# Table of Contents

What Is Amazon ElastiCache? .............................................................................................................. 1
Use Cases ........................................................................................................................................ 2
  In-Memory Data Cache ................................................................................................................... 2
  Gaming Leaderboards (Redis Sorted Lists) ................................................................................... 3
  Messaging (Redis pub/sub) ........................................................................................................... 3
  Recommendation Data (Redis Counters & Hashes) ................................................................. 5
  Other Redis Uses .......................................................................................................................... 5
  Testimonials .................................................................................................................................. 5
ElastiCache Resources ....................................................................................................................... 6
Tutorial Videos .................................................................................................................................. 8
  Introductory Video Tutorials ......................................................................................................... 8
  Advanced Video Tutorials ........................................................................................................... 8
Components and Features ............................................................................................................... 11
  Nodes ........................................................................................................................................ 11
  Shards (Redis) ............................................................................................................................ 12
  Clusters ....................................................................................................................................... 12
  Replication ................................................................................................................................... 14
  Regions and Availability Zones ................................................................................................. 15
  Endpoints .................................................................................................................................... 16
  Parameter Groups ....................................................................................................................... 16
  Security ...................................................................................................................................... 17
  Security Groups .......................................................................................................................... 17
  Subnet Groups ............................................................................................................................ 17
  Backups/Snapshots (Redis) ........................................................................................................ 17
Events ............................................................................................................................................. 18
ElastiCache for Redis Terminology ................................................................................................. 19
Accessing ElastiCache ....................................................................................................................... 21
Managing ElastiCache ....................................................................................................................... 22
  Managing ElastiCache (Console) ................................................................................................. 22
  Managing ElastiCache (AWS CLI) .............................................................................................. 22
  Managing ElastiCache (ElastiCache API) ................................................................................ 22
Getting Started ............................................................................................................................... 23
  Determine Your Requirements [Every time] ............................................................................... 24
    Memory and Processor Requirements ..................................................................................... 24
    Scaling Requirements ............................................................................................................... 25
    Failover Requirements ............................................................................................................. 25
    Access Requirements ............................................................................................................... 25
    Region and Availability Zone Requirements ........................................................................... 25
Step 1: Create an AWS Account and Permissions ........................................................................ 27
  Step 1a: Create Your AWS Account ......................................................................................... 27
  Step 1b: Set Up Your Permissions (New ElastiCache Customers only) .................................... 27
Step 2: Launch a Cluster ............................................................................................................... 28
Step 3: (Optional) View Cluster Details ....................................................................................... 30
Step 4: Authorize Access ............................................................................................................... 32
  Step 4.1: Determine Environment ............................................................................................. 32
  Step 4.2: Grant Access ................................................................................................................ 34
Step 5: Connect to a Cluster's Node ............................................................................................ 36
  Step 5.1: Find your Node Endpoints .......................................................................................... 36
  Step 5.2: Connect to a Memcached Node .................................................................................. 36
  Step 5.2: Connect to a Redis Cluster or Replication Group ....................................................... 37
Step 6: Delete Your Cluster ........................................................................................................... 39
Where Do I Go From Here? ............................................................................................................ 40
Engines and Versions ....................................................................................................................... 41
Amazon ElastiCache User Guide

Describing Backups (AWS CLI) ................................................................. 306
Describing Backups (ElastiCache API) .................................................... 306
Copying a Backup .................................................................................... 308
  Copying a Backup (Console) ................................................................. 308
  Copying a Backup (AWS CLI) ............................................................... 308
  Copying a Backup (ElastiCache API) ................................................... 309
Exporting a Backup .................................................................................. 310
  Step 1: Create an Amazon S3 Bucket .................................................. 310
  Step 2: Grant ElastiCache Read Access to Your Amazon S3 Bucket .... 311
  Step 3: Export an ElastiCache Backup ................................................ 312
Restoring from a Backup ......................................................................... 317
  Restoring From a Backup (Console) .................................................... 318
  Restoring From a Backup (AWS CLI) ................................................... 319
  Restoring From a Backup (ElastiCache API) ......................................... 319
Seeding a Cluster with a Backup (Redis) ................................................ 320
  Step 1: Create a Redis Backup ............................................................ 321
  Step 2: Create an Amazon S3 Bucket and Folder ............................... 321
  Step 3: Upload Your Backup to Amazon S3 ....................................... 322
  Step 4: Grant ElastiCache Read Access to the .rdb File ................. 322
  Step 5: Seed the ElastiCache Cluster With the .rdb File File Data ....... 323
Tagging Backups ...................................................................................... 325
Deleting a Backup ................................................................................... 326
  Deleting a Backup (Console) ............................................................... 326
  Deleting a Backup (AWS CLI) ............................................................ 326
  Deleting a Backup (ElastiCache API) .................................................. 326
Redis Append Only Files (AOF) ............................................................. 327
Security Groups [EC2-Classic] ............................................................... 328
  Creating a Security Group ................................................................. 329
    Using the AWS Management Console ............................................ 329
    Using the AWS CLI ......................................................................... 329
    Using the ElastiCache API .............................................................. 329
  Listing Available Security Groups .................................................... 331
    Using the AWS Management Console ............................................ 331
    Using the AWS CLI ......................................................................... 331
    Using the ElastiCache API .............................................................. 331
  Viewing a Security Group ................................................................. 333
    Using the AWS Management Console ............................................ 333
    Using the AWS CLI ......................................................................... 333
    Using the ElastiCache API .............................................................. 333
  Authorizing Network Access to an Amazon EC2 Security Group ......... 335
    Using the AWS Management Console ............................................ 335
    Using the AWS CLI ......................................................................... 335
    Using the ElastiCache API .............................................................. 336
Parameters and Parameter Groups ......................................................... 337
  Parameter Management ..................................................................... 338
  Parameter Group Tiers ...................................................................... 339
  Creating a Parameter Group .............................................................. 340
    Creating a Parameter Group (Console) .......................................... 340
    Creating a Parameter Group (AWS CLI) ......................................... 340
    Creating a Parameter Group (ElastiCache API) .............................. 341
  Listing Parameter Groups by Name .................................................. 343
    Listing Parameter Groups by Name (Console) ............................... 343
    Listing Parameter Groups by Name (AWS CLI) ............................... 343
    Listing Parameter Groups by Name (ElastiCache API) .................... 344
  Listing a Parameter Group’s Values .................................................. 346
    Listing a Parameter Group’s Values (Console) .............................. 346
    Listing a Parameter Group’s Values (AWS CLI) ............................. 346
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 data store or cache environment in the cloud. It provides a high-performance, scalable, and cost-effective caching solution, while removing the complexity associated with deploying and managing a distributed cache environment.

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.
- Multi-AZ with Automatic Failover of a failed primary cluster to a read replica in Redis clusters that support replication (called replication groups in the ElastiCache API and AWS CLI).
- Flexible Availability Zone placement of nodes and clusters.
- Integration with other AWS services such as Amazon EC2, Amazon CloudWatch, AWS CloudTrail, and Amazon SNS to provide a secure, high-performance, managed in-memory caching solution.

Topics

- ElastiCache Use Cases (p. 2)
- Amazon ElastiCache Resources (p. 6)
- ElastiCache Tutorial Videos (p. 8)
- ElastiCache Components and Features (p. 11)
- ElastiCache for Redis Terminology (p. 19)
- Accessing Amazon ElastiCache (p. 21)
- Managing ElastiCache (p. 22)
ElastiCache Use Cases

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 New York 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 opt out of the slower site in favor of the faster site. Tests done at Amazon in 2007, cited in How Webpage Load Time Is Related to Visitor Loss, revealed that for every 100-ms (1/10 second) increase in load time, sales decrease 1 percent. If someone wants data, whether for a webpage or a report that drives business decisions, you can deliver that data faster if it is cached, much faster. Can your business afford to not cache your webpages 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 not cache your less frequently requested items? Even the most optimized database query or remote API call is going to be noticeably slower than retrieving a flat key from an in-memory cache. 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 (submillisecond latency) and inexpensive access to copies of data. Most data stores have areas of data that are frequently accessed but seldom updated. Additionally, querying a database is always slower and more expensive than locating a key in a key-value pair cache. Some database queries are especially expensive to perform, for example, queries that involve joins across multiple tables or queries with intensive calculations. By caching such query results, you pay the price of the query once and then are able to quickly retrieve the data multiple times without having to re-execute the query.

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 doesn't make sense to cache webpages 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. 92).

**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, receives 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 receive any messages published to `news.sports.golf`.

Redis pub/sub functionality has no relation to any key space. Therefore, it doesn't 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 you do 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**

To learn about how businesses like Airbnb, PBS, Esri, and others are using Amazon ElastiCache to grow their businesses with improved customer experience, see Testimonials.
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. 8) 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. 23) 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. 41)

  ElastiCache supports two engines—Memcached and Redis. This topic helps you determine which engine is best for your scenario.

- **Choosing Your Node Size** (p. 99)

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

- **Best Practices for Amazon ElastiCache** (p. 76)

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

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

- **AWS Command Line Interface Documentation**

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

- **AWS CLI Documentation for ElastiCache**

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

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

- **Tools for Amazon Web Services**

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

- **Using the ElastiCache API** (p. 478)
If you don't want to use the AWS SDKs, you can interact with ElastiCache directly using the Query API. You can find troubleshooting tips and information on creating and authenticating requests and handling responses in this section.

- Amazon ElastiCache API Reference

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

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

Contents
- Introductory Video Tutorials (p. 8)
  - DAT204—Building Scalable Applications on AWS NoSQL Services (re:Invent 2015) (p. 8)
  - DAT207—Accelerating Application Performance with Amazon ElastiCache (AWS re:Invent 2013) (p. 8)
- Advanced Video Tutorials (p. 8)
  - DAT305—Amazon ElastiCache Deep Dive (re:Invent 2017) (p. 9)
  - DAT306—Amazon ElastiCache Deep Dive (re:Invent 2016) (p. 9)
  - DAT317—How IFTTT Uses ElastiCache for Redis to Predict Events (re:Invent 2016) (p. 9)
  - SDD402—Amazon ElastiCache Deep Dive (re:Invent 2014) (p. 10)
  - DAT307—Deep Dive into Amazon ElastiCache Architecture and Design Patterns (re:Invent 2013) (p. 10)

Introductory Video Tutorials

The following videos introduce you 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 should come out of this session having a better understanding of NoSQL and its powerful capabilities, ready to tackle your database challenges with confidence.

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

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

In this 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

The following videos cover more advanced Amazon ElastiCache topics.
Topics

- DAT305—Amazon ElastiCache Deep Dive (re:Invent 2017) (p. 9)
- DAT306—Amazon ElastiCache Deep Dive (re:Invent 2016) (p. 9)
- DAT317—How IFTTT Uses ElastiCache for Redis to Predict Events (re:Invent 2016) (p. 9)
- SDD402—Amazon ElastiCache Deep Dive (re:Invent 2014) (p. 10)
- DAT307—Deep Dive into Amazon ElastiCache Architecture and Design Patterns (re:Invent 2013) (p. 10)

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

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

The video introduces the following:

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

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

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

DAT317—How IFTTT Uses ElastiCache for Redis to Predict Events (re:Invent 2016)

IFTTT is a free service that empowers people to do more with the services they love, from automating simple tasks to transforming how someone interacts with and controls their home. IFTTT uses ElastiCache for Redis to store transaction run history and schedule predictions as well as indexes for log documents on Amazon S3. View this session to learn how the scripting power of Lua and the data types of Redis allow people to accomplish something they can't elsewhere.

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

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.
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.

ElastiCache Components and Features

In the topics in this section, you can find an overview of the major components of an Amazon ElastiCache deployment.

Topics
- ElastiCache Nodes (p. 11)
- ElastiCache Shards (Redis) (p. 12)
- ElastiCache Clusters (p. 12)
- ElastiCache Replication (Redis) (p. 14)
- Regions and Availability Zones (p. 15)
- ElastiCache Endpoints (p. 16)
- ElastiCache Parameter Groups (p. 16)
- ElastiCache Security (p. 17)
- ElastiCache Security Groups (p. 17)
- ElastiCache Subnet Groups (p. 17)
- ElastiCache Backups/Snapshots (Redis) (p. 17)
- ElastiCache Events (p. 18)

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 chosen 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. 199).

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. 117).

You can purchase nodes on a pay-as-you-go basis, where you only pay for your use of a node. Or you can purchase reserved nodes at a significantly reduced hourly rate. If your usage rate is high, purchasing reserved nodes can save you money. Suppose that your cluster is almost always in use, and you occasionally add nodes to handle use spikes. In this case, you can purchase a number of reserved nodes to run most of the time and purchase pay-as-you-go nodes for the times you occasionally need to add nodes. For more information on reserved nodes, see ElastiCache Reserved Nodes (p. 107).

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, your application immediately knows about all of the nodes in the cluster and can connect to all of them. You don't need to hard code the individual cache node endpoints in your application. For more information on Auto Discovery, see Node Auto Discovery (Memcached) (p. 123).

For more information on nodes, see ElastiCache Nodes (p. 97).
ElastiCache Shards (Redis)

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

A multiple node shard implements replication by having one read/write primary node and 1–5 replica nodes. For more information, see ElastiCache Replication (Redis) (p. 235).

Redis shard configurations

For more information on shards, see Shards (Redis) (p. 153).

ElastiCache Clusters

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

A Memcached cluster is a logical grouping of one or more ElastiCache Nodes (p. 11). Data is partitioned across the nodes in a Memcached 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 and from a cluster

For more detailed information, see the following related topics:

- ElastiCache Clusters (p. 154) and ElastiCache Nodes (p. 97)
  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. 83)

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

Typical Cluster Configurations

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

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

A Redis cluster contains 1–15 shards (in the API, called 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 1–20 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. 123).

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 has minimal impact upon your entire cluster and application. For more information, see Mitigating Failures (p. 83).

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. 88).

Redis Clusters

A Redis (cluster mode enabled) cluster contains 1–15 shards (in the API and CLI, called node groups). Redis (cluster mode disabled) clusters always contain just one shard (in the API and CLI, one node group). A Redis shard contains 1–6 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. 83).

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 Single-Node Redis (cluster mode disabled) Clusters (p. 204) or Scaling Redis (cluster mode disabled) Clusters with Replica Nodes (p. 213).
ElastiCache Replication (Redis)

Before you continue reading here, see ElastiCache for Redis Terminology (p. 19) to better understand the differences in terminology between the ElastiCache console and the ElastiCache API and AWS CLI.

Replication is implemented by grouping from 2 to 6 nodes in a shard (in the API and CLI, called a 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. 83).

Redis (cluster mode disabled) clusters support one shard (in the API and CLI, called a node group). Redis (cluster mode enabled) clusters support from 1–15 shards (in the API and CLI, called 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 and CLI perspective uses different terminology to maintain compatibility with previous versions, but the results are the same. The following table shows the API and CLI terms for implementing replication.

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

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

<table>
<thead>
<tr>
<th></th>
<th>Redis (cluster mode disabled)</th>
<th>Redis (cluster mode enabled)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Shards (Node groups)</td>
<td>1</td>
<td>1 to 15</td>
</tr>
<tr>
<td>Replicas per shard (node group)</td>
<td>0 to 5</td>
<td>0 to 5</td>
</tr>
</tbody>
</table>
## Regions and Availability Zones

Amazon ElastiCache is available in multiple regions around the world. Thus, you can launch ElastiCache clusters in the locations that meet your business requirements. For example, you can launch in the 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 can achieve the greatest possible fault tolerance. For more information about regions and Availability Zones, see Choosing Regions and Availability Zones (p. 58).

### Redis (cluster mode disabled) vs. Redis (cluster mode enabled)

<table>
<thead>
<tr>
<th>Feature</th>
<th>Redis (cluster mode disabled)</th>
<th>Redis (cluster mode enabled)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data partitioning</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Add/Delete replicas</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Add/Delete node groups</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Supports scale up</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Supports engine upgrades</td>
<td>Yes</td>
<td>N/A</td>
</tr>
<tr>
<td>Promote replica to primary</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Multi-AZ with automatic failover</td>
<td>Optional</td>
<td>Required</td>
</tr>
<tr>
<td>Backup/Restore</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

**Notes:**

- If any primary has no replicas and the primary fails, you will lose all that primary’s data.
- Backup and restore can be used to migrate to Redis (cluster mode enabled).
- Backup and restore can be used to resize your Redis (cluster mode enabled) cluster.

All of the shards (in the API and 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. 240).

### Replication: Limits and 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. 83).
Regions and Availability Zones

For information on regions supported by ElastiCache and their endpoints, see Supported Regions & Endpoints (p. 60).

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 automatically knows each node endpoint, even after adding or removing nodes from the cluster. For more information, see Node Auto Discovery (Memcached) (p. 123).

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. 62).

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. 337).
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. 388).

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. 328).

ElastiCache Subnet Groups

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

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

For more information about cache subnet group usage in an Amazon VPC environment, see Amazon Virtual Private Cloud (Amazon VPC) with ElastiCache (p. 388), Step 4: Authorize Access (p. 32), and Subnets and Subnet Groups (p. 379).

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.
Depending upon the version of Redis running on your cluster, the backup process requires differing amounts of reserved memory to succeed. For more information, see:

- ElastiCache Backup and Restore (Redis) (p. 293)
- How Synchronization and Backup are Implemented (p. 253)
- Performance Impact of Backups (p. 294)
- Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 77)

**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. 457).
ElastiCache for Redis Terminology

In October 2016, Amazon ElastiCache launched support for Redis 3.2. Among other things, the new features launched then added support for partitioning your data across up to 15 shards (called node groups in the ElastiCache API and AWS CLI). To preserve compatibility with previous versions, we extended API version 2015-02-02 operations to include the new Redis functionality. In parallel, we began using terminology in the ElastiCache console that is used in this new functionality and common across the industry. These changes mean that at some points, the terminology used in the API and CLI might be different from the terminology used in the console. The following list identifies terms that might differ between the API and CLI and the console.

**Cache Cluster or 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 **Create Cluster** button, which launches the process to create a cluster with or without replica nodes.

The ElastiCache API and AWS CLI continue to use the terms as they have in the past.

**Cluster vs. Replication Group**

The console now uses the term *cluster* for all ElastiCache for Redis clusters. The console uses the term cluster in all these circumstances:

- When the cluster is a single node Redis cluster.
- When the cluster is a Redis (cluster mode disabled) cluster that supports replication within a single shard (in the API and CLI, called a *node group*).
- When the cluster is a Redis (cluster mode enabled) cluster that supports replication within 1–15 shards.

The following diagram illustrates the various topologies of ElastiCache for Redis clusters from the console's perspective.

**ElastiCache for Redis: Console View**

![ElastiCache for Redis clusters (Console view)](image)

The ElastiCache API and AWS CLI operations still distinguish single node ElastiCache for Redis clusters from multi-node replication groups. The following diagram illustrates the various ElastiCache for Redis topologies from the ElastiCache API and AWS CLI perspective.
ElastiCache for Redis: API/CLI View

ElastiCache for Redis cluster and replication groups (API and CLI view)
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, you can access your ElastiCache instance from an Amazon EC2 in a different Amazon VPC.

If you launched your ElastiCache instance in EC2 Classic, you allow the EC2 instance to access your cluster by granting the Amazon EC2 security group associated with the instance access to your cache security group. By default, access to a cluster is restricted to the account that launched the cluster.

For more information on granting Amazon EC2 access to your cluster, see Step 4: Authorize Access (p. 32) and Accessing ElastiCache Resources from Outside AWS (p. 441).
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 CLI Command 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. 478).
Getting Started with Amazon ElastiCache

Beginning with determining the requirements for your cluster and 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 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. 235).

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
- Determine Your Requirements [Every time] (p. 24)
- Step 1: Create an AWS Account and Set Up Permissions [One time] (p. 27)
- Step 2: Launch a Cluster (p. 28)
- Step 3: (Optional) View Cluster Details (p. 30)
- Step 4: Authorize Access (p. 32)
- Step 5: Connect to a Cluster's Node (p. 36)
- Step 6: Delete Your Cluster [Avoid Unnecessary Charges] (p. 39)
- Where Do I Go From Here? (p. 40)
Determine Your Requirements [Every time]

Before you create a cluster or replication group, you should always determine the requirements for the cluster so that when you create the cluster or replication group it will meet your business needs and not need to be redone.

Topics
- Memory and Processor Requirements (p. 24)
- Scaling Requirements (p. 25)
- Failover Requirements (p. 25)
- Access Requirements (p. 25)
- Region and Availability Zone Requirements (p. 25)

Memory and Processor Requirements

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

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

Memcached Cluster Configuration

Memcached clusters are comprised of from 1 to 20 nodes. The data in a Memcached cluster is partitioned across all the nodes in the cluster. Your application connects with a Memcached cluster using a network address called an Endpoint. Each node in a Memcached cluster has its own endpoint which your application can use to read from or write to a specific node. In addition to the node endpoints, the Memcached cluster itself has an endpoint called the Configuration Endpoint which your application can use to read from or write to the cluster, leaving the determination of which node to read from or write to up to Automatic Discovery.

Redis Cluster Configurations

Amazon ElastiCache for Redis clusters provide a variety of node configurations. There are three node configurations when running the ElastiCache for Redis engine. Your application connects with your ElastiCache for Redis cluster using a network address called an Endpoint. Depending upon your ElastiCache for Redis cluster configuration, you will use the endpoints differently.

- Single Node

  The simplest configuration is a single node cluster. The node in a single-node Redis cluster must have sufficient memory to accommodate all your in-memory data plus overhead. A single-node Redis cluster does not provide high availability as there is no replication of data across multiple nodes.

- Single Shard Multiple Nodes

  A single shard configuration groups up to 6 nodes in a single cluster, a read/write Primary node and up to 5 read-only Replica nodes. Like the single node configuration, each node must be able to accommodate all your in-memory data plus overhead. Configurations with 1 or more replica nodes provide high availability by replicating your in-memory data across all nodes in the shard. Thus if one node fails you still have your data in one or more of the replica nodes.

- Multiple Shards Multiple Nodes
A multiple shard configuration partitions your data across up to 20 shards in the cluster. Each shard has a read/write Primary node and up to 5 read-only replica nodes. Since your in-memory data is partitioned across multiple shards, each node in a shard only needs enough memory to accommodate its copy of the shard's portion of your in-memory data plus overhead. Shard configurations with 1 or more replica nodes provide high availability by replicating your in-memory data across all nodes in the shard. Thus if one node fails you still have the shard's data in one or more of the replica nodes.

For more information, see Choosing Your Node Size (p. 99) in this guide.

### Scaling Requirements

All clusters can be scaled up by creating a new cluster with the new, larger node type. When scaling up a Memcached cluster the new cluster will start out empty. When scaling up a Redis cluster, you can seed the new cluster with data from the old cluster so that the new cluster is populated with data when you start using it.

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

To scale a Redis multiple shard cluster, you add or remove shards from the cluster. If your application is connecting to the cluster's configuration endpoint, you do not need to make any changes in your application when you add or remove shards.

For more information, see Scaling (p. 199) in this guide.

### Failover Requirements

Depending upon how critical access to a cluster's data is to your business, you may want to enable automatic failover on your cluster. When Automatic Failover is enabled, if a Redis Primary node fails for any reason, one of the shard's read-only replica nodes is promoted to Primary. If Automatic Failover is not enabled and the Primary fails, ElastiCache for Redis spins up a new node to fill the Primary role. This operation is much more time consuming than failing over to a replica node.

Automatic failover is only available on clusters running the Redis engine with multiple nodes.

For more information, see Replication: Multi-AZ with Automatic Failover (Redis) (p. 240) in this guide.

### Access Requirements

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

If you launched your cluster into EC2-VPC you need to grant network ingress to the cluster. If you launched your cluster into EC2-Classic you need to grant the Amazon Elastic Compute Cloud security group associated with the instance access to your ElastiCache security group. For detailed instructions, see Step 4: Authorize Access (p. 32) in this guide.

### Region and Availability Zone Requirements

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

- Choosing Regions and Availability Zones (p. 58)
- Mitigating Failures (p. 83)
Step 1: Create an AWS Account and Set Up Permissions [One time]

In order to use Amazon ElastiCache you must have an active AWS account and permissions to access ElastiCache as well as other AWS resources.

Step 1a: Create Your AWS Account

If you don't already have an AWS account, you'll be prompted to create one when you sign up. You will only be charged for usage time on AWS services that you sign up for.

To create an AWS account

1. Open https://aws.amazon.com/, and then choose Create an AWS Account.
   
   Note
   
   This might be unavailable in your browser if you previously signed into the AWS Management Console. In that case, choose Sign in to a different account, and then choose Create a new AWS account.

2. Follow the online instructions.

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

Step 1b: Set Up Your Permissions (New ElastiCache Customers only)

Amazon ElastiCache creates and uses Service linked roles (SLR) to provision resources and access other AWS resources and services on your behalf. In order for ElastiCache to create the SLR for you, you must use the AmazonElastiCacheFullAccess AWS managed policy, which comes pre-provisioned with permission that the service requires to create a Service Linked Role on your behalf.

If you are not using the default policy and choose to use a custom managed policy, ensure you have either permissions to call iam:createServiceLinkedRole or you have created the ElastiCache Service Linked Role.

For more information, see:

- Creating a New Policy (IAM)
- AWS Managed (Predefined) Policies for Amazon ElastiCache (p. 414)
- Using Service-Linked Roles for ElastiCache (p. 417)
Step 2: Launch a Cluster

Before you continue, be sure you have completed Determine Your Requirements [Every time] (p. 24).

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. 380).

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 Launch Cluster.
3. From the dropdown in the upper right corner, choose the region you want to launch this cluster in.
4. For Cluster engine, choose Redis.
5. Make sure Cluster Mode enabled (Scale Out) is not chosen.
6. 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 choose 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 choose "Create new" to create a new parameter group to use with this cluster. For this exercise, accept the default parameter group.

      For more information, see Creating a Parameter Group (p. 340).
   
   e. For Node type, choose the node type that you want to use for this cluster. For this exercise, above the table choose the t2 instance family, choose cache.t2.small, and finally choose Save.

      For more information, see Choosing Your Node Size (p. 99).
   
   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, choose None.

      When you choose None, the Replication group description field disappears.

7. Choose Advanced Redis settings and complete the section as follows:
Step 2: Launch a Cluster

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) Cluster with Replicas from Scratch (p. 267).

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

For more information, see Subnets and Subnet Groups (p. 379).

b. For Availability zone(s), you have two options.
   • No preference – ElastiCache will choose the Availability Zone.
   • Specify availability zones – You specify the Availability Zone for your cluster.

For this exercise, choose Specify availability zones and then choose an Availability Zone from the list below Primary.

For more information, see Choosing Regions and Availability Zones (p. 58).

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

For more information, see ElastiCache and Security Groups (p. 406).

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 Seeding a New Cluster with an Externally Created Backup (Redis) (p. 320).

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 and Restore (Redis) (p. 293).

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, choose No preference.

For more information, see Maintenance Window (p. 56).

g. For Notifications, leave it as Disabled.

8. Choose 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. 28).

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.

![Cluster List](image)

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. Doing this displays 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. Doing this displays 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.
### Step 3: (Optional) View Cluster Details

**API Version 2015-02-02**

<table>
<thead>
<tr>
<th>Node ID</th>
<th>Status</th>
<th>Current Role</th>
<th>Port</th>
<th>Endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>test-no-001</td>
<td>available</td>
<td>primary</td>
<td>6379</td>
<td></td>
</tr>
<tr>
<td>test-no-002</td>
<td>available</td>
<td>replica</td>
<td>6379</td>
<td></td>
</tr>
<tr>
<td>test-no-003</td>
<td>available</td>
<td>replica</td>
<td>6379</td>
<td></td>
</tr>
<tr>
<td>test-no-004</td>
<td>available</td>
<td>replica</td>
<td>6379</td>
<td></td>
</tr>
</tbody>
</table>

**Time Range:** Last Hour

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

**CPU Utilization (Percent)**
- test-nc-001
- test-nc-003

**Swap Usage (Bytes)**

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

- Access Patterns for Accessing an ElastiCache Cluster in an Amazon VPC (p. 393)
- Accessing ElastiCache Resources from Outside AWS (p. 441)

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

Steps to authorize access

- **Step 4.1: Determine the Cluster's Environment (p. 32)**
  - Determining Your Clusters Platform using the ElastiCache Console (p. 32)
  - Determining Your Clusters Platform using the AWS CLI (p. 33)
- **Step 4.2: Grant Access (p. 34)**
  - You Launched Your Cluster into EC2-VPC (p. 34)
  - You Launched Your Cluster Running in EC2-Classic (p. 35)

Step 4.1: Determine the Cluster's Environment

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

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

Determining Your Clusters Platform using the ElastiCache Console

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

**To determine a cluster's platform using the ElastiCache console**

2. In the left navigation pane, choose the engine that is running on your cluster – either **Memcached** or **Redis**.
3. In the list of clusters, expand the cluster you want to authorize access to by choosing the box to the left of the cluster name.
4. Locate **Subnet group**.
Step 4.1: Determine Environment

- If the **Subnet group** has a name, as shown here, you launched your cluster in EC2-VPC and should continue at You Launched Your Cluster into EC2-VPC (p. 34).
- If there is a dash (-) instead of a **Subnet group** name, you launched your cluster in EC2-Classic and should continue at You Launched Your Cluster Running in EC2-Classic (p. 35).

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

**Determining Your Clusters Platform using the AWS CLI**

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

**To determine a cluster's platform using the AWS CLI**

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

   For Linux, macOS, or Unix:
   ```bash
   aws elasticache describe-cache-clusters \
   --show-cache-cluster-details \
   --cache-cluster-id redis-two
   ```

   For Windows:
   ```bash
   aws elasticache describe-cache-clusters ^
   --show-cache-cluster-details ^
   --cache-cluster-id redis-two
   ```

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

   ```json
   {
     "CacheClusters": [
       {
         "Engine": "redis",
         "AuthTokenEnabled": false,
         "CacheParameterGroup": {
           "CacheNodeIdsToReboot": [],
           "CacheParameterGroupName": "default.redis3.2",
           "ParameterApplyStatus": "in-sync"
         },
         "CacheClusterId": "redis-two-001",
         "CacheSecurityGroups": [],
         "NumCacheNodes": 1,
         "AtRestEncryptionEnabled": false,
         "CacheClusterCreateTime": "2018-01-16T20:09:34.449Z",
         "CacheParameterGroupDescription": "Default parameter group for Redis 3.2"
       }
     ]
   }
   ```
Step 4.2: Grant Access

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/.
3. From the list of security groups, choose the security group for your Amazon VPC. Unless you created a security group for ElastiCache use, this security group will be named default.
4. Choose the Inbound tab, and then do the following:
   a. Choose Edit.
   b. Choose Add rule.
   c. In the Type column, choose Custom TCP rule.
   d. In the Port range box, type the port number for your cluster node. This number must be the same one that you specified when you launched the cluster. The default ports are as follows:
      - Memcached: port 11211
      - Redis: port 6379
   e. In the Source box, choose Anywhere which has the port range (0.0.0.0/0) so that any Amazon EC2 instance that you launch within your Amazon VPC can connect to your ElastiCache nodes.
Important
Opening up the ElastiCache cluster to 0.0.0.0/0 (Step 4.e.) does not expose the cluster to the Internet because it has no public IP address and therefore cannot be accessed from outside the VPC. However, the default security group may be applied to other Amazon EC2 instances in the customer’s account, and those instances may have a public IP address. If they happen to be running something on port 6379, then that service could be exposed unintentionally. Therefore, we recommend creating a VPC Security Group that will be used exclusively by ElastiCache. For more information, see Custom Security Groups.

f. Choose Save.

When you launch an Amazon EC2 instance into your Amazon VPC, that instance will be able to connect to your ElastiCache cluster.

You Launched Your Cluster Running in EC2-Classic

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

To grant an Amazon EC2 security group access to a cluster

2. To see a list of security groups, from the left navigation pane, choose Security Groups.
   
   Important
   If Security Groups is not listed in the navigation pane, you launched your cluster in EC2-VPC rather than EC2-Classic and should follow the instructions at You Launched Your Cluster into EC2-VPC (p. 34).
3. Choose the box to the left of default security group.
4. From the list at the bottom of the screen, choose the EC2 Security Group Name you want to authorize.
5. To authorize access, choose Add.

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

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

Step 5: Connect to a Cluster's Node

Before you continue, be sure you have completed Step 4: Authorize Access (p. 32).

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. 32).

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. 62)
- Finding a Memcached Cluster's Endpoints (Console) (p. 63)
- Finding a Redis (cluster mode disabled) Cluster's Endpoints (Console) (p. 65)
- Finding a Redis (cluster mode enabled) Cluster's Endpoints (Console) (p. 67)
- Finding Endpoints (AWS CLI) (p. 69)
- Finding Endpoints (ElastiCache API) (p. 73)

Step 5.2: Connect to a Memcached Node

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 the cluster's node. To do so, you must first determine the node endpoint. Because this is a single-node Redis (cluster mode disabled) cluster, this endpoint is used for both read and write operations.

**To find the endpoint for a single-node Redis (cluster mode disabled) cluster**

2. From the navigation pane, choose Redis.
3. From the list of Redis clusters, choose the box to the left of the single-node Redis (cluster mode disabled) cluster you just created (1 in the graphic).
4. In the cluster's details section, find the Primary Endpoint (2 in the graphic).
5. To the right of Primary Endpoint, locate and highlight the endpoint (3 in the graphic) and copy it to your clipboard for use in Step 5.2.

The form of the endpoint is in the format `cluster-name.xxxxxxxxx.node-id.region-and-az.cache.amazonaws.com:port`, as shown here:

```
redis-01.19gh21.0001.usw2.cache.amazonaws.com:6379
...(output omitted)....
```

Total download size: 63 k
Installed size: 109 k
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.

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

   This will produce output similar to the following.

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

6. At the command prompt of your Amazon EC2 instance, type the following command, substituting the endpoint of your node and port for those shown in this example.

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

   You are now connected to the cluster and can run Memcached commands like the following.

   ```bash
   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
   ```
sudo yum install gcc

This will produce output similar to the following.

```
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:

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

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

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

This results in a Redis command prompt similar to the following.

```
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 like the following.

```
set a "hello"     // Set key "a" with a string value and no expiration
OK
get a "hello"     // Get value for key "a"
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
"Good-bye"          // wait 5 seconds
get b  // key has expired, nothing returned
(nil)
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. 28).

**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/](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. 41)
  ElastiCache supports two engines—Memcached and Redis. This topic helps you determine which engine is best for your scenario.
- Choosing Your Node Size (p. 99)
  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 choosing the best node size.
- Best Practices for Amazon ElastiCache (p. 76)
  Identify and address issues that can impact the efficiency of your cluster.
Engines and Versions

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

**Important**
After you create a cache cluster or replication group, you can upgrade to a newer engine version (see Upgrading Engine Versions (p. 54)), 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**
- Choosing an Engine: Memcached, Redis (cluster mode disabled), or Redis (cluster mode enabled) (p. 42)
- Determine Available Engine Versions (p. 45)
- ElastiCache for Memcached Versions (p. 47)
- ElastiCache for Redis Versions (p. 49)
- Upgrading Engine Versions (p. 54)
- Maintenance Window (p. 56)
Choosing 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.

**Choose 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 cache objects, such as a database.

**Choose a version of ElastiCache for Redis if the following apply to your situation:**

- **Choose ElastiCache for Redis 3.2.10 if you require the ability to dynamically add or remove shards from your Redis (cluster mode enabled) cluster:**
  
  Important
  
  Currently ElastiCache for Redis 3.2.10 does not support encryption.

  For more information, see:
  
  - Best Practices: Online Resharding (p. 89)
  - At-rest encryption. For more information, see Online Resharding and Shard Rebalancing for ElastiCache for Redis—Redis (cluster mode enabled) (p. 226)

- **Choose ElastiCache for Redis 3.2.6 if you require all the functionality of the earlier Redis versions plus:**
  
  - In-transit encryption. For more information, see Amazon ElastiCache for Redis In-Transit Encryption (p. 429)
  - At-rest encryption. For more information, see Amazon ElastiCache for Redis At-Rest Encryption (p. 434)
  - HIPAA compliance certification. For more information, see HIPAA Compliance for Amazon ElastiCache for Redis (p. 439)

- **Choose Redis 3.2.4 (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 replicas in a node group (partition).

- **Choose Redis 2.8.x or Redis 3.2.4 (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.
Choosing an Engine: Memcached, Redis (cluster mode disabled), or Redis (cluster mode enabled)

- You need persistence of your key store.
- You need to replicate your data from the primary to one or more read replicas for read intensive applications.
- You need automatic failover if your primary node fails.
- You need publish and subscribe (pub/sub) capabilities—to inform clients about events on the server.
- You need backup and restore capabilities.
- You need to support multiple databases.
**Comparison summary of Memcached, Redis (cluster mode disabled), and Redis (cluster mode enabled)**

<table>
<thead>
<tr>
<th></th>
<th>Memcached</th>
<th>Redis (cluster mode disabled)</th>
<th>Redis (cluster mode enabled)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Engine versions</strong></td>
<td>1.4.x</td>
<td>2.8.x and 3.2.x</td>
<td>3.2.x</td>
</tr>
<tr>
<td><strong>Data types</strong></td>
<td>Simple</td>
<td>Redis 2.8.x - Complex *</td>
<td>Complex</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Redis 3.2.x - Complex</td>
<td></td>
</tr>
<tr>
<td><strong>Online resharding</strong></td>
<td>No</td>
<td>3.2.10 only</td>
<td>3.2.10 only</td>
</tr>
<tr>
<td><strong>Encryption</strong></td>
<td>No</td>
<td>3.2.6 only</td>
<td>3.2.6 only</td>
</tr>
<tr>
<td><strong>HIPAA Compliance</strong></td>
<td>No</td>
<td>3.2.6 only</td>
<td>3.2.6 only</td>
</tr>
<tr>
<td><strong>Multi-threaded</strong></td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td><strong>Cluster is modifiable</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>Limited</td>
</tr>
<tr>
<td><strong>Node type upgrade</strong></td>
<td>No</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Engine upgrading</strong></td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td><strong>Data partitioning</strong></td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Persistence of key store</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>High availability (Replication)</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Automatic Failover of Primary</strong></td>
<td>No</td>
<td>Optional</td>
<td>Required</td>
</tr>
<tr>
<td><strong>Pub/Sub capabilities</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Sorted lists</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Counters &amp; hashes</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Backup/Restore capabilities</strong></td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Geospatial indexing</strong></td>
<td>No</td>
<td>Redis 2.8.x - No</td>
<td>Yes</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Redis 3.2.x - Yes</td>
<td></td>
</tr>
</tbody>
</table>

**Notes:**

- string, objects (like databases)
- * string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog
- string, sets, sorted sets, lists, hashes, bitmaps, hyperloglog, geospatial indexes

After you choose the engine for your cluster, we recommend that you use the most recent version of that engine. For more information, see [ElastiCache for Memcached Versions (p. 47)](https://aws.amazon.com) or [ElastiCache for Redis Versions (p. 49)](https://aws.amazon.com).
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 choose 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. 156) or Creating a Redis Cluster with Replicas from Scratch (p. 260).

Determine Available Engine Versions (AWS CLI)

To determine which engine versions are available in a region, use the describe-cache-engine-versions operation. Use the optional parameter --region to specify which region you want the available engine versions for. If you omit the --region parameter, engine versions are described for your current region.

```
aws elasticache describe-cache-engine-versions --region us-east-2
```

The output of this operation should look 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. Use the optional parameter Region to specify which region you want the available engine versions for. If you omit the Region parameter, engine versions are described for your current region.
https://elasticache.us-west-2.amazonaws.com/
?Action=DescribeCacheEngineVersions
&Region=us-east-2
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>

For more information, see DescribeCacheEngineVersions.
ElastiCache supports the following Memcached versions and upgrading to newer versions. When upgrading to a newer version, pay careful attention to the conditions which if not met will cause your upgrade to fail.

**ElastiCache for Memcached Versions**

- Upgrading to a Newer Version (p. 47)
- Memcached Version 1.4.34 (p. 47)
- Memcached Version 1.4.33 (p. 47)
- Memcached Version 1.4.24 (p. 48)
- Memcached Version 1.4.14 (p. 48)
- Memcached Version 1.4.5 (p. 48)

**Upgrading to a Newer Version**

To upgrade to a newer Memcached version, modify your cache cluster specifying the new engine version you want to use. Upgrading to a newer Memcached version is a destructive process – you lose your data and start with a cold cache. For more information, see Modifying an ElastiCache Cluster (p. 179).

You should be aware of the following requirements when upgrading from an older version of Memcached to Memcached version 1.4.33 or newer. CreateCacheCluster and ModifyCacheCluster fails under the following conditions:

- If `slab_chunk_max > max_item_size`.
- If `max_item_size modulo slab_chunk_max != 0`.
- If `max_item_size > ((max_cache_memory - memcached_connections_overhead) / 4)`.

The value `(max_cache_memory - memcached_connections_overhead)` is the node's memory useable for data. For more information, see Memcached Connection Overhead (p. 359).

**Memcached Version 1.4.34**

ElastiCache for Memcached version 1.4.34 adds no new features to version 1.4.33. Version 1.4.34 is a bug fix release that is larger than the usual such release.

For more information, see Memcached 1.4.34 Release Notes at Memcached on GitHub.

**Memcached Version 1.4.33**

Memcached improvements added since version 1.4.24 include the following:

- Ability to dump all of the metadata for a particular slab class, a list of slab classes, or all slab classes. For more information, see Memcached 1.4.31 Release Notes.
- Improved support for large items over the 1 megabyte default. For more information, see Memcached 1.4.29 Release Notes.
- Ability to specify how long a client can be idle before being asked to close.

Ability to dynamically increase the amount of memory available to Memcached without having to restart the cluster. For more information, see Memcached 1.4.27 Release Notes.
• Logging of fetchers, mutations, and evictions are now supported. For more information, see Memcached 1.4.26 Release Notes.
• Freed memory can be reclaimed back into a global pool and reassigned to new slab classes. For more information, see Memcached 1.4.25 Release Notes.
• Several bug fixes.
• Some new commands and parameters. For a list, see Memcached 1.4.33 Added Parameters (p. 353).

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. 355).
• 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.
ElastiCache for Redis Versions

If you enable at-rest, in-transit encryption, and Redis AUTH when you create a Redis cluster using ElastiCache for Redis version 3.2.6, you can use Amazon ElastiCache for Redis to build HIPAA-compliant applications. You can store healthcare-related information, including protected health information (PHI), under an executed Business Associate Agreement (BAA) with AWS. AWS Services in Scope have been fully assessed by a third-party auditor and result in a certification, attestation of compliance, or Authority to Operate (ATO). For more information, see the following topics:

- AWS Cloud Compliance
- HIPAA Compliance
- AWS Services in Scope by Compliance Program
- Amazon ElastiCache for Redis Data Encryption (p. 429)
- Authenticating Users with AUTH (Redis) (p. 427)

Supported ElastiCache for Redis versions

- ElastiCache for Redis Version 3.2.10 (Enhanced) (p. 50)
- ElastiCache for Redis Version 3.2.6 (Enhanced) (p. 50)
- ElastiCache for Redis Version 3.2.4 (Enhanced) (p. 51)
- ElastiCache for Redis Version 2.8.24 (Enhanced) (p. 52)
- ElastiCache for Redis Version 2.8.23 (Enhanced) (p. 52)
- ElastiCache for Redis Version 2.8.22 (Enhanced) (p. 52)
- ElastiCache for Redis Version 2.8.21 (p. 53)
- ElastiCache for Redis Version 2.8.19 (p. 53)
- ElastiCache for Redis Version 2.8.6 (p. 53)
- ElastiCache for Redis Version 2.6.13 (p. 53)

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 when using the ElastiCache console. We recommend against using these Redis versions. If you need to use one of them, work with the AWS CLI or ElastiCache API.

For more information, see the following topics:

<table>
<thead>
<tr>
<th></th>
<th>AWS CLI</th>
<th>ElastiCache API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Cluster</td>
<td>Creating a Cache Cluster (AWS CLI) (p. 168)</td>
<td>Creating a Cache Cluster (ElastiCache API) (p. 170)</td>
</tr>
<tr>
<td></td>
<td>This action cannot be used to create a replication group with cluster mode enabled.</td>
<td>This action cannot be used to create a replication group with cluster mode enabled.</td>
</tr>
<tr>
<td>Modify Cluster</td>
<td>Modifying a Cache Cluster (AWS CLI) (p. 180)</td>
<td>Modifying a Cache Cluster (ElastiCache API) (p. 181)</td>
</tr>
<tr>
<td></td>
<td>This action cannot be used to create a replication group with cluster mode enabled.</td>
<td>This action cannot be used to create a replication group with cluster mode enabled.</td>
</tr>
</tbody>
</table>
### ElastiCache for Redis Version 3.2.10 (Enhanced)

Amazon ElastiCache for Redis introduces the next major version of the Redis engine supported by Amazon ElastiCache. ElastiCache for Redis 3.2.10 introduces online cluster resizing to add or remove shards from the cluster while it continues to serve incoming I/O requests. ElastiCache for Redis 3.2.10 users have all the functionality of earlier Redis versions except the ability to encrypt their data which is currently only available in version 3.2.6.

#### Comparing ElastiCache for Redis versions 3.2.6 and 3.2.10

<table>
<thead>
<tr>
<th>Version</th>
<th>AWS CLI</th>
<th>ElastiCache API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Online cluster resizing *</td>
<td>Creating a Redis (cluster mode disabled) Cluster with Replicas from Scratch (AWS CLI) (p. 261)</td>
<td>Creating a Redis (cluster mode disabled) Cluster with Replicas from Scratch (ElastiCache API) (p. 264)</td>
</tr>
<tr>
<td>In-transit encryption</td>
<td>Creating a Redis (cluster mode enabled) Cluster with Replicas from Scratch (AWS CLI) (p. 267)</td>
<td>Creating a Redis (cluster mode enabled) Cluster with Replicas from Scratch (ElastiCache API) (p. 271)</td>
</tr>
<tr>
<td>At rest encryption</td>
<td>Modifying a Replication Group (AWS CLI) (p. 284)</td>
<td>Modifying a Replication Group (ElastiCache API) (p. 285)</td>
</tr>
<tr>
<td></td>
<td>This action cannot be used to create a replication group with cluster mode enabled.</td>
<td>This action cannot be used to create a replication group with cluster mode enabled.</td>
</tr>
</tbody>
</table>

* Adding, removing, and rebalancing shards.

Required for HIPAA compliant applications.

For more information, see:
- Online Resharding and Shard Rebalancing for ElastiCache for Redis—Redis (cluster mode enabled) (p. 226)
- Best Practices: Online Resharding (p. 89)

### ElastiCache for Redis Version 3.2.6 (Enhanced)

Amazon ElastiCache for Redis introduces the next major version of the Redis engine supported by Amazon ElastiCache. ElastiCache for Redis 3.2.6 users have all the functionality of earlier Redis versions plus the option to encrypt their data. For more information, see:

- Amazon ElastiCache for Redis In-Transit Encryption (p. 429)
ElastiCache for Redis Version 3.2.4 (Enhanced)

Amazon ElastiCache for Redis version 3.2.4 introduces the next major version of the Redis engine supported by Amazon ElastiCache. ElastiCache for 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 Non-Cluster Mode and Cluster Mode

<table>
<thead>
<tr>
<th>Feature</th>
<th>Non-Cluster Mode</th>
<th>Cluster Mode</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data partitioning</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Geospatial indexing</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Change node type</td>
<td>Yes</td>
<td>Yes *</td>
</tr>
<tr>
<td>Replica scaling</td>
<td>Yes</td>
<td>Yes *</td>
</tr>
<tr>
<td>Scale out</td>
<td>No</td>
<td>Yes *</td>
</tr>
<tr>
<td>Database support</td>
<td>Multiple</td>
<td>Single</td>
</tr>
<tr>
<td>Parameter group</td>
<td>default.redis3.2</td>
<td>default.redis3.2.cluster.on*</td>
</tr>
</tbody>
</table>

* See Restoring From a Backup with Optional Cluster Resizing (p. 317)

** Or one derived from it.

**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 3.2.4 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. 363).

ElastiCache for 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.

ElastiCache for 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. 366) in the ElastiCache User Guide.

ElastiCache for 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. 253). 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 because replicas 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. 449).

- 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 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 needs to reconnect. Such situations are most likely to occur for PUBSUB clients.

ElastiCache for 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.

ElastiCache for 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 now has 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. 449).

ElastiCache for 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.

ElastiCache for 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 don't occur automatically. You must initiate them.

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. 179) or Modifying a Cluster with Replicas (p. 284).

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. You can seed this new cluster using a Redis (cluster mode disabled) snapshot if 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. 77). Please note that in these cases, the primary is unavailable to service requests during the upgrade process.
- For Redis clusters with Multi-AZ enabled, in addition to the preceding, we also recommend that you schedule engine upgrades during periods of low incoming write traffic. The primary continues 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.
### 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.

<table>
<thead>
<tr>
<th>Clusters</th>
<th>Replication Groups</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Using the console</strong></td>
<td></td>
</tr>
<tr>
<td>Modifying a Cluster (Console)</td>
<td>Modifying a Redis Cluster (Console)</td>
</tr>
<tr>
<td>(p. 179)</td>
<td>(p. 284)</td>
</tr>
<tr>
<td><strong>Using the AWS CLI</strong></td>
<td></td>
</tr>
<tr>
<td>Modifying a Cache Cluster (AWS</td>
<td>Modifying a Replication Group (AWS CLI)</td>
</tr>
<tr>
<td>CLI) (p. 180)</td>
<td>(p. 284)</td>
</tr>
<tr>
<td><strong>Using the ElastiCache API</strong></td>
<td></td>
</tr>
<tr>
<td>Modifying a Cache Cluster (ElastiCache API) (p. 181)</td>
<td>Modifying a Replication Group (ElastiCache API) (p. 285)</td>
</tr>
</tbody>
</table>

### 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 chosen (CLI use: `--apply-immediately`, API use: `ApplyImmediately=true`). (This effectively cancels the engine upgrade during the next maintenance window by performing it immediately.)

### Pending Operations vs. Blocked Operations

<table>
<thead>
<tr>
<th>Pending Operations</th>
<th>Blocked Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale up</td>
<td>Immediate engine upgrade</td>
</tr>
<tr>
<td>Engine upgrade</td>
<td>Immediate scale up</td>
</tr>
<tr>
<td>Scale up and engine upgrade</td>
<td>Immediate scale up</td>
</tr>
<tr>
<td>Scale up and engine upgrade</td>
<td>Immediate engine upgrade</td>
</tr>
</tbody>
</table>
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 chosen day of the week.

The 60-minute maintenance window is chosen at random from an 8-hour block of time per region. The following table lists the time blocks for each region from which the default maintenance windows are assigned. You may choose a preferred maintenance window outside the region's maintenance window block.

<table>
<thead>
<tr>
<th>Region Code</th>
<th>Region Name</th>
<th>Region Maintenance Window</th>
</tr>
</thead>
<tbody>
<tr>
<td>ap-northeast-1</td>
<td>Asia Pacific (Tokyo) Region</td>
<td>13:00–21:00 UTC</td>
</tr>
<tr>
<td>ap-south-1</td>
<td>Asia Pacific (Mumbai) Region</td>
<td>17:30–1:30 UTC</td>
</tr>
<tr>
<td>ap-southeast-1</td>
<td>Asia Pacific (Singapore) Region</td>
<td>14:00–22:00 UTC</td>
</tr>
<tr>
<td>ap-southeast-2</td>
<td>Asia Pacific (Sydney) Region</td>
<td>12:00–20:00 UTC</td>
</tr>
<tr>
<td>cn-north-1</td>
<td>China (Beijing) region</td>
<td>14:00–22:00 UTC</td>
</tr>
<tr>
<td>eu-central-1</td>
<td>EU (Frankfurt) Region</td>
<td>23:00–07:00 UTC</td>
</tr>
<tr>
<td>eu-west-1</td>
<td>EU (Ireland) Region</td>
<td>22:00–06:00 UTC</td>
</tr>
<tr>
<td>sa-east-1</td>
<td>South America (São Paulo) Region</td>
<td>01:00–09:00 UTC</td>
</tr>
<tr>
<td>us-east-1</td>
<td>US East (N. Virginia) Region</td>
<td>03:00–11:00 UTC</td>
</tr>
<tr>
<td>us-east-2</td>
<td>US East (Ohio) Region</td>
<td>04:00–12:00 UTC</td>
</tr>
<tr>
<td>us-gov-west-1</td>
<td>AWS GovCloud (US) region</td>
<td>06:00–14:00 UTC</td>
</tr>
<tr>
<td>us-west-1</td>
<td>US West (N. California) Region</td>
<td>06:00–14:00 UTC</td>
</tr>
<tr>
<td>us-west-2</td>
<td>US West (Oregon) Region</td>
<td>06:00–14:00 UTC</td>
</tr>
</tbody>
</table>
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 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. 179) or Modifying a Cluster with Replicas (p. 284).
Choosing Regions and Availability Zones

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

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

**Important**

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

To create or work with a cluster in a specific region, use the corresponding regional service endpoint. For service endpoints, see Supported Regions & Endpoints (p. 60).

Regions and Availability Zones

**Topics**

- Locating Your Nodes (p. 59)
- Supported Regions & Endpoints (p. 60)
Locating Your Nodes

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

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

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

- Creating a Cluster (p. 156)
- Creating a Redis Cluster with Replicas (p. 254)
- Modifying an ElastiCache Cluster (p. 179)
- Adding Nodes to a Cluster (p. 184)
- Adding a Read Replica to a Redis Cluster (p. 287)
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 Choosing Regions and Availability Zones (p. 58) at the top of this topic.

Regions where ElastiCache is supported

<table>
<thead>
<tr>
<th>Region Name/Region</th>
<th>Endpoint</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Ohio) Region</td>
<td>elasticache.us-east-2.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>us-east-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US East (N. Virginia) Region</td>
<td>elasticache.us-east-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>us-east-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US West (N. California) Region</td>
<td>elasticache.us-west-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>us-west-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>US West (Oregon) Region</td>
<td>elasticache.us-west-2.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>us-west-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Canada (Central) Region</td>
<td>elasticache.ca-central-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>ca-central-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asia Pacific (Mumbai) Region</td>
<td>elasticache.ap-south-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>ap-south-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asia Pacific (Seoul) Region</td>
<td>elasticache.ap-northeast-2.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>ap-northeast-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asia Pacific (Singapore) Region</td>
<td>elasticache.ap-southeast-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>ap-southeast-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asia Pacific (Sydney) Region</td>
<td>elasticache.ap-southeast-2.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>ap-southeast-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Region Name/Region</td>
<td>Endpoint</td>
<td>Protocol</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------</td>
<td>--------------</td>
</tr>
<tr>
<td>ap-southeast-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asia Pacific (Tokyo) Region</td>
<td>elasticache.ap-northeast-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>ap-northeast-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EU (Frankfurt) Region</td>
<td>elasticache.eu-central-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>eu-central-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EU (Ireland) Region</td>
<td>elasticache.eu-west-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>eu-west-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EU (London) Region</td>
<td>elasticache.eu-west-2.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>eu-west-2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EU (Paris) Region</td>
<td>elasticache.eu-west-3.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>eu-west-3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>South America (São Paulo) Region</td>
<td>elasticache.sa-east-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>sa-east-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>China (Beijing) Region</td>
<td>elasticache.cn-north-1.amazonaws.com.cn</td>
<td>HTTPS</td>
</tr>
<tr>
<td>cn-north-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>China (Ningxia) Region</td>
<td>elasticache.cn-northwest-1.amazonaws.com.cn</td>
<td>HTTPS</td>
</tr>
<tr>
<td>cn-northwest-1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>AWS GovCloud (US)</td>
<td>elasticache.us-gov-west-1.amazonaws.com</td>
<td>HTTPS</td>
</tr>
<tr>
<td>us-gov-west-1</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

For information on using the AWS GovCloud (US) with ElastiCache, see Services in the AWS GovCloud (US) region: ElastiCache.

Some regions support a subset of node types. For a table of supported node types by region, see Supported Node Types by Region (p. 118).

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.

Topics

- Finding a Memcached Cluster's Endpoints (Console) (p. 63)
- Finding a Redis (cluster mode disabled) Cluster's Endpoints (Console) (p. 65)
- Finding a Redis (cluster mode enabled) Cluster's Endpoints (Console) (p. 67)
- Finding Endpoints (AWS CLI) (p. 69)
- Finding Endpoints (ElastiCache API) (p. 73)
Finding a Memcached Cluster's Endpoints (Console)

All Memcached endpoints are read/write endpoints. To connect to nodes in a Memcached cluster your application can use either the endpoints for each node, or the cluster's configuration endpoint along with Automatic Discovery. To use Automatic Discovery you must use a client that supports Automatic Discovery.

When using Automatic Discovery, your client application connects to your Memcached cluster using the configuration endpoint. As you scale your cluster by adding or removing nodes, your application will automatically "know" all the nodes in the cluster and be able to connect to any of them. Without Automatic Discovery your application would have to do this, or you'd have to manually update endpoints in your application each time you added or removed a node. For additional information on Automatic Discovery, see Node Auto Discovery (Memcached) (p. 123).

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)

2. From the navigation pane, choose Memcached.
   The cache clusters screen will appear with a list of Memcached clusters.
3. Find the Memcached cluster you want the endpoints for.
   If all you want is the configuration endpoint, you're done. The configuration endpoint is in the Configuration Endpoint column and looks something like this, `clusterName.xxxxxx.cfg.usw2.cache.amazonaws.com:port`.
   If you want to also see the individual node endpoints or copy any of the endpoints to your clipboard, choose Copy Node Endpoint.

<table>
<thead>
<tr>
<th>Configuration Endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>clusterName.xxxxxx.cfg.usw2.cache.amazonaws.com:port</td>
</tr>
</tbody>
</table>

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

The highlighted endpoint is now copied to your clipboard.
Configuration and node endpoints look very similar. The differences are highlighted with **bold** following.

<table>
<thead>
<tr>
<th>Endpoint</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>myclustername.xxxxxx.cfg.usw2.cache.amazonaws.com:port</code></td>
<td>configuration endpoint contains “cfg”</td>
</tr>
<tr>
<td><code>myclustername.xxxxxx.0001.usw2.cache.amazonaws.com:port</code></td>
<td>node endpoint for node 0001</td>
</tr>
</tbody>
</table>

**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 a Redis (cluster mode disabled) Cluster's Endpoints (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

2. From the 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. If the Redis (cluster mode disabled) cluster has replica nodes, you can find the cluster's replica node endpoints by choosing the cluster's name.

   The nodes screen appears with each node in the cluster, primary and replicas, listed with its endpoint.

   Node endpoints for a Redis (cluster mode disabled) cluster

5. To copy an endpoint to your clipboard:
   
   a. One endpoint at a time, find then highlight the endpoint you want to copy.
   b. Right-click the highlighted endpoint, then choose Copy from the context menu.

   The highlighted endpoint is now copied to your clipboard.
A Redis (cluster mode disabled) primary endpoint looks something like the following. There is a difference depending upon whether or not In-Transit encryption is enabled.

**In-transit encryption not enabled**

```
clusterName.xxxxxx.nodeId.regionAndAz.cache.amazonaws.com:port
redis-01.7abc2d.0001.usw2.cache.amazonaws.com:6379
```

**In-transit encryption enabled**

```
master.clusterName.xxxxxx.regionAndAz.cache.amazonaws.com:port
master.ncit.ameaqx.usel.cache.amazonaws.com:6379
```
Finding a Redis (cluster mode enabled) Cluster's Endpoints (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**

2. From the navigation pane, choose *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**

2. From the navigation pane, choose *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 preceding procedures.
2. Highlight the endpoint that you want to copy.
3. Right-click the highlighted endpoint and choose Copy from the context menu.

   The highlighted endpoint is now copied to your clipboard.

A Redis (cluster mode enabled) configuration endpoint looks something like the following.

In-transit encryption not enabled

<table>
<thead>
<tr>
<th>endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>clusterName.xxxxxx.regionAndAz.cache.amazonaws.com:port</td>
</tr>
<tr>
<td>rce.ameaqx.use1.cache.amazonaws.com:6379</td>
</tr>
</tbody>
</table>

In-transit encryption enabled

<table>
<thead>
<tr>
<th>endpoint</th>
</tr>
</thead>
<tbody>
<tr>
<td>clustercfg.clusterName.xxxxxx.regionAndAz.cache.amazonaws.com:port</td>
</tr>
<tr>
<td>clustercfg.rce.ameaqx.use1.cache.amazonaws.com:6379</td>
</tr>
</tbody>
</table>
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) (p. 69)
- Finding the Endpoints for Replication Groups (AWS CLI) (p. 70)

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, macOS, or Unix:

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

For Windows:

```bash
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, macOS, or Unix:

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

For Windows:

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

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

```json
{
"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": [
      {
        "ReplicationGroupId": "myreplgroup",
        "MemberEnabled": "true",
        "MemberStatus": "available",
        "MemberId": "myreplgroup-001",
        "CacheClusterId": "myreplgroup-001"
      }
    ]
  }
}
```

API Version 2015-02-02
'"myreplgroup-001",
"myreplgroup-002",
"myreplgroup-003"
],
"PendingModifiedValues": {}  
}
}

For more information, see `describe-replication-groups` in the *AWS CLI Command 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) (p. 73)
- Finding Endpoints for Replication Groups (ElastiCache API) (p. 73)

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
```

API Version 2015-02-02
For more information, see `DescribeReplicationGroups`. 
ElastiCache Notifications

This topic covers ElastiCache notifications that you might be interested in. A notification is a situation or event that, in most cases, is temporary, lasting only until a solution is found and implemented. Notifications generally have a start date and a resolution date, after which the notification is no longer relevant. Any one notification might or might not be relevant to you. We recommend an implementation guideline that, if followed, improves the performance of your cluster.

Notifications do not announce new or improved ElastiCache features or functionality.

Alert: Memcached LRU Crawler Causing Segmentation Faults

Alert Date: February 28, 2017
In some circumstances, your cluster might display instability with a segmentation fault in the Memcached LRU Crawler. This is an issue within the Memcached engine that has existed for some time. The issue became apparent in Memcached 1.4.33 when the LRU Crawler was enabled by default.
If you are experiencing this issue, we recommend that you disable the LRU Crawler until there is a fix. To do so, use lru_crawler disable at the command line or modify the lru_crawler parameter value (preferred).
Resolved Date:
Resolution:
Best Practices for Amazon ElastiCache

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

Topics

- Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 77)
- Managing Reserved Memory (Redis) (p. 79)
- Mitigating Out-of-Disk-Space Issues When Using Redis AOF (p. 83)
- Mitigating Failures (p. 83)
- Configuring Your ElastiCache Client for Efficient Load Balancing (p. 88)
- Best Practices: Online Resharding (p. 89)
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. 253).

Redis snapshots and synchronizations before 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 causes the process to fail. Because of this, it is important to choose 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. 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 should 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.
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-percent` to 50 (50 percent) for Redis versions before 2.8.22 or 25 (25 percent) for Redis versions 2.8.22 and later. For more information, see Managing Reserved Memory (Redis) (p. 79).

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. 377).

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. 362).

For information on creating and modifying parameter groups, see Creating a Parameter Group (p. 340) and Modifying a Parameter Group (p. 348).
Managing Reserved Memory (Redis)

Reserved memory is memory set aside for non-data use. When performing a backup or failover, Redis uses available memory to record write operations to your cluster while the cluster’s data is being written to the .rdb file. If you don’t have sufficient memory available for all the writes, the process fails. Following, you can find information on options for managing reserved memory for ElastiCache for Redis and how to apply those options.

How Much Reserved Memory Do You Need?

If you are running a version of Redis prior to 2.8.22, you need to reserve more memory for backups and failovers than if you are running Redis 2.8.22 or later. This requirement is due to the different ways that ElastiCache for Redis implements the backup process. The rule of thumb is to reserve half of a node type's `maxmemory` value for Redis overhead for versions prior to 2.8.22, and one-fourth for Redis versions 2.8.22 and later. For more information, see Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 77) and How Synchronization and Backup are Implemented (p. 253).

Parameters to Manage Reserved Memory

The `reserved-memory` Parameter

Prior to March 16, 2017, all ElastiCache for Redis reserved memory management was done using the parameter `reserved-memory`. The default value of `reserved-memory` is 0. This default reserves no memory for Redis overhead and allows Redis to consume all of a node's memory with data. Changing `reserved-memory` so you have sufficient memory available for backups and failovers requires you to create a custom parameter group. In this custom parameter group, you set `reserved-memory` to a value appropriate for the Redis version running on your cluster and cluster’s node type.

The ElastiCache for Redis parameter `reserved-memory` is specific to ElastiCache for Redis and isn't part of the Redis distribution.

The following procedure shows how to use `reserved-memory` to manage the memory on your Redis cluster.

To reserve memory using `reserved-memory`

1. Create a custom parameter group specifying the parameter group family matching the engine version you're running—for example, specifying the `redis2.8` parameter group family.

```bash
aws elasticache create-cache-parameter-group \
--cache-parameter-group-name redis28-m3xl \
--cache-parameter-group-family redis2.8
```

2. Calculate how many bytes of memory you need to reserve for Redis overhead. You can find the value of `maxmemory` for your node type at Redis Node-Type Specific Parameters (p. 377).

3. Modify the custom parameter group so that the parameter `reserved-memory` is the number of bytes you calculated in the previous step. The following AWS CLI example assumes you’re running a version of Redis prior to 2.8.22 and need to reserve half of the node's `maxmemory`.

```bash
aws elasticache modify-cache-parameter-group \
--cache-parameter-group-name redis28-m3xl \
--parameter-name-values "ParameterName=reserved-memory, ParameterValue=7130316800"
```

Note that you need a separate custom parameter group for each node type that you use, because each node type has a different `maxmemory` value. Thus, each node type needs a different value for `reserved-memory`. 
4. Apply the custom parameter group to your cluster.

The following CLI example applies the `redis28-m3xl` parameter group to the cluster `my-redis-cluster`.

```
aws elasticache modify-cache-cluster \
  --cache-cluster-id my-redis-cluster \n  --cache-parameter-group-name redis28-m3xl \n  --apply-immediately
```

The following CLI example applies the `redis28-m3xl` parameter group to the replication group (in the console, the cluster) `my-redis-repl-grp`.

```
aws elasticache modify-replication-group \
  --replication-group-id my-redis-repl-grp \n  --cache-parameter-group-name redis28-m3xl \n  --apply-immediately
```

For more information, see Modifying an ElastiCache Cluster (p. 179) or Modifying a Cluster with Replicas (p. 284).

The reserved-memory-percent parameter

On March 16, 2017, Amazon ElastiCache introduced the parameter `reserved-memory-percent` and made it available on all versions of ElastiCache for Redis. The purpose of `reserved-memory-percent` is to simplify reserved memory management across all your clusters. It does so by enabling you to have a single parameter group for each parameter group family (such as `redis2.8`) to manage your clusters' reserved memory, regardless of node type. The default value for `reserved-memory-percent` is 25 (25 percent).

The ElastiCache for Redis parameter `reserved-memory-percent`, like `reserved-memory`, is specific to ElastiCache for Redis and isn't part of the Redis distribution.

To use `reserved-memory-percent` to manage the memory on your ElastiCache for Redis cluster, do one of the following:

- If you are running Redis 2.8.22 or later, just assign the default parameter group to your cluster. The default 25 percent should be adequate. If not, you can follow the steps in the next bullet to change the value.

- If you are running a version of Redis prior to 2.8.22, you likely will need to reserve more memory than `reserved-memory-percent`'s default 25 percent. To do so, use the following procedure.

To reserve memory using `reserved-memory-percent`

1. Create a custom parameter group specifying the parameter group family matching the engine version you're running—for example, specifying the `redis2.8` parameter group family. A custom parameter group is necessary because you cannot modify a default parameter group.

```
aws elasticache create-cache-parameter-group \
  --cache-parameter-group-name redis28-50 \
  --cache-parameter-group-family redis2.8
```
Because `reserved-memory-percent` reserves memory as a percent of a node's `maxmemory`, you don't need a custom parameter group for each node type.

2. Modify the custom parameter group so that `reserved-memory-percent` is 50 (50 percent).

   ```bash
   aws elasticache modify-cache-parameter-group
   --cache-parameter-group-name redis28-50
   --parameter-name-values "ParameterName=reserved-memory-percent, ParameterValue=50"
   ```

3. Use this custom parameter group for any Redis clusters running a version of Redis older than 2.8.22.

   The following CLI example applies the `redis28` parameter group to the cluster `my-redis-cluster`.

   ```bash
   aws elasticache modify-cache-cluster
   --cache-cluster-id my-redis-cluster
   --cache-parameter-group-name redis28-50
   --apply-immediately
   ```

   The following CLI example applies the `redis28-50` parameter group to the replication group (in the console, the cluster) `my-redis-repl-grp`.

   ```bash
   aws elasticache modify-replication-group
   --replication-group-id my-redis-repl-grp
   --cache-parameter-group-name redis28-50
   --apply-immediately
   ```

For more information, see Modifying an ElastiCache Cluster (p. 179) or Modifying a Cluster with Replicas (p. 284).

### Changing Between the reserved-memory and reserved-memory-percent Parameters

If you were a current ElastiCache customer on March 16, 2017, your default reserved memory management parameter is `reserved-memory`. If you became an ElastiCache customer after March 16, 2017, your default reserved memory management parameter is `reserved-memory-percent`. You can change your reserved memory management parameter if you want using either the AWS CLI or ElastiCache API.

The parameters `reserved-memory` and `reserved-memory-percent` are mutually exclusive. A parameter group will always have one but never both. You can change which parameter a parameter group uses for reserved memory management by modifying the parameter group. The parameter group must be a custom one, because you cannot modify default parameter groups.

The following CLI example modifies the custom parameter group `redis32-cluster-on` so that it uses `reserved-memory-percent` to manage reserved memory. Because the engine version is newer than 2.8.22, it sets the value of `reserved-memory-percent` to 25 (25 percent) even though that is the default. It does so because `reserved-memory-percent` must be assigned some value to convert the parameter group from `reserved-memory` to `reserved-memory-percent`.

```bash
aws elasticache modify-cache-parameter-group
--cache-parameter-group-name redis32-cluster-on
--parameter-name-values "ParameterName=reserved-memory-percent, ParameterValue=25"
```
The following CLI example modifies the custom parameter group `redis32-m3xl` so that it uses `reserved-memory` to manage reserved memory. Because the engine version is newer than 2.8.22, it sets the value to 3565158400 which is one-fourth of a `cache.m3.xlarge`’s maxmemory.

```
aws elasticache modify-cache-parameter-group \
    --cache-parameter-group-name redis32-m3xl \
    --parameter-name-values "ParameterName=reserved-memory, ParameterValue=3565158400"
```
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 replays 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 provisions a new node on a different server. In this case, the AOF file is no longer available and cannot be used to recover the data. Thus, Redis restarts 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. 83)
- ElastiCache Replication (Redis) (p. 235)
- Replication: Multi-AZ with Automatic Failover (Redis) (p. 240)

Mitigating Failures

When planning your Amazon ElastiCache implementation, you should plan so that failures have a minimal impact upon your application and data. The topics in this section cover approaches you can take to protect your application and data from failures.

**Topics**

- Mitigating Failures when Running Memcached (p. 83)
- Mitigating Failures when Running Redis (p. 84)
- Recommendations (p. 86)

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. 157).

Mitigating Availability Zone Failures

To mitigate the impact of an Availability Zone failure, locate your nodes in as many Availability Zones as possible. In the unlikely event of an AZ failure, you will lose the data cached in that AZ, not the data cached in the other AZs.

Why so many nodes?

If my region has only 3 Availability Zones, why do I need more than 3 nodes since if an AZ fails I lose approximately one-third of my data?

This is an excellent question. Remember that we're attempting to mitigate two distinct types of failures, node and Availability Zone. You're right, if your data is spread across Availability Zones and one of the zones fails, you will lose only the data cached in that AZ, irrespective of the number of nodes you have. However, if a node fails, having more nodes will reduce the proportion of data lost.

There is no "magic formula" for determining how many nodes to have in your cluster. You must weight the impact of data loss vs. the likelihood of a failure vs. cost, and come to your own conclusion.

For information on specifying the number of nodes in a Memcached cluster, go to Creating a Cluster (Console): Memcached (p. 157).

For more information on regions and Availability Zones, go to Choosing Regions and Availability Zones (p. 58).

Mitigating Failures when Running Redis

When running the Redis engine, you have the following options for minimizing the impact of a node or Availability Zone failure.

Mitigating Node Failures

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

Topics

- Mitigating Failures: Redis Append Only Files (AOF) (p. 85)
- Mitigating Failures: Redis Replication Groups (p. 85)
Mitigating 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 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 is not available and cannot be used to recover the data.

For more information, see Redis Append Only Files (AOF) (p. 327).

Mitigating Failures: Redis Replication Groups

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

When a read replica fails

1. ElastiCache detects the failed read replica.
2. ElastiCache takes the failed node off line.
3. ElastiCache launches and provisions a replacement node in the same AZ.
4. The new node synchronizes with the Primary node.

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

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

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

For more information, see Replication: Multi-AZ with Automatic Failover (Redis) (p. 240).

**When Multi-AZ with Auto Failover is disabled**

1. ElastiCache detects Primary failure.
2. ElastiCache takes the Primary offline.
3. ElastiCache creates and provisions a new Primary node to replace the failed Primary.
4. ElastiCache syncs the new Primary with one of the existing replicas.
5. When the sync is finished, the new node functions as the cluster's Primary node.

During this process, steps 1 through 4, your application cannot write to the Primary node. However, your application can continue reading from your replica nodes.

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

For more information, see ElastiCache Replication (Redis) (p. 235).

**Mitigating Availability Zone Failures**

To mitigate the impact of an Availability Zone failure, locate your nodes in as many Availability Zones as possible.

No matter how many nodes 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 nodes in multiple AZs, a failure of any AZ results in your losing only the nodes in that AZ.

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

For information on specifying the Availability Zones for Redis nodes, go to Creating a Redis (cluster mode disabled) Cluster (Console) (p. 159).

For more information on regions and Availability Zones, go to Choosing Regions and Availability Zones (p. 58).

**Recommendations**

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

**Minimizing the Impact of Failures**

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

**Minimizing the Impact of Availability Zone Failures**

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

**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 and Restore (Redis) (p. 293).
Configuring Your ElastiCache Client for Efficient Load Balancing

Note
This section applies to multi-node Memcached clusters.

To effectively use multiple ElastiCache Memcached nodes, you need to be able to spread your cache keys across the nodes. A simple way to load balance a cluster with \( n \) nodes is to calculate the hash of the object’s key and mod the result by \( n - \text{hash(key)} \mod n \). The resulting value (0 through \( n-1 \)) is the number of the node where you place the object.

This approach is simple and works well as long as the number of nodes (\( n \)) is constant. However, whenever you add or remove a node from the cluster, the number of keys that need to be moved is \( (n - 1) / n \) (where \( n \) is the new number of nodes). Thus, this approach results in a large number of keys being moved, which translates to a large number of initial cache misses, especially as the number of nodes gets large. Scaling from 1 to 2 nodes results in \( (2-1)/2 \) (50 percent) of the keys being moved, the best case. Scaling from 9 to 10 nodes results in \( (10-1)/10 \) (90 percent) of the keys being moved. If you're scaling up due to a spike in traffic, you don't want to have a large number of cache misses. A large number of cache misses results in hits to the database, which is already overloaded due to the spike in traffic.

The solution to this dilemma is consistent hashing. Consistent hashing uses an algorithm such that whenever a node is added or removed from a cluster, the number of keys that must be moved is roughly \( 1 / n \) (where \( n \) is the new number of nodes). Scaling from 1 to 2 nodes results in \( 1/2 \) (50 percent) of the keys being moved, the worst case. Scaling from 9 to 10 nodes results in \( 1/10 \) (10 percent) of the keys being moved.

As the user, you control which hashing algorithm is used for multi-node clusters. We recommend that you configure your clients to use consistent hashing. Fortunately, there are many Memcached client libraries in most popular languages that implement consistent hashing. Check the documentation for the library you are using to see if it supports consistent hashing and how to implement it.

If you are working in Java, PHP, or .NET, we recommend you use one of the Amazon ElastiCache client libraries.

Consistent Hashing Using Java

The ElastiCache Memcached Java client is based on the open-source spymemcached Java client, which has consistent hashing capabilities built in. The library includes a KetamaConnectionFactory class that implements consistent hashing. By default, consistent hashing is turned off in spymemcached.

For more information, 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.

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

In addition to the preceding code, we recommend that you also turn `memcached.sess_consistent_hash` on in your php.ini file.
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.

Best Practices: Online Resharding

`Resharding` involves adding and removing shards to your cluster and redistributing slots across shards. As a result, multiple things have an impact on the resharding operation, such as the load on the cluster, memory utilization, and overall size of data. For the best experience, we recommend that you follow overall cluster best practices for uniform workload pattern distribution. In addition, we recommend taking the following steps.

Before initiating resharding, we recommend the following:

- **Test your application** – Test your application behavior during resharding in a staging environment if possible.
- **Get early notification for scaling issues** – Because resharding is a compute intensive operation, we recommend keeping CPU utilization under 80 percent on multicore instances and less than 50 percent on single core instances during resharding. Monitor ElastiCache for Redis metrics and initiate resharding before your application starts observing scaling issues. Useful metrics to track are `CPUUtilization`, `NetworkBytesIn`, `NetworkBytesOut`, `CurrConnections`, `NewConnections`, `FreeableMemory`, `SwapUsage`, and `BytesUsedForCache`.
- **Ensure sufficient free memory is available before scaling in** – If you're scaling in, ensure that free memory available on the shards to be retained is at least 1.5 times the memory used on the shards you plan to remove.
- **Initiate resharding during off-peak hours** – This practice helps to reduce the latency and throughput impact on the client during the resharding operation. It also helps to complete resharding faster as more resources can be used for slot redistribution.
- **Review client timeout behavior** – Some clients might observe higher latency during online resharding. Configuring your client library with a higher timeout can help by giving the system time to connect even under higher load conditions on server. If you open a large number of connections to the server, consider adding exponential backoff to reconnect logic to prevent a burst of new connections hitting the server at the same time.

During resharding, we recommend the following:

- **Avoid expensive commands** – Avoid running any computationally and I/O intensive operations, such as the `KEYS` and `SMEMBERS` commands. We suggest this approach because these operations increase the load on the cluster and have an impact on the performance of the cluster. Instead, use the `SCAN` and `SSCAN` commands.
- **Follow Lua best practices** – Avoid long running Lua scripts, and always declare keys used in Lua scripts up front. We recommend this approach to determine that the Lua script is not using cross slot commands. Ensure that the keys used in Lua scripts belong to the same slot.

After resharding, note the following:
- Scale-in might be partially successful if insufficient memory is available on target shards. If such a result occurs, review available memory and retry the operation, if necessary.
- Slots with large items are not migrated. In particular, slots with items larger than 256 MB post-serialization are not migrated.
- The BRPOPLPUSH command is not supported if it operates on the slot being migrated. FLUSHALL and FLUSHDB commands are not supported inside Lua scripts during a resharding operation.
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.

- Cluster node quota exceeded (p. 91)
- Customer's node quota exceeded (p. 91)
- Manual snapshot quota exceeded (p. 91)

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 node that has the same data, such as a different node in a cluster.
Caching Strategies

This topic covers strategies for populating and maintaining your cache.

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

Topics
- Lazy Loading (p. 92)
- Write Through (p. 94)
- Adding TTL (p. 95)
- Related Topics (p. 96)

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.
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. 94) and Adding TTL (p. 95) strategies.

Lazy Loading Code

The following code is a pseudo code example of lazy loading logic.

```c
// *****************************************
// function that returns a customer's record.
```
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. 92) 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. 95) you can minimize wasted space.
Write Through Code

The following code is a pseudo code example of write through logic.

```c
// *****************************************
// 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:

```c
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.

```c
// *****************************************
// 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.

```c
// *****************************************
// 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. 41)
- Scaling (p. 199)
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 chosen 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 Choosing Your Node Size for Memcached Clusters (p. 99) and Choosing Your Node Size for Redis Clusters (p. 100) for a more detailed discussion of which node size to use.

Topics
- Redis Nodes and Shards (p. 97)
- Choosing Your Node Size (p. 99)
- Connecting to Nodes (p. 103)
- ElastiCache Reserved Nodes (p. 107)
- Supported Node Types (p. 117)
- Actions You Can Take When a Node is Scheduled for Replacement (p. 120)

Other ElastiCache Node Operations

Additional operations involving nodes:
- Adding Nodes to a Cluster (p. 184)
- Removing Nodes from a Cluster (p. 190)
- Scaling (p. 199)
- Finding Your ElastiCache Endpoints (p. 62)
- Node Auto Discovery (Memcached) (p. 123)

Redis Nodes and Shards

A shard (API/CLI: node group) is a hierarchical arrangement of nodes (each wrapped in a 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. 235), Adding a Read Replica to a Redis Cluster (p. 287) or Deleting a Read Replica (p. 292). Redis (cluster mode disabled) clusters can also scale vertically by changing node types. For more information, see Scaling Redis (cluster mode disabled) Clusters with Replica Nodes (p. 213).

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 Making Manual Backups (p. 297) and Restoring From a Backup with Optional Cluster Resizing (p. 317).
Choosing 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 choose will make a difference in the node size you needed by your application.

Topics
- Choosing Your Node Size for Memcached Clusters (p. 99)
- Choosing Your Node Size for Redis Clusters (p. 100)

Choosing Your Node Size for Memcached Clusters

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 nodes in the cluster by the RAM capacity of each node. The capacity of each node is based on the node type.

The number of nodes in the cluster is a key factor in the availability of your cluster running Memcached. The failure of a single node can have an impact on the availability of your application and the load on your back-end database while ElastiCache provisions a replacement for the failed node and it gets repopulated. You can reduce this potential availability impact by spreading your memory and compute capacity over a larger number of nodes, each with smaller capacity, rather than using a fewer number of high capacity nodes.

In a scenario where you want to have 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

<table>
<thead>
<tr>
<th>Node type</th>
<th>Memory</th>
<th>Cores</th>
<th>Hourly Cost *</th>
<th>Nodes Needed</th>
<th>Total Memory</th>
<th>Total Cores</th>
<th>Monthly Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache.t2.medium</td>
<td>3.22 GB</td>
<td>2</td>
<td>$ 0.068</td>
<td>13</td>
<td>41.86 GB</td>
<td>26</td>
<td>$ 636.48</td>
</tr>
<tr>
<td>cache.m3.large</td>
<td>6.05 GB</td>
<td>2</td>
<td>$ 0.182</td>
<td>7</td>
<td>42.35 GB</td>
<td>14</td>
<td>$ 917.28</td>
</tr>
<tr>
<td>cache.m4.large</td>
<td>6.42 GB</td>
<td>2</td>
<td>$ 0.156</td>
<td>7</td>
<td>44.94 GB</td>
<td>14</td>
<td>$ 768.24</td>
</tr>
<tr>
<td>cache.r3.large</td>
<td>13.50 GB</td>
<td>2</td>
<td>$ 0.228</td>
<td>3</td>
<td>40.50 GB</td>
<td>6</td>
<td>$ 492.48</td>
</tr>
<tr>
<td>cache.m4.xlarge</td>
<td>14.28 GB</td>
<td>4</td>
<td>$ 0.311</td>
<td>3</td>
<td>42.84 GB</td>
<td>12</td>
<td>$ 671.76</td>
</tr>
</tbody>
</table>
Choosing Your Node Size (Redis)

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

For clusters running Memcached, some of the available memory on each node is used for connection overhead. For more information, see Memcached Connection Overhead (p. 359).

Using multiple nodes will require spreading the keys across them. Each node has its own endpoint. For easy endpoint management, you can use the ElastiCache the Auto Discovery feature, which enables client programs to automatically identify all of the nodes in a cluster. For more information, see Node Auto Discovery (Memcached) (p. 123).

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. 446). 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 node type that provides more compute power.

**Choosing Your Node Size for Redis Clusters**

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.

- **What version of Redis are you running?**

  Redis versions prior to 2.8.22 require you to reserve more memory for failover, snapshot, synchronizing, and promoting a replica to primary operations. This requirement occurs because you must have sufficient memory available for all writes that occur during the process.

  Redis version 2.8.22 and later use a forkless save process that requires less available memory than the earlier process.

  For more information, see the following:

<table>
<thead>
<tr>
<th>Node type</th>
<th>Memory</th>
<th>Cores</th>
<th>Hourly Cost *</th>
<th>Nodes Needed</th>
<th>Total Memory</th>
<th>Total Cores</th>
<th>Monthly Cost</th>
</tr>
</thead>
</table>

* Hourly cost per node as of August 4, 2016.

Monthly cost at 100% usage for 30 days (720 hours).
• How Synchronization and Backup are Implemented (p. 253)
• Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 77)

• How write-heavy is your application?

Write heavy applications can require significantly more available memory, memory not used by 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 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. 77).

• 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.rge** 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 \( \frac{\text{bytes-for-data-and-overhead}}{\text{number-of-shards}} \) 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 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%.
Connecting to Nodes

In order for you to use your cache, your application must connect to the nodes in the cluster. This section covers how to connect to nodes in Memcached and Redis clusters.

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. 32).

Topics
- Connecting to Memcached Nodes (p. 103)
- Connecting to Redis Nodes (p. 104)

Connecting to Memcached Nodes

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

- Finding a Memcached Cluster's Endpoints (Console) (p. 63)
- Finding Endpoints (AWS CLI) (p. 69)
- Finding Endpoints (ElastiCache API) (p. 73)

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.

   ```bash
   sudo yum install telnet
   ```

   Output similar to the following appears.

   ```
   Loaded plugins: priorities, security, update-motd, upgrade-helper
   Setting up Install Process
   Resolving Dependencies
   --> Running transaction check
   ...(output omitted)...
   Total download size: 63 k
   ```
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.

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

Output similar to the following appears.

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

4. Test the connection by running Memcached commands.

You are now connected to a node, and you can run Memcached commands. The following is an example.

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

## Connecting to Redis Nodes

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

- Finding a Redis (cluster mode disabled) Cluster's Endpoints (Console) (p. 65)
- Finding a Redis (cluster mode enabled) Cluster's Endpoints (Console) (p. 67)
- Finding Endpoints (AWS CLI) (p. 69)
- Finding Endpoints (ElastiCache API) (p. 73)

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 the [http://redis.io/commands](http://redis.io/commands) webpage.

### To connect to a Redis cluster using the `redis-cli`

1. Connect to your Amazon EC2 instance using the connection utility of your choice.
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.

```bash
sudo yum install gcc
```

Output similar to the following appears.

```
Loaded plugins: priorities, security, update-motd, upgrade-helper
Setting up Install Process
Resolving Dependencies
--> Running transaction check
...(output omitted)...
Total download size: 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)...
```

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`.
```

```bash
wget http://download.redis.io/redis-stable.tar.gz
tar xzvf redis-stable.tar.gz
cd redis-stable
make distclean      // ubuntu systems only
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.

```
Repeat this step for each node in your cluster that you want to connect to.
```

```bash
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. Test the connection by running Redis commands.

You are now connected to the cluster and can run Redis commands. The following are some example commands with their Redis responses.
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  // wait 5 seconds
"Good-bye"
get b  // key has expired, nothing returned
(nil)
quit  // Exit from redis-cli
ElastiCache Reserved Nodes

Reserving one or more nodes may be a way for you to reduce costs. Reserved nodes are charged an up front fee that depends upon the node type and the length of reservation – 1 or 3 years. In addition to the up front charge there is an hourly usage charge which is significantly less than the hourly usage charge you incur with On-Demand nodes. To determine whether reserved nodes would be a cost savings for your use cases, determine the node size and number of nodes you need, estimate the usage of the node, then compare the total cost to you using On-Demand nodes versus reserved nodes. You can mix and match reserved and On-Demand node usage in your clusters. For pricing information, see Amazon ElastiCache Pricing.

You can use the AWS Management Console, the AWS CLI, or the ElastiCache API to list and purchase available reserved node offerings.

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

Topics

• Understanding Utilization Levels (p. 107)
• Getting Info About Reserved Node Offerings (p. 109)
• Purchasing a Reserved Node (p. 112)
• Getting Info About Your Reserved Nodes (p. 115)

Understanding Utilization Levels

There are three levels of node reservations – Heavy Utilization, Medium Utilization, and Light Utilization. Nodes can be reserved at any utilization level for either 1 or 3 years. The node type, utilization level, and reservation term will impact your total costs. You should verify the savings that reserved nodes can provide your business by comparing various models before you purchase reserved nodes.

Nodes purchased at one utilization level or term cannot be converted to a different utilization level or term.

Utilization Levels

Heavy Utilization reserved nodes enable workloads that have a consistent baseline of capacity or run steady-state workloads. Heavy Utilization reserved nodes require a high up-front commitment, but if you plan to run more than 79 percent of the reserved node term you can earn the largest savings (up to 70 percent off of the On-Demand price). With Heavy Utilization reserved nodes you pay a one-time fee, followed by a lower hourly fee for the duration of the term regardless of whether or not your node is running.

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

Light Utilization reserved nodes are ideal for periodic workloads that run only a couple of hours a day or a few days per week. Using Light Utilization reserved nodes, you pay a one-time fee followed by a discounted hourly usage fee when your node is running. You can start saving when your node is running more than 17 percent of the reserved node term, and you can save up to 56 percent off of the On-Demand rates over the entire term of your reserved node.
Reserved Cache Node Offerings

<table>
<thead>
<tr>
<th>Offering</th>
<th>Up-Front Cost</th>
<th>Usage Fee</th>
<th>Advantage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Heavy Utilization</td>
<td>Highest</td>
<td>Lowest hourly fee. Applied to the whole term whether or not you're using the reserved node.</td>
<td>Lowest overall cost if you plan to run your reserved nodes more than 79 percent of a three-year term.</td>
</tr>
<tr>
<td>Medium Utilization</td>
<td>Medium</td>
<td>Hourly usage fee charged for each hour the node is running. No hourly charge when the node is not running.</td>
<td>Suitable for elastic workloads or when you expect moderate usage, more than 40 percent of a three-year term.</td>
</tr>
<tr>
<td>Light Utilization</td>
<td>Lowest</td>
<td>Hourly usage fee charged for each hour the node is running. No hourly charge when the node is not running. Highest hourly fees of all the offering types, but fees apply only when the reserved node is running.</td>
<td>Highest overall cost if you plan to run all of the time; however, lowest overall cost if you anticipate you will use your reserved node infrequently, more than about 15 percent of a three-year term.</td>
</tr>
<tr>
<td>On-Demand Use (No reserved nodes)</td>
<td>None</td>
<td>Highest hourly fee. Applied whenever the node is running.</td>
<td>Highest hourly cost.</td>
</tr>
</tbody>
</table>

For more information, see Amazon ElastiCache Pricing.
Getting Info About Reserved Node Offerings

Before you purchase reserved nodes, you can get information about available reserved node offerings.

The following examples show how to get pricing and information about available reserved node offerings using the AWS Management Console, AWS CLI, and ElastiCache API.

**Topics**
- Getting Info About Reserved Node Offerings (Console) (p. 109)
- Getting Info About Reserved Node Offerings (AWS CLI) (p. 109)
- Getting Info About Reserved Node Offerings (ElastiCache API) (p. 110)

## Getting Info About Reserved Node Offerings (Console)

To get pricing and other information about available reserved cluster offerings using the AWS Management Console, follow the steps in the following procedure.

**To get information about available reserved node offerings**

2. In the navigation list, choose the **Reserved Cache Nodes** link.
3. Choose the **Purchase Reserved Cache Node** button.
4. From the **Product Description** drop down list box, choose 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. Choose Cancel to avoid purchasing these nodes and incurring charges.

## Getting Info About Reserved Node Offerings (AWS CLI)

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

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

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

```json
{
   "ReservedCacheNodesOfferings": [
      {
         "OfferingType": "Heavy Utilization",
         "FixedPrice": 4328.0,
         "ReservedCacheNodesOfferingId": "0192caa9-daf2-4159-b1e5-a79bb1916695",
         "UsagePrice": 0.0,
         "RecurringCharges": [
            {
            ...
```
For more information, see describe-reserved-cache-nodes-offerings in the AWS CLI Reference.

Getting Info About Reserved Node Offerings (ElastiCache API)

To get pricing and information about available reserved node offerings, call the DescribeReservedCacheNodesOfferings action.

Example

https://elasticache.us-west-2.amazonaws.com/
?Action=DescribeReservedCacheNodesOfferings
&Version=2014-12-01
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20141201T220302Z
&X-Amz-Algorithm
&X-Amz-SignedHeaders=Host
&X-Amz-Expires=20141201T220302Z
&X-Amz-Credential=<credential>
&X-Amz-Signature=<signature>

This call returns output similar to the following:

  <DescribeReservedCacheNodesOfferingsResult>
    <ReservedCacheNodesOfferings>
      <ReservedCacheNodesOffering>
        <Duration>31536000</Duration>
        <OfferingType>Medium Utilization</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <FixedPrice>1820.0</FixedPrice>
      </ReservedCacheNodesOffering>
    </ReservedCacheNodesOfferings>
  </DescribeReservedCacheNodesOfferingsResult>
</DescribeReservedCacheNodesOfferingsResponse>
For more information, see DescribeReservedCacheNodesOfferings in the ElastiCache API Reference.
Purchasing a Reserved Node

The following examples show how to purchase a reserved node offering using the AWS Management Console, the AWS CLI, and the ElastiCache API.

**Important**
Following the examples in this section will incur charges on your AWS account that you cannot reverse.

**Topics**
- Purchasing a Reserved Node (Console) (p. 112)
- Purchasing a Reserved Node (AWS CLI) (p. 112)
- Purchasing a Reserved Node (ElastiCache API) (p. 113)

Purchasing a Reserved Node (Console)

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

The following procedure uses the AWS Management Console to purchase the reserved node offering by offering id.

**To purchase reserved nodes**

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

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

8. Choose the Next button.

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

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

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

Purchasing a Reserved Node (AWS CLI)

The following example shows purchasing the specific reserved cluster offering, 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f, with a reserved node ID of myreservationID.
Type the following command at a command prompt:

For Linux, macOS, or Unix:

```bash
aws elasticache purchase-reserved-cache-nodes-offering \
  --reserved-cache-nodes-offering-id 649fd0c8-cf6d-47a0-bfa6-060f8e75e95f \
  --reserved-cache-node-id myreservationID
```

For Windows:

```bash
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:

<table>
<thead>
<tr>
<th>Reservation</th>
<th>ReservationId</th>
<th>Class</th>
<th>Start Time</th>
<th>Duration</th>
<th>Fixed Price</th>
<th>Usage Price</th>
<th>Count</th>
<th>State</th>
<th>Description</th>
<th>Offering Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>RESERVATION</td>
<td>myreservationid</td>
<td>cache.m1.small</td>
<td>2013-12-19T00:30:23.247Z</td>
<td>1y</td>
<td>455.00 USD</td>
<td>0.092 USD</td>
<td>1</td>
<td>payment-pending</td>
<td>memcached</td>
<td>Medium Utilization</td>
</tr>
</tbody>
</table>

For more information, see `purchase-reserved-cache-nodes-offering` in the AWS CLI Reference.

**Purchasing a Reserved Node (ElastiCache API)**

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

Call the `PurchaseReservedCacheNodesOffering` operation with the following parameters:

- `ReservedCacheNodesOfferingId` = `649fd0c8-cf6d-47a0-bfa6-060f8e75e95f`
- `ReservedCacheNodeID` = `myreservationID`
- `CacheNodeCount` = 1

**Example**

```xml
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:

```xml
  <PurchaseReservedCacheNodesOfferingResult>
    <ReservedCacheNode>
```

API Version 2015-02-02

113
Purchasing a Reserved Node

For more information, see PurchaseReservedCacheNodesOffering in the ElastiCache API Reference.
Getting Info About Your Reserved Nodes

You can get information about the reserved nodes you've purchased using the AWS Management Console, the AWS CLI, and the ElastiCache API.

Topics

- Getting Info About Your Reserved Nodes (Console) (p. 115)
- Getting Info About Your Reserved Nodes (AWS CLI) (p. 115)
- Getting Info About Your Reserved Nodes (ElastiCache API) (p. 115)

Getting Info About Your Reserved Nodes (Console)

The following procedure describes how to use the AWS Management Console to get information about the reserved nodes you purchased.

To get information about your purchased reserved nodes

2. In the navigation list, choose the Reserved Cache Nodes link.

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

Getting Info About Your Reserved Nodes (AWS CLI)

To get information about reserved nodes for your AWS account, type the following command at a command prompt:

```bash
aws elasticache describe-reserved-cache-nodes
```

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

```json
{
  "ReservedCacheNodeId": "myreservationid",
  "ReservedCacheNodesOfferingId": "649fd0c8-cf6d-47a0-bfa6-060f8e75e95f",
  "CacheNodeType": "cache.m1.small",
  "Duration": "31536000",
  "ProductDescription": "memcached",
  "OfferingType": "Medium Utilization",
  "MaxRecords": 0
}
```

For more information, see describe-reserved-cache-nodes in the AWS CLI Reference.

Getting Info About Your Reserved Nodes (ElastiCache API)

To get information about reserved nodes for your AWS account, call the DescribeReservedCacheNodes operation.

Example

```text
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:

```xml
  <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>
        <ReservedCacheNodeCount>1</ReservedCacheNodeCount>
        <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>
      <!-- ...some output omitted for brevity... -->
    </ReservedCacheNodes>
    </DescribeReservedCacheNodesResult>
  </DescribeReservedCacheNodesResponse>
```

For more information, see `DescribeReservedCacheNodes` in the ElastiCache API Reference.
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:**
    - **T2 node types:** cache.t2.micro, cache.t2.small, cache.t2.medium
    - **M3 node types:** cache.m3.medium, cache.m3.large, cache.m3.xlarge, cache.m3.2xlarge
    - **M4 node types:** cache.m4.large, cache.m4.xlarge, cache.m4.2xlarge, cache.m4.4xlarge, cache.m4.10xlarge
  - **Previous generation:** (not recommended)
    - **T1 node types:** cache.t1.micro
    - **M1 node types:** cache.m1.small, cache.m1.medium, cache.m1.large, cache.m1.xlarge
- **Compute optimized:**
  - **Previous generation:** (not recommended)
    - **C1 node types:** cache.c1.xlarge
- **Memory optimized:**
  - **Current generation:**
    - **R3 node types:** cache.r3.large, cache.r3.xlarge, cache.r3.2xlarge, cache.r3.4xlarge, cache.r3.8xlarge
    - **R4 node types:** cache.r4.large, cache.r4.xlarge, cache.r4.2xlarge, cache.r4.4xlarge, cache.r4.8xlarge, cache.r4.16xlarge
  - **Previous generation:** (not recommended)
    - **M2 node types:** cache.m2.xlarge, cache.m2.2xlarge, cache.m2.4xlarge

**Additional node type info**

- All T2 instances are created in an Amazon Virtual Private Cloud (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 instances.
- Redis Multi-AZ with automatic failover is supported on T2 instances only when running Redis (cluster mode enabled) - version 3.2.4 or later with the default.redis3.2.cluster.on parameter group or one derived from it.
- Redis configuration variables appendonly and appendfsync are not supported on Redis version 2.8.22 and later.

**Supported engine versions**

Supported engine versions vary by region. The latest engine versions are supported in all regions. To find the available engine versions in your region see Determining Available Engine Versions (p. 45).
### Supported Node Types by Region

<table>
<thead>
<tr>
<th>Region Name</th>
<th>Region</th>
<th>T2</th>
<th>M3</th>
<th>M4</th>
<th>R3</th>
<th>R4</th>
</tr>
</thead>
<tbody>
<tr>
<td>US East (Ohio)</td>
<td>us-east-2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>US East (N. Virginia)</td>
<td>us-east-1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>US West (N. California)</td>
<td>us-west-1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>US West (Oregon)</td>
<td>us-west-2</td>
<td>Yes</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Canada (Central)</td>
<td>ca-central-1</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Asia Pacific (Mumbai)</td>
<td>ap-south-1</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Asia Pacific (Seoul)</td>
<td>ap-northeast-2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Asia Pacific (Singapore)</td>
<td>ap-southeast-1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Asia Pacific (Sydney)</td>
<td>ap-southeast-2</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Asia Pacific (Tokyo)</td>
<td>ap-northeast-1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EU (Frankfurt)</td>
<td>eu-central-1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EU (Ireland)</td>
<td>eu-west-1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>EU (London)</td>
<td>eu-west-2</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EU (Paris)</td>
<td>eu-west-3</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>South America (São Paulo)</td>
<td>sa-east-1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>China (Beijing)</td>
<td>cn-north-1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>China (Ningxia)</td>
<td>cn-northwest-1</td>
<td>Yes</td>
<td></td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>AWS GovCloud (US)</td>
<td>gov-uswest-1</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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

  **Note**
  The ability to change your replacement window by moving your maintenance window is only available when the ElastiCache notification includes a maintenance window. If the notification does not include a maintenance window, you cannot change your replacement window.

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

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

  For instructions, see Maintenance Window (p. 56).

- **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. 190).
  2. Add a new node to the cluster. For instructions, see Adding Nodes to a Cluster (p. 184).
  3. If you are not using Node Auto Discovery (Memcached) (p. 123) 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, choose from the following menu.

- **Do nothing (p. 121)** – Let Amazon ElastiCache replace the node as scheduled.
- **Change your maintenance window (p. 121)** – Change your maintenance window to a better time.
- **Replace a read-replica (p. 121)** – A procedure to manually replace a read-replica in a Redis replication group.
- **Replace the primary node (p. 122)** – A procedure to manually replace the primary node in a Redis replication group.
- **Replace a standalone node (p. 122)** – 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 or a notification event from ElastiCache, if you change your maintenance window before the scheduled replacement time, your node will now be replaced at the new time. The new maintenance time can be no earlier than the originally scheduled time, and no later than a week from the originally scheduled time.

  **Note**
  
  The ability to change your replacement window by moving your maintenance window is only available when the ElastiCache notification includes a maintenance window. If the notification does not include a maintenance window, you cannot change your replacement window.

  For example:

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

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

  For instructions, see Maintenance Window (p. 56).

- **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. 284).
To replace a read replica

1. Delete the replica that is scheduled for replacement. For instructions, see Deleting a Cluster (p. 197).
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. 287).
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. 246).

- 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. 284).

To replace a primary node

1. Promote a read-replica to primary. For instructions, see Promoting a Read-Replica to Primary (p. 289).
2. Delete the node that is scheduled for replacement (the old primary). For instructions, see Deleting a Cluster (p. 197).
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. 287).
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. 246).

- 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 Making Manual Backups (p. 297).
2. Create a new node seeding it from the snapshot. For instructions, see Restoring From a Backup with Optional Cluster Resizing (p. 317).
3. Delete the node scheduled for replacement. For instructions, see Deleting a Cluster (p. 197).
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. 184).
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. 185).
3. Promote the newly created read-replica to primary. For instructions, see Promoting a Read-Replica to Primary (p. 289).
4. Delete the node scheduled for replacement. For instructions, see Deleting a Cluster (p. 197).
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 added for cache clusters running on Amazon ElastiCache Memcached. Redis (cluster mode enabled) clusters natively support auto discovery.

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

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

**Topics**
- Benefits of Auto Discovery (p. 124)
- How Auto Discovery Works (p. 125)
- Using Auto Discovery (p. 128)
- Connecting to Cache Nodes Manually (p. 133)
- Adding Auto Discovery To Your Client Library (p. 134)
- ElastiCache Clients with Auto Discovery (p. 135)
Benefits of Auto Discovery

Auto Discovery offers the following benefits:

- When you increase the number of nodes in a cache cluster, the new nodes register themselves with the configuration endpoint and with all of the other nodes. When you remove nodes from the cache cluster, the departing nodes deregister themselves. In both cases, all of the other nodes in the cluster are updated with the latest cache node metadata.
- Cache node failures are automatically detected; failed nodes are automatically replaced.
  
  **Note**
  
  Until node replacement completes, the node will continue to fail.
- A client program only needs to connect to the configuration endpoint. After that, the Auto Discovery library connects to all of the other nodes in the cluster.
- Client programs poll the cluster once per minute (this interval can be adjusted if necessary). If there are any changes to the cluster configuration, such as new or deleted nodes, the client receives an updated list of metadata. Then the client connects to, or disconnects from, these nodes as needed.

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

Topics
- Connecting to Cache Nodes (p. 125)
- Normal Cluster Operations (p. 126)
- Other Operations (p. 127)

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.
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.

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

Adding a Node

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

Deleting a Node

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

Replacing a Node

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

To begin using Auto Discovery, follow these steps:

- Step 1: Obtain the Configuration Endpoint (p. 128)
- Step 2: Download the ElastiCache Cluster Client (p. 129)
- Step 3: Modify Your Application Program (p. 130)

Step 1: Obtain the Configuration Endpoint

To connect to a cluster, client programs must know the cluster configuration endpoint. See the topic Finding a Memcached Cluster's Endpoints (Console) (p. 63)

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

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

Example Finding endpoints using the AWS CLI for ElastiCache

For Linux, macOS, or Unix:

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

For Windows:

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

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

```json
{
  "CacheClusters": [
    {
      "Engine": "memcached",
      "CacheNodes": [
        {
          "CacheNodeId": "0001",
          "Endpoint": {
            "Port": 11211,
            "Address": "mycluster.fnjyoz.cfg.0001.use1.cache.amazonaws.com"
          },
          "CacheNodeStatus": "available",
          "ParameterGroupStatus": "in-sync",
          "CustomerAvailabilityZone": "us-east-1e"
        },
        {
          "CacheNodeId": "0002",
          "Endpoint": {
            "Port": 11211,
```
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

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.

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. 138).

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. 136).
Step 3: Modify Your Application Program

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

**Important**
When specifying the cluster's configuration endpoint, be sure that the endpoint has ".cfg" in its address as shown here. Do not use a CNAME or an endpoint without ".cfg" in it.

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

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

**Topics**
- Using the ElastiCache Cluster Client for Java (p. 130)
- Using the ElastiCache Cluster Client for PHP (p. 130)
- Using the ElastiCache Cluster Client for .NET (p. 131)

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.

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

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

Using the ElastiCache Cluster Client for PHP

The program below demonstrates how to use the ElastiCache Cluster Client to connect to a cluster configuration endpoint and add a data item to the cache. Using Auto Discovery, the program will connect to all of the nodes in the cluster without any further intervention.
To use the ElastiCache Cluster Client for PHP, you will first need to install it on your Amazon EC2 instance. For more information, see Installing the ElastiCache Cluster Client for PHP (p. 138)

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

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

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

/**
 * The following will initialize a Memcached client to utilize the Auto Discovery feature.
 *
 * By configuring the client with the Dynamic client mode with single endpoint, the
 * client will periodically use the configuration endpoint to retrieve the current cache
 * cluster configuration. This allows scaling the cache cluster up or down in number of
 * nodes
 * without requiring any changes to the PHP application.
 *
 * By default the Memcached instances are destroyed at the end of the request.
 * To create an instance that persists between requests,
 * use persistent_id to specify a unique ID for the instance.
 * All instances created with the same persistent_id will share the same connection.
 */
$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 applications typically get their configurations from their config file. The following is a sample application config file.

```xml
<?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.

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

using System;
using Amazon.ElastiCacheCluster;
using Enyim.Caching;
using Enyim.Caching.Memcached;

public class DotNetAutoDiscoveryDemo {
    public static void Main(String[] args) {
        // instantiate a new client.
        ElastiCacheClusterConfig config = new ElastiCacheClusterConfig();
        MemcachedClient memClient = new MemcachedClient(config);

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

```java
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. 135) 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**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Required</th>
</tr>
</thead>
<tbody>
<tr>
<td>sub-command</td>
<td>The sub-command used to interact with a cache node. For Auto Discovery, this sub-command is <code>get</code>.</td>
<td>Yes</td>
</tr>
<tr>
<td>key</td>
<td>The key under which the cluster configuration is stored. For Auto Discovery, this key is named <code>cluster</code>.</td>
<td>Yes</td>
</tr>
</tbody>
</table>

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
hostname|ip-address|port hostname|ip-address|port hostname|ip-address|port
```

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
12
myCluster.pc4ldq.0001.use1.cache.amazonaws.com|10.82.235.120|11211
myCluster.pc4ldq.0002.use1.cache.amazonaws.com|10.80.249.27|11211
END
```

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

ElastiCache Clients with Auto Discovery

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

Topics
- Installing & Compiling Cluster Clients (p. 136)
- Configuring ElastiCache Clients (p. 147)
Installing & Compiling Cluster Clients

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

Topics

• Installing the ElastiCache Cluster Client for .NET (p. 136)
• Installing the ElastiCache Cluster Client for PHP (p. 138)
• Compiling the Source Code for the ElastiCache Cluster Client for PHP (p. 145)

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. 123). For sample .NET code to use the client, see Using the ElastiCache Cluster Client for .NET (p. 131).

Topics

• Installing .NET (p. 136)
• Download the ElastiCache .NET Cluster Client for ElastiCache (p. 136)
• Install AWS Assemblies with NuGet (p. 136)

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

2. On the navigation pane, click ElastiCache Cluster Client.
3. In the Download ElastiCache Memcached Cluster Client list, select .NET, and then click Download.

Install AWS Assemblies with NuGet

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

Installing NuGet

NuGet can be installed from the Installation Gallery on MSDN; 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.

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. 123). For sample PHP code to use the client, see Using the ElastiCache Cluster Client for PHP (p. 130).

Topics

- Downloading the Installation Package (p. 138)
- For Users Who Already Have php-memcached Extension Installed (p. 139)
- Installation Steps for New Users (p. 139)
- Removing the PHP Cluster Client (p. 144)

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:

  ```bash
  php -v
  ```

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

  ```text
  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:

  ```text
  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. 145).

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**

2. From the ElastiCache console, choose **ElastiCache Cluster Client**.
3. From the **Download ElastiCache Memcached Cluster Client** list, choose the ElastiCache Cluster Client that matches your PHP version and AMI architecture, then choose the **Download** button.

**For Users Who Already Have php-memcached Extension Installed**

**To update the php-memcached installation**

1. Remove the previous installation of the Memcached extension for PHP as described by the topic Removing the PHP Cluster Client (p. 144).
2. Install the new ElastiCache php-memcached extension as described previously in Installation Steps for New Users (p. 139).

**Installation Steps for New Users**

**Topics**

- Installing PHP 7.x for New Users (p. 139)
- Installing PHP 5.x for New Users (p. 141)

**Installing PHP 7.x for New Users**

**Topics**

- To install PHP 7 on a Ubuntu Server 14.04 LTS AMI (64-bit and 32-bit) (p. 139)
- To install PHP 7 on an Amazon Linux 201609 AMI (p. 140)
- To install PHP 7 on an SUSE Linux AMI (p. 140)

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

1. Launch a new instance from the AMI.
2. Run the following commands:
   
   ```
   sudo apt-get update
   sudo apt-get install gcc g++
   ```
3. Install PHP 7.
   ```
   sudo yum install php70
   ```
4. Download the Amazon ElastiCache Cluster Client.
   ```
   wget https://elasticache-downloads.s3.amazonaws.com/ClusterClient/PHP-7.0/latest-64bit
   ```
5. Extract latest-64bit.
   ```
   tar -zxvf latest-64bit
   ```

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

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

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

8. Start or restart your Apache server.

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

**To install PHP 7 on an Amazon Linux 201609 AMI**

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

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

3. Install PHP 7.

   ```bash
   sudo yum install php70
   ```

4. Download the Amazon ElastiCache Cluster Client.

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

5. Extract `latest-64bit`.

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

6. With root permission, copy the extracted artifact file `amazon-elasticache-cluster-client.so` into `/usr/lib64/php/7.0/modules/`.

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

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

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

8. Start or restart your Apache server.

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

**To install PHP 7 on an SUSE Linux AMI**

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

```bash
sudo zypper install gcc
```

3. Install PHP 7.

```bash
sudo yum install php70
```

4. Download the Amazon ElastiCache Cluster Client.

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

5. Extract latest-64bit.

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


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

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

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

8. Start or restart your Apache server.

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

## Installing PHP 5.x for New Users

### Topics
- To install PHP 5 on an Amazon Linux AMI 2014.03 (64-bit and 32-bit) (p. 141)
- To install PHP 5 on a Red Hat Enterprise Linux 7.0 AMI (64-bit and 32-bit) (p. 142)
- To install PHP 5 on a Ubuntu Server 14.04 LTS AMI (64-bit and 32-bit) (p. 142)
- To install PHP 5 for SUSE Linux Enterprise Server 11 AMI (64-bit or 32-bit) (p. 143)
- Other Linux distributions (p. 144)

### 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:

```bash
$ 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. 138).

4. Install `php-memcached`. The URI should be the download path for the installation package:

```bash
```

API Version 2015-02-02
141
Here is a sample installation command for PHP 5.4, 64-bit Linux. In this sample, replace X.Y.Z with the actual version number:

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

**Note**
Please use the latest version of the install artifact.

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

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

6. Start or restart your Apache server.

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

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

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

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

3. Download the correct php-memcached package for your Amazon EC2 instance and PHP version. For more information, see Downloading the Installation Package (p. 138).

4. Install php-memcached. The URI should be the download path for the installation package:

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


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

6. Start or restart your Apache server.

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

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

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

```
sudo apt-get update
```

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


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

6. Start or restart your Apache server.

```
sudo /etc/init.d/httpd start
```
Installing & Compiling Clients

3. Download the correct php-memcached package for your Amazon EC2 instance and PHP version. For more information, see Downloading the Installation Package (p. 138).

4. Install php-memcached. The URI should be the download path for the installation package.

```bash
# 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-client.so` from the downloaded `.tgz` file, and copy it to the `/usr/lib/php5/20121212*` directory.

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


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

6. Start or restart your Apache server.

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

**To install PHP 5 for SUSE Linux Enterprise Server 11 AMI (64-bit or 32-bit)**

1. Launch a SUSE Linux instance (either 64-bit or 32-bit) and log into it.

2. Install PHP dependencies:

```bash
$ 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. 138).

4. Install php-memcached. The URI should be the download path for the installation package.

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


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

6. Start or restart your Apache server.

```bash
sudo /etc/init.d/httpd start
```
Note
If Step 5 doesn’t work for any of the previous platforms, please verify the install path for amazon-elasticache-cluster-client.so, and specify the full path of the binary in the extension. Also, verify that the PHP in use is a supported version. We support versions 5.3 through 5.5.

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

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

Removing the PHP Cluster Client

Topics
- Removing an earlier version of PHP 7 (p. 144)
- Removing an earlier version of PHP 5 (p. 144)

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. 139).
2. Remove the line extension=amazon-elasticache-cluster-client.so from the php.ini file.
3. Start or restart your Apache server.

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

Removing an earlier version of PHP 5

To remove an earlier version of PHP 5

1. Remove the php-memcached extension:

```
sudo pecl uninstall __uri/AmazonElastiCacheClusterClient
```
2. Remove the memcached.ini file added in the appropriate directory as indicated in the previous installation steps.
Compiling the Source Code for the ElastiCache Cluster Client for PHP

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

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

**Topics**
- Compiling the libmemcached Library (p. 145)
- Compiling the ElastiCache Memcached Auto Discovery Client for PHP (p. 145)

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
     ```bash
     sudo yum install gcc gcc-c++ autoconf libevent-devel
     ```
   - On Ubuntu 14.04 AMI
     ```bash
     sudo apt-get update
     sudo apt-get install libevent-dev gcc g++ make autoconf libsasl2-dev
     ```
3. Pull the repository and compile the code.
   ```bash
   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 (p. 145)
- Compiling the ElastiCache Memcached Client for PHP 5 (p. 146)

Compiling the ElastiCache Memcached Client for PHP 7

Run the following set of commands under the code directory.

```bash
git clone https://github.com/awslabs/aws-elasticache-cluster-client-memcached-for-php.git
cd aws-elasticache-cluster-client-memcached-for-php
git checkout php7
sudo yum install php70-devel
```
Installing & Compiling Clients

To install and compile the clients for PHP, follow these steps:

```
phpize
./configure --with-libmemcached-dir=<libmemcached-install-directory> --disable-memcached-sasl
make
make install
```

**Note**

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

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

## Compiling the ElastiCache Memcached Client for PHP 5

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

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

An ElastiCache cluster is protocol-compliant with Memcached or Redis, depending on which cache engine was chosen 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 (p. 147)
- Finding Node Endpoints and Port Numbers (p. 147)
- Connecting for Using Auto Discovery (p. 150)
- Connecting to Nodes in a Redis Cluster (p. 150)
- DNS Names and Underlying IP (p. 152)

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
  - sync

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 chosen 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:

```json
{
  "CacheClusters": [
    {
      "Engine": "redis",
      "CacheNodes": [
        {
          "CacheNodeId": "0001",
          "Endpoint": {
            "Port": 6379,
            "Address": "redis0x1.7adw3s.0001.usw2.cache.amazonaws.com"
          },
          "CacheNodeStatus": "available",
          "ParameterGroupStatus": "in-sync",
          "CustomerAvailabilityZone": "us-west-2b"
        }
      ],
      "CacheParameterGroup": {
        "CacheNodeIdsToReboot": [],
        "CacheParameterGroupName": "default.redis3.2",
        "ParameterApplyStatus": "in-sync"
      },
      "SnapshotRetentionLimit": 1,
      "CacheClusterId": "redis0x1",
      "CacheSecurityGroups": [],
      "NumCacheNodes": 1,
      "SnapshotWindow": "00:00-01:00",
      "CacheClusterCreateTime": "2017-04-05T20:45:28.907Z",
      "AutoMinorVersionUpgrade": true,
      "CacheClusterStatus": "available",
      "PreferredAvailabiltyZone": "us-west-2b",
      "CacheSubnetGroupName": "default",
      "EngineVersion": "3.2.4",
      "PendingModifiedValues": {}
    }
  ]
}
```
"PreferredMaintenanceWindow": "sun:06:00-sun:07:00",
"CacheNodeType": "cache.m3.medium"
},

****** some output omitted for brevity ******

{
  "Engine": "memcached",
  "CacheNodes": [
    {
      "CacheNodeId": "0001",
      "Endpoint": {
        "Port": 11211,
        "Address": "mem03.5edv7s.0001.usw2.cache.amazonaws.com"
      },
      "CacheNodeStatus": "available",
      "ParameterGroupStatus": "in-sync",
      "CustomerAvailabilityZone": "us-west-2a"
    }
  ],
  "CacheParameterGroup": {
    "CacheNodeIdsToReboot": [],
    "CacheParameterGroupName": "default.memcached1.4",
    "ParameterApplyStatus": "in-sync"
  },
  "CacheClusterId": "mem03",
  "PreferredAvailabilityZone": "us-west-2a",
  "ConfigurationEndpoint": {
    "Port": 11211,
    "Address": "mem03.9dcv5r.cfg.usw2.cache.amazonaws.com"
  },
  "CacheSecurityGroups": [],
  "AutoMinorVersionUpgrade": true,
  "CacheClusterStatus": "available",
  "NumCacheNodes": 1,
  "SecurityGroups": [
    {
      "Status": "active",
      "SecurityGroupId": "sg-dbe93fa2"
    }
  ],
  "CacheSubnetGroupName": "default",
  "EngineVersion": "1.4.34",
  "PendingModifiedValues": {},
  "PreferredMaintenanceWindow": "thu:10:30-thu:11:30",
  "CacheNodeType": "cache.t2.micro"
},
]

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

Finding Cache Node Endpoints and Port Numbers (ElastiCache API)

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

Example
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. 123).

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 a Redis (cluster mode disabled) Cluster's Endpoints (Console) (p. 65).

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:

```bash
aws elasticache describe-replication-groups redis2x2
```

This command should produce output similar to the following:

```json
{
   "ReplicationGroups": [
   {
      "Status": "available",
```
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

```plaintext
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. 163). 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, with your data partitioned across the 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

- Memcached Version 1.4.34 (p. 47)
- Memcached Version 1.4.33 (p. 47)
- Memcached Version 1.4.24 (p. 48)
- Memcached Version 1.4.14 (p. 48)
- Memcached Version 1.4.5 (p. 48)
Redis Versions

- ElastiCache for Redis Version 3.2.10 (Enhanced) (p. 50)
- ElastiCache for Redis Version 3.2.6 (Enhanced) (p. 50)
- ElastiCache for Redis Version 3.2.4 (Enhanced) (p. 51)
- ElastiCache for Redis Version 2.8.23 (Enhanced) (p. 52)
- ElastiCache for Redis Version 2.8.22 (Enhanced) (p. 52)
- ElastiCache for Redis Version 2.8.19 (p. 53)
- ElastiCache for Redis Version 2.8.6 (p. 53)
- ElastiCache for Redis Version 2.6.13 (p. 53)

Other ElastiCache Cluster Operations

Additional operations involving clusters:

- Finding Your ElastiCache Endpoints (p. 62)
- Accessing ElastiCache Resources from Outside AWS (p. 441)
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. 41) 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. 41).
- Which node instance type do you need?
  For guidance on choosing an instance node type, see Choosing Your Node Size (p. 99).
- 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. 379).
  An advantage of launching in an 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. 441).
- 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. 340).
  If you're running Redis you may want to consider at least setting reserved-memory or reserved-memory-percent. For more information, see Managing Reserved Memory (Redis) (p. 79).
- Do you need to create your own Security Group or VPC Security Group?
  For more information, see Security Groups [EC2-Classic] (p. 328) and Security in Your VPC.
- How do you intend to implement fault tolerance?
  For more information, see Mitigating Failures (p. 83).

Topics
- Creating a Cluster (Console): Memcached (p. 157)
- Creating a Redis (cluster mode disabled) Cluster (Console) (p. 159)
- Creating a Redis (cluster mode enabled) Cluster (Console) (p. 163)
- Creating a Cache Cluster (AWS CLI) (p. 168)
- Creating a Cache Cluster (ElastiCache API) (p. 170)
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. 123).

To create a Memcached cluster using the ElastiCache console:

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

Memcached cluster with data partitioning

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

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

      Important
      You can upgrade to newer engine versions. For more information, see Upgrading Engine Versions (p. 54). 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. 353) and Creating a Parameter Group (p. 340).
e. For **Node type**, click the down arrow (▼). In the **Change node type** dialog box, choose the **Instance family** of the node type you want, choose the node type you want to use for this cluster, and then choose **Save**.

For more information, see Choosing Your Node Size (p. 99).

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. 201)

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

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

For more information, see Subnets and Subnet Groups (p. 379).

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

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

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

We recommend locating your nodes in multiple Availability Zones for improved fault tolerance. For more information, see Mitigating Availability Zone Failures (p. 84).

For more information, see Choosing Regions and Availability Zones (p. 58).

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

For more information, see ElastiCache and Security Groups (p. 406).

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

For more information, see Maintenance Window (p. 56).

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/](https://aws.amazon.com/sns/).

8. Review all your entries and choices, then go back and make any needed corrections. When you're ready, choose **Create** to launch your cluster.

As soon as your cluster's status is **available**, you can grant Amazon EC2 access to it, connect to it, and begin using it. For more information, see Step 4: Authorize Access (p. 32) and Step 5: Connect to a Cluster's Node (p. 36).

**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. 197).
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. This key 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. 235).

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. From the dropdown in the upper right corner, choose the region you want to launch this cluster in.
3. Choose Redis from the navigation pane.
4. Choose Create.
5. For Cluster engine, choose Redis, and then clear the Cluster Mode enabled (Scale Out) check box.
6. 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. In the Description box, type in a description for this cluster.
   c. For Engine version compatibility, choose the ElastiCache for 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. 54)), 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 when using the ElastiCache console. We recommend against using these Redis versions. If you need to use one of them, work with the AWS CLI or ElastiCache API.

For more information, see the following topics:

<table>
<thead>
<tr>
<th></th>
<th>AWS CLI</th>
<th>ElastiCache API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Cluster</td>
<td>Creating a Cache Cluster (AWS CLI) (p. 168)</td>
<td>Creating a Cache Cluster (ElastiCache API) (p. 170)</td>
</tr>
<tr>
<td></td>
<td>This action cannot be used to create a replication group with cluster mode enabled.</td>
<td>This action cannot be used to create a replication group with cluster mode enabled.</td>
</tr>
</tbody>
</table>
To encrypt your data while it is in transit, for Encryption, choose Yes.

If you chose Yes for Encryption, you can require users to enter a password when executing Redis commands. To require a password when executing commands, do the following:

i. Choose Yes from the AUTH list.

ii. Type in a password in the AUTH token box:

**AUTH Token Constraints when using with ElastiCache**

- Passwords must be at least 16 and a maximum of 128 printable characters.
- The printable characters @, `, and / cannot be used in passwords.
- AUTH can only be enabled when creating clusters where in-transit encryption is enabled.
- The password set at cluster creation cannot be changed.

We recommend that you follow a stricter policy such as:

- Must include a mix of characters that includes at least three of the following character types:
  - Uppercase characters
  - Lowercase characters
  - Digits
  - Non-alphanumeric characters ( !, &, #, $, ^, <, >, )
- Must not contain a dictionary word or a slightly modified dictionary word.
- Must not be the same as or similar to a recently used password.

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

For Parameter group, choose the parameter group you want to use with this cluster, or choose Create new to create a new parameter group with this cluster.
Parameter groups control the runtime parameters of your cluster. For more information on parameter groups, see Redis Specific Parameters (p. 362) and Creating a Parameter Group (p. 340).

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

For more information, see Choosing Your Node Size (p. 99).

i. 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 look like the following.

![Redis (cluster mode disabled) cluster created with no replica nodes](image)

If you choose one or more replicas, the cluster you create looks something like the following.

![Redis (cluster mode disabled) cluster created with replica nodes](image)

7. 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 that you enable Multi-AZ with Auto-Failover. For more information, see Mitigating Failures when Running Redis (p. 84).

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

For more information, see Subnets and Subnet Groups (p. 379).

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

   - **No preference** – ElastiCache chooses 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 choosing the Availability Zone from the list to the right of each node name.

For more information, see Choosing Regions and Availability Zones (p. 58).

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

For more information, see ElastiCache and Security Groups (p. 406).

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 Seeding a New Cluster with an Externally Created Backup (Redis) (p. 320).

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 and Restore (Redis) (p. 293).

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 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, 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. 56).

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/.

8. Review all your entries and choices, then go back and make any needed corrections. When you're ready, choose Create to launch your cluster.

As soon as your cluster's status is available, you can grant Amazon EC2 access to it, connect to it, and begin using it. For more information, see Step 4: Authorize Access (p. 32) and Step 5: Connect to a Cluster's Node (p. 36).

**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. 197).
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. For a comparison of Redis (cluster mode disabled) and the two types of Redis (cluster mode enabled), see Choosing an Engine: Memcached, Redis (cluster mode disabled), or Redis (cluster mode enabled) (p. 42).

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. From the dropdown in the upper right corner, choose the region you want to launch this cluster in.
3. Choose Redis from the navigation pane.
4. Choose Create.
5. For Cluster engine, choose Redis, and then choose Cluster Mode enabled (Scale Out). These selections create a Redis (cluster mode enabled) cluster that looks something like the following.

![Redis (cluster mode enabled) cluster created with replication and data partitioning](image)

6. 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 a description of the cluster.
   c. If you want to enable in-transit encryption for this cluster, choose In-transit encryption.

   If you choose In-transit encryption, two additional options appear: Redis auth token and a box where you type in the token (password) value.
   d. If you want to enable at-rest encryption for this cluster, choose At-rest encryption.
   e. To require a password for operations to be performed on this cluster:
      i. Choose Redis auth token.
      ii. In the Redis auth token box, type the token (password) that must be used when performing operations on this cluster.
AUTH Token Constraints when using with ElastiCache

- Passwords must be at least 16 and a maximum of 128 printable characters.
- The printable characters @, "", and / cannot be used in passwords.
- AUTH can only be enabled when creating clusters where in-transit encryption is enabled.
- The password set at cluster creation cannot be changed.

We recommend that you follow a stricter policy such as:

- Must include a mix of characters that includes at least three of the following character types:
  - Uppercase characters
  - Lowercase characters
  - Digits
  - Non-alphanumeric characters (1, &, #, $, ^, <, >, -)
- Must not contain a dictionary word or a slightly modified dictionary word.
- Must not be the same as or similar to a recently used password.

f. For **Engine version compatibility**, choose the ElastiCache for 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.

g. To encrypt your data while it is in transit, for **Encryption**, choose Yes.

h. If you chose Yes for Encryption, you can require users to enter a password when executing Redis commands. To require a password when executing commands, do the following:

i. Choose Yes from the **AUTH** list.

ii. Type in a password in the **AUTH token** box.

AUTH Token Constraints when using with ElastiCache

- Passwords must be at least 16 and a maximum of 128 printable characters.
- The printable characters @, "", and / cannot be used in passwords.
- AUTH can only be enabled when creating clusters where in-transit encryption is enabled.
- The password set at cluster creation cannot be changed.

We recommend that you follow a stricter policy such as:

- Must include a mix of characters that includes at least three of the following character types:
  - Uppercase characters
  - Lowercase characters
  - Digits
  - Non-alphanumeric characters (1, &, #, $, ^, <, >, -)
- Must not contain a dictionary word or a slightly modified dictionary word.
- Must not be the same as or similar to a recently used password.

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

j. 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. 362) and Creating a Parameter Group (p. 340).

k. 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 Choosing Your Node Size (p. 99).

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

In Redis (cluster mode enabled), depending upon the version of Redis running on your cluster, you may be able to change the number of shards in your cluster dynamically.

- **Redis 3.2.10**—If your cluster is running Redis 3.2.10 you can change the number of shards in your cluster dynamically. For more information, see Scaling for Amazon ElastiCache for Redis—Redis (cluster mode enabled) (p. 225).

- **Other Redis versions**—If your cluster is running a version of Redis other than version 3.2.10, to change the number of shards in your cluster, you must create a new cluster with the new number of shards. For more information, see Restoring From a Backup with Optional Cluster Resizing (p. 317).

m. 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 shard (API/CLI: node group), you must create a new cluster with the new number of replicas. For more information, see Seeding a New Cluster with an Externally Created Backup (Redis) (p. 320).

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

For more information, see Subnets and Subnet Groups (p. 379).

7. 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 16383).

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

- **No preference** – ElastiCache chooses the Availability Zone.

- **Specify availability zones** – You specify the Availability Zone for each cluster.

If you chose to specify the Availability Zones, for each cluster in each shard, choose the Availability Zone from the list.
For more information, see Choosing Regions and Availability Zones (p. 58).

Specifying Keypaces 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. 406).

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 Seeding a New Cluster with an Externally Created Backup (Redis) (p. 320).

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 and Restore (Redis) (p. 293).

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. 56).

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/.

8. Review all your entries and choices, then go back and make any needed corrections. When you're ready, 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:
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. 32) and Step 5: Connect to a Cluster's Node (p. 36).

**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. 197).
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) Cluster with Replicas from Scratch (AWS CLI) (p. 261).

**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. 197).

**Topics**
- Creating a Memcached Cache Cluster (AWS CLI) (p. 168)
- Creating a Redis (cluster mode disabled) Cache Cluster (AWS CLI) (p. 168)
- Creating a Redis (cluster mode enabled) Cluster (AWS CLI) (p. 169)

Creating a Memcached Cache Cluster (AWS CLI)

The following CLI code creates a Memcached cache cluster.

For Linux, macOS, or Unix:

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

For Windows:

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

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

**Example – A Redis (cluster mode disabled) Cluster with no read replicas**

The following CLI code creates a Redis (cluster mode disabled) cache cluster with no replicas.

For Linux, macOS, or Unix:

```bash
aws elasticache create-cache-cluster \
  --cache-cluster-id my-redis3-cluster \
  --cache-node-type cache.r4.large \
  --engine redis \
  --engine-version 3.2.4 \
```

API Version 2015-02-02
168
For Windows:

```bash
aws elasticache create-cache-cluster
  --cache-cluster-id my-redis3-cluster
  --cache-node-type cache.r4.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)

Redis (cluster mode enabled) clusters (API/CLI: replication groups) cannot be created using the ```create-cache-cluster``` operation. To create a Redis (cluster mode enabled) cluster (API/CLI: replication group), see Creating a Redis (cluster mode enabled) Cluster with Replicas from Scratch (AWS CLI) (p. 267).

For more information, go to the AWS CLI for ElastiCache reference topic ```create-replication-group```.
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) Cluster with Replicas from Scratch (ElastiCache API) (p. 264).

**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. 197).

**Topics**

- Creating a Memcached Cache Cluster (ElastiCache API) (p. 170)
- Creating a Redis (cluster mode disabled) Cache Cluster (ElastiCache API) (p. 170)
- Creating a Redis (cluster mode enabled) Cache Cluster (ElastiCache API) (p. 171)

Creating a Memcached Cache Cluster (ElastiCache API)

The following code creates a Memcached cluster with 4 nodes (ElastiCache API). Line breaks are added for ease of reading.

```plaintext
https://elasticache.us-west-2.amazonaws.com/
?Action=CreateCacheCluster
&CacheClusterId=myMemcachedCluster
&CacheNodeType=cache.r4.large
&Engine=memcached
&NumCacheNodes=4
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150508T220302Z
&Version=2015-02-02
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=<credential>
&X-Amz-Date=20150508T220302Z
&X-Amz-Expires=20150508T220302Z
&X-Amz-SignedHeaders=Host
&X-Amz-Signature=<signature>
```

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 are added for ease of reading.

```plaintext
https://elasticache.us-west-2.amazonaws.com/
?Action=CreateCacheCluster
&CacheClusterId=my-redis2-cluster
&CacheNodeType=cache.r4.large
&CacheParameterGroup=default.redis3.2
&Engine=redis
&EngineVersion=3.2.4
&NumCacheNodes=1
```
Creating a Redis (cluster mode enabled) Cache Cluster (ElastiCache API)

Redis (cluster mode enabled) clusters (API/CLI: replication groups) cannot be created using the CreateCacheCluster operation. To create a Redis (cluster mode enabled) cluster (API/CLI: replication group), see Creating a Redis (cluster mode enabled) Cluster with Replicas from Scratch (ElastiCache API) (p. 271).

For more information, go to the ElastiCache API reference topic CreateReplicationGroup.
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) (p. 172)
- Viewing a Redis (cluster mode disabled) Cluster's Details (Console) (p. 174)
- Viewing a Redis (cluster mode enabled) Cluster's Details (Console) (p. 175)
- Viewing a Cluster's Details (AWS CLI) (p. 176)
- Viewing a Cluster's Details (ElastiCache API) (p. 178)

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. From the dropdown in the upper right corner, choose the region you are interested in.
3. In the ElastiCache console dashboard, choose Memcached. This will display a list of all your clusters that are running any version of Memcached.
4. To see details of a cluster, choose the box to the left of the cluster's name.
5. To view node information:
   a. Choose the cluster's name.
   b. Choose the Nodes tab.
   c. To view metrics on one or more nodes, choose the box to the left of the Node ID, and then choose the time range for the metrics from the Time range list. Selecting multiple nodes will generate overlay graphs.
Metrics over the last hour for two Memcached nodes
Viewing a 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. Doing this displays 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. Doing this displays 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.
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. From the dropdown in the upper right corner, choose the region you are interested in.
3. In the ElastiCache console dashboard, choose Redis to display a list of all your clusters that are running any version of Redis.
4. 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 Clustered Redis engine, not just Redis.

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

6. 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](image)

### 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.

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

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

```bash
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, macOS, 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):

```json
{
  "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",
          "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",
          "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",
          "CustomerAvailabilityZone": "us-west-2b"
        }
      ],
      "CacheParameterGroup": {
        "CacheNodeIdsToReboot": [],
        "CacheParameterGroupName": "default.memcached1.4",
        "ParameterApplyStatus": "in-sync"
      }
```
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.

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. 353) and Redis Specific Parameters (p. 362). For information on rebooting a cluster, go to Rebooting a Cluster (p. 182).

Modifying a Cluster (Console)

To modify a cluster (console)

2. From the dropdown in the upper right corner, choose the region you are interested in.
3. In the navigation pane, choose Redis or Memcached.

A list of the chosen engine's clusters appears.

4. 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.
5. Choose Modify.

The Modify Cluster window appears.

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

Important

You can upgrade to newer engine versions (see Upgrading Engine Versions (p. 54)), 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 when using the ElastiCache console. We recommend against using these Redis versions. If you need to use one of them, work with the AWS CLI or ElastiCache API.

For more information, see the following topics:

<table>
<thead>
<tr>
<th>AWS CLI</th>
<th>ElastiCache API</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Create Cluster</strong></td>
<td>Creating a Cache Cluster (AWS CLI) (p. 168)</td>
</tr>
<tr>
<td>This action cannot be used to create a replication group with cluster mode enabled.</td>
<td>This action cannot be used to create a replication group with cluster mode enabled.</td>
</tr>
</tbody>
</table>
7. 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**), but you cannot downgrade to older engine versions except by deleting the existing cache cluster or replication group and creating it anew.

For Linux, macOS, 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 ^
```
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. 54)), but you cannot downgrade to older engine versions except by deleting the existing cache cluster or replication group and creating it anew.

Line breaks are added for ease of reading.

```plaintext
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, for some parameters, changing the parameter value in a parameter group is only applied after a reboot.

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

Rebooting a cluster is currently supported on Memcached and Redis (cluster mode disabled) clusters. Rebooting is not supported on Redis (cluster mode enabled) clusters.

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.

**Redis (cluster mode enabled) and Reboots**

If you make changes to parameters that require a Redis (cluster mode enabled) cluster reboot for the changes to be applied, follow these steps.

2. Delete the Redis (cluster mode enabled) cluster. See Deleting a Cluster (p. 197).
3. Restore the cluster using the altered parameter group and backup to seed the new cluster. See Restoring From a Backup with Optional Cluster Resizing (p. 317).

Rebooting a Cluster (Console)

You can reboot a cluster using the ElastiCache console.

**To reboot a cluster (console)**

2. From the dropdown in the upper right corner, choose the region you are interested in.
3. In the navigation pane, choose Memcached or Redis.
   
   A list of clusters running the chosen engine will appear.
4. Choose the cluster to reboot by choosing on the box to the left of the cluster’s name.

   The Reboot button will become active.

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

   The reboot cluster confirmation screen appears.
6. To reboot the cluster, choose Reboot. The status of the cluster will change to *rebooting cluster nodes*.
   
   To not reboot the cluster, choose Cancel.

To reboot multiple clusters, repeat steps 2 through 5 for each cluster you want to reboot.

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, macOS, or Unix:

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

For Windows:

```bash
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`.

```xml
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. 466).

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. 88).

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

- To add replication to a Redis cluster with no shards (p. 184)
- To add nodes to a Memcached or Redis (cluster mode disabled) cluster with one shard (console) (p. 185)

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

2. From the 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. 185).

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.

2. In the navigation pane, choose Memcached or Redis.

A list of clusters running the chosen engine appears.
3. From the list of clusters, choose the name of the cluster you want to add a node to. This cluster cannot be a Redis (cluster mode enabled) cluster or a Redis (cluster mode disabled) cluster with zero shards.

If your cluster is a Redis (cluster mode enabled) cluster, see Scaling for Amazon ElastiCache for Redis—Redis (cluster mode enabled) (p. 225).

If your cluster is a Redis (cluster mode disabled) cluster with zero shards, first complete the steps at To add replication to a Redis cluster with no shards (p. 184).
4. Choose Add node.
5. 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 add this node immediately, or choose No to add this node during your next maintenance window.

Impact of New Add and Remove Requests on Pending Requests

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Pending Operation</th>
<th>New Request</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>Delete</td>
<td>Delete</td>
<td>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.</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>Delete</td>
<td>Create</td>
<td>The new create request, pending or immediate, replaces the pending delete request.</td>
</tr>
</tbody>
</table>
For example, if nodes 0001, 0003, and 0007 are pending deletion and a new request to create a node is issued, a new node will be created and nodes 0001, 0003, and 0007 will not be deleted.

Scenario 3
Create
Delete
The new delete request, pending or immediate, replaces the pending create request.

For example, if there is a pending request to create two nodes and a new request is issued to delete node 0003, no new nodes will be created and node 0003 will be deleted.

Scenario 4
Create
Create
The new create request is added to the pending create request.

For example, if there is a pending request to create two nodes and a new request is issued to create three nodes, the new requests is added to the pending request and five nodes will be created.

**Important**
If the new create request is set to **Apply Immediately - Yes**, all create requests are performed immediately. If the new create request is set to **Apply Immediately - No**, all create requests are pending.

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

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

## 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. 255). 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. 192).
- **--apply-immediately** or **--no-apply-immediately** which specifies whether to add these nodes immediately or at the next maintenance window.

For Linux, macOS, or Unix:

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

For Windows:

```bash
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):

```json
{
  "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"
    }
  }
}
```
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 Adding Replicas to a Stand-Alone Redis (cluster mode disabled) Cluster (ElastiCache API) (p. 257). 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 NumCacheNodes 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. 194)).
  - ApplyImmediately Specifies whether to add these nodes immediately or at the next maintenance window.
  - Region Specifies the AWS region of the cluster you want to add nodes to.

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

**Example**

```python
https://elasticache.us-west-2.amazonaws.com/
?Action=ModifyCacheCluster
&ApplyImmediately=true
&NumCacheNodes=5
&CacheClusterId=myCacheCluster
&Region=us-east-2
&Version=2014-12-01
```
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. 88).

Topics
- Removing Nodes from a Cluster (Console) (p. 190)
- Removing Nodes from a Cluster (AWS CLI) (p. 192)
- Removing Nodes from a Cluster (ElastiCache API) (p. 194)

Removing Nodes from a Cluster (Console)

To remove nodes from a cluster (console)

2. From the dropdown in the upper right corner, choose the region of the cluster you want to remove nodes from.
3. In the navigation pane, choose Memcached or Redis.
   A list of clusters running the chosen engine appears.
4. From the list of clusters, choose the cluster name from which you want to remove a node.
   A list of the cluster's nodes appears.
5. Choose the box to the left of the node ID for the node you want to delete. Using the ElastiCache console, you can only delete one node at a time, so choosing multiple nodes will disable the Delete node button.
   The Delete Node dialog appears.
6. To delete the node, complete the Delete Node dialog box and choose Delete Node. To not delete the node, choose the Cancel.

Impact of New Add and Remove Requests on Pending Requests

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Pending Operation</th>
<th>New Request</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scenario 1</td>
<td>Delete</td>
<td>Delete</td>
<td>The new delete request, pending or immediate, replaces the pending delete request.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Scenario 2</td>
<td>Delete</td>
<td>Create</td>
<td>The new create request, pending or immediate, replaces the pending delete request.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>For example, if nodes 0001, 0003, and 0007 are pending deletion and a new request to create a node is issued,</td>
</tr>
</tbody>
</table>
### Scenarios

<table>
<thead>
<tr>
<th>Scenarios</th>
<th>Pending Operation</th>
<th>New Request</th>
<th>Results</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>a new node will be created and nodes 0001, 0003, and 0007 will not be deleted.</td>
</tr>
<tr>
<td>Scenario 3</td>
<td>Create</td>
<td>Delete</td>
<td>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.</td>
</tr>
</tbody>
</table>
| 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.

![Cache Cluster: mycachecluster](image1.png)

![Cache Cluster: mycachecluster](image2.png)
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, macOS, or Unix:

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

   For Windows:

   ```bash
   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):

   ```json
   {
     "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",
             "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",
             "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",
             "CustomerAvailabilityZone": "us-west-2b"
           }
       },
     ]
   }
   ```
2. Use the `modify-cache-cluster` CLI operation with a list of the nodes to remove, as in the following example.

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

- `--cache-cluster-id` The ID of the cache cluster you want to remove nodes from.
- `--num-cache-nodes` The `--num-cache-nodes` parameter specifies the number of nodes you want in this cluster after the modification is applied.
- `--cache-node-ids-to-remove` A list of node IDs you want removed from this cluster.
- `--apply-immediately` or `--no-apply-immediately` Specifies whether to remove these nodes immediately or at the next maintenance window.
- `--region` Specifies the region of the cluster you want to remove nodes from.

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

For Linux, macOS, or Unix:

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

For Windows:

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

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

```json
{
   "CacheCluster": {
      "Engine": "memcached",
      "CacheParameterGroup": {
         "CacheNodeIdsToReboot": [],
         "CacheParameterGroupName": "default.memcached1.4",
         "ParameterApplyStatus": "in-sync"
      },
      "CacheClusterId": "my-memcached-cluster",
      "PreferredAvailabilityZone": "us-east-2b",
      "ConfigurationEndpoint": {
         "Port": 11211,
         "Address": "rlh-mem000.7ef-example.cfg.usw2.cache.amazonaws.com"
      },
      "CacheSecurityGroups": [],
      "AutoMinorVersionUpgrade": true,
      "CacheClusterStatus": "modifying",
      "NumCacheNodes": 3,
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.

• **Region** Specifies the region of the cluster you want to remove a node from.

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

```plaintext
https://elasticache.us-west-2.amazonaws.com/
  ?Action=ModifyCacheCluster
  &CacheClusterId=myCacheCluster
  &ApplyImmediately=true
  &CacheNodeIdsToRemove.member.1=0004
  &CacheNodeIdsToRemove.member.2=0005
  &NumCacheNodes=3
  &Region us-east-2
  &Version=2014-12-01
  &SignatureVersion=4
  &SignatureMethod=HmacSHA256
  &Timestamp=20141201T220302Z
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Date=20141201T220302Z
  &X-Amz-SignedHeaders=Host
  &X-Amz-Expires=20141201T220302Z
  &X-Amz-Credential=<credential>
  &X-Amz-Signature=<signature>
```

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

2. From the dropdown in the upper right corner, choose the region you want to cancel a pending add or delete node operation in.
3. In the navigation pane, click Memcached or Redis. A list of clusters running the chosen engine will appear.
4. In the list of clusters, choose the name of the cluster that has pending operations you want to cancel.
5. To determine what operations are pending, choose the Description tab and check to see how many pending creations or deletions are shown. You cannot have both pending creations and pending deletions.
6. Choose the Nodes tab.
7. To cancel all pending operations, click Cancel Pending. The Cancel Pending dialog box appears.
8. Confirm that you want to cancel all pending operations by choosing the Cancel Pending button, or to keep the operations, choose Cancel.
Deleting a Cluster

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

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 Linux, macOS, or Unix:

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

For Windows:
aws elasticache delete-cache-cluster ^
   --cache-cluster-id myCluster ^
   --region us-east-2

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
   &Region us-east-2
   &SignatureVersion=4
   &SignatureMethod=HmacSHA256
   &Timestamp=20150202T220302Z
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
   &X-Amz-Date=20150202T220302Z
   &X-Amz-SignedHeaders=Host
   &X-Amz-Expires=20150202T220302Z
   &X-Amz-Credential=<credential>
   &X-Amz-Signature=<signature>

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

For more information, 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.

The following helps you find the correct topic for the scaling actions you want to perform.

**Scaling Memcached Clusters**

<table>
<thead>
<tr>
<th>Action</th>
<th>Topic/Link</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scaling out</td>
<td>Adding Nodes to a Cluster (p. 184)</td>
</tr>
<tr>
<td>Scaling in</td>
<td>Removing Nodes from a Cluster (p. 190)</td>
</tr>
<tr>
<td>Changing node types</td>
<td>Scaling Memcached Vertically (p. 202)</td>
</tr>
</tbody>
</table>

**Scaling Redis Clusters**

<table>
<thead>
<tr>
<th>Action</th>
<th>Redis (cluster mode disabled)</th>
<th>Redis (cluster mode enabled)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scaling in</td>
<td>Removing Nodes from a Cluster (p. 190)</td>
<td></td>
</tr>
<tr>
<td>Changing node types</td>
<td><strong>To a larger node type</strong></td>
<td><strong>Two-step process</strong></td>
</tr>
<tr>
<td></td>
<td>• Scaling Up Redis Clusters with Replicas (p. 214)</td>
<td>2. Restoring From a Backup with Optional Cluster Resizing (p. 317)</td>
</tr>
<tr>
<td></td>
<td><strong>To a smaller node type</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Scaling Down Single-Node Redis Clusters (p. 210)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>• Scaling Down Redis Clusters with Replicas (p. 220)</td>
<td></td>
</tr>
<tr>
<td>Changing the number of node groups</td>
<td>Not supported for Redis (cluster mode disabled) clusters</td>
<td>Offline Resharding and Cluster Reconfiguration for ElastiCache for Redis—Redis (cluster mode enabled) (p. 226)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Online Resharding and Shard Rebalancing for ElastiCache for Redis—Redis (cluster mode enabled) (p. 226)</td>
</tr>
</tbody>
</table>

**Topics**

API Version 2015-02-02
- Scaling Memcached (p. 201)
- Scaling Single-Node Redis (cluster mode disabled) Clusters (p. 204)
- Scaling Redis (cluster mode disabled) Clusters with Replica Nodes (p. 213)
- Scaling for Amazon ElastiCache for Redis—Redis (cluster mode enabled) (p. 225)
Memcached clusters are composed of 1 to 20 nodes. Scaling a Memcached cluster out and in is as easy as adding or removing nodes from the cluster.

If you need more than 20 nodes in a Memcached cluster, or more than 100 nodes total in 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 starts out empty unless your application populates it.

Topics
  • Scaling Memcached Horizontally (p. 201)
  • Scaling Memcached Vertically (p. 202)

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. 184)
  • Removing Nodes from a Cluster (p. 190)

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. 88).

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. 123). If you do not use auto discovery, each time you change the number of nodes in your Memcached cluster you must update the endpoints in your application.
Scaling Memcached Vertically

When you scale your Memcached cluster up or down, you must create a new cluster. Memcached clusters always start out empty unless your application populates it.

**Important**

If you are scaling down to a smaller node type, be sure that the smaller node type is adequate for your data and overhead. For more information, see *Choosing Your Node Size for Memcached Clusters* (p. 99).

**Topics**

- Scaling Memcached Vertically (Console) (p. 202)
- Scaling Memcached Vertically (AWS CLI) (p. 202)
- Scaling Memcached Vertically (ElastiCache API) (p. 202)

**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. 157).
2. In your application, update the endpoints to the new cluster's endpoints. For more information, see *Finding a Memcached Cluster's Endpoints (Console)* (p. 63).
3. Delete the old cluster. For more information, see *Deleting a Cluster (Console)* (p. 197).

**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. 168).
2. In your application, update the endpoints to the new cluster's endpoints. For more information, see *Finding Endpoints (AWS CLI)* (p. 69).
3. Delete the old cache cluster. For more information, see *Deleting a Cache Cluster (AWS CLI)* (p. 197).

**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. 170).
2. In your application, update the endpoints to the new cache cluster's endpoints. For more information, see *Finding Endpoints (ElastiCache API)* (p. 73).
3. Delete the old cache cluster. For more information, see Deleting a Cache Cluster (ElastiCache API) (p. 198).
Scaling Single-Node 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. 77).

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. 159).

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. 223) or Decreasing Read Capacity (p. 224).

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. 235).

Important
If your parameter group uses reserved-memory to set aside memory for Redis overhead, before you begin scaling be sure that you have a custom parameter group that reserves the correct amount of memory for your new node type. Alternatively, you can modify a custom parameter group so that it uses reserved-memory-percent and use that parameter group for your new cluster.

If you're using reserved-memory-percent, doing this is not necessary. For more information, see Managing Reserved Memory (Redis) (p. 79).

Topics
- Scaling Up Single-Node Redis (cluster mode disabled) Clusters (p. 205)
- Scaling Down Single-Node Redis Clusters (p. 210)
Scaling Up Single-Node Redis (cluster mode disabled) 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. 56).

### Blocked Redis operations

<table>
<thead>
<tr>
<th>Pending Operations</th>
<th>Blocked Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale up</td>
<td>Immediate engine upgrade</td>
</tr>
<tr>
<td>Engine upgrade</td>
<td>Immediate scale up</td>
</tr>
<tr>
<td>Scale up and engine upgrade</td>
<td>Immediate scale up</td>
</tr>
<tr>
<td></td>
<td>Immediate engine upgrade</td>
</tr>
</tbody>
</table>

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 chosen (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.

**Important**

If your parameter group uses `reserved-memory` to set aside memory for Redis overhead, before you begin scaling be sure that you have a custom parameter group that reserves the correct amount of memory for your new node type. Alternatively, you can modify a custom parameter group so that it uses `reserved-memory-percent` and use that parameter group for your new cluster.

If you're using `reserved-memory-percent`, doing this is not necessary.
Scaling Up Single-Node Redis (cluster mode disabled) 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)

2. From the 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:
   a. 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.
   b. If you're using reserved-memory to manage your memory, from the Parameter Group list, choose the custom parameter group that reserves the correct amount of memory for your new node type.
6. If you want to perform the scale up process right away, choose the Apply immediately box. If the Apply immediately box is not chosen, the scale-up process is performed during this cluster's next maintenance window.
7. Choose Modify.
   If you chose 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, macOS, or Unix:

   ```bash
   aws elasticache list-allowed-node-type-modifications \  
   --cache-cluster-id my-cache-cluster-id
   ```

   For Windows:

   ```bash
   aws elasticache list-allowed-node-type-modifications ^  
   --cache-cluster-id my-cache-cluster-id
   ```
Output from the above command looks something like this (JSON format).

```
{
  "ScaleUpModifications": [
    "cache.m3.2xlarge",
    "cache.m3.large",
    "cache.m3.xlarge",
    "cache.m4.10xlarge",
    "cache.m4.2xlarge",
    "cache.m4.4xlarge",
    "cache.m4.large",
    "cache.m4.xlarge",
    "cache.r3.2xlarge",
    "cache.r3.4xlarge",
    "cache.r3.8xlarge",
    "cache.r3.large",
    "cache.r3.xlarge"
  ]
}
```

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.
- `--cache-parameter-group-name` – [Optional] Use this parameter if you are using reserved-memory to manage your cluster's reserved memory. Specify a custom cache parameter group that reserves the correct amount of memory for your new node type. If you are using reserved-memory-percent you can omit this parameter.
- `--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, macOS, or Unix:

```
aws elasticache modify-cache-cluster \
  --cache-cluster-id my-redis-cache-cluster \
  --cache-node-type cache.m2.xlarge \
  --cache-parameter-group-name redis32-m2-xl \
  --apply-immediately
```

For Windows:

```
aws elasticache modify-cache-cluster ^
  --cache-cluster-id my-redis-cache-cluster ^
  --cache-node-type cache.m2.xlarge ^
  --cache-parameter-group-name redis32-m2-xl ^
  --apply-immediately
```

Output from the above command looks something like this (JSON format).

```
{
  "CacheCluster": {
```
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
   ```

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.
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.
- **CacheParameterGroupName** – [Optional] Use this parameter if you are using reserved-memory to manage your cluster's reserved memory. Specify a custom cache parameter group that reserves the correct amount of memory for your new node type. If you are using reserved-memory-percent you can omit this parameter.
- **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
&CacheParameterGroupName redis32-m2-xl
&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 walk 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. 77).

Topics
- Scaling Down a Single-Node Redis Cluster (Console) (p. 210)
- Scaling Down a Single-Node Redis Cache Cluster (AWS CLI) (p. 211)
- Scaling Down a Single-Node Redis (cluster mode disabled) Cache Cluster (ElastiCache API) (p. 211)

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.

Important
If your parameter group uses reserved-memory to set aside memory for Redis overhead, before you begin scaling be sure that you have a custom parameter group that reserves the correct amount of memory for your new node type. Alternatively, you can modify a custom parameter group so that it uses reserved-memory-percent and use that parameter group for your new cluster.

If you're using reserved-memory-percent, doing this is not necessary.
For more information, see Managing Reserved Memory (Redis) (p. 79).

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. 77).
2. If your parameter group uses reserved-memory to set aside memory for Redis overhead, ensure that you have a custom parameter group to set aside the correct amount of memory for your new node type.

   Alternatively, you can modify your custom parameter group to use reserved-memory-percent. For more information, see Managing Reserved Memory (Redis) (p. 79).
4. Take a snapshot of the cluster. For details on how to take a snapshot, see Creating a Manual Backup (Console) (p. 297).
5. Restore from this snapshot specifying the new node type for the new cluster and, if necessary, a parameter group to reserve the correct amount of memory. For more information, see Restoring From a Backup (Console) (p. 318).

   Alternatively, you can launch a new cluster using the new node type and parameter group, and seeding it from the snapshot. For more information, see Seeding a New Cluster with an Externally Created Backup (Redis) (p. 320).
6. In your application, update the endpoints to the new cluster's endpoints. For more information, see Finding a Redis (cluster mode disabled) Cluster's Endpoints (Console) (p. 65).
7. Delete the old cluster. For more information, see Deleting a Cluster (Console) (p. 197).
If you no longer need it, delete the snapshot. For more information, see Deleting a Backup
(Console) (p. 326).

Tip
If you don't mind your cluster being unavailable while it's being created or restored, you can eliminate
the need to update the endpoints in your application. To do so, delete the old cluster right after
taking the snapshot and reuse 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. 77).
2. If your parameter group uses reserved-memory to set aside memory for Redis overhead, ensure
   that you have a custom parameter group to set aside the correct amount of memory for your new
   node type.

   Alternatively, you can modify your custom parameter group to use reserved-memory-percent.
   For more information, see Managing Reserved Memory (Redis) (p. 79).
3. Create a snapshot of your existing Redis cache cluster. For instructions, see Creating a Manual
   Backup (AWS CLI) (p. 298).
4. Restore from the snapshot using the new, smaller node type as the cache cluster's node type, and,
   if needed, the new parameter group. For more information, see Restoring From a Backup (AWS
   CLI) (p. 319).
5. 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. 69).
6. Delete your old cache cluster. For more information, see Deleting a Cache Cluster (AWS
   CLI) (p. 197).
7. If you no longer need it, delete the snapshot. For more information, see Deleting a Backup (AWS
   CLI) (p. 326).

Tip
If you don't mind your cache cluster being unavailable while it's being created or restored, you can eliminate
the need to update the endpoints in your application. To do so, delete the old cache cluster right after
taking the snapshot and reuse 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. 77).
2. If your parameter group uses reserved-memory to set aside memory for Redis overhead, ensure
   that you have a custom parameter group to set aside the correct amount of memory for your new
   node type.
Alternatively, you can modify your custom parameter group to use reserved-memory-percent. For more information, see *Managing Reserved Memory (Redis)* (p. 79).

3. Create a snapshot of your existing Redis cache cluster. For instructions, see *Creating a Manual Backup (ElastiCache API)* (p. 301).

4. Restore from the snapshot using the new, smaller node type as the cache cluster's node type, and, if needed, the new parameter group. For more information, see *Restoring From a Backup (ElastiCache API)* (p. 319).

5. 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. 73).

6. Delete your old cache cluster. For more information, see *Deleting a Cache Cluster (ElastiCache API)* (p. 198).

7. If you no longer need it, delete the snapshot. For more information, see *Deleting a Backup (ElastiCache API)* (p. 326).

**Tip**

If you don't mind your cache cluster being unavailable while it's being created or restored, you can eliminate the need to update the endpoints in your application. To do so, delete the old cache cluster right after taking the snapshot and reuse the old cache cluster's name for the new cache cluster.
Scaling Redis (cluster mode disabled) 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 six 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) clusters support partitioning your data across up to 15 shards.

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. 77).

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. 235)
- Replication: Redis (cluster mode disabled) vs. Redis (cluster mode enabled) (p. 237)
- Replication: Multi-AZ with Automatic Failover (Redis) (p. 240)
- Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 77)

**Topics**

- Scaling Up Redis Clusters with Replicas (p. 214)
- Scaling Down Redis Clusters with Replicas (p. 220)
- Increasing Read Capacity (p. 223)
- Decreasing Read Capacity (p. 224)
Scaling Up Redis Clusters with Replicas

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 node. Reads from the replicas continue until step 5 when they are briefly interrupted while ElastiCache switches you from your current replicas to the new replicas.
2. Launches a new Redis replication group using the new node type.
3. Copies all the data from the current primary node to the new primary node.
4. Sync the new read replicas with the new primary node.
5. Updates the DNS entries so they point to the new nodes. Because of this you don't have to update the endpoints in your application.

   **Important**
   
   Reads from read replica nodes are interrupted while ElastiCache switches you from your current replicas to the new replicas.

6. Reinstates reads from and writes to the new primary node.
7. Deletes the old cluster (CLI/API: replication group).

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

<table>
<thead>
<tr>
<th>Pending Operations</th>
<th>Blocked Operations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scale up</td>
<td>Immediate engine upgrade</td>
</tr>
<tr>
<td>Engine upgrade</td>
<td>Immediate scale up</td>
</tr>
<tr>
<td>Scale up and engine upgrade</td>
<td>Immediate scale up</td>
</tr>
<tr>
<td></td>
<td>Immediate engine upgrade</td>
</tr>
</tbody>
</table>

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 chosen (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.

**Important**

If your parameter group uses `reserved-memory` to set aside memory for Redis overhead, before you begin scaling be sure that you have a custom parameter group that reserves the correct amount of memory for your new node type. Alternatively, you can modify a custom...
parameter group so that it uses reserved-memory-percent and use that parameter group for your new cluster.
If you're using reserved-memory-percent, doing this is not necessary. For more information, see Managing Reserved Memory (Redis) (p. 79).

**Scaling Up a Redis Cluster with Replicas (Console)**

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.

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.

**To scale up Redis cluster with replicas (console)**

2. From the 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. In the Modify Cluster wizard:
   a. 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.
   b. If you're using reserved-memory to manage your memory, from the Parameter Group list, choose the custom parameter group that reserves the correct amount of memory for your new node type.
6. If you want to perform the scale-up process right away, choose the Apply immediately check box. If the Apply immediately check box is left not chosen, 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 can 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.
Scaling Up Redis Clusters with Replicas

For Linux, macOS, or Unix:

```
aws elasticache list-allowed-node-type-modifications \ 
  --replication-group-id my-repl-group
```

For Windows:

```
aws elasticache list-allowed-node-type-modifications ^ \ 
  --replication-group-id my-repl-group
```

Output from this operation looks something like this (JSON format).

```
{
  "ScaleUpModifications": [
    "cache.m3.2xlarge",
    "cache.m3.large",
    "cache.m3.xlarge",
    "cache.m4.10xlarge",
    "cache.m4.2xlarge",
    "cache.m4.4xlarge",
    "cache.m4.large",
    "cache.m4.xlarge",
    "cache.r3.2xlarge",
    "cache.r3.4xlarge",
    "cache.r3.8xlarge",
    "cache.r3.large",
    "cache.r3.xlarge"
  ]
}
```

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.
- **--cache-parameter-group-name** – [Optional] Use this parameter if you are using reserved-memory to manage your cluster's reserved memory. Specify a custom cache parameter group that reserves the correct amount of memory for your new node type. If you are using reserved-memory-percent you can omit this parameter.
- **--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, macOS, or Unix:

```
aws elasticache modify-replication-group \ 
  --replication-group-id my-repl-group \ 
  --cache-node-type cache.m3.2xlarge \ 
  --cache-parameter-group-name redis32-m3-2xl \ 
  --apply-immediately
```

For Windows:
aws elasticache modify-replication-group
   --replication-group-id my-repl-group
   --cache-node-type cache.m3.2xlarge
   --cache-parameter-group-name redis32-m3-2xl
   --apply-immediately

Output from this command looks something like this (JSON format).

```json
{
   "ReplicationGroup": {
      "Status": "available",
      "Description": "Some description",
      "NodeGroups": [
         {
            "Status": "available",
            "NodeGroupMembers": [
               {
                  "CurrentRole": "primary",
                  "PreferredAvailabilityZone": "us-west-2b",
                  "CacheNodeId": "0001",
                 "ReadEndpoint": {
                     "Port": 6379,
                     "Address": "my-repl-group-001.8fdx4s.0001.usw2.cache.amazonaws.com"
               },
               "CacheClusterId": "my-repl-group-001"
            },
            {
               "CurrentRole": "replica",
               "PreferredAvailabilityZone": "us-west-2c",
               "CacheNodeId": "0001",
               "ReadEndpoint": {
                  "Port": 6379,
                  "Address": "my-repl-group-002.8fdx4s.0001.usw2.cache.amazonaws.com"
               },
               "CacheClusterId": "my-repl-group-002"
            }
         ],
         "NodeGroupId": "0001",
         "PrimaryEndpoint": {
            "Port": 6379,
            "Address": "my-repl-group.8fdx4s.ng.0001.usw2.cache.amazonaws.com"
         },
         "ReplicationGroupId": "my-repl-group",
         "SnapshotRetentionLimit": 1,
         "AutomaticFailover": "disabled",
         "SnapshotWindow": "12:00-13:00",
         "SnapshottingClusterId": "my-repl-group-002",
         "MemberClusters": [
            "my-repl-group-001",
            "my-repl-group-002"
         ],
         "PendingModifiedValues": {}
      }
   }
}
```

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, macOS, or Unix:
   ```bash
   aws elasticache describe-replication-group
   --replication-group-id my-replication-group
   ```

   For Windows:
   ```bash
   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=MyReplGroup
   &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 `ModifyReplicationGroup` 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.

• CacheParameterGroupName – [Optional] Use this parameter if you are using reserved-memory to manage your cluster’s reserved memory. Specify a custom cache parameter group that reserves the correct amount of memory for your new node type. If you are using reserved-memory-percent you can omit this parameter.

• ApplyImmediately – Set to true to cause the scale-up process to be applied immediately. To postpone the scale-up process to the next maintenance window, use ApplyImmediately=false.

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.

For more information, see DescribeReplicationGroups in the Amazon ElastiCache API Reference.
Scaling Down Redis Clusters with Replicas

The following sections walk 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. 77).

**Important**

If your parameter group uses `reserved-memory` to set aside memory for Redis overhead, before you begin scaling be sure that you have a custom parameter group that reserves the correct amount of memory for your new node type. Alternatively, you can modify a custom parameter group so that it uses `reserved-memory-percent` and use that parameter group for your new cluster.

If you're using `reserved-memory-percent`, doing this is not necessary. For more information, see Managing Reserved Memory (Redis) (p. 79).

**Topics**

- Scaling Down a Redis Cluster with Replicas (Console) (p. 220)
- Scaling Down a Redis Replication Group (AWS CLI) (p. 221)
- Scaling Down a Redis Replication Group (ElastiCache API) (p. 221)

**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. 77).
2. If your parameter group uses `reserved-memory` to set aside memory for Redis overhead, ensure that you have a custom parameter group to set aside the correct amount of memory for your new node type.
   
   Alternatively, you can modify your custom parameter group to use `reserved-memory-percent`. For more information, see Managing Reserved Memory (Redis) (p. 79).
4. Take a snapshot of the cluster's primary node. For details on how to take a snapshot, see Creating a Manual Backup (Console) (p. 297).
5. Restore from this snapshot specifying the new node type for the new cluster. For more information, see Restoring From a Backup (Console) (p. 318).
   
   Alternatively, you can launch a new cluster using the new node type and seeding it from the snapshot. For more information, see Seeding a New Cluster with an Externally Created Backup (Redis) (p. 320).
6. In your application, update the endpoints to the new cluster's endpoints. For more information, see Finding a Redis (cluster mode disabled) Cluster's Endpoints (Console) (p. 65).
7. Delete the old cluster. For more information, see Deleting a Replication Group (Console) (p. 286).
8. If you no longer need it, delete the snapshot. For more information, see Deleting a Backup (Console) (p. 326).
Tip
If you don't mind being unable to use your replication group while it's being created or restored, you can eliminate the need to update the endpoints in your application. To do so, delete the old cluster right after taking the snapshot and reuse 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 Choosing Your Node Size for Redis Clusters (p. 100).

2. If your parameter group uses reserved-memory to set aside memory for Redis overhead, ensure that you have a custom parameter group to set aside the correct amount of memory for your new node type.

   Alternatively, you can modify your custom parameter group to use reserved-memory-percent. For more information, see Managing Reserved Memory (Redis) (p. 79).

3. Create a snapshot of your existing Redis node. For instructions, see Creating a Manual Backup (AWS CLI) (p. 298).

4. Restore from the snapshot using the new, smaller node type as the new node type and, if needed, the new parameter group. For more information, see Restoring From a Backup (AWS CLI) (p. 319).

5. 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. 70).

6. Delete your old replication group. For more information, see Deleting a Replication Group (AWS CLI) (p. 286).

7. If you no longer need it, delete the snapshot. For more information, see Deleting a Backup (AWS CLI) (p. 326).

Tip
If you don't mind being unable to use your replication group while it's being created or restored, you can eliminate the need to update the endpoints in your application. To do so, delete the old replication group right after taking the snapshot and reuse 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 Choosing Your Node Size for Redis Clusters (p. 100).

2. If your parameter group uses reserved-memory to set aside memory for Redis overhead, ensure that you have a custom parameter group to set aside the correct amount of memory for your new node type.

   Alternatively, you can modify your custom parameter group to use reserved-memory-percent. For more information, see Managing Reserved Memory (Redis) (p. 79).

3. Create a snapshot of your existing Redis cache cluster. For instructions, see Creating a Manual Backup (ElastiCache API) (p. 301).
4. Restore from the snapshot using the new, smaller node type as the new node type and, if needed, the new parameter group. For more information, see Restoring From a Backup (ElastiCache API) (p. 319).

5. In your application, update the endpoints to the new cache cluster's endpoints. For more information, see Finding Endpoints (ElastiCache API) (p. 73).

6. Delete your old replication group. For more information, see Deleting a Replication Group (ElastiCache API) (p. 286).

7. If you no longer need it, delete the snapshot. For more information, see Deleting a Backup (ElastiCache API) (p. 326).

**Tip**

If you don't mind being unable to use your replication group while it's being created or restored, you can eliminate the need to update the endpoints in your application. To do so, delete the old replication group right after taking the snapshot and reuse 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.

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. 287).
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. 284).

For more information, see Deleting a Read Replica (p. 292).
Scaling for Amazon ElastiCache for Redis—Redis (cluster mode enabled)

As demand on your clusters changes, you might decide to improve performance or reduce costs by changing the number of shards in your Redis (cluster mode enabled) cluster. We recommend using online horizontal scaling to do so, because it allows your cluster to continue serving requests during the scaling process.

Conditions under which you might decide to rescale your cluster include the following:

- **Memory pressure:**
  
  If the nodes in your cluster are under memory pressure, you might decide to scale out so that you have more resources to better store data and serve requests.
  
  You can determine whether your nodes are under memory pressure by monitoring the following metrics: `FreeableMemory`, `SwapUsage`, and `BytesUseForCache`.

- **CPU or network bottleneck:**
  
  If latency/throughput issues are plaguing your cluster, you might need to scale out to resolve the issues.
  
  You can monitor your latency and throughput levels by monitoring the following metrics: `CPUUtilization`, `NetworkBytesIn`, `NetworkBytesOut`, `CurrConnections`, and `NewConnections`.

- **Your cluster is over-scaled:**
  
  Current demand on your cluster is such that scaling in doesn't hurt performance and reduces your costs.
  
  You can monitor your cluster's use to determine whether or not you can safely scale in using the following metrics: `FreeableMemory`, `SwapUsage`, `BytesUseForCache`, `CPUUtilization`, `NetworkBytesIn`, `NetworkBytesOut`, `CurrConnections`, and `NewConnections`.

**Performance Impact of Scaling**

When you scale using the offline process, your cluster is offline for a significant portion of the process and thus unable to serve requests. When you scale using the online method, because scaling is a compute-intensive operation, there is some degradation in performance, nevertheless, your cluster continues to serve requests throughout the scaling operation. How much degradation you experience depends upon your normal CPU utilization and your data.

There are two ways to scale your Redis (cluster mode enabled) cluster; offline and online. Whichever you choose, you can do the following:

- Change the number of node groups (shards) in the replication group by adding or removing node groups.
- Configure the slots in your new cluster differently than they were in the old cluster. Offline method only.
- Change the node type of the cluster's nodes. If you are changing to a smaller node type, be sure that the new node size has sufficient memory for your data and Redis overhead. Offline method only.

For more information, see *Choosing Your Node Size (p. 99)*.
Offline Resharing and Cluster Reconfiguration for ElastiCache for Redis—Redis (cluster mode enabled)

The main advantage you get from offline shard reconfiguration is that you can do more than merely add or remove shards from your replication group. When you reshard offline, in addition to changing the number of shards in your replication group, you can do the following:

- Change the node type of your replication group.
- Specify the Availability Zone for each node in the replication group.
- Upgrade to a newer engine version.
- Specify the number of replica nodes in each shard independently.
- Specify the keyspace for each shard.

The main disadvantage of offline shard reconfiguration is that your cluster is offline beginning with the restore portion of the process and continuing until you update the endpoints in your application. The length of time that your cluster is offline varies with the amount of data in your cluster.

**To reconfigure your shards Redis (cluster mode enabled) cluster offline**

1. Create a manual backup of your existing Redis cluster. For more information, see [Making Manual Backups](#).
2. Create a new cluster by restoring from the backup. For more information, see [Restoring From a Backup with Optional Cluster Resizing](#).
3. Update the endpoints in your application to the new cluster's endpoints. For more information, see [Finding Your ElastiCache Endpoints](#).

Online Resharing and Shard Rebalancing for ElastiCache for Redis—Redis (cluster mode enabled)

By using online resharding and shard rebalancing with Amazon ElastiCache for Redis, you can scale your ElastiCache for Redis (cluster mode enabled) dynamically with no downtime. This approach means that your cluster can continue to serve requests even while scaling or rebalancing is in process.

You can do the following:

- **Scale out** – Increase read and write capacity by adding shards (node groups) to your Redis (cluster mode enabled) cluster (replication group).

  If you add one or more shards to your replication group, the number of nodes in each new shard is the same as the number of nodes in the smallest of the existing shards.

- **Scale in** – Reduce read and write capacity, and thereby costs, by removing shards from your Redis (cluster mode enabled) cluster.

- **Rebalance** – Move the keyspaces among the shards in your ElastiCache for Redis (cluster mode enabled) cluster so they are as equally distributed among the shards as possible.
You can't do the following:

- **Scale up/down**: Change your node type. To do this, you must use the offline process.
- **Upgrade your engine**: Change your engine version to a newer version. To do this, you must use the offline process.
- **Configure shards independently**:
  - You can't specify the number of nodes in each shard independently. To do this, you must use the offline process.
  - You can't specify the keyspace for shards independently. To do this, you must use the offline process.

Currently, the following limitations apply to ElastiCache for Redis online resharding and rebalancing:

- These processes require Redis engine version 3.2.10 or newer. For information on upgrading your engine version, see Upgrading Engine Versions (p. 54).
- There are limitations with slots or keyspaces and large items:
  - If any of the keys in a shard contain a large item, that key isn't be migrated to a new shard when scaling out or rebalancing. This functionality can result in unbalanced shards.
  - If any of the keys in a shard contain a large item (items greater than 256 MB after serialization), that shard isn't deleted when scaling in. This functionality can result in some shards not being deleted.
  - When scaling out, the number of nodes in any new shards equals the number of nodes in the smallest existing shard.
  - When scaling out, any tags that are common to all existing shards are copied to the new shards.

For more information, see Best Practices: Online Resharding (p. 89).

You can horizontally scale or rebalance your ElastiCache for Redis (cluster mode enabled) clusters using the AWS Management Console, the AWS CLI, and the ElastiCache API.

# Adding Shards with Online Resharding

You can add shards to your Redis (cluster mode enabled) cluster using the AWS Management Console, AWS CLI, or ElastiCache API. When you add shards to a Redis (cluster mode enabled) cluster, any tags on the existing shards are copied over to the new shards.

**Topics**

- Adding Shards (Console) (p. 227)
- Adding Shards (AWS CLI) (p. 228)
- Adding Shards (ElastiCache API) (p. 229)

## Adding Shards (Console)

You can use the AWS Management Console to add one or more shards to your Redis (cluster mode enabled) cluster. The following procedure describes the process.

**To add shards to your Redis (cluster mode enabled) cluster**

2. From the navigation pane, choose Redis.
3. Locate and choose the name of the Redis (cluster mode enabled) cluster that you want to add shards to.
Tip
Redis (cluster mode enabled) clusters have a value of 1 or greater in the Shards column.

4. Choose Add shard.
   a. For Number of shards to be added, choose the number of shards you want added to this cluster.
   b. For Availability zone(s), choose either No preference or Specify availability zones.
   c. If you chose Specify availability zones, for each node in each shard, select the node's Availability Zone from the list of Availability Zones.
   d. Choose Add.

Adding Shards (AWS CLI)

The following process describes how to reconfigure the shards in your Redis (cluster mode enabled) cluster by adding shards using the AWS CLI.

Use the following parameters with modify-replication-group-shard-configuration.

Parameters

- --apply-immediately – Required. Specifies the shard reconfiguration operation is to be started immediately.
- --replication-group-id – Required. Specifies which replication group (cluster) the shard reconfiguration operation is to be performed on.
- --node-group-count – Required. Specifies the number of shards (node groups) to exist when the operation is completed. When adding shards, the value of --node-group-count must be greater than the current number of shards.

Optionally, you can specify the Availability Zone for each node in the replication group using --resharding-configuration.

- --resharding-configuration – Optional. A list of preferred Availability Zones for each node in each shard in the replication group. Use this parameter only if the value of --node-group-count is greater than the current number of shards. If this parameter is omitted when adding shards, Amazon ElastiCache selects the Availability Zones for the new nodes.

The following example reconfigures the keyspaces over four shards in the Redis (cluster mode enabled) cluster my-cluster. The example also specifies the Availability Zone for each node in each shard. The operation begins immediately.

Example - Adding Shards

For Linux, macOS, or Unix:

```bash
aws elasticache modify-replication-group-shard-configuration \
  --replication-group-id my-cluster \
  --node-group-count 4 \
  --resharding-configuration \n    "PreferredAvailabilityZones=us-east-2a,us-east-2c" \n    "PreferredAvailabilityZones=us-east-2b,us-east-2a" \n    "PreferredAvailabilityZones=us-east-2c,us-east-2d" \n    "PreferredAvailabilityZones=us-east-2d,us-east-2c" \n  --apply-immediately
```

For Windows:
For more information, see modify-replication-group-shard-configuration in the AWS CLI documentation.

Adding Shards (ElastiCache API)

You can use the ElastiCache API to reconfigure the shards in your Redis (cluster mode enabled) cluster online by using the `ModifyReplicationGroupShardConfiguration` operation.

Use the following parameters with `ModifyReplicationGroupShardConfiguration`.

**Parameters**

- `ApplyImmediately=true` – Required. Specifies the shard reconfiguration operation is to be started immediately.
- `ReplicationGroupId` – Required. Specifies which replication group (cluster) the shard reconfiguration operation is to be performed on.
- `NodeGroupCount` – Required. Specifies the number of shards (node groups) to exist when the operation is completed. When adding shards, the value of `NodeGroupCount` must be greater than the current number of shards.

Optionally, you can specify the Availability Zone for each node in the replication group using `ReshardingConfiguration`.

- `ReshardingConfiguration` – Optional. A list of preferred Availability Zones for each node in each shard in the replication group. Use this parameter only if the value of `NodeGroupCount` is greater than the current number of shards. If this parameter is omitted when adding shards, Amazon ElastiCache selects the Availability Zones for the new nodes.

The following process describes how to reconfigure the shards in your Redis (cluster mode enabled) cluster by adding shards using the ElastiCache API.

**Example - Adding Shards**

The following example adds node groups to the Redis (cluster mode enabled) cluster `my-cluster`, so there are a total of four node groups when the operation completes. The example also specifies the Availability Zone for each node in each shard. The operation begins immediately.

```
aws elasticache modify-replication-group-shard-configuration ^
--replication-group-id my-cluster ^
--node-group-count 4 ^
--resharding-configuration ^
  "PreferredAvailabilityZones=us-east-2a,us-east-2c" ^
  "PreferredAvailabilityZones=us-east-2b,us-east-2a" ^
  "PreferredAvailabilityZones=us-east-2c,us-east-2d" ^
  "PreferredAvailabilityZones=us-east-2d,us-east-2c" ^
--apply-immediately
```

Example - Adding Shards

The following example adds node groups to the Redis (cluster mode enabled) cluster `my-cluster`, so there are a total of four node groups when the operation completes. The example also specifies the Availability Zone for each node in each shard. The operation begins immediately.

```
https://elasticache.us-east-2.amazonaws.com/
?Action=ModifyReplicationGroupShardConfiguration
 &ApplyImmediately=true
 &NodeGroupCount=4
 &ReplicationGroupId=my-cluster
 &ReshardingConfiguration.ReshardingConfiguration.1.PreferredAvailabilityZones.AvailabilityZone.1=us-east-2a
 &ReshardingConfiguration.ReshardingConfiguration.1.PreferredAvailabilityZones.AvailabilityZone.2=us-east-2c
 &ReshardingConfiguration.ReshardingConfiguration.2.PreferredAvailabilityZones.AvailabilityZone.1=us-east-2b
 ```
Removing Shards with Online Resharding

You can remove shards from your Redis (cluster mode enabled) cluster using the AWS Management Console, AWS CLI, or ElastiCache API.

**Topics**

- Removing Shards (Console) (p. 230)
- Removing Shards (AWS CLI) (p. 231)
- Removing Shards (ElastiCache API) (p. 232)

**Removing Shards (Console)**

The following process describes how to reconfigure the shards in your Redis (cluster mode enabled) cluster by removing shards using the AWS Management Console.

Before removing node groups (shards) from your replication group, ElastiCache makes sure that all your data fits in the remaining shards. If the data fits, the specified shards are deleted from the replication group as requested. If the data doesn’t fit in the remaining node groups, the process is terminated. Terminating the process leaves your replication group with the same node group configuration as before the request was made.

You can use the AWS Management Console to remove one or more shards from your Redis (cluster mode enabled) cluster. You cannot remove all the shards in a replication group. Instead, you must delete the replication group. For more information, see Deleting a Cluster with Replicas (p. 286). The following procedure describes the process for deleting one or more shards.

**To remove shards from your Redis (cluster mode enabled) cluster**

2. From the navigation pane, choose Redis.
3. Locate and choose the name of the Redis (cluster mode enabled) cluster you want to remove shards from.

   **Tip**
   Redis (cluster mode enabled) clusters have a value of 1 or greater in the **Shards** column.
4. From the list of shards, choose the box by the name of each shard that you want to delete.
5. Choose **Delete shard**.

### Removing Shards (AWS CLI)

The following process describes how to reconfigure the shards in your Redis (cluster mode enabled) cluster by removing shards using the AWS CLI.

**Important**

Before removing node groups (shards) from your replication group, ElastiCache makes sure that all your data fits in the remaining shards. If the data fits, the specified shards (--node-groups-to-remove) are deleted from the replication group as requested and their keyspaces mapped into the remaining shards. If the data doesn't fit in the remaining node groups, the process is terminated. Terminating the process leaves your replication group with the same node group configuration as before the request was made.

You can use the AWS CLI to remove one or more shards from your Redis (cluster mode enabled) cluster. You cannot remove all the shards in a replication group. Instead, you must delete the replication group. For more information, see [Deleting a Cluster with Replicas](p. 286).

Use the following parameters with `modify-replication-group-shard-configuration`.

**Parameters**

- **--apply-immediately** – Required. Specifies the shard reconfiguration operation is to be started immediately.
- **--replication-group-id** – Required. Specifies which replication group (cluster) the shard reconfiguration operation is to be performed on.
- **--node-group-count** – Required. Specifies the number of shards (node groups) to exist when the operation is completed. When removing shards, the value of `--node-group-count` must be less than the current number of shards.
- **--node-groups-to-remove** – Required when `--node-group-count` is less than the current number of node groups (shards). A list of shard (node group) IDs to remove from the replication group.

The following procedure describes the process for deleting one or more shards.

**Example - Removing Shards**

The following example removes two node groups from the Redis (cluster mode enabled) cluster *my-cluster*, so there are a total of two node groups when the operation completes. The keyspaces from the removed shards are distributed evenly over the remaining shards.

For Linux, macOS, or Unix:

```bash
aws elasticache modify-replication-group-shard-configuration \
  --replication-group-id my-cluster \
  --node-group-count 2 \
  --node-groups-to-remove "0002" "0003" \
  --apply-immediately
```

For Windows:

```bash
aws elasticache modify-replication-group-shard-configuration ^
  --replication-group-id my-cluster ^
  --node-group-count 2 ^
  --node-groups-to-remove "0002" "0003" ^
```
For more information, see modify-replication-group-shard-configuration in the AWS CLI documentation.

Removing Shards (ElastiCache API)

You can use the ElastiCache API to reconfigure the shards in your Redis (cluster mode enabled) cluster online by using the ModifyReplicationGroupShardConfiguration operation.

The following process describes how to reconfigure the shards in your Redis (cluster mode enabled) cluster by removing shards using the ElastiCache API.

Important
Before removing node groups (shards) from your replication group, ElastiCache makes sure that all your data fits in the remaining shards. If the data fits, the specified shards (NodeGroupsToRemove) are deleted from the replication group as requested and their keyspaces mapped into the remaining shards. If the data doesn't fit in the remaining node groups, the process is terminated. Terminating the process leaves your replication group with the same node group configuration as before the request was made.

You can use the ElastiCache API to remove one or more shards from your Redis (cluster mode enabled) cluster. You cannot remove all the shards in a replication group. Instead, you must delete the replication group. For more information, see Deleting a Cluster with Replicas (p. 286).

Use the following parameters with ModifyReplicationGroupShardConfiguration.

Parameters

- ApplyImmediately=true – Required. Specifies the shard reconfiguration operation is to be started immediately.
- ReplicationGroupId – Required. Specifies which replication group (cluster) the shard reconfiguration operation is to be performed on.
- NodeGroupCount – Required. Specifies the number of shards (node groups) to exist when the operation is completed. When removing shards, the value of NodeGroupCount must be less than the current number of shards.
- NodeGroupsToRemove – Required when --node-group-count is less than the current number of node groups (shards). A list of shard (node group) IDs to remove from the replication group.

The following procedure describes the process for deleting one or more shards.

Example - Removing Shards

The following example removes two node groups from the Redis (cluster mode enabled) cluster my-cluster, so there are a total of two node groups when the operation completes. The keyspaces from the removed shards are distributed evenly over the remaining shards.

```
https://elasticache.us-east-2.amazonaws.com/
?Action=ModifyReplicationGroupShardConfiguration
&ApplyImmediately=true
&NodeGroupCount=2
&ReplicationGroupId=my-cluster
&NodeGroupsToRemove.member.1=0002
&NodeGroupsToRemove.member.2=0003
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20171002T192317Z
&X-Amz-Credential=<credential>
```
Online Shard Rebalancing

You can rebalance shards in your Redis (cluster mode enabled) cluster using the AWS Management Console, AWS CLI, or ElastiCache API.

Topics
- Online Shard Rebalancing (Console) (p. 233)
- Online Shard Rebalancing (AWS CLI) (p. 233)
- Online Shard Rebalancing (ElastiCache API) (p. 234)

Online Shard Rebalancing (Console)

The following process describes how to reconfigure the shards in your Redis (cluster mode enabled) cluster by rebalancing shards using the AWS Management Console.

To rebalance the keyspaces among the shards on your Redis (cluster mode enabled) cluster

2. From the navigation pane, choose Redis.
3. Choose the name of the Redis (cluster mode enabled) cluster that you want to rebalance.
   - Tip
     - Redis (cluster mode enabled) clusters have a value of 1 or greater in the Shards column.
5. When prompted, choose Rebalance. You might see a message similar to this one:
   - Slots in the replication group are uniformly distributed. Nothing to do. (Service: AmazonElastiCache; Status Code: 400; Error Code: InvalidReplicationGroupState; Request ID: 2246cebd-9721-11e7-8d5b-e1b0f086c8cf). If you do, choose Cancel.

Online Shard Rebalancing (AWS CLI)

Use the following parameters with modify-replication-group-shard-configuration.

Parameters
- -apply-immediately – Required. Specifies the shard reconfiguration operation is to be started immediately.
- --replication-group-id – Required. Specifies which replication group (cluster) the shard reconfiguration operation is to be performed on.
- --node-group-count – Required. To rebalance the keyspaces across all shards in the cluster, this value must be the same as the current number of shards.

The following process describes how to reconfigure the shards in your Redis (cluster mode enabled) cluster by rebalancing shards using the AWS CLI.

Example - Rebalancing the Shards in a Cluster

The following example rebalances the slots in the Redis (cluster mode enabled) cluster my-cluster so that the slots are distributed as equally as possible. The value of --node-group-count (4) is the number of shards currently in the cluster.
Online Shard Rebalancing (ElastiCache API)

You can use the ElastiCache API to reconfigure the shards in your Redis (cluster mode enabled) cluster online by using the ModifyReplicationGroupShardConfiguration operation.

Use the following parameters with ModifyReplicationGroupShardConfiguration.

Parameters

- **ApplyImmediately**=true – Required. Specifies the shard reconfiguration operation is to be started immediately.
- ReplicationGroupId – Required. Specifies which replication group (cluster) the shard reconfiguration operation is to be performed on.
- NodeGroupCount – Required. To rebalance the keyspaces across all shards in the cluster, this value must be the same as the current number of shards.

The following process describes how to reconfigure the shards in your Redis (cluster mode enabled) cluster by rebalancing the shards using the ElastiCache API.

**Example - Rebalancing a Cluster**

The following example rebalances the slots in the Redis (cluster mode enabled) cluster `my-cluster` so that the slots are distributed as equally as possible. The value of `NodeGroupCount` (4) is the number of shards currently in the cluster.

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

```
?Action=ModifyReplicationGroupShardConfiguration
&ApplyImmediately=true
&NodeGroupCount=4
&ReplicationGroupId=my-cluster
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20171002T192317Z
&X-Amz-Credential=<credential>
```

For more information, see `ModifyReplicationGroupShardConfiguration` in the ElastiCache API Reference.
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 will 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 nodes contain replicate data of the group’s single read/write primary node. In this scenario, if one node fails for any reason, you do not lose all your data since it is replicated in one or more other nodes. Due to replication latency, some data may be lost if it is the primary read/write node that fails.

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 mode disabled) cluster has one shard and 1 to 5 replica nodes

If the cluster with replicas has Multi-AZ with automatic failover enabled and the primary node 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. 84).

Topics

- Redis Replication (p. 236)
- Replication: Redis (cluster mode disabled) vs. Redis (cluster mode enabled) (p. 237)
- Replication: Multi-AZ with Automatic Failover (Redis) (p. 240)
- How Synchronization and Backup are Implemented (p. 253)
- Creating a Redis Cluster with Replicas (p. 254)
- Viewing a Replication Group’s Details (p. 274)
- Finding Replication Group Endpoints (p. 279)
- Modifying a Cluster with Replicas (p. 284)
- Deleting a Cluster with Replicas (p. 286)
- Adding a Read Replica to a Redis Cluster (p. 287)
- Promoting a Read-Replica to Primary (p. 289)
- Deleting a Read Replica (p. 292)
Redis Replication

Redis 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.

Topics
- Redis (cluster mode disabled) (p. 236)
- Redis (cluster mode enabled) (p. 236)

Redis (cluster mode disabled)

A Redis (cluster mode disabled) cluster has a single shard, inside of which is a collection of Redis nodes; one primary read-write node and up to five secondary, read-only replica nodes. 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 node in the cluster. Applications can write only to the primary node. Read replicas improve read throughput and guard against data loss in cases of a node failure.

Redis (cluster mode disabled) cluster with a single shard and replica nodes

You can use Redis (cluster mode disabled) clusters with replica nodes to scale your Redis solution for ElastiCache to handle applications that are 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 new node. From that point on, whenever data is written to the primary, the changes are asynchronously propagated to all the read replicas.

To improve fault tolerance and reduce write down time, enable Multi-AZ with automatic failover for your Redis (cluster mode disabled) cluster with replicas. For more information, see Replication: Multi-AZ with Automatic Failover (Redis) (p. 240).

You can change the roles of the nodes within the Redis (cluster mode disabled) 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. 289).

Redis (cluster mode enabled)

A Redis (cluster mode enabled) cluster is comprised of from 1 to 15 shards (API/CLI: node groups). Each shard has a primary node and up to five read-only replica nodes. Each read replica in a shard maintains
a copy of the data from the shard's 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) cluster's configuration endpoint to connect with the nodes in the cluster. For more information, see Finding Your ElastiCache Endpoints (p. 62).

Redis (cluster mode enabled) cluster with multiple shards and replica nodes

All of the nodes in a Redis (cluster mode enabled) 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.

Multi-AZ with automatic failover is required for all Redis (cluster mode enabled) clusters. For more information, see Replication: Multi-AZ with Automatic Failover (Redis) (p. 240).

Currently, in Redis (cluster mode enabled), there are some limitations.

- You cannot manually promote any of the replica nodes to primary.
- Multi-AZ with automatic failover is required.
- The structure of a cluster, node type and number of nodes, can only be changed by restoring from a backup. For more information, see Restoring From a Backup with Optional Cluster Resizing (p. 317). The number of shards in a Redis (cluster mode enabled) cluster can be changed dynamically, while the cluster continues to serve read and write requests. For more information, see Online Resharding and Shard Rebalancing for ElastiCache for Redis—Redis (cluster mode enabled) (p. 226).

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 (API/CLI: replication groups). A Redis (cluster mode disabled) cluster always has a single shard (API/CLI: node group) 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**

<table>
<thead>
<tr>
<th>Feature</th>
<th>Redis (cluster mode disabled)</th>
<th>Redis (cluster mode enabled)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modifiable</td>
<td>Yes. Supports adding and deleting replica nodes, and scaling up node type.</td>
<td>Limited. Version 3.2.10 can add and remove shards dynamically. Other changes require creating a new cluster.</td>
</tr>
<tr>
<td>Data Partitioning</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Shards</td>
<td>1</td>
<td>1 to 15 The number of shards (API/CLI: node groups) is set when the cluster (API/CLI: replication group) is created.</td>
</tr>
<tr>
<td>Read replicas</td>
<td>0 to 5</td>
<td>0 to 5 per shard.</td>
</tr>
<tr>
<td><strong>Important</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If you have no replicas and the node fails, you will experience total data loss.</td>
<td></td>
</tr>
<tr>
<td>Snapshots (Backups)</td>
<td>Yes, creating a single .rdb file.</td>
<td>Yes, creating a unique .rdb file for each shard.</td>
</tr>
<tr>
<td>Restore</td>
<td>Yes, using a single .rdb file.</td>
<td>Yes. You can resize your cluster when restoring.</td>
</tr>
<tr>
<td>Supported by</td>
<td>All Redis versions</td>
<td>Redis 3.2 and following</td>
</tr>
<tr>
<td>Engine upgradeable</td>
<td>Yes</td>
<td>N/A</td>
</tr>
</tbody>
</table>
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 (cluster mode disabled) Clusters with Replica Nodes (p. 213).

Redis (cluster mode enabled) supports partitioning your data across up to 15 node groups. You can dynamically change the number of shards as your business needs change. 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. For information on scaling your partitions, see Scaling for Amazon ElastiCache for Redis—Redis (cluster mode enabled) (p. 225).

- **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. For more information, see Choosing Your Node Size for Redis Clusters (p. 100).

- **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.
Replication: Multi-AZ with Automatic Failover (Redis)

Enabling Amazon ElastiCache's Multi-AZ with automatic failover functionality on your Redis cluster (in the API and AWS 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. Multi-AZ with automatic failover is only supported on Redis clusters that support replication.

Automatic Failover Contents
- Automatic Failover Overview (p. 240)
- Notes on Redis Multi-AZ with Automatic Failover (p. 241)
- Failure Scenarios with Multi-AZ and Automatic Failover Responses (p. 242)
  - When Only the Primary Node Fails (p. 242)
  - When the Primary Node and Some Read Replicas Fail (p. 243)
  - When the Entire Cluster Fails (p. 244)
- Enabling Multi-AZ with Automatic Failover (p. 246)
  - Enabling Multi-AZ with Automatic Failover (Console) (p. 246)
    - Enabling Multi-AZ with Automatic Failover When Creating a Cluster Using the ElastiCache Console (p. 246)
    - Enabling Multi-AZ with Automatic Failover on an Existing Cluster (Console) (p. 246)
  - Enabling Multi-AZ with Automatic Failover (AWS CLI) (p. 246)
  - Enabling Multi-AZ with Automatic Failover (ElastiCache API) (p. 247)
- Testing Multi-AZ with Automatic Failover (p. 249)
  - Testing Automatic Failover Using the AWS Management Console (p. 249)
  - Testing Automatic Failover Using the AWS CLI (p. 250)
  - Testing Automatic Failover Using the ElastiCache API (p. 252)

Automatic Failover Overview

An ElastiCache Redis cluster that supports replication, consists of one to 15 shards, called node groups in the API and CLI. Each shard consists of a primary node and one to five read replica nodes. In certain situations, if your cluster is Multi-AZ enabled, ElastiCache automatically detects the primary node's failure, selects a read replica node, and promotes it to primary. These situations include certain types of planned maintenance, or the unlikely event of a primary node or Availability Zone failure. This failure detection and replica promotion ensure that you can resume writing to the new primary as soon as promotion is complete.

ElastiCache also propagates the Domain Name Service (DNS) name of the promoted replica. It does so because then if your application is writing to the primary endpoint, no endpoint change is required in your application. However, because you read from individual endpoints, you 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. This process is much faster than recreating and provisioning a new primary, which is the process if you don't enable Multi-AZ with automatic failover.

You can enable Multi-AZ with Automatic Failover using the ElastiCache Management Console, the AWS CLI, or the ElastiCache API.
Notes on Redis Multi-AZ with Automatic Failover

The following points should be noted for Redis Multi-AZ with Automatic Failover:

- 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 node types.
- Redis Multi-AZ with Automatic Failover is supported on T2 node types only if you are running Redis version 3.2.4 or later with cluster mode enabled.
- 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 choosing the replica to promote to primary, ElastiCache chooses the replica with the least replication lag (that is, the one that is most current).

Promoting read replicas to primary:

- You can only promote a read replica to primary when Multi-AZ with Automatic Failover is disabled. To promote a read replica node 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.
- You cannot disable Multi-AZ with Automatic Failover on Redis (cluster mode enabled) clusters. Therefore, you cannot manually promote a replica to primary on any Redis (cluster mode enabled) cluster.
- ElastiCache Multi-AZ with Automatic Failover and append-only file (AOF) are mutually exclusive. If you enable one, you cannot enable the other.
- When 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

- For Redis version 2.8.22 and later, external replicas are not permitted.
- For 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 reprovisioning 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 depending on the nature of the failure.

**Failure Scenarios**
- When Only the Primary Node Fails (p. 242)
- When the Primary Node and Some Read Replicas Fail (p. 243)
- When the Entire Cluster Fails (p. 244)

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 replacement read replica is created and provisioned in the same Availability Zone as the failed primary.

**Automatic Failover for a failed primary node**

What ElastiCache Multi-AZ with Automatic Failover does when only the primary node fails is the following:

1. The failed primary node is taken offline.
2. The read replica with the least replication lag is promoted to primary.
3. A replacement read replica is launched and provisioned.

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 name of the promoted replica.
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 because the DNS name 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 a Redis (cluster mode disabled) Cluster's Endpoints (Console) (p. 65)
- Finding the Endpoints for Replication Groups (AWS CLI) (p. 70)
- Finding Endpoints for Replication Groups (ElastiCache API) (p. 73)

### When the Primary Node and Some Read Replicas Fail

If the primary and at least one read replica fails, the available replica with the least replication lag is promoted to primary cluster. New read replicas are also created and provisioned in the same Availability Zones as the failed nodes and replica that was promoted to primary.

What ElastiCache Multi-AZ does when the primary node and some read replicas fail is the following:

1. The failed primary node and failed read replicas are taken offline.
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, because ElastiCache propagates the DNS name 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.
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 because the DNS name 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 a Redis (cluster mode disabled) Cluster's Endpoints (Console) (p. 65)
- Finding the Endpoints for Replication Groups (AWS CLI) (p. 70)
- Finding Endpoints for Replication Groups (ElastiCache API) (p. 73)

**When the Entire Cluster Fails**

If everything fails, all the nodes are recreated and provisioned in the same Availability Zones as the original nodes.

In this scenario, all the data in the cluster is lost due to the failure of every node in the cluster. This occurrence is rare.

What ElastiCache Multi-AZ does when the entire cluster fails is the following:

1. The failed primary node and read replicas are taken offline.
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.

   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, you don't need to make any endpoint changes in your application.
For information about finding the endpoints of a replication group, see the following topics:

- Finding a Redis (cluster mode disabled) Cluster's Endpoints (Console) (p. 65)
- Finding the Endpoints for Replication Groups (AWS CLI) (p. 70)
- Finding Endpoints for Replication Groups (ElastiCache API) (p. 73)

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 or CLI, replication group) using the ElastiCache console, AWS CLI, or the ElastiCache API.

You can enable Multi-AZ with Automatic Failover only on Redis (cluster mode disabled) clusters that have at least one available read replica. Multi-AZ with Automatic Failover is required on all Redis (cluster mode enabled) clusters, whether or not they have read replicas. Clusters without read replicas do not provide high availability or fault tolerance. For information about creating a cluster with replication, see Creating a Redis Cluster with Replicas (p. 254). For information about adding a read replica to a cluster with replication, see Adding a Read Replica to a Redis Cluster (p. 287).

Topics

- Enabling Multi-AZ with Automatic Failover (Console) (p. 246)
- Enabling Multi-AZ with Automatic Failover (AWS CLI) (p. 246)
- Enabling Multi-AZ with Automatic Failover (ElastiCache API) (p. 247)

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.

Multi-AZ with Automatic Failover is enabled by default and cannot be disabled on Redis (cluster mode enabled) clusters.

Enabling Multi-AZ with Automatic Failover When Creating a Cluster Using the ElastiCache Console

For more information on this process, see Creating a Redis (cluster mode disabled) Cluster (Console) (p. 159). 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)

For more information on this process, see Modifying a Cluster (Console) (p. 179).

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 redis12.

**Important**
The replication group redis12 must already exist and have at least one available read replica.

For Linux, macOS, or Unix:

```
aws elasticache modify-replication-group
   --replication-group-id redis12
   --automatic-failover-enabled
   --apply-immediately
```

For Windows:

```
aws elasticache modify-replication-group ^
   --replication-group-id redis12 ^
   --automatic-failover-enabled ^
   --apply-immediately ^
```
The JSON output from this command should look something like this.

```json
{
  "ReplicationGroup": {
    "Status": "modifying",
    "Description": "One shard, two nodes",
    "NodeGroups": [
      {
        "Status": "modifying",
        "NodeGroupMembers": [
          {
            "CurrentRole": "primary",
            "PreferredAvailabilityZone": "us-west-2b",
            "CacheNodeId": "0001",
            "ReadEndpoint": {
              "Port": 6379,
              "Address": "redis12-001.v5r9dc.0001.usw2.cache.amazonaws.com"
            },
            "CacheClusterId": "redis12-001"
          },
          {
            "CurrentRole": "replica",
            "PreferredAvailabilityZone": "us-west-2a",
            "CacheNodeId": "0001",
            "ReadEndpoint": {
              "Port": 6379,
              "Address": "redis12-002.v5r9dc.0001.usw2.cache.amazonaws.com"
            },
            "CacheClusterId": "redis12-002"
          }
        ],
        "NodeGroupId": "0001",
        "PrimaryEndpoint": {
          "Port": 6379,
          "Address": "redis12.v5r9dc.ng.0001.usw2.cache.amazonaws.com"
        }
      }
    ],
    "ReplicationGroupId": "redis12",
    "SnapshotRetentionLimit": 1,
    "AutomaticFailover": "enabling",
    "SnapshotWindow": "07:00-08:00",
    "SnapshottingClusterId": "redis12-002",
    "MemberClusters": [
      "redis12-001",
      "redis12-002"
    ],
    "PendingModifiedValues": {}
  }
}
```

For more information, see these topics in the AWS CLI Command Reference:

- `create-cache-cluster`
- `create-replication-group`
- `modify-replication-group` in the AWS CLI Command Reference.

**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 `redis12`. 
Note
To use this example, the replication group redis12 must already exist and have at least one available read replica.

https://elasticache.us-west-2.amazonaws.com/
?Action=ModifyReplicationGroup
&ApplyImmediately=true
&AutoFailover=true
&ReplicationGroupId=redis12
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20140401T192317Z
&X-Amz-Credential=<credential>

For more information, see these topics in the ElastiCache API Reference:

- CreateCacheCluster
- CreateReplicationGroup
- ModifyReplicationGroup
Testing Multi-AZ with Automatic Failover

After you enable Multi-AZ with Automatic Failover, you can test it using the ElastiCache console, the AWS CLI, and the ElastiCache API.

When testing, note the following:

• You can use this operation to test automatic failover on up to five shards (called node groups in the ElastiCache API and AWS CLI) in any rolling 24-hour period.

• If you call this operation on shards in different clusters (called replication groups in the API and CLI), you can make the calls concurrently.

• If you call this operation multiple times on different shards in the same Redis (cluster mode enabled) replication group, the first node replacement must complete before a subsequent call can be made.

• To determine whether the node replacement is complete you can check Events using the Amazon ElastiCache console, the AWS CLI, or the ElastiCache API. Look for the following automatic failover related events, listed here in order of occurrence:
  1. Replication group message: Test Failover API called for node group <node-group-id>
  2. Cache cluster message: Failover from master node <primary-node-id> to replica node <node-id> completed
  3. Replication group message: Failover from master node <primary-node-id> to replica node <node-id> completed
  4. Cache cluster message: Recovering cache nodes <node-id>
  5. Cache cluster message: Finished recovery for cache nodes <node-id>

For more information, see the following:

• Viewing ElastiCache Events (p. 460) in the ElastiCache User Guide
• DescribeEvents in the ElastiCache API Reference
• describe-events in the AWS CLI Command Reference.

Testing Automatic Failover

• Testing Automatic Failover Using the AWS Management Console (p. 249)
• Testing Automatic Failover Using the AWS CLI (p. 250)
• Testing Automatic Failover Using the ElastiCache API (p. 252)

Testing Automatic Failover Using the AWS Management Console

The following procedure walks you through testing automatic failover.

To test automatic failover

2. In the navigation pane, choose Redis.
3. From the list of Redis clusters, choose the box to the left of the cluster you want to test. This cluster must have at least one read replica node.
4. In the **Details** area, confirm that this cluster is Multi-AZ enabled. If the cluster is not Multi-AZ enabled, either choose a different cluster or modify this cluster to enable Multi-AZ. For more information, see *Modifying a Cluster (Console)* (p. 179).

5. For Redis (cluster mode disabled), choose the cluster's name.
   
   For Redis (cluster mode enabled), do the following:
   
   a. Choose the cluster's name.
   
   b. On the **Shards** page, for the shard (called node group in the API and CLI) on which you want to test failover, choose the shard's name.

6. On the Nodes page, choose **Failover Primary**.
7. Choose **Continue** to fail over the primary, or **Cancel** to cancel the operation and not fail over the primary node.

   During the failover process, the console continues to show the node's status as **available**. To track the progress of your failover test, choose **Events** from the console navigation pane. On the **Events** tab, watch for events that indicate your failover has started (**Test Failover API called**) and completed (**Recovery completed**).

### Testing Automatic Failover Using the AWS CLI

You can test automatic failover on any Multi-AZ with Automatic Failover enabled cluster using the AWS CLI operation `test-failover`.

**Parameters**

- `--replication-group-id` – Required. The replication group (on the console, cluster) that is to be tested.
- `--node-group-id` – Required. The name of the node group you want to test automatic failover on. You can test a maximum of five node groups in a rolling 24-hour period.

The following example uses the AWS CLI to test automatic failover on the node group `redis00-0003` in the Redis (cluster mode enabled) cluster `redis00`.

**Example Test Automatic Failover**

For Linux, macOS, or Unix:

```bash
aws elasticache test-failover
```
For Windows:

```bash
aws elasticache test-failover
--replication-group-id redis00
--node-group-id redis00-0003
```

Output from the preceeding command looks something like this.

```json
{
  "ReplicationGroup": {
    "Status": "available",
    "Description": "1 shard, 3 nodes (1 + 2 replicas)",
    "NodeGroups": [
      {
        "Status": "available",
        "NodeGroupMembers": [
          {
            "CurrentRole": "primary",
            "PreferredAvailabilityZone": "us-west-2c",
            "CacheNodeId": "0001",
            "ReadEndpoint": {
              "Port": 6379,
              "Address": "redis1x3-001.7ekv3t.0001.usw2.cache.amazonaws.com"
            },
            "CacheClusterId": "redis1x3-001"
          },
          {
            "CurrentRole": "replica",
            "PreferredAvailabilityZone": "us-west-2a",
            "CacheNodeId": "0001",
            "ReadEndpoint": {
              "Port": 6379,
              "Address": "redis1x3-002.7ekv3t.0001.usw2.cache.amazonaws.com"
            },
            "CacheClusterId": "redis1x3-002"
          },
          {
            "CurrentRole": "replica",
            "PreferredAvailabilityZone": "us-west-2b",
            "CacheNodeId": "0001",
            "ReadEndpoint": {
              "Port": 6379,
              "Address": "redis1x3-003.7ekv3t.0001.usw2.cache.amazonaws.com"
            },
            "CacheClusterId": "redis1x3-003"
          }
        ],
        "NodeGroupId": "0001",
        "PrimaryEndpoint": {
          "Port": 6379,
          "Address": "redis1x3.7ekv3t.ng.0001.usw2.cache.amazonaws.com"
        }
      }
    ],
    "ClusterEnabled": false,
    "ReplicationGroupId": "redis1x3",
    "SnapshotRetentionLimit": 1,
    "AutomaticFailover": "enabled",
    "SnapshotWindow": "11:30-12:30",
    "SnapshottingClusterId": "redis1x3-002"
  }
}
```
Testing Automatic Failover Using the ElastiCache API

You can test automatic failover on any cluster enabled with Multi-AZ with Automatic Failover using the ElastiCache API operation TestFailover.

**Parameters**

- **ReplicationGroupId** – Required. The replication group (on the console, cluster) that is to be tested.
- **NodeGroupId** – Required. The name of the node group you want to test automatic failover on. You can test a maximum of five node groups in a rolling 24-hour period.

The following example tests automatic failover on the node group `redis00-0003` in the replication group (on the console, cluster) `redis00`.

**Example Testing Automatic Failover**

```url
https://elasticache.us-west-2.amazonaws.com/
?Action=TestFailover
&NodeGroupId=redis00-0003
&ReplicationGroupId=redis00
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20140401T192317Z
&X-Amz-Credential=<credential>
```

To track the progress of your failover, use the ElastiCache API DescribeEvents operation.

For more information, see the following:

- **TestFailover** in the *ElastiCache API Reference*
- **DescribeEvents** in the *ElastiCache API Reference*
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. 253) and ElastiCache Backup and Restore (Redis) (p. 293).

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. 77).
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. 255)**

Use this option to leverage an existing single-node Redis (cluster mode disabled) cluster. You will specify this existing cluster as the primary node 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. See Creating a Cluster with Replicas Using an Available Redis (cluster mode disabled) Cluster (p. 255).

**Important**

You cannot create a Redis (cluster mode enabled) cluster using an existing cluster. To create a Redis (cluster mode enabled) cluster (API/CLI: replication group) using the ElastiCache console, see Creating a Redis (cluster mode enabled) Cluster (Console) (p. 163).

**Option 2: Creating a Redis Cluster with Replicas from Scratch (p. 260)**

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. See Creating a Redis Cluster with Replicas from Scratch (p. 260).
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) Cluster (Console) (p. 267).

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 Redis Cluster with Replicas from Scratch (p. 260).

Creating a Cluster with Replicas Using an Available Redis Cluster (Console)

See the topic Adding Nodes to a Cluster (Console) (p. 184).

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.

When using the AWS CLI you create a replication group specifying the available stand-alone node as the cluster's primary node, --primary-cluster-id and the number of nodes you want in the cluster using the CLI command, create-replication-group. Include the following parameters.

--replication-group-id

The name of the replication group you are creating. The value of this parameter is used as the basis for the names of the added nodes with a sequential 3-digit number added to the end of the --replication-group-id. For example, new-group-001.

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.

--num-cache-clusters

The total number of nodes you want in this clusters, including the primary node. This parameter has a maximum value of 6.

--primary-cluster-id

The name of the available Redis (cluster mode disabled) cluster that you want to be the primary node in this replication group.

If you want to enable in-transit or at-rest encryption on this cluster, add these parameters:
• --transit-encryption-enabled

If you enable in-transit encryption, the cluster must be created in a Amazon VPC and you must also include the parameter --cache-subnet-group.

• --auth-token with the customer specified string value for your AUTH token (password) needed to perform operations on this cluster.

• --at-rest-encryption-enabled

The following command creates the replication group `new-group` using the available Redis (cluster mode disabled) cluster `redis01` as the replication group’s primary node. It creates 2 new nodes which are read replicas. The settings of `redis01` (that is, parameter group, security group, node type, engine version, etc.) will be applied to all nodes in the replication group.

For Linux, macOS, or Unix:

```bash
aws elasticache create-replication-group 
   --replication-group-id new-group 
   --replication-group-description "demo cluster with replicas" 
   --num-cache-clusters 3 
   --primary-cluster-id redis01
```

For Windows:

```bash
aws elasticache create-replication-group ^
   --replication-group-id new-group ^
   --replication-group-description "demo cluster with replicas" ^
   --num-cache-clusters 3 ^
   --primary-cluster-id redis01
```

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, macOS, or Unix:

```bash
aws elasticache create-cache-cluster
   --cache-cluster-id my-replica01
   --replication-group-id my-repl-group
```

For Windows:

```bash
aws elasticache create-cache-cluster ^
   --cache-cluster-id my-replica01 ^
   --replication-group-id my-repl-group
```

Output from this command will look something like this.

```json
{
   "ReplicationGroup": {
      "Status": "creating",
      "Description": "demo cluster with replicas",
      "ClusterEnabled": false,
      "ReplicationGroupId": "new-group",
      "SnapshotRetentionLimit": 1,
      "AutomaticFailover": "disabled",
      "SnapshotWindow": "00:00-01:00",
      "SnapshottingClusterId": "redis01",
      "MemberClusters": [
         "new-group-001",
         "new-group-002",
         "redis01"
      ],
      "CacheNodeType": "cache.m4.large",
      "PendingModifiedValues": {}
   }
}
```

For additional information, see the AWS CLI topics:

- create-replication-group
- modify-replication-group

### Adding Replicas to a Stand-Alone Redis (cluster mode disabled) Cluster (ElastiCache API)

When using the ElastiCache API you create a replication group specifying the available stand-alone node as the cluster's primary node, `PrimaryClusterId` and the number of nodes you want in the cluster using the CLI command, `CreateReplicationGroup`. Include the following parameters.

**ReplicationGroupId**

The name of the replication group you are creating. The value of this parameter is used as the basis for the names of the added nodes with a sequential 3-digit number added to the end of the `ReplicationGroupId`. For example, `new-group-001`.

**Redis (cluster mode disabled) Replication Group naming constraints**

- Must contain from 1 to 20 alphanumeric characters or hyphens.
Creating a Cluster with Replicas Using an Existing Cluster

- Must begin with a letter.
- Cannot contain two consecutive hyphens.
- Cannot end with a hyphen.

**ReplicationGroupDescription**

Description of the cluster with replicas.

**NumCacheClusters**

The total number of nodes you want in this cluster, including the primary node. This parameter has a maximum value of 6.

**PrimaryClusterId**

The name of the available Redis (cluster mode disabled) cluster that you want to be the primary node in this cluster.

The following command creates the cluster with replicas new-group using the available Redis (cluster mode disabled) cluster redis01 as the replication group's primary node. It creates 2 new nodes which are read replicas. The settings of redis01 (that is, parameter group, security group, node type, engine version, etc.) will be applied to all nodes in the replication group.

```
https://elasticache.us-west-2.amazonaws.com/
?Action=CreateReplicationGroup
&Engine=redis
&EngineVersion=3.2.4
&ReplicationGroupDescription=Demo%20cluster%20with%20replicas
&ReplicationGroupId=new-group
&PrimaryClusterId=redis01
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

For additional information, see the ElastiCache APL topics:

- CreateReplicationGroup
- ModifyReplicationGroup

**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 Redis Cluster with Replicas 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. 237).

Topics

- Creating a Redis (cluster mode disabled) Cluster with Replicas from Scratch (p. 261)
- Creating a Redis (cluster mode enabled) Cluster with Replicas from Scratch (p. 267)
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.

Creating a Redis (cluster mode disabled) Cluster with Replicas from Scratch

- Creating a Redis (cluster mode disabled) Cluster with Replicas from Scratch (Console) (p. 261)
- Creating a Redis (cluster mode disabled) Cluster with Replicas from Scratch (AWS CLI) (p. 261)
- Creating a Redis (cluster mode disabled) Cluster with Replicas from Scratch (ElastiCache API) (p. 264)

Creating a Redis (cluster mode disabled) 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. 159). Specify at least one replica node.

Creating a Redis (cluster mode disabled) Cluster with Replicas 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
```

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:**
    
    - **T2 node types:** cache.t2.micro, cache.t2.small, cache.t2.medium
    - **M3 node types:** cache.m3.medium, cache.m3.large, cache.m3.xlarge, cache.m3.2xlarge
    - **M4 node types:** cache.m4.large, cache.m4.xlarge, cache.m4.2xlarge, cache.m4.4xlarge, cache.m4.10xlarge
  
  - **Previous generation: (not recommended)**
    - **T1 node types:** cache.t1.micro
    - **M1 node types:** cache.m1.small, cache.m1.medium, cache.m1.large, cache.m1.xlarge

- **Compute optimized:**

  - **Previous generation: (not recommended)**
    - **C1 node types:** cache.c1.xlarge

- **Memory optimized:**
  
  - **Current generation:**
    - **R3 node types:** cache.r3.large, cache.r3.xlarge, cache.r3.2xlarge, cache.r3.4xlarge, cache.r3.8xlarge
    - **R4 node types:** cache.r4.large, cache.r4.xlarge, cache.r4.2xlarge, cache.r4.4xlarge, cache.r4.8xlarge, cache.r4.16xlarge
  
  - **Previous generation: (not recommended)**
    - **M2 node types:** cache.m2.xlarge, cache.m2.2xlarge, cache.m2.4xlarge

**Additional node type info**

- All T2 instances are created in an Amazon Virtual Private Cloud (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 instances.
- Redis Multi-AZ with automatic failover is supported on T2 instances only when running Redis (cluster mode enabled) - version 3.2.4 or later with the `default.redis3.2.cluster.on` parameter group or one derived from it.
- Redis configuration variables `appendonly` and `appendfsync` are not supported on Redis version 2.8.22 and later.

```
--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. 362).

```
--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.

If you want to enable in-transit or at-rest encryption on this cluster, add these parameters:

- --transit-encryption-enabled

  If you enable in-transit encryption, the cluster must be created in a Amazon VPC and you must also include the parameter --cache-subnet-group.

- --auth-token with the customer specified string value for your AUTH token (password) needed to perform operations on this cluster.

- --at-rest-encryption-enabled

The following operation creates a Redis (cluster mode disabled) replication group new-group with three nodes, a primary and two replicas.

For Linux, macOS, or Unix:

```bash
aws elasticache create-replication-group 
   --replication-group-id new-group 
   --replication-group-description "Demo cluster with replicas" 
   --num-cache-clusters 3 
   --cache-node-type cache.m4.large 
   --cache-parameter-group default.redis3.2 
   --engine redis 
   --engine-version 3.2.4
```

For Windows:

```bash
aws elasticache create-replication-group ^
   --replication-group-id new-group ^
   --replication-group-description "Demo cluster with replicas" ^
   --num-cache-clusters 3 ^
   --cache-node-type cache.m4.large ^
   --cache-parameter-group default.redis3.2 ^
   --engine redis ^
   --engine-version 3.2.4
```

Output from the this command is something like this.

```json
{
   "ReplicationGroup": {
      "Status": "creating",
      "Description": "Demo cluster with replicas",
      "ClusterEnabled": false,
      "ReplicationGroupId": "new-group",
      "SnapshotRetentionLimit": 0,
      "AutomaticFailover": "disabled",
      "SnapshotWindow": "01:30-02:30",
      "MemberClusters": [
         "new-group-001",
         "new-group-002",
         "new-group-003"
      ]
   }
}
```
"new-group-003"
},
"CacheNodeType": "cache.m4.large",
"PendingModifiedValues": {}
}
}

For additional information and parameters you might want to use, see the AWS CLI topic create-replication-group.

Creating a Redis (cluster mode disabled) Cluster with Replicas 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 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.

- **Current generation:**
  - **T2 node types:** cache.t2.micro, cache.t2.small, cache.t2.medium
  - **M3 node types:** cache.m3.medium, cache.m3.large, cache.m3.xlarge, cache.m3.2xlarge
  - **M4 node types:** cache.m4.large, cache.m4.xlarge, cache.m4.2xlarge, cache.m4.4xlarge, cache.m4.10xlarge
- **Previous generation:** (not recommended)
Creating a Redis Cluster with Replicas from Scratch

T1 node types: cache.t1.micro

M1 node types: cache.m1.small, cache.m1.medium, cache.m1.large, cache.m1.xlarge

- Compute optimized:
  - Previous generation: (not recommended)

C1 node types: cache.c1.xlarge

- Memory optimized:
  - Current generation:

R3 node types: cache.r3.large, cache.r3.xlarge, cache.r3.2xlarge, cache.r3.4xlarge, cache.r3.8xlarge

R4 node types: cache.r4.large, cache.r4.xlarge, cache.r4.2xlarge, cache.r4.4xlarge, cache.r4.8xlarge, cache.r4.16xlarge

- Previous generation: (not recommended)

M2 node types: cache.m2.xlarge, cache.m2.2xlarge, cache.m2.4xlarge

Additional node type info

- All T2 instances are created in an Amazon Virtual Private Cloud (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 instances.
- Redis Multi-AZ with automatic failover is supported on T2 instances only when running Redis (cluster mode enabled) - version 3.2.4 or later with the default.redis3.2.cluster.on parameter group or one derived from it.
- Redis configuration variables appendonly and appendfsync are not supported on Redis version 2.8.22 and later.

### 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. 362).

#### 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.

If you want to enable in-transit or at-rest encryption on this cluster, add these parameters:

- `--transit-encryption-enabled`

If you enable in-transit encryption, the cluster must be created in a Amazon VPC and you must also include the parameter `--cache-subnet-group`.
- `--auth-token` with the customer specified string value for your AUTH token (password) needed to perform operations on this cluster.
- `--at-rest-encryption-enabled`

The following operation creates the Redis (cluster mode disabled) replication group `myRep1Group` with three nodes, a primary and two replicas.

```plaintext
https://elasticache.us-west-2.amazonaws.com/?Action=CreateReplicationGroup
&CacheNodeType=cache.m4.large
&CacheParameterGroup=default.redis3.2
&Engine=redis
&EngineVersion=3.2.4
&NumCacheClusters=3
&ReplicationGroupDescription=test%20group
&ReplicationGroupId=myRep1Group
&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) Cluster with Replicas from Scratch

You can create a Redis (cluster mode enabled) cluster (API/CLI: replication group) using the ElastiCache console, the AWS CLI, or the ElastiCache API. A Redis (cluster mode enabled) replication group has from 1 to 15 shards (API/CLI: node groups), 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.

Creating a Redis (cluster mode enabled) Cluster
- Creating a Redis (cluster mode enabled) Cluster (Console) (p. 267)
- Creating a Redis (cluster mode enabled) Cluster with Replicas from Scratch (AWS CLI) (p. 267)
- Creating a Redis (cluster mode enabled) Cluster with Replicas from Scratch (ElastiCache API) (p. 271)

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

To create a Redis (cluster mode enabled) cluster, see Creating a Redis (cluster mode enabled) Cluster (Console) (p. 163). 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) Cluster with Replicas 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

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:
    - T2 node types: cache.t2.micro, cache.t2.small, cache.t2.medium
**M3 node types**: cache.m3.medium, cache.m3.large, cache.m3.xlarge, cache.m3.2xlarge

**M4 node types**: cache.m4.large, cache.m4.xlarge, cache.m4.2xlarge, cache.m4.4xlarge, cache.m4.10xlarge

- Previous generation: (not recommended)

**T1 node types**: cache.t1.micro

**M1 node types**: cache.m1.small, cache.m1.medium, cache.m1.large, cache.m1.xlarge

- Compute optimized:
- Previous generation: (not recommended)

**C1 node types**: cache.c1.xlarge

- Memory optimized:
- Current generation:

**R3 node types**: cache.r3.large, cache.r3.xlarge, cache.r3.2xlarge, cache.r3.4xlarge, cache.r3.8xlarge

**R4 node types**: cache.r4.large, cache.r4.xlarge, cache.r4.2xlarge, cache.r4.4xlarge, cache.r4.8xlarge, cache.r4.16xlarge

- Previous generation: (not recommended)

**M2 node types**: cache.m2.xlarge, cache.m2.2xlarge, cache.m2.4xlarge

**Additional node type info**
- All T2 instances are created in an Amazon Virtual Private Cloud (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 instances.
- Redis Multi-AZ with automatic failover is supported on T2 instances only when running Redis (cluster mode enabled) - version 3.2.4 or later with the `default.redis3.2.cluster.on` parameter group or one derived from it.
- Redis configuration variables `appendonly` and `appendfsync` are not supported on Redis version 2.8.22 and later.

```
--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. 363).

```
--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 0 to 5.
If you want to enable in-transit or at-rest encryption on this cluster, add these parameters:

- `--transit-encryption-enabled`

If you enable in-transit encryption, the cluster must be created in an Amazon VPC and you must also include the parameter `--cache-subnet-group`.

- `--auth-token` with the customer specified string value for your AUTH token (password) needed to perform operations on this cluster.

- `--at-rest-encryption-enabled`

The following operation creates the Redis (cluster mode enabled) replication group `new-group` with three node groups/shards (`--num-node-groups`), each with three nodes, a primary and two read replicas (`--replicas-per-node-group`).

For Linux, macOS, or Unix:

```
aws elastiache create-replication-group
    --replication-group-id new-group
    --replication-group-description "Demo cluster with replicas"
    --num-node-groups 3
    --replicas-per-node-group 2
    --cache-node-type cache.m4.large
    --cache-parameter-group default.redis3.2.cluster.on
    --engine redis
    --engine-version 3.2.4
```

For Windows:

```
aws elastiache create-replication-group
    --replication-group-id new-group
    --replication-group-description "Demo cluster with replicas"
    --num-node-groups 3
    --replicas-per-node-group 2
    --cache-node-type cache.m4.large
    --cache-parameter-group default.redis3.2.cluster.on
    --engine redis
    --engine-version 3.2.4
```

The preceding command generates the following output.

```
{
    "ReplicationGroup": {
        "Status": "creating",
        "Description": "Demo cluster with replicas",
        "ReplicationGroupId": "new-group",
        "SnapshotRetentionLimit": 0,
        "AutomaticFailover": "enabled",
        "SnapshotWindow": "05:30-06:30",
        "MemberClusters": [
            "new-group-0001-001",
            "new-group-0001-002",
            "new-group-0001-003",
            "new-group-0002-001",
            "new-group-0002-002",
            "new-group-0002-003",
            "new-group-0003-001",
            "new-group-0003-002",
            "new-group-0003-003"
        ],
        "PendingModifiedValues": {}
    }
}
```
When you create a Redis (cluster mode enabled) replication group from scratch, you are able to configure each shard in the cluster using the `--node-group-configuration` parameter as shown in the following example which configures two node groups (Console: shards). The first shard has two nodes, a primary and one read replica. The second shard has three nodes, a primary and two read replicas.

**--node-group-configuration**

The configuration for each node group. The `--node-group-configuration` 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 chooses the Availability Zone for the primary node.
  
  **Example:** `us-west-2a`.

- **ReplicaAvailabilityZones** – 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 `ReplicaCount`. If this parameter is omitted, ElastiCache chooses the Availability Zones for the replica nodes.
  
  **Example:** "us-west-2a,us-west-2b,us-west-2c"

- **ReplicaCount** – The number of replica nodes in this node group.

- **Slots** – A string that specifies the keyspace for the node group. The string is in the format `startKey-endKey`. If this parameter is omitted, ElastiCache allocates keys equally among the node groups.
  
  **Example:** "0-4999"

The following operation creates the Redis (cluster mode enabled) replication group `new-group` with two node groups/shards (`--num-node-groups`). Unlike the preceding example, each node group is configured differently from the other node group (`--node-group-configuration`).

For Linux, macOS, or Unix:

```bash
aws elasticache create-replication-group
  --replication-group-id rc-rg
  --replication-group-description "Sharded replication group"
  --engine redis
  --engine-version 3.2.4
  --cache-parameter-group default.redis3.2.cluster.on
  --snapshot-retention-limit 8
  --cache-node-type cache.m4.medium
  --num-node-groups 2
  --node-group-configuration
    "ReplicaCount=1,Slots=0-8999,PrimaryAvailabilityZone='us-east-1c',ReplicaAvailabilityZones='us-east-1a'",
    "ReplicaCount=2,Slots=9000-16383,PrimaryAvailabilityZone='us-east-1a',ReplicaAvailabilityZones='us-east-1a','us-east-1c'"
```

For Windows:

```bash
aws elasticache create-replication-group ^
  --replication-group-id rc-rg ^
  --replication-group-description "Sharded replication group" ^
```
Creating a Redis Cluster with Replicas from Scratch

```
--engine redis ^
--engine-version 3.2.4 ^
--cache-parameter-group default.redis3.2.cluster.on ^
--snapshot-retention-limit 8 ^
--cache-node-type cache.m4.medium ^
--num-node-groups 2 ^
--node-group-configuration \
  "ReplicaCount=1,Slots=0-8999,PrimaryAvailabilityZone='us-east-1c',ReplicaAvailabilityZones='us-east-1b'" \
  "ReplicaCount=2,Slots=9000-16383,PrimaryAvailabilityZone='us-east-1a',ReplicaAvailabilityZones='us-east-1a','us-east-1c'"
```

The preceding operation generates the following output.

```json
{
"ReplicationGroup": {
  "Status": "creating",
  "Description": "Sharded replication group",
  "ReplicationGroupId": "rc-rg",
  "SnapshotRetentionLimit": 8,
  "AutomaticFailover": "enabled",
  "SnapshotWindow": "10:00-11:00",
  "MemberClusters": [
    "rc-rg-0001-001",
    "rc-rg-0001-002",
    "rc-rg-0002-001",
    "rc-rg-0002-002",
    "rc-rg-0002-003"
  ],
  "PendingModifiedValues": {}
}
}
```

For additional information and parameters you might want to use, see the AWS CLI topic create-replication-group.

Creating a Redis (cluster mode enabled) Cluster with Replicas 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 chooses 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 chooses the Availability Zones for the replica nodes.

• ReplicaCount – The number of replica nodes in this node group.

• Slots – A string that specifies the keyspace for the node group. The string is in the format startKey-endKey. If this parameter is omitted, ElastiCache allocates keys equally among the node groups.

Example: "0-4999"

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:
    • T2 node types: cache.t2.micro, cache.t2.small, cache.t2.medium
    • M3 node types: cache.m3.medium, cache.m3.large, cache.m3.xlarge, cache.m3.2xlarge
    • M4 node types: cache.m4.large, cache.m4.xlarge, cache.m4.2xlarge, cache.m4.4xlarge, cache.m4.10xlarge
  • Previous generation: (not recommended)
    • T1 node types: cache.t1.micro
    • M1 node types: cache.m1.small, cache.m1.medium, cache.m1.large, cache.m1.xlarge
  • Compute optimized:
    • Previous generation: (not recommended)
      • C1 node types: cache.c1.xlarge
  • Memory optimized:
    • Current generation:
**R3 node types:** cache.r3.large, cache.r3.xlarge, cache.r3.2xlarge, cache.r3.4xlarge, cache.r3.8xlarge

**R4 node types:** cache.r4.large, cache.r4.xlarge, cache.r4.2xlarge, cache.r4.4xlarge, cache.r4.8xlarge, cache.r4.16xlarge

- Previous generation: (not recommended)

**M2 node types:** cache.m2.xlarge, cache.m2.2xlarge, cache.m2.4xlarge

**Additional node type info**
- All T2 instances are created in an Amazon Virtual Private Cloud (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 instances.
- Redis Multi-AZ with automatic failover is supported on T2 instances only when running Redis (cluster mode enabled) - version 3.2.4 or later with the default.redis3.2.cluster.on parameter group or one derived from it.
- Redis configuration variables appendonly and appendfsync are not supported on Redis version 2.8.22 and later.

**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. 363).

**Engine**
- redis

**EngineVersion**
- 3.2.4

If you want to enable in-transit or at-rest encryption on this cluster, add these parameters:

- **TransitEncryptionEnabled=true**
  - If you enable in-transit encryption, the cluster must be created in a Amazon VPC and you must also include the parameter CacheSubnetGroup.
  - **AuthToken** with the customer specified string value for your AUTH token (password) needed to perform operations on this cluster.
  - **AtRestEncryptionEnabled=true**

Line breaks are added for ease of reading.

```plaintext
https://elasticache.us-west-2.amazonaws.com/
?Action=CreateReplicationGroup
&CacheNodeType=cache.m4.large
&CacheParameterGroup=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
```

API Version 2015-02-02
273
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).

Viewing a Replication Group's Details

- Viewing a Redis (cluster mode disabled) with Replicas Details: Redis (cluster mode disabled) (p. 274)
  - Viewing a Redis (cluster mode disabled) Cluster with Replicas Details (Console) (p. 274)
  - Viewing a Redis (cluster mode disabled) Cluster with Replicas Details (AWS CLI) (p. 274)
  - Viewing a Redis (cluster mode disabled) Cluster with Replicas Details (ElastiCache API) (p. 275)
- Viewing a Replication Group's Details: Redis (cluster mode enabled) (p. 275)
  - Viewing a Redis (cluster mode enabled) Cluster's Details (Console) (p. 275)
  - Viewing a Redis (cluster mode enabled) Cluster's Details (AWS CLI) (p. 275)
  - Viewing a Redis (cluster mode enabled) Cluster's Details (ElastiCache API) (p. 275)
- Viewing a Replication Group's Details: (AWS CLI) (p. 275)
- Viewing a Replication Group's Details: (ElastiCache API) (p. 277)

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 (API/CLI: replication group) using the ElastiCache console, the AWS CLI for ElastiCache, or the ElastiCache API.

Viewing a Redis (cluster mode disabled) Cluster's Details

- Viewing a Redis (cluster mode disabled) Cluster with Replicas Details (Console) (p. 274)
- Viewing a Redis (cluster mode disabled) Cluster with Replicas Details (AWS CLI) (p. 274)
- Viewing a Redis (cluster mode disabled) Cluster with Replicas Details (ElastiCache API) (p. 275)

Viewing a Redis (cluster mode disabled) Cluster with Replicas Details (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. 174).

Viewing a Redis (cluster mode disabled) Cluster with Replicas Details (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. 275).
Viewing a Redis (cluster mode disabled) Cluster with Replicas Details (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. 277).

Viewing a Replication Group's Details: Redis (cluster mode enabled)

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

To view the details of a Redis (cluster mode enabled) cluster using the ElastiCache console, see Viewing a Redis (cluster mode enabled) Cluster's Details (Console) (p. 175).

Viewing a Redis (cluster mode enabled) Cluster's Details (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. 275).

Viewing a Redis (cluster mode enabled) Cluster's Details (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. 277).

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.

Example

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 `new-group`.

```
aws elasticache describe-replication-groups --replication-group-id new-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).

```
{
  "ReplicationGroups": [
    {
      "Status": "available",
      "Description": "test",
      "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"
      ]
    }
  ]
}
```
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 lists 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 lists 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 node 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 (API/CLI: node groups), which mean they also have multiple primary nodes, have a different endpoint structure than Redis (cluster mode disabled) clusters. 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 a Redis (cluster mode disabled) Cluster's Endpoints (Console) (p. 65)
- Finding a Redis (cluster mode enabled) Cluster's Endpoints (Console) (p. 67)
- Finding the Endpoints for Replication Groups (AWS CLI) (p. 70)
- Finding Endpoints for Replication Groups (ElastiCache API) (p. 73)

**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 node 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)

You can view the details, including endpoints, for a replication group using the AWS CLI `describe-replication-groups` command. Use the `-replication-group-id` parameter to specify which replication group's endpoints you want to find.

For an AWS CLI example for finding endpoints, see Finding the Endpoints for Replication Groups (AWS CLI) (p. 70).

Example Finding endpoints for a replication group (AWS CLI)

The following code lists the details, including the replication group's endpoints, for `my-repl-group`.

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

Output from this operation should look something like this (JSON format). The `ReadEndpoint` is each node's unique endpoint. You should use this endpoint for all read operations. For write operations, use the `PrimaryEndpoint` endpoint.

```
{
    "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-001.1abc4d.0001.usw2.cache.amazonaws.com"
                    }
                }
            ]
        }
    ]
}
```
Finding Replication Group Endpoints (ElastiCache API)

You can view the details for a replication group using the ElastiCache API DescribeReplicationGroups action with the ReplicationGroupId parameter to specify a specific replication group.

For an ElastiCache API example for finding endpoints, see Finding Endpoints for Replication Groups (ElastiCache API) (p. 73).

Example Finding endpoints for a replication group (ElastiCache API)

The following code lists the details, including the replication group's endpoints, for myReplGroup.

```
https://elasticache.us-west-2.amazonaws.com/
?Action=DescribeReplicationGroups
&MaxRecords=100
&ReplicationGroupId=myReplGrp
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```

Output from this operation should look something like this. The ReadEndpoint is each node's unique endpoint. You should use this endpoint for all read operations. For write operations, use the PrimaryEndpoint endpoint.

```
<DescribeReplicationGroupsResponse xmlns="http://elasticache.amazonaws.com/doc/2015-02-02/">
  <DescribeReplicationGroupsResult>
    <ReplicationGroups>
      <ReplicationGroup>
        <SnapshottingClusterId>myreplgrp</SnapshottingClusterId>
        <MemberClusters>
          <ClusterId>myreplgrp-001</ClusterId>
        </MemberClusters>
      </ReplicationGroup>
    </ReplicationGroups>
  </DescribeReplicationGroupsResult>
</DescribeReplicationGroupsResponse>
```
Redis (cluster mode disabled)

The primary endpoint, including the port, is found between the `<PrimaryEndpoint>` and `</PrimaryEndpoint>` tags.

```
<PrimaryEndpoint>
  <Port>6379</Port>
  <Address>myreplgrp.q68zge.ng.0001.use1devo.elmo-dev.amazonaws.com</Address>
</PrimaryEndpoint>
```

The read endpoints are found between the `<ReadEndpoint>` and `</ReadEndpoint>` tags for each node group in the replication group.

```
<ReadEndpoint>
  <Port>6379</Port>
  <Address>myreplgrp-001.q68zge.0001.use1devo.elmo-dev.amazonaws.com</Address>
</ReadEndpoint>
```
and

```xml
<CacheClusterId>myreplgrp-002</CacheClusterId>
<ReadEndpoint>
  <Port>6379</Port>
  <Address>myreplgrp-002.q68zge.0001.use1devo.elmo-dev.amazonaws.com</Address>
</ReadEndpoint>
```

**Redis (cluster mode enabled)**

The configuration endpoint, including the port, is found between the `<ConfigurationEndpoint>` and `</ConfigurationEndpoint>` tags.

```xml
<CacheClusterId>myreplgrp-001</CacheClusterId>
```

For more information, go to the ElastiCache API reference topic [DescribeReplicationGroups](https://docs.aws.amazon.com/AmazonElastiCache/latest/red不错).
Modifying a Cluster with Replicas

**Important Constraints**

- Currently, ElastiCache does not support modifying a Redis (cluster mode enabled) cluster using the API operation `ModifyReplicationGroup` (CLI: `modify-replication-group`). However, you can modify the number of shards (node groups) in a Redis (cluster mode enabled) cluster with the API operation `ModifyReplicationGroupShardConfiguration` (CLI: `modify-replication-group-shard-configuration`). For more information, see Scaling for Amazon ElastiCache for Redis—Redis (cluster mode enabled) (p. 225).

Other modifications to a Redis (cluster mode enabled) cluster require that you 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 cluster or replication group and creating it anew. For more information, see Upgrading Engine Versions (p. 54).

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 except by creating a backup of the current replication group then using that backup to seed a new Redis (cluster mode enabled) replication group.

**Topics**

- Modifying a Redis Cluster (Console) (p. 284)
- Modifying a Replication Group (AWS CLI) (p. 284)
- Modifying a Replication Group (ElastiCache API) (p. 285)

**Modifying a Redis Cluster (Console)**

To modify a Redis (cluster mode disabled) cluster, see Modifying an ElastiCache Cluster (p. 179).

**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, macOS, or Unix:

```bash
aws elasticache modify-replication-group \
   --replication-group-id myReplGroup \
   --automatic-failover-enabled
```

For Windows:

```bash
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 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. 197).

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. 184)
- Adding a Read Replica to a Replication Group (AWS CLI) (p. 287)
- Adding a Read Replica to a Replication Group (ElastiCache API) (p. 288)

**Topics**
- Adding a Read Replica to a Cluster (Console) (p. 287)
- Adding a Read Replica to a Replication Group (AWS CLI) (p. 287)
- Adding a Read Replica to a Replication Group (ElastiCache API) (p. 288)

**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. 184).

**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, macOS, or Unix:

```
aws elasticache create-cache-cluster \
  --cache-cluster-id my-read-replica \
  --replication-group-id my-replication-group
```

For Windows:

```
aws elasticache create-cache-cluster ^
  --cache-cluster-id my-read-replica ^
  --replication-group-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. 284).
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) (p. 289)
- Promoting a Read-Replica to Primary (AWS CLI) (p. 290)
- Promoting a Read-Replica to Primary (ElastiCache API) (p. 290)

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. 179).
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. 179)

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. 284).

The following AWS CLI command modifies the replication group new-group, making the read replica my-replica-1 the primary in the replication group.

For Linux, macOS, or Unix:

```bash
aws elasticache modify-replication-group \
  --replication-group-id new-group \
  --primary-cluster-id my-replica-1
```

For Windows:

```bash
aws elasticache modify-replication-group ^
  --replication-group-id new-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. 285).

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. 289).
  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. 284).
  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. 197)
- [Deleting a Cache Cluster (AWS CLI)](p. 197)
- [Deleting a Cache Cluster (ElastiCache API)](p. 198)
ElastiCache Backup and 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. 253).

For more information about the performance impact of the backup process, see *Performance Impact of Backups* (p. 294).

This section provides an overview of working with backup and restore.

**Important**
Though rare, there are times when the backup process fails to create a backup, including final backups. Insufficient reserved memory is often the cause of backup failures. Therefore, you should be sure you have sufficient reserved memory before attempting a backup. If you have insufficient memory you can either evict some keys, or increase the value of `reserved-memory-percent`.

For more information, see:
- *Ensuring You Have Sufficient Memory to Create a Redis Snapshot* (p. 77)
- *Managing Reserved Memory (Redis)* (p. 79)

If you are planning to delete the cluster and it's important to preserve the data, you can take an extra precaution by creating a manual backup first, verify that its status is *available*, and then proceed with deleting the cluster. This will ensure that if the backup fails you still have the cluster data available so you can retry making a backup, following the best practices outlined above.

**Topics**
- *Backup Constraints* (p. 294)
- *Backup Costs* (p. 294)
- *Performance Impact of Backups* (p. 294)
- *Scheduling Automatic Backups* (p. 296)
- *Making Manual Backups* (p. 297)
- *Creating a Final Backup* (p. 303)
- *Describing Backups* (p. 306)
Backup Constraints

The following constraints should be considered when planning or making backups:

- At this time, backup and restore are supported only for clusters running on Redis.
- For Redis (cluster mode disabled) clusters, backup and restore are not supported on cache.t1.micro or cache.t2.* nodes. All other cache node types are supported.
- For Redis (cluster mode enabled) clusters, backup and restore are supported for all node types.
- During any contiguous 24-hour period, you can create no more than 20 manual backups per node in the cluster.
- Redis (cluster mode enabled) only supports taking backups on the cluster level (for the API or CLI, the replication group level), not at the shard level (for the API or CLI, the node group level).
- During the backup process you cannot perform any additional API or CLI operations on the cluster.

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 operation: 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.

Improving Backup Performance

The following are guidelines for improving backup performance.

- Set the `reserved-memory-percent` parameter—To mitigate excessive paging, we recommend that you set the `reserved-memory-percent` 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` and `reserved-memory-percent` parameters, see Managing Reserved Memory (Redis) (p. 79).

- 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 BGSAVE, we recommend that you create backups from one of the read replicas. 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 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 with Optional Cluster Resizing (p. 317).

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. 179) or Modifying a Cluster with Replicas (p. 284).

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. 159). 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 Redis Cluster with Replicas from Scratch (p. 260).
Making 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 any manual backup. Even if you delete a cluster or node, any manual backups from that cluster or node are retained. If you no longer want to keep a manual backup, you must explicitly delete it yourself.

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 directly creating a manual backup, you can create a manual backup in one of the following ways:

- **Copying a Backup** (p. 308) It does not matter whether the source backup was created automatically or manually.
- **Creating a Final Backup** (p. 303) Create a backup immediately before deleting a cluster or node.

**Other topics of import**

- Backup Constraints (p. 294)
- Backup Costs (p. 294)
- Performance Impact of Backups (p. 294)

You can create a manual backup of a node using the AWS Management Console, the AWS CLI, or the ElastiCache API.

**Creating a Manual Backup (Console)**

**To create a backup of a cluster (console)**

2. From the 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 back up.
4. Choose **Backup**.
5. In the **Create Backup** dialog, type in a name for your backup in the **Backup Name** box. We recommend that the name indicate which cluster was backed up and the date and time the backup was made.

**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 operation with the following parameters:

• **--cache-cluster-id**
  • If the cluster you're backing up has no replica nodes, **--cache-cluster-id** is the name of the cluster you are backing up, e.g., *mycluster*.
  • If the cluster you're backing up has one or more replica nodes, **--cache-cluster-id** is the name of the node in the cluster you want to use for the backup, e.g., *mycluster-002*.

Only use this parameter when backing up a Redis (cluster mode disabled) cluster.

• **--replication-group-id** – Name of the Redis (cluster mode enabled) cluster (CLI/API: a replication group) to use as the source for the backup. Use this parameter when backing up a Redis (cluster mode enabled) cluster.

• **--snapshot-name** – Name of the snapshot to be created.

**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.

**AWS CLI Code Examples**

• Example 1: Backing Up a Redis (cluster mode disabled) Cluster That Has No Replica Nodes (p. 298)
• Example 2: Backing Up a Redis (cluster mode disabled) Cluster with Replica Nodes (p. 299)
• Example 3: Backing Up a Redis (cluster mode enabled) Cluster (p. 300)
• AWS CLI Related Topics (p. 300)

**Example 1: Backing Up a Redis (cluster mode disabled) Cluster That Has No Replica Nodes**

The following AWS CLI operation creates the backup *bkup-20150515* from the Redis (cluster mode disabled) cluster *myNonClusteredRedis* that has no read replicas.

For Linux, macOS, or Unix:

```bash
aws elasticache create-snapshot \\
    --cache-cluster-id myNonClusteredRedis \\
    --snapshot-name bkup-20150515
```

For Windows:
Example 2: Backing Up a Redis (cluster mode disabled) Cluster with Replica Nodes

The following AWS CLI operation creates the backup bkup-20150515 from the Redis (cluster mode disabled) cluster myNonClusteredRedis which has one or more read replicas.

For Linux, macOS, or Unix:

```
aws elasticache create-snapshot
  --cache-cluster-id myNonClusteredRedis
  --snapshot-name bkup-20150515
```

For Windows:

```
aws elasticache create-snapshot
  --cache-cluster-id myNonClusteredRedis-001
  --snapshot-name bkup-20150515
```

Example Output: Backing Up a Redis (cluster mode disabled) Cluster with Replica Nodes

Output from the operation will look something like the following.

```
{
  "Snapshot": {
    "Engine": "redis",
    "CacheParameterGroupName": "default.redis3.2",
    "VpcId": "vpc-91280df6",
    "CacheClusterId": "myNonClusteredRedis-001",
    "SnapshotRetentionLimit": 0,
    "NumCacheNodes": 1,
    "SnapshotName": "bkup-20150515",
    "CacheClusterCreateTime": "2017-01-12T18:59:48.048Z",
    "AutoMinorVersionUpgrade": true,
    "PreferredAvailabilityZone": "us-east-1c",
    "SnapshotStatus": "creating",
    "SnapshotSource": "manual",
    "SnapshotWindow": "08:30-09:30",
    "EngineVersion": "3.2.4",
    "NodeSnapshots": [
      {
        "CacheSize": "",
        "CacheNodeId": "0001",
        "CacheNodeCreateTime": "2017-01-12T18:59:48.048Z"
      }
    ],
    "CacheSubnetGroupName": "default",
    "Port": 6379,
    "PreferredMaintenanceWindow": "wed:07:30-wed:08:30",
    "CacheNodeType": "cache.m3.2xlarge"
  }
}
```
Example 3: Backing Up a Redis (cluster mode enabled) Cluster

The following AWS CLI operation creates the backup bkup-20150515 from the Redis (cluster mode enabled) cluster myClusteredRedis. Note the use of --replication-group-id instead of --cache-cluster-id to identify the source.

For Linux, macOS, or Unix:

```
aws elasticache create-snapshot
   --replication-group-id myClusteredRedis \
   --snapshot-name bkup-20150515
```

For Windows:

```
aws elasticache create-snapshot
   --replication-group-id myClusteredRedis \
   --snapshot-name bkup-20150515
```

Example Output: Backing Up a Redis (cluster mode enabled) Cluster

Output from this operation will look something like the following.

```
{
   "Snapshot": {
      "Engine": "redis",
      "CacheParameterGroupName": "default.redis3.2.cluster.on",
      "VpcId": "vpc-91280df6",
      "NodeSnapshots": [
         { "CacheSize": "", "NodeGroupId": "0001" },
         { "CacheSize": "", "NodeGroupId": "0002" }
      ],
      "NumNodeGroups": 2,
      "SnapshotName": "bkup-20150515",
      "ReplicationGroupId": "myClusteredRedis",
      "AutoMinorVersionUpgrade": true,
      "AutomaticFailover": "enabled",
      "SnapshotStatus": "creating",
      "SnapshotSource": "manual",
      "SnapshotWindow": "10:00-11:00",
      "EngineVersion": "3.2.4",
      "CacheSubnetGroupName": "default",
      "ReplicationGroupDescription": "2 shards 2 nodes each",
      "Port": 6379,
      "PreferredMaintenanceWindow": "sat:03:30-sat:04:30",
      "CacheNodeType": "cache.r3.large"
   }
}
```

AWS CLI Related Topics

For more information, see create-snapshot in the AWS CLI Command Reference.
Creating a Manual Backup (ElastiCache API)

To create a manual backup of a cluster using the ElastiCache API, use the `CreateSnapshot` ElastiCache API operation with the following parameters:

- **CacheClusterId**
  - If the cluster you're backing up has no replica nodes, `CacheClusterId` is the name of the cluster you are backing up, e.g., `mycluster`.
  - If the cluster you're backing up has one or more replica nodes, `CacheClusterId` is the name of the node in the cluster you want to use for the backup, e.g., `mycluster-002`.

  Only use this parameter when backing up a Redis (cluster mode disabled) cluster.

- **ReplicationGroupId** – Name of the Redis (cluster mode enabled) cluster (CLI/API: a replication group) to use as the source for the backup. Use this parameter when backing up a Redis (cluster mode enabled) cluster.

- **SnapshotName** – Name of the snapshot to be created.

  **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.

**API Code Examples**

- **Example 1: Backing Up a Redis (cluster mode disabled) Cluster That Has No Replica Nodes (p. 301)**
- **Example 2: Backing Up a Redis (cluster mode disabled) Cluster with Replica Nodes (p. 302)**
- **Example 3: Backing Up a Redis (cluster mode enabled) Cluster (p. 302)**
- **ElastiCache API Related Topics (p. 302)**

**Example 1: Backing Up a Redis (cluster mode disabled) Cluster That Has No Replica Nodes**

The following ElastiCache API operation creates the backup `bkup-20150515` from the Redis (cluster mode disabled) cluster `myNonClusteredRedis` that has no read replicas.

```plaintext
https://elasticache.us-west-2.amazonaws.com/?Action=CreateSnapshot
&CacheClusterId=myNonClusteredRedis
&SnapshotName=bkup-20150515
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>
```
Example 2: Backing Up a Redis (cluster mode disabled) Cluster with Replica Nodes

The following ElastiCache API operation creates the backup bkup-20150515 from the Redis (cluster mode disabled) cluster myNonClusteredRedis which has one or more read replicas.

```plaintext
https://elasticache.us-west-2.amazonaws.com/
  ?Action=CreateSnapshot
  &CacheClusterId=myNonClusteredRedis-001
  &SnapshotName=bkup-20150515
  &Version=2015-02-02
  &SignatureVersion=4
  &SignatureMethod=HmacSHA256
  &Timestamp=20150202T192317Z
  &X-Amz-Credential=<credential>
```

Example 3: Backing Up a Redis (cluster mode enabled) Cluster

The following ElastiCache API operation creates the backup bkup-20150515 from the Redis (cluster mode enabled) cluster myClusteredRedis. Note the use of ReplicationGroupId instead of CacheClusterId to identify the source.

```plaintext
https://elasticache.us-west-2.amazonaws.com/
  ?Action=CreateSnapshot
  &ReplicationGroupId=myClusteredRedis
  &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.

ElastiCache API Related Topics

For more information, see CreateSnapshot in the Amazon ElastiCache API Reference.
Creating 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 (for the API or CLI, a replication group) using the ElastiCache console.

To create a final backup when deleting a Redis cluster, on the delete dialog box (step 5), choose Yes and give the backup a name.

Related Topics
• Deleting a Cluster (Console) (p. 197)
• Deleting a Replication Group (Console) (p. 286)

Creating a Final Backup (AWS CLI)

You can create a final backup when deleting a Redis cluster (for the API or CLI, a replication group) using the AWS CLI.

Topics
• When Deleting a Redis Cluster With No Read Replicas (p. 303)
• When Deleting a Redis Cluster With Read Replicas (p. 304)

When Deleting a Redis Cluster With No Read Replicas

To create a final backup, use the delete-cache-cluster AWS CLI operation 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, macOS, 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 CLI Command Reference.
Creating a Final Backup (ElastiCache API)

When Deleting a Redis Cluster With Read Replicas

To create a final backup when deleting a replication group, use the delete-replication-group AWS CLI operation, 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, macOS, 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 CLI Command Reference.

Creating 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 (p. 304)
- When Deleting a Redis Replication Group (p. 305)

When Deleting a Redis Cluster

To create a final backup, use the DeleteCacheCluster ElastiCache API operation with the following parameters.

- `CacheClusterId` – Name of the cluster being deleted.
- `FinalSnapshotIdentifier` – Name of the backup.

The following ElastiCache API operation creates the backup `bkup-20150515-final` when deleting the cluster `myRedisCluster`.

```
&CacheClusterId=myRedisCluster
&FinalSnapshotIdentifier=bkup-20150515-final
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
```
When Deleting a Redis Replication Group

To create a final backup when deleting a replication group, use the DeleteReplicationGroup ElastiCache API operation, with the following parameters:

- `ReplicationGroupId` – Name of the replication group being deleted.
- `FinalSnapshotIdentifier` – Name of the final backup.

The following ElastiCache API operation 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

2. From the navigation pane, choose Backups.
3. Use the Filter list to display manual, automatic, or all backups.
4. To see the details of a particular backup, choose 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 operation uses the parameter --max-recordsto list up to 20 backups associated with your account. Omitting the parameter --max-recordss lists up to 50 backups.

```bash
aws elasticache describe-snapshots --max-records 20
```

The following operation uses the parameter --cache-cluster-id to list only the backups associated with the cluster my-cluster.

```bash
aws elasticache describe-snapshots --cache-cluster-id my-cluster
```

The following operation uses the parameter --snapshot-name to display the details of the backup my-backup.

```bash
aws elasticache describe-snapshots --snapshot-name my-backup
```

For more information, see describe-snapshots in the AWS CLI Command Reference.

Describing Backups (ElastiCache API)

To display a list of backups, use the DescribeSnapshots operation.

Examples

The following operation uses the parameter MaxRecordsto list up to 20 backups associated with your account. Omitting the parameter MaxRecords will list up to 50 backups.

```bash
https://elasticache.us-west-2.amazonaws.com/
?Action=DescribeSnapshots
&MaxRecords=20
&SignatureMethod=HmacSHA256
```

API Version 2015-02-02

306
The following operation 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-Signature=<signature>

The following operation 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-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. 310).

The following procedures show you how to copy a backup.

Copying a Backup (Console)

To copy a backup (console)

2. To see a list of your backups, from the left navigation pane choose Backups.
3. From the list of backups, choose the box to the left of the name of the backup you want to copy.
4. Choose Copy.
5. In the Create Copy of the Backup? dialog box, do the following:
   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. 310).
   c. Choose Copy.

Copying a Backup (AWS CLI)

To copy a backup, use the `copy-snapshot` operation.

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. 310).

The following example makes a copy of an automatic backup.

For Linux, macOS, or Unix:

```
aws elasticache copy-snapshot \
  --source-snapshot-name automatic.my-redis-primary-2014-03-27-03-15 \n  --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 operation 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. 310).

The following example makes a copy of an automatic backup.

**Example**

```text
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 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 Seeding a New Cluster with an Externally Created Backup (Redis) (p. 320). Another reason you might want to export your cluster’s data is to use the .rdb file for offline processing.

Important

- The ElastiCache backup and the Amazon S3 bucket that you want to copy it to must be in the same region.
- Though backups copied to an Amazon S3 bucket are encrypted, 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 Amazon S3 bucket to those who you want to access your data. For information about managing access to an Amazon 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. 407) in the ElastiCache User Guide.

Topics

- Step 1: Create an Amazon S3 Bucket (p. 310)
- Step 2: Grant ElastiCache Access to Your Amazon S3 Bucket (p. 311)
- Step 3: Export an ElastiCache Backup (p. 312)

Step 1: Create an Amazon S3 Bucket

The following procedure uses the Amazon S3 console to create an Amazon S3 bucket where you will export and store your ElastiCache backup.
To create an Amazon S3 bucket

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. From the Region list, choose a region for your Amazon S3 bucket. This region must be the same region as the ElastiCache backup you want to export.
   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

In order for ElastiCache to copy a snapshot to an Amazon S3 bucket, it must have access to the bucket.

The following procedure grants ElastiCache access to the Amazon S3 bucket you created in the previous step.

Warning
Even though backups copied to an Amazon S3 bucket are encrypted, your data may be accessed by anyone with access to your Amazon S3 bucket. Therefore, 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.

To grant ElastiCache access to an Amazon S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the name of the Amazon S3 bucket that you want to copy the backup to. This should be the S3 bucket you created in Step 1: Create an Amazon S3 Bucket (p. 310).
3. Make sure that the bucket's region is the same as your ElastiCache backup's region. If it isn't, return to Step 1: Create an Amazon S3 Bucket (p. 310) and create a new bucket in the same region as the cluster you will backup.
4. Choose Permissions.
5. Choose Access Control List.
6. Under Access for other AWS accounts, choose + Add account.
7. In the box, add the region's canonical id as shown in the following list:
   • China (Beijing) and China (Ningxia) Regions:
Step 3: Export an ElastiCache Backup

Now that you've created your S3 bucket and granted ElastiCache permissions to access it, you can use the ElastiCache console, the AWS CLI, or the ElastiCache API to export your snapshot to it. The following assumes that you have the following additional S3 specific IAM permissions.

```json
{
    "Statement": {
        "Effect": "Allow",
        "Action": [ "s3:GetBucketLocation", "s3:ListAllMyBuckets" ],
        "Resource": "arn:aws:s3:::*",
        "Version": "2012-10-17"
    }
}
```

**Topics**
- Exporting an ElastiCache Backup (Console) (p. 312)
- Exporting an ElastiCache Backup (AWS CLI) (p. 313)
- Exporting an ElastiCache Backup (ElastiCache API) (p. 315)

**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.
To export an ElastiCache backup to an Amazon S3 bucket

2. To see a list of your backups, from the left navigation pane choose Backups.
3. From the list of backups, choose the box to the left of the name of the backup you want to export.
4. Choose 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, choose 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. 310)).
      The Target S3 Location must be an Amazon S3 bucket in the backup’s region with the following permissions for the export process to succeed.
      • Object access – Read and Write.
      • Permissions access – Read.
      For more information, see Step 2: Grant ElastiCache Access to Your Amazon S3 Bucket (p. 311).
   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. Return to Step 2: Grant ElastiCache Access to Your Amazon S3 Bucket (p. 311) to add the permissions specified and retry exporting your backup.
   • ElastiCache has not been granted READ permissions %s on the S3 Bucket.
     Solution: Add Read permissions on the bucket.
   • ElastiCache has not been granted WRITE permissions %s on the S3 Bucket.
     Solution: Add Write permissions on the bucket.
   • ElastiCache has not been granted READ_ACP permissions %s on the S3 Bucket.
     Solution: Add Read for Permissions access on the bucket.

   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 the following permissions for the export process to succeed.

• Object access – Read and Write.
• Permissions access – Read.

For more information, see Step 2: Grant ElastiCache Access to Your Amazon S3 Bucket (p. 311).

The following operation copies a backup to my-s3-bucket.

For Linux, macOS, 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. Return to Step 2: Grant ElastiCache Access to Your Amazon S3 Bucket (p. 311) to add the permissions specified and retry exporting your backup.

• ElastiCache has not been granted READ permissions %s on the S3 Bucket.

   **Solution:** Add Read permissions on the bucket.

• ElastiCache has not been granted WRITE permissions %s on the S3 Bucket.

   **Solution:** Add Write permissions on the bucket.

• ElastiCache has not been granted READ_ACP permissions %s on the S3 Bucket.

   **Solution:** Add Read for Permissions access on the bucket.

For more information, see `copy-snapshot` in the AWS CLI Command 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 the following permissions for the export process to succeed.

- **Object access** – Read and Write.
- **Permissions access** – Read.

For more information, see Step 2: Grant ElastiCache Access to Your Amazon S3 Bucket (p. 311).

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 backup to it, you will receive one of the following error messages. Return to Step 2: Grant ElastiCache Access to Your Amazon S3 Bucket (p. 311) to add the permissions specified and retry exporting your backup.

- ElastiCache has not been granted READ permissions %s on the S3 Bucket.
  
  **Solution:** Add Read permissions on the bucket.

- ElastiCache has not been granted WRITE permissions %s on the S3 Bucket.
  
  **Solution:** Add Write permissions on the bucket.

- ElastiCache has not been granted READ_ACP permissions %s on the S3 Bucket.
Solution: Add Read for Permissions access on the bucket.

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 with Optional Cluster Resizing

You can restore the data from a Redis .rdb backup file to a new cluster at any time.

The Amazon ElastiCache for Redis restore process supports the following:

- Upgrading from a Redis (cluster mode disabled) cluster to a Redis (cluster mode enabled) cluster running Redis version 3.2.4.

- Migrating from one or more .rdb backup files you created from your self-managed Redis clusters to a single ElastiCache for Redis (cluster mode enabled) cluster.

  The .rdb files must be put in S3 to perform the restore.

- Specifying a number of shards (API/CLI: node groups) in the new cluster that is different from the number of shards in the cluster that was used to create the backup file.

- Specifying a different node type for the new cluster—larger or smaller. If scaling to a smaller node type, be sure that the new node type has sufficient memory for your data and Redis overhead. For more information, see Choosing Your Node Size (p. 99).

- Configuring the slots of the new Redis (cluster mode enabled) cluster differently than in the cluster that was used to create the backup file.

  **Important**

  - You cannot restore from a backup created using a Redis (cluster mode enabled) cluster to a Redis (cluster mode disabled) cluster.
  - Redis (cluster mode enabled) clusters do not support multiple databases. Therefore, when restoring to a Redis (cluster mode enabled) your restore will fail if the RDB file references more than one database.

Whether you make any changes when restoring a cluster from a backup or not is governed by the choices you make in the Restore Cluster dialog box when using the ElastiCache console, or parameter values when using the AWS CLI or ElastiCache API to restore.

During the restore operation, ElastiCache creates the new cluster, and then populates it with data from the backup file. When this process is complete, the Redis cluster is warmed up and ready to accept requests.

  **Important**

  Before you proceed, be sure you have created a backup of the cluster you want to restore from. For more information, see Making Manual Backups (p. 297).

  If you want to restore from an externally created backup, see Seeding a New Cluster with an Externally Created Backup (Redis) (p. 320).

The following procedures show you how to restore a backup to a new cluster using the ElastiCache console, the AWS CLI, or the ElastiCache API.
Restoring From a Backup (Console)

You can restore a Redis backup to either a single-node Redis (cluster mode disabled) cluster or to a Redis cluster with read replicas (replication group)— either Redis (cluster mode disabled) or Redis (cluster mode enabled).

To restore a backup to a new cluster (console)

2. From the navigation pane, choose Backups.
3. In the list of backups, choose the box to the left of the backup name you want to restore from.
4. Choose Restore.
5. Complete the Restore Cluster dialog box. Be sure to complete all the "Required" fields and any of the others you want to change from the defaults.

Redis (cluster mode disabled)

1. Cluster ID – Required. The name of the new cluster.
2. Engine version compatibility – The ElastiCache for Redis engine version you want to run.
3. Cluster mode enabled (scale out) – Choose this to convert your Redis (cluster mode disabled) cluster to a Redis (cluster mode enabled) (the engine version will become 3.2.4).

If you choose Cluster mode enabled (scale out):

a. Choose the number of shards you want in the new cluster (API/CLI: node groups).

b. Choose the number of read replicas you want in each shard.

c. Distribute your keys among the slots as you desire.

4. Node Type – Specify the node type you want for the new cluster.
5. Availability zone(s) – Specify how you want the cluster’s Availability Zones selected.
6. Port – Change this only if you want the new cluster to use a different port.
7. Choose a VPC – Choose the VPC in which to create this cluster.
8. Parameter Group – Choose a parameter group that reserves sufficient memory for Redis overhead for the node type you selected.

Redis (cluster mode enabled)

1. Cluster ID – Required. The name of the new cluster.
2. Cluster mode enabled (scale out) – Choose this for a Redis (cluster mode enabled) cluster. Clear it for a Redis (cluster mode disabled) cluster.
3. Node Type – Specify the node type you want for the new cluster.
4. Number of Shards – Choose the number of shards you want in the new cluster (API/CLI: node groups).
5. Replicas per Shard – Choose the number of read replica nodes you want in each shard.
6. **Slots and keyspaces** – Choose how you want keys distributed among the shards. If you choose to specify the key distributions complete the table specifying the key ranges for each shard.

7. **Availability zone(s)** – Specify how you want the cluster's Availability Zones selected.

8. **Port** – Change this only if you want the new cluster to use a different port.

9. **Choose a VPC** – Choose the VPC in which to create this cluster.

10. **Parameter Group** – Choose a parameter group that reserves sufficient memory for Redis overhead for the node type you selected.

   6. When the settings are as you want them, choose **Launch Cluster**.

### Restoring From a Backup (AWS CLI)

You can restore a Redis (cluster mode disabled) backup to either a single-node Redis (cluster mode disabled) cluster using the AWS CLI operation `create-cache-cluster` or a Redis cluster with read replicas (replication group)— either Redis (cluster mode disabled) or Redis (cluster mode enabled) using the AWS CLI operation `create-replication-group` and seeding it with a Redis `.rdb` file.

When using either the `create-cache-cluster` or `create-replication-group` operation, be sure to include the parameter `--snapshot-name` or `--snapshot-arns` to seed the new cluster or replication group with the data from the backup.

For more information, see the following:

- `create-cache-cluster` in the *AWS CLI Command Reference*.

- Creating a Redis Cluster with Replicas from Scratch (p. 260) in the *ElastiCache User Guide*.
- `create-replication-group` in the *AWS CLI Command Reference*.

### Restoring From a Backup (ElastiCache API)

You can restore a Redis backup to either a single-node Redis (cluster mode disabled) cluster using the ElastiCache API operation `CreateCacheCluster` or to a Redis cluster with read replicas (replication group)— either Redis (cluster mode disabled) or Redis (cluster mode enabled) using the ElastiCache API operation `CreateReplicationGroup` and seeding it with a Redis `.rdb` file.

When using either the `CreateCacheCluster` or `CreateReplicationGroup` operation, be sure to include the parameter `SnapshotName` or `SnapshotArns` to seed the new cluster or replication group with the data from the backup.

For more information, see the following:

- Creating a Cache Cluster (ElastiCache API) (p. 170) in the *ElastiCache User Guide*.
- `CreateCacheCluster` in the *ElastiCache API Reference*.

- Creating a Redis Cluster with Replicas from Scratch (p. 260) in the *ElastiCache User Guide*.
- `CreateReplicationGroup` in the *ElastiCache API Reference*.
Seeding a New Cluster with an Externally Created Backup (Redis)

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 for Redis cluster with your existing Redis data.

To see a new Redis cluster from a Redis backup created within Amazon ElastiCache, see Restoring From a Backup with Optional Cluster Resizing (p. 317).

When you use a Redis .rdb file to seed a new Redis cluster, you can do the following:

• Upgrade from a nonpartitioned cluster to a Redis (cluster mode enabled) cluster running Redis version 3.2.4.
• Specify a number of shards (called node groups in the API and CLI) in the new cluster that is different from the number of shards in the cluster that was used to create the backup file.
• Specify a different node type for the new cluster—larger or smaller than that used in the cluster that made the backup. If you scale to a smaller node type, be sure that the new node type has sufficient memory for your data and Redis overhead. For more information, see Ensuring You Have Sufficient Memory to Create a Redis Snapshot (p. 77).
• Distribute your keys in the slots of the new Redis (cluster mode enabled) cluster differently than in the cluster that was used to create the backup file.

Note
You cannot seed a Redis (cluster mode disabled) cluster from an .rdb file created from a Redis (cluster mode enabled) cluster.

Important

• You must ensure that your Redis backup data doesn't exceed the resources of the node. For example, you can't 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. 377) and Amazon ElastiCache Product Features and Details.

• Encrypting a Redis .rdb file with Amazon S3 server-side encryption (SSE) is not supported.

Following, you can find topics that walk you through migrating your Redis cluster from outside ElastiCache for Redis to ElastiCache for Redis.

Migrating to ElastiCache for Redis

• Step 1: Create a Redis Backup (p. 321)
• Step 2: Create an Amazon S3 Bucket and Folder (p. 321)
• Step 3: Upload Your Backup to Amazon S3 (p. 322)
• Step 4: Grant ElastiCache Read Access to the .rdb File (p. 322)
• Step 5: Seed the ElastiCache Cluster With the .rdb File Data (p. 323)

Topics

Step 1: Create a Redis Backup (p. 321)
Step 1: Create a Redis Backup

To create the Redis backup from which you will seed your ElastiCache for Redis instance

1. Connect to your existing Redis instance.
2. Run either the Redis BGSAVE or SAVE operation to create a backup. Note where your .rdb file is located.

   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: Create an Amazon S3 Bucket and Folder

When you have created the backup file, you need to upload it to a folder within an Amazon S3 bucket. To do that, you must first have an Amazon S3 bucket and folder within that bucket. If you already have an Amazon S3 bucket and folder with the appropriate permissions, you can skip to Step 3: Upload Your Backup to Amazon S3 (p. 322).

To create an Amazon S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Follow the instructions for creating an Amazon S3 bucket in Creating a Bucket in the Amazon Simple Storage Service Console User Guide.

   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).

   We strongly recommend that you create your Amazon S3 bucket in the same region as your new ElastiCache for Redis cluster. This approach will ensure the highest data transfer speed when ElastiCache reads your .rdb file from Amazon S3.

   Security Advisory
   To keep your data as secure as possible, make the permissions on your Amazon S3 bucket as restrictive as you can while still allowing the bucket and its contents to be used to seed your new Redis cluster.
To add a folder to an Amazon S3 bucket

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the name of the bucket you will upload your .rdb file to.
3. Choose Create folder.
4. Type in a name for your new folder.
5. Choose Save.

Make note of both the bucket name and the folder name.

Step 3: Upload Your Backup to Amazon S3

It is now time to upload the .rdb file you created in Step 1: Create a Redis Backup (p. 321) to the Amazon S3 bucket and folder you created in Step 2: Create an Amazon S3 Bucket and Folder (p. 321). For more information on this task, see Add an Object to a Bucket. Between steps 2 and 3, choose the name of the folder you created.

To upload your .rdb file to an Amazon S3 folder

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Choose the name of the Amazon S3 bucket you created in Step 2.
3. Choose the name of the folder you created in Step 2.
5. Choose Add files.
6. Browse to find the file or files you want to upload, then choose the file or files. To choose multiple files, hold down the Ctrl key while choosing a file name.
7. Choose Open.
8. Confirm the correct file or files are listed in the Upload dialog box, and then choose Upload.

It is important that you note the path to your .rdb file. For example, if my bucket name is myBucket and the path is myFolder/redis.rdb, you type myBucket/myFolder/redis.rdb. You need this path to seed the new cluster with the data in this backup.

For additional information, see Bucket Restrictions and Limitations in the Amazon Simple Storage Service Developer Guide.

Step 4: 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 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. The name of the selected file will appear above the tabs at the top of the page.
Step 5: Seed the ElastiCache Cluster With the .rdb File Data

Choose Permissions.

If `aws-scs-s3-readonly` or one of the canonical IDs in the following list is not listed as a user, do the following:

a. Under Access for other AWS accounts, choose + Add account.

b. In the box, add the region's canonical id as shown in the following list:

- China (Beijing) and China (Ningxia) Regions:

  b14d6a125bdf69854ed8ef2e71d8a20b7c490f252229b806e514966e490b8d83

- AWS GovCloud (US) Region:

  40fa568277ad703bd160f66ae4f83fc9d5f0d06c2f1b5060ca22442ac3ef8be6

  Important
  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).

- All other regions:

  540804c33a284a299d2547575ce1010f2312ef3da9b3a053c8bc45bf233e4353

c. Set the permissions on the bucket by choosing Yes for:

   i. Read object
   ii. Read object permissions

d. Choose Save.

7. Choose Overview.

8. Choose Download.

---

Step 5: 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. 156) or Creating a Redis Cluster with Replicas from Scratch (p. 260). 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, expand the **Advanced Redis settings** section and locate **Import data to cluster**. In the **Seed RDB file S3 location** box, type in the Amazon S3 path for the files(s). If you have multiple .rdb files, type in the path for each file in a comma separated list. 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 `create-cache-cluster` or `create-replication-group` operation, use the parameter `--snapshot-arns` to specify a fully qualified ARN for each .rdb file. For example, `arn:aws:s3:::myBucket/myFolder/myBackupFilename.rdb`. The ARN must resolve to the backup files you stored in Amazon S3.

- **Seed the ElastiCache Cluster With the .rdb File Data Using the ElastiCache API**

  If you use the `CreateCacheCluster` or `CreateReplicationGroup` ElastiCache API operation, use the parameter `SnapshotArns` to specify a fully qualified ARN for each .rdb file. For example, `arn:aws:s3:::myBucket/myFolder/myBackupFilename.rdb`. The ARN must resolve to the backup files 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. 460).
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.

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 Monitoring Costs with Cost Allocation Tags (p. 466).
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

2. In the navigation pane, choose Backups.
   
   The Backups screen appears with a list of your backups.
3. Choose the box to the left of the name of the backup you want to delete.
4. Choose Delete.
5. If you want to delete this backup, choose Delete on the Delete Backup confirmation screen. The status changes to deleting.

Deleting a Backup (AWS CLI)

Use the delete-snapshot AWS 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 CLI Command 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
```

API Version 2015-02-02
326
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. 84).

For more information see:

- Redis Specific Parameters (p. 362)
- Replication: Multi-AZ with Automatic Failover (Redis) (p. 240)
- Mitigating Failures (p. 83)
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. 388)

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

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

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

For more information about using ElastiCache with Amazon VPCs, see Amazon Virtual Private Cloud (Amazon VPC) with ElastiCache (p. 388).

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

Topics
- Creating a Security Group (p. 329)
- Listing Available Security Groups (p. 331)
- Viewing a Security Group (p. 333)
- Authorizing Network Access to an Amazon EC2 Security Group (p. 335)
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. 388).

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)

2. In the navigation pane, choose Security Groups.
4. In Create Security Group, type the name of the new security group in Security Group.
5. In Description, type a description for the new security group.
6. Choose Create.

Creating a Security Group (AWS CLI)

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

- --cache-security-group-name – The name of the security group you are creating.
  
  Example: mysecuritygroup

- --description – A description for this security group.
  
  Example: "My new security group"

For Linux, macOS, or Unix:

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

For Windows:

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

For more information, see create-cache-security-group.

Creating a Security Group (ElastiCache API)

Using the ElastiCache API operation CreateCacheSecurityGroup with the following parameters:

- CacheSecurityGroupName – The name of the security group you are creating.
  
  Example: mysecuritygroup
• Description – A URL encoded description for this security group.

   **Example:** My%20security%20group

**Example**

Line breaks are added for ease of reading.

```plaintext
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. 388).

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)

2. In the navigation pane, choose Security Groups.

   The available security groups appear in the Security Groups list.

Listing Available Security Groups (AWS CLI)

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

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

JSON output from this command will look something like this.

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

For more information, see describe-cache-security-groups.

Listing Available Security Groups (ElastiCache API)

Using the ElastiCache API, call DescribeCacheSecurityGroups.

**Example**

Line breaks are added for ease of reading.

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

API Version 2015-02-02

331
|---------------------------------------------------------------|
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. 388).

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)

2. In the navigation pane, choose Security Groups.
   The available cache security groups appear in the Security Groups list.
3. Choose a cache security group from the Security Groups list.
   The list of authorizations defined for the security group appears in the detail section at the bottom of the window.

Viewing a Security Group (AWS CLI)

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

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

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

JSON output from this command will look something like this.

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

For more information, see describe-cache-security-groups.

Viewing a Security Group (ElastiCache API)

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

API Version 2015-02-02

333
- `CacheSecurityGroupName` – the name of the cache security group to return details for.

**Example**

Line breaks are added for ease of reading.

```plaintext
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-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. 388).

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)**

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

**Authorizing Network Access to an Amazon EC2 Security Group (AWS CLI)**

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

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

**Example**

For Linux, macOS, or Unix:

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

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

The command should produce output similar to the following:

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

For more information, see `authorize-cache-security-group-ingress`.

**Authorizing Network Access to an Amazon EC2 Security Group (ElastiCache API)**

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

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

**Example**

```
https://elasticache.us-west-2.amazonaws.com/
  ?Action=AuthorizeCacheSecurityGroupIngress
  &EC2SecurityGroupOwnerId=987654321021
  &EC2SecurityGroupName=myec2group
  &Version=2015-02-02
  &SignatureVersion=4
  &SignatureMethod=HmacSHA256
  &Timestamp=20150202T220302Z
  &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Date=20150202T220302Z
  &X-Amz-SignedHeaders=Host
  &X-Amz-Expires=20150202T220302Z
  &X-Amz-Credential=<credential>
  &X-Amz-Signature=<signature>
```

For more information, see `AuthorizeCacheSecurityGroupIngress`.

API Version 2015-02-02

336
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. 353) and Redis Specific Parameters (p. 362).

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. 360) and Redis Node-Type Specific Parameters (p. 377).

Topics

- Parameter Management (p. 338)
- Cache Parameter Group Tiers (p. 339)
- Creating a Parameter Group (p. 340)
- Listing Parameter Groups by Name (p. 343)
- Listing a Parameter Group's Values (p. 346)
- Modifying a Parameter Group (p. 348)
- Deleting a Parameter Group (p. 351)
- Memcached Specific Parameters (p. 353)
- Redis Specific Parameters (p. 362)
Parameter Management

Parameters are grouped together into named parameter groups for easier parameter management. A parameter group represents a combination of specific values for the parameters that are passed to the engine software during startup. These values determine how the engine processes on each node will behave at runtime. The parameter values on a specific parameter group apply to all nodes that are associated with the group, regardless of which cluster they belong to.

To fine tune your cluster's performance, you can modify some parameter values or change the cluster's parameter group.

Constraints

- You cannot modify or delete the default parameter groups. If you need custom parameter values, you must create a custom parameter group.
- The parameter group family and the cluster you're assigning it to must be compatible. For example, if your cluster is running 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 running Redis 3.2.4 or later you have the option of running it as Redis (cluster mode disabled) or Redis (cluster mode enabled).
  - To run Redis (cluster mode disabled) use the parameter group `default.redis3.2` or one derived from it.
  - To run Redis (cluster mode enabled) use the parameter group `default.redis3.2.cluster.on` or one derived from it.
- 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. 353) and Redis Specific Parameters (p. 362). For information on rebooting a cluster, go to Rebooting a Cluster (p. 182).
Cache Parameter Group Tiers

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

Amazon ElastiCache parameter group tiers

**Global Default**

The top-level root parameter group for all Amazon ElastiCache customers in the region.

The global default cache parameter group:

- Is reserved for ElastiCache and not available to the customer.

**Customer Default**

A copy of the Global Default cache parameter group which is created for the customer's use.

The Customer Default cache parameter group:

- Is created and owned by ElastiCache.
- Is available to the customer for use as a cache parameter group for any clusters running an engine version supported by this cache parameter group. For example, default.redis2.8 supports Redis engine versions 2.8.x.
- Cannot be edited by the customer.

**Customer Owned**

A copy of the Customer Default cache parameter group. A Customer Owned cache parameter group is created whenever the customer creates a cache parameter group.

The Customer Owned cache parameter group:

- Is created and owned by the customer.
- Can be assigned to any of the customer's compatible clusters. For example, a cache parameter group created with the family redis2.8 is compatible with any cluster running Redis 2.8.x.
- Can be modified by the customer to create a custom cache parameter group.

Not all parameter values can be modified. For more information, see either Memcached Specific Parameters (p. 353) or Redis Specific Parameters (p. 362).
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

2. To see a list of all available parameter groups, in the left hand navigation pane choose Parameter Groups.
3. To create a parameter group, choose Create Parameter Group.

The Create Parameter Group screen will appear.

4. From the Family list, choose the parameter group family that will be the template for your parameter group.

The parameter group family, such as 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 choose 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, choose Create.

To terminate the process without creating the parameter group, choose Cancel.

8. When the parameter group is created, it will have the family's default values. To change the default values you must modify the parameter group. For more information, see Modifying a Parameter Group (p. 348).

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, macOS, 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": {
      "CacheParameterGroupName": "myredis28",
      "CacheParameterGroupFamily": "redis2.8",
      "Description": "My first parameter group"
   }
}
```

When the parameter group is created, it will have the family’s default values. To change the default values you must modify the parameter group. For more information, see Modifying a Parameter Group (p. 348).

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.

```plaintext
https://elasticache.us-west-2.amazonaws.com/
?Action=CreateCacheParameterGroup
&CacheParameterGroupFamily=redis2.8
&CacheParameterGroupName=myRedis28
&Description=My first parameter group
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&Version=2015-02-02
&X-Amz-Credential=<credential>
```

The response from this action should look something like this.

```xml
  <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. 348).

For more information, see *CreateCacheParameterGroup*. 

---

**API Version 2015-02-02**

342
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

2. To see a list of all available parameter groups, in the left hand navigation pane choose Parameter Groups.

Listing Parameter Groups by Name (AWS CLI)

To generate a list of parameter groups using the AWS CLI, use the command `describe-cache-parameter-groups`. If you provide a parameter group's name, only that parameter group will be listed. If you do not provide a parameter group's name, up to `--max-records` parameter groups will be listed. In either case, the parameter group's name, family, and description are listed.

The following sample code lists the parameter group `myRedis28`.

For Linux, macOS, 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.

```
{
  "CacheParameterGroups": [
  {
    "CacheParameterGroupName": "myredis28",
    "CacheParameterGroupFamily": "redis2.8",
    "Description": "My first parameter group"
  }
  ]
}
```

The following sample code lists up to 10 parameter groups.

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

The JSON output of this command will look something like this, listing the name, family, and description for each parameter group.

```
{

```
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.

```xml
</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.

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

2. To see a list of all available parameter groups, in the left hand navigation pane choose Parameter Groups.
3. Choose the parameter group for which you want to list the parameters and values by choosing the box to the left of the parameter group's name.

The parameters and their values will be listed at the bottom of the screen. Due to the number of parameters, you may have to scroll up and down to find the parameter you're interested in.

Listing a Parameter Group's Values (AWS CLI)

To list a parameter group's parameters and their values using the AWS CLI, use the command `describe-cache-parameters`.

The following sample code list all the parameters and their values for the parameter group `myRedis28`.

For Linux, macOS, or Unix:

```bash
aws elasticache describe-cache-parameters
   --cache-parameter-group-name myRedis28
```

For Windows:

```bash
aws elasticache describe-cache-parameters
   --cache-parameter-group-name myRedis28
```

The output of this command will look something like this.

```json
{
   "Parameters": [
   {
       "Description": "Apply rehashing or not.",
       "DataType": "string",
       "ChangeType": "requires-reboot",
       "IsModifiable": true,
       "AllowedValues": "yes,no",
       "Source": "system",
       "ParameterValue": "yes",
       "ParameterName": "activerehashing",
       "MinimumEngineVersion": "2.8.6"
   },
   (...sample truncated...)
   {
       "Description": "Enable Redis persistence."
   }
}
```
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`.

```plaintext
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.

```xml
  <DescribeCacheParametersResult>
    <CacheClusterClassSpecificParameters>
      <CacheNodeTypeSpecificParameter>
        <DataType>integer</DataType>
        <Source>system</Source>
        <IsModifiable>false</IsModifiable>
        <Description>The maximum configurable amount of memory to use to store items, in megabytes.</Description>
        <CacheNodeTypeSpecificValues>
          <CacheNodeTypeSpecificValue>
            <Value>1000</Value>
            <CacheClusterClass>cache.c1.medium</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>6000</Value>
            <CacheClusterClass>cache.c1.xlarge</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>7100</Value>
            <CacheClusterClass>cache.m1.large</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>1300</Value>
            <CacheClusterClass>cache.m1.small</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
        </CacheNodeTypeSpecificValues>
        <AllowedValues>1-100000</AllowedValues>
      </CacheNodeTypeSpecificParameter>
    </CacheClusterClassSpecificParameters>
  </DescribeCacheParametersResult>
</DescribeCacheParametersResponse>
```

For more information, see `describe-cache-parameters`.
Modifying a Parameter Group

Important
You cannot modify any default parameter group.

For more information, see DescribeCacheParameters.
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. 353) and Redis Specific Parameters (p. 362).

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

2. To see a list of all available parameter groups, in the left hand navigation pane choose Parameter Groups.
3. Choose the parameter group you want to modify by choosing the box to the left of the parameter group's name.
   
   The parameter group's parameters will be listed at the bottom of the screen. You may need to page through the list to see all the parameters.
4. To modify one or more parameters, choose Edit Parameters.
5. In the Edit Parameter Group: screen, scroll using the left and right arrows until you find the `binding_protocol` parameter, then type `ascii` in the Value column.
6. Choose Save Changes.
7. Find the name of the parameter you changed in one of these topics:
   - Memcached Specific Parameters (p. 353)
   - Redis Specific Parameters (p. 362)

    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. 182).

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 two parameters, `reserved-memory-percent` and `cluster-enabled` on the parameter group `myredis32-on-30`. We set `reserved-memory-percent` to 30 (30 percent) and `cluster-enabled` to yes so that the parameter group can be used with Redis (cluster mode enabled) clusters (replication groups).

For Linux, macOS, or Unix:

```bash
aws elasticache modify-cache-parameter-group \
    --cache-parameter-group-name myredis32-on-30 \
    --parameter-name-values \
        ParameterName=reserved-memory-percent,ParameterValue=30 \
        ParameterName=cluster-enabled,ParameterValue=yes
```

For Windows:

```bash
aws elasticache modify-cache-parameter-group ^
```
Modifying a Parameter Group (ElastiCache API)

To change a parameter group’s parameter values using the ElastiCache API, use the `ModifyCacheParameterGroup` action.

The following sample code sets the value of two parameters, `reserved-memory-percent` and `cluster-enabled` on the parameter group `myredis32-on-30`. We set `reserved-memory-percent` to 30 (30 percent) and `cluster-enabled` to `yes` so that the parameter group can be used with Redis (cluster mode enabled) clusters (replication groups).

```
https://elasticache.us-west-2.amazonaws.com/
?Action=ModifyCacheParameterGroup
&CacheParameterGroupName=myredis32-on-30
&ParameterNameValues.member.1.ParameterName=reserved-memory-percent
&ParameterNameValues.member.1.ParameterValue=30
&ParameterNameValues.member.2.ParameterName=cluster-enabled
&ParameterNameValues.member.2.ParameterValue=yes
&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. 182)`.

Find the name of the parameter you changed in one of these topics:
- Memcached Specific Parameters (p. 353)
- Redis Specific Parameters (p. 362)

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. 182)`.
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

2. To see a list of all available parameter groups, in the left hand navigation pane choose Parameter Groups.
3. Choose the parameter groups you want to delete by choosing the box to the left of the parameter group's name.
   The Delete button will become active.
4. Choose Delete.
   The Delete Parameter Groups confirmation screen will appear.
5. To delete the parameter groups, on the Delete Parameter Groups confirmation screen, choose Delete.
   To keep the parameter groups, choose Cancel.

Deleting a Parameter Group (AWS CLI)

To delete a parameter group using the AWS CLI, use the command delete-cache-parameter-group. For the parameter group to delete, the parameter group specified by --cache-parameter-group-name cannot have any clusters associated with it, nor can it be a default parameter group.

The following sample code deletes the myRedis28 parameter group.

For Linux, macOS, or Unix:

```bash
aws elasticache delete-cache-parameter-group \ 
   --cache-parameter-group-name myRedis28
```

For Windows:

```bash
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](https://elasticache.us-west-2.amazonaws.com/)

API Version 2015-02-02
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 1.4.34 Added Parameters (p. 353)
- Memcached 1.4.33 Added Parameters (p. 353)
- Memcached 1.4.24 Added Parameters (p. 355)
- Memcached 1.4.14 Added Parameters (p. 356)
- Memcached 1.4.5 Supported Parameters (p. 357)
- Memcached Connection Overhead (p. 359)
- Memcached Node-Type Specific Parameters (p. 360)

Memcached 1.4.34 Added Parameters

For Memcached 1.4.34, no additional parameters are supported.

Parameter group family: memcached1.4

Memcached 1.4.33 Added Parameters

For Memcached 1.4.33, the following additional parameters are supported.

Parameter group family: memcached1.4

<table>
<thead>
<tr>
<th>Name</th>
<th>Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>modern</td>
<td>Default: enabled</td>
<td>An alias to multiple features. Enabling modern is equivalent to turning following commands on and using a murmur3 hash algorithm: slab_reassign, slab_automove, lru_crawler, lru_maintainer, maxconns_fast, and hash_algorithm=murmur3.</td>
</tr>
<tr>
<td></td>
<td>Type: boolean</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Changes Take Effect: At launch</td>
<td></td>
</tr>
<tr>
<td>watch</td>
<td>Default: enabled</td>
<td>Logs fetches, evictions or mutations. When, for example, user turns watch on, they can see logs when get, set, delete, or update occur.</td>
</tr>
<tr>
<td></td>
<td>Type: boolean</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Changes Take Effect: Immediately</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Logs can get dropped if user hits their watcher_logbuf_size and worker_logbuf_size limits.</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Details</td>
<td>Description</td>
</tr>
<tr>
<td>--------------</td>
<td>--------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>idle_timeout</td>
<td>Default: 0 (disabled)</td>
<td>The minimum number of seconds a client will be allowed to idle before being asked to close. Range of values: 0 to 86400.</td>
</tr>
<tr>
<td></td>
<td>Type: integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Changes Take Effect: At Launch</td>
<td></td>
</tr>
<tr>
<td>cache_memlimit</td>
<td>Type: integer</td>
<td>If memory isn't being preallocated, allows dynamically increasing the memory limit of a running system. cache_memlimit N where N is a value in megabytes. Value can go up or down.</td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Changes Take Effect: Immediately</td>
<td></td>
</tr>
<tr>
<td>track_sizes</td>
<td>Default: disabled</td>
<td>Shows the sizes each slab group has consumed.</td>
</tr>
<tr>
<td></td>
<td>Type: boolean</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Changes Take Effect: At Launch</td>
<td></td>
</tr>
<tr>
<td>watcher_logbuf_size</td>
<td>Default: 256 (KB)</td>
<td>The watch command turns on stream logging for Memcached. However watch can drop logs if the rate of evictions, mutations or fetches are high enough to cause the logging buffer to become full. In such situations, users can increase the buffer size to reduce the chance of log losses.</td>
</tr>
<tr>
<td></td>
<td>Type: integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Changes Take Effect: At Launch</td>
<td></td>
</tr>
<tr>
<td>worker_logbuf_size</td>
<td>Default: 64 (KB)</td>
<td>The watch command turns on stream logging for Memcached. However watch can drop logs if the rate of evictions, mutations or fetches are high enough to cause the logging buffer get full. In such situations, users can increase the buffer size to reduce the chance of log losses.</td>
</tr>
<tr>
<td></td>
<td>Type: integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Changes Take Effect: At Launch</td>
<td></td>
</tr>
<tr>
<td>slab_chunk_max</td>
<td>Default: 524288 (bytes)</td>
<td>Specifies the maximum size of a slab. Setting smaller slab size uses memory more efficiently. Items larger than slab_chunk_max are split over multiple slabs.</td>
</tr>
<tr>
<td></td>
<td>Type: integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Changes Take Effect: At Launch</td>
<td></td>
</tr>
</tbody>
</table>
Memcached 1.4.24 Added Parameters

For Memcached 1.4.24, the following additional parameters are supported.

**Parameter group family:** memcached1.4

<table>
<thead>
<tr>
<th>Name</th>
<th>Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>disable_flush_all</td>
<td>Default: 0 (disabled)</td>
<td>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).</td>
</tr>
<tr>
<td>hash_algorithm</td>
<td>Default: jenkins</td>
<td>The hash algorithm to be used. Permitted values: murmur3 and jenkins.</td>
</tr>
<tr>
<td>lru_crawler</td>
<td>Default: 0 (disabled)</td>
<td>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</td>
</tr>
</tbody>
</table>

**Note**

You can temporarily enable lru_crawler at runtime from the command line. For more information, see the Description column.
### Memcached 1.4.14 Added Parameters

For Memcached 1.4.14, the following additional parameters are supported.

**Parameter group family:** memcached1.4

**Parameters added in Memcached 1.4.14**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>config_max</td>
<td>The maximum number of ElastiCache configuration entries.</td>
</tr>
<tr>
<td>config_size_max</td>
<td>The maximum size of the configuration entries, in bytes.</td>
</tr>
<tr>
<td>hashpower_init</td>
<td>The initial size of the ElastiCache hash table, expressed as a power of two.</td>
</tr>
</tbody>
</table>

#### config_max

- **Default:** 16
- **Type:** integer
- **Modifiable:** No

#### config_size_max

- **Default:** 65536
- **Type:** integer
- **Modifiable:** No

#### hashpower_init

- **Default:** The default is 16 ($2^{16}$), or 65536 keys.
Memcached 1.4.5 Supported Parameters

Parameter group family: memcached1.4

For Memcached 1.4.5, the following parameters are supported.

Parameters added in Memcached 1.4.5

<table>
<thead>
<tr>
<th>Name</th>
<th>Default</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>backlog_queue</td>
<td></td>
<td>The backlog queue limit.</td>
</tr>
<tr>
<td>binding_protocol</td>
<td>auto</td>
<td>The binding protocol.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Permissible values are: ascii and auto.</td>
</tr>
<tr>
<td>Name</td>
<td>Details</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>---------------------------------------------</td>
<td>----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cas_disabled</td>
<td>Default: 0 (false)</td>
<td>If 1 (true), check and set (CAS) operations will be disabled, and items stored will consume 8 fewer bytes than with CAS enabled.</td>
</tr>
<tr>
<td>chunk_size</td>
<td>Default: 48</td>
<td>The minimum amount, in bytes, of space to allocate for the smallest item's key, value, and flags.</td>
</tr>
<tr>
<td>chunk_size_growth_factor</td>
<td>Default: 1.25</td>
<td>The growth factor that controls the size of each successive Memcached chunk; each chunk will be <code>chunk_size_growth_factor</code> times larger than the previous chunk.</td>
</tr>
<tr>
<td>error_on_memory_exhausted</td>
<td>Default: 0 (false)</td>
<td>If 1 (true), when there is no more memory to store items, Memcached will return an error rather than evicting items.</td>
</tr>
<tr>
<td>large_memory_pages</td>
<td>Default: 0 (false)</td>
<td>If 1 (true), ElastiCache will try to use large memory pages.</td>
</tr>
<tr>
<td>lock_down_paged_memory</td>
<td>Default: 0 (false)</td>
<td>If 1 (true), ElastiCache will lock down all paged memory.</td>
</tr>
<tr>
<td>max_item_size</td>
<td>Default: 1048576</td>
<td>The size, in bytes, of the largest item that can be stored in the cluster.</td>
</tr>
</tbody>
</table>
### 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.

<table>
<thead>
<tr>
<th>Name</th>
<th>Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>max_simultaneous_connections</code></td>
<td></td>
<td>The maximum number of simultaneous connections.</td>
</tr>
<tr>
<td><code>maximize_core_file_limit</code></td>
<td></td>
<td>If 1 (true), ElastiCache will maximize the core file limit.</td>
</tr>
<tr>
<td><code>memcached_connections_overhead</code></td>
<td></td>
<td>The amount of memory to be reserved for Memcached connections and other miscellaneous overhead. For information about this parameter, see Memcached Connection Overhead (p. 359).</td>
</tr>
<tr>
<td><code>requests_per_event</code></td>
<td></td>
<td>The maximum number of requests per event for a given connection. This limit is required to prevent resource starvation.</td>
</tr>
</tbody>
</table>
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

<table>
<thead>
<tr>
<th>Node Type</th>
<th>max_cache_memory (MiB)</th>
<th>num-threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache.t1.micro</td>
<td>213</td>
<td>1</td>
</tr>
<tr>
<td>cache.t2.micro</td>
<td>555</td>
<td>1</td>
</tr>
<tr>
<td>cache.t2.small</td>
<td>1588</td>
<td>1</td>
</tr>
<tr>
<td>cache.t2.medium</td>
<td>3301</td>
<td>2</td>
</tr>
<tr>
<td>cache.m1.small</td>
<td>1300</td>
<td>1</td>
</tr>
<tr>
<td>cache.m1.medium</td>
<td>3350</td>
<td>1</td>
</tr>
<tr>
<td>cache.m1.large</td>
<td>7100</td>
<td>2</td>
</tr>
<tr>
<td>cache.m1.xlarge</td>
<td>14600</td>
<td>4</td>
</tr>
<tr>
<td>cache.m2.xlarge</td>
<td>16700</td>
<td>2</td>
</tr>
<tr>
<td>cache.m2.2xlarge</td>
<td>33800</td>
<td>4</td>
</tr>
<tr>
<td>cache.m2.4xlarge</td>
<td>68000</td>
<td>8</td>
</tr>
<tr>
<td>cache.m3.medium</td>
<td>2850</td>
<td>1</td>
</tr>
<tr>
<td>cache.m3.large</td>
<td>6200</td>
<td>2</td>
</tr>
<tr>
<td>cache.m3.xlarge</td>
<td>13600</td>
<td>4</td>
</tr>
<tr>
<td>cache.m3.2xlarge</td>
<td>28600</td>
<td>8</td>
</tr>
<tr>
<td>cache.m4.large</td>
<td>6573</td>
<td>2</td>
</tr>
<tr>
<td>cache.m4.xlarge</td>
<td>14618</td>
<td>4</td>
</tr>
<tr>
<td>cache.m4.2xlarge</td>
<td>30412</td>
<td>8</td>
</tr>
<tr>
<td>cache.m4.4xlarge</td>
<td>62234</td>
<td>16</td>
</tr>
<tr>
<td>cache.m4.10xlarge</td>
<td>158355</td>
<td>40</td>
</tr>
<tr>
<td>cache.c1.xlarge</td>
<td>6600</td>
<td>8</td>
</tr>
<tr>
<td>cache.r3.large</td>
<td>13800</td>
<td>2</td>
</tr>
<tr>
<td>cache.r3.xlarge</td>
<td>29100</td>
<td>4</td>
</tr>
<tr>
<td>cache.r3.2xlarge</td>
<td>59600</td>
<td>8</td>
</tr>
<tr>
<td>cache.r3.4xlarge</td>
<td>120600</td>
<td>16</td>
</tr>
<tr>
<td>cache.r3.8xlarge</td>
<td>242600</td>
<td>32</td>
</tr>
<tr>
<td>cache.r4.large</td>
<td>12590</td>
<td>2</td>
</tr>
</tbody>
</table>
### Memcached Node-Type Specific Parameters

<table>
<thead>
<tr>
<th>Node Type</th>
<th>max_cache_memory (MiB)</th>
<th>num-threads</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache.r4.xlarge</td>
<td>25652</td>
<td>4</td>
</tr>
<tr>
<td>cache.r4.2xlarge</td>
<td>51686</td>
<td>8</td>
</tr>
<tr>
<td>cache.r4.4xlarge</td>
<td>103815</td>
<td>16</td>
</tr>
<tr>
<td>cache.r4.8xlarge</td>
<td>208144</td>
<td>32</td>
</tr>
<tr>
<td>cache.r4.16xlarge</td>
<td>416776</td>
<td>64</td>
</tr>
</tbody>
</table>

**Note**

All T2 instances are created in an Amazon Virtual Private Cloud (Amazon VPC).
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, default.redis2.8, or default.redis3.2. 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.

Contents

- Redis 3.2.10 Parameter Changes (p. 363)
- Redis 3.2.6 Parameter Changes (p. 363)
- Redis 3.2.4 Parameter Changes (p. 363)
  - New Parameters for Redis 3.2.4 (p. 363)
  - Parameters Changed in Redis 3.2.4 (Enhanced) (p. 365)
- Redis 2.8.24 (Enhanced) Added Parameters (p. 366)
- Redis 2.8.23 (Enhanced) Added Parameters (p. 366)
  - How close-on-slave-write works (p. 366)
  - Why disable close-on-slave-write? (p. 368)
- Redis 2.8.22 (Enhanced) Added Parameters (p. 368)
- Redis 2.8.21 Added Parameters (p. 368)
- Redis 2.8.19 Added Parameters (p. 368)
- Redis 2.8.6 Added Parameters (p. 368)
- Redis 2.6.13 Parameters (p. 371)
- Redis Node-Type Specific Parameters (p. 377)

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 when using the ElastiCache console. We recommend against using these Redis versions. If you need to use one of them, work with the AWS CLI or ElastiCache API.

For more information, see the following topics:

<table>
<thead>
<tr>
<th></th>
<th>AWS CLI</th>
<th>ElastiCache API</th>
</tr>
</thead>
<tbody>
<tr>
<td>Create Cluster</td>
<td>Creating a Cache Cluster (AWS CLI) (p. 168)</td>
<td>Creating a Cache Cluster (ElastiCache API) (p. 170)</td>
</tr>
<tr>
<td></td>
<td>This action cannot be used to create a replication group with cluster mode enabled.</td>
<td>This action cannot be used to create a replication group with cluster mode enabled.</td>
</tr>
<tr>
<td>Modify Cluster</td>
<td>Modifying a Cache Cluster (AWS CLI) (p. 180)</td>
<td>Modifying a Cache Cluster (ElastiCache API) (p. 181)</td>
</tr>
<tr>
<td></td>
<td>This action cannot be used to create a replication group with cluster mode enabled.</td>
<td>This action cannot be used to create a replication group with cluster mode enabled.</td>
</tr>
<tr>
<td>Create Replication Group</td>
<td>Creating a Redis (cluster mode disabled) Cluster with Replicas from Scratch (AWS CLI) (p. 261)</td>
<td>Creating a Redis (cluster mode disabled) Cluster with Replicas from Scratch (ElastiCache API) (p. 264)</td>
</tr>
</tbody>
</table>
Redis 3.2.10 Parameter Changes

Parameter group family: redis3.2

ElastiCache for Redis 3.2.10 there are no additional parameters supported.

Redis 3.2.6 Parameter Changes

Parameter group family: redis3.2

For Redis 3.2.6 there are no additional parameters supported.

Redis 3.2.4 Parameter Changes

Parameter group family: redis3.2

Beginning with Redis 3.2.4 there are two default parameter groups.

- default.redis3.2 – When running 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 (p. 363)
- Parameters Changed in Redis 3.2.4 (Enhanced) (p. 365)

New Parameters for Redis 3.2.4

Parameter group family: redis3.2

For Redis 3.2.4 the following additional parameters are supported.

<table>
<thead>
<tr>
<th>Name</th>
<th>Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>list-max-ziplist-size</td>
<td>Default: -2</td>
<td>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</td>
</tr>
<tr>
<td></td>
<td>Type: integer</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Details</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-----------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Modifiable: No</td>
<td>or a maximum number of elements. For a fixed maximum size, use -5 through -1, meaning:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• -5: max size: 64 Kb - not recommended for normal workloads</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• -4: max size: 32 Kb - not recommended</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• -3: max size: 16 Kb - not recommended</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• -2: max size: 8 Kb - recommended</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• -1: max size: 4 Kb - recommended</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Positive numbers mean store up to exactly that number of elements per list node.</td>
</tr>
<tr>
<td>list-compress-depth</td>
<td>Default: 0</td>
<td>Lists may also be compressed. Compress depth is the number of quicklist ziplist nodes from each side of the list to exclude from compression.</td>
</tr>
<tr>
<td></td>
<td>Type: integer</td>
<td>The head and tail of the list are always uncompressed for fast push and pop operations.</td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td>Settings are:</td>
</tr>
<tr>
<td></td>
<td>Changes Take Effect: Immediately</td>
<td>• 0: Disable all compression.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 1: Start compressing with the 1st node in from the head and tail.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[head]-&gt;node-&gt;node-&gt;...-&gt;node-&gt;[tail]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>All nodes except [head] and [tail] compress.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• 2: Start compressing with the 2nd node in from the head and tail.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[head]-&gt;[next]-&gt;node-&gt;node-&gt;...-&gt;node-&gt;[prev]-&gt;[tail]</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[head], [next], [prev], [tail] do not compress. All other nodes compress.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Etc.</td>
</tr>
<tr>
<td>cluster-enabled</td>
<td>Default: yes/no</td>
<td>Indicates whether this is a Redis (cluster mode enabled) replication group in cluster mode (yes) or</td>
</tr>
<tr>
<td></td>
<td>Type: boolean</td>
<td>a Redis (cluster mode enabled) replication group in non-cluster mode (no). Redis (cluster mode enabled) replication groups in cluster mode can</td>
</tr>
<tr>
<td></td>
<td>Modifiable: No</td>
<td>partition their data across up to 15 node groups.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Redis 3.2.x has two default parameter groups.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• default.redis3.2 – default value no.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• default.redis3.2.cluster.on – default value yes.</td>
</tr>
<tr>
<td>Name</td>
<td>Details</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cluster-require-full-coverage</td>
<td>Default: no</td>
<td>When set to <code>yes</code>, 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 <code>cluster-require-full-coverage</code> option to <code>no</code>.</td>
</tr>
<tr>
<td>hll-sparse-max-bytes</td>
<td>Default: 3000</td>
<td>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 <code>PFADD</code> 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.</td>
</tr>
<tr>
<td>reserved-memory-percent</td>
<td>Default: 25</td>
<td>The percent of a node's memory reserved for non-data use. By default, the Redis data footprint grows until it consumes all of the node's memory. If this occurs, then node performance will likely suffer due to excessive memory paging. Byreserving memory, you can set aside some of the available memory for non-Redis purposes to help reduce the amount of paging. This parameter is specific to ElastiCache, and is not part of the standard Redis distribution. For more information, see <code>reserved-memory</code> and <code>Managing Reserved Memory (Redis)</code> (p. 79).</td>
</tr>
</tbody>
</table>

**Parameters Changed in Redis 3.2.4 (Enhanced)**

**Parameter group family:** redis3.2

For Redis 3.2.4 the following parameters were changed.
<table>
<thead>
<tr>
<th>Name</th>
<th>Details</th>
<th>Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>appendonly</td>
<td>Default: off</td>
<td>If you want to upgrade from an earlier Redis version, you must first turn appendonly off.</td>
</tr>
<tr>
<td></td>
<td>Modifiable: No</td>
<td></td>
</tr>
<tr>
<td>appendfsync</td>
<td>Default: off</td>
<td>If you want to upgrade from an earlier Redis version, you must first turn appendfsync off.</td>
</tr>
<tr>
<td></td>
<td>Modifiable: No</td>
<td></td>
</tr>
<tr>
<td>repl-timeout</td>
<td>Default: 60</td>
<td>Is now unmodifiable with a default of 60.</td>
</tr>
<tr>
<td></td>
<td>Modifiable: No</td>
<td></td>
</tr>
<tr>
<td>tcp-keepalive</td>
<td>Default: 300</td>
<td>Default was 0.</td>
</tr>
<tr>
<td>list-max-ziplist-entries</td>
<td></td>
<td>Parameter is no longer available.</td>
</tr>
<tr>
<td>list-max-ziplist-value</td>
<td></td>
<td>Parameter is no longer available.</td>
</tr>
</tbody>
</table>

**Redis 2.8.24 (Enhanced) Added Parameters**

*Parameter group family: redis2.8*

For Redis 2.8.24 there are no additional parameters supported.

**Redis 2.8.23 (Enhanced) Added Parameters**

*Parameter group family: redis2.8*

For Redis 2.8.23 the following additional parameter is supported.

<table>
<thead>
<tr>
<th>Name</th>
<th>Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>close-on-slave-write</td>
<td>Default: yes</td>
<td>If enabled, clients who attempt to write to a read-only replica will be disconnected.</td>
</tr>
<tr>
<td></td>
<td>Type: string (yes/no)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Changes Take Effect: Immediately</td>
<td></td>
</tr>
</tbody>
</table>

**How close-on-slave-write works**

The close-on-slave-write parameter is introduced by Amazon 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 failing cluster, 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 might prefer to keep your clients connected so as to not degrade read performance.

Redis 2.8.22 (Enhanced) Added Parameters

Parameter group family: redis2.8

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

Parameter group family: redis2.8

For Redis 2.8.21, there are no additional parameters supported.

Redis 2.8.19 Added Parameters

Parameter group family: redis2.8

For Redis 2.8.19 there are no additional parameters supported.

Redis 2.8.6 Added Parameters

Parameter group family: redis2.8

For Redis 2.8.6 the following additional parameters are supported.

<table>
<thead>
<tr>
<th>Name</th>
<th>Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>min-slaves-max-lag</td>
<td>Default: 10 Type: integer</td>
<td>The number of seconds within which the primary node must receive a ping request from a read replica. If</td>
</tr>
<tr>
<td>Name</td>
<td>Details</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------</td>
<td>--------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Changes Take Effect:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>min-slaves-to-write</td>
<td>Default: 0</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Type: integer</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Changes Take Effect:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>Name</td>
<td>Details</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>-------------------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>notify-keyspace-events</td>
<td>Default: (an empty string)</td>
<td>The types of keyspace events that Redis can notify clients of. Each event</td>
</tr>
<tr>
<td></td>
<td>Type: string</td>
<td>type is represented by a single letter:</td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td>• K — Keyspace events, published with a prefix of <strong>keyspace@&lt;db&gt;</strong></td>
</tr>
<tr>
<td></td>
<td>Changes Take Effect: Immediately</td>
<td>• E — Key-event events, published with a prefix of <strong>keyevent@&lt;db&gt;</strong></td>
</tr>
<tr>
<td></td>
<td></td>
<td>• g — Generic, non-specific commands such as <code>DEL</code>, <code>EXPIRE</code>, <code>RENAME</code>, etc.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• $ — String commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• l — List commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• s — Set commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• h — Hash commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• z — Sorted set commands</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• x — Expired events (events generated every time a key expires)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• e — Evicted events (events generated when a key is evicted for maxmemory)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• A — An alias for <code>g$lshzxe</code></td>
</tr>
</tbody>
</table>

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.
<table>
<thead>
<tr>
<th>Name</th>
<th>Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>repl-backlog-size</td>
<td>Default: 1048576 Type: integer Modifiable: Yes Changes Take Effect: Immediately</td>
<td>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. <strong>Note</strong> Beginning with Redis 2.8.22, this parameter applies to the primary cluster as well as the read replicas.</td>
</tr>
<tr>
<td>repl-backlog-ttl</td>
<td>Default: 3600 Type: integer Modifiable: Yes Changes Take Effect: Immediately</td>
<td>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.</td>
</tr>
<tr>
<td>repl-timeout</td>
<td>Default: 60 Type: integer Modifiable: Yes Changes Take Effect: Immediately</td>
<td>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</td>
</tr>
</tbody>
</table>

**Redis 2.6.13 Parameters**

*Parameter group family: redis2.6*

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.
<table>
<thead>
<tr>
<th>Name</th>
<th>Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>activerehashing</td>
<td>Default: yes</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Type: string (yes/no)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modifiable: At Creation</td>
<td>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.</td>
</tr>
<tr>
<td>appendonly</td>
<td>Default: no</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Type: string</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td>Changes Take Effect: Immediately</td>
</tr>
<tr>
<td></td>
<td></td>
<td>The default value is no, meaning AOF is turned off. Set this parameter to yes to enable AOF. For more information, see Mitigating Failures (p. 83).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Note For Multi-AZ replication groups, AOF is not allowed.</td>
</tr>
<tr>
<td>appendfsync</td>
<td>Default: everysec</td>
<td>Controls how often the AOF output buffer is written to disk:</td>
</tr>
<tr>
<td></td>
<td>Type: string</td>
<td>• no — the buffer is flushed to disk on an as-needed basis.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• everysec — the buffer is flushed once per second. This is the default.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• always — the buffer is flushed every time that data in the cluster is modified.</td>
</tr>
<tr>
<td></td>
<td>Modifiable: Yes</td>
<td>Changes Take Effect: Immediately</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Important Some aspects value of this parameter changed in Redis version 3.2.4. See Parameters Changed in Redis 3.2.4 (Enhanced) (p. 365).</td>
</tr>
<tr>
<td>client-output-</td>
<td>Default: 0</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>buffer-limit-</td>
<td>Type: integer</td>
<td>Changes Take Effect: Immediately</td>
</tr>
<tr>
<td>normal-hard-limit</td>
<td>Modifiable: Yes</td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Details</td>
<td>Description</td>
</tr>
<tr>
<td>------</td>
<td>---------</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| client-output-buffer-limit-normal-soft-limit | Default: 0  
Type: integer  
Modifiable: Yes  
Changes Take Effect: Immediately | 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). |
| client-output-buffer-limit-normal-soft-seconds | Default: 0  
Type: integer  
Modifiable: Yes  
Changes Take Effect: Immediately | 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). |
| client-output-buffer-limit-pubsub-hard-limit | Default: 33554432  
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. |
| 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. 377)  
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. 377)  
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`. |
<table>
<thead>
<tr>
<th>Name</th>
<th>Details</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>client-output-buffer-limit</td>
<td>Default: 60</td>
<td>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.</td>
</tr>
<tr>
<td>slave-soft-seconds</td>
<td>Type: integer</td>
<td></td>
</tr>
<tr>
<td>Modifiable: No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>databases</td>
<td>Default: 16</td>
<td>The number of logical partitions the databases is split into. We recommend keeping this value low.</td>
</tr>
<tr>
<td></td>
<td>Type: integer</td>
<td>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.</td>
</tr>
<tr>
<td>Modifiable: At Creation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hash-max-ziplist-entries</td>
<td>Default: 512</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Type: integer</td>
<td></td>
</tr>
<tr>
<td>Modifiable: Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changes Take Effect: Immediately</td>
<td></td>
<td></td>
</tr>
<tr>
<td>hash-max-ziplist-value</td>
<td>Default: 64</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Type: integer</td>
<td></td>
</tr>
<tr>
<td>Modifiable: Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changes Take Effect: Immediately</td>
<td></td>
<td></td>
</tr>
<tr>
<td>list-max-ziplist-entries</td>
<td>Default: 512</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Type: integer</td>
<td></td>
</tr>
<tr>
<td>Modifiable: Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changes Take Effect: Immediately</td>
<td></td>
<td></td>
</tr>
<tr>
<td>list-max-ziplist-value</td>
<td>Default: 64</td>
<td>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.</td>
</tr>
<tr>
<td></td>
<td>Type: integer</td>
<td></td>
</tr>
<tr>
<td>Modifiable: Yes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Changes Take Effect: Immediately</td>
<td></td>
<td></td>
</tr>
<tr>
<td>lua-time-limit</td>
<td>Default: 5000</td>
<td>The maximum execution time for a Lua script, in milliseconds, before ElastiCache takes action to stop the script.</td>
</tr>
<tr>
<td></td>
<td>Type: integer</td>
<td>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.</td>
</tr>
<tr>
<td>Modifiable: No</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Name</td>
<td>Details</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------</td>
<td>---------</td>
<td>-------------</td>
</tr>
<tr>
<td>maxclients</td>
<td>Default: 65000 Type: integer Modifiable: No</td>
<td>The maximum number of clients that can be connected at one time.</td>
</tr>
<tr>
<td>maxmemory-policy</td>
<td>Default: volatile-lru Type: string Modifiable: Yes Changes Take Effect: Immediately</td>
<td>The eviction policy for keys when maximum memory usage is reached. Valid values are: volatile-lru</td>
</tr>
<tr>
<td>maxmemory-samples</td>
<td>Default: 3 Type: integer Modifiable: Yes Changes Take Effect: Immediately</td>
<td>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.</td>
</tr>
<tr>
<td>reserved-memory</td>
<td>Default: 0 Type: integer Modifiable: Yes Changes Take Effect: Immediately</td>
<td>The total memory, in bytes, reserved for non-data usage. By default, the Redis node will grow until it consumes the node's maxmemory (see Redis Node-Type Specific Parameters (p. 377)). 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. This parameter is specific to ElastiCache, and is not part of the standard Redis distribution. For more information, see reserved-memory-percent and Managing Reserved Memory (Redis) (p. 79).</td>
</tr>
<tr>
<td>set-max-intset-entries</td>
<td>Default: 512 Type: integer Modifiable: Yes Changes Take Effect: Immediately</td>
<td>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.</td>
</tr>
<tr>
<td>slave-allow-chaining</td>
<td>Default: no Type: string Modifiable: No</td>
<td>Determines whether a read replica in Redis can have read replicas of its own.</td>
</tr>
<tr>
<td>Name</td>
<td>Details</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------</td>
<td>--------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>slowlog-log-slower-than</td>
<td>Default: 10000 Type: integer Modifiable: Yes Changes Take Effect: Immediately</td>
<td>The maximum execution time, in microseconds, for commands to be logged by the Redis Slow Log feature.</td>
</tr>
<tr>
<td>slowlog-max-len</td>
<td>Default: 128 Type: integer Modifiable: Yes Changes Take Effect: Immediately</td>
<td>The maximum length of the Redis Slow Log.</td>
</tr>
<tr>
<td>tcp-keepalive</td>
<td>Default: 0 Type: integer Modifiable: Yes Changes Take Effect: Immediately</td>
<td>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. <strong>Important</strong> Some aspects of this parameter changed in Redis version 3.2.4. See Parameters Changed in Redis 3.2.4 (Enhanced) (p. 365).</td>
</tr>
<tr>
<td>timeout</td>
<td>Default: 0 Type: integer Modifiable: Yes Changes Take Effect: Immediately</td>
<td>The number of seconds a node waits before timing out. Values are: • 0 – never disconnect an idle client. • 1–19 – invalid values. • ( \geq 20 ) – the number of seconds a node waits before disconnecting an idle client.</td>
</tr>
<tr>
<td>zset-max-ziplist-entries</td>
<td>Default: 128 Type: integer Modifiable: Yes Changes Take Effect: Immediately</td>
<td>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.</td>
</tr>
<tr>
<td>zset-max-ziplist-value</td>
<td>Default: 64 Type: integer Modifiable: Yes Changes Take Effect: Immediately</td>
<td>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.</td>
</tr>
</tbody>
</table>

**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.

<table>
<thead>
<tr>
<th>Node Type</th>
<th>maxmemory</th>
<th>client-output-buffer-limit-slave-hard-limit</th>
<th>client-output-buffer-limit-slave-soft-limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>cache.t1.micro</td>
<td>142606336</td>
<td>14260633</td>
<td>14260633</td>
</tr>
<tr>
<td>cache.t2.micro</td>
<td>581959680</td>
<td>58195968</td>
<td>58195968</td>
</tr>
<tr>
<td>cache.t2.small</td>
<td>1665138688</td>
<td>166513868</td>
<td>166513868</td>
</tr>
<tr>
<td>cache.t2.medium</td>
<td>3461349376</td>
<td>346134937</td>
<td>346134937</td>
</tr>
<tr>
<td>cache.m1.small</td>
<td>943718400</td>
<td>943718400</td>
<td>943718400</td>
</tr>
<tr>
<td>cache.m1.medium</td>
<td>3093299200</td>
<td>309329920</td>
<td>309329920</td>
</tr>
<tr>
<td>cache.m1.large</td>
<td>7025459200</td>
<td>702545920</td>
<td>702545920</td>
</tr>
<tr>
<td>cache.m1.xlarge</td>
<td>14889779200</td>
<td>1488977920</td>
<td>1488977920</td>
</tr>
<tr>
<td>cache.m2.xlarge</td>
<td>17091788800</td>
<td>1709178880</td>
<td>1709178880</td>
</tr>
<tr>
<td>cache.m2.2xlarge</td>
<td>35022438400</td>
<td>3502243840</td>
<td>3502243840</td>
</tr>
<tr>
<td>cache.m2.4xlarge</td>
<td>70883737600</td>
<td>7088373760</td>
<td>7088373760</td>
</tr>
<tr>
<td>cache.m3.medium</td>
<td>2988441600</td>
<td>309329920</td>
<td>309329920</td>
</tr>
<tr>
<td>cache.m3.large</td>
<td>6501171200</td>
<td>650117120</td>
<td>650117120</td>
</tr>
<tr>
<td>cache.m3.xlarge</td>
<td>14260633600</td>
<td>1426063360</td>
<td>1426063360</td>
</tr>
<tr>
<td>cache.m3.2xlarge</td>
<td>29989273600</td>
<td>2998927360</td>
<td>2998927360</td>
</tr>
<tr>
<td>cache.m4.large</td>
<td>6892593152</td>
<td>689259315</td>
<td>689259315</td>
</tr>
<tr>
<td>cache.m4.xlarge</td>
<td>15328501760</td>
<td>1532850176</td>
<td>1532850176</td>
</tr>
<tr>
<td>cache.m4.2xlarge</td>
<td>31889126359</td>
<td>3188912636</td>
<td>3188912636</td>
</tr>
<tr>
<td>cache.m4.4xlarge</td>
<td>65257290629</td>
<td>6525729063</td>
<td>6525729063</td>
</tr>
<tr>
<td>cache.m4.10xlarge</td>
<td>166047614239</td>
<td>16604761424</td>
<td>16604761424</td>
</tr>
<tr>
<td>cache.c1.xlarge</td>
<td>6501171200</td>
<td>650117120</td>
<td>650117120</td>
</tr>
<tr>
<td>cache.r3.large</td>
<td>14470348800</td>
<td>1468006400</td>
<td>1468006400</td>
</tr>
<tr>
<td>cache.r3.xlarge</td>
<td>30513561600</td>
<td>3040870400</td>
<td>3040870400</td>
</tr>
<tr>
<td>cache.r3.2xlarge</td>
<td>62495129600</td>
<td>6081740800</td>
<td>6081740800</td>
</tr>
<tr>
<td>Node Type</td>
<td>maxmemory</td>
<td>client-output-buffer-limit-slave-hard-limit</td>
<td>client-output-buffer-limit-slave-soft-limit</td>
</tr>
<tr>
<td>------------</td>
<td>--------------------</td>
<td>---------------------------------------------</td>
<td>---------------------------------------------</td>
</tr>
<tr>
<td>cache.r3.4xlarge</td>
<td>126458265600</td>
<td>12268339200</td>
<td>12268339200</td>
</tr>
<tr>
<td>cache.r3.8xlarge</td>
<td>254384537600</td>
<td>24536678400</td>
<td>24536678400</td>
</tr>
<tr>
<td>cache.r4.large</td>
<td>13201781556</td>
<td>1320178155</td>
<td>1320178155</td>
</tr>
<tr>
<td>cache.r4.xlarge</td>
<td>26898228839</td>
<td>2689822883</td>
<td>2689822883</td>
</tr>
<tr>
<td>cache.r4.2xlarge</td>
<td>54197537997</td>
<td>5419753799</td>
<td>5419753799</td>
</tr>
<tr>
<td>cache.r4.4xlarge</td>
<td>108858546586</td>
<td>10885854658</td>
<td>10885854658</td>
</tr>
<tr>
<td>cache.r4.8xlarge</td>
<td>218255432090</td>
<td>21825543209</td>
<td>21825543209</td>
</tr>
<tr>
<td>cache.r4.16xlarge</td>
<td>437021573120</td>
<td>43702157312</td>
<td>43702157312</td>
</tr>
</tbody>
</table>

**Note**
T1 instances do not support Multi-AZ with automatic failover.
T1 and T2 instances do not support Redis AOF.
All T2 instances are created in an Amazon Virtual Private Cloud (Amazon VPC).
T2 instances do not support Redis backup/restore.
T2 instances support Multi-AZ with automatic failover only when running Redis (cluster mode enabled).
Subnets and Subnet Groups

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

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

This section covers how to create and leverage subnets and subnet groups to manage access to your ElastiCache resources.

For more information about subnet group usage in an Amazon VPC environment, see [Step 4: Authorize Access](p. 32).

**Topics**
- Creating a Subnet Group (p. 380)
- Assigning a Subnet Group to a Cluster or Replication Group (p. 383)
- Modifying a Subnet Group (p. 384)
- Deleting a Subnet Group (p. 386)
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)**

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, macOS, or Unix:

```
aws elasticache create-cache-subnet-group \
  --cache-subnet-group-name mysubnetgroup \
  --cache-subnet-group-description "Testing" \
  --subnet-ids subnet-53df9c3a
```

For Windows:

```
aws elasticache create-cache-subnet-group ^
  --cache-subnet-group-name mysubnetgroup ^
  --cache-subnet-group-description "Testing" ^
  --subnet-ids subnet-53df9c3a
```

This command should produce output similar to the following:

```
{
  "CacheSubnetGroup": {
    "VpcId": "vpc-37c3cd17",
    "CacheSubnetGroupDescription": "Testing",
    "Subnets": [
      {
        "SubnetIdentifier": "subnet-53df9c3a",
        "SubnetAvailabilityZone": {
          "Name": "us-west-2a"
        }
      }
    ],
    "CacheSubnetGroupName": "mysubnetgroup"
  }
}
```

For more information, see the AWS CLI topic `create-cache-subnet-group`.

**Creating a Subnet Group (ElastiCache API)**

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

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

**Example**

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

**API Version 2015-02-02**

381
Create a Subnet Group (ElastiCache API)

```xml
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<CreateCacheSubnetGroupResult xmlns='http://aws.amazon.com/sdb/2010-07-01'>
<CacheSubnetGroup>
  <CacheSubnetGroupName>mysubnetgroup</CacheSubnetGroupName>
  <CacheSubnetGroupDescription>Testing</CacheSubnetGroupDescription>
  <SubnetIds>
    <member>subnet-53df9c3a</member>
  </SubnetIds>
</CreateCacheSubnetGroupResult>
```
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. 157). In step 5.a (Advanced Memcached Settings), choose a VPC subnet group.
- **Standalone Redis cluster** – To launch a single-node Redis cluster, see Creating a Redis (cluster mode disabled) Cluster (Console) (p. 159). In step 5.a (Advanced Redis Settings), choose 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 mode disabled) Cluster with Replicas from Scratch (Console) (p. 261). In step 5.b (Advanced Redis Settings), choose a VPC subnet group.
- **Redis (cluster mode enabled) replication group** – Creating a Redis (cluster mode enabled) Cluster (Console) (p. 267). In step 5.a (Advanced Redis Settings), choose a VPC subnet group.
Modifying a Subnet Group

You can modify a subnet group's description, or modify the list of subnet IDs associated with the subnet group. You cannot delete a subnet ID from a subnet group if a cluster is currently using that subnet.

The following procedures show you how to modify a subnet group.

Modifying Subnet Groups (Console)

To modify a subnet group

2. In the navigation pane, choose Subnet Groups.
3. In the list of subnet groups, choose the one you want to modify.
4. In the lower portion of the ElastiCache console, make any changes to the description or the list of subnet IDs for the subnet group. To save your changes, choose Save.

Modifying Subnet Groups (AWS CLI)

At a command prompt, use the command modify-cache-subnet-group to modify a subnet group.

For Linux, macOS, or Unix:

```bash
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:

```bash
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:

```json
{
  "CacheSubnetGroup": {
    "VpcId": "vpc-73cd3c17",
    "CacheSubnetGroupDescription": "New description",
    "Subnets": [
      {
        "SubnetIdentifier": "subnet-42df9c3a",
        "SubnetAvailabilityZone": {
          "Name": "us-west-2a"
        }
      },
      {
        "SubnetIdentifier": "subnet-48fc21a9",
        "SubnetAvailabilityZone": {
          "Name": "us-west-2a"
        }
      }
    ]
  }
}
```
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**


**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

2. In the navigation pane, choose Subnet Groups.
3. In the list of subnet groups, choose the one you want to delete and then choose Delete.
4. When you are asked to confirm this operation, choose Yes, Delete.

Deleting a Subnet Group (AWS CLI)

Using the AWS CLI, call the command `delete-cache-subnet-group` with the following parameter:

- `--cache-subnet-group-name mysubnetgroup`

For Linux, macOS, or Unix:

```bash
aws elasticache delete-cache-subnet-group --cache-subnet-group-name mysubnetgroup
```

For Windows:

```bash
aws elasticache delete-cache-subnet-group --cache-subnet-group-name mysubnetgroup
```

This command produces no output.

For more information, see the AWS CLI topic `delete-cache-subnet-group`.

Deleting a Subnet Group (ElastiCache API)

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

- `CacheSubnetGroupName=mysubnetgroup`

Example

Line breaks are added for ease of reading.

```xml
https://elasticache.us-west-2.amazonaws.com/
    ?Action=DeleteCacheSubnetGroup
    &CacheSubnetGroupName=mysubnetgroup
    &SignatureMethod=HmacSHA256
    &SignatureVersion=4
```

API Version 2015-02-02
386
This command produces no output.

For more information, see the ElastiCache API topic DeleteCacheSubnetGroup.
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
- Understanding ElastiCache and Amazon VPCs (p. 389)
- Access Patterns for Accessing an ElastiCache Cluster in an Amazon VPC (p. 393)
- Creating a Virtual Private Cloud (VPC) (p. 400)
- Creating a Cache Subnet Group (p. 402)
- Creating a Cache Cluster in an Amazon VPC (p. 403)
- Creating a Replication Group in an Amazon VPC (p. 404)
- Connecting to a Cluster or Replication Group Running in an Amazon VPC (p. 405)
Understanding ElastiCache and Amazon VPCs

ElastiCache is fully integrated with the Amazon Virtual Private Cloud (Amazon VPC). For ElastiCache users, this means the following:

- If your AWS account supports only the EC2-VPC platform, ElastiCache always launches your cluster in an Amazon VPC.
- If you’re new to AWS, your clusters will be deployed into an Amazon VPC. A default VPC will be created for you automatically.
- If you have a default VPC and don’t specify a subnet when you launch a cluster, the cluster launches into your default Amazon VPC.

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

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

The basic functionality of ElastiCache is the same in a virtual private cloud; ElastiCache manages software upgrades, patching, failure detection and recovery whether your clusters are deployed inside or outside an Amazon VPC.

ElastiCache cache nodes deployed outside an Amazon VPC are assigned an IP address to which the endpoint/DNS name resolves. This provides connectivity from Amazon Elastic Compute Cloud (Amazon EC2) instances. When you launch an ElastiCache cluster into an Amazon VPC private subnet, every cache node is assigned a private IP address within that subnet.

Overview of ElastiCache In an Amazon VPC

The following diagram and table describe the Amazon VPC environment, along with ElastiCache clusters and Amazon EC2 instances that are launched in the Amazon VPC.
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 an Amazon VPC allows you to:

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

For a comparison of Amazon EC2 Classic, Default VPC, and Non-default VPC, 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.

### Prerequisites

In order to create an ElastiCache cluster within an Amazon VPC, your Amazon VPC must meet the following requirements:

- The Amazon VPC must allow nondedicated Amazon EC2 instances. You cannot use ElastiCache in an Amazon VPC that is configured for dedicated instance tenancy.
- A cache subnet group must be defined for your Amazon VPC. ElastiCache uses that cache subnet group to select a subnet and IP addresses within that subnet to associate with your cache nodes.
- A cache security group must be defined for your Amazon VPC, or you can use the default provided.
- CIDR blocks for each subnet must be large enough to provide spare IP addresses for ElastiCache to use during maintenance activities.

### Routing and Security

You can configure routing in your Amazon VPC to control where traffic flows (for example, to the Internet gateway or virtual private gateway). With an Internet gateway, your Amazon VPC has direct
access to other AWS resources that are not running in your Amazon VPC. If you choose to have only a virtual private gateway with a connection to your organization's local network, you can route your Internet-bound traffic over the VPN and use local security policies and firewall to control egress. In that case, you incur additional bandwidth charges when you access AWS resources over the Internet.

You can use Amazon VPC security groups to help secure the ElastiCache clusters and Amazon EC2 instances in your Amazon VPC. Security groups act like a firewall at the instance level, not the subnet level.

**Note**
We strongly recommend that you use DNS names to connect to your cache nodes, as the underlying IP address can change if you reboot the cache node.

### Amazon VPC Documentation

Amazon VPC has its own set of documentation to describe how to create and use your Amazon VPC. The following table gives links to the Amazon VPC guides.

<table>
<thead>
<tr>
<th>Description</th>
<th>Documentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>How to get started using Amazon VPC</td>
<td>Amazon VPC Getting Started Guide</td>
</tr>
<tr>
<td>How to use Amazon VPC through the AWS Management Console</td>
<td>Amazon VPC User Guide</td>
</tr>
<tr>
<td>Complete descriptions of all the Amazon VPC commands</td>
<td>Amazon EC2 Command Line Reference (the Amazon VPC commands are part of the Amazon EC2 reference)</td>
</tr>
<tr>
<td>Complete descriptions of the Amazon VPC API actions, data types, and errors</td>
<td>Amazon EC2 API Reference (the Amazon VPC API actions are part of the Amazon EC2 reference)</td>
</tr>
<tr>
<td>Information for the network administrator who needs to configure the gateway at your end of an optional IPsec VPN connection</td>
<td>Amazon VPC Network Administrator Guide</td>
</tr>
</tbody>
</table>

For more detailed information about Amazon Virtual Private Cloud, see [Amazon Virtual Private Cloud](https://aws.amazon.com/).
Access Patterns for Accessing an ElastiCache Cluster in an Amazon VPC

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

Contents

- Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in the Same Amazon VPC (p. 393)
- Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs (p. 394)
  - Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs in the Same Region (p. 395)
  - Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs in Different Regions (p. 396)
- Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center (p. 397)
  - Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center Using VPN Connectivity (p. 397)
  - Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center Using Direct Connect (p. 398)

Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in the Same Amazon VPC

The most common use case is when an application deployed on an EC2 instance needs to connect to a Cluster in the same VPC.

The following diagram illustrates this scenario

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

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

- Memcached: 11211
- Redis: 6379

2. Create a VPC security group for your EC2 instances (web and application servers). This security group can, if needed, allow access to the EC2 instance from the Internet via the VPC's routing table. For example, you can set rules on this security group to allow TCP access to the EC2 instance over port 22.

3. Create custom rules in the security group for your Cluster that allow connections from the security group you created for your EC2 instances. This would allow any member of the security group to access the DB instances.

To create a rule in a VPC security group that allows connections from another security group

1. Sign in to the AWS Management Console and open the Amazon VPC console at https://console.aws.amazon.com/vpc.
2. In the navigation pane, choose Security Groups.
3. Select or create a security group that you will use for your Cluster instances. Choose Add Rule. This security group will allow access to members of another security group.
4. From Type choose Custom TCP Rule.
   a. For Port Range, specify the port you used when you created your cluster.
   b. In the Source box, start typing the ID of the security group. From the list select the security group you will use for your Amazon EC2 instances.
5. Choose Save when you finish.

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

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

- Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs in the Same Region (p. 395)
- Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs in Different Regions (p. 396)

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

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

Cluster accessed by an Amazon EC2 instance in a different Amazon VPC within the same Region - VPC Peering Connection

A VPC peering connection is a networking connection between two VPCs that enables you to route traffic between them using private IP addresses. Instances in either VPC can communicate with each other as if they are within the same network. You can create a VPC peering connection between your own Amazon VPCs, or with an Amazon VPC in another AWS account within a single region. To learn more about Amazon VPC peering, see the VPC documentation.

To access a cluster in a different Amazon VPC over peering

1. Make sure that the two VPCs do not have an overlapping IP range or you will not be able to peer them.
2. Peer the two VPCs. For more information, see Creating and Accepting a Amazon VPC Peering Connection.
3. Update your routing table. For more information, see Updating Your Route Tables for a VPC Peering Connection

Following is what the route tables look like for the example in the preceding diagram. Note that pcx-a894f1c1 is the peering connection.
Accessing an ElastiCache Cluster when it and the Amazon EC2 Instance are in Different Amazon VPCs

VPC Routing Table

4. Modify the Security Group of your ElastiCache cluster to allow inbound connection from the Application security group in the peered VPC. For more information, see Reference Peer VPC Security Groups.

Accessing a cluster over a peering connection will incur additional data transfer costs.

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

One common strategy for connecting multiple, geographically disperse VPCs and remote networks is to create a transit VPC that serves as a global network transit center. A transit VPC simplifies network management and minimizes the number of connections required to connect multiple VPCs and remote networks. This design can save time and effort and also reduce costs, as it is implemented virtually without the traditional expense of establishing a physical presence in a colocation transit hub or deploying physical network gear.

Connecting across different VPCs in different regions

Once the Transit Amazon VPC is established, an application deployed in a “spoke” VPC in one region can connect to an ElastiCache cluster in a “spoke” VPC within another region.
To access a cluster in a different VPC within a different Region

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

2. Update the VPC routing tables in the App and Cache VPCs to route traffic through the VGW (Virtual Private Gateway) and the VPN Appliance. In case of Dynamic Routing with Border Gateway Protocol (BGP) your routes may be automatically propagated.

3. Modify the Security Group of your ElastiCache cluster to allow inbound connection from the Application instances IP range. Note that you will not be able to reference the application server Security Group in this scenario.

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

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

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

Topics

- Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center Using VPN Connectivity (p. 397)
- Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center Using Direct Connect (p. 398)

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

The following diagram illustrates accessing an ElastiCache cluster from an application running in your corporate network using VPN connections.
Connecting to ElastiCache from your data center via a VPN

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

1. Establish VPN Connectivity by adding a hardware Virtual Private Gateway to your VPC. For more information, see Adding a Hardware Virtual Private Gateway to Your VPC.
2. Update the VPC routing table for the subnet where your ElastiCache cluster is deployed to allow traffic from your on-premises application server. In case of Dynamic Routing with BGP your routes may be automatically propagated.
3. Modify the Security Group of your ElastiCache cluster to allow inbound connection from the on-premises application servers.

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

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

The following diagram illustrates accessing an ElastiCache cluster from an application running on your corporate network using Direct Connect.
Connecting to ElastiCache from your data center via Direct Connect

To access an ElastiCache cluster from an application running in your network using Direct Connect

1. Establish Direct Connect connectivity. For more information, see, Getting Started with AWS Direct Connect.

2. Modify the Security Group of your ElastiCache cluster to allow inbound connection from the on-premises application servers.

Accessing a cluster over DX connection may introduce networking latencies and additional data transfer charges.
Creating a Virtual Private Cloud (VPC)

In this example, you create an Amazon VPC with a private subnet for each Availability Zone.

Creating an Amazon VPC (Console)

To create an ElastiCache 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 navigation pane of the Amazon VPC Management console, choose Security Groups.
   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. 380).

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. 403).
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. 157). 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. 159). 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. 405).
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 mode disabled) Cluster with Replicas from Scratch (Console) (p. 261) 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) Cluster (Console) (p. 267) 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. 405).
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](https://docs.aws.amazon.com/ec2/latest/userguide/) in the Amazon EC2 documentation.

**Important**
To avoid incurring additional charges on your AWS account, be sure to delete any AWS resources you no longer want after trying these examples.

For information on connecting to your Memcached or Redis cluster, see *Step 5: Connect to a Cluster's Node* (p. 36) in the ElastiCache User Guide.
Amazon ElastiCache Data Security and Compliance

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

Topics
- ElastiCache and Security Groups (p. 406)
- Authentication and Access Control for Amazon ElastiCache (p. 407)
- Authenticating Users with AUTH (Redis) (p. 427)
- Amazon ElastiCache for Redis Data Encryption (p. 429)
- HIPAA Compliance for Amazon ElastiCache for Redis (p. 439)

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 (p. 406)
- Amazon EC2-Classic: ElastiCache Security Groups (p. 406)

Amazon Virtual Private Cloud: Amazon VPC Security Groups

When running your clusters in an Amazon Virtual Private Cloud, you configure your Amazon VPC by choosing 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. 388).

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. 328).
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. 407)
- Access Control (p. 408)

Authentication

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

- **AWS account root user** – When you first create an AWS account, you begin with a single sign-in identity that has complete access to all AWS services and resources in the account. This identity is called the AWS account root user and is accessed by signing in with the email address and password that you used to create the account. We strongly recommend that you do not use the root user for your everyday tasks, even the administrative ones. Instead, adhere to the best practice of using the root user only to create your first IAM user. Then securely lock away the root user credentials and use them to perform only a few account and service management tasks.

- **IAM user** – An IAM user is an identity within your AWS account that has specific custom permissions (for example, permissions to create a cluster in ElastiCache). You can use an IAM user name and password to sign in to secure AWS webpages like the AWS Management Console, AWS Discussion Forums, or the AWS Support Center.

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

- **IAM role** – An IAM role is an IAM identity that you can create in your account that has specific permissions. It is similar to an IAM user, but it is not associated with a specific person. An IAM role enables you to obtain temporary access keys that can be used to access AWS services and resources. IAM roles with temporary credentials are useful in the following situations:

- **Federated user access** – Instead of creating an IAM user, you can use existing user identities from AWS Directory Service, your enterprise user directory, or a web identity provider. These are known as federated users. AWS assigns a role to a federated user when access is requested through an identity provider. For more information about federated users, see Federated Users and Roles in the IAM User Guide.
• **AWS service access** – You can use an IAM role in your account to grant an AWS service permissions to access your account's resources. For example, you can create a role that allows Amazon Redshift to access an Amazon S3 bucket on your behalf and then load data from that bucket into an Amazon Redshift cluster. For more information, see Creating a Role to Delegate Permissions to an AWS Service in the IAM User Guide.

• **Applications running on Amazon EC2** – You can use an IAM role to manage temporary credentials for applications that are running on an EC2 instance and making AWS API requests. This is preferable to storing access keys within the EC2 instance. To assign an AWS role to an EC2 instance and make it available to all of its applications, you create an instance profile that is attached to the instance. An instance profile contains the role and enables programs that are running on the EC2 instance to get temporary credentials. For more information, see Using an IAM Role to Grant Permissions to Applications Running on Amazon EC2 Instances in the IAM User Guide.

**Access Control**

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

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

• Overview of Managing Access Permissions to Your ElastiCache Resources (p. 409)
• Using Identity-Based Policies (IAM Policies) for Amazon ElastiCache (p. 413)
Overview of Managing Access Permissions to Your ElastiCache Resources

Every AWS resource is owned by an AWS account, and permissions to create or access a resource are governed by permissions policies. An account administrator can attach permissions policies to IAM identities (that is, users, groups, and roles), and some services (such as AWS Lambda) also support attaching permissions policies to resources.

Note
An account administrator (or administrator user) is a user with administrator privileges. For more information, see IAM Best Practices in the IAM User Guide.

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

Topics
- Amazon ElastiCache Resources and Operations (p. 409)
- Understanding Resource Ownership (p. 409)
- Managing Access to Resources (p. 410)
- Specifying Policy Elements: Actions, Effects, Resources, and Principals (p. 411)
- Specifying Conditions in a Policy (p. 411)

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.

<table>
<thead>
<tr>
<th>Resource Type</th>
<th>ARN Format</th>
</tr>
</thead>
</table>

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

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) (p. 410)
• Resource-Based Policies (p. 411)

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 (*).

```json
{
  "Version": "2012-10-17",
  "Statement": [
    { "Sid": "DescribeCacheClusters",
```

API Version 2015-02-02
410
For more information about using identity-based policies with ElastiCache, see Using Identity-Based Policies (IAM Policies) for Amazon ElastiCache (p. 413). 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. 409)), 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. 409).

- **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. 424).

**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. 409).

The sections in this topic cover the following:

- Permissions Required to Use the Amazon ElastiCache Console (p. 414)
- AWS Managed (Predefined) Policies for Amazon ElastiCache (p. 414)
- Customer Managed Policy Examples (p. 415)

The following shows an example of a permissions policy.

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

The policy has two statements:

- The first statement grants permissions for the Amazon ElastiCache actions (elasticache:CreateCacheCluster, elasticache:DescribeCacheClusters, elasticache:ModifyCacheCluster, and elasticache:RebootCacheCluster) on any cache cluster owned by the account. Currently, Amazon ElastiCache doesn't support permissions for actions at the resource-level. Therefore, the policy specifies a wildcard character (*) as the Resource value.

- The second statement grants permissions for the IAM action (iam:PassRole) on IAM roles. The wildcard character (*) at the end of the Resource value means that the statement allows permission for the iam:PassRole action on any IAM role. To limit this permission to a specific role, replace the wildcard character (*) in the resource ARN with the specific role name.

The policy doesn't specify the Principal element because in an identity-based policy you don't specify the principal who gets the permission. When you attach policy to a user, the user is the implicit principal. When you attach a permissions policy to an IAM role, the principal identified in the role's trust policy gets the permissions.

For a table showing all of the Amazon ElastiCache API actions and the resources that they apply to, see ElastiCache API Permissions: Actions, Resources, and Conditions Reference (p. 424).
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. 424).

To use the Amazon ElastiCache console, you need to grant permissions for additional actions as shown in the following permissions policy:

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

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

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

AWS Managed (Predefined) Policies for Amazon ElastiCache

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

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

- **AmazonElastiCacheReadOnlyAccess** - Grants read-only access to Amazon ElastiCache resources.
- **AmazonElastiCacheFullAccess** - Grants full access to Amazon ElastiCache resources.
Note
You can review these permissions policies by signing in to the IAM console and searching for specific policies there.

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

Customer Managed Policy Examples

If you are not using default policy and choose to use a custom managed policy, please ensure you have either permissions to call `iam:createServiceLinkedRole` (see Example 5: Allow a User to Call IAM CreateServiceLinkedRole API (p. 417)) or you have created the ElastiCache Service Linked Role.

When combined with the minimum permissions needed to use the Amazon ElastiCache console, the example policies in this section grant additional permissions. The examples are also relevant to the AWS SDKs and the AWS CLI. For more information about what permissions are needed to use the ElastiCache console, see Permissions Required to Use the Amazon ElastiCache Console (p. 414).

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 (p. 415)
- Example 2: Allow a User Read-Only Access to ElastiCache Resources (p. 416)
- Example 3: Allow a User to Perform Common ElastiCache System Administrator Tasks (p. 416)
- Example 4: Allow a User to Access All ElastiCache API Actions (p. 416)
- Example 5: Allow a User to Call IAM CreateServiceLinkedRole API (p. 417)

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.

```json
{
  "Version": "2012-10-17",
  "Statement": [
  ]
}
```
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.

```json
{
    "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.

```json
{
    "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.

```json
{
    "Version": "2012-10-17",
    "Statement": [ {
        "Sid": "ECAllowSpecific",
        "Effect": "Allow",
        "Action": [ "elasticache:*" ]
    } ]
}
```
Using Service-Linked Roles for ElastiCache

Amazon ElastiCache uses AWS Identity and Access Management (IAM) service-linked roles. A service-linked role is a unique type of IAM role that is linked directly to an AWS service, such as ElastiCache. ElastiCache service-linked roles are predefined by ElastiCache and include all the permissions that the service requires to call AWS services on behalf of your clusters.

A service-linked role makes setting up ElastiCache easier because you don't have to manually add the necessary permissions. The roles already exist within your AWS account but are linked to ElastiCache use cases and have predefined permissions. Only ElastiCache can assume these roles, and only these roles can use the predefined permissions policy. You can delete the roles only after first deleting their related resources. This protects your ElastiCache resources because you can't inadvertently remove necessary permissions to access the resources.

For information about other services that support service-linked roles, see AWS Services That Work with IAM and look for the services that have Yes in the Service-Linked Role column. Choose a Yes with a link to view the service-linked role documentation for that service.

Contents

- Service-Linked Role Permissions for ElastiCache (p. 418)
- Creating a Service-Linked Role (IAM) (p. 419)
  - Creating a Service-Linked Role (IAM Console) (p. 419)
  - Creating a Service-Linked Role (IAM CLI) (p. 419)
  - Creating a Service-Linked Role (IAM API) (p. 419)
- Editing the Description of a Service-Linked Role for ElastiCache (p. 420)
  - Editing a Service-Linked Role Description (IAM Console) (p. 420)
  - Editing a Service-Linked Role Description (IAM CLI) (p. 420)
Amazon ElastiCache User Guide
Using Service-Linked Roles

- Editing a Service-Linked Role Description (IAM API) (p. 420)
- Deleting a Service-Linked Role for ElastiCache (p. 421)
  - Cleaning Up a Service-Linked Role (p. 421)
  - Deleting a Service-Linked Role (IAM Console) (p. 422)
  - Deleting a Service-Linked Role (IAM CLI) (p. 422)
  - Deleting a Service-Linked Role (IAM API) (p. 422)

Service-Linked Role Permissions for ElastiCache

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

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

Permission policy:
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AuthorizeSecurityGroupIngress",
                "ec2:CreateNetworkInterface",
                "ec2:CreateSecurityGroup",
                "ec2:DeleteNetworkInterface",
                "ec2:DeleteSecurityGroup",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeNetworkInterfaces",
                "ec2:DescribeSecurityGroups",
                "ec2:DescribeSubnets",
                "ec2:DescribeVpcs",
                "ec2:ModifyNetworkInterfaceAttribute",
                "ec2:RevokeSecurityGroupIngress",
            ],
            "Resource": "*"
        }
    ]
}

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

Add the following policy statement to the permissions for that IAM entity:

{
    "Effect": "Allow",
    "Action": [
        "iam:CreateServiceLinkedRole",
        "iam:PutRolePolicy"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/elasticache.amazonaws.com/AWSServiceRoleForElastiCache",
    "Condition": {
        "StringLike": {
            "iam:AWSServiceName": "elasticache.amazonaws.com"
        }
    }
}

To allow an IAM entity to delete AWSServiceRoleForElastiCache service-linked roles

Add the following policy statement to the permissions for that IAM entity:


Alternatively, you can use an AWS managed policy to provide full access to ElastiCache.

### Creating a Service-Linked Role (IAM)

You can create a service-linked role using the IAM console, CLI, or API.

#### Creating a Service-Linked Role (IAM Console)

You can use the IAM console to create a service-linked role.

**To create a service-linked role (console)**

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane of the IAM console, choose **Roles**. Then choose **Create new role**.
3. Expand the **AWS service-linked role** section, and then select the service that you want to allow to assume this new service-linked role.
4. Next to the **AWSServiceRoleForElastiCache** service-linked role, choose **Select**.
5. For **Role name**, type a suffix to add to the service-linked role default name. This suffix helps you identify the purpose of this role. Role names must be unique within your AWS account. They are not distinguished by case. For example, you cannot create roles named both `<service-linked-role-name>_SAMPLE` and `<service-linked-role-name>_sample`. Because various entities might reference the role, you cannot edit the name of the role after it has been created.
6. (Optional) For **Role description**, edit the description for the new service-linked role.
7. Review the role and then choose **Create role**.

#### Creating a Service-Linked Role (IAM CLI)

You can use IAM operations from the AWS Command Line Interface to create a service-linked role with the trust policy and inline policies that the service needs to assume the role.

**To create a service-linked role (CLI)**

Use the following operation:

```
$ aws iam create-service-linked-role --aws-service-name elasticache.amazonaws.com
```

#### Creating a Service-Linked Role (IAM API)

You can use the IAM API to create a service-linked role with the trust policy and inline policies that the service needs to assume the role.

**To create a service-linked role (API)**

```json
{
  "Effect": "Allow",
  "Action": [
    "iam:DeleteServiceLinkedRole",
    "iam:GetServiceLinkedRoleDeletionStatus"
  ],
  "Resource": "arn:aws:iam::*:role/aws-service-role/elasticache.amazonaws.com/AWSServiceRoleForElastiCache*",
  "Condition": {
    "StringLike": {"iam:AWSServiceName": "elasticache.amazonaws.com"}
  }
}
```
Use the `CreateServiceLinkedRole` API call. In the request, specify a service name of elasticache.amazonaws.com.

**Editing the Description of a Service-Linked Role for ElastiCache**

ElastiCache does not allow you to edit the AWSServiceRoleForElastiCache service-linked role. After you create a service-linked role, you cannot change the name of the role because various entities might reference the role. However, you can edit the description of the role using IAM.

**Editing a Service-Linked Role Description (IAM Console)**

You can use the IAM console to edit a service-linked role description.

**To edit the description of a service-linked role (console)**

1. In the navigation pane of the IAM console, choose **Roles**.
2. Choose the name of the role to modify.
3. To the far right of **Role description**, choose **Edit**.
4. Type a new description in the box and choose **Save**.

**Editing a Service-Linked Role Description (IAM CLI)**

You can use IAM operations from the AWS Command Line Interface to edit a service-linked role description.

**To change the description of a service-linked role (CLI)**

1. (Optional) To view the current description for a role, use the AWS CLI for IAM operation `get-role`.
   
   **Example**
   
   ```bash
   $ aws iam get-role --role-name AWSServiceRoleForElastiCache
   ```
   
   Use the role name, not the ARN, to refer to roles with the CLI operations. For example, if a role has the following ARN: `arn:aws:iam::123456789012:role/myrole`, refer to the role as `myrole`.

2. To update a service-linked role's description, use the AWS CLI for IAM operation `update-role-description`.
   
   For Linux, macOS, or Unix:
   
   ```bash
   $ aws iam update-role-description 
   --role-name AWSServiceRoleForElastiCache 
   --description "new description"
   ```
   
   For Windows:
   
   ```bash
   $ aws iam update-role-description ^
   --role-name AWSServiceRoleForElastiCache ^
   --description "new description"
   ```

**Editing a Service-Linked Role Description (IAM API)**

You can use the IAM API to edit a service-linked role description.
To change the description of a service-linked role (API)

1. (Optional) To view the current description for a role, use the IAM API operation GetRole.

   **Example**

   ```
   https://iam.amazonaws.com/
   ?Action=GetRole
   &RoleName=AWSServiceRoleForElastiCache
   &Version=2010-05-08
   &AUTHPARAMS
   ```

2. To update a role's description, use the IAM API operation UpdateRoleDescription.

   **Example**

   ```
   https://iam.amazonaws.com/
   ?Action=UpdateRoleDescription
   &RoleName=AWSServiceRoleForElastiCache
   &Version=2010-05-08
   &Description="New description"
   ```

Deleting a Service-Linked Role for ElastiCache

If you no longer need to use a feature or service that requires a service-linked role, we recommend that you delete that role. That way you don't have an unused entity that is not actively monitored or maintained. However, you must clean up your service-linked role before you can delete it.

Amazon ElastiCache does not delete the service-linked role for you.

Cleaning Up a Service-Linked Role

Before you can use IAM to delete a service-linked role, you must first confirm that the role has no resources, clusters or replication groups, associated with the role.

To check whether the service-linked role has an active session in the IAM console

1. Sign in to the AWS Management Console and open the IAM console at **https://console.aws.amazon.com/iam/**.
2. In the navigation pane of the IAM console, choose **Roles**. Then choose the name (not the check box) of the AWSServiceRoleForElastiCache role.
3. On the **Summary** page for the selected role, choose the **Access Advisor** tab.
4. On the **Access Advisor** tab, review recent activity for the service-linked role.

To delete ElastiCache resources that require AWSServiceRoleForElastiCache (console)

- To delete a cluster, see one of the following:
  - Deleting a Cluster (Console) (p. 197)
  - Deleting a Cache Cluster (AWS CLI) (p. 197)
  - Deleting a Cache Cluster (ElastiCache API) (p. 198)
- To delete a replication group, see one of the following:
  - Deleting a Replication Group (Console) (p. 286)
  - Deleting a Replication Group (AWS CLI) (p. 286)
Deleting a Service-Linked Role (IAM Console)

You can use the IAM console to delete a service-linked role.

To delete a service-linked role (console)

1. Sign in to the AWS Management Console and open the IAM console at https://console.aws.amazon.com/iam/.
2. In the navigation pane of the IAM console, choose Roles. Then select the check box next to the role name that you want to delete, not the name or row itself.
3. For Role actions at the top of the page, choose Delete role.
4. In the confirmation dialog box, review the service last accessed data, which shows when each of the selected roles last accessed an AWS service. This helps you to confirm whether the role is currently active. If you want to proceed, choose Yes, Delete to submit the service-linked role for deletion.
5. Watch the IAM console notifications to monitor the progress of the service-linked role deletion. Because the IAM service-linked role deletion is asynchronous, after you submit the role for deletion, the deletion task can succeed or fail. If the task fails, you can choose View details or View Resources from the notifications to learn why the deletion failed.

Deleting a Service-Linked Role (IAM CLI)

You can use IAM operations from the AWS Command Line Interface to delete a service-linked role.

To delete a service-linked role (CLI)

1. If you don’t know the name of the service-linked role that you want to delete, type the following operation to list the roles and their Amazon Resource Names (ARNs) in your account:

   ```
   $ aws iam get-role --role-name role-name
   ```

   Use the role name, not the ARN, to refer to roles with the CLI operations. For example, if a role has the following ARN: arn:aws:iam::123456789012:role/myrole, you refer to the role as myrole.

2. Because a service-linked role cannot be deleted if it is being used or has associated resources, you must submit a deletion request. That request can be denied if these conditions are not met. You must capture the deletion-task-id from the response to check the status of the deletion task. Type the following to submit a service-linked role deletion request:

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

3. Type the following to check the status of the deletion task:

   ```
   $ aws iam get-service-linked-role-deletion-status --deletion-task-id deletion-task-id
   ```

   The status of the deletion task can be NOT_STARTED, IN_PROGRESS, SUCCEEDED, or FAILED. If the deletion fails, the call returns the reason that it failed so that you can troubleshoot.

Deleting a Service-Linked Role (IAM API)

You can use the IAM API to delete a service-linked role.
To delete a service-linked role (API)

1. To submit a deletion request for a service-linked role, call DeleteServiceLinkedRole. In the request, specify a role name.

   Because a service-linked role cannot be deleted if it is being used or has associated resources, you must submit a deletion request. That request can be denied if these conditions are not met. You must capture the DeletionTaskId from the response to check the status of the deletion task.

2. To check the status of the deletion, call GetServiceLinkedRoleDeletionStatus. In the request, specify the DeletionTaskId.

   The status of the deletion task can be NOT_STARTED, IN_PROGRESS, SUCCEEDED, or FAILED. If the deletion fails, the call returns the reason that it failed so that you can troubleshoot.
ElastiCache API Permissions: Actions, Resources, and Conditions Reference

When you are setting up Access Control (p. 408) 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:** *

TestFailover

**Action(s):** elasticache:TestFailover

**Resource:** *

### Authenticating Users with AUTH (Redis)

Using Redis `AUTH` command can improve data security by requiring the user to enter a password before they are granted permission to execute Redis commands on a password-protected Redis server.

This section covers ElastiCache's implementation of Redis' AUTH command, which has some differences from the Redis implementation.
Overview of ElastiCache for Redis AUTH

When using Redis AUTH with your ElastiCache cluster, there are some refinements you need to be aware of.

**AUTH Token Constraints when using with ElastiCache**

- Passwords must be at least 16 and a maximum of 128 printable characters.
- The printable characters @, ,, and / cannot be used in passwords.
- AUTH can only be enabled when creating clusters where in-transit encryption is enabled.
- The password set at cluster creation cannot be changed.

We recommend that you follow a stricter policy such as:

- Must include a mix of characters that includes at least three of the following character types:
  - Uppercase characters
  - Lowercase characters
  - Digits
  - Non-alphanumeric characters (!, @, #, $, ^, <, >)
- Must not contain a dictionary word or a slightly modified dictionary word.
- Must not be the same as or similar to a recently used password.

**Applying Authentication to an ElastiCache Command**

To require that users enter a password on a password-protected Redis server, add the line `requirepass` in the server's configuration file. ElastiCache for Redis commands must then include the parameter `--auth-token (API: AuthToken)` and the correct password to execute.

The following AWS CLI operation creates a replication group with Encryption In-Transit enabled and the AUTH token "this-is-a-sample-token". Replace the subnet group `sng-test` with a subnet group that exists.

**Key Parameters**

- `--engine` – must be redis.
- `--engine-version` – must be 3.2.6.
- `--transit-encryption-enabled` – encryption in-transit must be enabled to use authentication and for HIPAA compliance.
- `--auth-token` – must be the correct password for this password-protected Redis server and must be specified for HIPAA compliance.
- `--cache-subnet-group` – must be specified for HIPAA compliance.

For Linux, macOS, or Unix:

```
aws elasticache create-replication-group
```
For Windows:

```
aws elasticache create-replication-group ^
  --replication-group-id authtestgroup ^
  --replication-group-description authtest ^
  --engine redis ^
  --engine-version 3.2.6 ^
  --transit-encryption-enabled ^
  --cache-node-type cache.m4.large ^
  --num-node-groups 1 ^
  --replicas-per-node-group 2 ^
  --cache-parameter-group default.redis3.2.cluster.on ^
  --auth-token This-is-a-sample-token ^
  --cache-subnet-group sng-test
```

Related topics

- AUTH password at [redis.io](http://redis.io).

Amazon ElastiCache for Redis Data Encryption

To help keep your data secure, Amazon ElastiCache and Amazon EC2 provide mechanisms to guard against unauthorized access of your data on the server.

Amazon ElastiCache for Redis also provides optional encryption features for data on clusters running Redis version 3.2.6 and later:

- In-transit encryption encrypts your data whenever it is moving from one place to another, such as between nodes in your cluster or between your cluster and your application.
- At-rest encryption encrypts your on-disk data during sync and backup operations.

Topics

- Amazon ElastiCache for Redis In-Transit Encryption (p. 429)
- Amazon ElastiCache for Redis At-Rest Encryption (p. 434)

Amazon ElastiCache for Redis In-Transit Encryption

To help keep your data secure, Amazon ElastiCache and Amazon EC2 provide mechanisms to guard against unauthorized access of your data on the server. By providing in-transit encryption capability, ElastiCache gives you a tool you can use to help protect your data when it is moving from one location to another. For example, you might move data from a primary node to a read replica node within a replication group, or between your replication group and your application.
In-transit encryption is optional and can only be enabled on Redis replication groups when they are created. You enable in-transit encryption on a replication group by setting the parameter `TransitEncryptionEnabled` (CLI: `--transit-encryption-enabled`) to true when you create the replication group. You can do this whether you are creating the replication group using the AWS Management Console, the AWS CLI, or the ElastiCache API. If `TransitEncryptionEnabled` is set to true, you must also provide a value for `CacheSubnetGroup`.

**Important**
The parameters `TransitEncryptionEnabled` (CLI: `--transit-encryption-enabled`) are only available when using the CreateReplicationGroup (CLI: `create-replication-group`) operation.

**Topics**
- In-Transit Encryption Overview (p. 430)
- In-Transit Encryption Constraints (p. 430)
- Enabling In-Transit Encryption (p. 431)
- See Also (p. 433)

**In-Transit Encryption Overview**

Amazon ElastiCache in-transit encryption is an optional feature that allows you to increase the security of your data at its most vulnerable points—when it is in transit from one location to another. Because there is some processing needed to encrypt and decrypt the data at the endpoints, enabling in-transit encryption can have some performance impact. You should benchmark your data with and without in-transit encryption to determine the performance impact for your use cases.

ElastiCache in-transit encryption implements the following features:

- **Encrypted connections**—both the server and client connections are Secure Socket Layer (SSL) encrypted.
- **Encrypted replication**—data moving between a primary node and replica nodes is encrypted.
- **Server authentication**—clients can authenticate that they are connecting to the right server.
- **Client authentication**—using the Redis AUTH feature, the server can authenticate the clients.

**In-Transit Encryption Constraints**

The following constraints on Amazon ElastiCache in-transit encryption should be kept in mind when you plan your implementation:

- In-transit encryption is supported on replication groups running Redis version 3.2.6. It is not supported on clusters running Memcached.

In-transit encryption is supported only for replication groups running in an Amazon VPC.
- In-transit encryption is only supported for replication groups running most current generation node types. It is not supported on replication groups running on previous generation node types. For more information, see Supported Node Types (p. 117)
- In-transit encryption is enabled by setting the parameter `TransitEncryptionEnabled` to true. Because the default value of `TransitEncryptionEnabled` is false, you must explicitly set it to true when creating your replication group. If `TransitEncryptionEnabled` is set to true, you must also provide a value for `CacheSubnetGroup`.

In-transit encryption is not supported on clusters running Memcached.
- You can enable in-transit encryption on a replication group only when creating the replication group. You cannot toggle in-transit encryption on and off by modifying a replication group. For information on implementing in-transit encryption on an existing replication group, see Enabling In-Transit Encryption (p. 431).
• To connect to an in-transit encryption enabled replication group, a database must be enabled for transport layer security (TLS). To connect to a replication group that is not in-transit encryption enabled, the database cannot be TLS-enabled.

Because of the processing required to encrypt and decrypt the data at the endpoints, implementing in-transit encryption can reduce performance. Benchmark in-transit encryption compared to no encryption on your own data to determine its impact on performance for your implementation.

You can reduce the performance impact of in-transit encryption by persisting your SSL connections, because creating new connections can be expensive.

**Enabling In-Transit Encryption**

You can enable in-transit encryption when you create an ElastiCache for Redis replication group using the AWS Management Console, the AWS CLI, or the ElastiCache API.

**Enabling In-Transit Encryption on an Existing Cluster**

You can only enable in-transit encryption when you create a Redis replication group. If you have an existing replication group on which you want to enable in-transit encryption, do the following.

**To enable in-transit encryption for an existing Redis replication group**

1. Create a manual backup of the replication group. For more information, see [Making Manual Backups (p. 297)](#).
2. Create a new replication group by restoring from the backup setting the engine version to 3.2.6 and the parameter `TransitEncryptionEnabled` to `true` (CLI: `--transit-encryption-enabled`). For more information, see [Restoring From a Backup with Optional Cluster Resizing (p. 317)](#).
3. Update the endpoints in your application to the new replication group's endpoints. For more information, see [Finding Your ElastiCache Endpoints (p. 62)](#).
4. Delete the old replication group. For more information, see the following:
   - [Deleting a Cluster (p. 197)](#)
   - [Deleting a Cluster with Replicas (p. 286)](#)

**Enabling In-Transit Encryption Using the AWS Management Console**

To enable in-transit encryption when creating a replication group using the AWS Management Console, make the following selections:

- Choose Redis as your engine.
- Choose engine version 3.2.6.
- Choose Yes from the **Encryption in-transit** list.

For the step-by-step process, see the following:

- [Creating a Redis (cluster mode disabled) Cluster (Console) (p. 159)](#)
- [Creating a Redis (cluster mode enabled) Cluster (Console) (p. 163)](#)
**Enabling In-Transit Encryption Using the AWS CLI**

To enable in-transit encryption when creating a Redis replication group using the AWS CLI, use the parameter `transit-encryption-enabled`.

**Enabling In-Transit Encryption on a Redis (cluster mode disabled) Cluster (CLI)**

Use the AWS CLI operation `create-replication-group` and the following parameters to create a Redis replication group with replicas that has in-transit encryption enabled:

- `--engine` must be `redis`.
- `--engine-version` must be `3.2.6`.
- `--transit-encryption-enabled`. If you enable in-transit encryption you must also provide a value for the `--cache-subnet-group` parameter.
- `--num-cache-clusters` must be at least 1. The maximum value for this parameter is 6.

For more information, see the following:
- [Creating a Redis (cluster mode disabled) Cluster with Replicas from Scratch (AWS CLI) (p. 261)](#)
- [create-replication-group](#)

**Enabling In-Transit Encryption on a Redis (cluster mode enabled) Cluster (CLI)**

Use the AWS CLI operation `create-replication-group` and the following parameters to create a Redis (cluster mode enabled) replication group that has in-transit encryption enabled:

- `--engine` must be `redis`.
- `--engine-version` must be `3.2.6`.
- `--transit-encryption-enabled`. If you enable in-transit encryption you must also provide a value for the `--cache-subnet-group` parameter.
- Use one of the following parameter sets to specify the configuration of the replication group's node groups:
  - `--num-node-groups` to specify the number of shards (node groups) in this replication group. The maximum value of this parameter is 15.
  - `--replicas-per-node-group` to specify the number of replica nodes in each node group. The value specified here is applied to all shards in this replication group. The maximum value of this parameter is 5.
  - `--node-group-configuration` to specify the configuration of each shard independently.

For more information, see the following:
- [Creating a Redis (cluster mode enabled) Cluster with Replicas from Scratch (AWS CLI) (p. 267)](#)
- [create-replication-group](#)

**Enabling In-Transit Encryption Using the AWS API**

To enable in-transit encryption when creating a Redis replication group using the ElastiCache API, set the parameter `TransitEncryptionEnabled` to `true` with either `CreateCacheCluster` for a single node Redis replication group, or `CreateReplicationGroup` for a replication group with read replicas.
Enabling In-Transit Encryption on a Redis (cluster mode disabled) Cluster (API)

Use the ElastiCache API operation CreateReplicationGroup and the following parameters to create a Redis (cluster mode disabled) replication group that has in-transit encryption enabled:

- Engine must be redis.
- EngineVersion must be 3.2.6.
- TransitEncryptionEnabled must set to true.

If TransitEncryptionEnabled is set to true, you must also provide a value for CacheSubnetGroup.
- NumCacheClusters must be at least 1. The maximum value for this parameter is 6.

For more information, see the following:
- Creating a Redis (cluster mode disabled) Cluster with Replicas from Scratch (ElastiCache API) (p. 264)
- CreateReplicationGroup

Enabling In-Transit Encryption on a Redis (cluster mode enabled) Cluster (API)

Use the ElastiCache API operation CreateReplicationGroup and the following parameters to create a Redis (cluster mode enabled) replication group that has in-transit encryption enabled:

- Engine must be redis.
- EngineVersion must be 3.2.6.
- TransitEncryptionEnabled must set to true.
- Use one of the following parameter sets to specify the configuration of the replication group's node groups:
  - NumNodeGroups to specify the number of shards (node groups) in this replication group. The maximum value of this parameter is 15.
  - ReplicasPerNodeGroup to specify the number of replica nodes in each node group. The value specified here is applied to all shards in this replication group. The maximum value of this parameter is 5.
  - NodeGroupConfiguration to specify the configuration of each shard independently.

For more information, see the following:
- Creating a Redis (cluster mode enabled) Cluster with Replicas from Scratch (ElastiCache API) (p. 271)
- CreateReplicationGroup

See Also

- Amazon ElastiCache for Redis At-Rest Encryption (p. 434)
- Authenticating Users with AUTH (Redis) (p. 427)
- ElastiCache and Security Groups (p. 406)
- Authentication and Access Control for Amazon ElastiCache (p. 407)
Amazon ElastiCache for Redis At-Rest Encryption

To help keep your data secure, Amazon ElastiCache and Amazon S3 provide mechanisms to restrict access to your data when it's in your cache. For more information, see ElastiCache and Security Groups (p. 406) and Authentication and Access Control for Amazon ElastiCache (p. 407).

When ElastiCache for Redis at-rest encryption is enabled on a replication group, your data is encrypted when it's on the disk during sync and backup operations. This approach is different from ElastiCache for Redis in-transit encryption. In-transit encryption encrypts your data when it is moving from one place to another, such as between your replication group and your application. For information about ElastiCache for Redis in-transit encryption, see Amazon ElastiCache for Redis In-Transit Encryption (p. 429). At-rest encryption is optional, and can be enables on a replication group only when it is created.

**Topics**

- At-Rest Encryption Overview (p. 434)
- At-Rest Encryption Constraints (p. 434)
- Enabling At-Rest Encryption (p. 434)
- See Also (p. 439)

**At-Rest Encryption Overview**

Amazon ElastiCache for Redis at-rest encryption is an optional feature to increase data security by encrypting on-disk data during sync and backup or snapshot operations. Because there is some processing needed to encrypt and decrypt the data, enabling at-rest encryption can have some performance impact during these operations. You should benchmark your data with and without at-rest encryption to determine the performance impact for your use cases.

**At-Rest Encryption Constraints**

The following constraints on ElastiCache at-rest encryption should be kept in mind when you plan your implementation of ElastiCache encryption at-rest:

- At-rest encryption is supported only on replication groups running Redis version 3.2.6. It is not supported on clusters running Memcached.
- At-rest encryption is supported only for replication groups running in an Amazon VPC.
- At-rest encryption is supported for replication groups running any node type.

You can implement at-rest encryption on a replication group only when creating the replication group. You cannot toggle at-rest encryption on and off on a replication group. For more information, see Enabling At-Rest Encryption on an Existing Redis Cluster (p. 435).

Implementing at-rest encryption can reduce performance during backup and node sync operations. Benchmark at-rest encryption compared to no encryption on your own data to determine its impact on performance for your implementation.

**Enabling At-Rest Encryption**

You can enable ElastiCache at-rest encryption when you create a Redis replication group by setting the parameter `AtRestEncryptionEnabled` to `true`. You can't enable at-rest encryption on existing replication groups.

You can enable at-rest encryption when you create an ElastiCache for Redis replication group using the AWS Management Console, the AWS CLI, or the ElastiCache API.
Enabling At-Rest Encryption on an Existing Redis Cluster

You can only enable at-rest encryption when you create a Redis replication group. If you have an existing replication group on which you want to enable at-rest encryption, do the following.

To enable at-rest encryption on an existing replication group

1. Create a manual backup of your existing replication group. For more information, see Making Manual Backups (p. 297).
2. Create a new replication group by restoring from the backup. On the new replication group, enable at-rest encryption. For more information, see Restoring From a Backup with Optional Cluster Resizing (p. 317).
3. Update the endpoints in your application to point to the new replication group.
4. Delete the old replication group. For more information, see Deleting a Cluster (p. 197) or Deleting a Cluster with Replicas (p. 286).

Enabling At-Rest Encryption Using the AWS Management Console

To enable at-rest encryption when creating a replication group using the AWS Management Console, make the following selections:

- Choose Redis as your engine.
- Choose version 3.2.6 as your engine version.
- Choose Yes from the Encryption at-rest list.

For the step-by-step procedure, see the following:

- Creating a Redis (cluster mode disabled) Cluster (Console) (p. 159)
- Creating a Redis (cluster mode enabled) Cluster (Console) (p. 163)

Enabling At-Rest Encryption Using the AWS CLI

To enable at-rest encryption when creating a Redis cluster using the AWS CLI, use the --at-rest-encryption-enabled parameter when creating a replication group.

Enabling At-Rest Encryption on a Redis (cluster mode disabled) Cluster (CLI)

The following operation creates the Redis (cluster mode disabled) replication group my-classic-rg with three nodes (--num-cache-clusters), a primary and two read replicas. At-rest encryption is enabled for this replication group (--at-rest-encryption-enabled).

The following parameters and their values are necessary to enable encryption on this replication group:
At-Rest Encryption (Redis)

- `--engine` must be `redis`.
- `--at-rest-encryption-enabled` must be included to enable at-rest encryption.

**Example A Redis (cluster mode disabled) Cluster with Replicas**

For Linux, macOS, or Unix:

```bash
aws elasticache create-replication-group \
  --replication-group-id my-classic-rg \
  --replication-group-description "3 node replication group" \
  --cache-node-type cache.m4.large \
  --engine redis \
  --engine-version 2.8.24 \
  --at-rest-encryption-enabled \
  --num-cache-clusters 3 \
  --cache-parameter-group default.redis3.2
```

For Windows:

```bash
aws elasticache create-replication-group ^
  --replication-group-id my-classic-rg ^
  --replication-group-description "3 node replication group" ^
  --cache-node-type cache.m4.large ^
  --engine redis ^
  --engine-version 2.8.24 ^
  --at-rest-encryption-enabled ^
  --num-cache-clusters 3 ^
  --cache-parameter-group default.redis3.2
```

For additional information, see the following:

- Creating a Redis (cluster mode disabled) Cluster with Replicas from Scratch (AWS CLI) (p. 261)
- create-replication-group

**Enabling At-Rest Encryption on a Redis (cluster mode enabled) Cluster (CLI)**

The following operation creates the Redis (cluster mode enabled) replication group `my-clustered-rg` with three node groups or shards (`--num-node-groups`). Each has three nodes, a primary and two read replicas (`--replicas-per-node-group`). At-rest encryption is enabled for this replication group (`--at-rest-encryption-enabled`).

The following parameters and their values are necessary to enable encryption on this replication group:

- `--engine` must be `redis`.
- `--at-rest-encryption-enabled` must be included to enable at-rest encryption.
- To make the replication group a Redis (cluster mode enabled) replication group the following parameters are also required. These don’t affect whether at-rest encryption is enabled.
  - `--engine-version` must be 3.2.6.
  - `--cache-parameter-group` must be `default-redis3.2.cluster.on` or one derived from it to make this a cluster mode enabled replication group.

**Example A Redis (cluster mode enabled) Cluster**

For Linux, macOS, or Unix:

```bash
aws elasticache create-replication-group \
  --replication-group-id my-clustered-rg \
  --replication-group-description "3 node replication group" \
  --cache-node-type cache.m4.large \
  --engine redis \
  --engine-version 3.2.6 \
  --at-rest-encryption-enabled \
  --num-node-groups 3 \
  --replicas-per-node-group 2 \
  --cache-parameter-group default-redis3.2.cluster.on
```
aws elasticache create-replication-group \
  --replication-group-id my-clustered-rg \
  --replication-group-description "redis clustered cluster" \
  --cache-node-type cache.m3.large \
  --num-node-groups 3 \
  --replicas-per-node-group 2 \
  --engine redis \
  --engine-version 3.2.6 \
  --at-rest-encryption-enabled \
  --cache-parameter-group default.redis3.2.cluster.on

For Windows:

aws elasticache create-replication-group ^
  --replication-group-id my-clustered-rg ^
  --replication-group-description "redis clustered cluster" ^
  --cache-node-type cache.m3.large ^
  --num-node-groups 3 ^
  --replicas-per-node-group 2 ^
  --engine redis ^
  --engine-version 3.2.6 ^
  --at-rest-encryption-enabled ^
  --cache-parameter-group default.redis3.2.cluster.on

For additional information, see the following:

• Creating a Redis (cluster mode enabled) Cluster with Replicas from Scratch (AWS CLI) (p. 267)
• create-replication-group

Enabling At-Rest Encryption Using the ElastiCache API

To enable at-rest encryption when creating a Redis replication group using the ElastiCache API, set the parameter AtRestEncryptionEnabled to true with CreateReplicationGroup.

Enabling At-Rest Encryption on a Redis (cluster mode disabled) Cluster (API)

The following operation creates the Redis (cluster mode disabled) replication group my-classic-rg with three nodes (NumCacheClusters), a primary and two read replicas. At-rest encryption is enabled for this replication group (AtRestEncryptionEnabled=true).

The following parameters and their values are necessary to enable encryption on this replication group:

• Engine must be redis.
• AtRestEncryptionEnabled must be true.

Example – A Redis (cluster mode disabled) Cluster with Replicas

Line breaks are added for ease of reading.

https://elasticache.us-west-2.amazonaws.com/
?Action=CreateReplicationGroup
&AtRestEncryptionEnabled=true
&CacheNodeType=cache.m3.large
&CacheParameterGroup=default.redis3.2
&Engine=redis
&EngineVersion=2.8.24
&NumCacheClusters=3
&ReplicationGroupDescription=test%20group
For additional information, see the following:

- Creating a Redis (cluster mode disabled) Cluster with Replicas from Scratch (ElastiCache API) (p. 264)
- CreateReplicationGroup

Enabling At-Rest Encryption on a Redis (cluster mode enabled) Cluster (API)

The following operation creates the Redis (cluster mode enabled) replication group my-clustered-rg with three node groups/shards (NumNodeGroups), each with three nodes, a primary and two read replicas (ReplicasPerNodeGroup). At-rest encryption is enabled for this replication group (AtRestEncryptionEnabled=true).

The following parameters and their values are necessary to enable encryption on this replication group:

- **Engine** must be redis.
- **AtRestEncryptionEnabled** must be true.
- To make the replication group a Redis (cluster mode enabled) cluster the following parameters are also required. These don't affect whether at-rest encryption is enabled.
  - **EngineVersion** must be 3.2.6.
  - **CacheParameterGroup** must be default-redis3.2.cluster.on, or one derived from it.

Example – A Redis (cluster mode enabled) Cluster

Line breaks are added for ease of reading.

https://elasticache.us-west-2.amazonaws.com/
?Action=CreateReplicationGroup
&AtRestEncryptionEnabled=true
&CacheNodeType=cache.m3.large
&CacheParameterGroup=default.redis3.2.cluster.on
&Engine=redis
&EngineVersion=3.2.6
&NumNodeGroups=3
&ReplicasPerNodeGroup=2
&ReplicationGroupDescription=test%20group
&ReplicationGroupId=my-clustered-rg
&Version=2015-02-02
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&Timestamp=20150202T192317Z
&X-Amz-Credential=<credential>

For additional information, see the following:

- Creating a Redis (cluster mode enabled) Cluster with Replicas from Scratch (ElastiCache API) (p. 271)
- CreateReplicationGroup
HIPAA Compliance for Amazon ElastiCache for Redis

You can find information about Amazon ElastiCache for Redis requirements for HIPAA compliance at ElastiCache for Redis HIPAA Requirements (p. 439). There is no additional charge beyond the normal ElastiCache for Redis charges for making your cluster HIPAA-compliant.

The AWS HIPAA compliance program includes Amazon ElastiCache for Redis as a HIPAA Eligible Service. If you have an executed Business Associate Agreement (BAA) with AWS, you can use Amazon ElastiCache for Redis to build your HIPAA-compliant applications containing protected health information (PHI). For more information, see HIPAA Compliance. AWS Services in Scope have been fully assessed by a third-party auditor and result in a certification, attestation of compliance, or Authority to Operate (ATO). For more information, see AWS Services in Scope by Compliance Program.

ElastiCache for Redis HIPAA Requirements

To enable HIPAA support on your ElastiCache for Redis cluster, in addition to executing the BAA, your cluster and nodes within the cluster must satisfy the following requirements:

- **Data security requirements** – Your cluster must enable in-transit encryption, at-rest encryption, and Redis AUTH. For more information, see the following:
  - Amazon ElastiCache for Redis In-Transit Encryption (p. 429)
  - Amazon ElastiCache for Redis At-Rest Encryption (p. 434)
  - Authenticating Users with AUTH (Redis) (p. 427)

- **Engine version requirements** – Your cluster must be running ElastiCache for Redis version 3.2.6 to qualify for HIPAA compliance. For more information, see ElastiCache for Redis Version 3.2.6 (Enhanced) (p. 50).

- **Node type requirements** – Your cluster must be running a current-generation node type—M3, M4, T2, or R3. For more information, see the following:
  - Supported Node Types (p. 117)
  - Choosing Your Node Size for Redis Clusters (p. 100)

When you have created a compliant ElastiCache for Redis cluster, you can start storing PHI. If you want, you can seed the new cluster with data from an existing cluster. For more information, see the following:

- Creating a Cluster (p. 156)
- Creating a Redis Cluster with Replicas from Scratch (p. 260)
- Seeding a New Cluster with an Externally Created Backup (Redis) (p. 320)

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

- AWS Cloud Compliance
- Shared Responsibility Model
- HIPAA Compliance
• AWS Services in Scope by Compliance Program
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 an Amazon VPC, there are a number of ways to access a cluster from outside AWS, one of which is using a Network Address Translation (NAT) instance to provide outside access, as described in this topic. For other patterns, see Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center (p. 397).

Topics
- Requirements (p. 441)
- Considerations (p. 441)
- Limitations (p. 441)
- How to Access ElastiCache Resources from Outside AWS (p. 442)
- Related topics (p. 444)

Requirements

The following requirements must be met for you to access your ElastiCache resources from outside AWS using a NAT instance. For other ways of accessing ElastiCache from outside AWS see, Accessing an ElastiCache Cluster from an Application Running in a Customer's Data Center (p. 397).

- The cluster must reside within an Amazon VPC and be accessed through a Network Address Translation (NAT) instance.
- The NAT instance must be launched in the same Amazon 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 Amazon 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.
How to Access ElastiCache Resources from Outside AWS

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

**Tip**
You can modify the following process to work for any Amazon ElastiCache cluster. Just substitute the cluster's port number for the port numbers in the example.

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 choose 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

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom TCP Rule</td>
<td>TCP</td>
<td>11211-11213</td>
<td>198.51.100.27/32</td>
</tr>
<tr>
<td>SSH</td>
<td>TCP</td>
<td>22</td>
<td>198.51.100.27/32</td>
</tr>
</tbody>
</table>

- An outbound rule to allow TCP connections to each forwarded cache port (11211-11213).

NAT Instance Security Group - Outbound Rule

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Destination</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom TCP Rule</td>
<td>TCP</td>
<td>11211-11213</td>
<td>sg-bd56b7da</td>
</tr>
</tbody>
</table>

- 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

<table>
<thead>
<tr>
<th>Type</th>
<th>Protocol</th>
<th>Port Range</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Custom TCP Rule</td>
<td>TCP</td>
<td>11211-11213</td>
<td>sg-ce56b7a9</td>
</tr>
</tbody>
</table>

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
```
Monitoring Usage, Events, and Costs

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

**Topics**
- Monitoring Use with CloudWatch Metrics (p. 446)
- Monitoring ElastiCache Events (p. 457)
- Monitoring Costs with Cost Allocation Tags (p. 466)
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. 452).

Topics

- Dimensions for ElastiCache Metrics (p. 446)
- Host-Level Metrics (p. 446)
- Metrics for Memcached (p. 447)
- Metrics for Redis (p. 449)
- Which Metrics Should I Monitor? (p. 452)
- Choosing Metric Statistics and Periods (p. 454)
- Monitoring CloudWatch Cache Cluster and Cache Node Metrics (p. 454)

Dimensions for ElastiCache Metrics

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

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

Contents

- Host-Level Metrics (p. 446)
- Metrics for Memcached (p. 447)
- Metrics for Redis (p. 449)
- Which Metrics Should I Monitor?

Host-Level Metrics

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

See Also

- Metrics for Memcached (p. 447)
- Metrics for Redis (p. 449)
### Metrics for Memcached

The AWS/ElastiCache namespace includes the following metrics that are derived from the Memcached `stats` command. Each metric is calculated at the cache node level.


**See Also**
- Host-Level Metrics (p. 446)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesReadIntoMemcached</td>
<td>The number of bytes that have been read from the network by the cache node.</td>
<td>Bytes</td>
</tr>
<tr>
<td>BytesUsedForCacheItems</td>
<td>The number of bytes used to store cache items.</td>
<td>Bytes</td>
</tr>
<tr>
<td>BytesWrittenOutFromMemcached</td>
<td>The number of bytes that have been written to the network by the cache node.</td>
<td>Bytes</td>
</tr>
<tr>
<td>CasBadval</td>
<td>The number of CAS (check and set) requests the cache has received where the Cas value did not match the Cas value stored.</td>
<td>Count</td>
</tr>
<tr>
<td>CasHits</td>
<td>The number of Cas requests the cache has received where the requested key was found and the Cas value matched.</td>
<td>Count</td>
</tr>
<tr>
<td>CasMisses</td>
<td>The number of Cas requests the cache has received where the key requested was not found.</td>
<td>Count</td>
</tr>
<tr>
<td>CmdFlush</td>
<td>The number of flush commands the cache has received.</td>
<td>Count</td>
</tr>
<tr>
<td>CmdGet</td>
<td>The number of get commands the cache has received.</td>
<td>Count</td>
</tr>
<tr>
<td>CmdSet</td>
<td>The number of set commands the cache has received.</td>
<td>Count</td>
</tr>
</tbody>
</table>
### Metrics for Memcached

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrConnections</td>
<td>A count of the number of connections connected to the cache at an instant in time. ElastiCache uses two to three of the connections to monitor the cluster in each case.</td>
<td>Count</td>
</tr>
<tr>
<td>CurrItems</td>
<td>A count of the number of items currently stored in the cache.</td>
<td>Count</td>
</tr>
<tr>
<td>DecrHits</td>
<td>The number of decrement requests the cache has received where the requested key was found.</td>
<td>Count</td>
</tr>
<tr>
<td>DecrMisses</td>
<td>The number of decrement requests the cache has received where the requested key was not found.</td>
<td>Count</td>
</tr>
<tr>
<td>DeleteHits</td>
<td>The number of delete requests the cache has received where the requested key was found.</td>
<td>Count</td>
</tr>
<tr>
<td>DeleteMisses</td>
<td>The number of delete requests the cache has received where the requested key was not found.</td>
<td>Count</td>
</tr>
<tr>
<td>Evictions</td>
<td>The number of non-expired items the cache evicted to allow space for new writes.</td>
<td>Count</td>
</tr>
<tr>
<td>GetHits</td>
<td>The number of get requests the cache has received where the key requested was found.</td>
<td>Count</td>
</tr>
<tr>
<td>GetMisses</td>
<td>The number of get requests the cache has received where the key requested was not found.</td>
<td>Count</td>
</tr>
<tr>
<td>IncrHits</td>
<td>The number of increment requests the cache has received where the key requested was found.</td>
<td>Count</td>
</tr>
<tr>
<td>IncrMisses</td>
<td>The number of increment requests the cache has received where the key requested was not found.</td>
<td>Count</td>
</tr>
<tr>
<td>Reclaimed</td>
<td>The number of expired items the cache evicted to allow space for new writes.</td>
<td>Count</td>
</tr>
</tbody>
</table>

For Memcached 1.4.14, the following additional metrics are provided.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesUsedForHash</td>
<td>The number of bytes currently used by hash tables.</td>
<td>Bytes</td>
</tr>
<tr>
<td>CmdConfigGet</td>
<td>The cumulative number of <code>config get</code> requests.</td>
<td>Count</td>
</tr>
<tr>
<td>CmdConfigSet</td>
<td>The cumulative number of <code>config set</code> requests.</td>
<td>Count</td>
</tr>
<tr>
<td>CmdTouch</td>
<td>The cumulative number of <code>touch</code> requests.</td>
<td>Count</td>
</tr>
<tr>
<td>CurrConfig</td>
<td>The current number of configurations stored.</td>
<td>Count</td>
</tr>
<tr>
<td>EvictedUnfetched</td>
<td>The number of valid items evicted from the least recently used cache (LRU) which were never touched after being set.</td>
<td>Count</td>
</tr>
</tbody>
</table>
## Metrics for Redis

The **AWS/ElastiCache** namespace includes the following Redis metrics.

With the exception of `ReplicationLag`, these metrics are derived from the Redis `info` command. Each metric is calculated at the cache node level.


### See Also

---

The **AWS/ElastiCache** namespace includes the following calculated cache-level metrics.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExpiredUnfetched</td>
<td>The number of expired items reclaimed from the LRU which were never touched after being set.</td>
<td>Count</td>
</tr>
<tr>
<td>SlabsMoved</td>
<td>The total number of slab pages that have been moved.</td>
<td>Count</td>
</tr>
<tr>
<td>TouchHits</td>
<td>The number of keys that have been touched and were given a new expiration time.</td>
<td>Count</td>
</tr>
<tr>
<td>TouchMisses</td>
<td>The number of items that have been touched, but were not found.</td>
<td>Count</td>
</tr>
</tbody>
</table>

### Metrics for Redis

The **AWS/ElastiCache** namespace includes the following Redis metrics.

With the exception of `ReplicationLag`, these metrics are derived from the Redis `info` command. Each metric is calculated at the cache node level.


### See Also
### Host-Level Metrics (p. 446)

<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>BytesUsedForCache</td>
<td>The total number of bytes allocated by Redis.</td>
<td>Bytes</td>
</tr>
<tr>
<td>CacheHits</td>
<td>The number of successful key lookups.</td>
<td>Count</td>
</tr>
<tr>
<td>CacheMisses</td>
<td>The number of unsuccessful key lookups.</td>
<td>Count</td>
</tr>
<tr>
<td>CurrConnections</td>
<td>The number of client connections, excluding connections from read replicas. ElastiCache uses two to three of the connections to monitor the cluster in each case.</td>
<td>Count</td>
</tr>
<tr>
<td>EngineCPUUtilization</td>
<td>CPU utilization of the single core that Redis is running on. Since Redis is single threaded, this is the percent of your thread's capacity that is being used. This metric is available on clusters created or replaced after November 1, 2017.</td>
<td>Percent</td>
</tr>
<tr>
<td>Evictions</td>
<td>The number of keys that have been evicted due to the maxmemory limit.</td>
<td>Count</td>
</tr>
<tr>
<td>HyperLogLogBasedCmds</td>
<td>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).</td>
<td>Count</td>
</tr>
<tr>
<td>NewConnections</td>
<td>The total number of connections that have been accepted by the server during this period.</td>
<td>Count</td>
</tr>
<tr>
<td>Reclaimed</td>
<td>The total number of key expiration events.</td>
<td>Count</td>
</tr>
<tr>
<td>ReplicationBytes</td>
<td>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.</td>
<td>Bytes</td>
</tr>
<tr>
<td>ReplicationLag</td>
<td>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.</td>
<td>Seconds</td>
</tr>
<tr>
<td>SaveInProgress</td>
<td>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.</td>
<td>Count</td>
</tr>
</tbody>
</table>

These are aggregations of certain kinds of commands, derived from `info commandstats`:
<table>
<thead>
<tr>
<th>Metric</th>
<th>Description</th>
<th>Unit</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrItems</td>
<td>The number of items in the cache. This is derived from the Redis keyspace statistic, summing all of the keys in the entire keyspace.</td>
<td>Count</td>
</tr>
<tr>
<td>GetTypeCmds</td>
<td>The total number of get types of commands. This is derived from the Redis commandstats statistic by summing all of the get types of commands (get, mget, hget, etc.)</td>
<td>Count</td>
</tr>
<tr>
<td>HashBasedCmds</td>
<td>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.</td>
<td>Count</td>
</tr>
<tr>
<td>KeyBasedCmds</td>
<td>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.</td>
<td>Count</td>
</tr>
<tr>
<td>ListBasedCmds</td>
<td>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.</td>
<td>Count</td>
</tr>
<tr>
<td>SetBasedCmds</td>
<td>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.</td>
<td>Count</td>
</tr>
<tr>
<td>setTypeCmds</td>
<td>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.)</td>
<td>Count</td>
</tr>
<tr>
<td>SortedSetBasedCmds</td>
<td>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.</td>
<td>Count</td>
</tr>
<tr>
<td>StringBasedCmds</td>
<td>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.</td>
<td>Count</td>
</tr>
</tbody>
</table>
Amazon ElastiCache User Guide
Which Metrics Should I Monitor?

The following CloudWatch metrics offer good insight into ElastiCache performance. In most cases, we recommend that you set CloudWatch alarms for these metrics so that you can take corrective action before performance issues occur.

**Metrics to Monitor**
- **CPUUtilization (p. 452)**
- **SwapUsage (p. 452)**
- **Evictions (p. 452)**
- **CurrConnections (p. 453)**

**CPUUtilization**

This is a host-level metric reported as a percent. For more information, see Host-Level Metrics (p. 446).

- **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, depending on your cluster configuration, we recommend that you:
- **Redis (cluster mode disabled) clusters**: scale up by using a larger cache instance type.
- **Redis (cluster mode enabled) clusters**: add more shards to distribute the write workload across more primary nodes.

**SwapUsage**

This is a host-level metric reported in bytes. For more information, see Host-Level Metrics (p. 446).

- **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 you exceed your chosen threshold, scale your 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. 454).

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

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, macOS, 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

API Version 2015-02-02
&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 (p. 457)
- Viewing ElastiCache Events (p. 460)
- Event Notifications and Amazon SNS (p. 462)

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 choosing 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)**

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, macOS, or Unix:

```
aws elasticache modify-cache-cluster \
  --cache-cluster-id my-cluster \
```

For Windows:

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

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

2. Choose the engine running on the cluster you want to modify notifications for, either **Memcached** or **Redis**.

   A list of clusters running the chosen 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, macOS, or Unix:

```bash
aws elasticache modify-cache-cluster \
    --cache-cluster-id my-cluster \ 
    --notification-topic-status inactive
```

For Windows:

```bash
aws elasticache modify-cache-cluster ^ \
    --cache-cluster-id my-cluster ^ \
    --notification-topic-status inactive
```

### Enabling and Disabling Amazon SNS Notifications (ElastiCache API)

To disable Amazon SNS notifications, call the `ModifyCacheCluster` action with the following parameters:

- **CacheClusterId=my-cluster**
- **NotificationTopicStatus=inactive**

This call returns output similar to the following:

**Example**

```xml
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>
```

API Version 2015-02-02
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

2. In the 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 up to 40 cache cluster events.

```bash
aws elasticache describe-events --source-type cache-cluster --max-items 40
```

The following code lists all events for the past 24 hours (1440 minutes).

```bash
aws elasticache describe-events --duration 1440
```

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

```json
{
  "Events": [
    {
      "Date": "2017-03-30T14:39:14.295Z",
      "Message": "Automatic failover has been turned on for replication group redis12",
      "SourceIdentifier": "redis12",
      "SourceType": "replication-group"
    },
    {
      "Date": "2017-03-29T22:17:37.781Z",
      "Message": "Added cache node 0001 in Availability Zone us-west-2a",
      "SourceIdentifier": "redis01",
      "SourceType": "cache-cluster"
    }
  ]
}
```
Viewing ElastiCache Events

For more information, such as available parameters and permitted parameter values, see `describe-events`.

**Viewing ElastiCache Events (ElastiCache API)**

To generate a list of ElastiCache events using the ElastiCache API, use the `DescribeEvents` action. You can use optional parameters to control the type of events listed, the time frame of the events listed, the maximum number of events to list, and more.

The following code lists the 40 most recent cache-cluster events.

```plaintext
https://elasticache.us-west-2.amazonaws.com/?Action=DescribeEvents
&MaxRecords=40
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&SourceType=cache-cluster
&Timestamp=20150202T182218Z
&Version=2015-02-02
&X-Amz-Credential=<credential>
```

The following code lists the cache-cluster events for the past 24 hours (1440 minutes).

```plaintext
https://elasticache.us-west-2.amazonaws.com/?Action=DescribeEvents
&Duration=1440
&SignatureVersion=4
&SignatureMethod=HmacSHA256
&SourceType=cache-cluster
&Timestamp=20150202T182218Z
&Version=2015-02-02
&X-Amz-Credential=<credential>
```

The above actions should produce output similar to the following.

```xml
<DescribeEventsResponse xmlns="http://elasticache.amazonaws.com/doc/2015-02-02/>
<DescribeEventsResult>
<Events>
<Event>
<Message>Cache cluster created</Message>
<SourceType>cache-cluster</SourceType>
<Date>2015-02-02T18:22:18Z</Date>
<SourceIdentifier>my-redis-primary</SourceIdentifier>
</Event>
(...output omitted...)
</Events>
</DescribeEventsResult>
<ResponseMetadata>
<RequestId>e21c81b4-b9cd-11e3-8a16-7978bb24ffdf</RequestId>
</ResponseMetadata>
```

API Version 2015-02-02
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:

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ElastiCache:AddCacheNodeComplete</td>
<td>Finished modifying number of nodes from %d to %d</td>
<td>A cache node has been added to the cache cluster and is ready for use.</td>
</tr>
<tr>
<td>ElastiCache:AddCacheNodeFailed due to insufficient free IP addresses</td>
<td>&quot;Failed to modify number of nodes from %d to %d due to insufficient free IP addresses&quot;</td>
<td>A cache node could not be added because there are not enough available IP addresses.</td>
</tr>
<tr>
<td>ElastiCache:CacheClusterParametersChanged</td>
<td>Changed parameter %s to %s</td>
<td>One or more cache cluster parameters have been changed.</td>
</tr>
<tr>
<td></td>
<td>In case of create, also send &quot;Updated to use a CacheParameterGroup %s&quot;</td>
<td></td>
</tr>
</tbody>
</table>

API Version 2015-02-02
<table>
<thead>
<tr>
<th>Event Name</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ElastiCache:CacheClusterProvisioningComplete</td>
<td>&quot;Cache cluster created&quot;</td>
<td>The provisioning of a cache cluster is completed, and the cache nodes in the cache cluster are ready to use.</td>
</tr>
<tr>
<td>ElastiCache:CacheClusterProvisioningFailed due to incompatible network state</td>
<td>&quot;Failed to create the cache cluster due to incompatible network state&quot;</td>
<td>An attempt was made to launch a new cache cluster into a nonexistent virtual private cloud (VPC).</td>
</tr>
<tr>
<td>ElastiCache:CacheClusterRestoreFailed</td>
<td>&quot;Restore from %s failed for node %s&quot;</td>
<td>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 be restore-failed. You will need to delete the cache cluster and start over. For more information, see Seeding a New Cluster with an Externally Created Backup (Redis) (p. 320).</td>
</tr>
<tr>
<td>ElastiCache:CacheClusterScalingComplete</td>
<td>&quot;Succeeded applying modification to cache node type to %s.&quot;</td>
<td>Scale up for cache-cluster completed successfully.</td>
</tr>
<tr>
<td>ElastiCache:CacheClusterScalingFailed</td>
<td>&quot;Failed applying modification to cache node type to %s.&quot;</td>
<td>Scale-up operation on cache-cluster failed.</td>
</tr>
<tr>
<td>ElastiCache:CacheClusterSecurityGroupModified</td>
<td>&quot;Applied change to security group&quot;</td>
<td>One of the following events has occurred:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• The list of cache security groups authorized for the cache cluster has been modified.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• One or more new EC2 security groups have been authorized on any of the cache security groups associated with the cache cluster.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• One or more EC2 security groups have been revoked from any of the cache security groups associated with the cache cluster.</td>
</tr>
<tr>
<td>Event Name</td>
<td>Message</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------------------</td>
<td>-------------------------------------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>ElastiCache:CacheNodeReplaceComplete</td>
<td>&quot;Finished recovery for cache nodes %s&quot;</td>
<td>ElastiCache has detected that the host running a cache node is degraded or unreachable and has completed replacing the cache node. 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.</td>
</tr>
<tr>
<td>ElastiCache:CacheNodesRebooted</td>
<td>&quot;Cache node %s restarted&quot;</td>
<td>One or more cache nodes has been rebooted. Message (Memcached): &quot;Cache node %s shutdown&quot; Then a second message: &quot;Cache node %s restarted&quot;</td>
</tr>
<tr>
<td>ElastiCache:CreateReplicationGroupComplete</td>
<td>&quot;Replication group %s created&quot;</td>
<td>The replication group was successfully created.</td>
</tr>
<tr>
<td>ElastiCache:CreateReplicationGroupFailed</td>
<td>Updated to create replication group %s due to unsuccessful creation of its cache cluster(s). &quot;Deleting all cache clusters belonging to this replication group.&quot;</td>
<td>The replication group was not created.</td>
</tr>
<tr>
<td>ElastiCache:DeleteCacheClusterComplete</td>
<td>The cluster deleted</td>
<td>The deletion of a cache cluster and all associated cache nodes has completed.</td>
</tr>
<tr>
<td>ElastiCache:FailoverComplete</td>
<td>&quot;Failover to replica node %s completed&quot;</td>
<td>Failover over to a replica node was successful.</td>
</tr>
<tr>
<td>ElastiCache:NodeReplacementCanceled</td>
<td>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</td>
<td>A node in your cluster that was scheduled for replacement is no longer scheduled for replacement.</td>
</tr>
</tbody>
</table>
## Event Notifications and Amazon SNS

<table>
<thead>
<tr>
<th>Event Name</th>
<th>Message</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ElastiCache:NodeReplacementRescheduled</strong></td>
<td><em>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</em>&quot;</td>
<td>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. 120).</td>
</tr>
<tr>
<td><strong>ElastiCache:NodeReplacementScheduled</strong></td>
<td><em>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</em>&quot;</td>
<td>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. 120).</td>
</tr>
<tr>
<td><strong>ElastiCache:RemoveCacheNodeComplete</strong></td>
<td><em>Removed cache nodes %s</em>&quot;</td>
<td>A cache node has been removed from the cache cluster.</td>
</tr>
<tr>
<td><strong>ElastiCache:ReplicationGroupScalingComplete</strong></td>
<td><em>Succeeded applying modification to cache node type to %s.</em>&quot;</td>
<td>Scale-up operation on replication group completed successfully.</td>
</tr>
<tr>
<td><strong>ElastiCache:ReplicationGroupScalingFailed</strong></td>
<td><em>Failed applying modification to cache node type to %s.</em>&quot;</td>
<td>Scale-up operation on replication group failed.</td>
</tr>
<tr>
<td><strong>ElastiCache:SnapshotComplete</strong></td>
<td>&quot;Snapshot succeeded for snapshot with ID '%s' of cache cluster with ID '%s'&quot;</td>
<td>A cache snapshot has completed successfully.</td>
</tr>
<tr>
<td><strong>ElastiCache:SnapshotFailed</strong></td>
<td>&quot;Snapshot failed for snapshot with ID '%s' of cache cluster with ID '%s'&quot;</td>
<td>A cache snapshot has failed. See the cluster's cache events for more a detailed cause. If you describe the snapshot, see [DescribeSnapshots](p. 120), the status will be failed.</td>
</tr>
</tbody>
</table>

### Related topics

- Viewing ElastiCache Events (p. 460)
Monitoring Costs with Cost Allocation Tags

When you add cost allocation tags to your resources in Amazon ElastiCache, you can track costs by grouping expenses on your invoices by resource tag values.

An ElastiCache cost allocation tag is a key-value pair that you define and associate with an ElastiCache resource. The key and value are case-sensitive. You can use a tag key to define a category, and the tag value can be an item in that category. For example, you might define a tag key of CostCenter and a tag value of 10010, indicating that the resource is assigned to the 10010 cost center. You can also use tags to designate resources as being used for test or production by using a key such as Environment and values such as test or production. We recommend that you use a consistent set of tag keys to make it easier to track costs associated with your resources.

Use cost allocation tags to organize your AWS bill to reflect your own cost structure. To do this, sign up to get your AWS account bill with tag key values included. Then, to see the cost of combined resources, organize your billing information according to resources with the same tag key values. For example, you can tag several resources with a specific application name, and then organize your billing information to see the total cost of that application across several services.

You can also combine tags to track costs at a greater level of detail. For example, to track your service costs by region you might use the tag keys Service and Region. On one resource you might have the values ElastiCache and Asia Pacific (Singapore), and on another resource the values ElastiCache and EU (Frankfurt). You can then see your total ElastiCache costs broken out by region. For more information, see Use Cost Allocation Tags in the AWS Billing and Cost Management User Guide.

You can add ElastiCache cost allocation tags to Memcached clusters, Redis nodes, and backups. When you add, list, modify, copy, or remove a tag, the operation is applied only to the specified cluster, node, or backup.

Tags added to backups are not used for cost allocation reports. Tags on backups are used to retain or restore tags on clusters. When you create a backup, the tags on the cluster are copied to the backup. When you restore from a backup, the tags on the backup are copied to the cluster.

Characteristics of ElastiCache cost allocation tags

- Cost allocation tags are applied to ElastiCache resources which are specified in CLI and API operations as an ARN. The resource-type will be a "cluster" or a "snapshot".


  - **Memcached**: Tags are applied to clusters.


  - **Redis**: Tags are applied to individual nodes. Because of this, nodes in Redis clusters with replication can have different tags.

    Sample arns

    - Redis (cluster mode disabled) no replication:


    - Redis (cluster mode disabled) with replication:


    - Redis (cluster mode enabled):

Managing Tags Using the Console

- **Backups (Redis)**: Tags are applied to the backup.
  

- The tag key is the required name of the tag. The key's string value can be from 1 to 128 Unicode characters long and cannot be prefixed with `aws:`. The string can contain only the set of Unicode letters, digits, blank spaces, underscores (_), periods (.), colons (:), backslashes (\), equal signs (=), plus signs (+), hyphens (-), or at signs (@).

- The tag value is the optional value of the tag. The value's string value can be from 1 to 256 Unicode characters in length and cannot be prefixed with `aws:`. The string can contain only the set of Unicode letters, digits, blank spaces, underscores (_), periods (.), colons (:), backslashes (\), equal signs (=), plus signs (+), hyphens (-), or at signs (@).

- A tag can be applied to an ElastiCache resource; a cluster (Memcached), a node (Redis), or a backup (Redis).

- An ElastiCache resource can have a maximum of 50 tags.

- Values do not have to be unique in a tag set. For example, you can have a tag set where the keys `Service` and `Application` both have the value `ElastiCache`.

  AWS does not apply any semantic meaning to your tags. Tags are interpreted strictly as character strings. AWS does not automatically set any tags on any ElastiCache resource.

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

**Topics**

- Managing Your Tags Using the ElastiCache Console (p. 467)
- Managing Your Cost Allocation Tags Using the AWS CLI (p. 471)
- Managing Your Cost Allocation Tags Using the ElastiCache API (p. 474)
- Copying Tags to Your ElastiCache Resource (p. 476)

**Managing Your Tags Using the ElastiCache Console**

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

**Topics**

- Managing Tags on a Memcached Cluster (Console) (p. 468)
- Managing Tags on a Redis (cluster mode disabled) Cluster (Console) (p. 468)
- Managing Tags on a Redis (cluster mode enabled) Cluster (Console) (p. 469)
- Managing Tags on a Backup (Console) (p. 470)
Managing Tags on a Memcached Cluster (Console)

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

To add, modify, or remove a tag on a Memcached cluster using the ElastiCache management console

2. Choose Memcached.
3. Choose the box to the left of the cluster's name you want to add tags to.

   After you choose the cluster, you can see the tag names and values currently on this resource at the bottom of the details area.
4. Choose Manage Tags, and then use the dialog box to manage your tags.

   ![Manage Tags dialog box]

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

   To add, modify, or remove tags

   • To add a tag: In the Key column, type a key name in the box that displays Add key and an optional value in the box to the right of the key name.
   • To modify a tag: In the Value column, type a new value or remove the existing value for the tag.
   • To remove a tag: Choose the X to the right of the tag.

6. When you’re finished, choose Apply Changes.

Managing Tags on a Redis (cluster mode disabled) Cluster (Console)

The following procedure walks you through viewing, adding, modifying, or deleting one or more cost allocation tags on the nodes in a Redis (cluster mode disabled) cluster using the ElastiCache management console.
To add, modify, or remove a tag on a Redis (cluster mode disabled) node using the ElastiCache management console

2. Choose Redis.
3. Choose the name of the cluster on which you want to add, modify, or remove tags.
4. For each node in the cluster you want to view, add, modify, or remove tags on, do the following:
   a. Choose the box to the left of the node's name.
   b. Choose Actions, then choose Manage tags.

![Manage Tags](image)

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

Managing Tags on a Redis (cluster mode enabled) Cluster (Console)

The following procedure walks you through viewing, adding, modifying, or deleting one or more cost allocation tags on the nodes in a Redis (cluster mode enabled) cluster using the ElastiCache management console.
Managing Tags Using the Console

To add a tag to a Redis (cluster mode enabled) node using the ElastiCache management console

2. Choose Redis.
3. Choose the name of the cluster on which you want to add, modify, or remove tags.
4. Choose the name of the shard on which you want to add, modify, or remove tags.
5. For each node in the shard you want to view, add, modify, or remove tags on, do the following:
   a. Choose the box to the left of the node's name.
   b. Choose Actions, then choose Manage tags.
   c. For each tag you want to add, modify, or remove:
      To add, modify, or remove tags on a node
      - To add a tag: In the Key column, type a key name in the box that displays Add key. To add an optional value, tab to the Empty value box and type a value for the key.
      - To modify a tag: In the Value column to the right of the key name, type a new value or remove the existing value for the tag.
      - To remove a tag: Choose the X to the right of the tag.
   d. When you're finished, choose Apply Changes.

Managing Tags on a Backup (Console)

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

2. Choose Backups.
3. Choose the box to the left of the backup's name you want to add tags to.
After you choose the cluster, you can see the tag names and values currently on this resource at the bottom of the details area.

4. Choose **Manage Tags**, and then use the dialog box to manage your tags.

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

   **To add, modify, or remove tags**
   - **To add a tag**: In the **Key** column, type a key name in the box that displays *Add key* and an optional value in the box to the right of the key name.
   - **To modify a tag**: In the **Value** column, type a new value or remove the existing value for the tag.
   - **To remove a tag**: Choose the *X* to the right of the tag.

6. When you're finished, choose **Apply Changes**.

**Managing Your Cost Allocation Tags Using the AWS CLI**

You can use the AWS CLI to add, modify, or remove cost allocation tags.

Cost allocation tags are applied to ElastiCache resources. What that resource is and how it is specified in an ARN depends on the engine and structure of the cluster.

- **Memcached**: Tags are applied to clusters.
  

- **Redis**: Tags are applied to individual nodes. Because of this, nodes in Redis clusters with replication can have different tags.

  Sample arns
  - Redis (cluster mode disabled) no replication:
    
  - Redis (cluster mode disabled) with replication:
• Redis (cluster mode enabled):
• Backups (Redis): Tags are applied to the backup.

Topics
• Listing Tags Using the AWS CLI (p. 472)
• Adding Tags Using the AWS CLI (p. 473)
• Modifying Tags Using the AWS CLI (p. 473)
• Removing Tags Using the AWS CLI (p. 474)

Listing Tags Using the AWS CLI
You can use the AWS CLI to list tags on an existing ElastiCache resource by using the `list-tags-for-resource` operation.

The following code uses the AWS CLI to list the tags on the Memcached cluster `myCluster` in the us-west-2 region.

For Linux, macOS, or Unix:

```bash
aws elasticache list-tags-for-resource \
```

For Windows:

```bash
aws elasticache list-tags-for-resource ^ \
```

Output from this operation will look something like the following, a list of all the tags on the resource.

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

If there are no tags on the resource, the output will be an empty TagList.

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

For more information, see the AWS CLI for ElastiCache `list-tags-for-resource`.  

API Version 2015-02-02
472
Adding Tags Using the AWS CLI

You can use the AWS CLI to add tags to an existing ElastiCache resource by using the `add-tags-to-resource` CLI operation. If the tag key does not exist on the resource, the key and value are added to the resource. If the key already exists on the resource, the value associated with that key is updated to the new value.

The following code uses the AWS CLI to add the keys `Service` and `Region` with the values `elasticache` and `us-west-2` respectively to the resource `myCluster` in the us-west-2 region.

For Linux, macOS, or Unix:

```
aws elasticache add-tags-to-resource \
--tags Key=Service,Value=elasticache \
    Key=Region,Value=us-west-2
```

For Windows:

```
aws elasticache add-tags-to-resource ^
    --tags Key=PM ^
    Key=Region,Value=us-west-2
```

Output from this operation will look something like the following, a list of all the tags on the resource following the operation.

```
{
    "TagList": [
        {
            "Value": "10110",
            "Key": "CostCenter"
        },
        {
            "Value": "EC2",
            "Key": "Service"
        },
        {
            "Value": "",
            "Key": "PM"
        },
        {
            "Value": "us-west-2",
            "Key": "Region"
        }
    ]
}
```

For more information, see the AWS CLI for ElastiCache `add-tags-to-resource`.

You can also use the AWS CLI to add tags to a cluster when you create a new cluster by using the operation `create-cache-cluster`, or when you create a new replication group by using the operation `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.

Modifying Tags Using the 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 either add a new tag and value or to change the value associated with an existing tag.
- Use **remove-tags-from-resource** to remove specified tags from the resource.

Output from either operation will be a list of tags and their values on the specified resource.

### Removing Tags Using the AWS CLI

You can use the AWS CLI to remove tags from an existing ElastiCache resource by using the **remove-tags-from-resource** operation.

The following code uses the AWS CLI to remove the tags with the keys **Service** and **Region** from the resource **myCluster** in the us-west-2 region.

For Linux, macOS, or Unix:

```shell
aws elasticache remove-tags-from-resource
  --tag-keys PM Service
```

For Windows:

```shell
aws elasticache remove-tags-from-resource
  --tag-keys PM Service
```

Output from this operation will look something like the following, a list of all the tags on the resource following the operation.

```json
{
  "TagList": [
    {
      "Value": "10110",
      "Key": "CostCenter"
    },
    {
      "Value": "us-west-2",
      "Key": "Region"
    }
  ]
}
```

For more information, see the AWS CLI for ElastiCache **remove-tags-from-resource**.

### Managing Your Cost Allocation Tags Using the ElastiCache API

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

Cost allocation tags are applied to ElastiCache resources. What that resource is and how it is specified in an ARN depends on the engine and structure of the cluster.

- **Memcached**: Tags are applied to clusters.

- **Redis**: Tags are applied to individual nodes. Because of this, nodes in Redis clusters with replication can have different tags.

Sample arns
- Redis (cluster mode disabled) no replication:
  

- Redis (cluster mode disabled) with replication:


- Redis (cluster mode enabled):


- **Backups (Redis)**: Tags are applied to the backup.


**Topics**

- Listing Tags Using the ElastiCache API (p. 475)
- Adding Tags Using the ElastiCache API (p. 475)
- Modifying Tags Using the ElastiCache API (p. 476)
- Removing Tags Using the ElastiCache API (p. 476)

**Listing Tags Using the ElastiCache API**

You can use the ElastiCache API to list tags on an existing resource by using the `ListTagsForResource` operation.

The following code uses the ElastiCache API to list the tags on the resource `myCluster` in the us-west-2 region.

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

**Adding Tags Using the ElastiCache API**

You can use the ElastiCache API to add tags to an existing ElastiCache resource by using the `AddTagsToResource` operation. If the tag key does not exist on the resource, the key and value are added to the resource. If the key already exists on the resource, the value associated with that key is updated to the new value.

The following code uses the ElastiCache API to add the keys `Service` and `Region` with the values `elasticache` and `us-west-2` respectively to the resource `myCluster` in the us-west-2 region.

```
https://elasticache.us-west-2.amazonaws.com/
  ?Action=AddTagsToResource
```

API Version 2015-02-02

475
Modifying Tags Using the 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 operation to either add a new tag and value or to change the value of an existing tag.
- Use RemoveTagsFromResource to remove tags from the resource.

Output from either operation will be a list of tags and their values on the specified resource.

Use RemoveTagsFromResource to remove tags from the resource.

Removing Tags Using the ElastiCache API

You can use the ElastiCache API to remove tags from an existing ElastiCache resource by using the RemoveTagsFromResource operation.

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.

```
https://elasticache.us-west-2.amazonaws.com/
?Action=RemoveTagsFromResource
&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 backup, if there are any tags on the source backup, they are copied to the copy.
- **CreateSnapshot** or **create-snapshot** – When you create a backup, if there are any tags on the source cluster, they are copied to the backup.
- **DeleteSnapshot** or **delete-snapshot** – When you delete a backup, if there are any tags on the backup, they are deleted with the backup.
• **CreateCacheCluster** or **create-cache-cluster** – When you create a cluster and seed it from a backup, any tags on the backup are copies to the new cluster.

• **DeleteCacheCluster** or **delete-cache-cluster** – When you delete a cluster, any tags on the cluster are deleted with the cluster. However, if you make a final backup, the tags are copied to the backup.

• **CreateReplicationGroup** or **create-replication-group** – When you create a replication group and seed it from one or more backups, any tags on the backups are copies to the new replication group.

• **DeleteReplicationGroup** or **delete-replication-group** – When you delete a replication group, any tags on the replication group are deleted with the replication group. However, if you make a final backup, the tags are copied to the backup.
Using the ElastiCache API

This section provides task-oriented descriptions of how to use and implement ElastiCache operations. For a complete description of these operations, see the Amazon ElastiCache API Reference.

Topics
- Using the Query API (p. 478)
- Available Libraries (p. 480)
- Troubleshooting Applications (p. 480)
- Logging Amazon ElastiCache API Calls Using AWS CloudTrail (p. 481)

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).

```text
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.

```text
GET\nelasticache.amazonaws.com
```
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.
Amazon ElastiCache User Guide
Retrieving Errors

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

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.
API Version 2015-02-02
481


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.

```json
{
  "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"
  }
}
```

API Version 2015-02-02
482
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 Tutorials

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 18, 2017

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date Changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for EU (Paris).</td>
<td>ElastiCache added support for the EU (Paris) region. For more information, see: - Supported Regions &amp; Endpoints (p. 60) - Supported Node Types (p. 117)</td>
<td>December 18, 2017</td>
</tr>
<tr>
<td>Support for China (Ningxia) Region</td>
<td>Amazon ElastiCache added support for China (Ningxia) Region. For more information, see Supported Regions &amp; Endpoints (p. 60).</td>
<td>December 11, 2017</td>
</tr>
<tr>
<td>Support for Service Linked Roles</td>
<td>This release of ElastiCache added support for Service Linked Roles (SLR). For more information, see: - Using Service-Linked Roles for ElastiCache (p. 417) - Step 1b: Set Up Your Permissions (New ElastiCache Customers only) (p. 27)</td>
<td>December 7, 2017</td>
</tr>
<tr>
<td>Support for R4 node types</td>
<td>This release of ElastiCache added support R4 node types in all regions supported by ElastiCache. You can purchase R4 node types as On-Demand or as Reserved Cache Nodes. For more information, see: - Supported Node Types (p. 117) - Memcached Node-Type Specific Parameters (p. 360) - Redis Node-Type Specific Parameters (p. 377)</td>
<td>November 20, 2017</td>
</tr>
<tr>
<td>ElastiCache for Redis 3.2.10 and support for onlineresharding</td>
<td>Amazon ElastiCache for Redis adds support for ElastiCache for Redis 3.2.10. ElastiCache for Redis also introduces online cluster resizing to add or remove shards from the cluster while it continues to serve incoming I/O requests. For more information, see the following: - Choosing an Engine: Memcached, Redis (cluster mode disabled), or Redis (cluster mode enabled) (p. 42) - Best Practices: Online Resharding (p. 89)</td>
<td>November 9, 2017</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>• Online Resharding and Shard Rebalancing for ElastiCache for Redis—Redis (cluster mode enabled)</td>
<td>November 2, 2017</td>
<td></td>
</tr>
<tr>
<td>HIPAA compliance</td>
<td>ElastiCache for Redis version 3.2.6 is now certified for HIPAA compliance when encryption is enabled on your cluster. For more information, see HIPAA Compliance for Amazon ElastiCache for Redis (p. 439) and Amazon ElastiCache for Redis Data Encryption (p. 429).</td>
<td>October 25, 2017</td>
</tr>
</tbody>
</table>
| ElastiCache for Redis 3.2.6 and support for encryption | ElastiCache adds support for ElastiCache for Redis 3.2.6, which includes two encryption features:  
• In-transit encryption encrypts your data whenever it is in transit, such as between nodes in a cluster or between a cluster and your application.  
• At-rest encryption encrypts your on-disk data during sync and backup operations.  
For more information, see Amazon ElastiCache for Redis Data Encryption (p. 429) and Choosing an Engine: Memcached, Redis (cluster mode disabled), or Redis (cluster mode enabled) (p. 42) in this *ElastiCache User Guide*. | April 24, 2017 |
| Connection patterns topic | ElastiCache documentation adds a topic covering various patterns for accessing an ElastiCache cluster in an Amazon VPC.  
For more information, see Access Patterns for Accessing an ElastiCache Cluster in an Amazon VPC (p. 393) in the *ElastiCache User Guide*. | April 10, 2017 |
| Support for Memcached 1.4.34 | ElastiCache adds support Memcached version 1.4.34, which incorporates a number of fixes to earlier Memcached versions.  
For more information, see Memcached 1.4.34 Release Notes at Memcached on GitHub. | April 4, 2017 |
| Support for testing Automatic Failover | ElastiCache adds support for testing Automatic Failover on Redis clusters that support replication.  
For more information, see the following:  
• Testing Multi-AZ with Automatic Failover (p. 249) in the *ElastiCache User Guide*.  
• TestFailover in the *ElastiCache API Reference*.  
• test-failover in the *AWS CLI Reference*. | April 4, 2017 |
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date Changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enhanced Redis restore</td>
<td>ElastiCache adds enhanced Redis backup and restore with cluster resizing. This feature supports restoring a backup to a cluster with a different number of shards (for the API and CLI, a different number of node groups) than the cluster used to create the backup, along with different Redis slot configurations. For more information, see Restoring From a Backup with Optional Cluster Resizing (p. 317).</td>
<td>March 15, 2017</td>
</tr>
<tr>
<td>New Redis memory management parameter</td>
<td>ElastiCache adds a new Redis parameter, reserved-memory-percent, which makes managing your reserved memory easier. This parameter is available on all versions of ElastiCache for Redis. For more information, see Managing Reserved Memory (Redis) (p. 79) and New Parameters for Redis 3.2.4 (p. 363).</td>
<td>March 15, 2017</td>
</tr>
<tr>
<td>Support for Memcached 1.4.33</td>
<td>ElastiCache adds support for Memcached version 1.4.33. For more information, see Memcached Version 1.4.33 (p. 47) and Memcached 1.4.33 Added Parameters (p. 353).</td>
<td>December 20, 2016</td>
</tr>
<tr>
<td>Support for EU West (London) Region</td>
<td>ElastiCache adds support for EU (London) Region. Only node types T2 and M4 are currently supported. For more information, see Supported Regions &amp; Endpoints (p. 60) and Supported Node Types (p. 117).</td>
<td>December 13, 2016</td>
</tr>
<tr>
<td>Support for Canada (Montreal) Region</td>
<td>ElastiCache adds support for the Canada (Montreal) Region. Only node type M4 and T2 are currently supported in this region. For more information, see Supported Regions &amp; Endpoints (p. 60) and Supported Node Types (p. 117).</td>
<td>December 8, 2016</td>
</tr>
<tr>
<td>Support for M4 and R3 node types</td>
<td>ElastiCache adds support for R3 and M4 node types in South America (São Paulo) Region and M4 node types in China (Beijing) Region. For more information, see Supported Regions &amp; Endpoints (p. 60) and Supported Node Types (p. 117).</td>
<td>November 1, 2016</td>
</tr>
<tr>
<td>US East 2 (Ohio) Region support</td>
<td>ElastiCache adds support for the US East (Ohio) Region (us-east-2) with M4, T2, and R3 node types. For more information, see Supported Regions &amp; Endpoints (p. 60) and Supported Node Types (p. 117).</td>
<td>October 17, 2016</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
<td>----------------</td>
</tr>
</tbody>
</table>
| 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. 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. 41)  
  - ElastiCache Components and Features (p. 11) — note the sections on Nodes, Shards, Clusters, and Replication.  
| M4 node type support         | ElastiCache adds support for the M4 family of node types in most regions supported by ElastiCache. You can purchase M4 node types as On-Demand or as Reserved Cache Nodes. For more information, see Supported Node Types (p. 117), Memcached Node-Type Specific Parameters (p. 360), and Redis Node-Type Specific Parameters (p. 377). | August 3, 2016 |
| Mumbai Region support        | ElastiCache adds support for the Asia Pacific (Mumbai) Region. For more information, see Supported Regions & Endpoints (p. 60).                                                                                       | June 27, 2016  |
| Snapshot export              | ElastiCache adds the ability to export a Redis snapshot so you can access it from outside ElastiCache. For more information, see Exporting a Backup (p. 310) in the Amazon ElastiCache User Guide and CopySnapshot in the Amazon ElastiCache API Reference. | May 26, 2016   |
| Node type scale up           | ElastiCache adds the ability to scale up your Redis node type. For more information, see Scaling (p. 199).                                                                                                   | March 24, 2016 |
| Easy engine upgrade          | ElastiCache adds the ability to easily upgrade your Redis cache engine. For more information, see Upgrading Engine Versions (p. 54).                                                                           | March 22, 2016 |
| Support for R3 node types     | ElastiCache adds support for R3 node types in the China (Beijing) and South America (Sao Paulo) regions. For more information, see Supported Node Types (p. 117).                                                    | March 16, 2016 |
## Change Log

<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date Changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accessing ElastiCache using a Lambda function</td>
<td>Added a tutorial on configuring a Lambda function to access ElastiCache in an Amazon VPC. For more information, see <a href="#">ElastiCache Tutorials</a> (p. 485).</td>
<td>February 12, 2016</td>
</tr>
<tr>
<td>Support for Redis 2.8.24</td>
<td>ElastiCache adds 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 <a href="#">ElastiCache for Redis Version 2.8.24 (Enhanced)</a> (p. 52) and <a href="#">Redis 2.8 Release Notes</a>.</td>
<td>January 20, 2016</td>
</tr>
<tr>
<td>Support for Asia Pacific (Seoul) Region</td>
<td>ElastiCache adds support for the Asia Pacific (Seoul) (<code>ap-northeast-2</code>) Region with t2, m3, and r3 node types.</td>
<td>January 6, 2016</td>
</tr>
<tr>
<td>Amazon ElastiCache console change.</td>
<td>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 <a href="#">ElastiCache for Redis Versions</a> (p. 49).</td>
<td>December 15, 2015</td>
</tr>
<tr>
<td>Support for Redis 2.8.23.</td>
<td>ElastiCache adds 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 <code>close-on-slave-write</code> which, if enabled, disconnects clients who attempt to write to a read-only replica. For more information, see <a href="#">ElastiCache for Redis Version 2.8.23 (Enhanced)</a> (p. 52).</td>
<td>November 13, 2015</td>
</tr>
</tbody>
</table>
| Support for Redis 2.8.22. | ElastiCache adds 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 [ElastiCache for Redis Version 2.8.22 (Enhanced)](#) (p. 52).  

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 |
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date Changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Support for Memcached 1.4.28.</td>
<td>ElastiCache adds support for Memcached version 1.4.24 and Memcached improvements since version 1.4.14. This release adds support for least recently used (LRU) cache management as a background task, choice of <em>jenkins</em> or <em>murmur3</em> as your hashing algorithm, new commands, and miscellaneous bug fixes. For more information, go to Memcached release notes and ElastiCache for Memcached Versions (p. 47) in the ElastiCache User Guide.</td>
<td>August 27, 2015</td>
</tr>
<tr>
<td>Support for Redis 2.8.21.</td>
<td>ElastiCache adds 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.</td>
<td>July 29, 2015</td>
</tr>
<tr>
<td>Support for Memcached Auto Discovery using PHP 5.6.</td>
<td>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. 145).</td>
<td></td>
</tr>
<tr>
<td>New topic: Accessing ElastiCache from outside AWS</td>
<td>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. 441).</td>
<td>July 9, 2015</td>
</tr>
<tr>
<td>Node replacement messages added</td>
<td>ElastiCache adds three messages pertaining to scheduled node replacement. ElastiCache:NodeReplacementScheduled, ElastiCache:NodeReplacementRescheduled, and ElastiCache:NodeReplacementCanceled. For more information and actions you can take when a node is scheduled for replacement, go to ElastiCache's Event Notifications and Amazon SNS (p. 462).</td>
<td>June 11, 2015</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
</tbody>
</table>
| Support for Redis v. 2.8.19. | ElastiCache adds 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 adds support for cost allocation tags.  
For more information, see Monitoring Costs with Cost Allocation Tags (p. 466). | February 9, 2015 |
| Support for EU (Frankfurt) Region | ElastiCache adds support for the EU (Frankfurt) (`eu-central-1`) Region. | January 19, 2015 |
| Multi-AZ with auto failover support for Redis replication groups | ElastiCache adds 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. 240). | October 24, 2014 |
| AWS CloudTrail logging of API calls supported | ElastiCache adds support for using AWS CloudTrail to log all ElastiCache API calls.  
For more information, see Logging Amazon ElastiCache API Calls Using AWS CloudTrail (p. 481). | September 15, 2014 |
| New instance sizes supported | ElastiCache adds support for additional General Purpose (T2) instances.  
For more information, see Parameters and Parameter Groups (p. 337). | September 11, 2014 |
<table>
<thead>
<tr>
<th>Change</th>
<th>Description</th>
<th>Date Changed</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flexible node placement supported for Memcached</td>
<td>ElastiCache adds support for creating Memcached nodes across multiple Availability Zones. For more information, see Step 2: Launch a Cluster (p. 28).</td>
<td>July 23, 2014</td>
</tr>
<tr>
<td>New instance sizes supported</td>
<td>ElastiCache adds support for additional General Purpose (M3) instances and Memory Optimized (R3) instances. For more information, see Parameters and Parameter Groups (p. 337).</td>
<td>July 1, 2014</td>
</tr>
<tr>
<td>PHP auto discovery</td>
<td>Added support for PHP version 5.5 auto discovery. For more information, see Installing the ElastiCache Cluster Client for PHP (p. 138).</td>
<td>May 13, 2014</td>
</tr>
<tr>
<td>Backup and restore for Redis clusters</td>
<td>In this release, ElastiCache allows customers to create snapshots of their Redis clusters, and create new clusters using these snapshots. A backup is a copy of the cluster at a specific moment in time, and consists of cluster metadata and all of the data in the Redis cache. Backups 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 and Restore (Redis) (p. 293).</td>
<td>April 24, 2014</td>
</tr>
<tr>
<td>Redis 2.8.6</td>
<td>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.</td>
<td>March 13, 2014</td>
</tr>
<tr>
<td>Redis cache engine</td>
<td>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.</td>
<td>September 3, 2013</td>
</tr>
<tr>
<td>Change</td>
<td>Description</td>
<td>Date Changed</td>
</tr>
<tr>
<td>--------</td>
<td>-------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Support for default Amazon Virtual Private Cloud (VPC)</td>
<td>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. 388).</td>
<td>January 8, 2013</td>
</tr>
<tr>
<td>PHP support for cache node auto discovery</td>
<td>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.</td>
<td>January 2, 2013</td>
</tr>
<tr>
<td>Support for Amazon Virtual Private Cloud (VPC)</td>
<td>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. 388).</td>
<td>December 20, 2012</td>
</tr>
<tr>
<td>Cache node auto discovery and new cache engine version</td>
<td>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. 337).</td>
<td>November 28, 2012</td>
</tr>
<tr>
<td>New cache node types</td>
<td>This release provides four additional cache node types.</td>
<td>November 13, 2012</td>
</tr>
<tr>
<td>Reserved cache nodes</td>
<td>This release adds support for reserved cache nodes.</td>
<td>April 5, 2012</td>
</tr>
<tr>
<td>New guide</td>
<td>This is the first release of Amazon ElastiCache User Guide.</td>
<td>August 22, 2011</td>
</tr>
</tbody>
</table>
AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the AWS General Reference.