001.usw2.cache.amazonaws.com"
                     },
                     "CacheClusterId": "myreplgroup-003"
                  }
               ],
               "NodeGroupId": "0001",
               "PrimaryEndpoint": {
                  "Port": 6379,
                  "Address":
 "myreplgroup.1abc4d.ng.0001.usw2.cache.amazonaws.com"
               }
            }
         ],
         "ReplicationGroupId": "myreplgroup",
         "AutomaticFailover": "enabled",
         "SnapshottingClusterId": "myreplgroup-002",
         "MemberClusters": [
            "myreplgroup-001",
            "myreplgroup-002",
            "myreplgroup-003"
         ],
         "PendingModifiedValues": {}
      }
   ]
}
```


For more information, see describe-replication-groups in the *AWS Command Line Interface Reference*.

Finding Endpoints (ElastiCache API)

You can use the Amazon ElastiCache API to discover the endpoints for nodes, clusters, and replication groups

Topics

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 Redis clusters, the action returns the cluster endpoint. For Memcached clusters, the action returns the configuration endpoint. If you include the optional parameter *`ShowCacheNodeInfo`*, the action also returns the endpoints of the individual nodes in the cluster.

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.

Finding Endpoints for Replication Groups (ElastiCache API)

You can use the ElastiCache API to discover the endpoints for a replication group and its clusters with the `DescribeReplicationGroups` action. The action returns the replication group's primary endpoint and a list of all the clusters in the replication group with their endpoints.

The following operation retrieves the primary endpoint (PrimaryEndpoint) and individual node endpoints (ReadEndpoint) for the replication group `myreplgroup`. Use the primary endpoint for all write operations and the individual node endpoints for all read operations.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeReplicationGroups
    &ReplicationGroupId=myreplgroup
    &SignatureVersion=4
```

```
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&Version=2015-02-02
&X-Amz-Credential=<credential>
```

For more information, see DescribeReplicationGroups.

Best Practices for Implementing Amazon ElastiCache

This topic identifies best practices for implementing Amazon ElastiCache.

Topics

Ensuring You Have Sufficient Memory to Create a Redis Snapshot

Redis snapshots and synchronizations in version 2.8.22 and later

Redis 2.8.22 introduces a forkless save process that allows you to allocate more of your memory to your application's use without incurring increased swap usage during synchronizations and saves. For more information, see How Synchronization and Backup are Implemented (p. 203).

Redis snapshots and synchronizations prior to version 2.8.22

When you work with Redis ElastiCache, Redis calls a background write command in a number of cases:

- When creating a snapshot for a backup.
- When synchronizing replicas with the primary in a replication group.
- When enabling the append-only file feature (AOF) for Redis.
- When promoting a replica to master (which causes a primary/replica sync).

Whenever Redis executes a background write process, you must have sufficient available memory to accommodate the process overhead. Failure to have sufficient memory available will cause the process to fail. Because of this, it is important to select a node instance type that has sufficient memory when creating your Redis cluster.

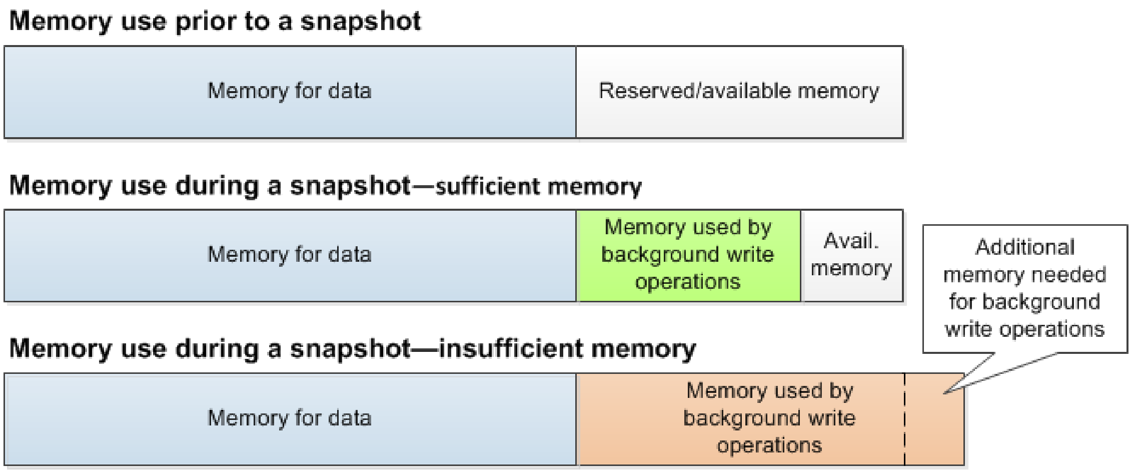
Background Write Process and Memory Usage

Whenever a background write process is called, Redis forks its process (remember, Redis is single threaded). One fork persists your data to disk in a Redis .rdb snapshot file. The other fork services all read and write operations. In order to ensure that your snapshot is a point-in-time snapshot, all data updates and additions are written to an area of available memory separate from the data area.

As long as you have sufficient memory available to record all write operations while the data is being persisted to disk, you will have no insufficient memory issues. You are likely to experience insufficient memory issues if any of the following are true:

- Your application performs many write operations, thus requiring a large amount of available memory to accept the new or updated data.
- You have very little memory available in which to write new or updated data.
- You have a large dataset that takes a long time to persist to disk, thus requiring a large number of write operations.

The following diagram illustrates memory use when executing a background write process.



For information on the impact of doing a backup on performance, see Performance Impact of Backups (p. 236).

For more information on how Redis performs snapshots, see http://redis.io.

For more information on regions and availability zones, see Selecting Regions and Availability Zones (p. 45).

Avoiding Running Out of Memory When Executing a Background Write

Whenever a background write process such as BGSAVE or BGREWRITEAOF is called, to keep the process from failing, you must have more memory available than will be consumed by write operations during the process. The worst case scenario is that during the background write operation every Redis record is updated and some new records are added to the cache. Because of this, we recommend that you set *reserved-memory* to at least half of the value of *maxmemory*. For *maxmemory* values by node type, see Redis Node-Type Specific Parameters (p. 305).

The *maxmemory* value indicates the memory available to you for data and operational overhead. Because you cannot modify the *reserved-memory* parameter in the default parameter group, you must create a custom parameter group for the cluster. The default value for `reserved-memory` is 0, which allows Redis to consume all of *maxmemory* with data, potentially leaving too little memory for other uses, such as a background write process. For `maxmemory` values by node instance type, see Redis Node-Type Specific Parameters (p. 305).

You can also use `reserved-memory` parameter to reduce the amount of memory Redis uses on the box.

For more information on Redis-specific parameters in ElastiCache, see Redis Specific Parameters (p. 292).

For information on creating and modifying parameter groups, see Creating a Parameter Group (p. 273) and Modifying a Parameter Group (p. 282).

Mitigating Out-of-Disk-Space Issues When Using Redis AOF

When planning your Amazon ElastiCache implementation, you should plan so that failures have the least impact possible.

You enable AOF because an AOF file is useful in recovery scenarios. In case of a node restart or service crash, Redis will replay the updates from an AOF file, thereby recovering the data lost due to the restart or crash.

> **Warning**
> AOF cannot protect against all failure scenarios. For example, if a node fails due to a hardware fault in an underlying physical server, ElastiCache will provision a new node on a different server. In this case, the AOF file will no longer be available and cannot be used to recover the data. Thus, Redis will restart with a cold cache.

Enabling Redis Multi-AZ as a Better Approach to Fault Tolerance

If you are enabling AOF to protect against data loss, consider using a replication group with Multi-AZ enabled instead of AOF. When using a Redis replication group, if a replica fails, it is automatically replaced and synchronized with the primary cluster. If Multi-AZ is enabled on a Redis replication group and the primary fails, it fails over to a read replica. Generally, this functionality is much faster than rebuilding the primary from an AOF file. For greater reliability and faster recovery, we recommend that you create a replication group with one or more read replicas in different availability zones and enable Multi-AZ instead of using AOF. Because there is no need for AOF in this scenario, ElastiCache disables AOF on Multi-AZ replication groups.

For more information, see the following topics:

- Mitigating Failures (p. 62)
- ElastiCache Replication (Redis) (p. 190)
- Replication: Multi-AZ with Automatic Failover (Redis) (p. 196)

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

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 a 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, go to Creating a Cluster (Console): Memcached (p. 129).

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 only 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 cache 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 and come to your own conclusion.

For information on specifying the number of nodes in a Memcached cluster, go to Creating a Cluster (Console): Memcached (p. 129).

For more information on regions and availability zones, go to Selecting Regions and Availability Zones (p. 45).

Mitigating Failures when Running Redis

When running the Redis engine, you have the following options for minimizing the impact of a cluster or availability zone failure.

Mitigating Cluster Failures

To mitigate the impact of Redis cluster failures, you have the following options:

Topics

Mitigating Cluster Failures: Redis Append Only Files (AOF)

When AOF is enabled for Redis, whenever data is written to your Redis cluster, a corresponding transaction record is written to a Redis append only file (AOF). If your Redis process restarts, ElastiCache creates a replacement cluster and provisions it. You can then run the AOF against the cluster to repopulate it with data.

Some of the shortcomings of using Redis AOF to mitigate cluster failures are:

- It is time consuming.

  Creating and provisioning a cluster can take several minutes. Depending upon the size of the AOF, running it against the cluster will add even more time during which your application cannot access your cluster for data, forcing it to hit the database directly.

- The AOF can get big.

  Because every write to your cluster is written to a transaction record, AOFs can become very large, larger than the .rdb file for the dataset in question. Because ElastiCache relies on the local instance store, which is limited in size, enabling AOF can cause out-of-disk-space issues. You can avoid out-of-disk-space issues by using a replication group with Multi-AZ enabled.

- Using AOF cannot protect you from all failure scenarios.

  For example, if a cluster fails due to a hardware fault in an underlying physical server, ElastiCache will provision a new cluster on a different server. In this case, the AOF is not available and cannot be used to recover the data, leaving Redis to start with a cold cache.

For more information, see Redis Append Only Files (AOF) (p. 262).

Mitigating Cluster Failures: Redis Replication Groups

A Redis replication group is comprised of a single primary cluster which your application can both read from and write to, and from 1 to 5 read-only replica clusters. Whenever data is written to the primary cluster it is also asynchronously updated on the read replica clusters.

When a read replica fails

1. ElastiCache detects the failed read replica.
2. ElastiCache takes the failed cluster off line.

3. ElastiCache launches and provisions a replacement cluster in the same AZ.
4. The new cluster synchronizes with the Primary cluster.

During this time your application can continue reading and writing using the other clusters.

Redis Multi-AZ with Automatic Failover

You can enable Multi-AZ with automatic failover on your Redis replication groups. Whether you enable Multi-AZ with auto failover or not, a failed Primary will be detected and replaced automatically. How this takes place varies whether or not Multi-AZ is or is not enabled.

When Multi-AZ with auto failover is enabled

1. ElastiCache detects the Primary failure.
2. ElastiCache promotes the read replica with the least replication lag to primary.
3. The other replicas sync with the new primary.
4. ElastiCache spins up a read replica in the failed primary's AZ.
5. The new cluster syncs with the newly promoted primary.

Failing over to a replica cluster is generally faster than creating and provisioning a new cluster. This means your application can resume writing to your cluster sooner than if Multi-AZ were not enabled.

For more information, see Replication: Multi-AZ with Automatic Failover (Redis) (p. 196).

When Multi-AZ with auto failover is disabled

1. ElastiCache detects Primary failure.
2. ElastiCache promotes a random read replica to primary.
3. The other replicas sync with the new primary.
4. ElastiCache spins up a read replica in the failed primary's AZ.
5. The new cluster syncs with the newly promoted primary.
6. The new cluster is promoted to Primary. This keeps the Primary in the same AZ as before the failure.

Reads from the primary could fail just before and during a failover since the primary is dead. Failover relies on DNS which may take some time to update. During this time your application cannot write to the primary cluster. However, your application can continue reading from your replica clusters.

For added protection, we recommend that you launch the clusters in your replication group in different availability zones (AZs). If you do this, an AZ failure will only impact the clusters in that AZ and not the others.

For more information, see ElastiCache Replication (Redis) (p. 190).

Mitigating Availability Zone Failures

To mitigate the impact of an availability zone failure, locate your clusters in as many availability zones as possible.

No matter how many clusters you have, if they are all located in the same availability zone, a catastrophic failure of that AZ results in your losing all your cache data. However, if you locate your clusters in multiple AZs, a failure of any AZ results in your losing only the clusters in that AZ.

Any time you lose a cluster you can experience a performance degradation since read operations are now shared by fewer clusters. This performance degradation will continue until the clusters are replaced. Because your data is not partitioned across Redis clusters, you risk some data loss only when the primary cluster is lost.

For information on specifying the availability zones for Redis clusters, go to Creating a Redis (cluster mode disabled) Cluster (Console) (p. 130).

For more information on regions and availability zones, go to Selecting Regions and Availability Zones (p. 45).

Recommendations

There are two types of failures you need to plan for, individual node or cluster failures and broad availability zone failures. The best failure mitigation plan will address both kinds of failures.

Minimizing the Impact of Node and Cluster Failures

To minimize the impact of a node or cluster failure, we recommend that your implementation use multiple nodes or clusters.

If you're running Memcached and partitioning your data across nodes, the more nodes you use the smaller the data loss if any one node fails.

If you're running Redis, we also recommend that you enable Multi-AZ on your replication group so that ElastiCache will automatically fail over to a replica if the primary cluster fails.

Minimizing the Impact of Availability Zone Failures

To minimize the impact of an availability zone failure, we recommend launching your nodes or clusters in as many different availability zones as are available. Spreading your nodes or clusters evenly across AZs will minimize the impact in the unlikely event of an AZ failure.

Other precautions

If you're running Redis, then in addition to the above, we recommend that you schedule regular backups of your cluster. Backups (snapshots) create a .rdb file you can use to restore your cluster in case of failure or corruption. For more information, see ElastiCache Backup & Restore (Redis) (p. 235).

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* - `hash(key) mod n`. The resulting value (0 through *n*–1) is the number of the node where you place the object.

This approach is simple and works well as long as the number of nodes (*n*) is constant. However, whenever you add or remove a node from the cluster, the number of keys that need to be moved is *(n - 1) / n* (where *n* is the new number of nodes). Thus, this approach will result 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, go to 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 preceeding code, we recommend that you also turn *`memcached.sess_consistent_hash`* on in your php.ini file.

For more information, go to 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, go to the `memcached/locator` documentation at https://github.com/enyim/EnyimMemcached/wiki/MemcachedClient-Configuration#user-content-memcachedlocator.

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 or Redis. For descriptions of error messages from sources other than ElastiCache, see the documentation from the source that is generating the error message.

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: **The maximum number of manual snapshots for this cluster taken within 24 hours has been reached** or **The maximum number of manual snapshots for this node taken within 24 hours has been reached its quota of %n**

Cause: You attempted to take a manual snapshot of a cluster when you have already taken the maximum number of manual snapshots allowed in a 24-hour period.

Solution: Wait 24 hours to attempt another manual snapshot of the cluster. Or if you need to take a manual snapshot now, take the snapshot of another cluster that has the same data, such as a different cluster in a replication group.

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

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.

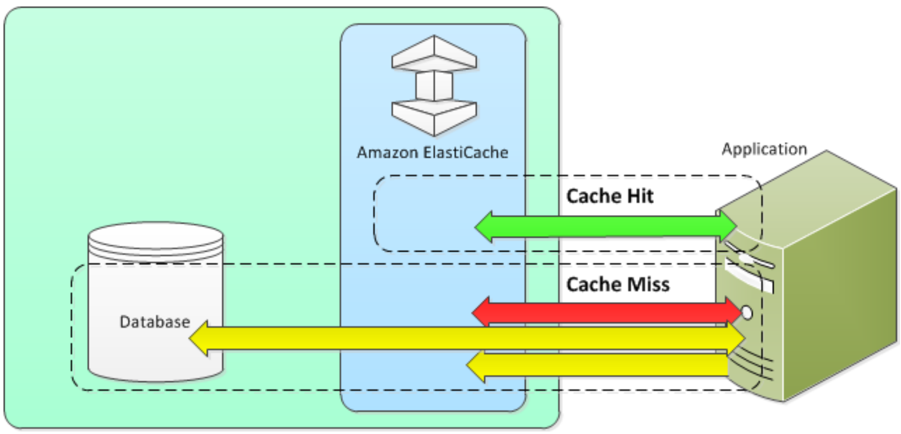
2. Cache returns the data to the application.

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. 72) and Adding TTL (p. 73) 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. 70) 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. 73) 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 (Redis can specify seconds or milliseconds) 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 Redis `set` command or 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

- What Should I Cache? (p. 2)

- Engines and Versions (p. 32)
- Scaling (p. 169)

ElastiCache 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 either Memcached or Redis, depending on what was selected when the cluster was created. 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.

The node instance type you need for your deployment is influenced by both the amount of data you want in your cluster and the engine you use. Generally speaking, due to its support for sharding, Memcached deployments will have more and smaller nodes while Redis deployments will use fewer, larger node types. See Selecting Your Memcached Node Size (p. 78) and Selecting Your Redis Node Size (p. 79) for a more detailed discussion of which node size to use.

Topics

Other ElastiCache Node Operations

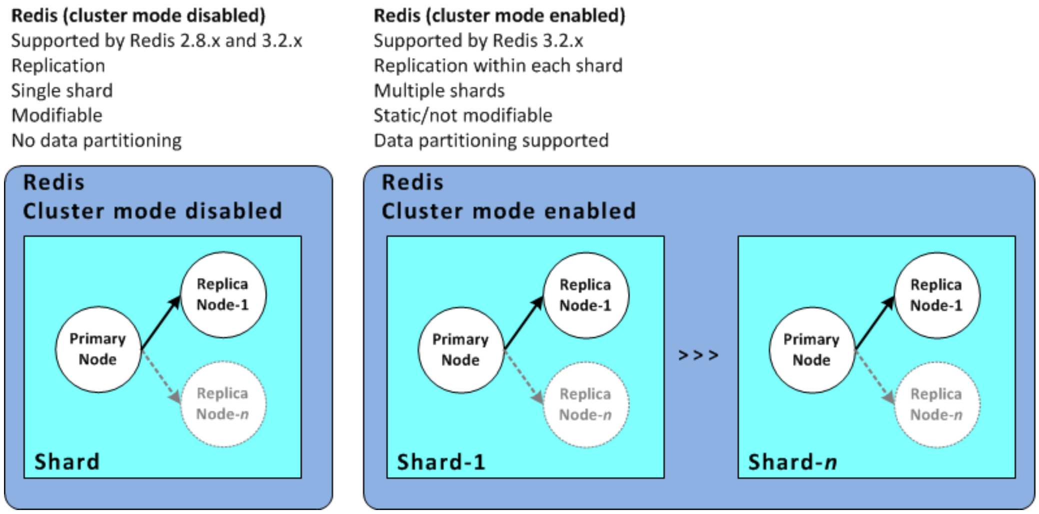
Additional operations involving nodes:

- Adding Nodes to a Cluster (p. 154)
- Removing Nodes from a Cluster (p. 160)
- Scaling (p. 169)
- Finding Your ElastiCache Endpoints (p. 48)
- Node Auto Discovery (Memcached) (p. 95)

Shards (Redis)

A shard (API/CLI: node group) is a hierarchical arrangement of nodes (each wrapped in a cache cluster). Shards support replication. Within a shard, one node functions as the read/write primary node. All the other nodes in a shard function as read-only replicas of the primary node. Redis version 3.2 and later support multiple shards within a cluster (API/CLI: replication group) thereby enabling partitioning your data in a Redis (cluster mode enabled) cluster.

The following diagram illustrates the differences between a Redis (cluster mode disabled) cluster and a Redis (cluster mode enabled) cluster.



Both Redis (cluster mode disabled) and Redis (cluster mode enabled) support replication via shards. The API operation, DescribeReplicationGroups (CLI: describe-replication-groups) lists the node groups with the member nodes, the node's role within the node group as well as other information.

When you create a Redis cluster, you specify whether or not you want to create a cluster with clustering enabled. Redis (cluster mode disabled) clusters never have more than one shard which can be scaled horizontally by adding (up to a total of 5) or deleting read replica nodes. For more information, see ElastiCache Replication (Redis) (p. 190), Adding a Read Replica to a Redis Cluster (p. 229) or Deleting a Read Replica (p. 234). Redis (cluster mode disabled) clusters can also scale vertically by changing node types. For more information, see Scaling Redis Clusters with Replica Nodes (p. 180).

When you create a Redis (cluster mode enabled) cluster, you specify from 1 to 15 shards. Currently, however, unlike Redis (cluster mode disabled) clusters, once a Redis (cluster mode enabled) cluster is created, its structure cannot be altered in any way; you cannot add or delete nodes or shards. If you need to add or delete nodes, or change node types, you must create the cluster anew.

When you create a new cluster, as long as the cluster group has the same number of shards as the old cluster, you can seed it with data from the old cluster so it doesn't start out empty. This can be helpful if you need change your node type or engine version. For more information, see Taking Manual Backups (p. 239) and Restoring From a Backup (p. 255).

Selecting Your Node Size

This section helps you determine what node instance type you need for your scenarios. Since the engines, Memcached and Redis, implement clusters differently, the engine you select will make a difference in the node size you needed by your application.

Topics

Selecting Your Memcached Node Size

Memcached clusters contain one or more 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 many smaller nodes. Further, as your needs change, you can add 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 cache nodes in the cluster by the RAM capacity of each node. The capacity of each cache node is based on the cache node type.

The number of cache nodes in the cluster is a key factor in the availability of your cluster running Memcached. The failure of a single cache 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 cache node and it gets repopulated. You can reduce this potential availability impact by spreading your memory and compute capacity over a larger number of cache nodes, each with smaller capacity, rather than using a fewer number of high capacity nodes.

In a scenario where you want to have 40 GB of cache memory, you can set it up in any of the following configurations:

- 13 `cache.t2.medium` nodes with 3.22 GB of memory and 2 threads each = 41.86 GB and 26 threads.

- 7 `cache.m3.large` nodes with 6.05 GB of memory and 2 threads each = 42.35 GB and 14 threads.

  7 `cache.m4.large` nodes with 6.42 GB of memory and 2 threads each = 44.94 GB and 14 threads.

- 3 `cache.r3.large` nodes with 13.50 GB of memory and 2 threads each = 40.50 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	Cost *	Nodes Needed	Total Memory	Total Cores	Monthly Cost #
cache.t2.medium	3.22 GB	2	$ 0.068	13	41.86 GB	26	$ 636.48
cache.m3.large	6.05 GB	2	$ 0.182	7	42.35 GB	14	$ 917.28
cache.m4.large	6.42 GB	2	$ 0.156	7	44.94 GB	14	$ 768.24

Node type	Memory	Cores	Cost *	Nodes Needed	Total Memory	Total Cores	Monthly Cost #
cache.r3.large	13.50 GB	2	$ 0.228	3	40.50 GB	6	$ 492.48
cache.m4.xlarge	14.28 GB	4	$ 0.311	3	42.84 GB	12	$ 671.76
* Hourly cost per node as of August 4, 2016.							
# 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 cache node is used for connection overhead. For more information, see Memcached Connection Overhead (p. 290)

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 cache cluster. For more information, see Node Auto Discovery (Memcached) (p. 95).

If you're unsure about how much capacity you need, for testing we recommend starting with one `cache.m3.medium` 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. 353). For production and larger workloads, the R3 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 cache node type that provides more compute power.

Selecting Your Redis Node Size

Answering the following questions will help you determine the minimum node type you need for your Redis implementation.

- How much total memory do you need for your data?

  You can get a general estimate by taking the size of the items you want to cache and multiplying it by the number of items you want to keep in the cache at the same time. To get a reasonable estimation of the item size, serialize your cache items then count the characters, then divide this over the number of shards in your cluster.

- How write-heavy is your application?

  Write heavy applications can require significantly more available memory, memory not used by Redis data, when taking snapshots or failing over. Whenever the `BGSAVE` process is performed–when taking a snapshot, when syncing a primary cluster with a replica in a cluster, when enabling the append-only file (AOF) feature, or promoting a replica to primary (if you have Multi-AZ with auto failover enabled)–you must have sufficient memory that is unused by data to accommodate all the writes that transpire during the BGSAVE process. Worst case would be when all of your Redis data is rewritten during the process, in which case you would need a node instance size with twice as much memory as needed for data alone.

For more detailed information, go to Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60).

- Will your implementation be a standalone Redis (cluster mode disabled) cluster or a Redis (cluster mode enabled) cluster with multiple shards?

  Redis (cluster mode disabled) cluster

  If you're implementing a Redis (cluster mode disabled) cluster, your node type must be able to accommodate all your data plus the necessary overhead as described in the previous bullet.

  For example, if you estimate that the total size of all your items to be 12 GB, you can use a `cache.m3.xlarge` node with 13.3 GB of memory or a `cache.r3.large` node with 13.5 GB of memory. However, you may need more memory for BGSAVE operations. If your application is write heavy, you should double the memory requirements to at least 24 GB, meaning you should use either a `cache.m3.2xlarge` with 27.9 GB of memory or a `cache.r3.xlarge` with 28.4 GB of memory.

  Redis (cluster mode enabled) with multiple shards

  If you're implementing a Redis (cluster mode enabled) cluster with multiple shards, then the node type must be able to accommodate `bytes-for-data-and-overhead / number-of-shards` bytes of data.

  For example, if you estimate that the total size of all your items to be 12 GB and you have 2 shards, you can use a `cache.m3.large` node with 6.05 GB of memory (12 GB / 2). However, you may need more memory for BGSAVE operations. If your application is write heavy, you should double the memory requirements to at least 12 GB per shard, meaning you should use either a `cache.m3.xlarge` with 13.3 GB of memory or a `cache.r3.large` with 13.5 GB of memory.

  Currently you cannot add shards to a Redis (cluster mode enabled) cluster. Therefore, you may want to use a somewhat larger node type to accommodate anticipated growth.

While your cluster is running, you can monitor the memory usage, processor utilization, cache hits, and cache misses metrics that are published to CloudWatch. If your cluster does not have the desired hit rate or you notice that keys are being evicted too often, you can choose a different cache node size with larger CPU and memory specifications.

When monitoring CPU usage, remember that Redis is single-threaded, so you need to multiply the reported CPU usage by the number of CPU cores to get that actual usage. For example, a four core CPU reporting a 20% usage rate is actually the one core Redis is using running at 80%.

ElastiCache Reserved Nodes

Reserved cache nodes let you make a one-time up-front payment for a cache node and reserve the cache node for a one- or three-year term at significantly lower hourly rates.

For the t2, m3, and r3 families, reserved cache nodes are available as Heavy Utilization offerings. Reserved cache nodes for older node types are available in three varieties—Heavy Utilization, Medium Utilization, and Light Utilization—that enable you to optimize your ElastiCache costs based on your expected utilization.

You can use the command line tools, the API, or the AWS Management Console to list and purchase available reserved cache node offerings. The three types of reserved cache node offerings are based on class and duration.

For more information on reserved nodes, go to Amazon ElastiCache Reserved Cache Nodes.

Topics

Reserved Node Offerings

Heavy Utilization reserved cache nodes enable workloads that have a consistent baseline of capacity or run steady-state workloads. Heavy Utilization reserved cache nodes require the highest up-front commitment, but if you plan to run more than 79 percent of the reserved cache node term you can earn the largest savings (up to 70 percent off of the On-Demand price). Unlike the other reserved cache nodes, with Heavy Utilization reserved cache 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 cache node is running.

Medium Utilization reserved cache nodes are the best option if you plan to leverage your reserved cache nodes a substantial amount of the time, but you want either a lower one-time fee or the flexibility to stop paying for your cache node when you shut it off. Medium Utilization reserved cache nodes are a more cost-effective option when you plan to run more than 40 percent of the reserved cache nodes term. This option can save you up to 64 percent off of the On-Demand price. With Medium Utilization reserved cache nodes, you pay a slightly higher one-time fee than with Light Utilization reserved cache nodes, and you receive lower hourly usage rates when you run a cache node.

Light Utilization reserved cache 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 cache nodes, you pay a one-time fee followed by a discounted hourly usage fee when your cache node is running. You can start saving when your cache node is running more than 17 percent of the reserved cache node term, and you can save up to 56 percent off of the On-Demand rates over the entire term of your reserved cache 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 cache node.	Lowest overall cost if you plan to use your reserved cache nodes more than 79 percent of a three-year term.

Offering	Up-Front Cost	Usage Fee	Advantage
Medium Utilization	Average	Hourly usage fee charged for each hour you use the cache node.	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. Highest fees of all the offering types, but fees apply only when you're using the reserved cache node.	Highest overall cost if you plan to run all of the time; however, lowest overall cost if you anticipate you will use your reserved cache nodes infrequently, more than about 15 percent of a three-year term.

Describing Available Reserved Cache Node Offerings

Before you purchase a reserved cluster, you can get information about available reserved cluster offerings.

The following example shows how to get pricing and information about available reserved cluster offerings.

Describing Available Reserved Cache Node Offerings (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, select the **Reserved Cache Nodes** link.
3. Select the **Purchase Reserved Cache Node** button.
4. From the **Product Description** drop down list box, select the engine - Memcached or Redis.
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 shows in the **Purchase Reserved Cache Nodes** wizard.
6. Select **Cancel** to avoid purchasing these nodes and incurring charges.

Describing Available Reserved Cache Node Offerings (AWS CLI)

To get pricing and information about available reserved cluster offerings, type the following command at a command prompt:

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

This call returns output similar to the following:

```
OFFERING  OfferingId                            Class            Duration
 Fixed Price  Usage Price  Description  Offering Type
OFFERING  438012d3-4052-4cc7-b2e3-8d3372e0e706  cache.m1.large   1y
 1820.00 USD  0.368 USD    memcached    Medium Utilization
OFFERING  649fd0c8-cf6d-47a0-bfa6-060f8e75e95f  cache.m1.small   1y
 227.50 USD  0.046 USD    memcached    Medium Utilization
OFFERING  123456cd-ab1c-47a0-bfa6-12345667232f  cache.m1.small   1y
 162.00 USD   0.00 USD    memcached    Heavy Utilization
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   0.123   USD       Hourly
OFFERING  123456cd-ab1c-37a0-bfa6-12345667232d  cache.m1.large   1y
 700.00 USD   0.00 USD    memcached    Heavy Utilization
    Recurring Charges:   Amount  Currency  Frequency
```

```
    Recurring Charges:   1.25    USD       Hourly
OFFERING  123456cd-ab1c-17d0-bfa6-12345667234e  cache.m1.xlarge  1y
 4242.00 USD   2.42 USD    memcached    Light Utilization
```

Describing Available Reserved Cache Node Offerings (ElastiCache API)

To get pricing and information about available reserved cluster 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=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:

```
<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>

      (...output omitted...)

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

Some of the output has been omitted for brevity.

Purchasing a Reserved Node

The following example shows how to purchase a reserved node offering.

> **Important**
> Following the examples in this section will incur charges on your AWS account.

Purchasing a Reserved Node (Console)

This example shows purchasing a specific reserved cache node offering, *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f*, with a reserved cache node ID of *myreservationID*.

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, select the **Reserved Cache Nodes** link.
3. Select the **Purchase Reserved Cache Node** button.
4. Select the cache node type from the **Product Description** drop-down list box.
5. Select the cache node class from the **Cache Node Class** drop-down list box.
6. Select length of time you want the cache node reserved from the **Term** drop-down list box.
7. Select the offering type from the **Offering Type** drop-down list box.
8. You can optionally enter a reserved cache 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.
9. Select the **Next** button.

   The **Purchase Reserved Cache Node** dialog box shows a summary of the reserved cache node attributes that you've selected and the payment due.
10. Select the **Yes, Purchase** button to proceed and purchase the reserved cache node.

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

Purchasing a Reserved Node (AWS CLI)

The following example shows purchasing a specific reserved cluster offering, *649fd0c8-cf6d-47a0-bfa6-060f8e75e95f*, with a reserved cluster ID of *myreservationID*.

Type the following command at a command prompt:

For Linux, OS X, 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
```

Purchasing a Reserved Node (ElastiCache API)

The following example shows purchasing a specific reserved cluster 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>
```

Describing Your Reserved Nodes

You can get information about reserved nodes for your AWS account as described following.

Describing Your Reserved Nodes (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, select the **Reserved Cache Nodes** link.

   The reserved cache nodes for your account appear in the Reserved Cache Nodes list. You can select any of the reserved cache nodes in the list to see detailed information about the reserved cache node in the detail pane at the bottom of the console.

Describing 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 --headers
```

This command should return output similar to the following:

```
RESERVATION  ReservationId     Class           Start Time
   Duration  Fixed Price  Usage Price  Count  State    Description
 Offering Type
RESERVATION  ki-real-ri-test5  cache.m1.small  2013-07-09T23:37:44.720Z  1y
        455.00 USD   0.092 USD    1      retired  memcached        Medium
 Utilization
```

Describing 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>
      <ReservedCacheNode>

      (...output omitted...)

      </ReservedCacheNode>
    </ReservedCacheNodes>
  </DescribeReservedCacheNodesResult>
  <ResponseMetadata>
    <RequestId>23400d50-2978-11e1-9e6d-771388d6ed6b</RequestId>
  </ResponseMetadata>
</DescribeReservedCacheNodesResponse>
```


Some of the output has been omitted for brevity.

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: `cache.t2.micro`, `cache.t2.small`, `cache.t2.medium`, `cache.m3.medium`, `cache.m3.large`, `cache.m3.xlarge`, `cache.m3.2xlarge`, `cache.m4.large`, `cache.m4.xlarge`, `cache.m4.2xlarge`, `cache.m4.4xlarge`, `cache.m4.10xlarge`
  - Previous generation: `cache.t1.micro`, `cache.m1.small`, `cache.m1.medium`, `cache.m1.large`, `cache.m1.xlarge`
- Compute optimized: `cache.c1.xlarge`
- Memory optimized:
  - Current generation: `cache.r3.large`, `cache.r3.xlarge`, `cache.r3.2xlarge`, `cache.r3.4xlarge`, `cache.r3.8xlarge`
  - Previous generation: `cache.m2.xlarge`, `cache.m2.2xlarge`, `cache.m2.4xlarge`

Supported node types are available in all regions except as noted in the following table.

Exceptions

Region Name	Region	Exception
Asia Pacific (Seoul)	ap-northeast-2	Supports only *current generation* node types.
EU (Frankfurt)	eu-central-1	Supports only *current generation* node types.
AWS GovCloud (US)	us-gov-west-1	Supports only *current generation* node types. Does not support M4 node types.
US East (Ohio)	us-east-2	Supports only node types T2, M4, and R3. Supports only the following engine versions: • Memcached: 1.4.24 • Redis 2.8.21, 2.8.23, 2.8.24, and 3.2.4

Note

- All T2 instances are created in an Amazon VPC (Amazon VPC).
- Redis backup and restore is not supported for T2 instances.
- Redis append-only files (AOF) are not supported for T1 or T2 instances.
- Redis Multi-AZ with automatic failover is not supported on T1 or T2 instances.
- Redis configuration variables *`appendonly`* and *`appendfsync`* are not supported on Redis version 2.8.22 and later.

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

- Amazon ElastiCache Product Features and Details

- Memcached Node-Type Specific Parameters
- Redis Node-Type Specific Parameters

Actions You Can Take When a Node is Scheduled for Replacement

The following sections specify actions you can take when ElastiCache schedules one or more of your nodes for replacement.

Memcached

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 from ElastiCache, if you change your maintenance window before the scheduled replacement time, your node will now be replaced at the new time. The new maintenance time can be no earlier than the originally scheduled time, and no later than a week from the originally scheduled time.

  For example, suppose your scheduled maintenance is planned for Monday, July 4th and your maintenance window is set to Mondays, 04:00-05:00 UTC. If you now change the maintenance window to Monday, 08:00-09:00 UTC, your replacement will occur between 08:00-09:00 UTC on Monday, July 4th. If you change your maintenance window to Monday, 01:00-02:00 UTC, your replacement will occur between 01:00-02:00 UTC on Monday, July 11th i.e. the following week. For instructions, see Maintenance Window (p. 43).
- **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. 160).
  2. Add a new node to the cluster. For instructions, see Adding Nodes to a Cluster (p. 154).
  3. If you are not using Node Auto Discovery (Memcached) (p. 95) on this cluster, go to your application and replace every instance of the old node's endpoint with the new node's endpoint.

Redis

The following list identifies actions you can take when ElastiCache schedules one of your Redis nodes for replacement. To expedite finding the information you need for your situation, select from the following menu.

- Do nothing (p. 93) – Let Amazon ElastiCache replace the node as scheduled.
- Change your maintenance window (p. 93) – Change your maintenance window to a better time.
- Replace a read-replica (p. 93) – A procedure to manually replace a read-replica in a Redis replication group.
- Replace the primary node (p. 93) – A procedure to manually replace the primary node in a Redis replication group.
- Replace a standalone node (p. 94) – Two different procedures to replace a standalone Redis node.

Redis node replacement options

- **Do nothing** – If you do nothing, ElastiCache will replace the node as scheduled.

  If the node is a member of a Redis (cluster mode disabled) cluster, the replacement node will sync with the primary node.

  If the node is standalone, ElastiCache will first launch a replacement node and then sync from the existing node. The existing node will not be available for service requests during this time. Once the sync is complete, the existing node is terminated and the new node takes its place. ElastiCache makes a best effort to retain your data during this operation.
- **Change your maintenance window** – For scheduled maintenance events where you receive an email from ElastiCache, if you change your maintenance window before the scheduled replacement time, your node will now be replaced at the new time. The new maintenance time can be no earlier than the originally scheduled time, and no later than a week from the originally scheduled time.

  For example, suppose your scheduled maintenance is planned for Monday, July 4th and your maintenance window is set to Mondays, 04:00-05:00 UTC. If you now change the maintenance window to Monday, 08:00-09:00 UTC, your replacement will occur between 08:00-09:00 UTC on Monday, July 4th. If you change your maintenance window to Monday, 01:00-02:00 UTC, your replacement will occur between 01:00-02:00 UTC on Monday, July 11th i.e. the following week. For instructions, see Maintenance Window (p. 43).
- **Replace a read-replica** – If the node is a read-replica, replace the node.

  If your cluster has only 2 nodes and Multi-AZ is enabled, you must disable Multi-AZ before you can delete the replica. For instructions, see Modifying a Cluster with Replicas (p. 226).

  #### To replace a read replica

  1. Delete the replica that is scheduled for replacement. For instructions, see Deleting a Cluster (p. 167).
  2. Add a new replica to replace the one that is scheduled for replacement. If you use the same name as the replica you just deleted, you can skip step 3. For instructions, see Adding a Read Replica to a Redis Cluster (p. 229).
  3. In your application, replace the old replica's endpoint with the new replica's endpoint.
  4. If you disabled Multi-AZ at the start, re-enable it now. For instructions, see Enabling Multi-AZ with Automatic Failover (p. 200).
- **Replace the primary node** – If the node is the primary node, promote a read-replica to primary, and then delete the former primary node.

  If your cluster has only two nodes and Multi-AZ is enabled, you must disable Multi-AZ before you can delete the replica in step 2. For instructions, see Modifying a Cluster with Replicas (p. 226).

  #### To replace a primary node

  1. Promote a read-replica to primary. For instructions, see Promoting a Read-Replica to Primary (p. 231).
  2. Delete the node that is scheduled for replacement (the old primary). For instructions, see Deleting a Cluster (p. 167).
  3. Add a new replica to replace the one scheduled for replacement. If you use the same name as the node you just deleted, you can skip step 4.

     For instructions, see Adding a Read Replica to a Redis Cluster (p. 229).
  4. In your application, replace the old node's endpoint with the new node's endpoint.
  5. If you disabled Multi-AZ at the start, re-enable it now. For instructions, see Enabling Multi-AZ with Automatic Failover (p. 200).

- **Replace a standalone node** – If the node does not have any read replicas, you have two options to replace it:

  Option 1: Replace the node using backup and restore

  1. Create a snapshot of the node. For instructions, see Taking Manual Backups (p. 239).
  2. Create a new node seeding it from the snapshot. For instructions, see Restoring From a Backup (p. 255).
  3. Delete the node scheduled for replacement. For instructions, see Deleting a Cluster (p. 167).
  4. In your application, replace the old node's endpoint with the new node's endpoint.

  Option 2: Replace the node using replication

  1. Add replication to the cluster with the node scheduled for replacement as the primary. Do not enable Multi-AZ on this cluster. For instructions, see To add replication to a Redis cluster with no shards (p. 154).
  2. Add a read-replica to the cluster. For instructions, see To add nodes to a Memcached or Redis (cluster mode disabled) cluster with one shard (console) (p. 155).
  3. Promote the newly created read-replica to primary. For instructions, see Promoting a Read-Replica to Primary (p. 231).
  4. Delete the node scheduled for replacement. For instructions, see Deleting a Cluster (p. 167).
  5. In your application, replace the old node's endpoint with the new node's endpoint.

Node Auto Discovery (Memcached)

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 only available for cache clusters running the Memcached engine. Redis cache clusters are single node clusters, thus there is no need to identify and track all the nodes in a Redis cluster.

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

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.

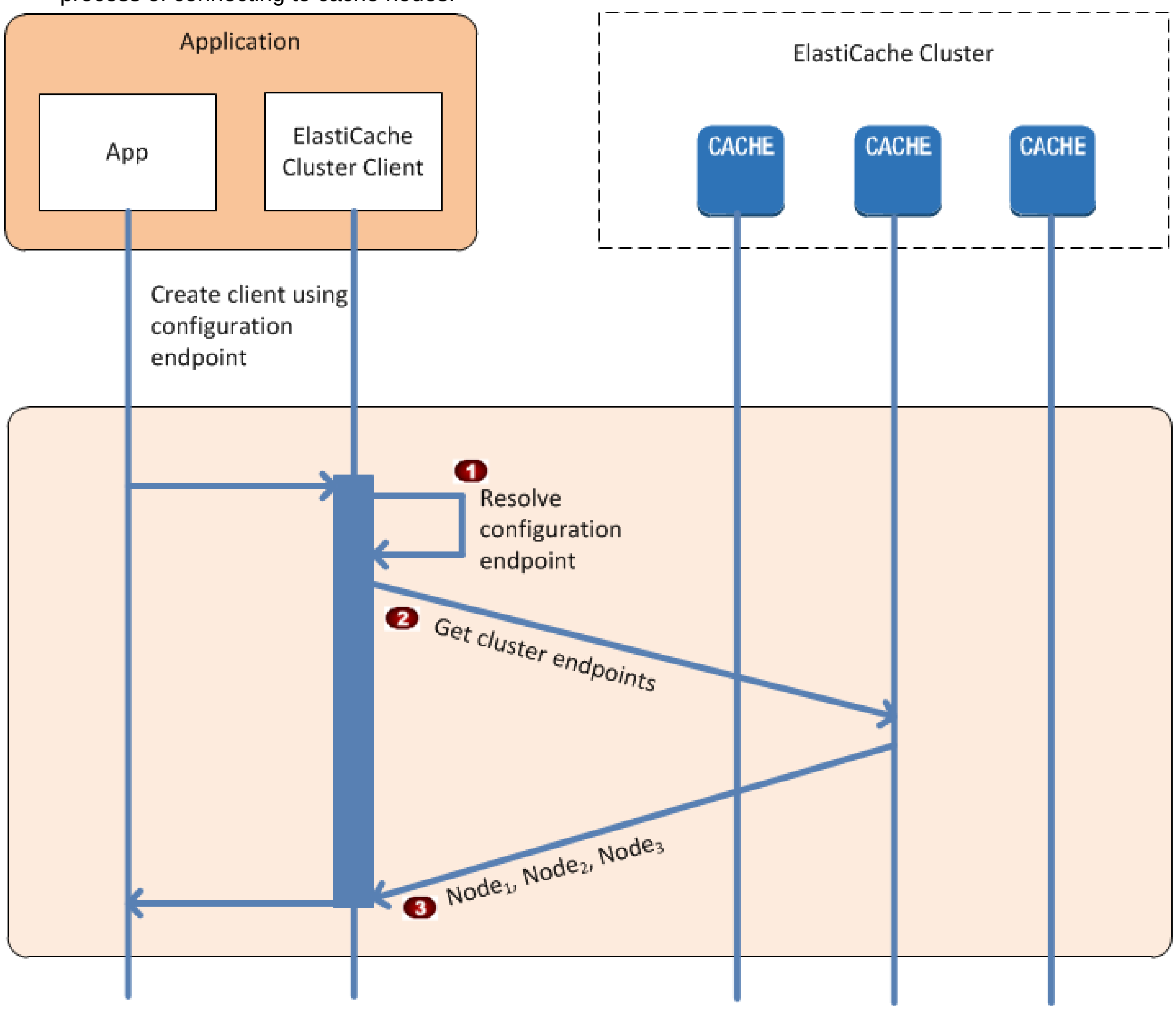
How Auto Discovery Works

Topics

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.



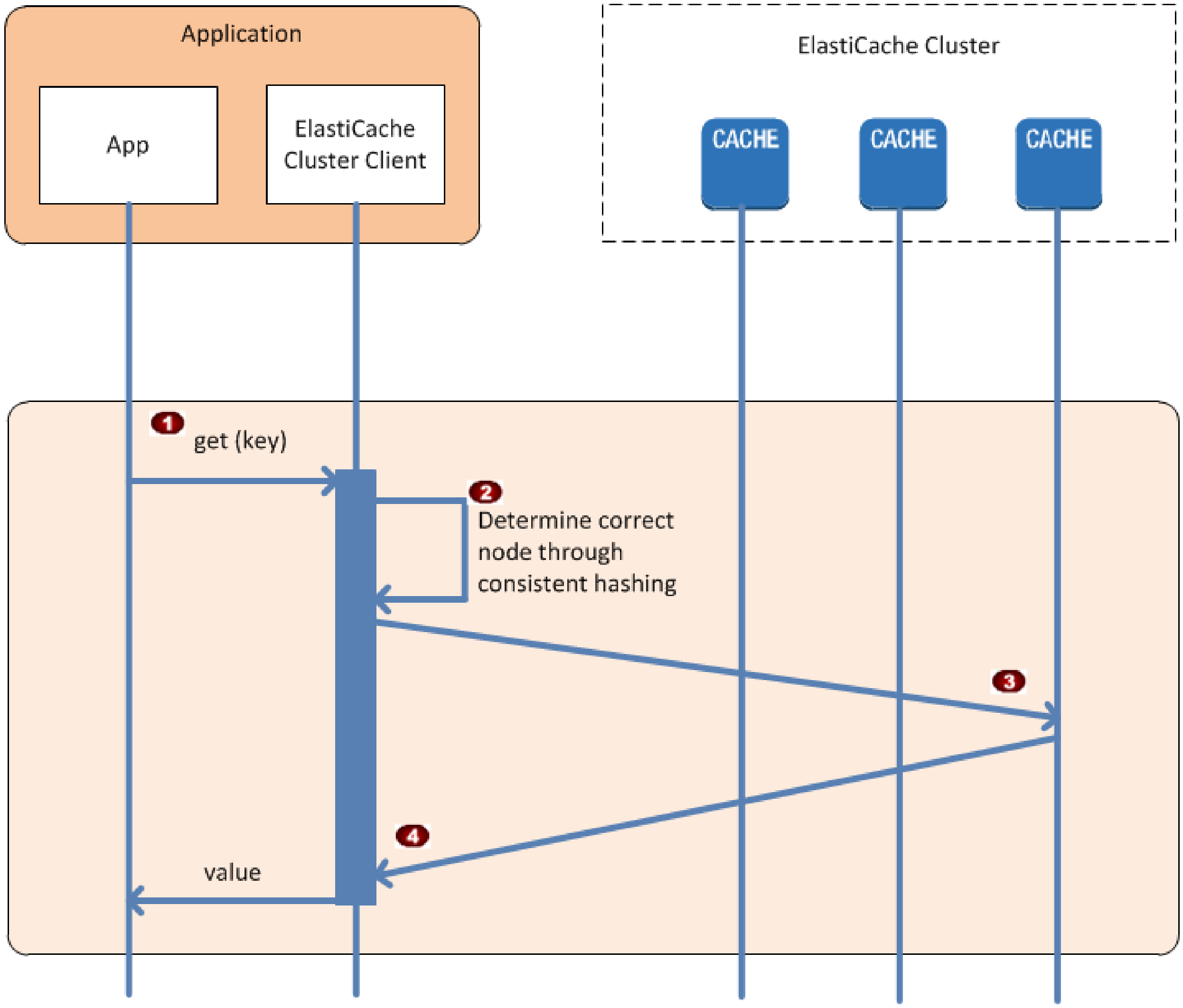
Process of Connecting to Cache Nodes

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 *get* 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.

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.

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.

Using Auto Discovery

To begin using Auto Discovery, follow these steps:

- Step 1: Obtain the Configuration Endpoint (p. 101)
- Step 2: Download the ElastiCache Cluster Client (p. 103)
- Step 3: Modify Your Application Program (p. 103)

Step 1: Obtain the Configuration Endpoint

To connect to a cluster, client programs must know the cluster configuration endpoint. See the topic Finding the Endpoints for a Memcached Cluster (Console) (p. 49)

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

Example Finding endpoints using the AWS CLI for ElastiCache

For Linux, OS X, or Unix:

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

For Windows:

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

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


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

Step 2: Download the ElastiCache Cluster Client

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

To download the ElastiCache Cluster Client

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

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

> **Note**
> To use the ElastiCache Cluster Client for PHP, you will first need to install it on your Amazon EC2 instance. For more information, see Installing the ElastiCache Cluster Client for PHP (p. 112).
> 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. 110).

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.

Topics

Using the ElastiCache Cluster Client for Java

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

```
package com.amazon.elasticache;

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

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

public class AutoDiscoveryDemo {

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

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

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

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

Using the ElastiCache Cluster Client for PHP

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

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. 112)

```
<?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 = "php-autodiscovery.1zvgtq.cfg.use1.cache.amazonaws.com";

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

 /**
  * The following will initialize a Memcached client to utilize the Auto 
 Discovery feature.
  * 
  * By configuring the client with the Dynamic client mode with single 
 endpoint, the
  * client will periodically use the configuration endpoint to retrieve the 
 current cache
  * cluster configuration. This allows scaling the cache cluster up or down 
 in number of nodes
  * without requiring any changes to the PHP application. 
  *
  * By default the Memcached instances are destroyed at the end of the 
 request. 
  * To create an instance that persists between requests, 
  *    use persistent_id to specify a unique ID for the instance. 
```

```
 * All instances created with the same persistent_id will share the same
connection.
 * See http://php.net/manual/en/memcached.construct.php for more
information.
 */
 $dynamic_client = new Memcached('persistent-id');
 $dynamic_client->setOption(Memcached::OPT_CLIENT_MODE,
Memcached::DYNAMIC_CLIENT_MODE);
 $dynamic_client->addServer($server_endpoint, $server_port);

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


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

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

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.use1.cache.amazonaws.com"
 port="11211" />
    </clusterclient>
</configuration>
```

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

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

using System;

using Amazon.ElastiCacheCluster;

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

public class DotNetAutoDiscoveryDemo  {

    public static void Main(String[] args)  {

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

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

    }  // end Main

}  // end class DotNetAutoDiscoverDemo
```

Connecting to Cache Nodes Manually

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

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

Example

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

```
...

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

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

...
```

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

Adding Auto Discovery To Your Client Library

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

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

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

The outputs from these two commands are identical, and are described in the Output Format (p. 109) section below.

Cache Engine Version 1.4.14 or Higher

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

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

Syntax

`config [sub-command] [key]`

Options

Name	Description	Required
`sub-command`	The sub-command used to interact with a cache node. For Auto Discovery, this sub-command is `get`.	Yes
`key`	The key under which the cluster configuration is stored. For Auto Discovery, this key is named `cluster`.	Yes

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

```
config get cluster
```

Cache Engine Version Lower Than 1.4.14

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

```
get AmazonElastiCache:cluster
```

> **Note**
> Do not tamper with the "AmazonElastiCache:cluster" key, since this is where the cluster configuration information resides. If you do overwrite this key, then the client may be

incorrectly configured for a brief period of time (no more than 15 seconds) before ElastiCache automatically and correctly updates the configuration information.

Output Format

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

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

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

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

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

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

Example

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

```
CONFIG cluster 0 147\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

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

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 Node Auto Discovery (Memcached) (p. 95). For sample .NET code to use the client, see Using the ElastiCache Cluster Client for .NET (p. 105).

Topics

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 left 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; go to 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, go to 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, go to 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 Node Auto Discovery (Memcached) (p. 95). For sample PHP code to use the client. see Using the ElastiCache Cluster Client for PHP (p. 104).

Topics

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. 118).

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, click **Download ElastiCache Cluster Client**.
3. Choose the ElastiCache Cluster Client that matches your PHP version and AMI architecture, and click the **Download ElastiCache Cluster Client** button.

Installation Steps for New Users

Topics

Installing PHP 7.x for New Users

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.
4. Download and unzip Amazon ElastiCache Cluster Client.
5. With root permissions, copy the extracted artifact file `amazon-elasticache-cluster-client.so` into `/usr/lib/php/20151012`.
6. Insert the line `extension=amazon-elasticache-cluster-client.so` into the file `/etc/php/7.0/cli/php.ini`.

To install PHP 7 on an Amazon Linux 201509 AMI or Red Hat 7 AMI

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

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

3. Install PHP 7.
4. Download and unzip Amazon ElastiCache Cluster Client.
5. With root permission, copy the extracted artifact file `amazon-elasticache-cluster-client.so` into `/usr/lib64/php/modules/`.
6. Insert the line `extension=amazon-elasticache-cluster-client.so` into file `/etc/php.ini`.

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.
4. Download and unzip Amazon ElastiCache Cluster Client.
5. With root permission, copy the extracted artifact file `amazon-elasticache-cluster-client.so` into `/usr/lib64/php7/extensions/`.
6. Insert the line `extension=amazon-elasticache-cluster-client.so` into the file `/etc/php7/cli/php.ini`.

Installing PHP 5.x for New Users

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. 112).
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 /etc/php.d/memcached.ini
```

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. 112).
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 /etc/
php.d/memcached.ini
```

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
```

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. 112).
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 /etc/php5/cli/conf.d/memcached.ini
```

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. 112).
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 /etc/
php5/conf.d/memcached.ini
```

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.

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. 116).
2. Install the new ElastiCache `php-memcached` extension as described previously in Installation Steps for New Users (p. 113).

Removing the PHP Cluster Client

Topics

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. 116).
2. Remove the line `extension=amazon-elasticache-cluster-client.so` from the `php.ini` file.

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

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/awslabs/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

Run the following set of commands under the code directory:

```
phpize
```

```
./configure --with-libmemcached-dir=<path to libmemcached build 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 build 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.

```
phpize
./configure --with-libmemcached-dir=<path to libmemcached build directory>
make
make install
```

Configuring ElastiCache Clients

An ElastiCache cluster is protocol-compliant with Memcached or Redis, depending on which cache engine was selected when the cluster was created. The code, applications, and most popular tools that you use today with your existing Memcached or Redis environments will work seamlessly with the service.

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

Topics

Restricted Commands

In order to deliver a managed service experience, ElastiCache restricts access to certain cache engine-specific commands that require advanced privileges.

- For cache clusters running Memcached, there are no restricted commands.
- For cache clusters running Redis, the following commands are unavailable:
  - `bgrewriteaof`
  - `bgsave`
  - `config`
  - `debug`
  - `migrate`
  - `save`
  - `slaveof`
  - `shutdown`

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 either **Memcached** or **Redis**.

   A list of all clusters running the selected engine appears.
2. Continue below for the engine and configuration you're running.

Memcached

1. Choose the name of the cluster of interest.

2. Locate the **Port** and **Endpoint** columns for the node you're interested in.

Redis: Non-cluster mode

1. Choose the name of the cluster of interest.
2. Locate the **Port** and **Endpoint** columns for the node you're interested in.

Redis: Cluster mode

1. Choose the name of the cluster of interest.

   A list of all the shards in that cluster appears.
2. Choose the name of the shard of interest.

   A list of all the nodes in that shard appears
3. 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
```

This command should produce output similar to the following:

```

CACHECLUSTER  my-memcached       https://console.aws.amazon.com/elasticache/
home#client-download:  2013-07-09T22:12:42.151Z  cache.t1.micro  memcached
 available  1  us-west-2a  1.4.14
      CACHESECURITYGROUP   default  active
      CACHEPARAMETERGROUP  default.memcached1.4  in-sync
      CACHENODE  0001  available  my-memcached.f310xz.cache.amazonaws.com
 11211  in-sync
CACHECLUSTER  my-redis-primary    https://console.aws.amazon.com/elasticache/
home#client-download:  2013-07-10T22:47:16.586Z  cache.m1.small  redis
 available  1  us-west-2a  2.6.13  repgroup01
      CACHESECURITYGROUP  default  active
      CACHEPARAMETERGROUP  default redis2.6  in-sync
      CACHENODE  0001  available  my-redis-
primary.f310xz.0001.cache.amazonaws.com  6379  in-sync
CACHECLUSTER  my-redis-replica-01  https://console.aws.amazon.com/
elasticache/home#client-download:  2013-07-10T23:11:07.704Z  cache.m1.small
 redis      available  1  us-west-2b  2.6.13  repgroup01
      CACHESECURITYGROUP  default  active
      CACHEPARAMETERGROUP  default redis2.6  in-sync
      CACHENODE  0001  available  my-redis-
replica-01.f310xz.0001.cache.amazonaws.com  6379  in-sync
```

The fully qualified DNS names and port numbers are in the CACHENODE lines in 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 Node Auto Discovery (Memcached) (p. 95).

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

Connecting to Nodes in a Redis Cluster

> **Note**
> At this time, clusters (API/CLI: replication groups) that support replication and read replicas are only supported for clusters running Redis.

For clusters, ElastiCache provides console, CLI, and API interfaces to obtain connection information for individual nodes.

For read-only activity, applications can connect to any node in the cluster. However, for write activity, we recommend that your applications connect to the primary endpoint (Redis (cluster mode disabled)) or configuration endpoint (Redis (cluster mode enabled)) for the cluster instead of connecting directly to a node. This will ensure that your applications can always find the correct node, even if you decide to reconfigure your cluster by promoting a read replica to the primary role.

Connecting to Clusters in a Cluster (Console)

To determine endpoints and port numbers

- See the topic, Finding the Endpoints for a Redis (cluster mode disabled) Cluster (Console) (p. 51).

Connecting to Clusters in a Replication Group (AWS CLI)

To determine cache node endpoints and port numbers

Use the command `describe-replication-groups` with the name of your replication group:

```
aws elasticache describe-replication-groups my-repgroup
```

This command should produce output similar to the following:

```
REPLICATIONGROUP  my-repgroup  My replication group  available
      CLUSTERID  my-redis-primary
      CLUSTERID  my-replica-1
      NODEGROUP  0001  my-repgroup.f310xz.ng.0001.cache.amazonaws.com  6379
 available
            NODEGROUPMEMBER  my-redis-primary    0001  my-redis-
primary.f310xz.0001.cache.amazonaws.com   6379  us-west-2a  primary
            NODEGROUPMEMBER  my-replica-1        0001  my-
replica-1.f310xz.0001.cache.amazonaws.com       6379  us-west-2b  replica
```

Connecting to Clusters in a Replication Group (ElastiCache API)

To determine cache node endpoints and port numbers

Call `DescribeReplicationGroups` with the following parameter:

`ReplicationGroupId` = the name of your replication group.

Example

```
https://elasticache.us-west-2.amazonaws.com /
    ?Action=DescribeCacheClusters
    &ReplicationGroupId=repgroup01
    &Version=2014-09-30
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20140421T220302Z
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Date=20140421T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Expires=20140421T220302Z
    &X-Amz-Credential=<credential>
    &X-Amz-Signature=<signature>
```

DNS Names and Underlying IP

Memcached and Redis 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 Memcached and Redis 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.

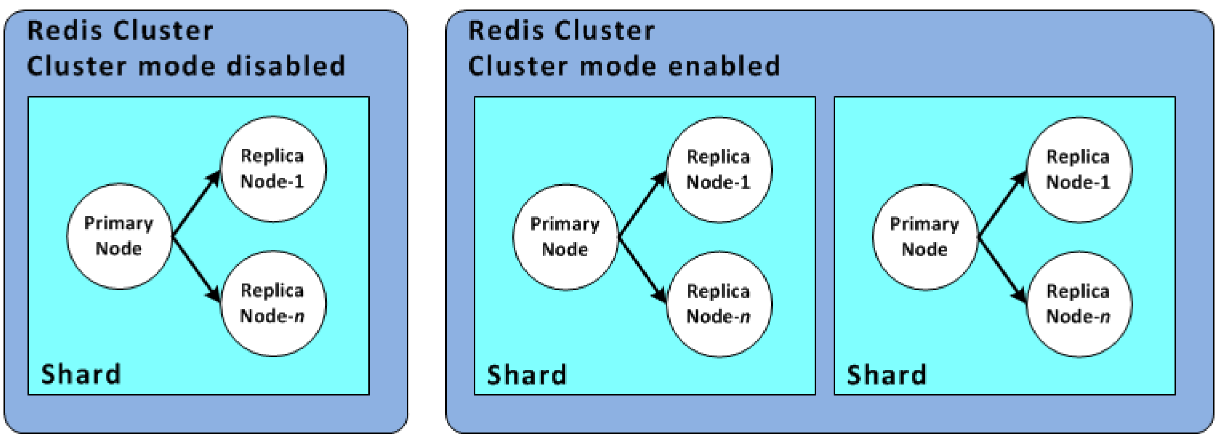
Shards (Redis)

A shard (API/CLI: node group) is a collection of 1 to 6 Redis nodes. A Redis (cluster mode disabled) cluster will never have more than one shard. Redis (cluster mode enabled) clusters can have from 1 to 15 shards. The cluster's data is partitioned across the cluster's shards. If there is more than one node in a shard, the shard implements replication with one node being the read/write primary node and the other nodes read-only replica nodes.

When you create a Redis (cluster mode enabled) cluster using the ElastiCache console, you specify the number of shards in the cluster and the number of nodes in the shards. For more information, see Creating a Redis (cluster mode enabled) Cluster (Console) (p. 134). If you use the ElastiCache API or AWS CLI to create a cluster (called *replication group* in the API/CLI), you can configure the number of nodes in a shard (API/CLI: node group) independently. For more information, see:

- API: CreateReplicationGroup
- CLI: create-replication-group

Each node in a shard has the same compute, storage and memory specifications. The ElastiCache API lets you control shard-wide attributes, such as the number of nodes, security settings, and system maintenance windows.

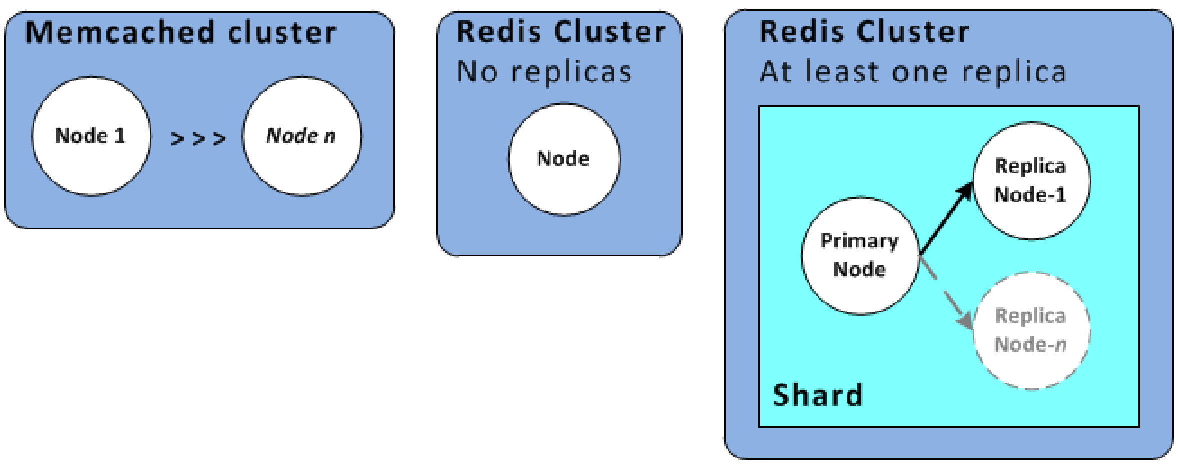


Redis shard configurations

ElastiCache Clusters

A *cluster* is a collection of one or more cache nodes, all of which run an instance of supported cache engine software, Memcached or Redis. When you create a cluster, you specify the engine that all of the nodes will use.

The following diagram illustrates a typical Memcached and a typical Redis cluster. Memcached clusters contain from 1 to 20 nodes across which you can horizontally partition your data. Redis clusters can contain a single node or up to 6 nodes inside a shard (API/CLI: node group), Redis (cluster mode disabled) clusters always have a single shard. Redis (cluster mode enabled) clusters can have up to 15 shards. When you have multiple nodes in a shard, one of the nodes is a read/write primary node. All other nodes in the shard are read-only replicas.



Typical Memcached and Redis Clusters

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 versions of each engine. Unless you have specific reasons, we recommend always using the your engine's latest version.

Memcached Versions

Redis Versions

Other ElastiCache Cluster Operations

Additional operations involving clusters:

- Finding Your ElastiCache Endpoints (p. 48)
- Accessing ElastiCache Resources from Outside AWS (p. 348)

Topics

Creating a Cluster

When you launch an Amazon ElastiCache cluster, you can choose to use the Memcached or Redis engine. The Redis engine has two flavors, Redis (cluster mode disabled) and Redis (cluster mode enabled) To determine which engine will best suit your needs, see Engines and Versions (p. 32) in this guide.

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

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

- Which engine you will use?

  For a comparison of engines and engine versions, see Engines and Versions (p. 32).
- Which node instance type do you need?

  For guidance on selecting an instance node type, see Selecting Your Node Size (p. 78).
- 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. 307).
  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. 348).
- 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. 273).

  If you're running Redis you may want to consider at least setting `reserved-memory`. For more information, see Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60).
- Do you need to create your own *Security Group* or *VPC Security Group*?

  For more information, see Security Groups [EC2-Classic] (p. 263) and Security in Your VPC.
- How do you intend to implement fault tolerance?

  For more information, see Mitigating Failures (p. 62).

Topics

Creating a Cluster (Console): Memcached

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 Node Auto Discovery (Memcached) (p. 95).

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. Choose **Get Started Now**.

   If you already have an available cluster, choose **Memcached** from the left navigation pane.
3. Choose **Create**.
4. For **Cluster engine**, choose *Memcached*.
5. 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. 41). 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. 286) and Creating a Parameter Group (p. 273).

   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 Selecting Your Node Size (p. 78).

   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 Memcached (p. 170)
6. 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. 307).

b. Choose how you want the **Availability zone(s)** selected for this cluster. You have two options.

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

     If you selected 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. 63).

   For more information, see Selecting Regions and Availability Zones (p. 45).

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

   For more information, see ElastiCache and Security Groups (p. 331).

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*, from the lists choose the *Start day*, *Start time*, and *Duration* (in hours) for your maintenance window. All times are UCT times.

   For more information, see Maintenance Window (p. 43).

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/.

7. Review all your entries and selections, then go back and make any needed corrections. When everything is just the way you want it, 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 4: Authorize Access (p. 24) and Step 5: Connect to a Cluster's Node (p. 26).

> **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. 167).

Creating a Redis (cluster mode disabled) Cluster (Console)

ElastiCache supports replication when you use the Redis engine. To monitor the latency between when data is written to a Redis read/write primary cluster and when it is propagated to a read-only secondary cluster, ElastiCache adds to the cluster a special key, `ElastiCacheMasterReplicationTimestamp`, which is the current Universal Coordinated Time (UCT time. Because a Redis cluster might be added to a replication group at a later time, this key is included in all Redis clusters, even if initially they are not members of a replication group. For more information on replication groups, see ElastiCache Replication (Redis) (p. 190).

To create a standalone Redis (cluster mode disabled) cluster

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Choose **Get Started Now**.

   If you already have an available cluster, choose **Redis** from the left navigation pane.
3. Choose **Create**.
4. For **Cluster engine**, choose **Redis**, and then clear the **Cluster Mode enabled (Scale Out)** check box.
5. Complete the **Redis settings** section.

   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. For **Engine version compatibility**, choose the Redis 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.

   Important
   You can upgrade to newer engine versions (see Upgrading Engine Versions (p. 41)), but you cannot downgrade to older engine versions except by deleting the existing Cluster and creating it anew.

   Because the newer Redis versions provide a better and more stable user experience, Redis versions 2.6.13, 2.8.6, and 2.8.19 are deprecated from the ElastiCache console. While we recommend against it, if you must use one of these older Redis versions, you can use the AWS CLI or ElastiCache API.

   For more information see the following topics:

	AWS AWS CLI	ElastiCache API
Create Cluster	Creating a Cache Cluster (AWS CLI) (p. 137) #	Creating a Cache Cluster (ElastiCache API) (p. 140) #
Modify Cluster	Modifying a Cache Cluster (AWS CLI) (p. 150) #	Modifying a Cache Cluster (ElastiCache API) (p. 150) #
Create Replication Group	Creating a Redis (cluster mode disabled) Replication Group from Scratch (AWS CLI) (p. 210) Creating a Redis (cluster mode enabled) Replication Group from Scratch (AWS CLI) (p. 215)	Creating a Redis (cluster mode disabled) Replication Group from Scratch (ElastiCache API) (p. 212) Creating a Redis (cluster mode enabled) Replication Group from Scratch (ElastiCache API) (p. 218)

	AWS AWS CLI	ElastiCache API
Modify Replication Group	Modifying a Replication Group (AWS CLI) (p. 226) #	Modifying a Replication Group (ElastiCache API) (p. 226) #
# Cannot be used to create a Redis (cluster mode enabled) cluster or replication group.		
# Cannot be used to modify a Redis (cluster mode enabled) cluster or replication group.		

c. In **Port**, accept the default port, 6379. If you have a reason to use a different port, type the port number.

d. For **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.

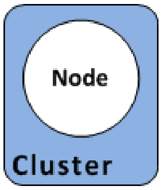
Parameter groups control the runtime parameters of your cluster. For more information on parameter groups, see Redis Specific Parameters (p. 292) and Creating a Parameter Group (p. 273).

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 Selecting Your Node Size (p. 78).

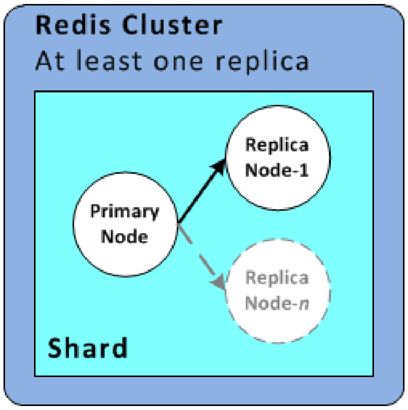
f. For **Number of replicas**, choose the number of read replicas you want for this cluster.

If you choose *None*, the **description** and **Multi-AZ with Auto-Failover** fields disappear and the cluster your create will look like this.



Redis (cluster mode disabled) cluster created with no replica nodes

If you choose one or more replicas, the cluster you create will look something like this.



Redis (cluster mode disabled) cluster created with replica nodes

6. Choose **Advanced Redis settings** and complete the section.

   a. If you chose to have one or more replicas, the **Multi-AZ with Auto-Failover** check box is available. We strongly suggest you enable Multi-AZ with Auto-Failover. For more information, see Mitigating Failures when Running Redis (p. 64).

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

For more information, see Subnets and Subnet Groups (p. 307).

c. Choose how you want the **Availability zone(s)** selected for this cluster. You have two options.

   - **No preference** – ElastiCache selects the availability zones for your cluster's nodes.
   - **Specify availability zones** – A list of your nodes appears allowing you to specify the Availability Zone for each node in your cluster by selecting the availability zone from the list to the right of each node name.

   For more information, see Selecting Regions and Availability Zones (p. 45).

d. For **Security groups**, choose the security groups you want for this cluster.

   For more information, see ElastiCache and Security Groups (p. 331).

e. If you are going to seed your cluster with data from a .RDB file, in the **Seed RDB file S3 location** box, type the Amazon S3 location of the .RDB file.

   For more information, see Using a Backup to Seed a Cluster (p. 257).

f. If you want regularly scheduled automatic backups, choose **Enable automatic backups**, and then type the number of days you want an automatic backup retained before it is automatically deleted. If you don't want regularly scheduled automatic backups, clear the **Enable automatic backups** check box. In either case, you always have the option to create manual backups which must be deleted manually.

   For more information on Redis backup and restore, see ElastiCache Backup & Restore (Redis) (p. 235).

g. 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* from the lists, choose the *Start day*, *Start time*, and *Duration* (in hours) for your maintenance window. All times are UCT times.

   For more information, see Maintenance Window (p. 43).

h. 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/.

7. Review all your entries and selections, then go back and make any needed corrections. When everything is just the way you want it, 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 4: Authorize Access (p. 24) and Step 5: Connect to a Cluster's Node (p. 26).

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. 167).

Creating a Redis (cluster mode enabled) Cluster (Console)

If you are running Redis 3.2.4 or later, you can create a Redis (cluster mode enabled) cluster. Redis (cluster mode enabled) clusters support partitioning your data across 1 to 15 shards (API/CLI: node groups) but with some limitations (currently). For a comparison of Redis (cluster mode disabled) and the two flavors of Redis (cluster mode enabled), see Selecting an Engine: Memcached, Redis (cluster mode disabled), or Redis (cluster mode enabled) (p. 33).

You can create a Redis (cluster mode enabled) cluster (API/CLI: replication group) using the ElastiCache management console, the AWS CLI for ElastiCache, and the ElastiCache API.

To create a Redis (cluster mode enabled) 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. Choose **Get Started Now**.

   If you already have an available cluster, choose **Redis** from the left navigation pane.
3. Choose **Create**.
4. For **Cluster engine**, choose **Redis**, and then choose **Cluster Mode enabled (Scale Out)**.
5. Complete the **Redis (cluster mode enabled) settings** section.

   a. In the **Name** box, 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. In the **Description** box, type in a description of this cluster.

   c. For **Engine version compatibility**, choose 3.2.4.

   d. In the **Port** box, accept the default port, 6379. If you have a reason to use a different port, type the port number.

   e. For **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 Redis Specific Parameters (p. 292) and Creating a Parameter Group (p. 273).

   f. For **Node type**, choose 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 Selecting Your Node Size (p. 78).

   g. For **Number of shards**, choose the number of shards (partitions) you want for this Redis (cluster mode enabled) cluster.

      In Redis (cluster mode enabled) once you create the cluster the number of shards is fixed and cannot be changed. If you find you need more or fewer shards, you must create a new cluster with the new number of shards. In Redis (cluster mode enabled), if the number of shards in the new cluster differs from the number of shards in the existing cluster, you will not be able to seed the new cluster with .RDB files from the old cluster.

h. For **Replicas per shard**, choose the number of read replica nodes you want in each shard.

   The following restrictions exist for Redis (cluster mode enabled).

   - The number of replicas is the same for each shard when creating the cluster using the console.
   - The number of read replicas per shard is fixed and cannot be changed. If you find you need more or fewer replicas per node group, you must create a new cluster with the new number of replicas. As long as the number of node groups in the new cluster is the same as in the old cluster, you will be able to seed the new cluster using .RDB files from the old cluster. For more information, see Using a Backup to Seed a Cluster (p. 257).

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

   For more information, see Subnets and Subnet Groups (p. 307).

6. Click **Advanced Redis settings** and complete the section.

   a. For **Slots and keyspaces**, choose how you want your keys distributed over your shards (partitions). There are 16,384 keys to be distributed (numbered 0 through 16,383).

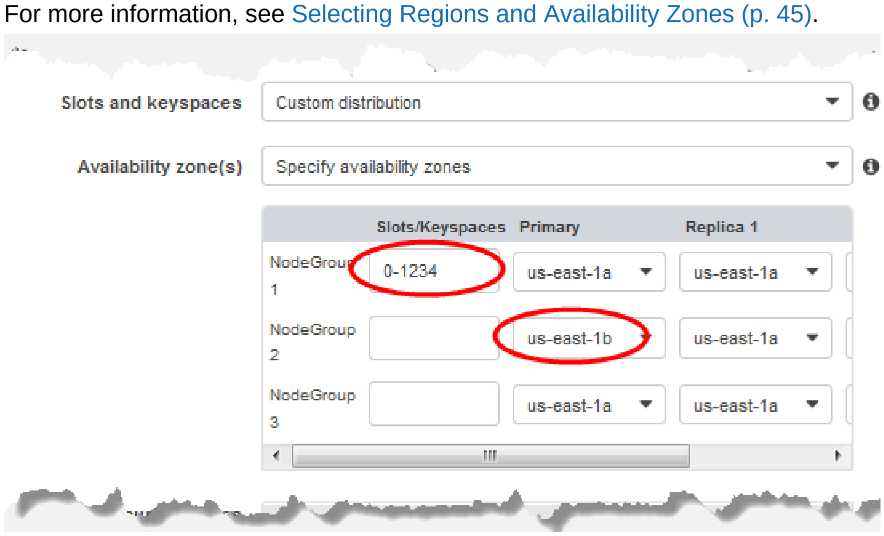
      - *Equal distribution* – ElastiCache distributes your keyspace as equally as possible over your shards.
      - *Custom distribution* – you specify the range of keys for each shard in the table below **Availability zone(s)**.

   b. Choose how you want the **Availability zone(s)** selected for this cluster. You have two options.

      - **No preference** – ElastiCache will choose the availability zone.
      - **Specify availability zones** – You specify the availability zone for each cluster.

        If you selected to specify the availability zones, for each cluster in each shard, choose the availability zone from the list.

   For more information, see Selecting Regions and Availability Zones (p. 45).



Specifying Keyspaces and Availability Zones

c. For **Security groups**, choose the security groups you want for this cluster.

   For more information, see ElastiCache and Security Groups (p. 331).

d. If you are going to seed your cluster with data from a .RDB file, in the **Seed RDB file S3 location** box, enter the S3 location of the .RDB file.

   For more information, see Using a Backup to Seed a Cluster (p. 257).

   For Redis (cluster mode enabled) you must have a separate .RDB file for each node group.

e. If you want regularly scheduled automatic backups, choose **Enable automatic backups** the type the number of days you want each automatic backup retained before it is automatically deleted. If you don't want regularly scheduled automatic backups, clear the **Enable automatic backups** check box. In either case, you always have the option to create manual backups.

   For more information on Redis backup and restore, see ElastiCache Backup & Restore (Redis) (p. 235).

f. 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 to 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* from the lists, choose the *Start day*, *Start time*, and *Duration* (in hours) for your maintenance window. All times are UCT times.

   For more information, see Maintenance Window (p. 43).

g. For **Notifications**, choose an existing Amazon Simple Notification Service (Amazon SNS) topic, or choose Manual ARN input and type in the topic's 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/.

7. Review all your entries and selections, then go back and make any needed corrections. When everything is just the way you want it, choose **Create cluster** to launch your cluster, or **Cancel** to cancel the operation.

To create the equivalent using the ElastiCache API or AWS CLI instead of the ElastiCache console, see:

- API: CreateReplicationGroup
- CLI: create-replication-group

As soon as your cluster's status is *available*, you can grant EC2 access to it, connect to it, and begin using it. For more information, see Step 4: Authorize Access (p. 24) and Step 5: Connect to a Cluster's Node (p. 26).

> **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. 167).

Creating a Cache Cluster (AWS CLI)

To create a cluster using the AWS CLI, use the `create-cache-cluster` command. The following example creates a single node Redis (cluster mode enabled) cluster named *my-redis-cluster* and seeds it with the snapshot file `snap.rdb` that has been copied to Amazon S3.

If you want a Redis cluster that supports replication, see Creating a Redis (cluster mode disabled) Replication Group from Scratch (AWS CLI) (p. 210).

> **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. 167).

Topics

Creating a Memcached Cache Cluster (AWS CLI)

The following CLI code creates a Memcached cache cluster.

For Linux, OS X, or Unix:

```
aws elasticache create-cache-cluster \
    --cache-cluster-id my-memcached-cluster \
    --cache-node-type cache.m4.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-memcached-cluster ^
    --cache-node-type cache.m4.large ^
    --engine memcached ^
    --engine-version 1.4.24 ^
    --cache-parameter-group default.memcached1.4 ^
    --num-cache-nodes 3
```

Creating a Redis (cluster mode disabled) Cache Cluster (AWS CLI)

Example A Redis (cluster mode disabled) Cluster

The following CLI code creates a Redis (cluster mode disabled) cache cluster.

For Linux, OS X, or Unix:

```
aws elasticache create-cache-cluster \
    --cache-cluster-id my-redis3-cluster \
    --cache-node-type cache.m4.large \
    --engine redis \
    --engine-version 3.2.4 \
    --num-cache-nodes 1 \
    --cache-parameter-group default.redis3.2 \
    --snapshot-arns arn:aws:s3:myS3Bucket/snap.rdb
```

For Windows:

```
aws elasticache create-cache-cluster ^
    --cache-cluster-id my-redis3-cluster ^
    --cache-node-type cache.m4.large ^
    --engine redis ^
    --engine-version 3.2.4 ^
    --num-cache-nodes 1 ^
    --cache-parameter-group default.redis3.2 ^
    --snapshot-arns arn:aws:s3:myS3Bucket/snap.rdb
```

Creating a Redis (cluster mode enabled) Cluster (AWS CLI)

The following CLI code creates a Redis (cluster mode enabled) cache cluster. The parameter `--cache-parameter-group` determines whether this is a clustered or non-clustered cluster running Redis version 3.2.4.

- `--cache-parameter-group default.redis3.2.cluster.on` – Makes this a clustered Redis cluster.
- `--cache-parameter-group default.redis3.2` – Makes this a non-clustered Redis cluster even though it is running Redis 3.2.4.

For Linux, OS X, or Unix:

```
aws elasticache create-cache-cluster \
    --cache-cluster-id my-redis3-cluster \
    --cache-node-type cache.m4.large \
    --engine redis \
    --engine-version 3.2.4 \
    --num-cache-nodes 1 \
    --cache-parameter-group default.redis3.2.cluster.on \
    --snapshot-arns arn:aws:s3:myS3Bucket/snap.rdb
```

For Windows:

```
aws elasticache create-cache-cluster ^
```

```
    --cache-cluster-id my-redis3-cluster ^
    --cache-node-type cache.m4.large ^
    --engine redis ^
    --engine-version 3.2.4 ^
    --num-cache-nodes 1 ^
    --cache-parameter-group default.redis3.2.cluster.on ^
    --snapshot-arns arn:aws:s3:myS3Bucket/snap.rdb
```

For more information, go to the AWS CLI for ElastiCache reference topic `create-cache-cluster`.

Creating a Cache Cluster (ElastiCache API)

To create a cluster using the ElastiCache API, use the `CreateCacheCluster` action. The following example creates a single node Redis cluster named *my-redis-cluster* and seeds it with the snapshot file `dump.rdb` that has been copied to Amazon S3.

If you want a Redis cluster that supports replication, see Creating a Redis (cluster mode disabled) Replication Group from Scratch (ElastiCache API) (p. 212).

> **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. 167).

Topics

Creating a Memcached Cache Cluster (ElastiCache API)

The following code creates a Memcached cache cluster (ElastiCache API).

Line breaks in the following code example are added for ease of reading.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=CreateCacheCluster
    &CacheClusterId=myMemcachedCluster
    &CacheNodeType=cache.m4.large
    &Engine=redis
    &NumCacheNodes=1
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &SnapshotArns.member.1=arn%3Aaws%3As3%3A%3A%3AmyS3Bucket%2Fdump.rdb
    &Timestamp=20150508T220302Z
    &Version=2015-02-02
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20150508T220302Z
    &X-Amz-Expires=20150508T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Signature=<signature>
```

Creating a Redis (cluster mode disabled) Cache Cluster (ElastiCache API)

The following code creates a Redis (cluster mode disabled) cache cluster (ElastiCache API).

Line breaks in the following code example are added for ease of reading.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=CreateCacheCluster
    &CacheClusterId=my-redis2-cluster
    &CacheNodeType=cache.m4.large
    &CacheParameterGroup=default
```

```
    &Engine=redis
    &EngineVersion=3.2.4
    &NumCacheNodes=1
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &SnapshotArns.member.1=arn%3Aaws%3As3%3A%3A%3AmyS3Bucket%2Fdump.rdb
    &Timestamp=20150508T220302Z
    &Version=2015-02-02
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20150508T220302Z
    &X-Amz-Expires=20150508T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Signature=<signature>
```

Creating a Redis (cluster mode enabled) Cache Cluster (ElastiCache API)

The following CLI code creates a Redis (cluster mode enabled) cache cluster. The parameter `CacheParameterGroup` determines whether this is a clustered or non-clustered cluster running Redis version 3.2.4.

- `CacheParameterGroup=default.redis3.2.cluster.on` – Makes this a clustered Redis cluster.
- `CacheParameterGroup=default.redis3.2` – Makes this a non-clustered Redis cluster even though it is running Redis 3.2.4.

Line breaks in the following code example are added for ease of reading.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=CreateCacheCluster
    &CacheClusterId=my-redis3-cluster
    &CacheNodeType=cache.m4.large
    &CacheParameterGroup=default.redis3.2.cluster.on
    &Engine=redis
    &EngineVersion=3.2.4
    &NumCacheNodes=1
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &SnapshotArns.member.1=arn%3Aaws%3As3%3A%3A%3AmyS3Bucket%2Fdump.rdb
    &Timestamp=20150508T220302Z
    &Version=2015-02-02
    &X-Amz-Algorithm=AWS4-HMAC-SHA256
    &X-Amz-Credential=<credential>
    &X-Amz-Date=20150508T220302Z
    &X-Amz-Expires=20150508T220302Z
    &X-Amz-SignedHeaders=Host
    &X-Amz-Signature=<signature>
```

For more information, go to the ElastiCache API reference topic `CreateCacheCluster`.

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: Memcached (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.

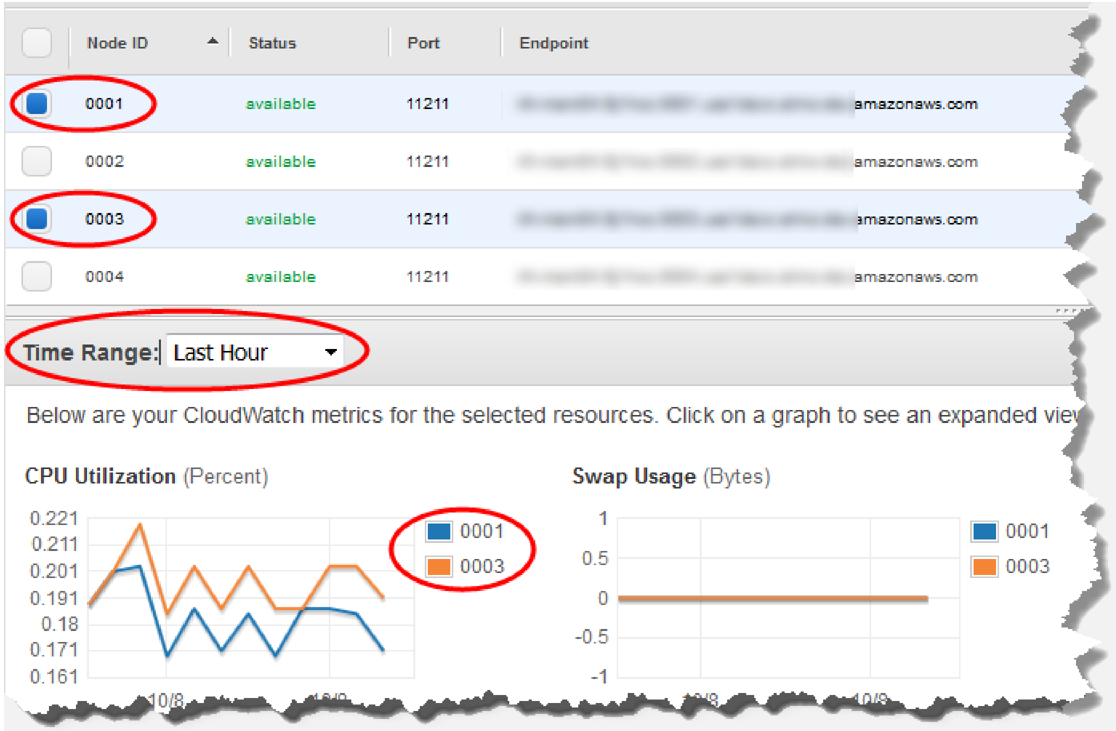
To view a Memcached cluster's details

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the ElastiCache console dashboard, choose **Memcached**. This will display a list of all your clusters that are running any version of Memcached.
3. To see details of a cluster, choose the box to the left of the cluster's name.
4. 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 Redis (cluster mode disabled) Cluster's Details (Console)

You can view the details of a Redis (cluster mode disabled) 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 Redis (cluster mode disabled) cluster using the ElastiCache console.

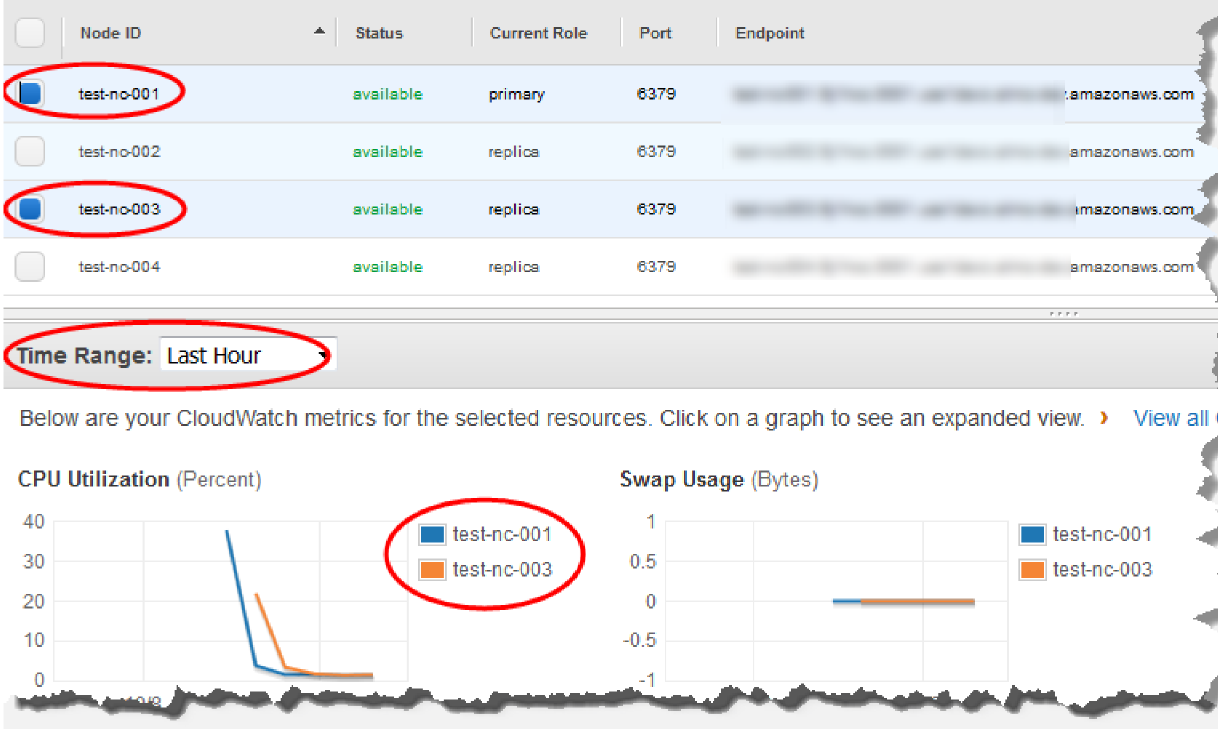
To view a Redis (cluster mode disabled) cluster's details

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the ElastiCache console dashboard, choose **Redis** to display a list of all your clusters that are running any version of Redis.
3. To see details of a cluster, select the check box to the left of the cluster's name. Make sure you select a cluster running the Redis engine, not Clustered Redis. This diplays details about the cluster, including the cluster's primary endpoint.
4. To view node information:

   a. Choose the cluster's name.

   b. Choose the **Nodes** tab. This diplays details about each node, including the node's endpoint which you need to use to read from the cluster.

   c. To view metrics on one or more nodes, select the box to the left of the node ID, then select the time range for the metrics from the **Time range** list. If you select multiple nodes, you can see overlay graphs.



Metrics over the last hour for two Redis nodes

Viewing a Redis (cluster mode enabled) Cluster's Details (Console)

You can view the details of a Redis (cluster mode enabled) 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 Redis (cluster mode enabled) cluster using the ElastiCache console.

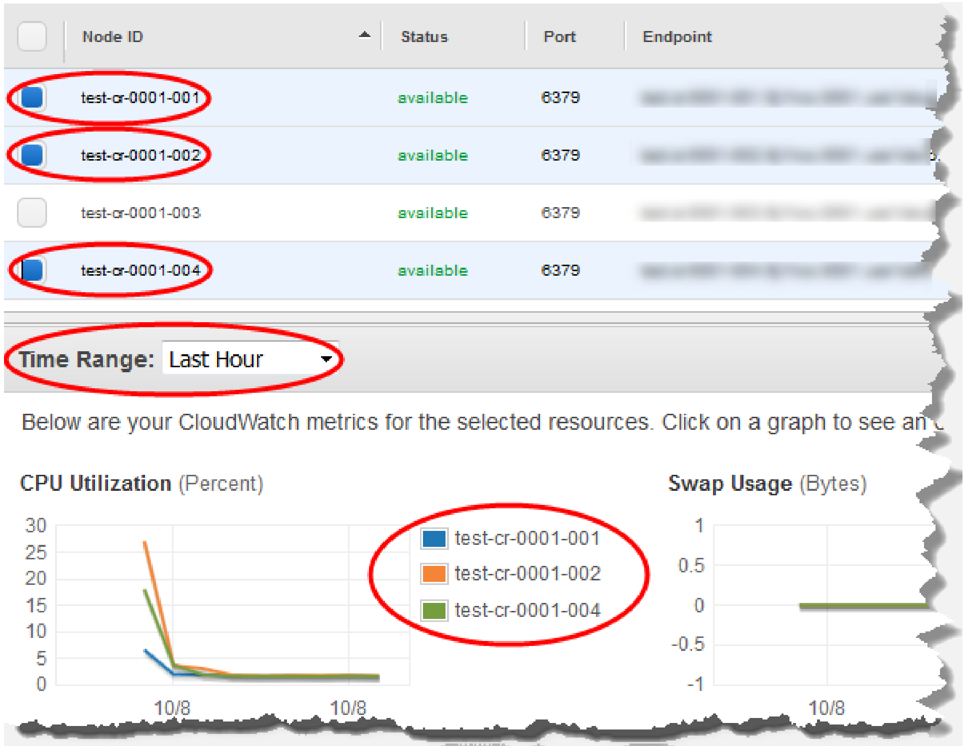
To view a Redis (cluster mode enabled) cluster's details

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the ElastiCache console dashboard, choose **Redis** to display a list of all your clusters that are running any version of Redis.
3. To see details of a Redis (cluster mode enabled) cluster, choose the box to the left of the cluster's name. Make sure you choose a cluster running the Clusterd Redis engine, not just Redis.

   The screen expands below the cluster and display details about the cluster, including the cluster's configuration endpoint.
4. To see a listing of the cluster's shards and the number of nodes in each shard, choose the cluster's name.
5. To view specific information on a node:

   a. Choose the shard's ID.
   b. Choose the **Nodes** tab.

      This will display information about each node, including each node's endpoint that you need to use to read data from the cluster.
   c. To view metrics on one or more nodes, choose the box to the left of the node's 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 Redis 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 `myCluster`.

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

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

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

Use the command `describe-cache-cluster` to display a list of nodes for a cluster, as in the following example, and note the identifiers of the nodes you want to remove.

For Linux, OS X, or Unix:

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

For Windows:

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

```
    --cache-cluster-id my-memcached-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-memcached-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-memcached-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-memcached-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-memcached-cluster",
            "PreferredAvailabilityZone": "us-west-2b",
            "ConfigurationEndpoint": {
                "Port": 11211,
                "Address": "my-memcached-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, go to 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 `myCluster`.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheClusters
   &CacheClusterId=myCluster
   &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, go to 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.

Changes in a cluster's parameters by changing the cluster's parameter group or by changing the value of a parameter in the cluster's parameter group are applied only after the cluster is rebooted.

Modifying a Cluster (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. In the left navigation pane, choose **Redis** or **Memcached**.

   A list of the selected engine's clusters appears.
3. In the list of clusters, choose the name of the cluster you want to modify. The modifications you can make to a Redis (cluster mode enabled) cluster are limited to modifications to security groups, description, parameter groups, backup options, maintenance window, and SNS notifications.
4. Choose **Modify**.

   The **Modify Cluster** window appears.
5. In the **Modify Cluster** window, make the modification(s) you want.

   Important
   You can upgrade to newer engine versions (see Upgrading Engine Versions (p. 41)), but you cannot downgrade to older engine versions except by deleting the existing cluster and creating it anew.

   Because the newer Redis versions provide a better and more stable user experience, Redis versions 2.6.13, 2.8.6, and 2.8.19 are deprecated from the ElastiCache console. While we recommend against it, if you must use one of these older Redis versions, you can use the AWS CLI or ElastiCache API.

   For more information see the following topics:

	AWS AWS CLI	ElastiCache API
Create Cluster	Creating a Cache Cluster (AWS CLI) (p. 137) #	Creating a Cache Cluster (ElastiCache API) (p. 140) #
Modify Cluster	Modifying a Cache Cluster (AWS CLI) (p. 150) #	Modifying a Cache Cluster (ElastiCache API) (p. 150) #
Create Replication Group	Creating a Redis (cluster mode disabled) Replication Group from Scratch (AWS CLI) (p. 210) Creating a Redis (cluster mode enabled) Replication	Creating a Redis (cluster mode disabled) Replication Group from Scratch (ElastiCache API) (p. 212) Creating a Redis (cluster mode enabled) Replication Group

	AWS AWS CLI	ElastiCache API
	Group from Scratch (AWS CLI) (p. 215)	from Scratch (ElastiCache API) (p. 218)
Modify Replication Group	Modifying a Replication Group (AWS CLI) (p. 226) #	Modifying a Replication Group (ElastiCache API) (p. 226) #
# Cannot be used to create a Redis (cluster mode enabled) cluster or replication group.		
# Cannot be used to modify a Redis (cluster mode enabled) cluster or replication group.		

The **Apply Immediately** box applies only to modifications in node type and engine version. If you want to apply any of these changes immediately, select the **Apply Immediately** check box. If this box is not selected, engine version and node type modifications will be applied during the next maintenance window. Other modifications, such as changing the maintenance window, are applied immediately.

6. Choose **Modify**.

Modifying a Cache Cluster (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 `myCluster` and applies the change immediately.

> **Important**
> You can upgrade to newer engine versions (see Upgrading Engine Versions (p. 41)), but you cannot downgrade to older engine versions except by deleting the existing cache cluster or replication group and creating it anew.

For Linux, OS X, or Unix:

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

For Windows:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id myCluster ^
    --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, go to the AWS CLI for ElastiCache topic `modify-cache-cluster`.

Modifying a Cache Cluster (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 `myCluster` and applies the change immediately.

Important
You can upgrade to newer engine versions (see Upgrading Engine Versions (p. 41)), but you cannot downgrade to older engine versions except by deleting the existing cache cluster or replication group and creating it anew.

Line breaks in the following code example are added for ease of reading.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &CacheClusterId=myCluster
    &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, go to 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, changing a parameter value in a parameter group is only applied to the cluster after the cluster is rebooted.

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.

Rebooting a Cluster (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. In the left navigation pane, choose **Memcached** or **Redis**.

   A list of clusters running the selected engine will appear.
3. Choose the cluster to reboot by selecting 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.
4. Choose **Reboot**.

   The reboot cluster confirmation screen appears.
5. 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.

Rebooting a Cache Cluster (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 *myCluster*.

For Linux, OS X, or Unix:

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

For Windows:

```
aws elasticache reboot-cache-cluster ^
    --cache-cluster-id myCluster ^
    --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.

Rebooting a Cache Cluster (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 *myCluster*.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=RebootCacheCluster
   &CacheClusterId=myCluster
   &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.

Monitoring a Cluster's Costs

Cost allocation tags are key-value pairs that you can use to track and manage your AWS costs by grouping expenses on your invoices by the tag values on a resource.

You can 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 one or more services.

For more information on Cost Allocation tags and steps to add or remove them from a cluster, see Monitoring Costs with Cost Allocation Tags (p. 373).

Adding Nodes to a Cluster

Adding nodes to a cluster currently applies only if you are running Memcached or Redis (cluster mode disabled). If you are running Redis (cluster mode disabled), the nodes you add to the cluster are replica nodes.

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

Each time you change the number of nodes in your Memcached cluster, you must re-map at least some of your keyspace so it maps to the correct node. For more detailed information on load balancing your Memcached cluster, see Configuring Your ElastiCache Client for Efficient Load Balancing (p. 66).

Adding Nodes to a Cluster (Console)

The process to add a node to a Memcached or Redis (cluster mode disabled) cluster with replication enabled is the same. If you want to add a node to a single-node Redis (cluster mode disabled) cluster (one without replication enabled), it's a two-step process: first add replication, and then add a replica node.

Topics

The following procedure adds replication to a single-node Redis that does not have replication enabled. When you add replication, the existing node becomes the primary node in the replication-enabled cluster. After replication is added, you can add up to 5 replica nodes to the cluster.

To add replication to a Redis cluster with no shards

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

   A list of clusters running the Redis engine is displayed.
3. Choose the name of a cluster that you want to add to.

   The following is true of a Redis cluster that does not have replication enabled:

   - It is running Redis, not Clustered Redis.

- It has zero shards.

  If the cluster has any shards, replication is already enabled on it and you can continue at To add nodes to a Memcached or Redis (cluster mode disabled) cluster with one shard (console) (p. 155).

4. Choose **Add replication**.
5. In **Add Replication**, type a description for this replication-enabled cluster.
6. Choose **Add**.

   As soon as the cluster's status returns to *available* you can continue at the next procedure and add replicas to the cluster.

To add nodes to a Memcached or Redis (cluster mode disabled) cluster with one shard (console)

The following procedure can be used to add nodes to a Memcached cluster or Redis (cluster mode disabled) cluster which has replication enabled. Currently you cannot add or remove nodes from a Redis (cluster mode enabled) cluster.

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

   A list of clusters running the selected engine appears.
3. From the list of clusters, choose the name of the cluster you want to add a node to. This cannot be a cluster running the Clustered Redis engine or a Redis cluster with zero shards.

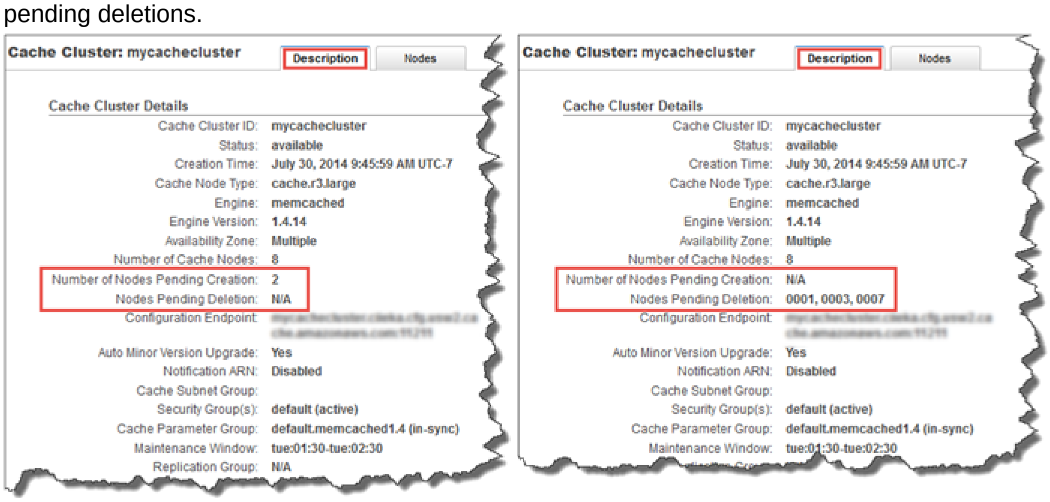
   A list of the cluster's nodes appears and the **Add node** button becomes active.
4. Choose **Add node** at the top of the page.
5. In **Add Node**, complete the information requested in the **Add Node** (Memcached) or **Add Read Replica to Cluster** (Redis) dialog box.
6. Choose the **Apply Immediately - Yes** button to apply this change immediately, or choose **No** to postpone the change until your next maintenance window.

Impact of New Add and Remove Requests on Pending Requests

Scenarios	Pending Operation	New Request	Results
Scenario 1	Delete	Delete	The new delete request, pending or immediate, replaces the pending delete request. For example, if nodes 0001, 0003, and 0007 are pending deletion and a new request to delete nodes 0002 and 0004 is issued, only nodes 0002 and 0004 will be deleted. Nodes 0001, 0003, and 0007 will not be deleted.
Scenario 2	Delete	Create	The new create request, pending or immediate, replaces the pending delete request. For example, if nodes 0001, 0003, and 0007 are pending deletion and a new request to create a node is issued, a new node will be created and nodes 0001, 0003, and 0007 will not be deleted.

Scenarios	Pending Operation	New Request	Results
Scenario 3	Create	Delete	The new delete request, pending or immediate, replaces the pending create request. For example, if there is a pending request to create two nodes and a new request is issued to delete node 0003, no new nodes will be created and node 0003 will be deleted.
Scenario 4	Create	Create	The new create request is added to the pending create request. For example, if there is a pending request to create two nodes and a new request is issued to create three nodes, the new requests is added to the pending request and five nodes will be created. **Important** If the new create request is set to **Apply Immediately - Yes**, all create requests are performed immediately. If the new create request is set to **Apply Immediately - No**, all create requests are pending.

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



7. Choose the **Add** button.

   After a few moments, the new nodes should appear in the nodes list with a status of **creating**. If they don't appear, refresh your browser page.

Adding Nodes to a Cache Cluster (AWS CLI)

If you want to add nodes to an existing Redis (cluster mode disabled) replication group (console: Cluster) that does not have replication enabled, you must first create the replication group specifying the existing cluster as the primary. For more information, see Creating a Replication Group Using an

Available Redis Cache Cluster (AWS CLI) (p. 205). After the replication group is *available*, you can continue with the following process.

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 Removing Nodes from a Cluster (AWS CLI) (p. 161).
- *`--apply-immediately`* or *`--no-apply-immediately`* which specifies whether to add these nodes immediately or at the next maintenance window.

For Linux, OS X, or Unix:

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

For Windows:

```
aws elasticache modify-cache-cluster ^
    --cache-cluster-id my-cache-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-cache-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`.

Adding Nodes to a Cache Cluster (ElastiCache API)

If you want to add nodes to an existing Redis (cluster mode disabled) replication group (console: Cluster) that does not have replication enabled, you must first create the replication group specifying the existing cluster as the Primary. For more information, see Creating a Replication Group Using an Available Redis Cache Cluster (ElastiCache API) (p. 206). After the replication group is *available*, you can continue with the following process.

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 Removing Nodes from a Cluster (ElastiCache API) (p. 164)).
  - *`ApplyImmediately`* Specifies whether to add these nodes immediately or at the next maintenance window.

  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=myCacheCluster
    &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

Removing nodes from a cluster applies only if you are not running the Clustered Redis engine.

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. 66).

Topics

Removing Nodes from a Cluster (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. In the left navigation pane, choose **Memcached** or **Redis**.

   A list of clusters running the selected engine appears.
3. 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.
4. Choose the box to the left of the node ID for the node you want to delete. Using the ElastiCache console, you can only delete one node at a time, so selecting multiple nodes will disable the **Delete node** button.

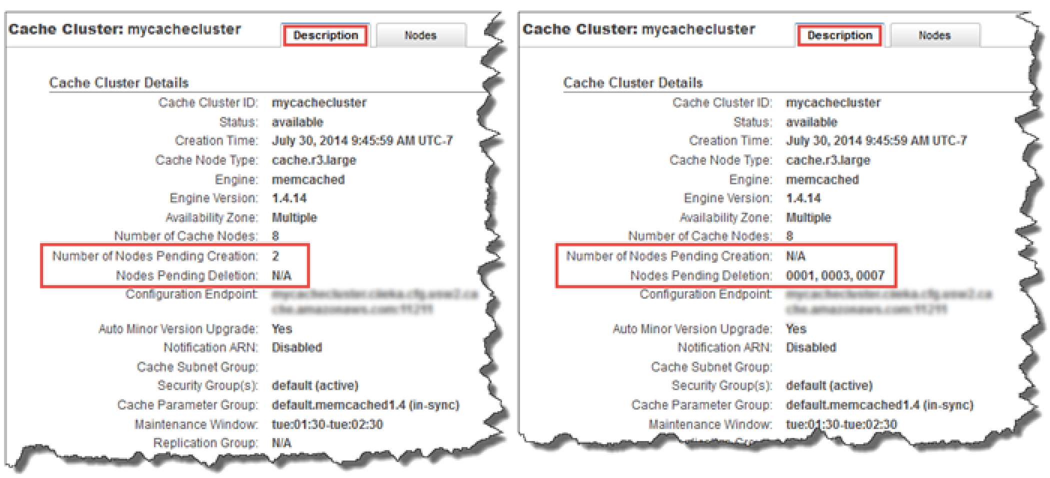
   The *Delete Node* dialog appears.
5. To delete the node, complete the **Delete Node** dialog box and choose **Delete Node**. To not delete the node, choose the **Cancel**.

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, a new node will be created and nodes 0001, 0003, and 0007 will not be deleted.

Scenarios	Pending Operation	New Request	Results
Scenario 3	Create	Delete	The new delete request, pending or immediate, replaces the pending create request. For example, if there is a pending request to create two nodes and a new request is issued to delete node 0003, no new nodes will be created and node 0003 will be deleted.
Scenario 4	Create	Create	The new create request is added to the pending create request. For example, if there is a pending request to create two nodes and a new request is issued to create three nodes, the new requests is added to the pending request and five nodes will be created. **Important** If the new create request is set to **Apply Immediately - Yes**, all create requests are performed immediately. If the new create request is set to **Apply Immediately - No**, all create requests are pending.

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



Removing Nodes from a Cluster (AWS CLI)

1. Use the command `describe-cache-cluster` to display a list of nodes for a cluster, as in the following example, and note the identifiers of the nodes you want to remove.

   For Linux, OS X, or Unix:

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

For Windows:

```
aws elasticache describe-cache-clusters ^
    --cache-cluster-id my-memcached-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-memcached-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-memcached-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-memcached-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-memcached-cluster",
            "PreferredAvailabilityZone": "us-west-2b",
```

```
            "ConfigurationEndpoint": {
                "Port": 11211,
                "Address": "my-memcached-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"
        }
    ]
}
```

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.

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

   For Linux, OS X, or Unix:

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

   For Windows:

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

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


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

Removing Nodes from a Cluster (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.

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

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyCacheCluster
    &CacheClusterId=myCacheCluster
    &ApplyImmediately=true
    &CacheNodeIdsToRemove.member.1=0004
    &CacheNodeIdsToRemove.member.2=0005
    &NumCacheNodes=3
    &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`.

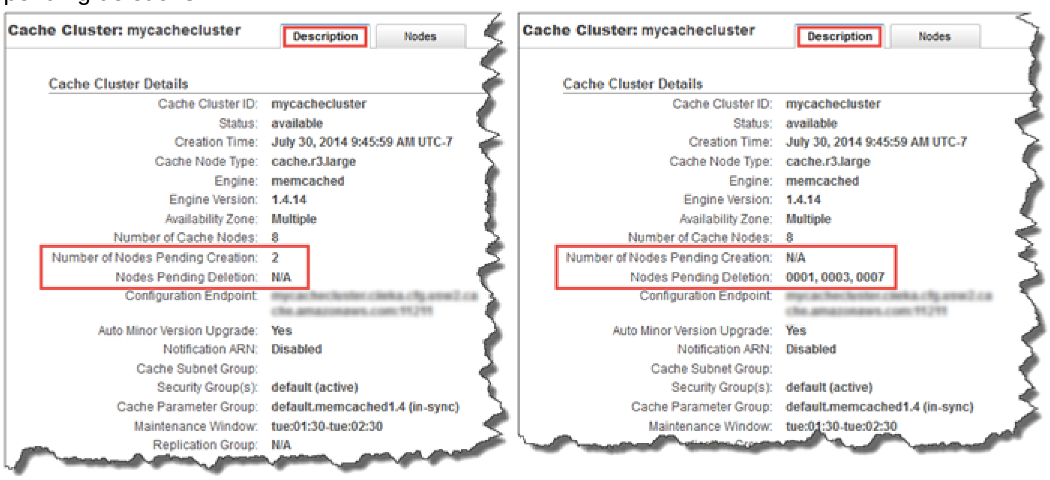
Canceling Pending Add or Delete Node Operations

Canceling Pending Add or Delete Node Operations (Console)

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

To cancel a pending operation

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the left navigation pane, click **Memcached** or **Redis**. A list of clusters running the selected engine will appear.
3. In the list of clusters, choose the name of the cluster that has pending operations you want to cancel.
4. 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.



5. Choose the **Nodes** tab.
6. To cancel all pending operations, click **Cancel Pending**. The **Cancel Pending** dialog box appears.
7. Confirm that you want to cancel all pending operations by selecting the **Cancel Pending** button, or to keep the operations, choose **Cancel**.

Deleting a Cluster

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

Deleting a Cluster (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, either Memcached or Redis.

   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:

   a. If this is a Redis cluster, specify whether or not a final snapshot should be taken, and, if you want a final snapshot, the name of the snapshot.
   b. 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.

Deleting a Cache Cluster (AWS CLI)

The following code deletes the cache cluster `myCluster`.

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

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 more information, go to the AWS CLI for ElastiCache topic `delete-cache-cluster`.

Deleting a Cache Cluster (ElastiCache API)

The following code deletes the cluster `myCluster`.

```
https://elasticache.us-west-2.amazonaws.com/
```

```
?Action=DeleteCacheCluster
&CacheClusterId=myCluster
&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, go to the ElastiCache API reference topic `DeleteCacheCluster`.

Scaling

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.

> **Important**
> Amazon ElastiCache does not support modifying a Redis (cluster mode enabled) replication group. Therefore, the only way to scale your Redis (cluster mode enabled) replication group is to create a new replication group which has the node type, number of node groups, and number of replicas per node group you want to scale to, and then delete the existing replication group. If you are not changing the number of node groups, you can seed the new replication group with data from the existing replication group. For more information, see ElastiCache Backup & Restore (Redis) (p. 235).

Topics

Scaling Memcached

Memcached clusters are comprised of from 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 a 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, you must create a new Memcached cluster which will start out empty unless your application populates it.

Topics

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 a 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. 154)
- Removing Nodes from a Cluster (p. 160)

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. 66).

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, Node Auto Discovery (Memcached) (p. 95). 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 Selecting Your Memcached Node Size (p. 78).

Topics

Scaling Memcached Vertically (Console)

The following procedure walks you through scaling your Memcached 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 Cluster (Console): Memcached (p. 129).
2. In your application, update the endpoints to the new cluster's endpoints. For more information, see Finding the Endpoints for a Memcached Cluster (Console) (p. 49).
3. Delete the old cluster. For more information, see Deleting a Cluster (Console) (p. 167)..

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 Cache Cluster (AWS CLI) (p. 137).
2. In your application, update the endpoints to the new cluster's endpoints. For more information, see Finding Endpoints (AWS CLI) (p. 54).
3. Delete the old cache cluster. For more information, see Deleting a Cache Cluster (AWS CLI) (p. 167).

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 Cache Cluster (ElastiCache API) (p. 140).
2. In your application, update the endpoints to the new cache cluster's endpoints. For more information, see Finding Endpoints (ElastiCache API) (p. 58).

3. Delete the old cache cluster. For more information, see Deleting a Cache Cluster (ElastiCache API) (p. 167).

Scaling Redis (cluster mode disabled) Clusters

Redis (cluster mode disabled) nodes must be large enough to contain all the cache's data plus Redis overhead. To change the data capacity of your Redis (cluster mode disabled) cluster, you must scale vertically; scaling up to a larger node type to increase data capacity, or scaling down to a smaller node type to reduce data capacity.

The ElastiCache scaling up process is designed to make a best effort to retain your existing data and requires successful Redis replication. For Redis (cluster mode disabled) Redis clusters, we recommend that sufficient memory be made available to Redis as described in the topic Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60).

The scaling down process is completely manual and makes no attempt at data retention other than what you do.

You cannot partition your data across multiple Redis (cluster mode disabled) clusters. However, if you only need to increase or decrease your cluster's read capacity, you can create a Redis (cluster mode disabled) cluster with replica nodes and add or remove read replicas. To create a Redis (cluster mode disabled) cluster with replica nodes using your single-node Redis cache cluster as the primary cluster, see Creating a Redis (cluster mode disabled) Cluster (Console) (p. 130).

After you create the cluster with replicas, you can increase read capacity by adding read replicas. Later, if you need to, you can reduce read capacity by removing read replicas. For more information, see Increasing Read Capacity (p. 188) or Decreasing Read Capacity (p. 189).

In addition to being able to scale read capacity, Redis (cluster mode disabled) clusters with replicas provide other business advantages. For more information, see ElastiCache Replication (Redis) (p. 190).

Topics

Scaling Up Single-Node Redis Clusters

When you scale up a single-node Redis cluster, ElastiCache performs the following process, whether you use the ElastiCache console, the AWS CLI, or the ElastiCache API.

1. All reads from and writes to the cache cluster are blocked.
2. A new cache cluster with the new node type is spun up in the same availability zone as the existing cache cluster.
3. The cache data in the existing cache cluster is copied to the new cache cluster. How long this process takes depends upon your node type and how much data is in the cache cluster.
4. Reads and writes are resumed using the new cache cluster. Because the new cache cluster's endpoints are the same as they were for the old cache cluster, you do not need to update the endpoints in your application.
5. ElastiCache deletes the old cache cluster.

Because writes to and reads from your cache cluster are blocked during the scale-up process, you should schedule the scale up for a time of low demand on your cache cluster.

As shown in the following table, your Redis scale-up operation is blocked if you have an engine upgrade scheduled for the next maintenance window. For more information on Maintenance Windows, see Maintenance Window (p. 43).

Blocked Redis operations

Pending Operations	Blocked Operations
Scale up	Immediate engine upgrade
Engine upgrade	Immediate scale up
Scale up and engine upgrade	Immediate scale up
	Immediate engine upgrade

If you have a pending operation that is blocking you, you can do one of the following.

- Schedule your Redis scale-up operation for the next maintenance window by clearing the **Apply immediately** check box (CLI use: `--no-apply-immediately`, API use: `ApplyImmediately=false`).
- Wait until your next maintenance window (or after) to perform your Redis scale up operation.
- Add the Redis engine upgrade to this cache cluster modification with the **Apply Immediately** check box selected (CLI use: `--apply-immediately`, API use: `ApplyImmediately=true`). This unblocks your scale up operation by causing the engine upgrade to be performed immediately.

You can scale up a single-node Redis (cluster mode disabled) cluster using the ElastiCache console, the AWS CLI, or ElastiCache API.

Scaling Up Single-Node Redis Clusters (Console)

The following procedure describes how to scale up a single-node Redis cluster using the ElastiCache Management Console.

To scale up a single-node Redis 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 left navigation pane, choose **Redis**.
3. From the list of clusters, choose the cluster you want to scale up (it must be running the Redis engine, not the Clustered Redis engine).
4. Choose **Modify**.
5. In the **Modify Cluster** wizard, choose the node type you want to scale to from the **Node type** list.

   The list identifies all the node types you can scale up to.
6. If you want to perform the scale up process right away, select the **Apply immediately** box. If the **Apply immediately** box is left unselected, the scale-up process is performed during this cluster's next maintenance window.
7. Choose **Modify**.

   If you selected **Apply immediately** in the previous step, the cluster's status changes to *modifying*. When the status changes to *available*, the modification is complete and you can begin using the new cluster.

Scaling Up Single-Node Redis Cache Clusters (AWS CLI)

The following procedure describes how to scale up a single-node Redis cache cluster using the AWS CLI.

To scale up a single-node Redis cache cluster (AWS CLI)

1. Determine the node types you can scale up to by running the AWS CLI `list-allowed-node-type-modifications` command with the following parameter.

   - *`--cache-cluster-id`* – Name of the single-node Redis cache cluster you want to scale up.

   For Linux, OS X, or Unix:

   ```
   aws elasticache list-allowed-node-type-modifications \
       --cache-cluster-id my-cache-cluster-id
   ```

   For Windows:

   ```
   aws elasticache list-allowed-node-type-modifications ^
       --cache-cluster-id my-cache-cluster-id
   ```

   For more information, see list-allowed-node-type-modifications in the *AWS CLI Reference*.

2. Modify your existing cache cluster specifying the cache cluster to scale up and the new, larger node type, using the AWS CLI `modify-cache-cluster` command and the following parameters.

   - *`--cache-cluster-id`* – The name of the cache cluster you are scaling up.
   - *`--cache-node-type`* – The new node type you want to scale the cache cluster up to. This value must be one of the node types returned by the `list-allowed-node-type-modifications` command in step 1.
   - *`--apply-immediately`* – Causes the scale-up process to be applied immediately. To postpone the scale-up process to the cluster's next maintenance window, use the *`--no-apply-immediately`* parameter.

   For Linux, OS X, or Unix:

   ```
   aws elasticache modify-cache-cluster \
       --cache-cluster-id my-redis-cache-cluster \
       --cache-node-type cache.m2.xlarge \
       --apply-immediately
   ```

   For Windows:

   ```
   aws elasticache modify-cache-cluster ^
       --cache-cluster-id my-redis-cache-cluster ^
       --cache-node-type cache.m2.xlarge ^
       --apply-immediately
   ```

   For more information, see modify-cache-cluster in the *AWS CLI Reference*.

3. If you used the *`--apply-immediately`*, check the status of the new cache cluster using the AWS CLI `describe-cache-clusters` command with the following parameter. When the status changes to *available*, you can begin using the new, larger cache cluster.

   - *`--cache-cache cluster-id`* – The name of your single-node Redis cache cluster. Use this parameter to describe a particular cache cluster rather than all cache clusters.

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

For more information, see describe-cache-clusters in the *AWS CLI Reference*.

Scaling Up Single-Node Redis Cache Clusters (ElastiCache API)

The following procedure describes how to scale up a single-node Redis cache cluster using the ElastiCache API.

To scale up a single-node Redis cache cluster (ElastiCache API)

1. Determine the node types you can scale up to by running the ElastiCache API `ListAllowedNodeTypeModifications` action with the following parameter.

   - *`CacheClusterId`* – The name of the single-node Redis cache cluster you want to scale up.

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

   For more information, see ListAllowedNodeTypeModifications in the *Amazon ElastiCache API Reference*.

2. Modify your existing cache cluster specifying the cache cluster to scale up and the new, larger node type, using the `ModifyCacheCluster` ElastiCache API action and the following parameters.

   - *`CacheClusterId`* – The name of the cache cluster you are scaling up.
   - *`CacheNodeType`* – The new, larger node type you want to scale the cache cluster up to. This value must be one of the node types returned by the `ListAllowedNodeTypeModifications` action in step 1.
   - *`ApplyImmediately`* – Set to `true` to cause the scale-up process to be performed immediately. To postpone the scale-up process to the cluster's next maintenance window, use *`ApplyImmediately`*`=false`.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyCacheCluster
   &ApplyImmediately=true
   &CacheClusterId=MyRedisCacheCluster
   &CacheNodeType=cache.m2.xlarge
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
```

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

For more information, see ModifyCacheCluster in the *Amazon ElastiCache API Reference*.

3. If you used `ApplyImmediately=true`, check the status of the new cache cluster using the ElastiCache API `DescribeCacheClusters` action with the following parameter. When the status changes to *available*, you can begin using the new, larger cache cluster.

   - *`CacheClusterId`* – The name of your single-node Redis cache cluster. Use this parameter to describe a particular cache cluster rather than all cache clusters.

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

For more information, see DescribeCacheClusters in the *Amazon ElastiCache API Reference*.

Scaling Down Single-Node Redis Clusters

The ElastiCache process for scaling your Redis cluster down is completely manual and makes no attempt at data retention other than what you do.

The following sections walks you through how to scale a single-node Redis cluster down to a smaller node type. Ensuring that the new, smaller node type is large enough to accommodate all the data and Redis overhead is important to the long term success of your new Redis cluster. For more information, see Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60).

Topics

Scaling Down a Single-Node Redis Cluster (Console)

The following procedure walks you through scaling your single-node Redis cluster down to a smaller node type using the ElastiCache console.

To scale down your single-node Redis cluster (console)

1. Ensure that the smaller node type is adequate for your data and overhead needs. For more information, see Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60).
2. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
3. Take a snapshot of the cluster. For details on how to take a snapshot, see Creating a Manual Backup (Console) (p. 239).
4. Restore from this snapshot specifying the new node type for the new cluster. For more information, see Restoring From a Backup (Console) (p. 255).

   Alternatively, you can launch a new cluster using the new node type and seeding it from the snapshot. For more information see Using a Backup to Seed a Cluster (p. 257).
5. In your application, update the endpoints to the new cluster's endpoints. For more information, see Finding the Endpoints for a Redis (cluster mode disabled) Cluster (Console) (p. 51).
6. Delete the old cluster. For more information, see Deleting a Cluster (Console) (p. 167).
7. If you no longer need it, delete the snapshot. For more information, see Deleting a Backup (Console) (p. 261).

Tip
If you don't mind your cluster being unavailable while it is being created or restored, you can eliminate the need to update the endpoints in your application by deleting the old cluster right after taking the snapshot and re-using the old cluster's name for the new cluster.

Scaling Down a Single-Node Redis Cache Cluster (AWS CLI)

The following procedure walks you through scaling your single-node Redis cache cluster down to a smaller node type using the AWS CLI.

To scale down a single-node Redis cache cluster (AWS CLI)

1. Ensure that the smaller node type is adequate for your data and overhead needs. For more information, see Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60).

2. Create a snapshot of your existing Redis cache cluster. For instructions, see Creating a Manual Backup (AWS CLI) (p. 240).
3. Restore from the snapshot using the new, smaller node type as the cache cluster's node type. For more information, see Restoring From a Backup (AWS CLI) (p. 255).
4. In your application, update the endpoints to the new cache cluster's endpoints. For more information, see Finding Endpoints for Nodes and Clusters (AWS CLI) (p. 54).
5. Delete your old cache cluster. For more information, see Deleting a Cache Cluster (AWS CLI) (p. 167).
6. If you no longer need it, delete the snapshot. For more information, see Deleting a Backup (AWS CLI) (p. 261).

Tip
If you don't mind your cache cluster being unavailable while it is being created or restored, you can eliminate the need to update the endpoints in your application by deleting the old cache cluster right after taking the snapshot and re-using the old cache cluster's name for the new cache cluster.

Scaling Down a Single-Node Redis (cluster mode disabled) Cache Cluster (ElastiCache API)

The following procedure walks you through scaling your single-node Redis cache cluster down to a smaller node type using the ElastiCache API.

To scale down a single-node Redis cache cluster (ElastiCache API)

1. Ensure that the smaller node type is adequate for your data and overhead needs. For more information, see Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60).
2. Create a snapshot of your existing Redis cache cluster. For instructions, see Creating a Manual Backup (ElastiCache API) (p. 240).
3. Restore from the snapshot using the new, smaller node type as the cache cluster's node type. For more information, see Restoring From a Backup (ElastiCache API) (p. 256).
4. In your application, update the endpoints to the new cache cluster's endpoints. For more information, see Finding Endpoints for Nodes and Clusters (ElastiCache API) (p. 58).
5. Delete your old cache cluster. For more information, see Deleting a Cache Cluster (ElastiCache API) (p. 167).
6. If you no longer need it, delete the snapshot. For more information, see Deleting a Backup (ElastiCache API) (p. 261).

Tip
If you don't mind your cache cluster being unavailable while it is being created or restored, you can eliminate the need to update the endpoints in your application by deleting the old cache cluster right after taking the snapshot and re-using the old cache cluster's name for the new cache cluster.

Scaling Redis Clusters with Replica Nodes

A Redis cluster with replica nodes (called *replication group* in the API/CLI) provides high availability via replication that has Multi-AZ with automatic failover enabled. A cluster with replica nodes is a logical collection of up to 6 Redis clusters where one cluster, the Primary, is able to serve both read and write requests. All the other clusters in the cluster are read-only replicas of the Primary. Data written to the Primary is asynchronously replicated to all the read replicas in the cluster. Because Redis (cluster mode disabled) does not support partitioning your data across multiple clusters, each cluster in a Redis (cluster mode disabled) replication group contains the entire cache dataset. Redis (cluster mode enabled) custers support partitioning your data across up to 15 shards.

Important
Due to the current limitations on Redis (cluster mode enabled), to scale the node type, number of shards (called *node groups* in the API/CLI), or the number of replicas in the shard of a Redis (cluster mode enabled) cluster with replica nodes, you must create a new Redis (cluster mode enabled) cluster with the new node type, number of shards, or number of replicas and then delete the old cluster.

To change the data capacity of your cluster you must scale it up to a larger node type, or down to a smaller node type.

To change the read capacity of your cluster, add more read replicas, up to a maximum of 5, or remove read replicas.

The ElastiCache scaling up process is designed to make a best effort to retain your existing data and requires successful Redis replication. For Redis clusters with replicas, we recommend that sufficient memory be made available to Redis as described in the topic Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60).

The scaling down process is completely manual and makes no attempt at data retention other than what you do.

Related topics

- ElastiCache Replication (Redis) (p. 190)
- Replication: Redis (cluster mode disabled) vs. Redis (cluster mode enabled) (p. 193)
- Replication: Multi-AZ with Automatic Failover (Redis) (p. 196)
- Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60)

Topics

Scaling Up Redis Clusters with Replicas

> **Important**
> Due to the current limitations on Redis (cluster mode enabled), to scale the number of node groups or the node types in the node groups of a Redis (cluster mode enabled) cluster with replicas, you must create a new Redis (cluster mode enabled) cluster with the new number of node groups or node type and then delete the old cluster.

Amazon ElastiCache provides console, CLI, and API support for scaling your Redis (cluster mode disabled) replication group up.

When the scale-up process is initiated, ElastiCache does the following:

1. Blocks all reads from and writes to the primary cluster.
2. Launches a new Redis replication group using the new node type.
3. Copies all the data from the primary cluster to the new primary cluster.
4. Sync the new read replicas with the new primary cluster.
5. Updates the DNS entries so they point to the new clusters. Because of this you don't have to update the endpoints in your application.

   > **Important**
   > Reads from read replica clusters will be interrupted while ElastiCache switches you from your current replicas to the new replicas.
6. Reinstates reads from and writes to the new primary cluster.
7. Deletes the old cluster (called *replication group* in the API/CLI).

How long this process takes is dependent upon your node type and how much data is in your cluster.

As shown in the following table, your Redis scale-up operation is blocked if you have an engine upgrade scheduled for the cluster's next maintenance window.

Blocked Redis operations

Pending Operations	Blocked Operations
Scale up	Immediate engine upgrade
Engine upgrade	Immediate scale up
Scale up and engine upgrade	Immediate scale up
	Immediate engine upgrade

If you have a pending operation that is blocking you, you can do one of the following.

- Schedule your Redis scale-up operation for the next maintenance window by clearing the **Apply immediately** check box (CLI use: `--no-apply-immediately`, API use: `ApplyImmediately=false`).
- Wait until your next maintenance window (or after) to perform your Redis scale-up operation.
- Add the Redis engine upgrade to this cache cluster modification with the **Apply Immediately** check box selected (CLI use: `--apply-immediately`, API use: `ApplyImmediately=true`). This unblocks your scale-up operation by causing the engine upgrade to be performed immediately.

The following sections describe how to scale your Redis cluster with replicas up using the ElastiCache console, the AWS CLI, and the ElastiCache API.

Scaling Up a Redis Cluster with Replicas (Console)

The following process scales your cluster with replicas from its current node type to a new, larger node type using the ElastiCache console. During this process, until the status changes from *modifying* to *available*, all reads and writes between your application and the primary cache cluster are blocked.

The amount of time it takes to scale up to a larger node type varies, depending upon the node type and the amount of data in your current cluster.

To scale up Redis cluster with replicas (console)

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. From the left navigation pane, choose **Redis**
3. From the list of clusters, choose the cluster you want to scale up. This cluster must be running the Redis engine and not the Clustered Redis engine.
4. Choose **Modify**.
5. For **Node Types**, in the **Modify Cluster** dialog box, choose the node type you want to scale up to.
6. If you want to perform the scale-up process right away, select the **Apply immediately** check box. If the **Apply immediately** check box is left unselected, the scale-up process is performed during this cluster's next maintenance window.
7. Choose **Modify**.
8. When the cluster's status changes from *modifying* to *available*, your cluster has scaled to the new node type and you may resume using it. There is no need to update the endpoints in your application.

Scaling Up a Redis Replication Group (AWS CLI)

The following process scales your replication group (console: cluster with replicas) from its current node type to a new, larger node type using the AWS CLI. During this process, until the status changes from *modifying* to *available*, all reads and writes between your application and the primary cache cluster are blocked.

The amount of time it takes to scale up to a larger node type varies, depending upon your node type and the amount of data in your current cache cluster.

To scale up a Redis Replication Group (AWS CLI)

1. Determine which node types you can scale up to by running the AWS CLI `list-allowed-node-type-modifications` command with the following parameter.

   - *`--replication-group-id`* – the name of the replication group. Use this parameter to describe a particular replication group rather than all replication groups.

   For Linux, OS X, or Unix:

   ```
   aws elasticache list-allowed-node-type-modifications \
       --replication-group-id my-replication-group
   ```

   For Windows:

   ```
   aws elasticache list-allowed-node-type-modifications ^
       --replication-group-id my-replication-group
   ```

For more information, see list-allowed-node-type-modifications in the *AWS CLI Reference*.

2. Scale your current replication group up to the new node type using the AWS CLI `modify-replication-group` command with the following parameters.

   - *`--replication-group-id`* – the name of the replication group.
   - *`--cache-node-type`* – the new, larger node type of the cache clusters in this replication group. This value must be one of the instance types returned by the `list-allowed-node-type-modifications` command in step 1.
   - *`--apply-immediately`* – Causes the scale-up process to be applied immediately. To postpone the scale-up operation to the next maintenance window, use *`--no-apply-immediately`*.

   For Linux, OS X, or Unix:

   ```
   aws elasticache modify-replication-group \
       --replication-group-id my-replication-group \
       --cache-node-type cache.m3.large \
       --apply-immediately
   ```

   For Windows:

   ```
   aws elasticache modify-replication-group ^
       --replication-group-id my-replication-group ^
       --cache-node-type cache.m3.large ^
       --apply-immediately
   ```

   For more information, see modify-replication-group in the *AWS CLI Reference*.

3. If you used the *`--apply-immediately`* parameter, monitor the status of the replication group using the AWS CLI `describe-replication-group` command with the following parameter. When the status changes from *modifying* to *available*, you can begin writing to your new, scaled up replication group.

   - *`--replication-group-id`* – the name of the replication group. Use this parameter to describe a particular replication group rather than all replication groups.

   For Linux, OS X, or Unix:

   ```
   aws elasticache describe-replication-group \
       --replication-group-id my-replication-group
   ```

   For Windows:

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

   For more information, see describe-replication-groups in the *AWS CLI Reference*.

Scaling Up a Redis Replication Group (ElastiCache API)

The following process scales your replication group from its current node type to a new, larger node type using the ElastiCache API. During this process, until the status changes from *modifying* to *available*, all reads and writes between your application and the primary cache cluster are blocked. However, reads from the read replica cache clusters continue uninterrupted.

The amount of time it takes to scale up to a larger node type varies, depending upon your node type and the amount of data in your current cache cluster.

To scale up a Redis Replication Group (ElastiCache API)

1. Determine which node types you can scale up to using the ElastiCache API `ListAllowedNodeTypeModifications` action with the following parameter.

   - *`ReplicationGroupId`* – the name of the replication group. Use this parameter to describe a specific replication group rather than all replication groups.

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

   For more information, see ListAllowedNodeTypeModifications in the *Amazon ElastiCache API Reference*.
2. Scale your current replication group up to the new node type using the `ModifyRedplicationGroup` ElastiCache API action and with the following parameters.

   - *`ReplicationGroupId`* – the name of the replication group.
   - *`CacheNodeType`* – the new, larger node type of the cache clusters in this replication group. This value must be one of the instance types returned by the `ListAllowedNodeTypeModifications` action in step 1.
   - *`ApplyImmediately`* – Set to `true` to causes the scale-up process to be applied immediately. To postpone the scale-up process to the next maintenance window, use *`ApplyImmediately`*`=false`.

   ```
   https://elasticache.us-west-2.amazonaws.com/
      ?Action=ModifyReplicationGroup
      &ApplyImmediately=true
      &CacheNodeType=cache.m3.large
      &ReplicationGroupId=myReplicationGroup
      &SignatureVersion=4
      &SignatureMethod=HmacSHA256
      &Timestamp=20141201T220302Z
      &Version=2014-12-01
      &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 ModifyReplicationGroup in the *Amazon ElastiCache API Reference*.
3. If you used *`ApplyImmediately`*`=true`, monitor the status of the replication group using the ElastiCache API `DescribeReplicationGroups` action with the following parameters. When

the status changes from *modifying* to *available*, you can begin writing to your new, scaled up replication group.

- *ReplicationGroupId* – the name of the replication group. Use this parameter to describe a particular replication group rather than all replication groups.

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

For more information, see DescribeReplicationGroups in the *Amazon ElastiCache API Reference*.

Scaling Down Redis Clusters with Replicas

The following sections walks you through how to scale a Redis (cluster mode disabled) cache cluster with replica nodes down to a smaller node type. Ensuring that the new, smaller node type is large enough to accommodate all the data and overhead is very important to success. For more information, see Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60).

Topics

Scaling Down a Redis Cluster with Replicas (Console)

The following process scales your Redis cluster with replica nodes to a smaller node type using the ElastiCache console.

To scale down a Redis cluster with replica nodes (console)

1. Ensure that the smaller node type is adequate for your data and overhead needs. For more information, see Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60).
2. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
3. Take a snapshot of the cluster’s primary node. For details on how to take a snapshot, see Creating a Manual Backup (Console) (p. 239).
4. Restore from this snapshot specifying the new node type for the new cluster. For more information, see Restoring From a Backup (Console) (p. 255).

   Alternatively, you can launch a new cluster using the new node type and seeding it from the snapshot. For more information see Using a Backup to Seed a Cluster (p. 257).
5. In your application, update the endpoints to the new cluster's endpoints. For more information, see Finding the Endpoints for a Redis (cluster mode disabled) Cluster (Console) (p. 51).
6. Delete the old cluster. For more information, see Deleting a Replication Group (Console) (p. 228).
7. If you no longer need it, delete the snapshot. For more information, see Deleting a Backup (Console) (p. 261).

Tip
If you don't mind being unable to use your replication group while it is being created or restored, you can eliminate the need to update the endpoints in your application by deleting the old cluster right after taking the snapshot and re-using the old cluster's name for the new cluster.

Scaling Down a Redis Replication Group (AWS CLI)

The following process scales your Redis replication group to a smaller node type using the AWS CLI.

To scale down a Redis replication group (AWS CLI)

1. Ensure that the smaller node type is adequate for your data and overhead needs. For more information, see Selecting Your Redis Node Size (p. 79).
2. Create a snapshot of your existing Redis node. For instructions, see Creating a Manual Backup (AWS CLI) (p. 240).

3. Restore from the snapshot using the new, smaller node type as the new node type. For more information, see Restoring From a Backup (AWS CLI) (p. 255).
4. In your application, update the endpoints to the new cache cluster's endpoints. For more information, see Finding the Endpoints for Replication Groups (AWS CLI) (p. 56).
5. Delete your old replication group. For more information, see Deleting a Replication Group (AWS CLI) (p. 228).
6. If you no longer need it, delete the snapshot. For more information, see Deleting a Backup (AWS CLI) (p. 261).

Tip
If you don't mind being unable to use your replication group while it is being created or restored, you can eliminate the need to update the endpoints in your application by deleting the old replication group right after taking the snapshot and re-using the old replication group's name for the new replication group.

Scaling Down a Redis Replication Group (ElastiCache API)

The following process scales your Redis replication group to a smaller node type using the ElastiCache API.

To scale down a Redis replication group (ElastiCache API)

1. Ensure that the smaller node type is adequate for your data and overhead needs. For more information, see Selecting Your Redis Node Size (p. 79).
2. Create a snapshot of your existing Redis cache cluster. For instructions, see Creating a Manual Backup (ElastiCache API) (p. 240).
3. Restore from the snapshot using the new, smaller node type as the new node type. For more information, see Restoring From a Backup (ElastiCache API) (p. 256).
4. In your application, update the endpoints to the new cache cluster's endpoints. For more information, see Finding Endpoints (ElastiCache API) (p. 58).
5. Delete your old replication group. For more information, see Deleting a Replication Group (ElastiCache API) (p. 228).
6. If you no longer need it, delete the snapshot. For more information, see Deleting a Backup (ElastiCache API) (p. 261).

Tip
If you don't mind being unable to use your replication group while it is being created or restored, you can eliminate the need to update the endpoints in your application by deleting the old replication group right after taking the snapshot and re-using the old replication group's name for the new replication group.

Increasing Read Capacity

To increase read capacity, add read replicas (up to a maximum of five) to your Redis replication group.

> **Important**
> Due to the current limitations on Redis (cluster mode enabled), to scale the number of replicas in the shards of a Redis (cluster mode enabled) cluster, you must create a new Redis (cluster mode enabled) cluster with the new number of replicas and then delete the old cluster.

You can scale your Redis cluster's read capacity using the ElastiCache console, the AWS CLI, or the ElastiCache API. For more information, see Adding a Read Replica to a Redis Cluster (p. 229).

Decreasing Read Capacity

To decrease read capacity, delete one or more read replicas from your Redis cluster with replicas (called *replication group* in the API/CLI). If the cluster is Multi-AZ with automatic failover enabled, you cannot delete the last read replica without first disabling Multi-AZ with automatic failover. For more information, see Modifying a Cluster with Replicas (p. 226).

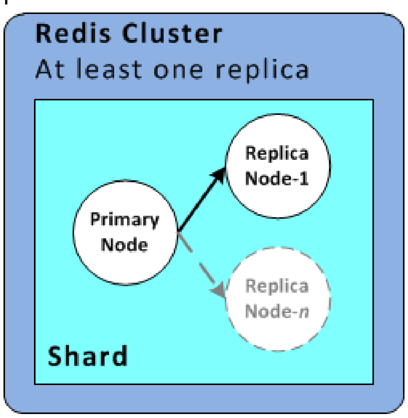
Important
Due to the current limitations on Redis (cluster mode enabled), to scale the number of replicas in the node groups of a Redis (cluster mode enabled) cluster with replicas, you must create a new Redis (cluster mode enabled) cluster with the new number of replicas and then delete the old replication group.

For more information, see Deleting a Read Replica (p. 234).

ElastiCache Replication (Redis)

Single-node Amazon ElastiCache Redis clusters are in-memory entities with limited data protection services (AOF). if your cluster fails for any reason, you risk lose all the cluster's data. However, if you're running the Redis engine, you can group 2 to 6 nodes into a cluster with replicas where 1 to 5 read-only clusters contain replicate data of the group's single read/write primary. In this scenario, if one cluster fails for any reason you do not lose all your data since it is replicated in one or more other clusters.

As seen in the following graphic, the replication structure is contained within a shard (called *node group* in the API/CLI) which is contained within a Redis cluster. Redis (cluster mode disabled) clusters always have one shard. Redis (cluster mode enabled) clusters can have up to 15 shards with the cluster's data partitioned across the shards.



Redis cluster with replica nodes

If the cluster with replicas has Multi-AZ with automatic failover enabled and the primary fails, the primary fails over to a read replica. Because the data is updated on the replica nodes asynchronously, there may be some data loss due to latency in updating the replica nodes. For more information, see Mitigating Failures when Running Redis (p. 64).

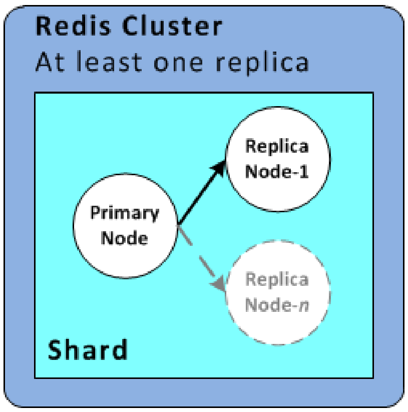
Topics

Redis Replication

Redis now implements replication in two ways: 1) Redis (cluster mode disabled) with a single shard that contains all of the cluster's data in each node, and 2) Redis (cluster mode enabled) with data partitioned across up to 15 shards.

Redis (cluster mode disabled)

A Redis (cluster mode disabled) cluster is a collection of Redis nodes, with one primary read-write node and up to five secondary, read-only nodes, which are called *read replicas*. Each read replica maintains a copy of the data from the cluster's primary node. Asynchronous replication mechanisms are used to keep the read-replicas synchronized with the primary. Applications can read from any cluster in the cluster. Applications can write only to the primary node. Read replicas improve read throughput and guard against data loss.



Redis cluster with replica nodes

You can use Redis (cluster mode disabled) clusters with replicas to scale your Redis solution for ElastiCache to handle applications that are highly read-intensive or to support large numbers of clients that simultaneously read from the same cluster.

All of the nodes in a Redis (cluster mode disabled) cluster must reside in the same region. To improve fault tolerance, you can provision read replicas in multiple Availability Zones within that region. When you add a read replica to a cluster, all of the data from the primary is copied to the read replica. From that point, whenever data is written to the primary, the changes are asynchronously propagated to all the read replicas. Your applications can connect to a read replica and access data in the cluster, although they cannot write any data to a replica.

To improve fault tolerance and reduce write down time, implement Multi-AZ with automatic failover for your Redis cluster with replicas. For more information, see Replication: Multi-AZ with Automatic Failover (Redis) (p. 196).

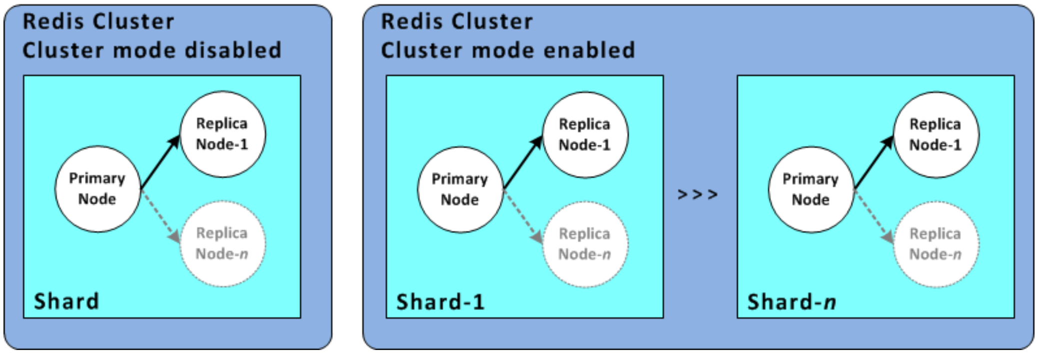
You can change the roles of the nodes within the cluster, with the primary and one of the replicas exchanging roles. You might decide to do this for performance tuning reasons. For example, with a web application that has heavy write activity, you can choose the node that has the lowest network latency. For more information, see Promoting a Read-Replica to Primary (p. 231).

Redis (cluster mode enabled)

A Redis (cluster mode enabled) cluster is comprised of from 1 to 15 shards. Each shard has a primary node and up to five secondary, read-only nodes, which are called *read replicas*. Each read replica maintains a copy of the data from the primary. Asynchronous replication mechanisms are used to keep the read-replicas synchronized with the primary. Applications can read from any node in the cluster.

Applications can write only to the primary nodes. Read replicas enhance read scalability and guard against data loss. Data is partitioned across the shards in a Redis (cluster mode enabled) cluster.

Applications use the Redis (cluster mode enabled) use clusters *configuration endpoint* to connect with the nodes in the cluster. For more information, see Finding Your ElastiCache Endpoints (p. 48).



Redis (cluster mode enabled) cluster with replica nodes

All of the nodes in a cluster must reside in the same region. To improve fault tolerance, you can provision both primaries and read replicas in multiple Availability Zones within that region. Your applications can connect to a read replica and access data in the cluster, although they cannot write any data to a replica.

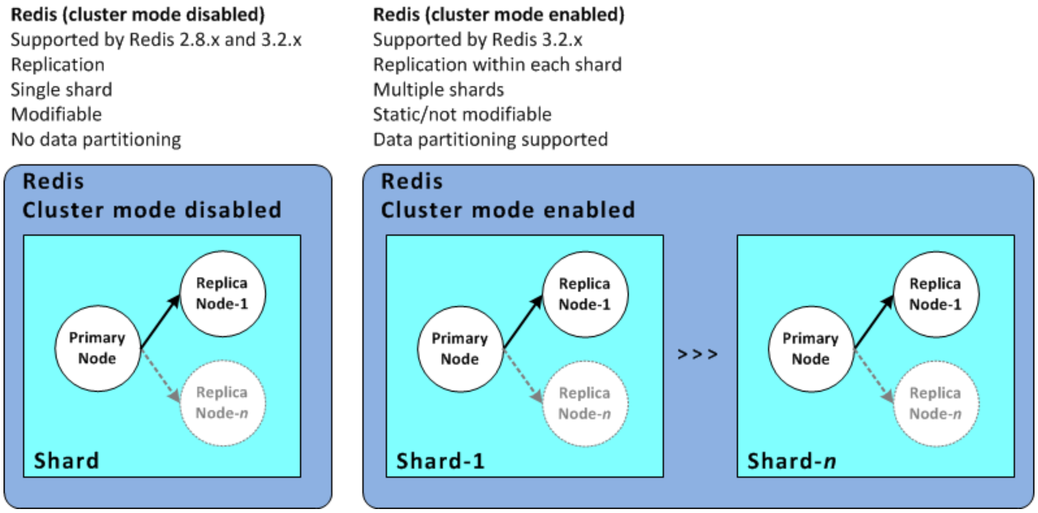
Multi-AZ with automatic failover is required for Redis (cluster mode enabled) cluster. For more information, see Replication: Multi-AZ with Automatic Failover (Redis) (p. 196).

Currently, in Redis (cluster mode enabled), there are some limitations.

- You cannot promote any of the replica nodes to primary.
- Multi-AZ with automatic failover is required.
- The structure of the cluster, node type, number of shards, and number of nodes, is fixed at the time of creation and cannot be changed.

Replication: Redis (cluster mode disabled) vs. Redis (cluster mode enabled)

Beginning with Redis version 3.2, you have the ability to create one of two distinct types of Redis clusters. A Redis (cluster mode disabled) cluster always has a single shard with up to 5 read replica nodes. A Redis (cluster mode enabled) cluster has up to 15 shards with 1 to 5 read replica nodes in each.



Redis (cluster mode disabled) and Redis (cluster mode enabled) clusters

The following table summarizes important differences between Redis (cluster mode disabled) and Redis (cluster mode enabled) clusters.

Comparing Redis (cluster mode disabled) and Redis (cluster mode enabled) Clusters

Feature	Redis (cluster mode disabled)	Redis (cluster mode enabled)
Modifiable	Yes. Supports adding and deleting replica nodes, and scaling up node type.	No. Currently does not support making changes after the cluster is created as long as cluster mode is enabled.
Data Partitioning	No	Yes
Shards	1	1 to 15 The number of node groups is set when the replication group is created.
Read replicas	0 to 5	0 to 5 per shard. **Important** If you have no replicas and the node fails, you will experience data loss.
Multi-AZ with Automatic Failover	Yes, with at least 1 replica. Optional, on by default.	Yes. Required.
Snapshots (Backups)	Yes, creating a single .rdb file.	Yes, creating a unique .rdb file for each shard.
Restore	Yes, using a single .rdb file.	Yes, using a unique .rdb file for each shard.
Supported by	All Redis versions	Redis 3.2 and following
Engine upgradeable	Yes	N/A

Which should I choose?

When choosing between Redis (cluster mode disabled) or Redis (cluster mode enabled) you should consider the following factors:

- **Scaling v. Partitioning** – Business needs change. You need to either provision for peak demand or scale as demand changes. Redis (cluster mode disabled) supports scaling. You can scale read capacity by adding or deleting replica nodes, or you can scale capacity by scaling up to a larger node type. Both of these operations take time. For more information, see Scaling Redis Clusters with Replica Nodes (p. 180).

  Redis (cluster mode enabled) currently does not support scaling, but does support partitioning your data across up to 15 node groups. One advantage of partitioning is that you spread your load over a greater number of endpoints which reduces access bottle necks during peak demand. Additionally, you can accommodate a larger data set since the data can be spread across multiple servers. Redis (cluster mode enabled) does not support changing the number of partitions in your replication group once it is created.

- **Node size v. Number of nodes** – Because a Redis (cluster mode disabled) cluster has only one shard, the node type must be large enough to accommodate all the cluster's data plus necessary overhead. On the other hand, because you can partition your data across several shards when using a Redis (cluster mode enabled) cluster, the node types can be smaller, though you will need more of them.

- **Reads v. Writes** – If the primary load on your cluster is applications reading data, you can scale a Redis (cluster mode disabled) cluster by adding and deleting read replicas, though you should note that there is a maximum of 5 read replicas. If the load on your cluster is write heavy, you can benefit from the additional write endpoints of a Redis (cluster mode enabled) cluster with multiple shards.

Whichever type of cluster you choose to implement, be sure to choose a node type that is adequate for your current and future needs. For more information, see Selecting Your Redis Node Size (p. 79).

Replication: Multi-AZ with Automatic Failover (Redis)

Enabling Amazon ElastiCache's Multi-AZ with Automatic Failover functionality on your Redis cluster (API/CLI: replication group) improves your fault tolerance in those cases where your cluster's read/write primary cluster becomes unreachable or fails for any reason.

Topics

Automatic Failover Overview

An ElastiCache cluster consists of a primary node and up to five read replica nodes. During certain types of planned maintenance, or in the unlikely event of a primary node or Availability Zone failure, if your cluster is Multi-AZ enabled, ElastiCache will automatically detect the primary node's failure, select a read replica node and promote it to primary so that you can resume writing to the new primary as soon as promotion is complete. ElastiCache also propagates the DNS of the promoted replica so that if your application is writing to the primary endpoint, no endpoint change will be required in your application. However, because you read from individual endpoints, you will need to change the read endpoint of the replica promoted to primary to the new replica's endpoint.

The promotion process generally takes just a few minutes, which is much faster than recreating and provisioning a new primary if you do not enable Multi-AZ.

You can enable Multi-AZ with Automatic Failover using the ElastiCache console, the AWS CLI, or the ElastiCache API.

Notes on Redis Multi-AZ with Automatic Failover

The following points should be noted:

- Multi-AZ with Automatic Failover is supported on Redis version 2.8.6 and later.
- Redis Multi-AZ with Automatic Failover is not supported on t1 and t2 cache node types.
- Redis replication is asynchronous. Therefore, when a primary cluster fails over to a replica, a small amount of data might be lost due to replication lag.
- When selecting the replica to promote to primary, ElastiCache selects the replica with the least replication lag (that is, the one that is most current).
- When you enable Multi-AZ with Automatic Failover on a cluster, a replica node cannot be manually promoted to primary cluster. Thus, if the primary in AZ-a fails over to a replica in AZ-b, the primary stays in AZ-b. To promote the new replica cluster in AZ-a to primary, you must first disable Multi-AZ with Automatic Failover on the cluster, do the promotion, and then re-enable Multi-AZ with Automatic Failover.
- ElastiCache Multi-AZ with Automatic Failover and append-only file (AOF) are mutually exclusive. If you enable one, you cannot enable the other.
- In the case where a node's failure is caused by the rare event of an entire Availability Zone failing, the replica replacing the failed primary is created only when the Availability Zone is back up. For example, consider a replication group with the primary in AZ-a and replicas in AZ-b and AZ-c. If the primary fails, the replica with the least replication lag is promoted to primary cluster. Then,

ElastiCache creates a new replica in AZ-a (where the failed primary was located) only when AZ-a is back up and available.

- A customer-initiated reboot of a primary does not trigger Automatic Failover. Other reboots and failures do trigger Automatic Failover.
- Whenever the primary is rebooted, it is cleared of data when it comes back online. When the read replicas see the cleared primary cluster, they clear their copy of the data, which causes data loss.
- After a read replica has been promoted, the other replicas sync with the new primary. After the initial sync, the replicas' content is deleted and they sync the data from the new primary, causing a brief interruption during which the replicas are not accessible. This sync process also causes a temporary load increase on the primary while syncing with the replicas. This behavior is native to Redis and isn't unique to ElastiCache Multi-AZ. For details regarding this Redis behavior, see http://redis.io/topics/replication.

Important

- **Redis version 2.8.22 and later**

  External replicas are not permitted.

- **Redis versions prior to 2.8.22**

  We recommend that you do not connect an external Redis replica to an ElastiCache Redis cluster that is Multi-AZ with Automatic Failover enabled. This is an unsupported configuration that can create issues that prevent ElastiCache from properly performing failover and recovery. If you need to connect an external Redis replica to an ElastiCache cluster, make sure that Multi-AZ with Automatic Failover is disabled before you make the connection.

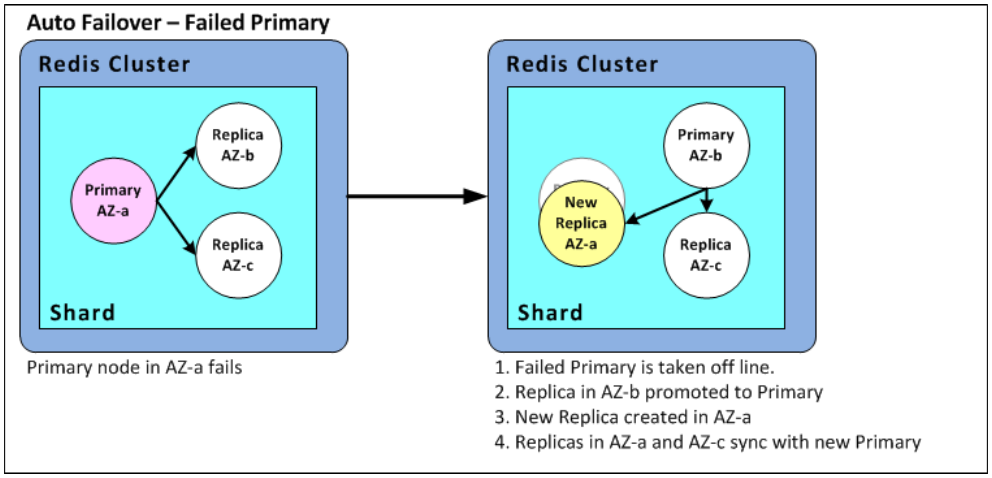
Failure Scenarios with Multi-AZ and Automatic Failover Responses

Prior to the introduction of Multi-AZ with Automatic Failover, ElastiCache detected and replaced a cluster's failed nodes by recreating and re-provisioning the failed node. By enabling Multi-AZ with Automatic Failover, a failed primary node fails over to the replica with the least replication lag. The selected replica is automatically promoted to primary, which is much faster than creating and reprovisioning a new primary node. This process usually takes just a few minutes until you can write to the cluster again.

When Multi-AZ with Automatic Failover is enabled, ElastiCache continually monitors the state of the primary node. If the primary node fails, one of the following actions is performed.

When only the primary node fails

If only the primary node fails, the read replica with the least replication lag is promoted to primary, and a new read replica is created and provisioned to replace the promoted read replica.



Automatic Failover for a failed primary node

ElastiCache Multi-AZ with Automatic Failover Actions when only the primary node fails

1. The failed primary node is taken off line.
2. The read replica with the least replication lag is promoted to primary.

   Writes can resume as soon as the promotion process is complete, typically just a few minutes. If your application is writing to the Primary Endpoint, there is no need to change the endpoint for writes as ElastiCache propagates the DNS of the promoted replica.
3. A replacement read replica is launched and provisioned.

   The replacement read replica is launched in the Availability Zone that the failed primary node was in so that the distribution of nodes is maintained.
4. The replicas sync with the new primary node.

You need to make the following changes to your application after the new replica is available:

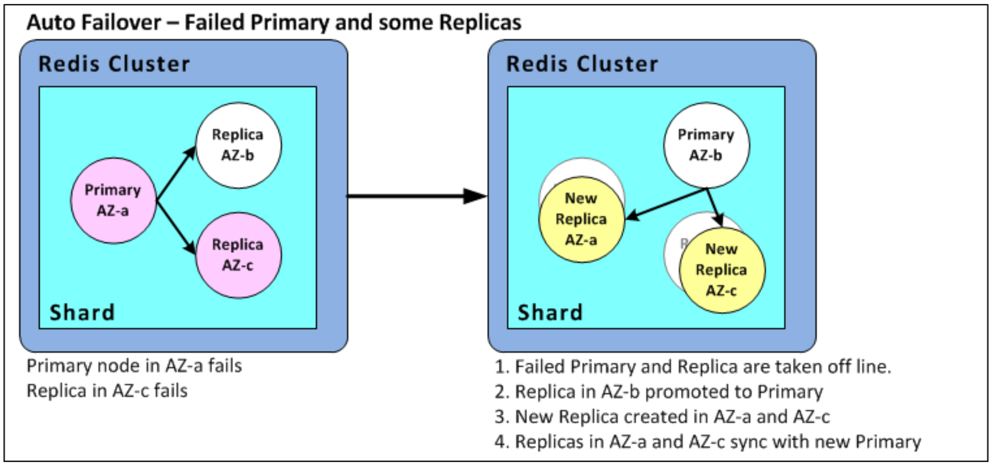
- Primary endpoint–Do not make any changes to your application since the DNS of the new primary node is propagated to the primary endpoint.
- Read endpoint–Replace the read endpoint of the failed primary with the read endpoint of the new replica.

For information about finding the endpoints of a cluster, see the following topics:

- Finding the Endpoints for a Redis (cluster mode disabled) Cluster (Console) (p. 51)
- Finding the Endpoints for Replication Groups (AWS CLI) (p. 56)
- Finding Endpoints for Replication Groups (ElastiCache API) (p. 58)

When the primary node and some read replicas fail

If everything fails except one read replica, the remaining available replica is promoted to primary cluster and new read replicas are created and provisioned.



Automatic Failover for a failed primary node and read replica

ElastiCache Multi-AZ Actions when the primary node and some read replicas fail

1. The failed primary node and failed read replicas are taken off line.
2. The available replica with the least replication lag is promoted to primary node.

   Writes can resume as soon as the promotion process is complete, typically just a few minutes. If your application is writing to the Primary Endpoint, there is no need to change the endpoint for writes as ElastiCache propagates the DNS of the promoted replica.
3. Replacement replicas are created and provisioned.

   The replacement replicas are created in the Availability Zones of the failed nodes so that the distribution of nodes is maintained.
4. All clusters sync with the new primary node.

You need to make the following changes to your application after the new nodes are available:

- Primary endpoint–Do not make any changes to your application since the DNS of the new primary node is propagated to the primary endpoint.
- Read endpoint–Replace the read endpoint of the failed primary and failed replicas with the node endpoints of the new replicas.

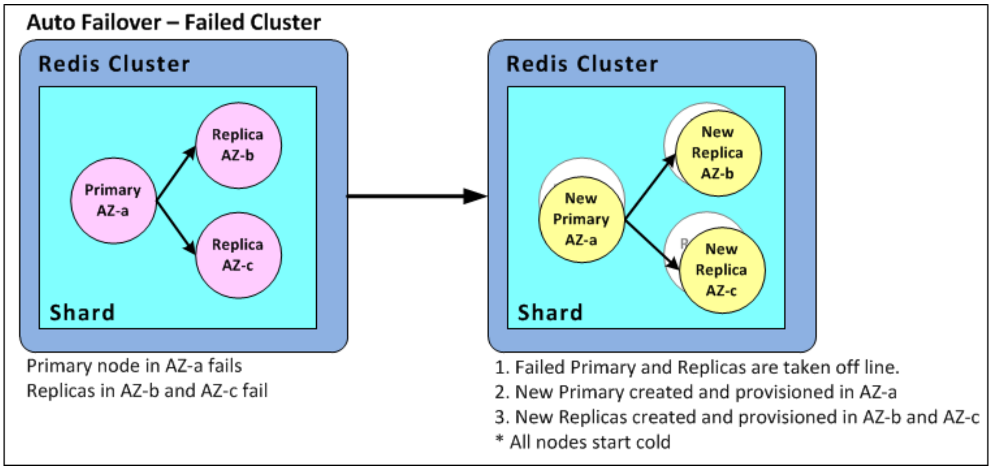
For information about finding the endpoints of a replication group, see the following topics:

- Finding the Endpoints for a Redis (cluster mode disabled) Cluster (Console) (p. 51)
- Finding the Endpoints for Replication Groups (AWS CLI) (p. 56)
- Finding Endpoints for Replication Groups (ElastiCache API) (p. 58)

When the entire cluster fails

If everything fails, all the nodes are recreated and provisioned.

In this scenario, all the data in the cluster is lost due to the failure of every node in the cluster. This is a rare occurrence.



Automatic Failover for a failed cluster

ElastiCache Multi-AZ Actions when the entire cluster fails

1. The failed primary node and read replicas are taken off line.
2. A replacement primary node is created and provisioned.
3. Replacement replicas are created and provisioned.

   The replacements are created in the Availability Zones of the failed nodes so that the distribution of nodes is maintained.

   Note
   Because the entire cluster failed, data is lost and all the new nodes start cold.

Because each of the replacement nodes will have the same endpoint as the node it is replacing, there is no need for you to make any endpoint changes in your application.

For information about finding the endpoints of a replication group, see the following topics:

- Finding the Endpoints for a Redis (cluster mode disabled) Cluster (Console) (p. 51)
- Finding the Endpoints for Replication Groups (AWS CLI) (p. 56)
- Finding Endpoints for Replication Groups (ElastiCache API) (p. 58)

We recommend that you create the primary node and read replicas in different Availability Zones to raise your fault tolerance level.

Enabling Multi-AZ with Automatic Failover

You can enable Multi-AZ with Automatic Failover when you create or modify a cluster (API/CLI: replication group) using the AWS console, AWS CLI, or the ElastiCache API.

Multi-AZ with Automatic Failover can only be enabled on Redis clusters that have at least one available read replica. For information about creating a cluster with replication, see Creating a Redis Cluster with Replicas (p. 204). For information about adding a read replica to a cluster with replication, see Adding a Read Replica to a Redis Cluster (p. 229).

Topics

Enabling Multi-AZ with Automatic Failover (Console)

You can enable Multi-AZ with Automatic Failover using the ElastiCache console when you create a new Redis cluster or by modifying an existing Redis cluster with replication.

Enabling Multi-AZ with Automatic Failover When Creating a Cluster Using the ElastiCache Console

See the topic Creating a Redis (cluster mode disabled) Cluster (Console) (p. 130). Be sure to have one or more replicas and enable Multi-AZ with Automatic Failover.

Enabling Multi-AZ with Automatic Failover on an Existing Cluster (Console)

See the topic Modifying a Cluster (Console) (p. 149).

Enabling Multi-AZ with Automatic Failover (AWS CLI)

The following code example uses the AWS CLI to enable Multi-AZ with Automatic Failover for the replication group *myReplGroup*.

> **Important**
> The replication group *myReplGroup* must already exist and have at least one available read replica.

For Linux, OS X, or Unix:

```
aws elasticache modify-replication-group \
    --replication-group-id myReplGroup \
    --automatic-failover-enabled
```

For Windows:

```
aws elasticache modify-replication-group ^
    --replication-group-id myReplGroup ^
    --automatic-failover-enabled
```

For more information, see the AWS CLI topics, create-cache-cluster, create-replication-group, and modify-replication-group.

Enabling Multi-AZ with Automatic Failover (ElastiCache API)

The following code example uses the ElastiCache API to enable Multi-AZ with Automatic Failover for the replication group *myReplGroup*.

> **Note**
> The replication group *myReplGroup* must already exist and have at least one available read replica.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=ModifyReplicationGroup
    &AutoFailover=true
    &ReplicationGroupId=myReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20140401T192317Z
```

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

For more information, see the ElastiCache API reference for CreateCacheCluster, CreateReplicationGroup, and ModifyReplicationGroup.

How Synchronization and Backup are Implemented

All supported versions of Redis support backup and synchronization between the primary and replica clusters. However, the way that backup and synchronization is implemented varies depending on the Redis version.

Redis Version 2.8.22 and Later

Redis replication, in versions 2.8.22 and later, choose between two methods. For more information, see Redis Versions Prior to 2.8.22 (p. 203) and ElastiCache Backup & Restore (Redis) (p. 235).

During the forkless process, if the write loads are heavy, writes to the cluster are delayed to ensure that you don't accumulate too many changes and thus prevent a successful snapshot.

Redis Versions Prior to 2.8.22

Redis backup and synchronization in versions prior to 2.8.22, is a three-step process.

1. Fork, and in the background process, serialize the cluster data to disk. This creates a point-in-time snapshot.
2. In the foreground, accumulate a change log in the *client output buffer*.

   Important
   If the change log exceeds the *client output buffer* size, the backup or synchronization fails. For more information, see Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60).
3. Finally, transmit the cache data and then the change log to the replica cluster.

Creating a Redis Cluster with Replicas

You have the following options for creating a cluster with replica nodes. Which you use depends on whether you already have an available Redis (cluster mode disabled) cluster not associated with any cluster that has replicas to use as the primary node, or you need to create the primary node in with the cluster, and read replicas. Currently, a Redis (cluster mode enabled) cluster must be created from scratch.

Option 1: Creating a Cluster with Replicas Using an Available Redis (cluster mode disabled) Cluster (p. 204)
Use this option to leverage an existing single-node Redis (cluster mode disabled) cluster. You will specify this cluster as the primary in the new cluster, and then individually add 1 to 5 read replica to the cluster. If the existing cluster is active, read replicas synchronize with it as they are created.

> **Important**
> You cannot create a Redis (cluster mode enabled) cluster using an existing cluster.
> To create a Redis (cluster mode enabled) cluster using the ElastiCache console, see Creating a Redis (cluster mode enabled) Cluster (Console) (p. 134).

Option 2: Creating a Replication Group from Scratch (p. 209)
Use this option if you don't already have an available Redis (cluster mode disabled) cluster to use as the cluster's primary, or if you want to create a Redis (cluster mode enabled) cluster. If you're creating a Redis (cluster mode disabled) cluster, using this option you specify how many read replicas you want. If you're creating a Redis (cluster mode enabled) cluster, using this option you specify both the number of shards and replicas per shard you want. The process then creates the cluster, shards, primary node and the specified number of read replicas without further input or actions from you.

Creating a Cluster with Replicas Using an Available Redis (cluster mode disabled) Cluster

An available cluster is an existing single-node Redis cluster. Currently, Redis (cluster mode enabled) does not support creating a cluster with replicas using an available single-node cluster. If you want to create a Redis (cluster mode enabled) cluster, see Creating a Redis (cluster mode enabled) Replication Group from Scratch (Console) (p. 215).

The following procedure can only be used if you have a Redis (cluster mode disabled) single-node cluster. This cluster becomes the primary in the new cluster. If you do not have a Redis (cluster mode disabled) cluster you can use as the new cluster's primary, see Creating a Replication Group from Scratch (p. 209).

Related Topics

- See the topic Adding Nodes to a Cluster (Console) (p. 154)
- Creating a Replication Group Using an Available Redis Cache Cluster (AWS CLI) (p. 205)
- Creating a Replication Group Using an Available Redis Cache Cluster (ElastiCache API) (p. 206)

Creating a Cluster with Replicas Using an Available Redis Cluster (Console)

See the topic Adding Nodes to a Cluster (Console) (p. 154).

Creating a Replication Group Using an Available Redis Cache Cluster (AWS CLI)

There are two steps to creating a replication group with read replicas when using an available Redis Cache Cluster for the primary when using the AWS CLI.

First, create the replication group specifying the available Redis cache cluster as the primary

To create a replication group when you already have an available Redis cache cluster, use the AWS CLI `create-replication-group` command, being sure to include the following parameters.

--replication-group-id
The name of the replication group you are creating.

Redis (cluster mode disabled) Replication Group 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.

--replication-group-description
Description of the replication group.

--primary-cluster-id
The name of the available Redis cache cluster that will be the primary cluster in this replication group.

The following command creates the replication group `my-repl-group` using the available Redis cache cluster `my-primary` as the replication group's primary cluster. The settings of `my-primary` (that is, parameter group, security group, node type, etc.) will be applied to all nodes in the replication group.

For Linux, OS X, or Unix:

```
aws elasticache create-replication-group \
   --replication-group-id my-repl-group \
   --replication-group-description "test group" \
   --primary-cluster-id my-primary
```

For Windows:

```
aws elasticache create-replication-group ^
   --replication-group-id my-repl-group ^
   --replication-group-description "test group" ^
   --primary-cluster-id my-primary
```

For additional information and parameters you might want to use, see the AWS CLI topic create-replication-group.

Next, add read replicas to the replication group

After the replication group is created, add one to five read replicas to it using the `create-cache-cluster` command, being sure to include the following parameters.

--cache-cluster-id
The name of the cluster you are adding to the replication group.

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.

--replication-group-id
The name of the replication group to which you are adding this cache cluster.

Repeat this command for each read replica you want to add to the replication group, changing only the value of the `--cache-cluster-id` parameter.

> **Note**
> Remember, a replication group cannot have more than five read replicas. Attempting to add a read replica to a replication group that already has five read replicas causes the operation to fail.

The following code adds the read replica `my-replica01` to the replication group `my-repl-group`. The settings of the primary cluster–parameter group, security group, node type, etc.–will be applied to nodes as they are added to the replication group.

For Linux, OS X, or Unix:

```
aws elasticache create-cache-cluster \
   --cache-cluster-id my-replica01 \
   --replication-group-id my-repl-group
```

For Windows:

```
aws elasticache create-cache-cluster ^
   --cache-cluster-id my-replica01 ^
   --replication-group-id my-repl-group
```

For additional information and parameters you might want to use, see the AWS CLI topic create-cache-cluster.

Creating a Replication Group Using an Available Redis Cache Cluster (ElastiCache API)

There are two steps to creating a replication group with read replicas when using an available Redis cache cluster for the primary.

First, create the replication group using the available Redis cache cluster as the primary cluster

To create a replication group when you already have an available Redis cache cluster, use the ElastiCache API `CreateReplicationGroup` operation, being sure to include the following parameters.

ReplicationGroupId
The name of the replication group you are creating.

Redis (cluster mode disabled) Replication Group 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.

ReplicationGroupDescription
Your description of the replication group.

PrimaryClusterId
The name of the available Redis cache cluster that will be the primary cluster in this replication group.

The following operation creates the replication group `myReplGroup` using the available Redis cache cluster `myPrimary` as the replication group's primary cluster. The settings of `myPrimary` (parameter group, security group, node type, etc.) will be applied to all nodes in the replication group.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateReplicationGroup
   &ReplicationGroupDescription=My%20replication%20group
   &ReplicationGroupId=myReplGroup
   &PrimaryClusterId=myPrimary
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

For additional information and parameters you might want to use, see the ElastiCache API topic CreateReplicationGroup.

Next, add read replicas to the replication group

After the replication group is created, add one to five read replicas to it using the `CreateCacheCluster` operation, being sure to include the following parameters.

CacheClusterId
The name of the cluster you are adding to the replication group.

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.

ReplicationGroupId
The name of the replication group to which you are adding this cache cluster.

Repeat this operation for each read replica you want to add to the replication group, changing only the value of the *`CacheClusterId`* parameter.

The following code adds the read replica `myReplica01` to the replication group `myReplGroup` The settings of the primary cluster–parameter group, security group, node type, etc.–will be applied to nodes as they are added to the replication group.

```
https://elasticache.us-west-2.amazonaws.com/
 ?Action=CreateCacheCluster
 &CacheClusterId=myReplica01
```

```
 &ReplicationGroupId=myReplGroup
 &SignatureMethod=HmacSHA256
 &SignatureVersion=4
 &Version=2015-02-02
 &X-Amz-Algorithm=AWS4-HMAC-SHA256
 &X-Amz-Credential=[your-access-key-id]/20150202/us-west-2/elasticache/
aws4_request
 &X-Amz-Date=20150202T170651Z
 &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
 &X-Amz-Signature=[signature-value]
```

For additional information and parameters you might want to use, see the ElastiCache API topic CreateCacheCluster.

Creating a Replication Group from Scratch

This topic covers how to create a Redis replication group without using an existing Redis cluster as the primary. You can create a Redis (cluster mode disabled) or Redis (cluster mode enabled) replication group from scratch using the ElastiCache console, the AWS CLI, or the ElastiCache API.

Before you continue, decide whether you want to create a Redis (cluster mode disabled) or a Redis (cluster mode enabled) replication group. For guidance in deciding, see Replication: Redis (cluster mode disabled) vs. Redis (cluster mode enabled) (p. 193).

Topics

Creating a Redis (cluster mode disabled) Cluster with Replicas from Scratch

You can create a Redis (cluster mode disabled) replication group from scratch using the ElastiCache console, the AWS CLI, or the ElastiCache API. A Redis (cluster mode disabled) replication group always has one node group, a primary cluster, and up to 5 read replicas. Redis (cluster mode disabled) replication groups do not support partitioning your data.

Topics

Creating a Redis Cluster with Replicas from Scratch (Console)

To create a Redis (cluster mode disabled) cluster with replicas, see Creating a Redis (cluster mode disabled) Cluster (Console) (p. 130). Specify at least one replica node.

Creating a Redis (cluster mode disabled) Replication Group from Scratch (AWS CLI)

The following procedure creates a Redis (cluster mode disabled) replication group using the AWS CLI.

When you create a Redis (cluster mode disabled) replication group from scratch, you create the replication group and all its nodes with a single call to the AWS CLI `create-replication-group` command. Include the following parameters.

--replication-group-id
The name of the replication group you are creating.

Redis (cluster mode disabled) Replication Group 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.

--replication-group-description
(Optional) Description of the replication group.

--num-cache-clusters
The total number of clusters (nodes) you want created with this replication group, primary and read replicas combined.

If you enable Multi-AZ (`--automatic-failover-enabled`), the value of `--num-cache-clusters` must be at least 2.

--cache-node-type
The node type for each node in the replication group.

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: `cache.t2.micro`, `cache.t2.small`, `cache.t2.medium`, `cache.m3.medium`, `cache.m3.large`, `cache.m3.xlarge`, `cache.m3.2xlarge`, `cache.m4.large`, `cache.m4.xlarge`, `cache.m4.2xlarge`, `cache.m4.4xlarge`, `cache.m4.10xlarge`
- Previous generation: `cache.t1.micro`, `cache.m1.small`, `cache.m1.medium`, `cache.m1.large`, `cache.m1.xlarge`

- Compute optimized: `cache.c1.xlarge`
- Memory optimized:
  - Current generation: `cache.r3.large`, `cache.r3.xlarge`, `cache.r3.2xlarge`, `cache.r3.4xlarge`, `cache.r3.8xlarge`
  - Previous generation: `cache.m2.xlarge`, `cache.m2.2xlarge`, `cache.m2.4xlarge`

Supported node types are available in all regions except as noted in the following table.

Exceptions

Region Name	Region	Exception
Asia Pacific (Seoul)	ap-northeast-2	Supports only *current generation* node types.
EU (Frankfurt)	eu-central-1	Supports only *current generation* node types.
AWS GovCloud (US)	us-gov-west-1	Supports only *current generation* node types. Does not support M4 node types.
US East (Ohio)	us-east-2	Supports only node types T2, M4, and R3. Supports only the following engine versions: • Memcached: 1.4.24 • Redis 2.8.21, 2.8.23, 2.8.24, and 3.2.4

Note

- All T2 instances are created in an Amazon VPC (Amazon VPC).
- Redis backup and restore is not supported for T2 instances.
- Redis append-only files (AOF) are not supported for T1 or T2 instances.
- Redis Multi-AZ with automatic failover is not supported on T1 or T2 instances.
- Redis configuration variables *`appendonly`* and *`appendfsync`* are not supported on Redis version 2.8.22 and later.

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

- Amazon ElastiCache Product Features and Details
- Memcached Node-Type Specific Parameters
- Redis Node-Type Specific Parameters

--cache-parameter-group
Specify a parameter group that corresponds to your engine version. If you are running Redis 3.2.4 or later, specify the *`default.redis3.2`* parameter group or a parameter group derived from *`default.redis3.2`* to create a Redis (cluster mode disabled) replication group. For more information, see Redis Specific Parameters (p. 292).

--engine
redis

--engine-version
To have the richest set of features, choose the latest engine version.

The names of the nodes will be derived from the replication group name by postpending `-00#` to the replication group name. For example, using the replication group name `myReplGroup`, the name for the primary will be `myReplGroup-001` and the read replicas `myReplGroup-002` through `myReplGroup-006`.

The following command creates the replication group `my-repl-group` with the following parameter values.

- `--replication-group-id` *my-repl-group*
- `--replication-group-description` *"test group"*
- `--num-cache-clusters` *3*
- `--cache-node-type` *cache.m3.large*
- `--engine` *redis*
- `--engine-version` *2.8.4*

For Linux, OS X, or Unix:

```
aws elasticache create-replication-group \
   --replication-group-id my-repl-group \
   --replication-group-description "test group" \
   --num-cache-clusters 3 \
   --cache-node-type cache.m3.large \
   --cache-parameter-group default.redis3.2 \
   --engine redis \
   --engine-version 3.2.4
```

For Windows:

```
aws elasticache create-replication-group ^
   --replication-group-id my-repl-group ^
   --replication-group-description "test group" ^
   --num-cache-clusters 3 ^
   --cache-node-type cache.m3.large ^
   --cache-parameter-group default.redis3.2 ^
   --engine redis ^
   --engine-version 3.2.4
```

For additional information and parameters you might want to use, see the AWS CLI topic create-replication-group.

Creating a Redis (cluster mode disabled) Replication Group from Scratch (ElastiCache API)

The following procedure creates a Redis (cluster mode disabled) replication group using the ElastiCache API.

When you create a Redis (cluster mode disabled) replication group from scratch, you create the replication group and all its nodes with a single call to the ElastiCache API `CreateReplicationGroup` operation. Include the following parameters.

ReplicationGroupId

The name of the replication group you are creating.

Redis (cluster mode enabled) Replication Group 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.

ReplicationGroupDescription
Your description of the replication group.

NumCacheClusters
The total number of cache clusters (nodes) you want created with this replication group, primary and read replicas combined.

If you enable Multi-AZ (`AutomaticFailoverEnabled=true`), the value of `NumCacheClusters` must be at least 2.

CacheNodeType
The node type for each node in the replication group.

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: `cache.t2.micro`, `cache.t2.small`, `cache.t2.medium`, `cache.m3.medium`, `cache.m3.large`, `cache.m3.xlarge`, `cache.m3.2xlarge`, `cache.m4.large`, `cache.m4.xlarge`, `cache.m4.2xlarge`, `cache.m4.4xlarge`, `cache.m4.10xlarge`
  - Previous generation: `cache.t1.micro`, `cache.m1.small`, `cache.m1.medium`, `cache.m1.large`, `cache.m1.xlarge`
- Compute optimized: `cache.c1.xlarge`
- Memory optimized:
  - Current generation: `cache.r3.large`, `cache.r3.xlarge`, `cache.r3.2xlarge`, `cache.r3.4xlarge`, `cache.r3.8xlarge`
  - Previous generation: `cache.m2.xlarge`, `cache.m2.2xlarge`, `cache.m2.4xlarge`

Supported node types are available in all regions except as noted in the following table.

Exceptions

Region Name	Region	Exception
Asia Pacific (Seoul)	ap-northeast-2	Supports only *current generation* node types.
EU (Frankfurt)	eu-central-1	Supports only *current generation* node types.
AWS GovCloud (US)	us-gov-west-1	Supports only *current generation* node types. Does not support M4 node types.
US East (Ohio)	us-east-2	Supports only node types T2, M4, and R3. Supports only the following engine versions: • Memcached: 1.4.24 • Redis 2.8.21, 2.8.23, 2.8.24, and 3.2.4

Note

- All T2 instances are created in an Amazon VPC (Amazon VPC).
- Redis backup and restore is not supported for T2 instances.

- Redis append-only files (AOF) are not supported for T1 or T2 instances.
- Redis Multi-AZ with automatic failover is not supported on T1 or T2 instances.
- Redis configuration variables *appendonly* and *appendfsync* are not supported on Redis version 2.8.22 and later.

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

- Amazon ElastiCache Product Features and Details
- Memcached Node-Type Specific Parameters
- Redis Node-Type Specific Parameters

CacheParameterGroup
Specify a parameter group that corresponds to your engine version. If you are running Redis 3.2.4 or later, specify the *default.redis3.2* parameter group or a parameter group derived from *default.redis3.2* to create a Redis (cluster mode disabled) replication group. For more information, see Redis Specific Parameters (p. 292).

Engine
redis

EngineVersion
3.2.4

The names of the nodes will be derived from the replication group name by postpending `-00#` to the replication group name. For example, using the replication group name `myReplGroup`, the name for the primary will be `myReplGroup-001` and the read replicas `myReplGroup-002` through `myReplGroup-006`.

The following operation creates the replication group `myReplGroup` with the following parameter values.

- *ReplicationGroupId* *myReplGroup*
- *ReplicationGroupDescription* *"test group"*
- *NumCacheClusters* *3*
- *CacheNodeType* *cache.m3.large*
- *Engine* *redis*

Line breaks are added for readability.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateReplicationGroup
   &CacheNodeType=cache.m3.large
   &CacheParameterGroup=default.redis3.2
   &Engine=redis
   &NumCacheClusters=3
   &ReplicationGroupDescription=test%20group
   &ReplicationGroupId=myReplGroup
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

For additional information and parameters you might want to use, see the ElastiCache API topic CreateReplicationGroup.

Creating a Redis (cluster mode enabled) Replication Group from Scratch

You can create a Redis (cluster mode enabled) cluster with replicas (called *replication group* in the API/CLI) from scratch using the ElastiCache console, the AWS CLI, or the ElastiCache API. A Redis (cluster mode enabled) replication group has from 1 to 15 shards (called *node groups* in the API/CLI), a primary cluster in each shard, and up to 5 read replicas in each shard. When you use the ElastiCache console to create the cluster, the number of read replicas is the same for every shard

Topics

Creating a Redis (cluster mode enabled) Replication Group from Scratch (Console)

To create a Redis (cluster mode enabled) cluster with replicas, see Creating a Redis (cluster mode enabled) Cluster (Console) (p. 134). Be sure to enable cluster mode, **Cluster Mode enabled (Scale Out)**, and specify at least two shards and one replica node.

Creating a Redis (cluster mode enabled) Replication Group from Scratch (AWS CLI)

The following procedure creates a Redis (cluster mode enabled) replication group using the AWS CLI.

When you create a Redis (cluster mode enabled) replication group from scratch, you create the replication group and all its nodes with a single call to the AWS CLI `create-replication-group` command. Include the following parameters.

--replication-group-id
: The name of the replication group you are creating.

Redis (cluster mode enabled) Replication Group 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.

--replication-group-description
: (Optional) Description of the replication group.

--cache-node-type
: The node type for each node in the replication group.

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: `cache.t2.micro`, `cache.t2.small`, `cache.t2.medium`, `cache.m3.medium`, `cache.m3.large`, `cache.m3.xlarge`, `cache.m3.2xlarge`,

    `cache.m4.large`, `cache.m4.xlarge`, `cache.m4.2xlarge`, `cache.m4.4xlarge`, `cache.m4.10xlarge`
  - Previous generation: `cache.t1.micro`, `cache.m1.small`, `cache.m1.medium`, `cache.m1.large`, `cache.m1.xlarge`
- Compute optimized: `cache.c1.xlarge`
- Memory optimized:
  - Current generation: `cache.r3.large`, `cache.r3.xlarge`, `cache.r3.2xlarge`, `cache.r3.4xlarge`, `cache.r3.8xlarge`
  - Previous generation: `cache.m2.xlarge`, `cache.m2.2xlarge`, `cache.m2.4xlarge`

Supported node types are available in all regions except as noted in the following table.

Exceptions

Region Name	Region	Exception
Asia Pacific (Seoul)	ap-northeast-2	Supports only *current generation* node types.
EU (Frankfurt)	eu-central-1	Supports only *current generation* node types.
AWS GovCloud (US)	us-gov-west-1	Supports only *current generation* node types. Does not support M4 node types.
US East (Ohio)	us-east-2	Supports only node types T2, M4, and R3. Supports only the following engine versions: • Memcached: 1.4.24 • Redis 2.8.21, 2.8.23, 2.8.24, and 3.2.4

> **Note**
> - All T2 instances are created in an Amazon VPC (Amazon VPC).
> - Redis backup and restore is not supported for T2 instances.
> - Redis append-only files (AOF) are not supported for T1 or T2 instances.
> - Redis Multi-AZ with automatic failover is not supported on T1 or T2 instances.
> - Redis configuration variables *`appendonly`* and *`appendfsync`* are not supported on Redis version 2.8.22 and later.

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

- Amazon ElastiCache Product Features and Details
- Memcached Node-Type Specific Parameters
- Redis Node-Type Specific Parameters

--cache-parameter-group
: Specify the *`default.redis3.2.cluster.on`* parameter group or a parameter group derived from *`default.redis3.2.cluster.on`* to create a Redis (cluster mode enabled) replication group. For more information, see Redis 3.2.4 Parameter Changes (p. 293).

--engine
: redis

--engine-version
: 3.2.4

--num-node-groups
: The number of node groups in this replication group. Valid values are 1 to 15.

--replicas-per-node-group
The number of replica nodes in each node group. Valid values are 1 to 5.

--node-group-configuration
The configuration for each node group. The *--node-group-configuration* parameter consists of the following fields.

- *--primary-availability-zone* – The Availability Zone where the primary node of this node group is located. If this parameter is omitted, ElastiCache selects the Availability Zone for the primary node.

  Example: us-west-2a.
- *--replica-availability-zones* – A comma separated list of Availability Zones where the read replicas are located. The number of Availability Zones in this list must match the value of *--replica-count*. If this parameter is omitted, ElastiCache selects the Availability Zones for the replica nodes.

  Example: "us-west-2a,us-west-2b,us-west-2c"
- *--replica-count* – The number of replica nodes in this node group.
- *--slots* – A string of comma delimited values where the first set of values are the slot numbers (zero based), and the second set of values are the keyspaces for each slot. If this parameter is omitted, ElastiCache allocates keys equally among the slots.

  Example: The following example is for three slots, numbered 0, 1, and 2.

  "0,1,2,0-4999,5000-9999,10000-16,383

The names of the nodes will be derived from the replication group name by postpending `-00#` to the replication group name. For example, using the replication group name `myReplGroup`, the name for the primary will be `myReplGroup-001` and the read replicas `myReplGroup-002` through `myReplGroup-006`.

The following command creates the replication group `my-repl-group` with the following parameter values.

- *--replication-group-id my-repl-group*
- *--replication-group-description "test group"*
- *--num-node-groups 3*
- *--replicas-per-node-group 2*
- *--cache-node-type cache.m3.large*
- *--cache-parameter-group default.redis3.2.cluster.on*
- *--engine redis*
- *--engine-version 3.2.4*

For Linux, OS X, or Unix:

```
aws elasticache create-replication-group \
   --replication-group-id my-repl-group \
   --replication-group-description "test group" \
   --num-node-groups 3 \
   --replicas-per-node-group 2 \
   --cache-node-type cache.m3.large \
   --cache-parameter-group default.redis3.2.cluster.on \
   --engine redis \
```

```
   --engine-version 3.2.4
```

For Windows:

```
aws elasticache create-replication-group ^
   --replication-group-id my-repl-group ^
   --replication-group-description "test group" ^
   --num-node-groups 3 ^
   --replicas-per-node-group 2 ^
   --cache-node-type cache.m3.large ^
   --cache-parameter-group default.redis3.2.cluster.on ^
   --engine redis ^
   --engine-version 3.2.4
```

For additional information and parameters you might want to use, see the AWS CLI topic create-replication-group.

Creating a Redis (cluster mode enabled) Replication Group from Scratch (ElastiCache API)

The following procedure creates a Redis (cluster mode enabled) replication group using the ElastiCache API.

When you create a Redis (cluster mode enabled) replication group from scratch, you create the replication group and all its nodes with a single call to the ElastiCache API `CreateReplicationGroup` operation. Include the following parameters.

ReplicationGroupId
The name of the replication group you are creating.

Redis (cluster mode enabled) Replication Group 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.

ReplicationGroupDescription
Description of the replication group.

NumNodeGroups
The number of node groups you want created with this replication group. Valid values are 1 to 15.

ReplicasPerNodeGroup
The number of replica nodes in each node group. Valid values are 1 to 5.

NodeGroupConfiguration
The configuration for each node group. The *`NodeGroupConfiguration`* parameter consists of the following fields.

- *`PrimaryAvailabilityZone`* – The Availability Zone where the primary node of this node group is located. If this parameter is omitted, ElastiCache selects the Availability Zone for the primary node.

  Example: us-west-2a.

- *`ReplicaAvailabilityZones`* – A list of Availability Zones where the read replicas are located. The number of Availability Zones in this list must match the value of *`ReplicaCount`*. If this parameter is omitted, ElastiCache selects the Availability Zones for the replica nodes.

- *ReplicaCount* – The number of replica nodes in this node group.
- *Slots* – A string of comma delimited values where the first set of values are the slot numbers (zero based), and the second set of values are the keyspaces for each slot. If this parameter is omitted, ElastiCache allocates keys equally among the slots.

  Example: The following example is for three slots, numbered 0, 1, and 2.

  "0,1,2,0-4999,5000-9999,10000-16,383

CacheNodeType

The node type for each node in the replication group.

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: `cache.t2.micro`, `cache.t2.small`, `cache.t2.medium`, `cache.m3.medium`, `cache.m3.large`, `cache.m3.xlarge`, `cache.m3.2xlarge`, `cache.m4.large`, `cache.m4.xlarge`, `cache.m4.2xlarge`, `cache.m4.4xlarge`, `cache.m4.10xlarge`
  - Previous generation: `cache.t1.micro`, `cache.m1.small`, `cache.m1.medium`, `cache.m1.large`, `cache.m1.xlarge`
- Compute optimized: `cache.c1.xlarge`
- Memory optimized:
  - Current generation: `cache.r3.large`, `cache.r3.xlarge`, `cache.r3.2xlarge`, `cache.r3.4xlarge`, `cache.r3.8xlarge`
  - Previous generation: `cache.m2.xlarge`, `cache.m2.2xlarge`, `cache.m2.4xlarge`

Supported node types are available in all regions except as noted in the following table.

Exceptions

Region Name	Region	Exception
Asia Pacific (Seoul)	ap-northeast-2	Supports only *current generation* node types.
EU (Frankfurt)	eu-central-1	Supports only *current generation* node types.
AWS GovCloud (US)	us-gov-west-1	Supports only *current generation* node types. Does not support M4 node types.
US East (Ohio)	us-east-2	Supports only node types T2, M4, and R3. Supports only the following engine versions: • Memcached: 1.4.24 • Redis 2.8.21, 2.8.23, 2.8.24, and 3.2.4

Note

- All T2 instances are created in an Amazon VPC (Amazon VPC).
- Redis backup and restore is not supported for T2 instances.
- Redis append-only files (AOF) are not supported for T1 or T2 instances.

- Redis Multi-AZ with automatic failover is not supported on T1 or T2 instances.
- Redis configuration variables *appendonly* and *appendfsync* are not supported on Redis version 2.8.22 and later.

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

- Amazon ElastiCache Product Features and Details
- Memcached Node-Type Specific Parameters
- Redis Node-Type Specific Parameters

CacheParameterGroup
Specify the *default.redis3.2.cluster.on* parameter group or a parameter group derived from *default.redis3.2.cluster.on* to create a Redis (cluster mode enabled) replication group. For more information, see Redis 3.2.4 Parameter Changes (p. 293).

Engine
redis

EngineVersion
3.2.4

The names of the nodes will be derived from the replication group name by postpending `-00#` to the replication group name. For example, using the replication group name `myReplGroup`, the name for the primary will be `myReplGroup-001` and the read replicas `myReplGroup-002` through `myReplGroup-006`.

The following command creates the replication group `myReplGroup` with the following parameter values.

- *ReplicationGroupId* *myReplGroup*
- *ReplicationGroupDescription* *"test group"*
- *NumNodeGroups* *3*
- *ReplicasPerNodeGroup* *2*
- *CacheNodeType* *cache.m3.large*
- *CacheParameterGroup* *default.redis3.2.cluster.on*
- *Engine* *redis*
- *EngineVersion* *3.2.4*

Line breaks are added for readability.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateReplicationGroup
   &CacheNodeType=cache.m3.large
   &CacheParemeterGroup=default.redis3.2.cluster.on
   &Engine=redis
   &EngineVersion=3.2.4
   &NumNodeGroups=3
   &ReplicasPerNodeGroup=2
   &ReplicationGroupDescription=test%20group
   &ReplicationGroupId=myReplGroup
   &Version=2015-02-02
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

For additional information and parameters you might want to use, see the ElastiCache API topic CreateReplicationGroup.

Viewing a Replication Group's Details

There are times you may want to view the details of a replication group. You can use the ElastiCache console, the AWS CLI for ElastiCache, or the ElastiCache API. The console process is different for Redis (cluster mode disabled) and Redis (cluster mode enabled).

Topics

Viewing a Redis (cluster mode disabled) with Replicas Details: Redis (cluster mode disabled)

You can view the details of a Redis (cluster mode disabled) cluster with replicas (called *replication group* in the API/CLI) using the ElastiCache console, the AWS CLI for ElastiCache, or the ElastiCache API.

Topics

Viewing a Redis (cluster mode disabled) Cluster with Replicas Details: Redis (cluster mode disabled) (Console)

To view the details of a Redis (cluster mode disabled) cluster with replicas using the ElastiCache console, see the topic Viewing a Redis (cluster mode disabled) Cluster's Details (Console) (p. 144).

Viewing a Replication Group's Details: Redis (cluster mode disabled) (AWS CLI)

For an AWS CLI example that displays a Redis (cluster mode disabled) replication group's details, see Viewing a Replication Group's Details: (AWS CLI) (p. 222).

Viewing a Replication Group's Details: Redis (cluster mode disabled) (ElastiCache API)

For an ElastiCache API example that displays a Redis (cluster mode disabled) replication group's details, see Viewing a Replication Group's Details: (ElastiCache API) (p. 224).

Viewing a Replication Group's Details: Redis (cluster mode enabled)

Viewing a Redis (cluster mode enabled) Cluster with Replicas Details: Redis (cluster mode disabled) (Console)

To view the details of a Redis (cluster mode enabled) cluster with replicas using the ElastiCache console, see Viewing a Redis (cluster mode enabled) Cluster's Details (Console) (p. 145).

Viewing a Replication Group's Details: Redis (cluster mode enabled) (AWS CLI)

For an ElastiCache CLI example that displays a Redis (cluster mode enabled) replication group's details, see Viewing a Replication Group's Details: (AWS CLI) (p. 222).

Viewing a Replication Group's Details: Redis (cluster mode enabled) (ElastiCache API)

For an ElastiCache API example that displays a Redis (cluster mode enabled) replication group's details, see Viewing a Replication Group's Details: (ElastiCache API) (p. 224).

Viewing a Replication Group's Details: (AWS CLI)

You can view the details for a replication using the AWS CLI `describe-replication-groups` command. Use the following optional parameters to refine the listing. Omitting the parameters returns the details for up to 100 replication groups.

Optional Parameters

- *`--replication-group-id`* – Use this parameter to list the details of a specific replication group. If the specified replication group has more than one node group, results are returned grouped by node group.
- *`--max-items`* – Use this parameter to limit the number of replication groups listed. The value of *`--max-items`* cannot be less than 20 or greater than 100.

The following code lists the details for up to 100 replication groups.

```
aws elasticache describe-replication-groups
```

The following code lists the details for my-repl-group.

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

The following code lists the details for up to 25 replication groups.

```
aws elasticache describe-replication-groups --max-items 25
```

Output from this operation should look something like this (JSON format).

Example

```
{
   "ReplicationGroups": [
      {
         "Status": "available",
         "Description": " ",
         "NodeGroups": [
            {
               "Status": "available",
                  "NodeGroupMembers": [
                     {
                        "CurrentRole": "primary",
                        "PreferredAvailabilityZone": "us-west-2a",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                           "Port": 6379,
                           "Address": "rg-
name-001.1abc4d.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "rg-name-001"
                     },
                     {
                        "CurrentRole": "replica",
                        "PreferredAvailabilityZone": "us-west-2b",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                           "Port": 6379,
                           "Address": "rg-
name-002.1abc4d.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "rg-name-002"
                     },
                     {
                        "CurrentRole": "replica",
                        "PreferredAvailabilityZone": "us-west-2c",
                        "CacheNodeId": "0001",
                        "ReadEndpoint": {
                           "Port": 6379,
                           "Address": "rg-
name-003.1abc4d.0001.usw2.cache.amazonaws.com"
                        },
                        "CacheClusterId": "rg-name-003"
                     }
                  ],
                  "NodeGroupId": "0001",
                  "PrimaryEndpoint": {
                     "Port": 6379,
                     "Address": "rg-
name.1abc4d.ng.0001.usw2.cache.amazonaws.com"
                  }
               }
            ],
            "ReplicationGroupId": "rg-name",
            "AutomaticFailover": "enabled",
            "SnapshottingClusterId": "rg-name-002",
            "MemberClusters": [
               "rg-name-001",
               "rg-name-002",
               "rg-name-003"
            ],
            "PendingModifiedValues": {}
         },
         {
         ... some output omitted for previty
         }
   ]
}
```

For more information, see the AWS CLI for ElastiCache topic `describe-replication-groups`.

Viewing a Replication Group's Details: (ElastiCache API)

You can view the details for a replication using the AWS CLI `DescribeReplicationGroups` operation. Use the following optional parameters to refine the listing. Omitting the parameters returns the details for up to 100 replication groups.

Optional Parameters

- *`ReplicationGroupId`* – Use this parameter to list the details of a specific replication group. If the specified replication group has more than one node group, results are returned grouped by node group.
- *`MaxRecords`* – Use this parameter to limit the number of replication groups listed. The value of *`MaxRecords`* cannot be less than 20 or greater than 100. The default is 100.

Example

The following code list the details for up to 100 replication groups.

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

The following code lists the details for `myReplGroup`.

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

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

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

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

Finding Replication Group Endpoints

An application can connect to any node in a replication group, provided that it has the DNS endpoint and port number for that node. Depending upon whether you are running a Redis (cluster mode disabled) or a Redis (cluster mode enabled) replication group, you will be interested in different endpoints.

Redis (cluster mode disabled)

Redis (cluster mode disabled) clusters with replicas have two types of endpoints; the *primary endpoint* and the *node endpoints*. The primary endpoint is a DNS name that always resolves to the primary cluster in the cluster. The primary endpoint is immune to changes to your cluster, such as promoting a read replica to the primary role. For write activity, we recommend that your applications connect to the primary endpoint instead of connecting directly to the primary.

For read activity, applications can connect to any node in the cluster. Unlike the primary endpoint, node endpoints resolve to specific endpoints. If you make a change in your cluster, such as adding or deleting a replica, you must update the node endpoints in your application.

Redis (cluster mode enabled)

Redis (cluster mode enabled) clusters with replicas, because they have multiple shards (called *node groups* in the API/CLI), which mean they also have multiple primary nodes, have a different endpoint structure than Redis (cluster mode disabled). Redis (cluster mode enabled) has a *configuration endpoint* which "knows" all the primary and node endpoints in the cluster. Your application connects to the configuration endpoint. Whenever your application writes to or reads from the cluster's configuration endpoint, Redis, behind the scenes, determines which shard the key belongs to and which endpoint in that shard to use. It is all quite transparent to your application.

You can find the endpoints for a cluster using the ElastiCache console, the AWS CLI, or the ElastiCache API.

Finding Replication Group Endpoints

To find the endpoints for your replication group, see one of the following topics:

- Finding the Endpoints for a Redis (cluster mode disabled) Cluster (Console) (p. 51)
- Finding the Endpoints for a Redis (cluster mode enabled) Cluster (Console) (p. 52)
- Finding the Endpoints for Replication Groups (AWS CLI) (p. 56)
- Finding Endpoints for Replication Groups (ElastiCache API) (p. 58)

Modifying a Cluster with Replicas

Important Constraints

- Currently, ElastiCache does not support modifying a Redis (cluster mode enabled) cluster. If you need to modify such a cluster, create the cluster anew with the new cluster incorporating the changes.
- You can upgrade to newer engine versions, but you cannot downgrade to earlier engine versions except by deleting the existing cache cluster or replication group and creating it anew. For more information, see Upgrading Engine Versions (p. 41).

You can modify a Redis (cluster mode disabled) cluster's settings using the ElastiCache console, the AWS CLI, or the ElastiCache API. Currently, ElastiCache does not support modifying a Redis (cluster mode enabled) replication group.

Topics

Modifying a Redis Cluster (Console)

To modify a Redis (cluster mode disabled) cluster, see Modifying an ElastiCache Cluster (p. 149).

Modifying a Replication Group (AWS CLI)

The following AWS CLI command enables Multi-AZ on an existing Redis replication group. You can use the same command to make other modifications to a replication group.

For Linux, OS X, or Unix:

```
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --automatic-failover-enabled
```

For Windows:

```
aws elasticache modify-replication-group ^
   --replication-group-id myReplGroup ^
   --automatic-failover-enabled
```

For more information on the AWS CLI `modify-replication-group` command, see modify-replication-group.

Modifying a Replication Group (ElastiCache API)

The following ElastiCache API operation enables Multi-AZ on an existing Redis replication group. You can use the same operation to make other modifications to a replication group.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyReplicationGroup
```

```
   &AutomaticFailoverEnabled=true
   &ReplicationGroupId=myReplGroup
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20141201T220302Z
   &Version=2014-12-01
   &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 on the ElastiCache API `ModifyReplicationGroup` operation, see ModifyReplicationGroup.

Deleting a Cluster with Replicas

If you no longer need a one of your clusters with replicas (called *replication groups* in the API/CLI), you can delete it. When you delete a replication group, ElastiCache deletes all of the nodes in that group.

Once you have begun this operation, it cannot be interrupted or cancelled.

Deleting a Replication Group (Console)

To delete a cluster that has replicas, see Deleting a Cluster (p. 167).

Deleting a Replication Group (AWS CLI)

Use the command `delete-replication-group` to delete a replication group.

```
aws elasticache delete-replication-group --replication-group-id my-repgroup
```

A prompt asks you to confirm your decision. Enter *y* (yes) to start the operation immediately. After the process starts, it is irreversible.

```

   After you begin deleting this replication group, all of its nodes will be
 deleted as well.
   Are you sure you want to delete this replication group? [Ny]y

REPLICATIONGROUP  my-repgroup  My replication group  deleting
```

Deleting a Replication Group (ElastiCache API)

Call `DeleteReplicationGroup` with the *`ReplicationGroup`* parameter.

Example

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DeleteReplicationGroup
   &ReplicationGroupId=my-repgroup
   &Version=2014-12-01
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20141201T220302Z
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Date=20141201T220302Z
   &X-Amz-SignedHeaders=Host
   &X-Amz-Expires=20141201T220302Z
   &X-Amz-Credential=<credential>
   &X-Amz-Signature=<signature>
```

> **Note**
> If you set the *`RetainPrimaryCluster`* parameter to `true`, all of the read replicas will be deleted, but the primary cluster will be retained.

Adding a Read Replica to a Redis Cluster

Important
Currently, ElastiCache does not support adding read replicas to a Redis (cluster mode enabled). If you need more read replicas, create the cluster anew with the desired number of read replicas.

As your read traffic increases, you might want to spread those reads across more nodes thereby reducing the read pressure on any one node. This topic covers how to add a read replica to a cluster. You can add a read replica to a cluster using the ElastiCache Console, the AWS CLI, or the ElastiCache API.

- Adding Nodes to a Cluster (p. 154)
- Adding a Read Replica to a Replication Group (AWS CLI) (p. 229)
- Adding a Read Replica to a Replication Group (ElastiCache API) (p. 230)

Topics

Adding a Read Replica to a Cluster (Console)

To add a replica to a Redis (cluster mode disabled) cluster, see Adding Nodes to a Cluster (p. 154).

Adding a Read Replica to a Replication Group (AWS CLI)

To add a read replica to a replication group, use the AWS CLI `create-cache-cluster` command, with the parameter `--replication-group-id` to specify which replication group to add the cluster (node) to.

A replication group can have a maximum of 5 read replicas. If you attempt to add a read replica to a replication group that already has 5 read replicas, the operation will fail.

The following example creates the cluster `my-read-replica` and adds it to the replication group `my-replication-group`. The node types, parameter groups, security groups, maintenance window and other settings for my read replica will be the same as the other nodes in my replication group

For Linux, OS X, or Unix:

```
aws elasticache create-cache-cluster \
   --cache-cluster-id my-read-replica \
   --replicationgroup-id my-replication-group
```

For Windows:

```
aws elasticache create-cache-cluster ^
   --cache-cluster-id my-read-replica ^
   --replicationgroup-id my-replication-group
```

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

Adding a Read Replica to a Replication Group (ElastiCache API)

To add a read replica to a replication group, use the ElastiCache `CreateCacheCluster` operation, with the parameter *`ReplicationGroupId`* to specify which replication group to add the cluster (node) to.

A replication group can have a maximum of five read replicas. If you attempt to add a read replica to a replication group that already has five read replicas, the operation will fail.

The following example creates the cluster `myReadReplica` and adds it to the replication group `myReplicationGroup`. The node types, parameter groups, security groups, maintenance window and other settings for my read replica will be the same as the other nodes in my replication group

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

For more information, see the ElastiCache API topic CreateCacheCluster.

Promoting a Read-Replica to Primary

Important
Currently, ElastiCache does not support promoting a read replica to primary for a Redis (cluster mode enabled) replication group.

You can promote a read replica to primary using the ElastiCache console, the AWS CLI, or the ElastiCache API. However, you cannot promote a read replica to primary while Multi-AZ is enabled on the replication group. If Multi-AZ is enabled you must:

To promote a read replica node to primary

1. Modify the replication group to disable Multi-AZ (this does not require that all your clusters be in the same Availability Zone).

   For information on modifying a replication group's settings, see Modifying a Cluster with Replicas (p. 226).
2. Promote the read replica to primary.
3. Modify the replication group to re-enable Multi-AZ.

Multi-AZ with automatic failover is not available on replication groups running Redis 2.6.13.

Topics

Promoting a Read-Replica to Primary (Console)

To promote a read replica to primary (console)

1. If the replica you want to promote is a member of a Redis (cluster mode disabled) cluster with replicas where Multi-AZ is enabled, modify the cluster to disable Multi-AZ before you proceed (this does not require that all your clusters be in the same Availability Zone). For more information on modifying a cluster, see Modifying a Cluster (Console) (p. 149).
2. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
3. Choose **Redis**.

   A list of clusters running Redis appears.
4. From the list of clusters, choose the name of the cluster you wish to modify. This cluster must be running the "Redis" engine, not the "Clusterd Redis" engine, and it must have 2 or more nodes.

   A list of the cluster's nodes appears.
5. Choose the box to the left of the name of the replica node you want to promote to Primary.

   Choose **Promote**.
6. In the **Promote Read Replica** dialog box:

   a. Choose *Yes* to promote the read replica immediately, or *No* to promote it at the cluster's next maintenance window.
   b. Choose **Promote** to promote the read replica or **Cancel** to cancel the operation.

7. If the cluster had Multi-AZ enabled before you began the promotion process, modify the cluster to re-enable Multi-AZ. For more information about modifying a cluster, see Modifying a Cluster (Console) (p. 149)

Promoting a Read-Replica to Primary (AWS CLI)

You cannot promote a read replica to primary if the replication group is Multi-AZ enabled. If the replica you want to promote is a member of a replication group where Multi-AZ is enabled, you must modify the replication group to disable Multi-AZ before you proceed (this does not require that all your clusters be in the same Availability Zone). For more information on modifying a replication group, see Modifying a Replication Group (AWS CLI) (p. 226).

The following AWS CLI command modifies the replication group `my-repl-group`, making the read replica `my-replica-1` the primary in the replication group.

For Linux, OS X, or Unix:

```
aws elasticache modify-replication-group \
   --replication-group-id my-repl-group \
   --primary-cluster-id my-replica-1
```

For Windows:

```
aws elasticache modify-replication-group ^
   --replication-group-id my-repl-group ^
   --primary-cluster-id my-replica-1
```

For more information on modifying a replication group, see the AWS CLI topic modify-replication-group

Promoting a Read-Replica to Primary (ElastiCache API)

You cannot promote a read replica to primary if the replication group is Multi-AZ enabled. If the replica you want to promote is a member of a replication group where Multi-AZ is enabled, you must modify the replication group to disable Multi-AZ before you proceed (this does not require that all your clusters be in the same Availability Zone). For more information on modifying a replication group, see Modifying a Replication Group (ElastiCache API) (p. 226).

The following ElastiCache API action modifies the replication group `myReplGroup`, making the read replica `myReplica-1` the primary in the replication group.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyReplicationGroup
   &ReplicationGroupId=myReplGroup
   &PrimaryClusterId=myReplica-1
   &Version=2014-12-01
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20141201T220302Z
   &X-Amz-Algorithm=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 on modifying a replication group, see the ElastiCache API topic ModifyReplicationGroup

Deleting a Read Replica

Important
Currently, ElastiCache does not support deleting a read replica from a Redis (cluster mode enabled) replication group. If you need to reduce the number of read replicas, create the cluster anew with the desired number of read replicas.

As read traffic on your replication group changes you might want to add or remove read replicas. Removing a node from a replication group is the same as just deleting a cluster, though there are some restrictions.

Restriction on removing nodes from a replication group

- You cannot remove the primary from a replication group. If you want to delete the primary, you must do the following:
  1. Promote a read replica to primary. For more information on promoting a read replica to primary, see Promoting a Read-Replica to Primary (p. 231).
  2. Delete the old primary. See the next point for a restriction on this method.
- If Multi-AZ is enabled on a replication group, you cannot remove the last read replica from the replication group. In this case you must:
  1. Modify the replication group by disabling Multi-AZ. For more information, see Modifying a Cluster with Replicas (p. 226).
  2. Delete the read-replica.

You can remove a read replica from a replication group using the ElastiCache console, the AWS CLI for ElastiCache, or the ElastiCache API.

For directions on deleting a cluster see:

- Deleting a Cluster (Console) (p. 167)
- Deleting a Cache Cluster (AWS CLI) (p. 167)
- Deleting a Cache Cluster (ElastiCache API) (p. 167)

ElastiCache Backup & Restore (Redis)

Amazon ElastiCache clusters running Redis can back up their data. The backup can be used to restore a cluster or seed a new cluster. The backup consists of the cluster's metadata, along with all of the data in the cluster. All backups are written to Amazon Simple Storage Service (Amazon S3), which provides durable storage. At any time, you can restore your data by creating a new Redis cluster and populating it with data from a backup. ElastiCache lets you manage backups using the AWS Management Console, the AWS Command Line Interface (AWS CLI), and the ElastiCache API.

Beginning with Redis version 2.8.22, the backup method is selected based upon available memory. If there is sufficient available memory, a child process is spawned which writes all changes to the cache's reserved memory while the cache is being backed up. This child process could, depending on the number of writes to the cache during the backup process, consume all *`reserved memory`*, causing the backup to fail.

If there is insufficient memory available, a forkless, cooperative background process is employed. The forkless method can impact both latency and throughput. For more information, see How Synchronization and Backup are Implemented (p. 203).

For more information about the performance impact of the backup process, see Performance Impact of Backups (p. 236).

This section provides an overview of working with backup and restore.

Constraints

The following constraints should be considered when planning or making backups.

- At this time, backup and restore is supported only for clusters running on Redis.
- Backup and restore is not supported on `cache.t1.micro` or `cache.t2.*` nodes for Redis (cluster mode disabled) clusters. All other cache node types are supported.

  Fore Redis (cluster mode enabled) clusters, backups and restore is supported for all node types.
- During any contiguous 24-hour period, you can create no more than 20 manual backups per cluster.
- Redis (cluster mode enabled) only supports taking backups on the cluster (API/CLI: replication group) level, not at the shard (API/CLI: node group) level.

- When restoring or seeding from a backup, the number of shards in the cluster group must match the number of shards in the cluster used to create the backup.

Costs

ElastiCache allows you to store one backup for each active Redis cluster free of charge. Storage space for additional backups is charged at a rate of $0.085/GB per month for all regions. There are no data transfer fees for creating a backup, or for restoring data from a backup to a Redis cluster.

Performance Impact of Backups

The backup process depends upon which Redis version you're running. Beginning with Redis 2.8.22, the process is forkless.

Backups when running Redis 2.8.22 and later

Redis backups, in versions 2.8.22 and later, choose between two backup methods. If there is insufficient memory to support a forked backup, ElastiCache use a forkless method that employs cooperative background processing. If there is sufficient memory to support a forked save process, the same process as in prior Redis versions is employed.

If the write load is high during a forkless backup, writes to the cache are delayed to ensure that you don't accumulate too many changes and thus prevent a successful backup.

Backups when running Redis versions prior to 2.8.22

Backups are created using Redis' native BGSAVE command: The Redis process on the cache node spawns a child process to write all the data from the cache to a Redis .rdb file. It can take up to ten seconds to spawn the child process, and during this time the parent process is unable to accept incoming application requests. After the child process is running independently, the parent process resumes normal operations. The child process exits when the backup operation is complete.

While the backup is being written, additional cache node memory is used for new writes. If this additional memory usage exceeds the node's available memory, processing can become slow due to excessive paging, or fail.

The following are guidelines for improving backing up performance.

- Set the *reserved-memory* parameter—To mitigate excessive paging, we recommend that you set the *reserved-memory* parameter. This parameter prevents Redis from consuming all of the node's available memory, and can help reduce the amount of paging. You might also see performance improvements by simply using a larger node. For more information about the *reserved-memory* parameter and node memory sizes, see Redis Specific Parameters (p. 292).

- Create backups from a read replica—If you are running Redis in a node group with more than one node, you can take a backup from the primary node or one of the read replicas. Because of the system resources required during a BGSAVE, we recommend that you create backups from one of the read replicas, rather than the primary. While the backup is being created from the replica, the primary node remains unaffected by BGSAVE resource requirements, and can continue serving requests without slowing down.

If you delete a replication group and request a final backup, ElastiCache will always take the backup from the primary node. This ensures that you capture the very latest Redis data, before the replication group is deleted.

Scheduling Automatic Backups

For any Redis cluster, you can enable *automatic* backups. When automatic backups are enabled, ElastiCache creates a backup of the cluster on a daily basis. Automatic backups can help guard against data loss: In the event of a failure, you can create a new cluster, restoring all of your data from the most recent backup. The result is a warm-started cluster, pre-loaded with your data and ready for use. For more information, go to Restoring From a Backup (p. 255).

When you schedule automatic backups, you should plan the following settings:

- **Backup window** – A period during each day when ElastiCache will begin creating a backup. The minimum length for the backup window is 60 minutes. You can set the backup window for any time when it's most convenient for you, or for a time of day that avoids doing backups during particularly high-utilization periods.

  If you do not specify a backup window, ElastiCache will assign one automatically.

- **Backup retention limit** – The number of days the backup will be retained in Amazon S3. For example, if you set the retention limit to 5, then a backup taken today would be retained for 5 days. When the retention limit expires, the backup is automatically deleted.

  The maximum backup retention limit is 35 days. If the backup retention limit is set to 0, automatic backups are disabled for the cluster.

You can enable or disable automatic backups on an existing Redis cluster or replication group by modifying it using the ElastiCache console, the AWS CLI, or the ElastiCache API. For more information on how to enable or disable automatic backups on an existing cluster or replication group, go to Modifying an ElastiCache Cluster (p. 149) or Modifying a Cluster with Replicas (p. 226).

You can enable or disable automatic backups when creating a Redis cluster or replication group using the ElastiCache console, the AWS CLI, or the ElastiCache API. You can enable automatic backups when you create a Redis cluster by checking the **Enable Automatic Backups** box in the **Advanced Redis Settings** section. For more information, see step 2 of Creating a Redis (cluster mode disabled) Cluster (Console) (p. 130). You can enable automatic backups when you create a Redis replication group if you are not using an existing cluster as the primary cluster. For more information, see Creating a Replication Group from Scratch (p. 209).

Taking Manual Backups

In addition to automatic backups, you can create a *manual* backup at any time. Unlike automatic backups which are automatically deleted after a specified retention period, manual backups do not have a retention period after which they are automatically deleted. You must manually delete a manual backup.

Manual backups are useful for testing and archiving. For example, suppose that you've developed a set of baseline data for testing purposes; you can create a manual backup of the data and restore it whenever you want. After you test an application that modifies the data, you can reset the data by creating a new cluster and restoring from your baseline backup. When the cluster is ready, you can test your applications against the baseline data again—and repeat this process as often as needed.

In addition to creating a manual backup, you can create a manual backup in one of the following ways:

- Copying a Backup (p. 247) It does not matter whether the source backup was created automatically or manually.
- Taking a Final Backup (p. 242) Create a backup immediately before deleting a cluster or replication group.

There is a limit in place on the rate of manual backup creation: During any contiguous 24-hour period, you can create no more than 20 manual backups per cluster.

Manual backups do not have retention limits, therefore ElastiCache does not automatically delete them. Even if you delete a cluster, any manual backups from that cluster are retained. If you no longer want to keep a manual backup, you must explicitly delete it yourself.

You can create a manual backup of a cluster using the ElastiCache console, the AWS CLI, or the ElastiCache API.

Creating a Manual Backup (Console)

To create a backup of 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 left navigation pane, choose **Redis**.

   The Redis clusters screen appears.
3. Choose the box to the left of the name of the Redis cluster you want to backup.
4. Choose **Backup**.

   The **Create Backup** dialog appears.
5. In the **Backup name** box, type in a name for your backup.

   #### 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.
6. Choose **Create Backup**.

   The status of the cluster changes to *snapshotting*. When the status returns to *available* the backup is complete.

Creating a Manual Backup (AWS CLI)

To create a manual backup of a cluster using the AWS CLI, use the `create-snapshot` AWS CLI command with the following parameters.

- `--cache-cluster-id` – Name of the cache cluster.
- `--snapshot-name` – Name of the snapshot.

  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.

The following AWS CLI command creates the backup `bkup-20150515` from the cluster `myRedisCluster`.

For Linux, OS X, or Unix:

```
aws elasticache create-snapshot \
    --cache-cluster-id myRedisCluster \
    --snapshot-name bkup-20150515
```

For Windows:

```
aws elasticache create-snapshot ^
    --cache-cluster-id myRedisCluster ^
    --snapshot-name bkup-20150515
```

For more information, see create-snapshot in the *AWS Command Line Interface Reference*.

Creating a Manual Backup (ElastiCache API)

To create a manual backup of a cluster using the ElastiCache API, use the `CreateSnapshot` ElastiCache API action with the following parameters.

- `CacheClusterId` – Name of the cache cluster.
- `SnapshotName` – Name of the backup.

  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.

To create a manual backup of a cluster using the ElastiCache API, use the `CreateSnapshot` ElastiCache API action, specifying the name of the cluster to backup (*`CacheClusterId`*) and the name of the backup (*`bkup-20150515`*).

The following ElastiCache API action creates the backup `bkup-20150515` from the cluster `myRedisCluster`.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=CreateSnapshot
    &CacheClusterId=myRedisCluster
    &SnapshotName=bkup-20150515
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

For more information, see CreateSnapshot in the *Amazon ElastiCache API Reference*.

Taking a Final Backup

You can create a final backup using the ElastiCache console, the AWS CLI, or the ElastiCache API.

Creating a Final Backup (Console)

You can create a final backup when you delete either a Redis cluster (API/CLI: replication group) using the ElastiCache console.

To create a final backup when deleting a Redis cluster, on the delete dialog (step 5), select *Yes* and give the backup a name.

Related Topics

- Deleting a Cluster (Console) (p. 167)
- Deleting a Replication Group (Console) (p. 228)

Taking a Final Backup (AWS CLI)

You can create a final backup when deleting a Redis cluster (API/CLI: replication group) using the AWS CLI.

Topics

When Deleting a Redis Cluster

To create a final backup, use the `delete-cache-cluster` AWS CLI command with the following parameters.

- *`--cache-cluster-id`* – Name of the cluster being deleted.
- *`--final-snapshot-identifier`* – Name of the backup.

The following code creates the final backup `bkup-20150515-final` when deleting the cluster `myRedisCluster`.

For Linux, OS X, or Unix:

```
aws elasticache delete-cache-cluster \
        --cache-cluster-id myRedisCluster \
        --final-snapshot-identifier bkup-20150515-final
```

For Windows:

```
aws elasticache delete-cache-cluster ^
        --cache-cluster-id myRedisCluster ^
        --final-snapshot-identifier bkup-20150515-final
```

For more information, see delete-cache-cluster in the *AWS Command Line Interface Reference*.

When Deleting a Redis Replication Group

To create a final backup when deleting a replication group, use the `delete-replication-group` AWS CLI command, with the following parameters.

- *`--replication-group-id`* – Name of the replication group being deleted.
- *`--final-snapshot-identifier`* – Name of the final backup.

The following code takes the final backup `bkup-20150515-final` when deleting the replication group `myReplGroup`.

For Linux, OS X, or Unix:

```
aws elasticache delete-replication-group \
        --replication-group-id myReplGroup \
        --final-snapshot-identifier bkup-20150515-final
```

For Windows:

```
aws elasticache delete-replication-group ^
        --replication-group-id myReplGroup ^
        --final-snapshot-identifier bkup-20150515-final
```

For more information, see delete-replication-group in the *AWS Command Line Interface Reference*.

Taking a Final Backup (ElastiCache API)

You can create a final backup when deleting a Redis cluster or replication group using the ElastiCache API.

Topics

When Deleting a Redis Cluster

To create a final backup, use the `DeleteCacheCluster` ElastiCache API action with the following parameters.

- *`CacheClusterId`* – Name of the cluster being deleted.
- *`FinalSnapshotIdentifier`* – Name of the backup.

The following ElastiCache API action creates the backup `bkup-20150515-final` when deleting the cluster `myRedisCluster`.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DeleteCacheCluster
    &CacheClusterId=myRedisCluster
    &FinalSnapshotIdentifier=bkup-20150515-final
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
```

```
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

For more information, see DeleteCacheCluster in the *Amazon ElastiCache API Reference*.

When Deleting a Redis Replication Group

To create a final backup when deleting a replication group, use the `DeleteReplicationGroup` ElastiCache API action, with the following parameters.

- *`ReplicationGroupId`* – Name of the replication group being deleted.
- *`FinalSnapshotIdentifier`* – Name of the final backup.

The following ElastiCache API action creates the backup `bkup-20150515-final` when deleting the replication group `myReplGroup`.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DeleteReplicationGroup
    &FinalSnapshotIdentifier=bkup-20150515-final
    &ReplicationGroupId=myReplGroup
    &Version=2015-02-02
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20150202T192317Z
    &X-Amz-Credential=<credential>
```

For more information, see DeleteReplicationGroup in the *Amazon ElastiCache API Reference*.

Describing Backups

The following procedures show you how to display a list of your backups. If you desire, you can also view the details of a particular backup.

Describing Backups (Console)

To display backups 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. From the left navigation pane, choose **Backups**.
3. Use the **Filter** field to display *manual*, *automatic*, or *all* backups.
4. To see the details of a particular backup, select the box to the left of the backup's name.

Describing Backups (AWS CLI)

To display a list of backups and optionally details about a specific backup, use the `describe-snapshots` CLI operation.

Examples

The following command uses the parameter *`--max-records`* to list up to 20 backups associated with your account. Omitting the parameter *`--max-records`* lists up to 50 backups.

```
aws elasticache describe-snapshots --max-records 20
```

The following command uses the parameter *`--cache-cluster-id`* to list only the backups associated with the cluster `my-cluster`.

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

The following command uses the parameter *`--snapshot-name`* to display the details of the backup `my-backup`.

```
aws elasticache describe-snapshots --snapshot-name my-backup
```

For more information, see describe-snapshots in the AWS Command Line Interface Reference.

Describing Backups (ElastiCache API)

To display a list of backups, use the `DescribeSnapshots` action.

Examples

The following action uses the parameter *`MaxRecords`* to list up to 20 backups associated with your account. Omitting the parameter *`MaxRecords`* will list up to 50 backups.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeSnapshots
    &MaxRecords=20
```

```
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &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>
```

The following action uses the parameter *CacheClusterId* to list all backups associated with the cluster `MyCluster`.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeSnapshots
    &CacheClusterId=MyCluster
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &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>
```

The following action uses the parameter *SnapshotName* to display the details for the backup `MyBackup`.

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DescribeSnapshots
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
    &SnapshotName=MyBackup
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &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 DescribeSnapshots.

Copying a Backup

You can make a copy of any backup, whether it was created automatically or manually. You can also export your backup so you can access it from outside ElastiCache. For guidance on exporting your backup, see Exporting a Backup (p. 249).

The following procedures show you how to copy a backup.

Copying a Backup (Console)

To copy a backup (console)

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. On the ElastiCache console dashboard, choose **Backups**.
3. In the list of backups, choose the box to the left of the name of the backup you want to copy.
4. Select **Copy**.
5. In the **Create Copy of the Backup?** dialog box:
   a. In the **New backup name** box, type a name for your new backup.
   b. Leave the optional **Target S3 Bucket** box blank. This field should only be used to export your backup and requires special S3 permissions. For information on exporting a backup, see Exporting a Backup (p. 249).
   c. Select **Copy**.

Copying a Backup (AWS CLI)

To copy a backup, use the `copy-snapshot` command.

Parameters

- `--source-snapshot-name` – Name of the backup to be copied.
- `--target-snapshot-name` – Name of the backup's copy.
- `--target-bucket` – Reserved for exporting a backup. Do not use this parameter when making a copy of a backup. For more information, see Exporting a Backup (p. 249).

The following example makes a copy of an automatic backup.

For Linux, OS X, or Unix:

```
aws elasticache copy-snapshot \
    --source-snapshot-name automatic.my-redis-primary-2014-03-27-03-15 \
    --target-snapshot-name my-backup-copy
```

For Windows:

```
aws elasticache copy-snapshot ^
    --source-snapshot-name automatic.my-redis-primary-2014-03-27-03-15 ^
    --target-snapshot-name my-backup-copy
```

For more information, see `copy-snapshot` in the *AWS CLI*.

Copying a Backup (ElastiCache API)

To copy a backup, use the CopySnapshot action with the following parameters:

Parameters

- `SourceSnapshotName` – Name of the backup to be copied.
- `TargetSnapshotName` – Name of the backup's copy.
- `TargetBucket` – Reserved for exporting a backup. Do not use this parameter when making a copy of a backup. For more information, see Exporting a Backup (p. 249).

The following example makes a copy of an automatic backup.

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=CopySnapshot
    &SourceSnapshotName=automatic.my-redis-primary-2014-03-27-03-15
    &TargetSnapshotName=my-backup-copy
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &Version=2014-12-01
    &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 CopySnapshot in the *Amazon ElastiCache API Reference*.

Exporting a Backup

Amazon ElastiCache supports exporting your ElastiCache backup to an Amazon Simple Storage Service (Amazon S3) bucket, which gives you access to it from outside ElastiCache. You can export a backup using the ElastiCache console, the AWS CLI, or the ElastiCache API.

Exporting a backup can be helpful if you need to launch a cluster in a another region. You can export your data in one region, copy the .rdb file to the new region, and then use that .rdb file to seed the new cluster instead of waiting for the new cluster to populate through use. For information about seeding a new cluster, see Using a Backup to Seed a Cluster (p. 257). Another reason you might want to export your cluster's data is to use the .rdb file for offline processing.

Important

- The backup and the Amazon S3 bucket that you want to copy it to must be in the same region.
- Backups copied to an Amazon S3 bucket are unencrypted. We strongly recommend that you do not grant others access to the Amazon S3 bucket where you want to store your backups.

Before you can export a backup to an Amazon S3 bucket you must have an Amazon S3 bucket in the same region as the backup, and then grant ElastiCache access to the bucket. The first two steps show you how to do this.

Warning: Data Vulnerability
The following scenarios expose your data in ways you may not want.

- **When another person has access to the Amazon S3 bucket you exported your backup to.**

  To control access to your backups, only allow access to the Amazon S3 bucket to those who you want to access your data. For information about managing access to an Amazon S3 bucket, see Managing Access in the *Amazon S3 Developer Guide*.

- **When another person has permissions to use the CopySnapshot API.**

  Users or groups that have permissions to use the *CopySnapshot* API can create their own Amazon S3 buckets and copy backups to it. To control access to your backups, use an IAM policy to control who has the ability to use the *CopySnapshot* API. For more information about using IAM to control the use of ElastiCache APIs, see Authentication and Access Control for Amazon ElastiCache (p. 333) in the *ElastiCache User Guide*.

Topics

Step 1: Create an Amazon S3 Bucket

The following procedure creates an Amazon S3 bucket where you can export and store an ElastiCache backup.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.

2. Choose **Create Bucket**.
3. In **Create a Bucket - Select a Bucket Name and Region**, do the following:

   a. In **Bucket Name**, type a name for your Amazon S3 bucket.

      The name of your Amazon S3 bucket must be DNS-compliant. Otherwise, ElastiCache cannot access your backup file. The rules for DNS compliance are:

      - Names must be at least 3 and no more than 63 characters long.
      - Names must be a series of one or more labels separated by a period (.) where each label:
        - Starts with a lowercase letter or a number.
        - Ends with a lowercase letter or a number.
        - Contains only lowercase letters, numbers, and dashes.
      - Names cannot be formatted as an IP address (e.g., 192.0.2.0).

   b. In **Region**, choose the same region that your backup is in.
   c. Choose **Create**.

For more information about creating an Amazon S3 bucket, see Creating a Bucket in the *Amazon Simple Storage Service Console User Guide*.

Step 2: Grant ElastiCache Access to Your Amazon S3 Bucket

The following procedure grants Amazon ElastiCache access to your S3 bucket so it can copy a backup to the bucket.

> **Warning**
> Backups copied to an Amazon S3 bucket are unencrypted. Your data may be accessed by anyone with access to your Amazon S3 bucket. We strongly recommend that you set up IAM policies to prevent unauthorized access to this Amazon S3 bucket. For more information, see Managing Access in the *Amazon S3 Developer Guide*.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **All Buckets**, and then choose the name of the Amazon S3 bucket that you want the exported backup written to. This should be the S3 bucket you created in Step 1: Create an Amazon S3 Bucket (p. 249).
3. Choose **Properties**, and then choose **Permissions**.
4. Make sure that the bucket's region is the same as your backup's region. If it isn't, return to Step 1: Create an Amazon S3 Bucket (p. 249) and create a new bucket in the same region as the backup that you want to export.
5. Choose **Add more permissions**.
6. In **Grantee**, type the region's canonical id as shown in the following list:

   - China (Beijing) Region – `b14d6a125bdf69854ed8ef2e71d8a20b7c490f252229b806e514966e490b8d83`
   - Asia Pacific (Seoul) Region – `540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353`
   - AWS GovCloud (US) Region – `40fa568277ad703bd160f66ae4f83fc9dfdfd06c2f1b5060ca22442ac3ef8be6`

     > **Important**
     > The backup must be exported to an S3 bucket in AWS GovCloud (US).

- All other regions – `540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353`

7. Choose **List**, **Upload/Delete**, and **View Permissions**. ElastiCache must have these permissions to create an exported backup in the S3 bucket.
8. Choose **Save**.

Your Amazon S3 bucket is now ready for you to export an ElastiCache backup to using the ElastiCache console, the AWS CLI, or the ElastiCache API.

Step 3: Export an ElastiCache Backup

Now that you've created your S3 bucket and granted ElastiCache permissions to access it, use one of the following methods to export your backup.

Topics

Exporting an ElastiCache Backup (Console)

The following process uses the ElastiCache console to export a backup to an Amazon S3 bucket so that you can access it from outside ElastiCache. The Amazon S3 bucket must be in the same region as the ElastiCache backup.

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the ElastiCache console dashboard, select **Backups**.
3. From the list of backups, select the box to the left of the name of the backup you want to export.
4. Select **Copy**.
5. In **Create a Copy of the Backup?**, do the following:

   a. In **New backup name** box, type a name for your new backup.

      The name must be between 1 and 1,000 characters and able to be UTF-8 encoded.

      ElastiCache adds an instance identifier and `.rdb` to the value that you enter here. For example, if you enter `my-exported-backup`, ElastiCache creates `my-exported-backup-0001.rdb`.

   b. From the **Target S3 Location** list, select the name of the Amazon S3 bucket that you want to copy your backup to (the bucket that you created in Step 1: Create an Amazon S3 Bucket (p. 249)).

      The **Target S3 Location** must be an Amazon S3 bucket in the backup's region with **List**, **Upload/Delete**, and **View Permissions** permissions granted to ElastiCache for the backup export process to succeed. For more information, see Step 2: Grant ElastiCache Access to Your Amazon S3 Bucket (p. 250).

   c. Choose **Copy**.

Note

If your S3 bucket does not have the permissions needed for ElastiCache to export a backup to it, you will receive one of the following error messages. Add the permissions specified and retry exporting your backup.

- ElastiCache has not been granted READ permissions %s on the S3 Bucket.

  Solution: Add List and Read permissions on the bucket.
- ElastiCache has not been granted WRITE permissions %s on the S3 Bucket.

  Solution: Add Upload/Delete permissions on the bucket.
- ElastiCache has not been granted READ_ACP permissions %s on the S3 Bucket.

  Solution: Add **View Permissions** on the bucket.

For information on adding permissions, see Step 2: Grant ElastiCache Access to Your Amazon S3 Bucket (p. 250)

If you want to copy your backup to another region, use Amazon S3 to copy it. For more information, see Copying an Object in the *Amazon Simple Storage Service Console User Guide*.

Exporting an ElastiCache Backup (AWS CLI)

Export the backup to an Amazon S3 bucket using the `copy-snapshot` CLI operation with the following parameters:

Parameters

- `--source-snapshot-name` – Name of the backup to be copied.
- `--target-snapshot-name` – Name of the backup's copy.

  The name must be between 1 and 1,000 characters and able to be UTF-8 encoded.

  ElastiCache adds an instance identifier and `.rdb` to the value you enter here. For example, if you enter `my-exported-backup`, ElastiCache creates `my-exported-backup-0001.rdb`.
- `--target-bucket` – Name of the Amazon S3 bucket where you want to export the backup. A copy of the backup is made in the specified bucket.

  The *`--target-bucket`* must be an Amazon S3 bucket in the backup's region with **List**, **Upload/Delete**, and **View Permissions** permissions granted to ElastiCache for the backup export process to succeed. For more information, see Step 2: Grant ElastiCache Access to Your Amazon S3 Bucket (p. 250).

The following command copies a backup to my-s3-bucket.

For Linux, OS X, or Unix:

```
aws elasticache copy-snapshot \
    --source-snapshot-name automatic.my-redis-primary-2016-06-27-03-15 \
    --target-snapshot-name my-exported-backup \
    --target-bucket my-s3-bucket
```

For Windows:

```
aws elasticache copy-snapshot ^
    --source-snapshot-name automatic.my-redis-primary-2016-06-27-03-15 ^
    --target-snapshot-name my-exported-backup ^
    --target-bucket my-s3-bucket
```

Note

If your S3 bucket does not have the permissions needed for ElastiCache to export a backup to it, you will receive one of the following error messages. Add the permissions specified and retry exporting your backup.

- ElastiCache has not been granted READ permissions %s on the S3 Bucket.

  Solution: Add **List** and **Read** permissions on the bucket.
- ElastiCache has not been granted WRITE permissions %s on the S3 Bucket.

  Solution: Add **Upload/Delete** permissions on the bucket.
- ElastiCache has not been granted READ_ACP permissions %s on the S3 Bucket.

  Solution: Add **View Permissions** on the bucket.

For information on adding permissions, see Step 2: Grant ElastiCache Access to Your Amazon S3 Bucket (p. 250)

For more information, see copy-snapshot in the *AWS Command Line Interface Reference*.

If you want to copy your backup to another region, use Amazon S3 copy. For more information, see Copying an Object in the *Amazon Simple Storage Service Console User Guide*.

Exporting an ElastiCache Backup (ElastiCache API)

Export the backup to an Amazon S3 bucket using the `CopySnapshot` API operation with these parameters.

Parameters

- `SourceSnapshotName` – Name of the backup to be copied.
- `TargetSnapshotName` – Name of the backup's copy.

  The name must be between 1 and 1,000 characters and able to be UTF-8 encoded.

  ElastiCache will add an instance identifier and `.rdb` to the value you enter here. For example, if you enter `my-exported-backup`, you will get `my-exported-backup-0001.rdb`.
- `TargetBucket` – Name of the Amazon S3 bucket where you want to export the backup. A copy of the backup is made in the specified bucket.

  The *`TargetBucket`* must be an Amazon S3 bucket in the backup's region with **List**, **Upload/Delete**, and **View Permissions** permissions granted to ElastiCache for the backup export process to succeed. For more information, see Step 2: Grant ElastiCache Access to Your Amazon S3 Bucket (p. 250).

The following example makes a copy of an automatic backup to the Amazon S3 bucket `my-s3-bucket`.

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=CopySnapshot
    &SourceSnapshotName=automatic.my-redis-primary-2016-06-27-03-15
    &TargetBucket=my-s3-bucket
    &TargetSnapshotName=my-backup-copy
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &Version=2016-01-01
    &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>
```

Note

If your S3 bucket does not have the permissions needed for ElastiCache to export a snapshot to it, you will receive one of the following error messages. Add the permissions specified and retry exporting your backup.

- ElastiCache has not been granted READ permissions %s on the S3 Bucket.

  Solution: Add **List** and **Read** permissions on the bucket.
- ElastiCache has not been granted WRITE permissions %s on the S3 Bucket.

  Solution: Add **Upload/Delete** permissions on the bucket.
- ElastiCache has not been granted READ_ACP permissions %s on the S3 Bucket.

  Solution: Add **View Permissions** on the bucket.

For information on adding permissions, see Step 2: Grant ElastiCache Access to Your Amazon S3 Bucket (p. 250)

For more information, see CopySnapshot in the *Amazon ElastiCache API Reference*.

If you want to copy your backup to another region, use Amazon S3 copy to copy the exported backup to the Amazon S3 bucket in another region. For more information, see Copying an Object in the *Amazon Simple Storage Service Console User Guide*.

Restoring From a Backup

You can restore the data from a backup into a new cluster at any time. By default, the new cluster will have the same configuration that the source cluster did when the backup was created; however, you can override some of the parameters, such as node size.

During the restore operation, ElastiCache creates the new cluster, and then populates the cache with data from the backup file. When this process is complete, the Redis cache is warmed up and the cluster is ready to accept requests.

The following procedures show you how to restore a backup to a new cluster.

Restoring From a Backup (Console)

You can restore a cluster from a backup using the ElastiCache console.

To restore a backup to a new 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 left navigation pane, choose **Backups**.
3. In the list of backups, select the backup you want to restore from.
4. Select **Restore**.
5. Complete the **Restore Cluster** dialog box.

   [Optional] You can customize the new cluster by selecting new values for **Instance Type**, **Cache Port**, and some other properties.
6. When the settings are as you want them, choose **Launch Cluster**.

Restoring From a Backup (AWS CLI)

To restore data from a backup into a new cluster, use the create-cache-cluster CLI operation with the following parameters:

- `--snapshot-name` – Name of the backup to restore from.
- `--cache-cluster-id` – Name of the new, restored cache cluster.

The following example creates a new cache cluster named *my-restored-redis* and restores the data from *my-manual-backup* into it.

For Linux, OS X, or Unix:

```
aws elasticache create-cache-cluster \
    --cache-cluster-id my-restored-redis \
    --snapshot-name my-manual-backup
```

For Windows:

```
aws elasticache create-cache-cluster ^
    --cache-cluster-id my-restored-redis ^
    --snapshot-name my-manual-backup
```

For more information, see create-cache-cluster in the *AWS Command Line Interface Reference*.

Restoring From a Backup (ElastiCache API)

To restore data from a backup into a new cluster, use the CreateCacheCluster API operation with the following parameter:

- `SnapshotName` – Name of the backup to restore from.
- `CacheClusterId` – Name of the new, restored cache cluster.

The following example creates a new cluster named *my-restored-redis* and restores the data from *my-manual-backup*.

Example

```
https://elasticache.us-west-2.amazonaws.com/
    ?Action=CreateCacheCluster
    &CacheClusterId=my-restored-redis
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &SnapshotName=my-manual-backup
    &Timestamp=20141201T220302Z
    &Version=2015-02-02
    &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 CreateCacheCluster in the *Amazon ElastiCache API Reference*.

Using a Backup to Seed a Cluster

When you create a new Redis cluster, you can seed it with data from a Redis .rdb backup file. Seeding the cluster is useful if you currently manage a Redis instance outside of ElastiCache and want to populate your new ElastiCache cluster with your existing Redis data.

Important
You must ensure that your Redis backup data does not exceed the resources of the node. For example, you cannot upload an .rdb file with 5 GB of Redis data to a cache.m3.medium node that has 2.9 GB of memory.
If the backup is too large, the resulting cluster will have a status of restore-failed. If this happens, you must delete the cluster and start over.
For a complete listing of node types and specifications, see Redis Node-Type Specific Parameters (p. 305) and Amazon ElastiCache Product Features and Details.

The following topics walk you through migrating your non-ElastiCache Redis cluster to Amazon ElastiCache.

Topics

Step 1: Create a Redis Backup

To create the Redis backup from which you will seed your ElastiCache Redis instance

1. Connect to your existing Redis instance.
2. Run either the `BGSAVE` or `SAVE` command to create the backup.

   `BGSAVE` is asynchronous and does not block other clients while processing. For more information, see BGSAVE at the Redis website.

   `SAVE` is synchronous and blocks other processes until finished. For more information, see SAVE at the Redis website.

For additional information on creating a backup, see Redis Persistence at the Redis website.

Step 2: Upload Your Backup to Amazon S3

Once you have created the backup file, you need to upload it to an Amazon S3 bucket. For more information on this task, see the Amazon Simple Storage Service Getting Started Guide.

It is important that you note the path to your S3 bucket for the .rdb file. For example, if my bucket name was `myBucket` and the path was `myFolder/redis.rdb`, you would enter `myBucket/myFolder/redis.rdb`. You need this path to seed the new cluster with the data in this backup.

The name of your Amazon S3 bucket must be DNS-compliant. Otherwise, ElastiCache cannot access your backup file. The rules for DNS compliance are:

- Names must be at least 3 and no more than 63 characters long.
- Names must be a series of one or more labels separated by a period (.) where each label:

  - Starts with a lowercase letter or a number.
  - Ends with a lowercase letter or a number.
  - Contains only lowercase letters, numbers, and dashes.
- Names cannot be formatted as an IP address (e.g., 192.0.2.0).

For additional information, see Bucket Restrictions and Limitations in the *Amazon Simple Storage Service Developer Guide*.

We strongly recommend that you use an Amazon S3 bucket that is in the same region as your ElastiCache cluster. This approach will ensure the highest data transfer speed when ElastiCache reads your .rdb file from Amazon S3.

Step 3: Grant ElastiCache Read Access to the .rdb File

To grant ElastiCache read access to the backup file

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose **All Buckets**, and then choose the name of the S3 bucket that contains your .rdb file.
3. Choose the name of the folder that contains your .rdb file.
4. Choose the name of your .rdb backup file.
5. Choose the **Actions** drop-down menu, and then choose **Properties**.
6. In the **Grantee** box, type this email address: `aws-scs-s3-readonly@amazon.com`. The `aws-scs-s3-readonly@amazon.com` account is used exclusively for customers uploading Redis backup data from Amazon S3.

   Important
   For the following regions, connect to the canonical ID rather than `aws-scs-s3-readonly@amazon.com`:

   - China (Beijing) Region: `b14d6a125bdf69854ed8ef2e71d8a20b7c490f252229b806e514966e490b8d83`
   - EU (Frankfurt) Region: `540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353`
   - Asia Pacific (Seoul) Region: `540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353`
   - AWS GovCloud (US) Region: `40fa568277ad703bd160f66ae4f83fc9dfdfd06c2f1b5060ca22442ac3ef8be6`

     Note
     The backup must be located in an S3 bucket in AWS GovCloud (US) for you to download it to a Redis cluster in AWS GovCloud (US).
7. Choose **Open/Download**, and then choose **Save**.

Step 4: Seed the ElastiCache Cluster With the .rdb File Data

Now you are ready to create an ElastiCache cluster and seed it with the data from the .rdb file. To create the cluster, follow the directions at Creating a Cluster (p. 128). Be sure to choose Redis as your cluster engine.

The method you use to tell ElastiCache where to find the Redis backup you uploaded to Amazon S3 depends on the method you use to create the cluster:

- **Seed the ElastiCache Cluster With the .rdb File Data Using the ElastiCache Console**

  After you choose the Redis engine, on the **Specify Cluster Details** page, at the bottom of the **Configuration section**, locate **S3 Location of the Redis .rdb file** and type the Amazon S3 path for the backup you uploaded to your S3 bucket. The Amazon S3 path will look something like `myBucket/myFolder/myBackupFilename.rdb`.
- **Seed the ElastiCache Cluster With the .rdb File Data Using the AWS CLI**

  If you use the `aws elasticache create-cache-cluster` command, use the `--snapshot-arns` parameter to specify a fully qualified ARN. For example, `arn:aws:s3:::`*`myBucket`*`/`*`myFolder`*`/`*`myBackupFilename`*`.rdb`. The ARN must resolve to the backup file you stored in Amazon S3.
- **Seed the ElastiCache Cluster With the .rdb File Data Using the ElastiCache API**

  If you use the `CreateCacheCluster` ElastiCache API, use the `SnapshotArns` parameter to specify a fully qualified ARN. For example, `arn:aws:s3:::`*`myBucket`*`/`*`myFolder`*`/`*`myBackupFilename`*`.rdb`. The ARN must resolve to the backup file you stored in Amazon S3.

During the process of creating your cluster, the data in your Redis backup will be written to the cluster. You can monitor the progress by viewing the ElastiCache event messages. To do this, go to the ElastiCache console and choose **Cache Events**. You can also use the AWS ElastiCache command line interface or ElastiCache API to obtain event messages. For more information, see Viewing ElastiCache Events (p. 367).

Tagging Backups

Cost allocation tags are a means of tracking your costs across multiple AWS services by grouping your expenses on invoices by tag values. To learn more about cost allocation tags, see Use Cost Allocation Tags and Monitoring Costs with Cost Allocation Tags (p. 373).

Using the ElastiCache console, the AWS CLI, or ElastiCache API you can add, list, modify, remove, or copy cost allocation tags on your backups. For more information, see the following topics.

- Adding Tags to Your ElastiCache Resource (p. 375)
- Listing Your ElastiCache Resource's Tags (p. 377)
- Modifying Your ElastiCache Resource's Tags (p. 379)
- Removing Tags from Your ElastiCache Resource (p. 380)
- Copying Tags to Your ElastiCache Resource (p. 381)

Deleting a Backup

An automatic backup is automatically deleted when its retention limit expires. If you delete a cluster, all of its automatic backups are also deleted. If you delete a replication group, all of the automatic backups from the clusters in that group are also deleted.

ElastiCache provides a deletion API that lets you delete a backup at any time, regardless of whether the backup was created automatically or manually. (Since manual backups do not have a retention limit, manual deletion is the only way to remove them.)

You can delete a backup using the ElastiCache console, the AWS CLI, or the ElastiCache API.

Deleting a Backup (Console)

The following procedure deletes a backup using the ElastiCache console.

To delete a backup

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the left navigation pane, choose **Backups**.

   The Backups screen appears with a list of your backups.
3. Choose the box to the lert of the name of the backup you want to delete.
4. Choose **Delete**.

   The **Delete Backup** confirmation screen appears.
5. If you want to delete this backup, choose **Delete**. The status changes to *deleting*.

   If you want to keep this backup, choose **Cancel**.

Deleting a Backup (AWS CLI)

Use the delete-snapshot AWS CLI CLI operation with the following parameter to delete a backup .

- `--snapshot-name` – Name of the backup to be deleted.

The following code deletes the backup `myBackup`.

```
aws elasticache delete-snapshot --snapshot-name myBackup
```

For more information, see delete-snapshot in the *AWS Command Line Interface Reference*.

Deleting a Backup (ElastiCache API)

Use the DeleteSnapshot API operation with the following parameter to delete a backup .

- `SnapshotName` – Name of the backup to be deleted.

The following code deletes the backup `myBackup`.

```
https://elasticache.us-west-2.amazonaws.com/
```

```
?Action=DeleteSnapshot
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&SnapshotId=myBackup
&Timestamp=20150202T192317Z
&Version=2015-02-02
&X-Amz-Credential=<credential>
```

For more information, see DeleteSnapshot in the *Amazon ElastiCache API Reference*.

Redis Append Only Files (AOF)

By default, the data in a Redis node on ElastiCache resides only in memory, and is not persistent. If a node is rebooted, or if the underlying physical server experiences a hardware failure, the data in the cache is lost.

If you require data durability, you can enable the Redis append-only file feature (AOF). When this feature is enabled, the node writes all of the commands that change cache data to an append-only file. When a node is rebooted and the cache engine starts, the AOF is "replayed"; the result is a warm Redis cache with all of the data intact.

AOF is disabled by default. To enable AOF for a cluster running Redis, you must create a parameter group with the *`appendonly`* parameter set to yes, and then assign that parameter group to your cluster. You can also modify the *`appendfsync`* parameter to control how often Redis writes to the AOF file.

Important
Append-only files (AOF) are not supported for cache.t1.micro and cache.t2.* nodes. For nodes of these types, the *`appendonly`* parameter value is ignored.
For Multi-AZ replication groups, AOF is disabled.
AOF is not supported on Redis versions 2.8.22 and later.

Warning
AOF cannot protect against all failure scenarios. For example, if a node fails due to a hardware fault in an underlying physical server, ElastiCache will provision a new node on a different server. In this case, the AOF file will no longer be available and cannot be used to recover the data. Thus, Redis will restart with a cold cache.
For greater reliability and faster recovery, we recommend that you create one or more read replicas in different availability zones for your cluster, and enable Multi-AZ on the replication group instead of using AOF. AOF is disabled for Multi-AZ replication groups.
For more information on mitigating failures, see Mitigating Failures when Running Redis (p. 64).

For more information see:

- Redis Specific Parameters (p. 292)
- Replication: Multi-AZ with Automatic Failover (Redis) (p. 196)
- Mitigating Failures (p. 62)

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 Virtual Private Cloud (Amazon VPC) with ElastiCache (p. 316)

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 operaton (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 Virtual Private Cloud (Amazon VPC) with ElastiCache (p. 316).

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

Topics

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 Virtual Private Cloud (Amazon VPC) with ElastiCache (p. 316).

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 left 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, OS X, 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 in the following code example 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 Virtual Private Cloud (Amazon VPC) with ElastiCache (p. 316).

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 left 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
```

For more information, see describe-cache-security-groups.

Listing Available Security Groups (ElastiCache API)

Using the ElastiCache API, call `DescribeCacheSecurityGroups`.

Example

Line breaks in the following code example 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 Virtual Private Cloud (Amazon VPC) with ElastiCache (p. 316).

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 left navigation pane, choose **Security Groups**.

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

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 in the following code example 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 Virtual Private Cloud (Amazon VPC) with ElastiCache (p. 316).

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 left navigation pane, select **Security Groups**.
3. In the **Security Groups** list, select 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, select 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, OS X, 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:

```
SECGROUP  Name     Description
SECGROUP  default  default
      EC2-SECGROUP  myec2group  987654321021  authorizing
```

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.

Parameters and 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 by engine and version, see Memcached Specific Parameters (p. 286) and Redis Specific Parameters (p. 292).

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. 291) and Redis Node-Type Specific Parameters (p. 305).

Topics

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.

- 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 Redis version 2.8.6, you can only use parameter groups, default or custom, from the Redis 2.8 family, not the Redis 2.6 parameter group 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. 286) and Redis Specific Parameters (p. 292). For information on rebooting a cluster, go to Rebooting a Cluster (p. 152).

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. In the left hand navigation pane, select **Parameter Groups**.

   The Parameter Groups screen will appear with a list of all available parameter groups.
3. To create a parameter group, select **Create Parameter Group**.

   The **Create Parameter Group** screen will appear.
4. From the **Family** list, select the parameter group family that will be the template for your parameter group.

   The parameter group family, such as *redis2.8*, 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. For example, you cannot create a parameter group with the family *redis2.8* and use it with clusters running Redis version 2.6.
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 select the parameter group by its name. Therefore, we recommend that the name be informative and somehow identify the parameter group's family. For example, *Redis2-8-24-Custom*.

   #### 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, select **Create**.

   To terminate the process without creating the parameter group, select **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. 282).

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. For example, `redis2.8`.
- *`--description`* — A user supplied description for the parameter group.

The following example creates a parameter group named *myRedis28* using the redis2.8 family as the template.

For Linux, OS X, or Unix:

```
aws elasticache create-cache-parameter-group \
    --cache-parameter-group-name myRedis28  \
    --cache-parameter-group-family redis2.8 \
    --description "My first  parameter group"
```

For Windows:

```
aws elasticache create-cache-parameter-group ^
    --cache-parameter-group-name myRedis28  ^
    --cache-parameter-group-family redis2.8 ^
    --description "My first  parameter group"
```

The output from this command should look something like this.

```
CACHEPARAMETERGROUP  myRedis28              redis2.8      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. 282).

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, `redis2.8`.
- *`Description`* — A user supplied description for the parameter group.

The following example creates a parameter group named *myRedis28* using the redis2.8 family as the template.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=CreateCacheParameterGroup
   &CacheParameterGroupFamily=redis2.8
   &CacheParameterGroupName=myRedis28
   &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>myRedis28</CacheParameterGroupName>
      <CacheParameterGroupFamily>redis2.8</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. 282).

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. In the left hand navigation pane, select **Parameter Groups**.

   The Parameter Groups screen will appear with a list of all available parameter groups. Each row in the list displays the parameter group's name, family, and description.

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.

The following sample code lists the parameter group *myRedis28*.

For Linux, OS X, or Unix:

```
aws elasticache describe-cache-parameter-groups \
    --cache-parameter-group-name myRedis28
```

For Windows:

```
aws elasticache describe-cache-parameter-groups ^
    --cache-parameter-group-name myRedis28
```

The output of this command will look something like this, listing the name, family, and description for the parameter group.

```
CACHEPARAMETERGROUP  myRedis28  redis2.8  My Redis 2.8 parameter group
```

The following sample code lists up to 10 parameter groups.

```
aws elasticache describe-cache-parameter-groups --max-records 10
```

The output of this command will look something like this, listing the name, family, and description for each parameter group.

```
CACHEPARAMETERGROUP  myRedis28  redis2.8     My Redis 2.8 parameter group
CACHEPARAMETERGROUP  myMem14    memcached1.4 My Memcached 1.4 parameter group
```

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.

The following sample code lists the parameter group *myRedis28*.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameterGroups
   &CacheParameterGroupName=myRedis28
   &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>redis 2.8</CacheParameterGroupFamily>
        <Description>My Redis 2.8 parameter group</Description>
      </CacheParameterGroup>
    </CacheParameterGroups>
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>
```

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 Redis 2.8 parameter group</Description>
      </CacheParameterGroup>
      <CacheParameterGroup>
        <CacheParameterGroupName>myMem14</CacheParameterGroupName>
        <CacheParameterGroupFamily>memcached1.4</CacheParameterGroupFamily>
        <Description>My 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. In the left hand navigation pane, select **Parameter Groups**.

   The Parameter Groups screen will appear with a list of all available parameter groups.
3. Select the parameter group for which you want to list the parameters and values by selecting 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`.

The following sample code list all the parameters and their values for the parameter group *myRedis28*.

For Linux, OS X, or Unix:

```
aws elasticache describe-cache-parameters \
    --cache-parameter-group-name myRedis28
```

For Windows:

```
aws elasticache describe-cache-parameters ^
    --cache-parameter-group-name myRedis28
```

The output of this command will look something like this.

```
CACHEPARAMETER  Parameter Name                 Parameter Value  Source  Data
 Type  Is Modifiable  Minimum Version
CACHEPARAMETER  backlog_queue_limit            1024             system
 integer    false          1.4.5
CACHEPARAMETER  binding_protocol               auto             system
 string     false          1.4.5
CACHEPARAMETER  cas_disabled                   0                system
 boolean    true           1.4.5
CACHEPARAMETER  chunk_size                     48               system
 integer    true           1.4.5
CACHEPARAMETER  chunk_size_growth_factor       1.25             system
 float      true           1.4.5
```

```
CACHEPARAMETER  error_on_memory_exhausted       0               system
 boolean     true           1.4.5
CACHEPARAMETER  large_memory_pages              0               system
 boolean     false          1.4.5
(...sample truncated...)
```

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.

The following sample code list all the parameters for the parameter group *myRedis28*.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DescribeCacheParameters
   &CacheParameterGroupName=myRedis28
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T192317Z
   &Version=2015-02-02
   &X-Amz-Credential=<credential>
```

The response from this action will look something like this. This response has been truncated.

```
<DescribeCacheParametersResponse xmlns="http://elasticache.amazonaws.com/
doc/2013-06-15/">
  <DescribeCacheParametersResult>
    <CacheClusterClassSpecificParameters>
      <CacheNodeTypeSpecificParameter>
        <DataType>integer</DataType>
        <Source>system</Source>
        <IsModifiable>false</IsModifiable>
        <Description>The maximum configurable amount of memory to use to
 store items, in megabytes.</Description>
        <CacheNodeTypeSpecificValues>
          <CacheNodeTypeSpecificValue>
            <Value>1000</Value>
            <CacheClusterClass>cache.c1.medium</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>6000</Value>
            <CacheClusterClass>cache.c1.xlarge</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>7100</Value>
            <CacheClusterClass>cache.m1.large</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>1300</Value>
            <CacheClusterClass>cache.m1.small</CacheClusterClass>
          </CacheNodeTypeSpecificValue>

...output omitted...
```

```
        </CacheNodeTypeSpecificValues>
        <AllowedValues>1-100000</AllowedValues>
        <ParameterName>max_cache_memory</ParameterName>
        <MinimumEngineVersion>1.4.5</MinimumEngineVersion>
      </CacheNodeTypeSpecificParameter>
      <CacheNodeTypeSpecificParameter>
        <DataType>integer</DataType>
        <Source>system</Source>
        <IsModifiable>false</IsModifiable>
        <Description>The number of memcached threads to use.</Description>
        <CacheNodeTypeSpecificValues>
          <CacheNodeTypeSpecificValue>
            <Value>2</Value>
            <CacheClusterClass>cache.c1.medium</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>8</Value>
            <CacheClusterClass>cache.c1.xlarge</CacheClusterClass>
          </CacheNodeTypeSpecificValue>

...output omitted...

        </CacheNodeTypeSpecificValues>
        <AllowedValues>1-8</AllowedValues>
        <ParameterName>num_threads</ParameterName>
        <MinimumEngineVersion>1.4.5</MinimumEngineVersion>
      </CacheNodeTypeSpecificParameter>
    </CacheClusterClassSpecificParameters>
    <Parameters>
      <Parameter>
        <ParameterValue>1024</ParameterValue>
        <DataType>integer</DataType>
        <Source>system</Source>
        <IsModifiable>false</IsModifiable>
        <Description>The backlog queue limit.</Description>
        <AllowedValues>1-10000</AllowedValues>
        <ParameterName>backlog_queue_limit</ParameterName>
        <MinimumEngineVersion>1.4.5</MinimumEngineVersion>
      </Parameter>
      <Parameter>
        <ParameterValue>auto</ParameterValue>
        <DataType>string</DataType>
        <Source>system</Source>
        <IsModifiable>false</IsModifiable>
        <Description>Binding protocol.</Description>
        <AllowedValues>auto,binary,ascii</AllowedValues>
        <ParameterName>binding_protocol</ParameterName>
        <MinimumEngineVersion>1.4.5</MinimumEngineVersion>
      </Parameter>

...output omitted...

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

For more information, see `DescribeCacheParameters`.

Modifying a Parameter Group

Important
You cannot modify any default parameter group.

You can modify some parameter values in a parameter group. These parameter values are applied to clusters associated with the parameter group. For more information on when a parameter value change is applied to a parameter group, see Memcached Specific Parameters (p. 286) and Redis Specific Parameters (p. 292).

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. In the left hand navigation pane, select **Parameter Groups**.

   The Parameter Groups screen will appear with a list of all available parameter groups.
3. Select the parameter group you want to modify by selecting 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, select **Edit Parameters**.

   The **Edit Parameter Group:** screen appears.
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. Select **Save Changes**.
7. Find the name of the parameter you changed in one of these topics:

   - Memcached Specific Parameters (p. 286)
   - Redis Specific Parameters (p. 292)

   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. 152).

Modifying a Parameter Group (AWS CLI)

To change a parameter's value using the AWS CLI, use the command `modify-cache-parameter-group`.

The following sample code sets the value of *`binding_protocol`* to `ascii` in the *MyMemPG* parameter group. You would use the same code, changing only the name of the parameter group, the name of the parameter, and the new value for the parameter.

For Linux, OS X, or Unix:

```
aws elasticache modify-cache-parameter-group \
    --cache-parameter-group-name MyMemPG \
    --parameter-name-values
 ParameterName=binding_protocol,ParameterValue=ascii
```

For Windows:

```
aws elasticache modify-cache-parameter-group ^
    --cache-parameter-group-name MyMemPG ^
    --parameter-name-values
 ParameterName=binding_protocol,ParameterValue=ascii
```

For more information, see `modify-cache-parameter-group`.

Find the name of the parameter you changed in one of these topics:

- Memcached Specific Parameters (p. 286)
- Redis Specific Parameters (p. 292)

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. 152).

Modifying a Parameter Group (ElastiCache API)

To change a parameter group's parameter values using the ElastiCache API, use the `ModifyCacheParameterGroup` action.

The following sample code sets the value of *`binding_protocol`* to `ascii` in the *myMemPG* parameter group. You would use the same code, changing only the name of the parameter group, the name of the parameter, and the new value for the parameter.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ModifyCacheParameterGroup
   &CacheParameterGroupName=MyMemPG
   &ParameterNameValues.member.1.ParameterName=binding_protocol
   &ParameterNameValues.member.1.ParameterValue=ascii
   &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 takes place only after a reboot, reboot each cluster that uses the changed parameter group. For more information, see Rebooting a Cluster (p. 152).

Find the name of the parameter you changed in one of these topics:

- Memcached Specific Parameters (p. 286)
- Redis Specific Parameters (p. 292)

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. 152).

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. In the left hand navigation pane, select **Parameter Groups**.

   The Parameter Groups screen will appear with a list of all available parameter groups.
3. Select the parameter groups you want to delete by selecting the box to the left of the parameter group's name.

   The **Delete** button will become active.
4. Select **Delete**.

   The **Delete Parameter Groups** confirmation screen will appear.
5. To delete the parameter groups, select **Delete**.

   To keep the parameter groups, select **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 *myRedis28* parameter group.

For Linux, OS X, or Unix:

```
aws elasticache delete-cache-parameter-group \
    --cache-parameter-group-name myRedis28
```

For Windows:

```
aws elasticache delete-cache-parameter-group ^
    --cache-parameter-group-name myRedis28
```

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.

The following sample code deletes the *myRedis28* parameter group.

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=DeleteCacheParameterGroup
   &CacheParameterGroupName=myRedis28
   &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 (`default.memcached1.4`) 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.

Topics

Memcached 1.4.24 Added Parameters

For Memcached 1.4.24, the following additional parameters are supported.

Name	Details	Description
`disable_flush_all`	Default: 0 (disabled) Type: boolean Modifiable: Yes Changes Take Effect: At launch	Add parameter (*`-F`*) to disable flush_all. Useful if you never want to be able to run a full flush on production instances. Values: 0, 1 (user can do a *`flush_all`* when the value is 0).
`hash_algorithm`	Default: jenkins Type: string Modifiable: Yes Changes Take Effect: At launch	The hash algorithm to be used. Permitted values: murmur3 and jenkins.
`lru_crawler`	Default: 0 (disabled) Type: boolean Modifiable: Yes Changes Take Effect: After restart **Note** You can temporarily enable *`lru_crawler`* at runtime from the command line. For more information, see the	Cleans slab classes of items that have expired. This is a low impact process that runs in the background. Currently requires initiating a crawl using a manual command. To temporarily enable, run `lru_crawler enable` at the command line. `lru_crawler 1,3,5` crawls slab classes 1, 3, and 5 looking for expired items to add to the freelist. Values: 0,1 **Note** Enabling *`lru_crawler`* at the command line enables

Name	Details	Description
	Description column.	the crawler until either disabled at the command line or the next reboot. To enable permanently, you must modify the parameter value. For more information, see Modifying a Parameter Group (p. 282).
lru_maintainer	Default: 0 (disabled) Type: boolean Modifiable: Yes Changes Take Effect: At launch	A background thread that shuffles items between the LRU's 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.

Parameters added in Memcached 1.4.14

Name	Description
config_max	Default: 16 Type: integer Modifiable: No The maximum number of ElastiCache configuration entries.
config_size_max	Default: 65536 Type: integer Modifiable: No The maximum size of the configuration entries, in bytes.

Name	Details	Description
hashpower_init	Default: 16 Type: integer Modifiable: No	The initial size of the ElastiCache hash table, expressed as a power of two. The default is 16 (2^16), or 65536 keys.
maxconns_fast	Default: 0 (false) Type: Boolean Modifiable: Yes Changes Take Effect: After restart	Changes 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.
slab_automove	Default: 0 Type: integer Modifiable: Yes Changes Take Effect: After restart	Adjusts the slab automove algorithm: If this parameter is set to 0 (zero), the automove algorithm is disabled. If it is set to 1, ElastiCache takes a slow, conservative approach to automatically moving slabs. If it is set to 2, ElastiCache aggressively moves slabs whenever there is an eviction. (This mode is not recommended except for testing purposes.)
slab_reassign	Default: 0 (false) Type: Boolean Modifiable: Yes Changes Take Effect: After restart	Enable or disable slab reassignment. If this parameter is set to 1, you can use the "slabs reassign" command to manually reassign memory.

Memcached 1.4.5 Supported Parameters

For Memcached 1.4.5, the following parameters are supported.

Parameters added in Memcached 1.4.5

Name	Details	Description
backlog_queue_limit	Default: 1024 Type: integer Modifiable: No	The backlog queue limit.
binding_protocol	Default: auto	The binding protocol.

Name	Details	Description
	Type: string Modifiable: Yes Changes Take Effect: After restart	Permissible values are: `ascii` and `auto`. For guidance on modifying the value of *`binding_protocol`*, see Modifying a Parameter Group (p. 282).
`cas_disabled`	Default: 0 (false) Type: Boolean Modifiable: Yes Changes Take Effect: After restart	If `1` (true), check and set (CAS) operations will be disabled, and items stored will consume 8 fewer bytes than with CAS enabled.
`chunk_size`	Default: 48 Type: integer Modifiable: Yes Changes Take Effect: After restart	The minimum amount, in bytes, of space to allocate for the smallest item's key, value, and flags.
`chunk_size_growth_factor`	Default: 1.25 Type: float Modifiable: Yes Changes Take Effect: After restart	The growth factor that controls the size of each successive Memcached chunk; each chunk will be *`chunk_size_growth_factor`* times larger than the previous chunk.
`error_on_memory_exhausted`	Default: 0 (false) Type: Boolean Modifiable: Yes Changes Take Effect: After restart	If `1` (true), when there is no more memory to store items, Memcached will return an error rather than evicting items.
`large_memory_pages`	Default: 0 (false) Type: Boolean Modifiable: No	If `1` (true), ElastiCache will try to use large memory pages.
`lock_down_paged_memory`	Default: 0 (false) Type: Boolean Modifiable: No	If `1` (true), ElastiCache will lock down all paged memory.
`max_item_size`	Default: 1048576 Type: integer Modifiable: Yes Changes Take Effect: After restart	The size, in bytes, of the largest item that can be stored in the cluster.

Name	Details	Description
max_simultaneous_connections	Default: 65000 Type: integer Modifiable: No	The maximum number of simultaneous connections.
maximize_core_file_limit	Default: 0 (false) Type: Boolean Modifiable: Changes Take Effect: No	If `1` (true), ElastiCache will maximize the core file limit.
memcached_connections_overhead	Default: 100 Type: integer Modifiable: Yes Changes Take Effect: After restart	The amount of memory to be reserved for Memcached connections and other miscellaneous overhead. For information about this parameter, see Memcached Connection Overhead (p. 290).
requests_per_event	Default: 20 Type: integer Modifiable: No	The maximum number of requests per event for a given connection. This limit is required to prevent resource starvation.

Memcached Connection Overhead

On each node, the memory made available for storing items is the total available memory on that node (which is stored in the *max_cache_memory* parameter) minus the memory used for connections and other overhead (which is stored in the *memcached_connections_overhead* parameter). For example, a node of type `cache.m1.small` has a *max_cache_memory* of 1300MB. With the default *memcached_connections_overhead* value of 100MB, the Memcached process will have 1200MB available to store items.

The default values for the *memcached_connections_overhead* parameter satisfy most use cases; however, the required amount of allocation for connection overhead can vary depending on multiple factors, including request rate, payload size, and the number of connections.

You can change the value of the *memcached_connections_overhead* to better suit the needs of your application. For example, increasing the value of the *memcached_connections_overhead* parameter will reduce the amount of memory available for storing items and provide a larger buffer for connection overhead. Decreasing the value of the *memcached_connections_overhead* parameter will give you more memory to store items, but can increase your risk of swap usage and degraded performance. If you observe swap usage and degraded performance, try increasing the value of the *memcached_connections_overhead* parameter.

Important

For the `cache.t1.micro` node type, the value for *memcached_connections_overhead* is determined as follows:

- If you cluster is using the default parameter group, ElastiCache will set the value for *memcached_connections_overhead* to 13MB.
- If your cluster is using a parameter group that you have created yourself, you can set the value of *memcached_connections_overhead* to a value of your choice.

Memcached Node-Type Specific Parameters

Although most parameters have a single value, some parameters have different values depending on the node type used. The following table shows the default values for the *max_cache_memory* and *num_threads* parameters for each node type. The values on these parameters cannot be modified.

Node Type-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

Redis Specific Parameters

If you do not specify a parameter group for your Redis cluster, then a default parameter group will be used (either `default.redis2.6` or `default.redis2.8`). You cannot change the values of any parameters in the default parameter group; however, you can create a custom parameter group and assign it to your cluster at any time as long as the values of conditionally modifiable parameters are the same in both parameter groups.

Topics

Note

Because the newer Redis versions provide a better and more stable user experience, Redis versions 2.6.13, 2.8.6, and 2.8.19 are deprecated from the ElastiCache console. While we recommend against it, if you must use one of these older Redis versions, you can use the AWS CLI or ElastiCache API.
For more information see the following topics:

	AWS AWS CLI	ElastiCache API
Create Cluster	Creating a Cache Cluster (AWS CLI) (p. 137) #	Creating a Cache Cluster (ElastiCache API) (p. 140) #
Modify Cluster	Modifying a Cache Cluster (AWS CLI) (p. 150) #	Modifying a Cache Cluster (ElastiCache API) (p. 150) #
Create Replication Group	Creating a Redis (cluster mode disabled) Replication Group from Scratch (AWS CLI) (p. 210) Creating a Redis (cluster mode enabled) Replication Group from Scratch (AWS CLI) (p. 215)	Creating a Redis (cluster mode disabled) Replication Group from Scratch (ElastiCache API) (p. 212) Creating a Redis (cluster mode enabled) Replication Group from Scratch (ElastiCache API) (p. 218)
Modify Replication Group	Modifying a Replication Group (AWS CLI) (p. 226) #	Modifying a Replication Group (ElastiCache API) (p. 226) #
# Cannot be used to create a Redis (cluster mode enabled) cluster or replication group.		
# Cannot be used to modify a Redis (cluster mode enabled) cluster or replication group.		

Redis 3.2.4 Parameter Changes

Beginning with Redis 3.2.4, there are two default parameter groups.

- *default.redis3.2* – When using Redis 3.2.4, specify this parameter group or one derived from it, if you want to create a Redis (cluster mode disabled) replication group and still use the additional features of Redis 3.2.4.
- *default.redis3.2.cluster.on* – Specify this parameter group or one derived from it, when you want to create a Redis (cluster mode enabled) replication group.

Topics

New Parameters for Redis 3.2.4

For Redis 3.2.4, the following additional parameters are supported.

Name	Details	Description
list-max-ziplist-size	Default: -2 Type: integer Modifiable: No	Lists are encoded in a special way to save space. The number of entries allowed per internal list node can be specified as a fixed maximum size or a maximum number of elements. For a fixed maximum size, use -5 through -1, meaning: • -5: max size: 64 Kb - not recommended for normal workloads • -4: max size: 32 Kb - not recommended • -3: max size: 16 Kb - not recommended • -2: max size: 8 Kb - recommended • -1: max size: 4 Kb - recommended • Positive numbers mean store up to exactly that number of elements per list node.
list-compress-depth	Default: 0 Type: integer Modifiable: Yes Changes Take Effect: Immediately	Lists may also be compressed. Compress depth is the number of quicklist ziplist nodes from each side of the list to exclude from compression. The head and tail of the list are always uncompressed for fast push and pop operations. Settings are: • 0: Disable all compression. • 1: Start compressing with the 1st node in from the head and tail. [head]->node->node->...->node->[tail] All nodes except [head] and [tail] compress. • 2: Start compressing with the 2nd node in from the head and tail. [head]->[next]->node->node->...->node->[prev]->[tail]

Name	Details	Description
		[head], [next], [prev], [tail] do not compress. All other nodes compress. • Etc.
`cluster-enabled`	Default: yes/no # Type: boolean Modifiable: No	Indicates whether this is a Redis (cluster mode enabled) replication group in cluster mode (yes) or a Redis (cluster mode enabled) replication group in non-cluster mode (no). Redis (cluster mode enabled) replication groups in cluster mode can partition their data across up to 15 node groups. # Redis 3.2.*x* has two default parameter groups. • *`default.redis3.2`* – default value no. • *`default.redis3.2.cluster.on`* – default value `yes`. .
`cluster-require-full-coverage`	Default: no Type: boolean Modifiable: yes Changes Take Effect: Immediately	By default Redis (cluster mode enabled) nodes in cluster mode stop accepting queries if they detect there is at least one hash slot uncovered (no available node is serving it). This way if the cluster is partially down, the cluster becomes unavailable. It automatically becomes available again as soon as all the slots are covered again. However, sometimes you want the subset of the cluster which is working to continue to accept queries for the part of the key space that is still covered. In order to do so, just set the *`cluster-require-full-coverage`* option to `no`.
`hll-sparse-max-bytes`	Default: 3000 Type: integer Modifiable: Yes Changes Take Effect: Immediately	HyperLogLog sparse representation bytes limit. The limit includes the 16 byte header. When a HyperLogLog using the sparse representation crosses this limit, it is converted into the dense representation. A value greater than 16000 is not recommended, because at that point the dense representation is more memory efficient. We recommend a value of ~3000 in order to have the benefits of the space efficient encoding without slowing down PFADD too much, which is O(N) with the sparse encoding. The value can be raised to ~10000 when CPU is not a concern, but space is, and the data set is composed of many HyperLogLogs with cardinality in the 0 - 15000 range.

Parameters Changed in Redis 3.2.4 (Enhanced)

For Redis 3.2.4, the following parameters were changed beginning in Redis 3.2.4.

Name	Details	Change
`appendonly`	Default: off Modifiable: No	If you want to upgrade from an earlier Redis version, you must first turn *`appendonly`* off.
`appendfsync`	Default: off Modifiable: No	If you want to upgrade from an earlier Redis version, you must first turn *`appendfsync`* off.
`repl-timeout`	Default: 60 Modifiable: No	Is now unmodifiable with a default of 60.
`tcp-keepalive`	Default: 300	Default was 0.
`list-max-ziplist-entries`		Parameter is no longer available.
`list-max-ziplist-value`		Parameter is no longer available.

Redis 2.8.24 (Enhanced) Added Parameters

For Redis 2.8.24, there are no additional parameters supported.

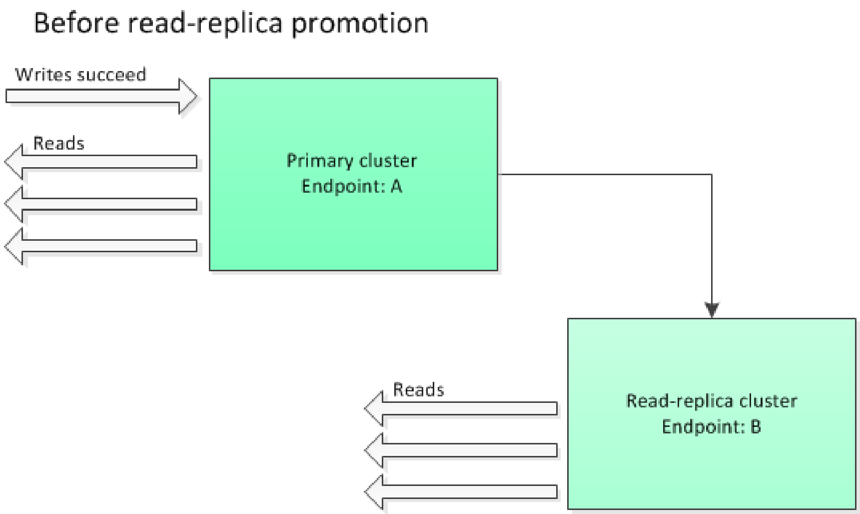
Redis 2.8.23 (Enhanced) Added Parameters

For Redis 2.8.23, the following additional parameter is supported.

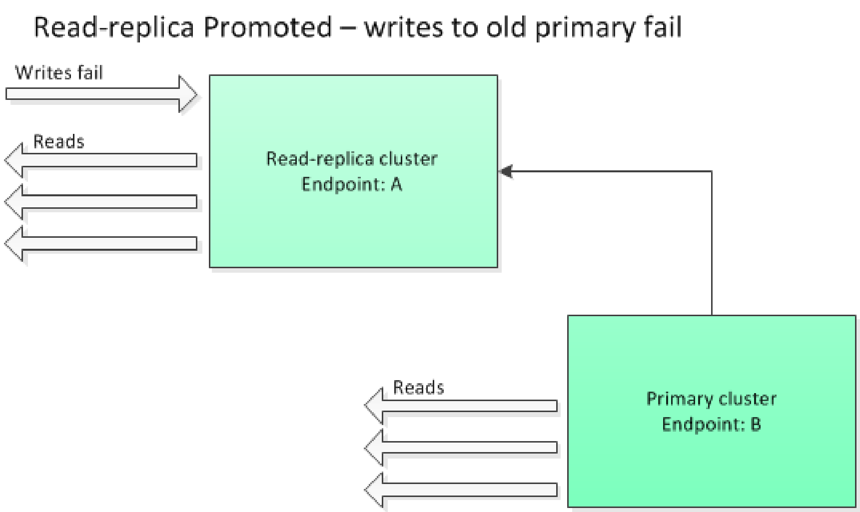
Name	Details	Description
`close-on-slave-write`	Default: yes Type: string (yes/no) Modifiable: Yes Changes Take Effect: Immediately	If enabled, clients who attempt to write to a read-only replica will be disconnected.

close-on-slave-write

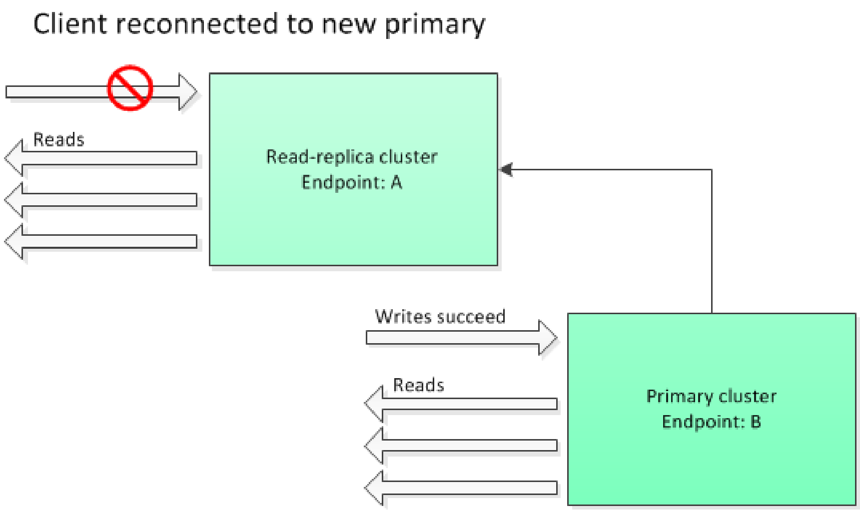
The *`close-on-slave-write`* parameter is introduced by ElastiCache to give you more control over how your cluster responds when a primary node and a read replica node swap roles due to promoting a read replica to primary.



If the read-replica cluster is promoted to primary for any reason other than a Multi-AZ enabled replication group failing over, the client will continue trying to write to endpoint A. Because endpoint A is now the endpoint for a read-replica, these writes will fail. This is the behavior for Redis prior to ElastiCache introducing `close-on-slave-write` and the behavior if you disable `close-on-slave-write`.



With `close-on-slave-write` enabled, any time a client attempts to write to a read-replica, the client connection to the cluster is closed. Your application logic should detect the disconnection, check the DNS table, and reconnect to the primary endpoint, which now would be endpoint B.



Why disable `close-on-slave-write`?

If disabling *close-on-slave-write* results in writes to the cluster failing, why would you want to disable *close-on-slave-write*?

As previously mentioned, with *close-on-slave-write* enabled, any time a client attempts to write to a read-replica the client connection to the cluster is closed. Since establishing a new connection to the node takes time, disconnecting and reconnecting as a result of a write request to the replica would also impact the latency of read requests that were served through the same connection, until a new connection is established. Therefore, if your application is especially read-heavy or very latency-sensitive, you my prefer to keep your clients connected so as to not degrade read performance.

Redis 2.8.22 (Enhanced) Added Parameters

For Redis 2.8.22, there are no additional parameters supported.

Important

- Beginning with Redis version 2.8.22, *repl-backlog-size* applies to the primary cluster as well as to replica clusters.
- Beginning with Redis version 2.8.22, the *repl-timeout* parameter is not supported. If it is changed, ElastiCache will overwrite with the default (60s), as we do with *appendonly*.

The following parameters are no longer supported.

- *appendonly*
- *appendfsync*
- *repl-timeout*

Redis 2.8.21 Added Parameters

For Redis 2.8.21, there are no additional parameters supported.

Redis 2.8.19 Added Parameters

For Redis 2.8.19, there are no additional parameters supported.

Redis 2.8.6 Added Parameters

For Redis 2.8.6, the following additional parameters are supported.

Name	Details	Description
min-slaves-max-lag	Default: 10 Type: integer Modifiable: Yes Changes Take Effect: Immediately	The number of seconds within which the primary node must receive a ping request from a read replica. If this amount of time passes and the primary does not receive a ping, then the replica is no longer considered available. If the number of available replicas drops below min-slaves-to-write, then the primary will stop accepting writes at that point. If either this parameter or min-slaves-to-write is 0, then the primary node will always accept writes requests, even if no replicas are available.
min-slaves-to-write	Default: 0 Type: integer Modifiable: Yes Changes Take Effect: Immediately	The minimum number of read replicas which must be available in order for the primary node to accept writes from clients. If the number of available replicas falls below this number, then the primary node will no longer accept write requests. If either this parameter or min-slaves-max-lag is 0, then the primary node will always accept writes requests, even if no replicas are available.
notify-keyspace-events	Default: (an empty string) Type: string Modifiable: Yes Changes Take Effect: Immediately	The types of keyspace events that Redis can notify clients of. Each event type is represented by a single letter: • **K** — Keyspace events, published with a prefix of *__keyspace@<db>__* • **E** — Key-event events, published with a prefix of *__keyevent@<db>__* • **g** — Generic, non-specific commands such as *DEL*, *EXPIRE*, *RENAME*, etc. • **$** — String commands • **l** — List commands • **s** — Set commands • **h** — Hash commands

Name	Details	Description
		• **z** — Sorted set commands • **x** — Expired events (events generated every time a key expires) • **e** — Evicted events (events generated when a key is evicted for maxmemory) • **A** — An alias for *g$lshzxe* You can have any combination of these event types. For example, *AKE* means that Redis can publish notifications of all event types. Do not use any characters other than those listed above; attempts to do so will result in error messages. By default, this parameter is set to an empty string, meaning that keyspace event notification is disabled.
repl-backlog-size	Default: 1048576 Type: integer Modifiable: Yes Changes Take Effect: Immediately	The size, in bytes, of the primary node backlog buffer. The backlog is used for recording updates to data at the primary node. When a read replica connects to the primary, it attempts to perform a partial sync (`psync`), where it applies data from the backlog to catch up with the primary node. If the `psync` fails, then a full sync is required. The minimum value for this parameter is 16384. **Note** Beginning with Redis 2.8.22, this parameter applies to the primary cluster as well as the read replicas.

Name	Details	Description
repl-backlog-ttl	Default: 3600 Type: integer Modifiable: Yes Changes Take Effect: Immediately	The number of seconds that the primary node will retain the backlog buffer. Starting from the time the last replica node disconnected, the data in the backlog will remain intact until *repl-backlog-ttl* expires. If the replica has not connected to the primary within this time, then the primary will release the backlog buffer. When the replica eventually reconnects, it will have to perform a full sync with the primary. If this parameter is set to 0, then the backlog buffer will never be released.
repl-timeout	Default: 60 Type: integer Modifiable: Yes Changes Take Effect: Immediately	Represents the timeout period, in seconds, for: • Bulk data transfer during synchronization, from the read replica's perspective • Primary node timeout from the replica's perspective • Replica timeout from the primary node's perspective

Redis 2.6.13 Parameters

Redis 2.6.13 was the first version of Redis supported by ElastiCache. The following table shows the Redis 2.6.13 parameters that ElastiCache supports.

Name	Details	Description
activerehashing	Default: yes Type: string (yes/no) Modifiable: At Creation	Determines whether to enable Redis' active rehashing feature. The main hash table is rehashed ten times per second; each rehash operation consumes 1 millisecond of CPU time. This value is set when you create the parameter group. When assigning a new parameter group to a cluster, this value must be the same in both the old and new parameter groups.
appendonly	Default: no Type: string Modifiable: Yes Changes Take Effect: Immediately	Enables or disables Redis' append only file feature (AOF). AOF captures any Redis commands that change data in the cache, and is used to recover from certain node failures. The default value is *no*, meaning AOF is turned off. Set this parameter to *yes* to enable AOF.

<table>
<tr><th>Name</th><th>Details</th><th>Description</th></tr>
<tr><td></td><td></td><td>For more information, see Mitigating Failures (p. 62).

Note
Append Only Files (AOF) is not supported for cache.t1.micro and cache.t2.* nodes. For nodes of this type, the appendonly parameter value is ignored.

Note
For Multi-AZ replication groups, AOF is not allowed.</td></tr>
<tr><td>appendfsync</td><td>Default: everysec

Type: string

Modifiable: Yes

Changes Take Effect: Immediately</td><td>Controls how often the AOF output buffer is written to disk:

• no — the buffer is flushed to disk on an as-needed basis.
• everysec — the buffer is flushed once per second. This is the default.
• always — the buffer is flushed every time that data in the cluster is modified.</td></tr>
<tr><td>client-output-buffer-limit-normal-hard-limit</td><td>Default: 0

Type: integer

Modifiable: Yes

Changes Take Effect: Immediately</td><td>If a client's output buffer reaches the specified number of bytes, the client will be disconnected. The default is zero (no hard limit).</td></tr>
<tr><td>client-output-buffer-limit-normal-soft-limit</td><td>Default: 0

Type: integer

Modifiable: Yes

Changes Take Effect: Immediately</td><td>If a client's output buffer reaches the specified number of bytes, the client will be disconnected, but only if this condition persists for client-output-buffer-limit-normal-soft-seconds. The default is zero (no soft limit).</td></tr>
<tr><td>client-output-buffer-limit-normal-soft-seconds</td><td>Default: 0

Type: integer

Modifiable: Yes

Changes Take Effect: Immediately</td><td>If a client's output buffer remains at client-output-buffer-limit-normal-soft-limit bytes for longer than this number of seconds, the client will be disconnected. The default is zero (no time limit).</td></tr>
<tr><td>client-output-buffer-limit-pubsub-hard-limit</td><td>Default: 33554432

Type: integer

Modifiable: Yes

Changes Take Effect: Immediately</td><td>For Redis publish/subscribe clients: If a client's output buffer reaches the specified number of bytes, the client will be disconnected.</td></tr>
</table>

Name	Details	Description
client-output-buffer-limit-pubsub-soft-limit	Default: 8388608 Type: integer Modifiable: Yes Changes Take Effect: Immediately	For Redis publish/subscribe clients: If a client's output buffer reaches the specified number of bytes, the client will be disconnected, but only if this condition persists for *client-output-buffer-limit-pubsub-soft-seconds*.
client-output-buffer-limit-pubsub-soft-seconds	Default: 60 Type: integer Modifiable: Yes Changes Take Effect: Immediately	For Redis publish/subscribe clients: If a client's output buffer remains at *client-output-buffer-limit-pubsub-soft-limit* bytes for longer than this number of seconds, the client will be disconnected.
client-output-buffer-limit-slave-hard-limit	Default: For values see Redis Node-Type Specific Parameters (p. 305) Type: integer Modifiable: No	For Redis read replicas: If a client's output buffer reaches the specified number of bytes, the client will be disconnected.
client-output-buffer-limit-slave-soft-limit	Default: For values see Redis Node-Type Specific Parameters (p. 305) Type: integer Modifiable: No	For Redis read replicas: If a client's output buffer reaches the specified number of bytes, the client will be disconnected, but only if this condition persists for *client-output-buffer-limit-slave-soft-seconds*.
client-output-buffer-limit-slave-soft-seconds	Default: 60 Type: integer Modifiable: No	For Redis read replicas: If a client's output buffer remains at *client-output-buffer-limit-slave-soft-limit* bytes for longer than this number of seconds, the client will be disconnected.
databases	Default: 16 Type: integer Modifiable: At Creation	The number of logical partitions the databases is split into. We recommend keeping this value low. This value is set when you create the parameter group. When assigning a new parameter group to a cluster, this value must be the same in both the old and new parameter groups.
hash-max-ziplist-entries	Default: 512 Type: integer Modifiable: Yes Changes Take Effect: Immediately	Determines the amount of memory used for hashes. Hashes with fewer than the specified number of entries are stored using a special encoding that saves space.

Name	Details	Description
hash-max-ziplist-value	Default: 64 Type: integer Modifiable: Yes Changes Take Effect: Immediately	Determines the amount of memory used for hashes. Hashes with entries that are smaller than the specified number of bytes are stored using a special encoding that saves space.
list-max-ziplist-entries	Default: 512 Type: integer Modifiable: Yes Changes Take Effect: Immediately	Determines the amount of memory used for lists. Lists with fewer than the specified number of entries are stored using a special encoding that saves space.
list-max-ziplist-value	Default: 64 Type: integer Modifiable: Yes Changes Take Effect: Immediately	Determines the amount of memory used for lists. Lists with entries that are smaller than the specified number of bytes are stored using a special encoding that saves space.
lua-time-limit	Default: 5000 Type: integer Modifiable: No	The maximum execution time for a Lua script, in milliseconds, before ElastiCache takes action to stop the script. If *lua-time-limit* is exceeded, all Redis commands will return an error of the form ____-*BUSY*. Since this state can cause interference with many essential Redis operations, ElastiCache will first issue a *SCRIPT KILL* command. If this is unsuccessful, ElastiCache will forcibly restart Redis.
maxclients	Default: 65000 Type: integer Modifiable: No	The maximum number of clients that can be connected at one time.
maxmemory-policy	Default: volatile-lru Type: string Modifiable: Yes Changes Take Effect: Immediately	The eviction policy for keys when maximum memory usage is reached. Valid values are: `volatile-lru | allkeys-lru | volatile-random | allkeys-random | volatile-ttl | noeviction` For more information, see What eviction policies do you support? at RedisLabs.
maxmemory-samples	Default: 3 Type: integer Modifiable: Yes Changes Take Effect: Immediately	For least-recently-used (LRU) and time-to-live (TTL) calculations, this parameter represents the sample size of keys to check. By default, Redis chooses 3 keys and uses the one that was used least recently.

Name	Details	Description
reserved-memory	Default: 0 Type: integer Modifiable: Yes Changes Take Effect: Immediately	The total memory reserved for non-cluster usage. By default, the Redis node will grow until it consumes the node's *maxmemory* (see Redis Node-Type Specific Parameters (p. 305)). If this occurs, then node performance will likely suffer due to excessive memory paging. By reserving memory you can set aside some of the available memory for non-Redis purposes to help reduce the amount of paging. For example, suppose you have a cache.m1.small node, with a *maxmemory* of 900MB bytes. If you set aside 180 MB, then Redis will never consume this memory; instead, this 180 MB is reserved for the operating system and other background processes on the node. Consider increasing the value of the *reserved-memory* parameter if you are using read replicas, append-only files (AOF), or other Redis features that consume more memory. This parameter is specific to ElastiCache, and is not part of the standard Redis distribution. For more information, see Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 60).
set-max-intset-entries	Default: 512 Type: integer Modifiable: Yes Changes Take Effect: Immediately	Determines the amount of memory used for certain kinds of sets (strings that are integers in radix 10 in the range of 64 bit signed integers). Such sets with fewer than the specified number of entries are stored using a special encoding that saves space.
slave-allow-chaining	Default: no Type: string Modifiable: No	Determines whether a read replica in Redis can have read replicas of its own.
slowlog-log-slower-than	Default: 10000 Type: integer Modifiable: Yes Changes Take Effect: Immediately	The maximum execution time, in microseconds, for commands to be logged by the Redis Slow Log feature.
slowlog-max-len	Default: 128 Type: integer Modifiable: Yes Changes Take Effect: Immediately	The maximum length of the Redis Slow Log.

Name	Details	Description
tcp-keepalive	Default: 0 Type: integer Modifiable: Yes Changes Take Effect: Immediately	If this is set to a nonzero value (N), node clients are polled every N seconds to ensure that they are still connected. With the default setting of 0, no such polling occurs.
timeout	Default: 0 Type: integer Modifiable: Yes Changes Take Effect: Immediately	If this is set to a nonzero value (N), the node closes a connection if the client is idle for N seconds. With the default setting of 0, the node does not disconnect idle clients.
zset-max-ziplist-entries	Default: 128 Type: integer Modifiable: Yes Changes Take Effect: Immediately	Determines the amount of memory used for sorted sets. Sorted sets with fewer than the specified number of elements are stored using a special encoding that saves space.
zset-max-ziplist-value	Default: 64 Type: integer Modifiable: Yes Changes Take Effect: Immediately	Determines the amount of memory used for sorted sets. Sorted sets with entries that are smaller than the specified number of bytes are stored using a special encoding that saves space.

Note
If you do not specify a parameter group for your Redis 2.6.13 cluster, then a default parameter group (`default.redis2.6`) will be used. You cannot change the values of any parameters in the default parameter group; however, you can always create a custom parameter group and assign it to your cluster at any time.

Redis 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 *maxmemory*, *client-output-buffer-limit-slave-hard-limit*, and *client-output-buffer-limit-slave-soft-limit* parameters for each node type. The value of *maxmemory* is the maximum number of bytes available to you for use, data and other uses, on the node.

Note
The *maxmemory* parameter cannot be modified.

Node Type	maxmemory	client-output-buffer-limit-slave-hard-limit	client-output-buffer-limit-slave-soft-limit
cache.t1.micro	142606336	14260633	14260633
cache.t2.micro	581959680	58195968	58195968

Node Type	maxmemory	client-output-buffer-limit-slave-hard-limit	client-output-buffer-limit-slave-soft-limit
cache.t2.small	1665138688	166513868	166513868
cache.t2.medium	3461349376	346134937	346134937
cache.m1.small	943718400	943718400	943718400
cache.m1.medium	3093299200	309329920	309329920
cache.m1.large	7025459200	702545920	702545920
cache.m1.xlarge	14889779200	1488977920	1488977920
cache.m2.xlarge	17091788800	1709178880	1709178880
cache.m2.2xlarge	35022438400	3502243840	3502243840
cache.m2.4xlarge	70883737600	7088373760	7088373760
cache.m3.medium	2988441600	309329920	309329920
cache.m3.large	6501171200	650117120	650117120
cache.m3.xlarge	14260633600	1426063360	1426063360
cache.m3.2xlarge	29989273600	2998927360	2998927360
cache.m4.large	6892593152	689259315	689259315
cache.m4.xlarge	15328501760	1532850176	1532850176
cache.m4.2xlarge	31889126359	3188912636	3188912636
cache.m4.4xlarge	65257290629	6525729063	6525729063
cache.m4.10xlarge	166047614239	16604761424	16604761424
cache.c1.xlarge	6501171200	650117120	650117120
cache.r3.large	14470348800	1468006400	1468006400
cache.r3.xlarge	30513561600	3040870400	3040870400
cache.r3.2xlarge	62495129600	6081740800	6081740800
cache.r3.4xlarge	126458265600	12268339200	12268339200
cache.r3.8xlarge	254384537600	24536678400	24536678400

Note

t2 instances do not support Redis backup/restore.

t1 and t2 instances do not support Redis AOF or Multi-AZ.

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 select 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 4: Authorize Access (p. 24).

Topics

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.

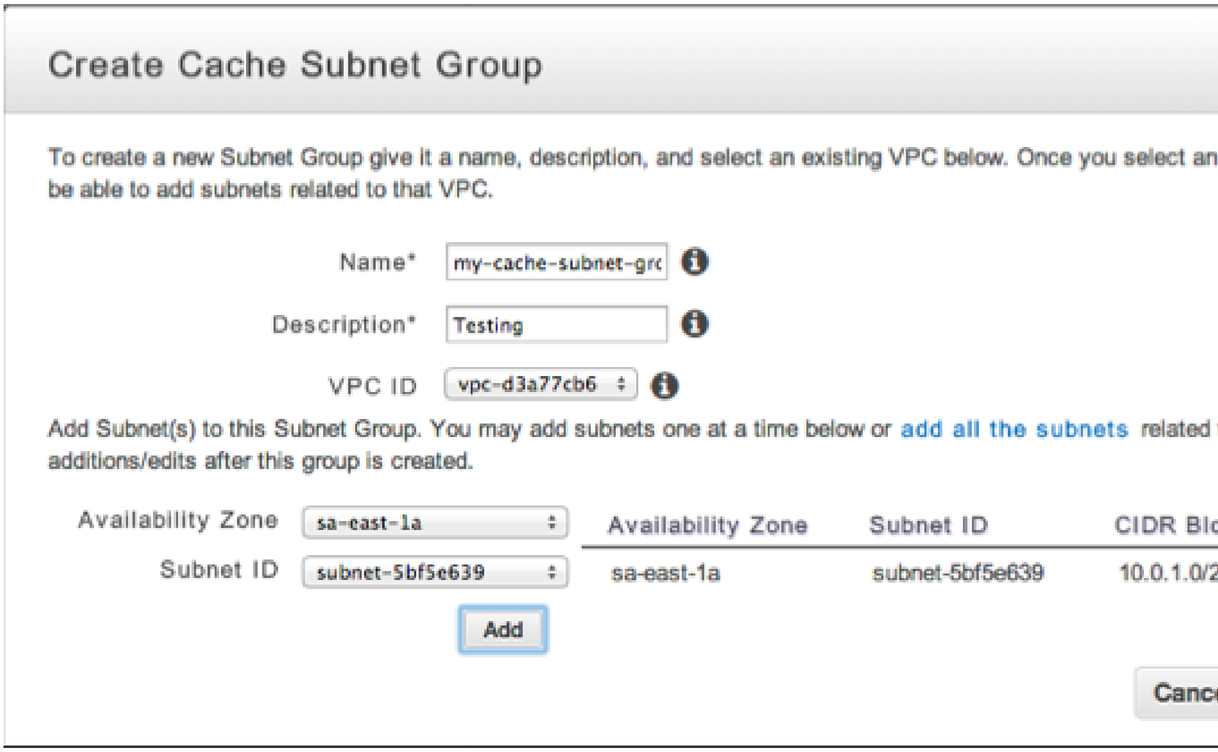
Creating a Subnet Group (Console)

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**.



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, OS X, 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:

```
SUBNETGROUP  mysubnetgroup  Testing  vpc-5a2e4c35
 SUBNET  subnet-53df9c3a  us-west-2b
```

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 or Replication Group

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

- **Memcached cluster** – To launch a Memcached cluster, see Creating a Cluster (Console): Memcached (p. 129). In step 5.a (**Advanced Memcached Settings**), select a VPC subnet group.
- **Standalone Redis cluster** – To launch a single-node Redis cluster, see Creating a Redis (cluster mode disabled) Cluster (Console) (p. 130). In step 5.a (**Advanced Redis Settings**), select a VPC subnet group.
- **Redis (cluster mode disabled) replication group** – To launch a Redis (cluster mode disabled) replication group in a VPC, see Creating a Redis Cluster with Replicas from Scratch (Console) (p. 210). In step 5.b (**Advanced Redis Settings**), select a VPC subnet group.
- **Redis (cluster mode enabled) replication group** – Creating a Redis (cluster mode enabled) Replication Group from Scratch (Console) (p. 215). In step 5.a (**Advanced Redis Settings**), select 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 left navigation pane, choose **Subnet Groups**.
3. In the list of subnet groups, select 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, OS X, 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:

```
SUBNETGROUP  mysubnetgroup  Testing  vpc-5a2e4c35
 SUBNET  subnet-42df9c3a  us-west-2b
 SUBNET  subnet-48fc21a9  us-west-2b
```

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 left navigation pane, choose **Subnet Groups**.
3. In the list of subnet groups, select 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)

At a command prompt, use the command `delete-cache-subnet-group` to delete a subnet group.

For Linux, OS X, 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 in the following code example 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
```

Amazon Virtual Private Cloud (Amazon VPC) with ElastiCache

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

ElastiCache and Amazon VPC

Note
ElastiCache is fully integrated with 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 a VPC.
- If you're new to AWS, your clusters will be deployed into a 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 VPC.

For more information, see Detecting Your Supported Platforms and Whether You Have a Default VPC.

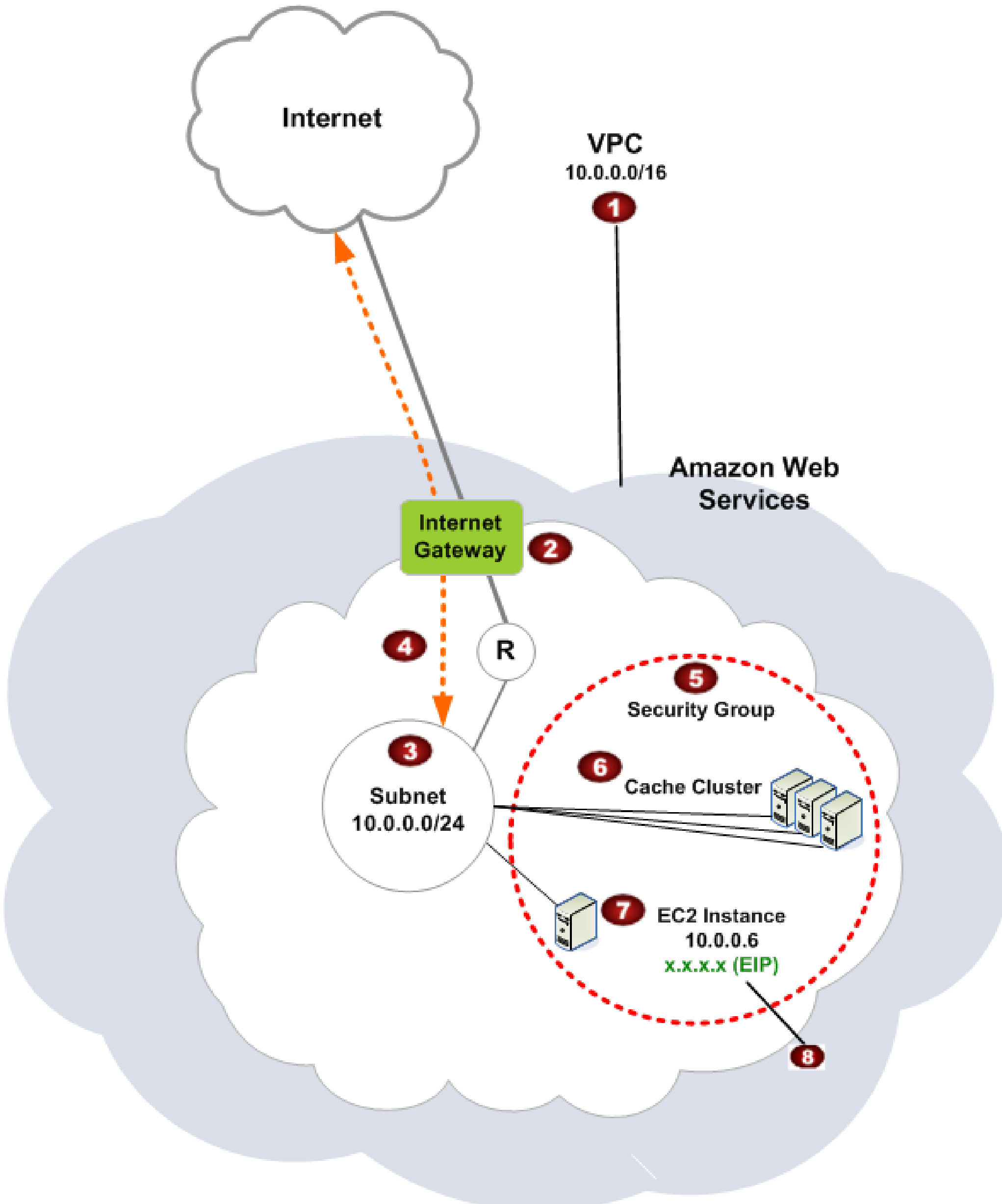
With Amazon Virtual Private Cloud, you can create a virtual network in the AWS cloud that closely resembles a traditional data center. You can configure your Amazon VPC, including selecting its IP address range, creating subnets, and configuring route tables, network gateways, and security settings.

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.



	The Amazon VPC is an isolated portion of the AWS cloud that is assigned its own block of IP addresses.
	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.
	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.
	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.

	An Amazon VPC security group controls inbound and outbound traffic for your ElastiCache clusters and Amazon EC2 instances.
	You can launch an ElastiCache cluster in the subnet. The cache nodes have private IP addresses from the subnet's range of addresses.
	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.
	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 a 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, go to 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.

> **Note**
> ElastiCache is fully integrated with 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 a VPC.
> - If you're new to AWS, your clusters will be deployed into a 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 VPC.
>
> For more information, see Detecting Your Supported Platforms and Whether You Have a Default VPC.

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.

Clusters in an Amazon VPC can be accessed by Amazon EC2 instances in the same Amazon VPC. If these Amazon EC2 instances are deployed in a public subnet with associated Elastic IPs, you can access the Amazon EC2 instances via the Internet.

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.

Creating a Virtual Private Cloud (VPC)

In this example, you create an Amazon VPC with a private subnet for each Availability Zone.

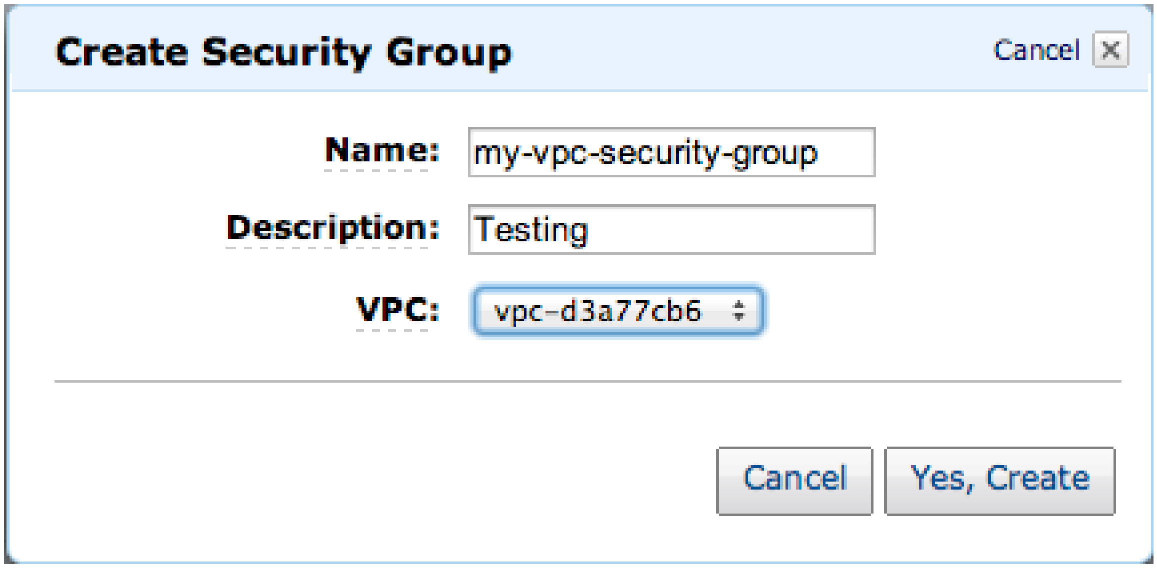
Creating an Amazon VPC (Console)

To create an ElastiCache cache 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:

      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 left 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.



   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, go to Creating a Subnet Group (p. 308).

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. 324).

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 Cluster (Console): Memcached (p. 129). In step 6.c select a VPC subnet group.
- To launch a Redis cache cluster, see Creating a Redis (cluster mode disabled) Cluster (Console) (p. 130). In step 6.d 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 or Replication Group Running in an Amazon VPC (p. 326).

Creating a Replication Group in an Amazon VPC

In this example, you create a Redis replication group in your Amazon VPC.

Creating a Replication Group in an Amazon VPC (Console)

To launch a Redis (cluster mode disabled) replication group in a VPC, see Creating a Redis Cluster with Replicas from Scratch (Console) (p. 210) In step 5.b, select a VPC subnet group.

To launch a Redis (cluster mode enabled) replication group, see Creating a Redis (cluster mode enabled) Replication Group from Scratch (Console) (p. 215) In step 6.d, select a VPC subnet group.

You have now launched a Redis replication group inside an Amazon VPC. For an example of one way to connect to your new replication group running in the Amazon VPC, continue to Connecting to a Cluster or Replication Group Running in an Amazon VPC (p. 326).

Connecting to a Cluster or Replication Group 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.

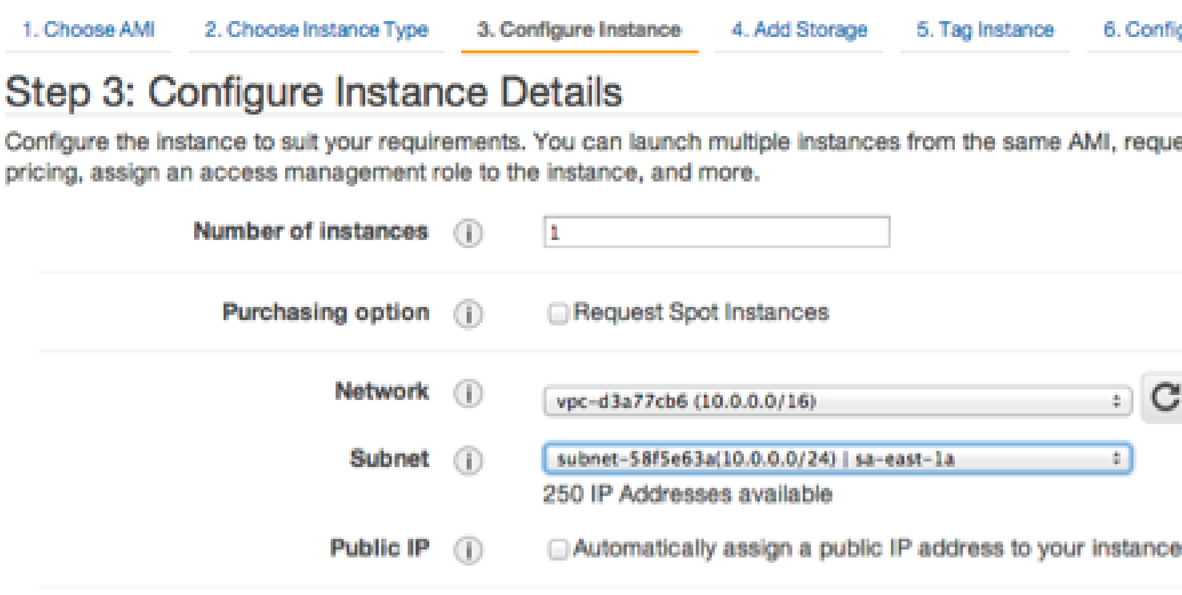
Topics

1. Create an Amazon EC2 Instance

The following procedure creates an Amazon EC2 instance in your VPC.

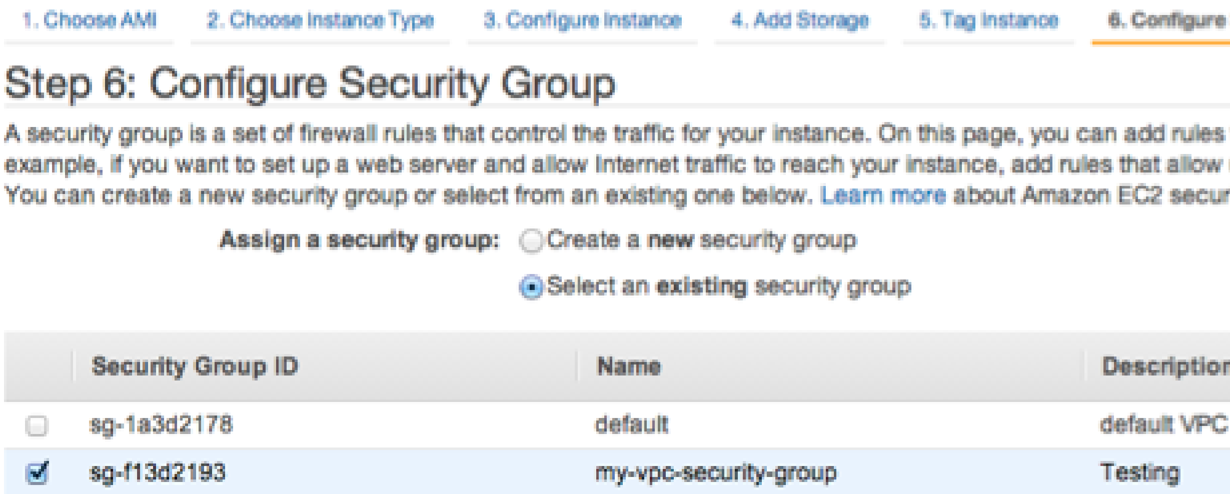
1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the console, choose **Launch Instance** and follow these steps:
3. On the **Choose an Amazon Machine Image (AMI)** page, choose the 64-bit Amazon Linux AMI, and then choose **Select.**
4. On the **Choose an Instance Type** page, choose **3. Configure Instance**.
5. On the **Configure Instance Details** page, make the following selections:

   a. In the **Network** list, choose your Amazon VPC.
   b. In the **Subnet** list, choose your public subnet.



When the settings are as you want them, choose **4. Add Storage**.

6. On the **Add Storage** page, choose **5. Tag Instance**.
7. On the **Tag Instance** page, type a name for your Amazon EC2 instance, and then choose **6. Configure Security Group**.
8. On the **Configure Security Group** page, choose **Select an existing security group**.

   Select the name of your Amazon VPC security group, and then choose **Review and Launch**.
9. On the **Review Instance and Launch** page, choose **Launch**.

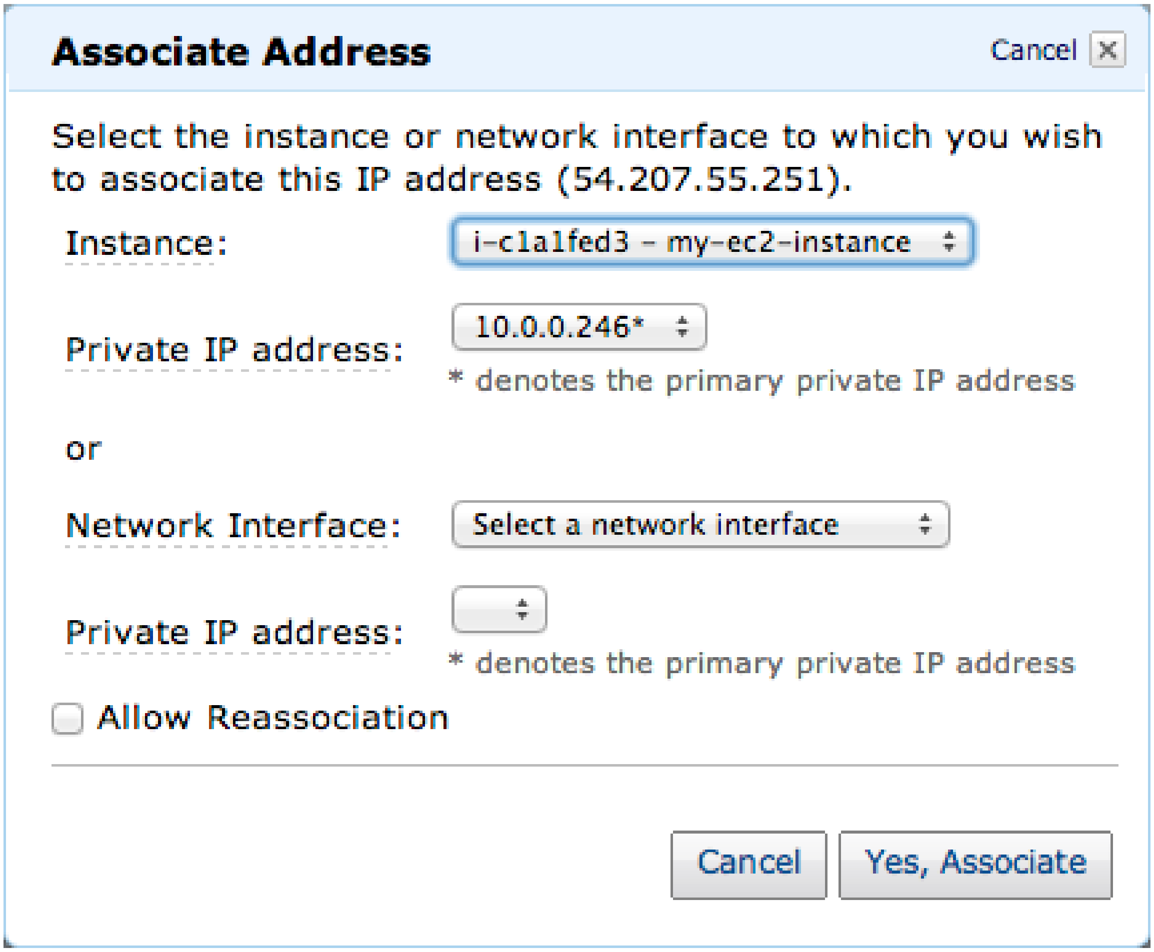
   In the **Select an existing key pair or create a new key pair** window, specify a key pair that you want to use with this instance.

   Note
   For information about managing key pairs, go to the Amazon EC2 Getting Started Guide.
10. When you are ready to launch your Amazon EC2 instance, choose **Launch Instances**.

2. Assign IP Address to Your Amazon EC2 Instance

You can now assign an Elastic IP address to the Amazon EC2 instance that you just created. You need to use this IP address to connect to the Amazon EC2 instance.

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation list, choose **Elastic IPs**.
3. Choose **Allocate New Address**.
4. In the **Allocate New Address** dialog box, in the **EIP used in** box, choose **VPC**, and then choose **Yes, Allocate**.
5. Select the Elastic IP address that you just allocated from the list and choose **Associate Address**.
6. In the **Associate Address** dialog box, in the **Instance** box, choose the ID of the Amazon EC2 instance that you launched, and then choose **Yes, Associate**.



You can now use SSH to connect to the Amazon EC2 instance using the Elastic IP address that you created.

Tip
For instructions about using SSH to connect to a Linux/UNIX instance, see Connect to Your Linux/UNIX Instance in the Amazon EC2 Getting Started Guide.

3. Connect to Your Amazon EC2 Instance

The following steps connects you to your Amazon EC2 instance.

1. Open a command window. At the command prompt, issue the following command, replacing *mykeypair.pem* with the name of your key pair file and *54.207.55.251* with your Elastic IP address.

```
ssh -i mykeypair.pem ec2-user@54.207.55.251
```

   Important
   Do not log out of your Amazon EC2 instance yet.

2. You are now ready to interact with your ElastiCache cluster. Before you can do that, if you haven't already done so, you need to install the *telnet* utility.
   To install telnet and interact with your cache cluster (AWS CLI)

   For Linux, OS X, or Unix:

Open a command window. At the command prompt, issue the following command. At the confirmation prompt, type *y*.

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

 ...(output omitted)...

 Total download size: 63 k
 Installed size: 109 k
 Is this ok [y/N]: y
 Downloading Packages:
 telnet-0.17-47.7.amzn1.x86_64.rpm                       |  63 kB
 00:00

 ...(output omitted)...

 Complete!
```

For Windows

Open a command window. At the command prompt, issue the following command. If the **User Account Control** dialog box appears, confirm that the action it displays is what you want, and then click **Continue**.

```
pkgmgr /iu:"TelnetClient"
```

3. Go to the ElastiCache console at https://console.aws.amazon.com/elasticache/ and obtain the endpoint for one of the nodes in your cache cluster.
4. Use *telnet* to connect to your cache node endpoint over port 11211. Replace the hostname shown below with the hostname of your cache node.

```
telnet my-cache-cluster.7wufxa.0001.use1.cache.amazonaws.com 11211
```

You are now connected to the cache engine and can issue commands. In this example, you add a data item to the cache and then get it immediately afterward. Finally, you'll disconnect from the cache node.

To store a key and a value, type the following two lines:

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

The cache engine responds with the following:

```
STORED
```

To retrieve the value for `mykey`, type the following:

```
get mykey
```

The cache engine responds with the following:

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

To disconnect from the cache engine, type the following:

```
quit
```

Important
To avoid incurring additional charges on your AWS account, be sure to delete any AWS resources you no longer want after trying these examples.

Security for Amazon ElastiCache

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

- **ElastiCache and Security Groups (p. 331)** explains the type of security group you need for your installation.
- **Authentication and Access Control for Amazon ElastiCache (p. 333)** for granting and limiting actions of users, groups, and roles.

ElastiCache and Security Groups

Because data security is important, ElastiCache provides means for you to control who has access to your data. How you control access to your data is dependent upon whether or not you launched your clusters in an Amazon Virtual Private Cloud (Amazon VPC) or Amazon EC2-Classic.

Topics

Amazon Virtual Private Cloud: Amazon VPC Security Groups

When running your clusters in an Amazon Virtual Private Cloud, you configure your Amazon VPC by selecting its IP address range, creating subnets, and configuring 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, which should not be confused with Amazon ElastiCache security groups. For more information, see Amazon Virtual Private Cloud (Amazon VPC) with ElastiCache (p. 316).

Amazon EC2-Classic: ElastiCache Security Groups

Amazon ElastiCache allows you to control access to your clusters using ElastiCache cache security groups. An ElastiCache cache 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. For more information, see Security Groups [EC2-Classic] (p. 263) .

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. 333)
- Access Control (p. 334)

Authentication

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

- **AWS account root user** – When you sign up for AWS, you provide an email address and password that is associated with your AWS account. These are your *root credentials* and they provide complete access to all of your AWS resources.

  Important
  For security reasons, we recommend that you use the root credentials only to create an *administrator user*, which is an *IAM user* with full permissions to your AWS account. Then, you can use this administrator user to create other IAM users and roles with limited permissions. For more information, see IAM Best Practices and Creating an Admin User and Group in the *IAM User Guide*.
- **IAM user** – An IAM user is simply an identity within your AWS account that has specific custom permissions (for example, permissions to create a cache 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 the 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 another IAM identity 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 preexisting 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*.

- **Cross-account access** – You can use an IAM role in your account to grant another AWS account permissions to access your account's resources. For an example, see Tutorial: Delegate Access Across AWS Accounts Using IAM 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 stored in the 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** – Instead of storing access keys within the EC2 instance for use by applications running on the instance and making AWS API requests, you can use an IAM role to manage temporary credentials for these applications. To assign an AWS role to an EC2 instance and make it available to all of its applications, you can create an instance profile that is attached to the instance. An instance profile contains the role and enables programs running on the EC2 instance to get temporary credentials. For more information, see Using Roles for Applications on Amazon EC2 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. 335)
- Using Identity-Based Policies (IAM Policies) for Amazon ElastiCache (p. 339)

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

In Amazon ElastiCache, the primary resource is a *cache cluster*. Amazon ElastiCache also supports an additional resource type, a *snapshot*. However, you can create snapshots only in the context of an existing Redis cache cluster. A snapshot is referred to as *subresource*.

These resources and subresources have unique Amazon Resource Names (ARNs) associated with them as shown in the following table.

Resource Type	ARN Format
Cache Cluster	`arn:aws:elasticache:`*`region`*`:`*`account-id`*`:cluster:`*`resource-name`*
Snapshot	`arn:aws:elasticache:`*`region`*`:`*`account-id`*`:snapshot:`*`resource-name`*

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 polices) 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)

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. 339). 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. 335)), 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: `CreateSnapshot`, `DeleteSnapshot`, and `DescribeSnapshots`. 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. 335).
- **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. 344).

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. 335).

The sections in this topic cover the following:

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": "*"
       },
       {
       "Effect": "Allow",
       "Action": [
          "iam:PassRole"],
       "Resource": "arn:aws:iam::account-id:role/*"
       }
   ]
}
```

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. 344).

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. 344).

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

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. 340).

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. Also, if your users need to create a Redis cache cluster with replication enabled, you need to grant permissions for them to perform the `elasticache:CreateReplicationGroup` action.

Examples

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": "ECAllowSpecific",
      "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": "ECAllowSpecific",
      "Effect":"Allow",
      "Action":[
          "elasticache:*" ],
      "Resource":"*"
      }
   ]
}
```

ElastiCache API Permissions: Actions, Resources, and Conditions Reference

When you are setting up Access Control (p. 334) 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:DescribeSnapshots`). 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: *

CopySnapshot
: **Action(s):** `elasticache:CopySnapshot`

  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: *

CreateReplicationGroup
: **Action(s):** `elasticache:CreateReplicationGroup`

  `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.

CreateSnapshot
: **Action(s):** `elasticache:CreateSnapshot`

  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: *

DeleteReplicationGroup
: **Action(s):** `elasticache:DeleteReplicationGroup`

  Resource: *

DeleteSnapshot
: **Action(s):** `elasticache:DeleteSnapshot`

  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: *

DescribeReplicationGroups

Action(s): `elasticache:DescribeReplicationGroups`

Resource: *

DescribeReservedCacheNodes

Action(s): `elasticache:DescribeReservedCacheNodes`

Resource: *

DescribeReservedCacheNodesOfferings

Action(s): `elasticache:DescribeReservedCacheNodesOfferings`

Resource: *

DescribeSnapshots

Action(s): `elasticache:DescribeSnapshots`

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: *

ModifyReplicationGroup

Action(s): `elasticache:ModifyReplicationGroup`

Resource: *

PurchaseReservedCacheNodesOffering

Action(s): `elasticache:PurchaseReservedCacheNodesOffering`

Resource: *

RebootCacheCluster
Action(s): `elasticache:RebootCacheCluster`

Resource: *

RemoveTagsFromResource
Action(s): `elasticache:RemoveTagsFromResource`

Resource: *

ResetCacheParameterGroup
Action(s): `elasticache:ResetCacheParameterGroup`

Resource: *

RevokeCacheSecurityGroupIngress
Action(s): `elasticache:RevokeCacheSecurityGroupIngress`

Resource: *

Accessing ElastiCache Resources from Outside AWS

Amazon ElastiCache is an AWS service that provides cloud-based in-memory key-value store. On the back end it uses either the Memcached or Redis engine. 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.

Topics

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 the IP address *10.0.1.230*, the default Memcached port *11211*, and 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*.

When you finish creating your NAT instance using the following steps, the following should be true.

- IP forwarding is enabled for the NAT instance. The following command can be used to confirm this.

```
cat /proc/sys/net/ipv4/ip_forward
```

- Masquerading is enabled. The following command can be used to enable masquerading.

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

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	198.51.100.27/32

   - An outbound rule to allow TCP connections to each forwarded cache port (11211-11213).

   NAT Instance Security Group - Outbound Rule

Type	Protocol	Port Range	Destination
Custom TCP Rule	TCP	11211-11213	sg-bd56b7da (Cache Cluster Security Group)

   - An inbound rule for the cluster's security group that allows TCP connections from the NAT instance to the cache port on each instance in the cluster (11211-11213).

   Cache Cluster Security Group - Inbound Rule

Type	Protocol	Port Range	Source
Custom TCP Rule	TCP	11211-11213	sg-ce56b7a9 (NAT instance 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
```

See also

- NAT Instances
- Configuring ElastiCache Clients
- High Availability for Amazon VPC NAT Instances: An Example

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

ElastiCache provides metrics that enable you to monitor your clusters. You can access these metrics through CloudWatch. For more information on CloudWatch, go to 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. 359).

Topics

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 Command Line Interface Reference*.

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

Topics

Host-Level Metrics

The following table lists host-level metrics provided by ElastiCache for individual cache nodes.

See Also

- Metrics for Memcached (p. 354)
- Metrics for Redis (p. 356)

Metric	Description	Unit
CPUUtilization	The percentage of CPU utilization.	Percent
FreeableMemory	The amount of free memory available on the host.	Bytes
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 following table lists the metrics provided by ElastiCache 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. 353)

Metric	Description	Unit
BytesReadIntoMemcached	The number of bytes that have been read from the network by the cache node.	Bytes
BytesUsedForCacheItems	The number of bytes used to store cache items.	Bytes
BytesWrittenOutFromMemcached	The number of bytes that have been written to the network by the cache node.	Bytes
CasBadval	The number of CAS (check and set) requests the cache has received where the Cas value did not match the Cas value stored.	Count
CasHits	The number of Cas requests the cache has received where the requested key was found and the Cas value matched.	Count
CasMisses	The number of Cas requests the cache has received where the key requested was not found.	Count
CmdFlush	The number of **flush** commands the cache has received.	Count
CmdGet	The number of **get** commands the cache has received.	Count
CmdSet	The number of **set** commands the cache has received.	Count

Metric	Description	Unit
CurrConnections	A count of the number of connections connected to the cache at an instant in time.	Count
CurrItems	A count of the number of items currently stored in the cache.	Count
DecrHits	The number of decrement requests the cache has received where the requested key was found.	Count
DecrMisses	The number of decrement requests the cache has received where the requested key was not found.	Count
DeleteHits	The number of delete requests the cache has received where the requested key was found.	Count
DeleteMisses	The number of delete requests the cache has received where the requested key was not found.	Count
Evictions	The number of non-expired items the cache evicted to allow space for new writes.	Count
GetHits	The number of get requests the cache has received where the key requested was found.	Count
GetMisses	The number of get requests the cache has received where the key requested was not found.	Count
IncrHits	The number of increment requests the cache has received where the key requested was found.	Count
IncrMisses	The number of increment requests the cache has received where the key requested was not found.	Count
Reclaimed	The number of expired items the cache evicted to allow space for new writes.	Count

For Memcached 1.4.14, the following additional metrics are provided.

Metric	Description	Unit
BytesUsedForHash	The number of bytes currently used by hash tables.	Bytes
CmdConfigGet	The cumulative number of **config get** requests.	Count
CmdConfigSet	The cumulative number of **config set** requests.	Count
CmdTouch	The cumulative number of **touch** requests.	Count
CurrConfig	The current number of configurations stored.	Count
EvictedUnfetched	The number of valid items evicted from the least recently used cache (LRU) which were never touched after being set.	Count

Metric	Description	Unit
ExpiredUnfetched	The number of expired items reclaimed from the LRU which were never touched after being set.	Count
SlabsMoved	The total number of slab pages that have been moved.	Count
TouchHits	The number of keys that have been touched and were given a new expiration time.	Count
TouchMisses	The number of items that have been touched, but were not found.	Count

The following table describes the available calculated cache-level metrics.

Metric	Description	Unit
NewConnections	The number of new connections the cache has received. This is derived from the memcached `total_connections` statistic by recording the change in `total_connections` across a period of time. This will always be at least 1, due to a connection reserved for a ElastiCache.	Count
NewItems	The number of new items the cache has stored. This is derived from the memcached `total_items` statistic by recording the change in `total_items` across a period of time.	Count
UnusedMemory	The amount of memory not used by data. This is derived from the Memcached statistics `limit_maxbytes` and `bytes` by subtracting `bytes` from `limit_maxbytes`. Because Memcached overhead uses memory in addition to that used by data, `UnusedMemory` should not be considered to be the amount of memory available for additional data. You may experience evictions even though you still have some unused memory. For more detailed information, see Memcached item memory usage.	Bytes

Metrics for Redis

The following table lists the metrics provided by ElastiCache. With the exception of `ReplicationLag`, these metrics are derived from the Redis **info** command. Each metric is calculated at the cache node level.

For complete documentation of the Redis **info** command, see Redis commands website.

See Also

- Host-Level Metrics (p. 353)

Metric	Description	Unit
`BytesUsedForCache`	The total number of bytes allocated by Redis.	Bytes
`CacheHits`	The number of successful key lookups.	Count
`CacheMisses`	The number of unsuccessful key lookups.	Count
`CurrConnections`	The number of client connections, excluding connections from read replicas.	Count
`Evictions`	The number of keys that have been evicted due to the `maxmemory` limit.	Count
`HyperLogLogBasedCmds`	The total number of `HyperLogLog` based commands. This is derived from the Redis `commandstats` statistic by summing all of the **pf** type of commands (**pfadd**, **pfcount**, **pfmerge**).	Count
`NewConnections`	The total number of connections that have been accepted by the server during this period.	Count
`Reclaimed`	The total number of key expiration events.	Count
`ReplicationBytes`	For primaries with attached replicas, `ReplicationBytes` reports the number of bytes that the primary is sending to all of its replicas. This metric is representative of the write load on the replication group. For replicas and standalone primaries, `ReplicationBytes` is always 0.	Bytes
`ReplicationLag`	This metric is only applicable for a node running as a read replica. It represents how far behind, in seconds, the replica is in applying changes from the primary node.	Seconds
`SaveInProgress`	This binary metric returns 1 whenever a background save (forked or forkless) is in progress, and 0 otherwise. A background save process is typically used during snapshots and syncs. These operations can cause degraded performance. Using the `SaveInProgress` metric, you can diagnose whether or not degraded performance was caused by a background save process.	Count

These are aggregations of certain kinds of commands, derived from **info commandstats**:

Metric	Description	Unit
`CurrItems`	The number of items in the cluster. This is derived from the Redis `keyspace` statistic, summing all of the keys in the entire keyspace.	Count
`GetTypeCmds`	The total number of **get** types of commands. This is derived from the Redis `commandstats`	Count

Metric	Description	Unit
	statistic by summing all of the **get** types of commands (**get**, **mget**, **hget**, etc.)	
`HashBasedCmds`	The total number of commands that are hash-based. This is derived from the Redis `commandstats` statistic by summing all of the commands that act upon one or more hashes.	Count
`KeyBasedCmds`	The total number of commands that are key-based. This is derived from the Redis `commandstats` statistic by summing all of the commands that act upon one or more keys.	Count
`ListBasedCmds`	The total number of commands that are list-based. This is derived from the Redis `commandstats` statistic by summing all of the commands that act upon one or more lists.	Count
`SetBasedCmds`	The total number of commands that are set-based. This is derived from the Redis `commandstats` statistic by summing all of the commands that act upon one or more sets.	Count
`SetTypeCmds`	The total number of **set** types of commands. This is derived from the Redis `commandstats` statistic by summing all of the **set** types of commands (**set**, **hset**, etc.)	Count
`SortedSetBasedCmds`	The total number of commands that are sorted set-based. This is derived from the Redis `commandstats` statistic by summing all of the commands that act upon one or more sorted sets.	Count
`StringBasedCmds`	The total number of commands that are string-based. This is derived from the Redis `commandstats` statistic by summing all of the commands that act upon one or more strings.	Count

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.

CPUUtilization

This is a host-level metric reported as a percent. For more information, see Host-Level Metrics (p. 353).

- *Memcached:* Since 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.
- *Redis:* Since Redis is single-threaded, the threshold is calculated as (90 / number of processor cores). For example, suppose you are using a *cache.m1.xlarge* node, which has four cores. In this case, the threshold for *CPUUtilization* would be (90 / 4), or 22.5%.

  You will need to determine your own threshold, based on the number of cores in the cache node that you are using. If you exceed this threshold, and your main workload is from read requests, scale your cache cluster out by adding read replicas. If the main workload is from write requests, we recommend scaling up by using a larger cache instance type.

SwapUsage

This is a host-level metric reported in bytes. For more information, see Host-Level Metrics (p. 353).

- *Memcached:* This metric should not exceed 50 MB. If it does, we recommend that you increase the *ConnectionOverhead parameter value*.
- *Redis:* At this time, we have no recommendation for this parameter; you do not need to set a CloudWatch alarm for it.

Evictions

This is a cache engine metric, published for both Memcached and Redis cache clusters. We recommend that you determine your own alarm threshold for this metric based on your application needs.

- *Memcached:* If your exceed your chosen threshold, scale you cache cluster up by using a larger node type, or scale out by adding more nodes.
- *Redis:* If you exceed your chosen threshold, scale your cluster up by using a larger node type.

CurrConnections

This is a cache engine metric, published for both Memcached and Redis cache clusters. We recommend that you determine your own alarm threshold for this metric based on your application needs.

Whether you are running Memcached or Redis, 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 Cache Cluster and Cache Node Metrics (p. 360).

Monitoring CloudWatch Cache Cluster and Cache 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, go to 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 Cache Cluster and Cache 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 Cache Cluster and Cache 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, OS X, or Unix:

  ```
  mon-get-stats CPUUtilization \
      --dimensions="CacheClusterId=mycachecluster,CacheNodeId=0002" \
      --statistics=Average \
      --namespace="AWS/ElastiCache" \
      --start-time 2013-07-05T00:00:00 \
      --end-time 2013-07-06T00:00:00 \
      --period=60
  ```

  For Windows:

  ```
  mon-get-stats CPUUtilization ^
      --dimensions="CacheClusterId=mycachecluster,CacheNodeId=0002" ^
      --statistics=Average ^
      --namespace="AWS/ElastiCache" ^
      --start-time 2013-07-05T00:00:00 ^
      --end-time 2013-07-06T00:00:00 ^
      --period=60
  ```

Monitoring CloudWatch Cache Cluster and Cache Node Metrics Using the CloudWatch API

To gather CPU utilization statistics for a cache cluster

- Call the CloudWatch API `GetMetricStatistics` with the following parameters (note that the start and end times are shown as examples only; you will need to substitute your own appropriate start and end times):
  - `Statistics.member.1=Average`
  - `Namespace=AWS/ElastiCache`
  - `StartTime=2013-07-05T00:00:00`
  - `EndTime=2013-07-06T00:00:00`
  - `Period=60`
  - `MeasureName=CPUUtilization`
  - `Dimensions=CacheClusterId=mycachecluster,CacheNodeId=0002`

Example

```
http://monitoring.amazonaws.com/
    ?SignatureVersion=4
    &Action=GetMetricStatistics
    &Version=2014-12-01
    &StartTime=2013-07-16T00:00:00
    &EndTime=2013-07-16T00:02:00
    &Period=60
    &Statistics.member.1=Average
    &Dimensions.member.1="CacheClusterId=mycachecluster"
    &Dimensions.member.2="CacheNodeId=0002"
    &Namespace=AWS/ElastiCache
    &MeasureName=CPUUtilization
    &Timestamp=2013-07-07T17%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

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. To add an Amazon SNS topic for a replication group, in step 2, instead of selecting a cluster, choose a replication group then follow the same remaining steps.

> **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`. To add or modify an Amazon SNS topic for a replication group, use the AWS CLI command `modify-replication-group`.

The following code example adds an Amazon SNS topic arn to *my-cluster*.

For Linux, OS X, 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 and modify-replication-group.

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`

To add or modify an Amazon SNS topic for a replication group, call the `ModifyReplicationGroup` action.

Example

```
https://elasticache.amazon.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=false
    &CacheClusterId=my-cluster
    &NotificationTopicArn=arn%3Aaws%3Asns%3Aus-
west-2%3A565419523791%3AElastiCacheNotifications
    &Version=2014-12-01
    &SignatureVersion=4
    &SignatureMethod=HmacSHA256
    &Timestamp=20141201T220302Z
    &X-Amz-Algorithm=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 and ModifyReplicationGroup.

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. Choose the engine running on the cluster you want to modify notifications for, either **Memcached** or **Redis**.

   A list of clusters running the selected engine is displayed.
3. In either the **Memcached** or **Redis** list, 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, OS X, 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
```

This command produces output similar to the following:

```
CACHECLUSTER  my-cluster  2013-07-26T01:21:46.607Z  cache.m1.large  memcached
available  3  us-west-2c  1.4.5
   SECGROUP  default  active
   PARAMGRP  default.memcached1.4  in-sync
   NOTIFICATION  arn:aws:sns:us-west-2:565419523791:ElastiCacheNotifications
   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. In the left navigation pane, choose **Events**.

   The *Events* screen appears listing all available events. Each row of the list represents one event and displays the event source, the event type (cache-cluster, cache-parameter-group, cache-security-group, cache-subnet-group, or replication-group), the GMT time of the event, and the 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 the 40 most recent cache-cluster events.

```
aws elasticache describe-events --source-type cache-cluster --max-items 40
```

The following code lists the cache-cluster events for the past 24 hours (1440 minutes).

```
aws elasticache describe-events --source-type cache-cluster --duration 1440
```

For more information, go to `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>my-redis-primary</SourceIdentifier>
            </Event>

 (...output omitted...)

        </Events>
    </DescribeEventsResult>
    <ResponseMetadata>
        <RequestId>e21c81b4-b9cd-11e3-8a16-7978bb24ffdf</RequestId>
    </ResponseMetadata>
</DescribeEventsResponse>
```

For more information, go to `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, go to 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	`"Updated 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.
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:CacheClusterRestoreFailed	`"Restore from %s failed for node %s"`	ElastiCache was unable to populate the cache cluster with Redis snapshot data. This could be due to a nonexistent snapshot file in Amazon S3, or incorrect permissions on that file. If you describe the cache cluster, the status will

Event Name	Message	Description
		be `restore-failed`. You will need to delete the cache cluster and start over. For more information, see Using a Backup to Seed a Cluster (p. 257).
ElastiCache:CacheClusterScalingComplete	`"Succeeded 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	`"Applied change to security group"`	One of the following events has occurred: • The list of cache security groups authorized for the cache cluster has been modified. • One or more new EC2 security groups have been authorized on any of the cache security groups associated with the cache cluster. • One or more EC2 security groups have been revoked from any of the cache security groups associated with the cache cluster.
ElastiCache:CacheNodeReplaceComplete	`"Finished recovery for cache nodes %s"`	ElastiCache has detected that the host running a cache node is degraded or unreachable and has completed replacing the cache node. **Note** The DNS entry for the replaced cache node is not changed. In most instances, you do not need to refresh the server-list for your clients when this event occurs. However, some cache client libraries may stop using the cache node even after ElastiCache has replaced the cache node; in this case, the application should refresh the server-list when this event occurs.

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:CreateReplicationGroupComplete	`"Replication group %s created"`	The replication group was successfully created.
ElastiCache:CreateReplicationGroupFailed	`"Failed to create replication group %s due to unsuccessful creation of its cache cluster(s)."` and `"Deleting all cache clusters belonging to this replication group."`	The replication group was not created.
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	`"The 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	`"The 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, go to Actions You Can Take When a Node is Scheduled for Replacement (p. 92).

Event Name	Message	Description
ElastiCache:NodeReplacementScheduled	`"The 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, go to Actions You Can Take When a Node is Scheduled for Replacement (p. 92).
ElastiCache:RemoveCacheNodeComplete	`"Removed cache nodes %s"`	A cache node has been removed from the cache cluster.
ElastiCache:ReplicationGroupScalingCompleted	`"Succeeded applying modification to cache node type to %s."`	Scale-up operation on replication group completed successfully.
ElastiCache:ReplicationGroupScalingFailed	`"Failed applying modification to cache node type to %s."`	Scale-up operation on replication group failed.
ElastiCache:SnapshotComplete	`"Snapshot succeeded for snapshot with ID '%s' of cache cluster with ID '%s'"`	A cache snapshot has completed successfully.
ElastiCache:SnapshotFailed	`"Snapshot failed for snapshot with ID '%s' of cache cluster with ID '%s'"`	A cache snapshot has failed. See the cluster's cache events for more a detailed cause. If you describe the snapshot, see `DescribeSnapshots`, the status will be `failed`.

For information on viewing ElastiCache events, see Viewing ElastiCache Events (p. 367).

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. A tag key can be used 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, but also see them broken out by region. For more information, go to Use Cost Allocation Tags in the *AWS Billing and Cost Management User Guide*.

You can add ElastiCache cost allocation tags to clusters and snapshots. When you add, list, modify, copy, or remove a tag, the action is applied only to the specified cluster or snapshot, even if it is a cluster in a replication group.

Tags added to snapshots are not used for cost allocation reports. Tags on snapshots are used to retain or restore tags on clusters. When you create a snapshot, the tags on the cluster are copied to the snapshot. When you restore from a snapshot, the tags on the snapshot are copied to the cluster.

Characteristics of an ElastiCache cost allocation tags

- 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, white-space, '_', '.', ':', '/', '=', '+', '-', and '@'.
- 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, white-space, '_', '.', ':', '/', '=', '+', '-', and '@'.
- A resource can have a maximum of 10 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.

> **Important**
> Amazon ElastiCache tagging is not supported by the ElastiCache CLI. Use the AWS CLI for ElastiCache instead.

Topics

Adding Tags to Your ElastiCache Resource

You can add tags to an ElastiCache resource by using the ElastiCache management console, AWS CLI, or ElastiCache API.

Topics

Adding Tags to Your ElastiCache Resource (Console)

You can use the ElastiCache management console to add tags to an ElastiCache resource. A resource can have a maximum of 10 tags.

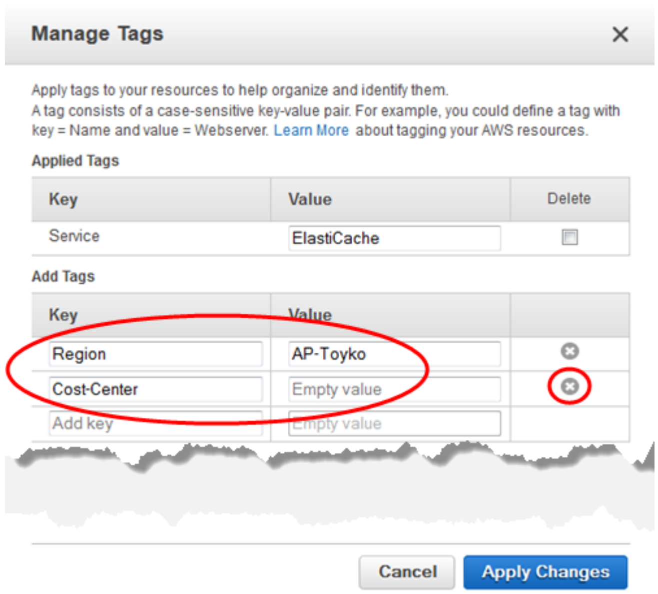
To add a tag to an ElastiCache resource using the ElastiCache management console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Depending what ElastiCache resource you want to add a tag to, select **Memcached**, **Redis**, or **Snapshots**.
3. Select the specific ElastiCache resource you want to add a tag to.

   After you select the resource, you can see the tag names and values currently on this resource at the bottom of the details area.
4. Select **Actions** then, from the menu list, select **Manage Tags**.
5. To add a tag to this resource, do the following in the **Manage Tags** dialog box:
   1. In the **Key** column, type a key name in the box that displays *Add key*.
   2. To add value to this key, type the key's value in the text box in the *Value* column at the key name's right.

   To add multiple tags to this resource, repeat the preceding steps for each tag you want to add.

   If you enter a tag key you don't want to add to this resource, delete it by selecting the "X" to the right of the tag.



Manage Tags dialog

6. When finished, select **Apply Changes** to keep your changes, or **Cancel** to discard your changes.

Adding Tags to Your ElastiCache Resource (AWS CLI)

You can use the AWS CLI to add tags to an existing ElastiCache resource by using the `add-tags-to-resource` AWS CLI for ElastiCache command.

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 resource `myCluster` in the us-west-2 region.

The `resource-name` parameter value is in the format of an ARN:
`arn:aws:elasticache:<region>:<customer id>:<resource type>:<resource name>`

For Linux, OS X, or Unix:

```
aws elasticache add-tags-to-resource \
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:myCluster \
 --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:myCluster ^
 --tags Key=Service,Value=elasticache ^
        Key=Region,Value=us-west-2
```

For more information, go to 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 command create-cache-cluster, or when you create a new replication group by using the command create-replication-group. Note that you cannot add tags during resource creation with the ElastiCache management console. After the cluster or replication group is created, you can then use the console to add tags to the resource.

Adding Tags to Your ElastiCache Resource (ElastiCache API)

You can use the ElastiCache API to add tags to an existing ElastiCache resource by using the AddTagsToResource action.

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 `myCluster` in the us-west-2 region.

The `ResourceName` parameter value is in the format of an ARN:
`arn:aws:elasticache:<region>:<customer id>:<resource type>:<resource name>`

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=AddTagsToResource
   &ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:myCluster
   &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, go to AddTagsToResource in the ElastiCache API documentation.

Listing Your ElastiCache Resource's Tags

You can use the ElastiCache management console, AWS CLI, or ElastiCache API to list all the tags on a specified resource.

Topics

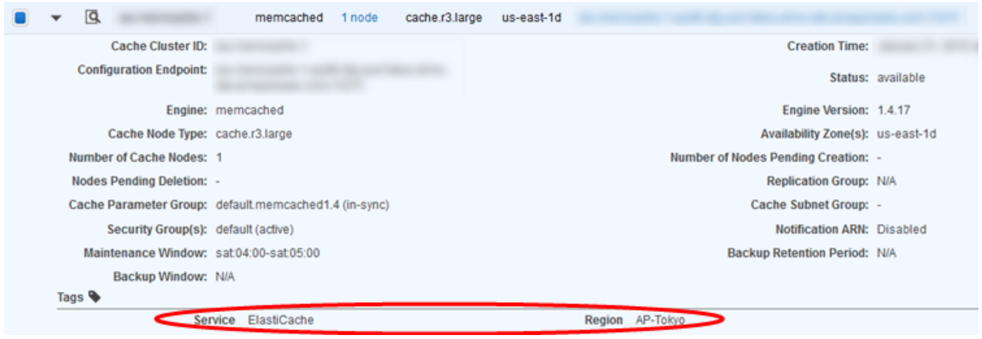
Listing Your ElastiCache Resource's Tags (Console)

You can use the ElastiCache management console to view which tags are on a resource.

To view the tags on a resource using the ElastiCache management console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Depending on the type of resource for which you want to see the current tags, select **Memcached**, **Redis**, or **Snapshots**.
3. Select the specific ElastiCache resource for which you want to see the tag list.

   The tags currently on this resource are listed at the bottom of the details area.



Listing Your ElastiCache Resource's Tags (AWS CLI)

You can use the AWS CLI to list tags on an existing ElastiCache resource by using the list-tags-for-resource command.

The following code uses the ElastiCache AWS CLI to list the tags on the resource `myCluster` in the us-west-2 region.

The *`resource-name`* parameter value is in the format of an ARN:
`arn:aws:elasticache:<region>:<customer id>:<resource type>:<resource name>`

For Linux, OS X, or Unix:

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

For Windows:

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

For more information, go to AWS CLI for ElastiCache list-tags-for-resource.

Listing Your ElastiCache Resource's Tags (ElastiCache API)

You can use the ElastiCache API to list tags on an existing resource by using the ListTagsForResource action.

The following code uses the ElastiCache API to list the tags on the resource `myCluster` in the us-west-2 region.

The `ResourceName` parameter value is in the format of an ARN:
`arn:aws:elasticache:<region>:<customer id>:<resource type>:<resource name>`

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=ListTagsForResource
   &ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:myCluster
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Version=2015-02-02
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Modifying Your ElastiCache Resource's Tags

You can modify the existing tags on an ElastiCache resource by using the ElastiCache management console, AWS CLI, or ElastiCache API.

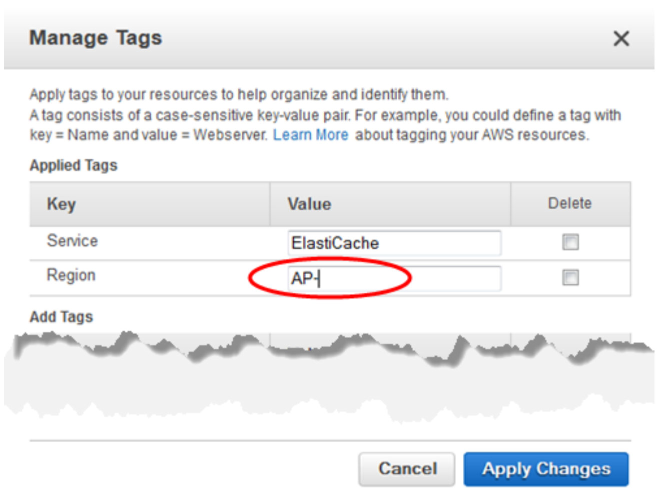
Topics

Modifying Your ElastiCache Resource's Tags (Console)

You can use the ElastiCache management console to modify existing tags on a resource.

To modify a tag on an ElastiCache resource using the ElastiCache management console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Depending what ElastiCache resource you want to modify a tag on, select either **Memcached**, **Redis**, or **Snapshots**.
3. Select the specific ElastiCache resource for which you want to modify a tag.
4. Select **Actions** and then, from the menu list, select **Manage Tags**.
5.
   - To modify a tag's value, either delete the value in the **Value** text box to the right of the tag's key name, or type a value.
   - To delete a tag, select the "X" on the right of the tag's row.



Manage Tags dialog

6. When finished, select **Apply Changes** to keep your changes, or **Cancel** to discard your changes.

Modifying Your ElastiCache Resource's Tags (AWS CLI)

You can use the AWS CLI to modify the tags on an ElastiCache resource.

To modify the value of a tag, use add-tags-to-resource to add the tag with the new value, or use remove-tags-from-resource to remove specified tags from the resource.

Modifying Your ElastiCache Resource's Tags (ElastiCache API)

You can use the ElastiCache API to modify the tags on an ElastiCache resource.

To modify the value of a tag, use AddTagsToResource action to add tags, or use RemoveTagsFromResource to remove tags from the resource.

Removing Tags from Your ElastiCache Resource

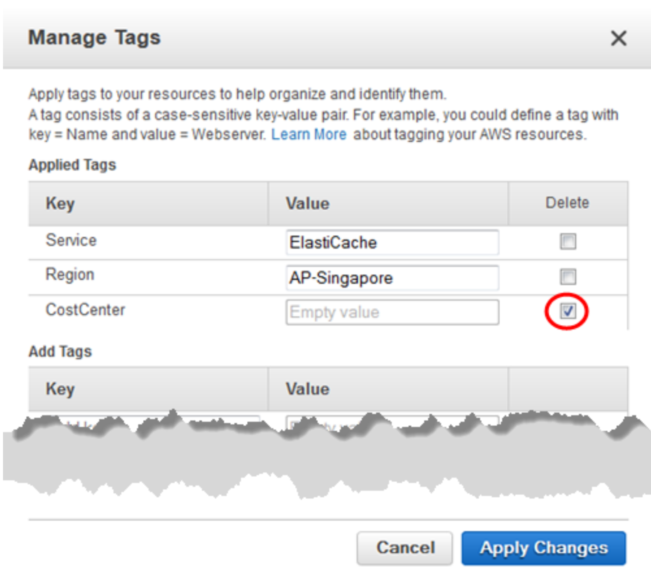
You can remove one or more tags from a resource using the ElastiCache management console, AWS CLI, or ElastiCache API.

Removing Tags from Your ElastiCache Resource (Console)

You can use the ElastiCache management console to remove tags from an existing ElastiCache resource.

To remove a tag from an ElastiCache resource using the ElastiCache management console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Depending what ElastiCache resource you want to remove a tag from, select either **Clusters** or **Snapshots**.
3. Select the ElastiCache resource you want to remove a tag from.
4. Select **Manage Tags** at the top of the screen.
5. For each tag that you want to remove from this resource, select the **Delete** check box to the right of the tag.



6. When finished, select **Apply Changes** to keep your changes, or **Cancel** to discard your changes.

Removing Tags from Your ElastiCache Resource (AWS CLI)

You can use the AWS CLI to remove tags from an existing ElastiCache resource by using the remove-tags-from-resource command.

The following code uses the AWS CLI to remove the tags with the keys `Service` and `Region` from the resource `myCluster` in the us-west-2 region.

The `resource-name` parameter value is in the format of an ARN: `arn:aws:elasticache:<region>:<customer id>:<resource type>:<resource name>`

For Linux, OS X, or Unix:

```
aws elasticache remove-tags-from-resource \
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:myCluster \
 --tag-keys Service Region
```

For Windows:

```
aws elasticache remove-tags-from-resource ^
 --resource-name arn:aws:elasticache:us-west-2:0123456789:cluster:myCluster ^
 --tag-keys Service Region
```

For more information, go to AWS CLI for ElastiCache remove-tags-from-resource.

Removing Tags from Your ElastiCache Resource (ElastiCache API)

You can use the ElastiCache API to remove tags from an existing ElastiCache resource by using the RemoveTagsFromResource action.

The following code uses the ElastiCache API to remove the tags with the keys `Service` and `Region` from the resource `myCluster` in the us-west-2 region.

The `ResourceName` parameter value is in the format of an ARN: `arn:aws:elasticache:<region>:<customer id>:<resource type>:<resource name>`

```
https://elasticache.us-west-2.amazonaws.com/
   ?Action=RemoveTagsFromResource
   &ResourceName=arn:aws:elasticache:us-west-2:0123456789:cluster:myCluster
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &TagKeys.member.1=Service
   &TagKeys.member.2=Region
   &Version=2015-02-02
   &Timestamp=20150202T192317Z
   &X-Amz-Credential=<credential>
```

Copying Tags to Your ElastiCache Resource

When you perform certain operations on your ElastiCache resources using the ElastiCache API or AWS CLI, if tags exist on the resource the tags are copied. The following list identifies those operations and what copying occurs.

- **CopySnapshot or copy-snapshot** – When you make a copy of a snapshot, if there are any tags on the source snapshot, they are copied to the copy.
- **CreateSnapshot or create-snapshot** – When you create a snapshot, if there are any tags on the source cluster, they are copied to the snapshot.
- **RestoreFromSnapshot or restore-from-snapshot** – When you restore a cluster from a snapshot created by ElastiCache, if there are any tags on the snapshot, they are copied to the restored cluster.

- **DeleteSnapshot or delete-snapshot** – When you delete a snapshot, if there are any tags on the snapshot, they are deleted with the snapshot.
- **DeleteCluster or delete-cluster** – When you delete a cluster, any tags on the cluster are deleted with the cluster. However, if you make a final snapshot, the tags are copied to the snapshot.

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

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, go to 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
        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=2877960fced9040b41b4feaca835fd5cfeb9264f768e6a0236c9143f915ffa56
```

For detailed information on the signing process and calculating the request signature, see the topic Signature Version 4 Signing Process and its subtopics.

Available Libraries

AWS provides software development kits (SDKs) for software developers who prefer to build applications using language-specific APIs instead of the Query API. These SDKs provide basic functions (not included in the APIs), such as request authentication, request retries, and error handling so that it is easier to get started. SDKs and additional resources are available for the following programming languages:

- Java
- Windows and .NET
- PHP
- Python
- Ruby

For information about other languages, go to 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, go to https://forums.aws.amazon.com/ .

Logging Amazon ElastiCache API Calls Using AWS CloudTrail

Amazon ElastiCache is integrated with AWS CloudTrail, a service that captures API calls made by or on behalf of ElastiCache in your AWS account and delivers the log files to an Amazon S3 bucket that you specify. CloudTrail captures API calls from the ElastiCache console, the ElastiCache API, or the ElastiCache CLI. Using the information collected by CloudTrail, you can determine what request was made to ElastiCache, the source IP address from which the request was made, who made the request, when it was made, and so on.

To learn more about CloudTrail, including how to configure and enable it, go to the *AWS CloudTrail User Guide*.

ElastiCache Information in CloudTrail

When CloudTrail logging is enabled in your AWS account, API calls made to ElastiCache actions are tracked in log files. For example, calls to the **CreateCacheCluster**, **DescribeCacheCluster**, and **ModifyCacheCluster** APIs generate entries in the CloudTrail log files. All of the ElastiCache actions are logged. For a full list of ElastiCache actions, go to http://docs.aws.amazon.com/AmazonElastiCache/latest/APIReference/.

Each log file contains not only ElastiCache records but also other AWS service records. CloudTrail determines when to create and write to a new log file based on a time period and file size.

Every log entry contains information about who generated the request. The user identity information in the log helps you determine whether the request was made with root or IAM user credentials, with temporary security credentials for a role or federated user, or by another AWS service. For more information, go to the documentation for the **userIdentity** field in the *CloudTrail Event Reference*.

You can store your log files in your bucket for as long as you want. You can also define Amazon S3 lifecycle rules to archive or delete log files automatically. By default, your log files are encrypted using Amazon S3 server-side encryption (SSE).

If you want to take quick action upon log file delivery, you can have CloudTrail publish Amazon SNS notifications when new log files are delivered. For more information, see Configuring Amazon SNS Notifications.

You can also aggregate ElastiCache log files from multiple AWS regions and multiple AWS accounts into a single Amazon S3 bucket. For more information, see Aggregating CloudTrail Log Files to a Single Amazon S3 Bucket.

Deciphering ElastiCache Log File Entries

CloudTrail log files can contain one or more log entries, where each entry is made up of multiple JSON-formatted events. A *log entry* represents a single request from any source and includes information about the requested action, any parameters, the date and time of the action, and so on. The log entries are not guaranteed to be in any particular order. That is, they are not an ordered stack trace of the public API calls.

The following example shows a CloudTrail log entry that records a `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":"&url-console-domain;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 records a `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":{
        "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":"&url-console-domain;elasticache/
home#client-download:",
        "cacheParameterGroup":{
            "cacheParameterGroupName":"default.memcached1.4",
            "cacheNodeIdsToReboot":{
            },
```

```
            "parameterApplyStatus":"in-sync"
        },
        "cacheClusterCreateTime":"Dec 1, 2014 10:16:06 PM",
        "preferredAvailabilityZone":"Multiple",
        "numCacheNodes":2,
        "cacheNodeType":"cache.m1.small",
        "cacheClusterStatus":"modifying",
        "autoMinorVersionUpgrade":true,
        "preferredMaintenanceWindow":"thu:05:00-thu:06:00",
        "cacheClusterId":"test-memcached",
        "engineVersion":"1.4.14",
        "cacheSecurityGroups":[
            {
                "status":"active",
                "cacheSecurityGroupName":"default"
            }
        ],
        "configurationEndpoint":{
            "address":"test-
memcached.example.cfg.use1prod.cache.amazonaws.com",
            "port":11211
        },
        "pendingModifiedValues":{
            "numCacheNodes":3
        }
    },
    "requestID":"807f4bc3-354c-11e4-9376-c1d979ba565a",
    "eventID":"e9163565-376f-4223-96e9-9f50528da645"
}
```

ElastiCache Turorials

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

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

Document History

The following table describes the important changes to the documentation since the last release of the *Amazon ElastiCache User Guide*.

- **API version:** 2015-02-02
- **Latest documentation update:** December 8, 2016

Change	Description	Date Changed
Support for Canada (Montreal) region	ElastiCache added support for the Canada (Montreal) region. Only node type M4 and T2 are currently supported in this region. For more information, see Supported Regions & Endpoints (p. 46) and Supported Node Types (p. 90) exceptions.	December 8, 2016
Support for M4 and R3 node types	ElastiCache added 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 Supported Regions & Endpoints (p. 46) and Supported Node Types (p. 90) exceptions.	November 1, 2016
US East 2 (Ohio) region support	ElastiCache added support for the US East (Ohio) Region (*us-east-2*) with M4, T2, and R3 node types. For more information, see Supported Regions & Endpoints (p. 46) and Supported Node Types (p. 90) exceptions.	October 17, 2016
Support for Redis Cluster	ElastiCache adds support for Redis Cluster (enhanced). Customers using Redis Cluster, can partition their data across up to 15 shards (node groups). Each shard supports replication with up to 5 read replicas per shard. Redis Cluster automatic failover times are about one fourth as long as those of earlier versions.	October 12, 2016

Change	Description	Date Changed
	This release includes a redesigned management console that uses terminology in keeping with industry usage. For more information, see the following topics: • Engines and Versions (p. 32) • ElastiCache Components & Features (p. 11) — note the sections on Nodes, Shards, Clusters, and Replication. • ElastiCache Terminology (p. 9)	
M4 node type support	ElastiCache added 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 Supported Node Types (p. 90), Memcached Node-Type Specific Parameters (p. 291), and Redis Node-Type Specific Parameters (p. 305).	August 3, 2016
Mumbai region support	ElastiCache added support for the Asia Pacific (Mumbai) Region. For more information, see Supported Regions & Endpoints (p. 46).	June 27, 2016
Snapshot export	ElastiCache added the ability to export a Redis snapshot so you can access it from outside ElastiCache. For more information, see Exporting a Backup (p. 249) in the *Amazon ElastiCache User Guide* and CopySnapshot in the *Amazon ElastiCache API Reference*.	May 26, 2016
Node type scale up	ElastiCache added the ability to scale up your Redis node type. For more information, see Scaling (p. 169).	March 24, 2016
Easy engine upgrade	ElastiCache added the ability to easily upgrade your Redis cache engine. For more information, see Upgrading Engine Versions (p. 41).	March 22, 2016
Support for R3 node types	ElastiCache added support for R3 node types in the China (Beijing) and South America (Sao Paulo) regions. For more information, see Supported Node Types (p. 90).	March 16, 2016
Accessing ElastiCache using a Lambda function	Added a tutorial on configuring a Lambda function to access ElastiCache in an Amazon VPC. For more information, see ElastiCache Turorials (p. 391).	February 12, 2016

Change	Description	Date Changed
Support for Redis 2.8.24	ElastiCache added support for Redis version 2.8.24 with improvements added since Redis 2.8.23. Improvements include bug fixes and support for logging bad memory access addresses. For more information, see Redis Version 2.8.24 (Enhanced) (p. 40) and Redis 2.8 Release Notes.	January 20, 2016
Support for Asia Pacific (Seoul) region	ElastiCache added support for the Asia Pacific (Seoul) (*ap-northeast-2*) region with t2, m3, and r3 node types.	January 6, 2016
Amazon ElastiCache console change.	Because the newer Redis versions provide a better and more stable user experience, Redis versions 2.6.13, 2.8.6, and 2.8.19 are no longer listed in the ElastiCache Management Console. For other options and more information, see Comparing Redis Versions (p. 38).	December 15, 2015
Support for Redis 2.8.23.	ElastiCache added support for Redis version 2.8.23 with improvements added since Redis 2.8.22. Improvements include bug fixes and support for the new parameter `close-on-slave-write` which, if enabled, disconnects clients who attempt to write to a read-only replica. For more information, see Redis Version 2.8.23 (Enhanced) (p. 40).	November 13, 2015
Support for Redis 2.8.22.	ElastiCache added support for Redis version 2.8.22 with ElastiCache added enhancements and improvements since version 2.8.21. Improvements include: • Implementation of a forkless save process that enables a successful save when low available memory could cause a forked save to fail. • Additional CloudWatch metrics – *SaveInProgress* and *ReplicationBytes*. • To enable partial synchronizations, the Redis parameter `repl-backlog-size` now applies to all clusters. For a complete list of changes and more information, see Redis Version 2.8.22 (Enhanced) (p. 40). This documentation release includes a reorganization of the documentation and removal of the ElastiCache command line interface (CLI) documentation. For command line use, refer to the AWS Command Line for ElastiCache.	September 28, 2015

Change	Description	Date Changed
Support for Memcached 1.4.28.	ElastiCache added support for Memcached version 1.4.24 and Memcached improvements since version 1.4.14. This release adds support for least recently used (LRU) cache management as a background task, choice of *jenkins* or *murmur3* as your hashing algorithm, new commands, and miscellaneous bug fixes. For more information, go to Memcached release notes and Comparing Memcached Versions (p. 37) in the ElastiCache User Guide.	August 27, 2015
Support for Redis 2.8.21. Support for Memcached Auto Discovery using PHP 5.6.	ElastiCache added support for Redis version 2.8.21 and Redis improvements since version 2.8.19. This Redis release includes several bug fixes. For more information, go to Redis 2.8 release notes. This release of Amazon ElastiCache adds support for Memcached Auto Discovery client for PHP version 5.6. For more information, go to Compiling the Source Code for the ElastiCache Cluster Client for PHP (p. 118).	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 , go to ElastiCache's Accessing ElastiCache Resources from Outside AWS (p. 348).	July 9, 2015
Node replacement messages added	ElastiCache added 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, go to ElastiCache's Event Notifications and Amazon SNS (p. 368).	June 11, 2015

Change	Description	Date Changed
Support for Redis v. 2.8.19.	ElastiCache added support for Redis version 2.8.19 and Redis improvements since version 2.8.6. This support includes support for: • The HyperLogLog data structure, with the Redis commands PFADD, PFCOUNT, and PFMERGE. • Lexicographic range queries with the new commands ZRANGEBYLEX, ZLEXCOUNT, and ZREMRANGEBYLEX. • Introduced a number of bug fixes, namely preventing a primary node from sending stale data to replica nodes by failing the master SYNC when a background save (bgsave) child process terminates unexpectedly. For more information on HyperLogLog, go to Redis new data structure: the HyperLogLog. For more information on PFADD, PFCOUNT, and PFMERGE, go to the Redis Documentation and click **HyperLogLog**.	March 11, 2015
Support for cost allocation tags	ElastiCache added support for cost allocation tags. For more information, see Monitoring Costs with Cost Allocation Tags (p. 373).	February 9, 2015
Support for AWS GovCloud (US) region	ElastiCache added support for the AWS GovCloud (US) (*us-gov-west-1*) region.	January 29, 2015
Support for EU (Frankfurt) region	ElastiCache added support for the EU (Frankfurt) (*eu-central-1*) region.	January 19, 2015
Multi-AZ with auto failover support for Redis replication groups	ElastiCache added support for Multi-AZ with automatic failover from the primary node to a read replica in a Redis replication group. ElastiCache monitors the health of the replication group. If the primary fails, ElastiCache automatically promotes a replica to primary, then replaces the replica. For more information, see Replication: Multi-AZ with Automatic Failover (Redis) (p. 196).	October 24, 2014
AWS CloudTrail logging of API calls supported	ElastiCache added support for using AWS CloudTrail to log all ElastiCache API calls. For more information, see Logging Amazon ElastiCache API Calls Using AWS CloudTrail (p. 387).	September 15, 2014
New instance sizes supported	ElastiCache added support for additional General Purpose (T2) instances. For more information, see Parameters and Parameter Groups (p. 272).	September 11, 2014

Change	Description	Date Changed
Flexible node placement supported for Memcached	ElastiCache added support for creating Memcached nodes across multiple Availability Zones. For more information, see Step 2: Launch a Cluster (p. 22).	July 23, 2014
New instance sizes supported	ElastiCache added support for additional General Purpose (M3) instances and Memory Optimized (R3) instances. For more information, see Parameters and Parameter Groups (p. 272).	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. 112).	May 13, 2014
Backup and restore for Redis clusters	In this release, ElastiCache allows customers to create snapshots of their Redis clusters, and create new clusters using these snapshots. A snapshot is a backup copy of the cluster at a specific moment in time, and consists of cluster metadata and all of the data in the Redis cache. Snapshots are stored in Amazon S3, and customers can restore the data from a snapshot into a new cluster at any time. For more information, see ElastiCache Backup & Restore (Redis) (p. 235).	April 24, 2014
Redis 2.8.6	ElastiCache supports Redis 2.8.6, in addition to Redis 2.6.13. With Redis 2.8.6, customers can improve the resiliency and fault tolerance of read replicas, with support for partial resynchronization, and a user-defined minimum number of read replicas that must be available at all times. Redis 2.8.6 also offers full support for publish-and-subscribe, where clients can be notified of events that occur on the server.	March 13, 2014
Redis cache engine	ElastiCache offers Redis cache engine software, in addition to Memcached. Customers who currently use Redis can "seed" a new ElastiCache Redis cache cluster with their existing data from a Redis snapshot file, easing migration to a managed ElastiCache environment. To support Redis replication capabilities, the ElastiCache API now supports replication groups. Customers can create a replication group with a primary Redis cache node, and add one or more read replica nodes that automatically stay synchronized with cache data in the primary node. Read-intensive applications can be offloaded to a read replica, reducing the load on the primary node. Read replicas can also guard against data loss in the event of a primary cache node failure.	September 3, 2013

Change	Description	Date Changed
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 Virtual Private Cloud (Amazon VPC) with ElastiCache (p. 316).	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 Virtual Private Cloud (Amazon VPC) with ElastiCache (p. 316).	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 Parameters and Parameter Groups (p. 272).	November 28, 2012
New cache node types	This release provides four additional cache node types.	November 13, 2012
Reserved cache nodes	This release adds support for reserved cache nodes.	April 5, 2012
New guide	This is the first release of *Amazon ElastiCache User Guide*.	August 22, 2011

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.