Amazon ElastiCache

User Guide

API Version 2013-06-15

Amazon ElastiCache: User Guide

Table of Contents

What Is Amazon ElastiCache?

Topics

Welcome to the *Amazon ElastiCache User Guide*. ElastiCache is a web service that makes it easy to set up, manage, and scale a distributed in-memory cache environment in the cloud. It provides a high-performance, resizable, 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 automatically. For enhanced security, ElastiCache runs in the Amazon Virtual Private Cloud (Amazon VPC) environment, giving you complete control over network access to your cache cluster. With just a few clicks in the AWS Management Console, you can add resources to your ElastiCache environment, such as additional nodes or read replicas, 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 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 cache node membership.

ElastiCache has multiple features to enhance reliability for critical production deployments, such as automatic detection of and recovery from cache node failures. ElastiCache also works together with other Amazon Web Services (such as Amazon EC2, CloudWatch, and Amazon SNS) to provide a secure, high-performance and managed in-memory caching solution.

Are You a First-Time ElastiCache User?

If you are a first-time user of ElastiCache, we recommend that you begin by reading the following sections:

- **Service Highlights and Pricing** – The product detail page provides a general product overview of ElastiCache, service highlights, and pricing.
- **Getting Started** – The Getting Started (p. 6) 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.

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

- ElastiCache Terminology and Concepts (p. 17)

  This section explains the components of a cache cluster, including cache nodes, cache engines, replication groups, and cache cluster security.
- Creating a Cache Cluster (p. 68)

  This section provides instructions and examples for creating and maintaining cache clusters. You can perform these tasks from either the AWS Management Console or the ElastiCache command line interface (CLI).

If you want to use the ElastiCache CLI, these materials can help you get started:

- Setting Up the ElastiCache Command Line Interface (p. 63)

  This section provides information on downloading the ElastiCache CLI, getting the CLI working on your system, and providing your AWS credentials.
- Amazon ElastiCache Command Line Reference

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

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

- Tools for Amazon Web Services

  Amazon Web Services provides a number of software development kits (SDKs) with support for ElastiCache. You can code against ElastiCache using Java, .NET, PHP, Ruby, and other languages. These SDKs can greatly simplify your application development by formatting your requests to ElastiCache, parsing responses, and providing retry logic and error handling.
- Using the ElastiCache API (p. 140)

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

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

Data Model

The Amazon ElastiCache data model concepts include cache nodes, cache clusters, security configuration, and replication groups. The ElastiCache data model also includes resources for event notification and performance monitoring; these resources complement the core concepts.

Cache Nodes

A cache node is the smallest building block of an ElastiCache deployment. Each node has its own memory, storage and processor resources, and runs a dedicated instance of cache engine software — either Memcached or Redis. ElastiCache provides a number of different cache node configurations for you to choose from, depending on your needs. You can use these cache nodes on an on-demand basis, or take advantage of reserved cache nodes at significant cost savings.

Cache Cluster

A cache cluster is a collection of one or more cache nodes, each of which runs its own instance of supported cache engine software. You can launch a new cache cluster with a single ElastiCache operation (`CreateCacheCluster`), specifying the number of cache nodes you want and the runtime parameters for the cache engine software on all of the nodes. Each node in a cache cluster has the same compute, storage and memory specifications, and they all run the same cache engine software (Memcached or Redis). The ElastiCache API lets you control cluster-wide attributes, such as the number of cache nodes, security settings, version upgrades, and system maintenance windows.

Cache Parameter Groups

Cache parameter groups are an easy way to manage runtime settings for supported cache engine software. Both Memcached and Redis have many parameters to control memory usage, cache eviction policies, item sizes, and more; a cache parameter group is a named collection of Memcached- or Redis-specific parameters that you can apply to a cache cluster, thereby guaranteeing that all of the nodes in that cluster are configured in exactly the same way.

Cache Replication Groups

If you are running a single-node Redis cache cluster, you can create a replication group for that cluster, and then add one or more read replicas. A replication group consists of one primary cache cluster for handling read-write traffic, plus additional cache clusters that act as read-only replicas of the primary. ElastiCache ensures that the read replicas are continually synchronized with the primary and, in the event of a primary cache node failure, rebuilds the primary using data from one of the surviving read replicas. In this way, replication groups allow you to scale your read-only traffic, while guarding against data loss caused by hardware failure.

Security

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

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

Supported Operations

To work with caching infrastructure components, ElastiCache offers a set of operations that you can invoke from the AWS Management Console, the ElastiCache command line interface (CLI), or the ElastiCache application programming interface (API).

Cache Cluster Operations

ElastiCache provides operations to create, modify, and delete cache clusters. You can use the `CreateCacheCluster` operation to launch a new cache cluster, specifying the number and type of cache nodes, the cache engine software to use (Memcached or Redis), and other configuration parameters. Once the cache cluster is launched, you can use the `DescribeCacheClusters` operation to view information about the cluster, including the host names and port numbers of the cache nodes; your application can then connect to the cache nodes and begin using the cache.

While your cache cluster is running, you can use the `ModifyCacheCluster` operation to add or remove cache nodes, modify the cache engine parameters, or change the security settings for all nodes in the cluster. If you decide that you no longer need the cache cluster, you can use the `DeleteCacheCluster` operation to shut down all of the cache nodes and release all of the other resources associated with the cluster.

For more information, see Managing Cache Clusters (p. 68)

For information on migrating your existing Redis cache clusters to ElastiCache, see Migrate Your Redis Cache to ElastiCache (p. 100)

Cache Node Parameter Operations

The supported cache engines in ElastiCache provide several runtime parameters, which you can use to fine-tune your cache cluster's performance. The `CreateCacheParameterGroup` operation lets you create a "template" set of parameters that are compatible with the cache engine that you are using (Memcached or Redis). You can the use the `ModifyCacheCluster` operation to apply the parameters to all of the cache cluster nodes; you can do so immediately, or wait for the next system maintenance window for your cache cluster.

For more information, see Managing Cache Parameter Groups (p. 104)

Cache Replication Group Operations (Redis only)

If you have a single-node cache cluster running the Redis cache engine, you can define that cluster as the primary node in a replication group. The API call `CreateReplicationGroup` creates a new replication group, with your Redis cache cluster as the primary. To add read replicas, use the `CreateCacheCluster` API call to create additional single-node cache clusters, running in your new replication group. From that point on, the read replicas will remain synchronized with the primary; your applications can perform read-only operations on the replicas, and read-write operations on the primary. See Managing Replication Groups (p. 85) for more information on creating and adding Redis cache clusters to a replication group using the CLI or AWS console.

If you no longer need the replication group, you can use `DeleteReplicationGroup` to delete it. This powerful operation removes the cache nodes, cache clusters, and all of the other resources used by the replication group.

For more information, see Managing Replication Groups (p. 85)

Accessing Amazon ElastiCache

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, and other metrics. For more information, go to the AWS Management Console.

You can also use the ElastiCache command line interface (CLI). The CLI makes it easy to perform one-at-a-time operations, such as starting or stopping your cache cluster. You can also invoke ElastiCache CLI commands from a scripting language of your choice, letting you automate repeating tasks. For more information about the CLI, see Setting Up the ElastiCache Command Line Interface (p. 63).

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.

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. 140).

Regions and Endpoints

By default, the AWS SDKs and console for ElastiCache 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 own HTTP requests, the AWS SDKs, and the console. For a current list of supported regions and endpoints, see Regions and Endpoints.

Getting Started

Topics

If you want to migrate your existing Redis cache to ElastiCache, see the topic Migrate Your Redis Cache to ElastiCache (p. 100)

Step 1: Before You Begin

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

To sign up for ElastiCache

1. Go to http://aws.amazon.com, and then click **Sign Up**.
2. Follow the on-screen instructions.

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

Step 2: Create a Cache Cluster

After you have signed up for ElastiCache, you can create a cache cluster.

In this step, you will use the AWS Management Console to create a cache cluster. You can choose either Memcached or Redis as the cache engine software for this cluster.

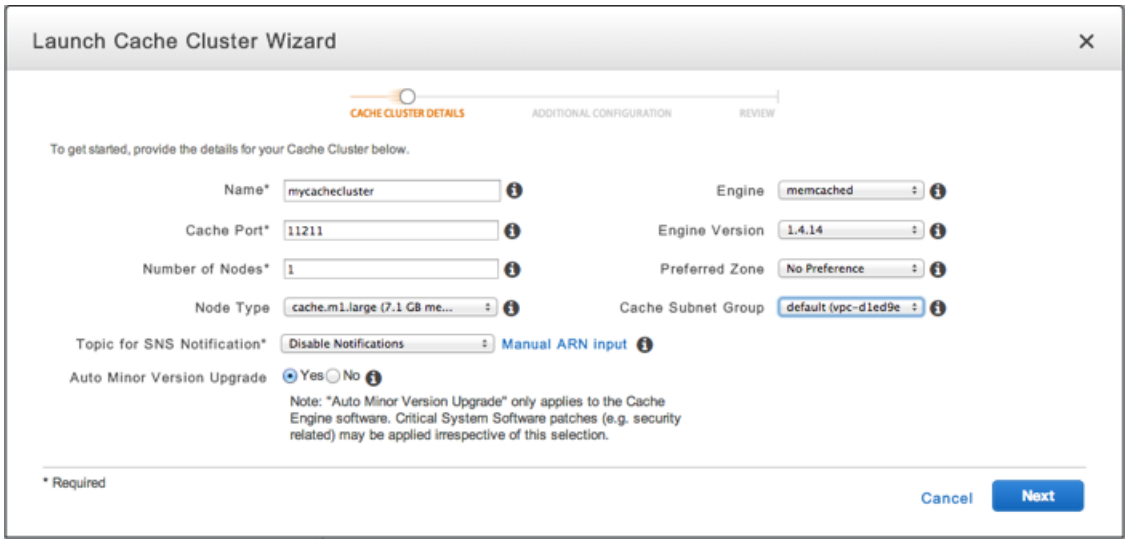
Important
The cache cluster you're about to launch will be live (and not running in a sandbox). You will incur the standard ElastiCache usage fees for the instance until you delete it. The total charges will be minimal (typically less than a dollar) if you complete the exercise described here in one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

In this section, you will use the AWS Management Console to create a cache cluster. You can choose either Memcached or Redis as the cache engine software for this cluster. Redis cache clusters always have one node. Memcached cache clusters can have between 1 and 20 nodes. Multiple nodes cost more than a single node. For more information about ElastiCache usage rates, go to the ElastiCache product page.

If you are a new ElastiCache user, your cache cluster will be launched in an Amazon Virtual Private Cloud environment (Amazon VPC); for more information, see Using ElastiCache with Amazon Virtual Private Cloud (VPC) (p. 113).

To launch a cache 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 ElastiCache Console Dashboard, click **Launch Cache Cluster** to start the Launch Cache Cluster Wizard.



3. On the **Cache Cluster Details** page, in the **Name** text box, enter a name for your cache cluster.
4. In the **Engine** box, click the cache engine software that you want to run, *memcached* or *redis*.
5. In the **Cache Port** text box, you can specify a new port number for your cache cluster, or leave it at its default value:

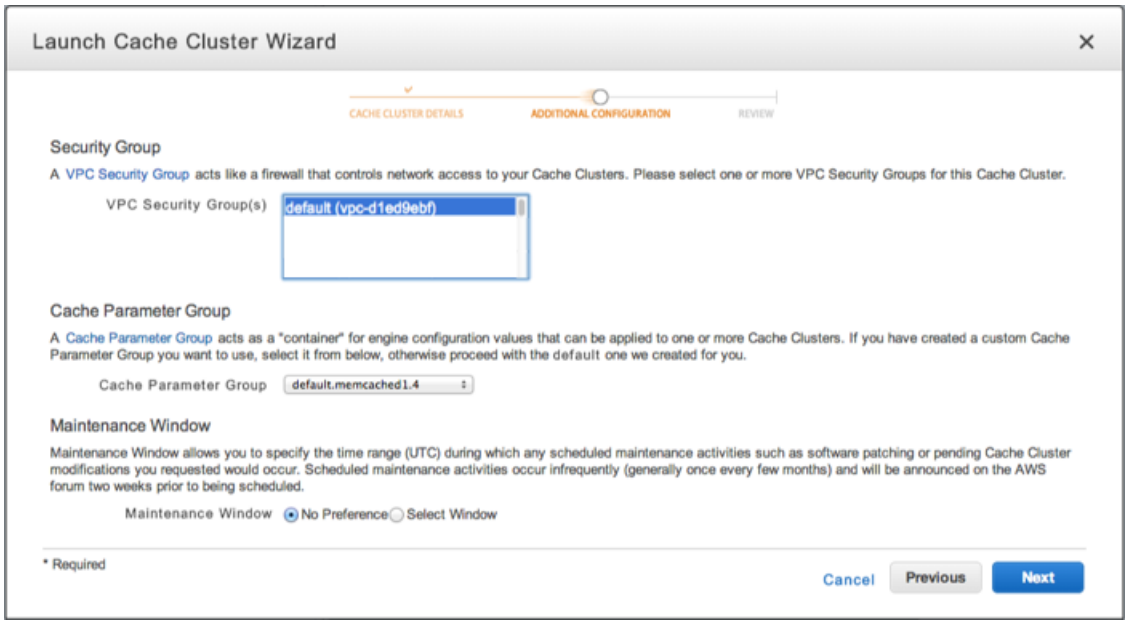
   - For Memcached, the default port number is **11211**.
   - For Redis, the default port number is **6379**.

6. In the **Number of Nodes** text box, enter the number of nodes you want to create. For Redis this must be *1*.
7. In the **Cache Subnet Group** box, click the subnet where you would like to launch the cache cluster:

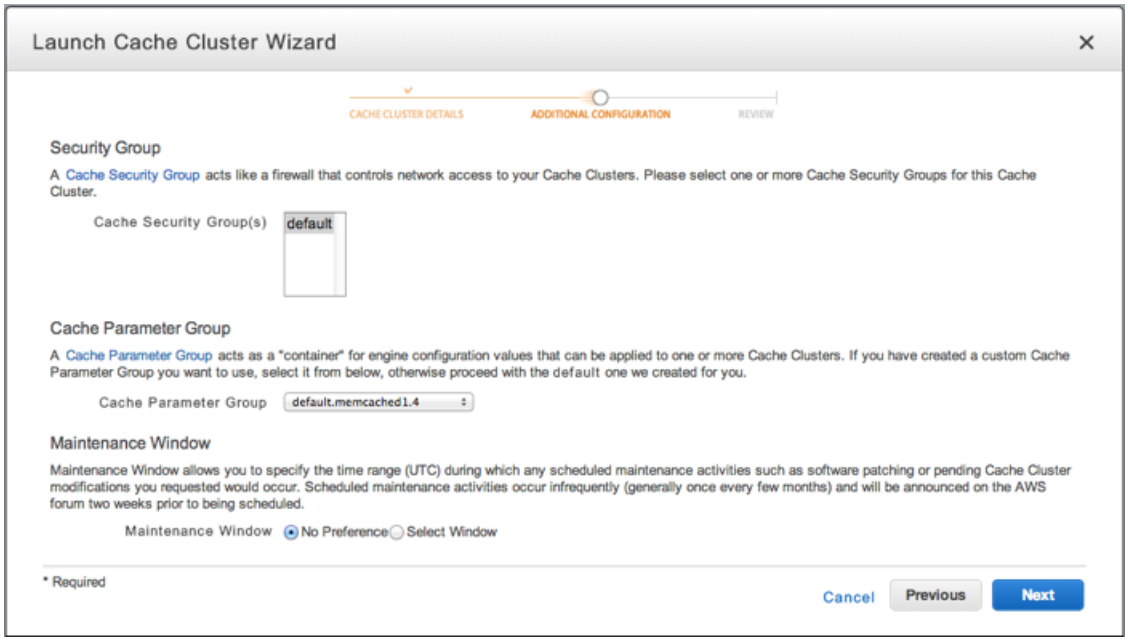
- **VPC** (recommended) – The cache cluster will be launched in an Amazon Virtual Private Cloud environment. If you are a new ElastiCache user, a VPC is automatically created on your behalf, along with a cache subnet group named *default*; you should use the *default* cache subnet group for this exercise.
- **Not in VPC** – The cache cluster will be launched in the AWS public cloud.

8. In the **Topic for SNS notifications** box, click **Disable Notifications**.
9. Leave the rest of the options on the **Cache Cluster Details** page at their default values, and then click **Next**.
10. On the **Additional Configuration** page, you can specify a security group, cache parameter group, and maintenance window for your cache cluster.

    If you are launching your cache cluster in a VPC, the **Security Group** portion of the page will show the VPC security group:

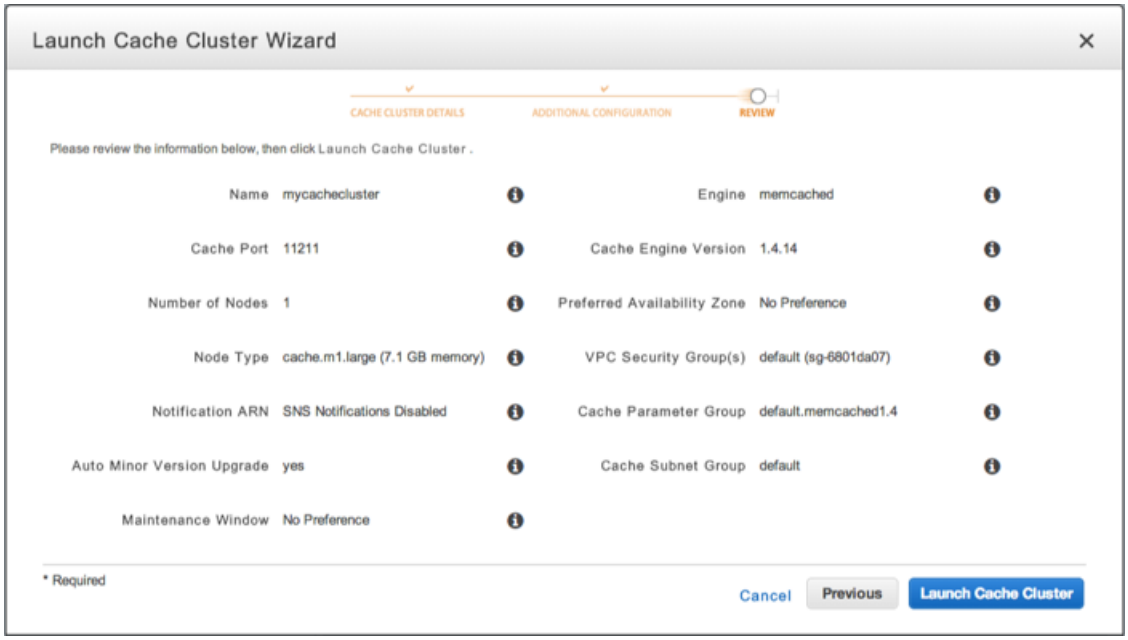


    If you chose to launch your cache cluster outside of a VPC, the **Security Group** portion of the page will show a cache security group instead:



For this exercise, leave all of the values as they are, and then click **Next**.

11. On the **Review** page, review the options for your cache cluster:



12. Review the options for your cache cluster:

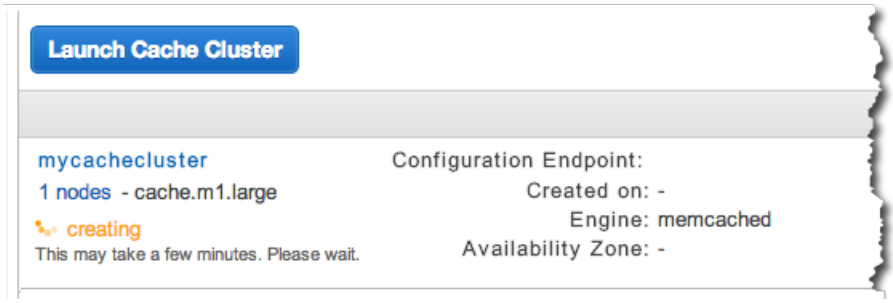
    - If you need to correct any options, click **Previous** to return to previous panels and make corrections.
    - When all your options are entered as you want them, click **Launch Cache Cluster**.

    After you click **Launch Cache Cluster**, a message displays stating that your cache cluster is being created.

The launch process can take a few minutes to complete.

13. Click **Close**.

    When the **Cache Clusters** panel appears, your cache cluster is displayed in the list with the **creating** status until the creation is complete and your cache cluster is ready for use.



Step 3: Authorize Access

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

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

If You Launched Your Cache Cluster Into a VPC

You can connect to your ElastiCache cluster only from an Amazon EC2 that is running in the same VPC security group. You will need to grant network ingress to the cache cluster.

To grant network ingress from a VPC security group to a cache cluster

1. Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.
2. In the navigation list, under **Network & Security**, click **Security Groups**.
3. In the list of security groups, click the security group for your VPC. If you are a new ElastiCache user, this security group will be named *default*.
4. In the **Details** pane, click **Inbound**, and do the following:

   a. In the **Create a new rule** box, choose **Custom TCP rule**.

b. In the **Port range** field, enter the port number for your cache cluster node. This must be the same number that you specified when you launched the cache cluster.
c. In the **Source** field, leave the value at its default (0.0.0.0/0) so that any Amazon EC2 instance that you launch within your VPC can connect to your ElastiCache nodes.
d. Click **Add Rule**, and then click **Apply Rule Changes**.

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

If You Launched Your Cache Cluster in the AWS Public Cloud

In order to allow an Amazon EC2 instance to access your cache cluster, you will need to grant the EC2 security group associated with the instance access to your *cache security group*.

To grant an EC2 security group access to a cache security group

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the navigation list, click **Cache Security Groups**.
3. In the **Cache Security Groups** list, select the check box next to the cache security group for the cache cluster that you are granting access to. For this tutorial, we'll use the cache security group named *default*.
4. On the detail panel, do the following:

   a. If the **AWS Account Id** and **EC2 Security Group Name** fields are blank, click **(use default)**; this will automatically populate these fields.
   b. Otherwise, in the **EC2 Security Group Name** field, choose the name of the EC2 security group that you want to grant access to.

      Tip
      If you want to fill in either of these fields manually, click **(change)**.

   c. When the settings are as you want them, click **Add**.

   It takes approximately one minute for changes to access permissions to take effect. Your cache security group will show a status of **Authorizing** next to the Amazon EC2 security group.

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

Step 4: Connect to a Cache Node

Topics

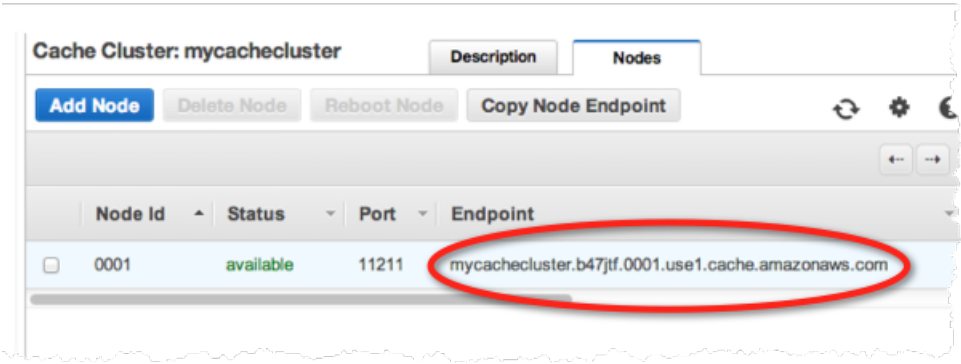
Note
This section assumes that you've created an Amazon EC2 instance and can connect to it. For instructions on how to do this, go to the Amazon Elastic Compute Cloud Getting Started Guide. An Amazon EC2 instance can connect to a cache node only if you have authorized it to do so. For more information, see Step 3: Authorize Access (p. 10) .

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

To determine the endpoint for a cache node

1. On the **Cache Clusters** page of the AWS Management Console, click the name of a cache cluster.
2. On the detail page for the cache cluster, click the **Nodes** tab.
3. On the **Nodes** tab, note the endpoint of a cache node to use in the next step.

   Note
   The endpoint for your cache node isn't available until your cache node is in the **`available`** state.



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

- Connect to a Memcached Cache Node (p. 12)
- Connect to a Redis Cache Node (p. 14)

Connect to a Memcached Cache Node

In the following example, you use the *telnet* utility to connect to a cache node that is running Memcached.

Note
For more information about Memcached and available Memcached commands, go to http://memcached.org.

To connect to a cache 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 EC2 instance, go to the Amazon Elastic Compute Cloud Getting Started Guide.

2. You will need to download and install the *telnet* utility on your EC2 instance. At the command prompt of your EC2 instance, issue the following command. At the confirmation prompt, enter *y*.

```
sudo yum install telnet

Loaded plugins: priorities, security, update-motd, upgrade-helper
Setting up Install Process
Resolving Dependencies
--> Running transaction check

...(output omitted)...

Total download size: 63 k
Installed size: 109 k
Is this ok [y/N]: y
Downloading Packages:
telnet-0.17-47.7.amzn1.x86_64.rpm                          |  63 kB     00:00


...(output omitted)...

Complete!
```

3. At the command prompt of your EC2 instance, enter the following command, substituting the endpoint of your cache node for the one shown in this example.

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

You will see output similar to the following.

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

You are now connected to a cache node, and you can run Memcached commands. For 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 cache miss
END
>
```

Connect to a Redis Cache Node

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

> **Note**
> For more information about Redis and available Redis commands, go to http://redis.io/commands.

To connect to a cache node 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 EC2 instance, go to the Amazon Elastic Compute Cloud 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, issue the following command. At the confirmation prompt, enter *y*.

```
sudo yum install gcc

Loaded plugins: priorities, security, update-motd, upgrade-helper
Setting up Install Process
Resolving Dependencies
--> Running transaction check

...(output omitted)...

Total download size: 27 M
Installed size: 53 M
Is this ok [y/N]: y
Downloading Packages:
(1/11): binutils-2.22.52.0.1-10.36.amzn1.x86_64.rpm      | 5.2 MB     00:00

(2/11): cpp46-4.6.3-2.67.amzn1.x86_64.rpm                | 4.8 MB     00:00

(3/11): gcc-4.6.3-3.10.amzn1.noarch.rpm                  | 2.8 kB     00:00


...(output omitted)...

Complete!
```

3. Now you will need to download and compile the *redis-cli* utility. This utility is included in the Redis software distribution. At the command prompt of your EC2 instance, open a command prompt and enter the following commands:

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

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

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

You will see a Redis command prompt similar to the following.

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

You are now connected to the cache node and can run Redis commands. For example:

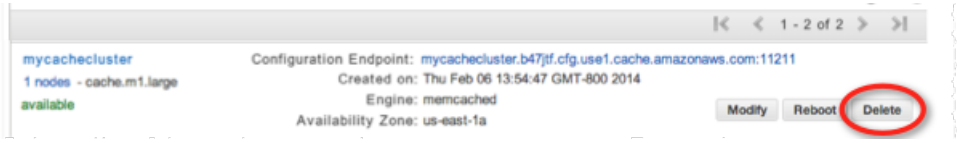
```
set a "hello"     // Set key "a" with a string value and no expiration
OK
get a              // Get value for key "a"
"hello"
get b              // Get value for key "b" results in cache miss
(nil)
quit               // Exit from redis-cli
```

Step 5: Delete Your Cache Cluster

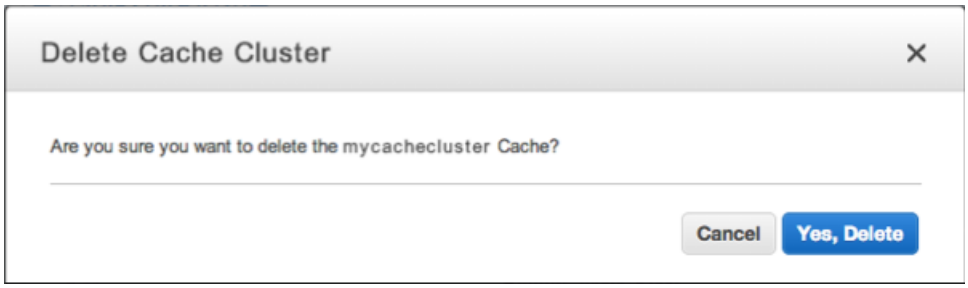
As soon as your cache cluster becomes available, you're billed for each hour or partial hour that you keep the cache cluster running, even if the cache cluster is idle. When you no longer need the cache cluster, you can delete it.

To delete your cache cluster

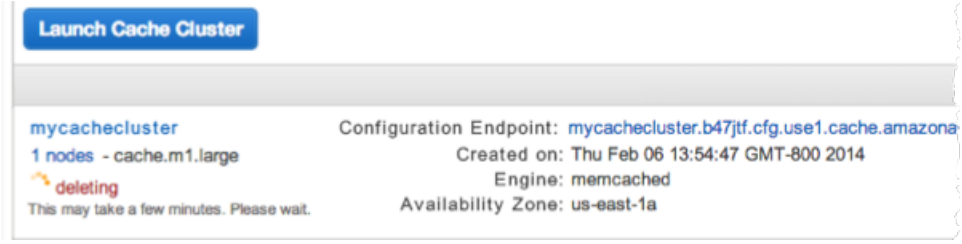
1. In the AWS Management Console, locate the cache cluster in your list of cache clusters on the **Cache Clusters** page.
2. Click the **Delete** link next to the cache cluster.



3. In the **Delete Cache Cluster** dialog box, click **Yes, Delete**.



ElastiCache begins deleting the cache cluster. The status changes to `deleting` in the **Cache Clusters** list.



As soon as the cache cluster status changes to `deleted`, you stop incurring charges for that cache cluster.

Congratulations! You successfully launched, authorized access to, connected to, and deleted a cache 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.

- ElastiCache Terminology and Concepts (p. 17)
- Setting Up the ElastiCache Command Line Interface (p. 63)
- Managing ElastiCache (p. 67)

ElastiCache Terminology and Concepts

Topics

This chapter introduces you to ElastiCache terminology and concepts. Many of the concepts introduced in this chapter are explored in greater depth in later chapters.

Cache Clusters

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

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

Every cache cluster must have a *cache cluster identifier*. The cache cluster identifier is a customer-supplied "name" for the cache cluster. This identifier specifies a particular cache cluster when interacting with the

ElastiCache API and commands. The cache cluster identifier must be unique for that customer in an AWS region.

The following are descriptions of supported cache engine software for ElastiCache.

Memcached

Memcached is a distributed caching solution that uses using client-side hashing to distribute data evenly across one or more cache nodes. You can scale up a Memcached cluster by adding more cache nodes. For more information go to http://memcached.org.

ElastiCache supports the following versions of Memcached:

Memcached 1.4.5

Memcached 1.4.5 was the first release supported by ElastiCache.

Memcached 1.4.14

Memcached 1.4.14 adds several bug fixes and new features, including:

- **Ability to rebalance and fine-tune slab memory**—Use the `slab_automove` and `slab_reassign` parameters to change the way that Memcached manages slab memory.
- **New touch commands and counters.**

Redis

Redis is a key-value store that supports abstract data types and optional data durability. A Redis cache cluster consists of a single cache node; you can scale up by using a larger cache node. For more information, go to http://redis.io.

ElastiCache supports the following versions of Redis:

Redis 2.6.13

Redis 2.6.13 was the first release supported by ElastiCache.

Redis 2.8.6

Redis 2.8.6 adds several bug fixes and new features, including:

- **Partial resynchronization (psync)**—In a replication group, the primary cache node maintains a *backlog*, which is a buffer that holds data that has not yet been sent to the read replicas. If a replica is disconnected from the primary node, then write requests will continue to accumulate in the primary's backlog. When a replica reestablishes contact with the primary, it can request a psync to obtain only the portion of backlog data that the replica missed while it was disconnected. In general, a psync is much faster than a full sync, so a replica can quickly catch up with the primary. You can control the size of the backlog buffer. You can also control the time-to-live attribute of the backlog. If the replica is unavailable for an extended period of time, exceeding the backlog time-to-live, then the backlog is discarded and the replica must perform a full sync when it reconnects.
- **Primary node only allows writes if there are enough available replicas**—In normal circumstances, each read replica pings the primary once every second. By default, if the primary does not receive a ping from a replica, then it will still continue accepting writes from clients. However, this could lead to write loss as data cannot be sent to the replicas. You can now change this behavior, so that the primary can stop accepting writes if the number of available replicas drops below a user-specified threshold. Redis 2.8.6 lets you specify the minimum number of replicas that must be available. You can also

specify the number of seconds within which the primary must receive pings from the replica, or else stop accepting writes.

- **Event notifications with Redis publish-and-subscribe**— Redis clients can subscribe to server-side events of interest, for example, when an item is added to the cache. When an event of interest occurs, Redis publishes notifications to clients that are subscribed to these events. You can specify the types of events that Redis publishes. For more information, go to http://redis.io/topics/notifications.

Cache Nodes

Topics

A *cache node* is the smallest building block of an ElastiCache deployment. It is a fixed-size chunk of secure, network-attached RAM. Each cache node runs an instance of either Memcached or Redis, depending on what was selected when the cache cluster was created. Each cache node has its own DNS name and port. Multiple types of cache nodes are supported, each with varying amounts of associated memory.

> **Note**
> For a complete list of cache node types and specifications, go to Amazon ElastiCache Product Features and Details and either Cache Node Type-Specific Parameters for Memcached or Cache Node Type-Specific Parameters for Redis.

Cache Node Considerations for Memcached

With a cache cluster running Memcached, you can

- Easily select and change the amount of memory and compute capacity available in your cache cluster.
- Easily select the Availability Zone for your cache cluster nodes.

Node Size Considerations

The total memory capacity of your cache cluster is calculated by multiplying the number of cache nodes in the cluster by the capacity of each node. The capacity of each cache node is based on the cache node type.

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

In a scenario where you want to have 20GB of cache memory, you can set it up in one of the following ways:

- Use 15 `cache.m1.small` cache nodes with 1.3 GB of memory each = 19.5 GB
- Use 3 `cache.m1.large` cache nodes with 7.1 GB of memory each = 21.3 GB
- Use 3 `cache.c1.xlarge` cache nodes with 6.6 GB of memory each = 19.8 GB

These options provide you with similar memory capacity, but different computational capacity for your cache cluster.

> **Note**
> For cache clusters running Memcached, some of the available memory on each cache node is used for connection overhead. For more information, see the section Understanding and Tuning Memcached Connection Overhead (p. 29).

If you're unsure about how much capacity you need, we recommend starting with one `cache.m1.small` cache node type and monitoring the memory usage, CPU utilization and Cache Hit Rate with the ElastiCache metrics that are published to CloudWatch.

If your cache cluster does not have the desired hit rate, you can easily add more nodes, thereby increasing the total available memory in your cache cluster. You will need to obtain an updated endpoint list from the ElastiCache CLI, API or AWS Management Console, and configure your clients to use the additional node(s).

If your cache cluster turns out to be bound by CPU but has sufficient hit rate, then try setting up a new cluster with a different cache node type.

ElastiCache supports adding or removing cache nodes from an existing cache cluster using the AWS Management Console, the API, and the command line tools, allowing you to increase both memory and compute capacity of the cluster at any time.

> **Note**
> ElastiCache does not currently support dynamically changing the cache node type for a cache cluster after it has been created. If you wish to change the Node Type of a cache cluster, you will need to set up a new cache cluster with the desired Node Type, and migrate your application to that cache cluster.

You can scale a Memcached cache cluster by adding or deleting cache nodes. For more information, see Adding or Removing Cache Nodes (p. 70).

Availability Zone Considerations

A Memcached cache cluster can have up to 20 nodes.

For information about how to create or modify a Memcached cache cluster, see Managing Cache Clusters. For more information about using the CLI and API, see the following topics:

- Using the CLI to create a new cache cluster - elasticache-create-cache-cluster
- Using the CLI to modify an existing cache cluster - elasticache-modify-cache-cluster
- Using the API to create a new cache cluster - CreateCacheCluster
- Using the API to modify an exisitng cache cluster - ModifyCacheCluster

For more information about regions and Availability Zones, see Regions and Availability Zones (p. 46).

Cache Node Considerations for Redis

Topics

> **Note**
> At this time, ElastiCache supports single-node Redis cache clusters.

To determine the appropriate size for your cache node, we recommend that you estimate the total amount of memory that you will need for your Redis cache, and choose a cache node type that has enough

memory for your requirements. For example, if you estimate that the total size of all your items will be 10 GB, then you can use a *cache.m1.xlarge* node with 14.6 GB of memory.

You should also consider which cache node type has the best processor characteristics for your needs. Redis is single-threaded, so multiple processor cores will not provide improved performance. However, choosing a cache node type with a faster processor speed can significantly improve throughput. For example, compare these two cache node types:

- *cache.m1.xlarge* — 14 GB of memory; 4 virtual cores with 2 ECUs each
- *cache.m2.xlarge* — 16.7 GB of memory; 2 virtual cores with 3.25 ECUs each

ECU stands for "EC2 Compute Unit", a measure of relative processor capacity for compute resources in the AWS cloud.

While these cache node types have similar memory characteristics, the per-core processing speed of a *cache.m2.xlarge* is about 60% faster than that of the *cache.m1.xlarge*.

While your cache cluster is running, you can monitor the memory usage, processor utilization, CacheHits and CacheMisses metrics that are published to CloudWatch. If your cache cluster does not have the desired hit rate or you notice that keys are being evicted too often, you can choose a different cache node size with larger CPU and memory specifications. You will need to obtain an updated endpoint list from the ElastiCache CLI, API or AWS Management Console, and configure your clients to use this new node.

Reserved Cache Nodes

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

Reserved cache nodes are available in three varieties — Heavy Utilization, Medium Utilization, and Light Utilization — that enable you to optimize your ElastiCache costs based on your expected utilization.

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

Reserved Cache Node Offerings

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

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

Light Utilization reserved cache nodes are ideal for periodic workloads that run only a couple of hours a day or a few days per week. Using Light Utilization reserved cache nodes, you pay a one-time fee followed by a discounted hourly usage fee when your cache node is running. You can start saving when your

cache node is running more than 17 percent of the reserved cache node term, and you can save up to 56 percent off of the On-Demand rates over the entire term of your reserved cache node.

Remember that discounted usage fees for reserved cache node purchases are tied to cache node type. If you shut down a running cache node on which you have been getting a discounted rate as a result of a reserved cache node purchase, and the term of the reserved cache node has not yet expired, you will continue to get the discounted rate if you launch another cache node with the same specifications during the term.

The following table summarizes the differences between the reserved cache nodes offering types.

Reserved Cache Node Offerings

Offering	Upfront Cost	Usage Fee	Advantage
Heavy Utilization	Highest	Lowest hourly fee. Applied to the whole term whether or not you're using the reserved cache node.	Lowest overall cost if you plan to utilize your reserved cache nodes more than 79 percent of a 3-year term.
Medium Utilization	Average	Hourly usage fee charged for each hour you use the cache node.	Suitable for elastic workloads or when you expect moderate usage, more than 40 percent of a 3-year term.
Light Utilization	Lowest	Hourly usage fee charged. Highest fees of all the offering types, but they apply only when you're using the reserved cache node.	Highest overall cost if you plan to run all of the time, however lowest overall cost if you anticipate you will use your reserved cache nodes infrequently, more than about 15 percent of a 3-year term.

For more information on working with reserved cache nodes, go to Managing Reserved Cache Nodes (p. 75).

Replication Groups and Read Replicas

Topics

By default, cache clusters are standalone entities without any redundant data protection services. If your cache cluster is running Redis, however, you can create a *replication group* to enhance scalability and guard against data loss.

A replication group is a collection of single-node cache clusters, with one read-write primary cluster and up to five read-only clusters, which are called read replicas. Each replica maintains a copy of the data from the primary cache cluster, and uses asynchronous replication mechanisms to keep itself synchronized with the primary. Applications can read from any cluster in the replication group.

Note
At this time, replication groups are supported only for cache clusters running Redis.

You can use replication groups to scale your ElastiCache solution to handle applications that are highly read-intensive or to support large numbers of clients that simultaneously read from the same cache.

All of the nodes in a replication group must reside in the same Region; however, you can provision read replicas in multiple Availability Zones within that Region. When you add a read replica to a replication group, all of the data from the primary cache cluster is copied to the read replica. From that point, whenever data is written to the primary, the changes are immediately propagated to the read replicas. Your applications can connect to the read replica and access data in the cache (although they cannot write any data to the replica).

You can change the roles of the cache nodes within the replication group, where 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 with heavy write activity, you can choose the node that has the lowest network latency or is "closest" to your application. For more information, see Promoting a Read Replica to the Primary Role (p. 87)

Replication groups can guard against potential data loss if a primary cache cluster is unexpectedly terminated. After ElastiCache provisions a new primary node, the Redis cache engine on that node immediately synchronizes itself with one of the surviving read replicas in the replication group. When the synchronization is complete, the result is a warm Redis cache. The primary will then begin accepting write requests, and automatically propagating those writes to the read replicas.

Tip
For maximum protection against data loss, you can enable AOF for the primary cache cluster, in addition to creating read replicas. For more information on AOF, see Append-Only Files (AOF) (p. 40).
Note that AOF is not supported for cache nodes of type *cache.t1.micro*.

Creating a Replication Group

The process for creating a replication group is as follows:

1. Create a cache cluster to act as the primary for the replication group.
2. (Optional) When you create the primary cache cluster, you can seed the cache with data from a Redis RDB snapshot file. For more information, see Creating a Cache Cluster (p. 68) and Migrate Your Redis Cache to ElastiCache (p. 100).
3. Create a replication group, specifying the cache cluster that you just created as the primary cluster.
4. Create a cache cluster to act as a read replica. This cluster will be added to the replication group, and will synchronize itself with the primary.
5. (optional) To add more read replicas to the replication group, repeat the previous step. A replication group can have as many as five read replicas.

For detailed procedures on each step, see Managing Replication Groups (p. 85).

Primary Endpoint

An application can connect to any node in a replication group, provided that it has the DNS endpoint and port number for that node. Read-only applications can connect to any node in the replication group — but all write activity must take place at the primary node.

Every replication group has a *primary endpoint*, which is a DNS name that always resolves to the primary node in the replication group. The primary endpoint is immune to changes to your replication group, such as promoting a read replica to the primary role. Even if you make changes to the replication group, such

as promoting a read replica to become the new primary node, the primary endpoint always points to the primary node for the replication group

For read-only activity, applications can connect to any node in the replication group. However, for write activity, we recommend that your applications connect to the primary endpoint instead of connecting directly to the primary node.

Backup and Restore for Cache Clusters

Topics

In ElastiCache, clusters running Redis can use snapshots for backing up and restoring data. A *snapshot* is a backup copy of an entire Redis cluster at a specific moment in time. The snapshot consists of the cluster metadata, along with all of the data in the Redis cache. All snapshots 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 snapshot. ElastiCache lets you manage snapshots using the AWS Management Console, the ElastiCache command line interface (CLI), and the ElastiCache application programming interface (API).

This section provides an overview of working with backup and restore for clusters running Redis.

Constraints

At this time, backup and restore is supported only for clusters running Redis.

Backup and restore is not supported on `cache.t1.micro` cache nodes. All other cache node types are supported.

Costs

ElastiCache allows you to store one snapshot for each active Redis cluster, free of charge. Storage space for additional snapshots is charged at a rate of $0.085/GB per month for all regions. There are no data transfer fees for creating a snapshot, or for restoring data from a snapshot to a Redis cluster.

Automatic Snapshots

For any Redis cluster, you can enable *automatic* snapshots, where ElastiCache creates a snapshot of the cluster on a daily basis. Automatic snapshots can help guard against data loss: In the event of a node failure, you can create a new cluster, restoring all of your data from the most recent snapshot. The result is a warm-started Redis cluster, pre-loaded with your data and ready for use.

When you configure an automatic snapshot, you should consider the following settings:

- **Snapshot window**—A period during each day when ElastiCache will begin creating a snapshot. The minimum length for the snapshot window is 60 minutes. You can set the snapshot window for any time

when it's most convenient for you, or for a time of day that avoids doing snapshots during a particularly high-utilization period.

If you do not specify a backup window, ElastiCache will assign one automatically.

- **Snapshot retention limit**—The number of days the snapshot will be retained in Amazon S3. For example, if you set the retention limit to 5, then a snapshot taken today would be retained for 5 days. When the retention limit expires, the snapshot is automatically deleted.

  The maximum snapshot retention limit is 35 days. The minimum is 0, meaning that automatic snapshots are disabled for the cluster.

For more information, see Managing Backup and Restore (p. 91)

Manual Snapshots

In addition to automatic snapshots, you can create a *manual* snapshot at any time. For example, if an automatic snapshot is nearing its snapshot retention limit, you can make a copy of that snapshot and keep the copy indefinitely, until you decide to delete it.

Manual snapshots are also useful for archiving purposes. For example, suppose that you've developed a set of baseline data for testing purposes; you can create a manual snapshot 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 snapshot. When the cluster is ready, you can test your applications against the baseline data again—and repeat this process as often as needed.

You can create a manual snapshot in one of the following ways:

- Create a snapshot of a cluster. This is in addition to any automatic snapshots you have enabled on the cluster.
- Make a copy of an existing snapshot. It does not matter whether the source snapshot was created automatically or manually.
- Take a final snapshot immediately before deleting a cluster or replication group.

There is a limit in place on the rate of manual snapshot creation: During any contiguous 24-hour period, you can create no more than 20 manual snapshots per cluster.

Manual snapshots do not have retention limits, and ElastiCache does not automatically delete them. Even if you delete a cluster, any manual snapshots from that cluster will be retained. If you no longer want to keep a manual snapshot, you must explicitly delete it yourself.

For more information, see Creating a Manual Snapshot (p. 96)

Performance Impact of Snapshots

Snapshots are created using Redis' native BGSAVE command: The Redis process on the cache node spawns a child process to write all the data from the cache to a Redis RDB file. It can take up to ten seconds to spawn the child process, and during this time the parent process is unable to accept incoming application requests. After the child process is running independently, the parent process resumes normal operations. The child process exits when the snapshot operation is complete.

While the snapshot is being written, additional cache node memory is used for new writes. If this additional memory usage exceeds the available memory on the node, processing can become slow due to excessive paging.

The following are guidelines for improving snapshotting performance.

- Set the *reserved-memory* parameter—To mitigate excessive paging, we recommend that you set the *reserved-memory* parameter. This parameter prevents Redis from consuming all of the available memory on the node, and can help reduce the amount of paging. You might also see performance improvements by simply using a larger node. For more information about the *reserved-memory* parameter and node memory sizes, see Parameters for Redis (p. 31).
- Create snapshots on a read replica—If you are running Redis in a replication group, you can take a snapshot from the primary node or one of the read replicas. Because of the system resources required during a BGSAVE, we recommend that you create snapshots on one of the read replicas, rather than the master. While the snapshot is being created on 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 snapshot, ElastiCache will always take the snapshot at the primary node. This ensures that you capture the very latest Redis data, before the replication group is deleted.

Restoring From a Snapshot

You can restore the data from a snapshot into a new cluster at any time. By default, the new cluster will have the same configuration that the source cluster did when the snapshot was created; however, you can override some of the parameters, such as node size.

During the restore operation, ElastiCache creates the new cluster, and then populates the Redis cache with data from the snapshot file. When this process is complete, the Redis cache is warmed up and the cluster is ready to accept requests.

For more information, see Restoring a Snapshot To a New Cache Cluster (p. 99).

Deleting Snapshots

An automatic snapshot is automatically deleted when its retention limit expires. If you delete a cluster, all of its automatic snapshots are also deleted. If you delete a replication group, all of the automatic snapshots from the clusters in that group are also deleted.

> **Note**
> When you delete a cluster or a replication group, you have the option of taking a final snapshot before the deletion begins. In the case of a replication group, the snapshot is taken at the primary node.

ElastiCache provides a deletion API that lets you delete a snapshot at any time, regardless of whether the snapshot was created automatically or manually. (Since manual snapshots do not have a retention limit, manual deletion is the only way to remove them.)

For more information, see Deleting Snapshots (p. 102).

Cache Parameter Groups

Topics

With ElastiCache, you can use *cache parameter groups* to control the runtime parameters of your cache nodes. A cache parameter group represents a combination of specific values for each parameter that is passed to the cache engine software during startup. These values determine how the cache engine

processes on each cache node will behave at runtime. The parameter values on a specific cache parameter group apply to all cache nodes that are associated with the group, regardless of which cache cluster they belong to.

For a list of supported parameters, their default values, and which ones can be modified, see `DescribeEngineDefaultParameters` in the Amazon ElastiCache API Reference.

To manage parameter groups, you can use the following API actions:

- **`CreateCacheParameterGroup`**—Creates a cache parameter group.
- **`DescribeCacheParameterGroups`**—Returns information about cache parameter groups associated with your AWS account.
- **`DescribeCacheParameters`**—Returns information about parameters that are part of a cache parameter group.
- **`ModifyCacheParameterGroup`**—Updates the parameters in a cache parameter group.
- **`DeleteCacheParameter`**—Deletes a named cache parameter group.
- **`ResetCacheParameterGroup`**—Resets individual parameters or all parameters in a parameter group to their default values.

You can change the parameter group associated with a cache cluster at any time (using the `ModifyCacheCluster` action). The changes will not be applied to the running cache cluster until each cache node in the cache cluster is rebooted. To reboot one or more cache nodes in a cache cluster, you can call the `RebootCacheCluster` action.

For more information on working with cache parameter groups, see Managing Cache Parameter Groups (p. 104).

Parameters for Memcached

Topics

If you do not specify a cache parameter group for your Memcached cache cluster, then a default cache parameter group (`default.memcached1.4`) will be used. You cannot change the values of any parameters in the default cache parameter group; however, you can always create a custom cache parameter group and assign it to your cache cluster at any time.

Memcached 1.4.5 Parameters

The following table shows the Memcached 1.4.5 parameters that ElastiCache supports.

Name	Default	Type	Modifiable	Description
backlog_queue_limit	1024	integer	No	The backlog queue limit.
binding_protocol	auto	string	No	The binding protocol.
cas_disabled	0 (false)	Boolean	Yes	If 1 (true), check and set (CAS) operations will be disabled, and items stored will consume 8 bytes less than with CAS enabled.

Name	Default	Type	Modifiable	Description
chunk_size	48	integer	Yes	The minimum amount, in bytes, of space to allocate for the smallest item's key, value, and flags, in bytes.
chunk_size_growth_factor	1.25	float	Yes	The growth factor that controls the size of each successive memcached chunk; each chunk will be chunk_size_growth_factor times larger than the previous chunk.
error_on_memory_exhausted	0 (false)	Boolean	Yes	If 1 (true), when there is no more memory to store items, memcached will return an error rather than evicting items.
large_memory_pages	0 (false)	Boolean	No	If 1 (true), ElastiCache will try to use large memory pages.
lock_down_paged_memory	0 (false)	Boolean	No	If 1 (true), ElastiCache will lock down all paged memory.
max_item_size	1048576	integer	Yes	The size, in bytes, of the largest item that can be stored in the cache.
max_simultaneous_connections	65000	integer	No	The maximum number of simultaneous connections.
maximize_core_file_limit	0 (false)	Boolean	No	If 1 (true), ElastiCache will maximize the core file limit.
memcached_connections_overhead	100	integer	Yes	The amount of memory to be reserved for memcached connections and other miscellaneous overhead. For information about this parameter, see Understanding and Tuning Memcached Connection Overhead (p. 29).
requests_per_event	20	integer	No	The maximum number of requests per event for a given connection. This limit is required to prevent resource starvation.

Memcached 1.4.14 Parameters

For Memcached 1.4.14, the following additional parameters are supported.

Name	Default	Type	Modifiable	Description
config_max	16	integer	No	The maximum number of ElastiCache configuration entries.

Name	Default	Type	Modifiable	Description
config_size_max	65536	integer	No	The maximum size of the configuration entries, in bytes.
hashpower_init	16	integer	No	The initial size of the ElastiCache hash table, expressed in powers of two. The default is 2^16, or 65536 keys.
maxconns_fast	0 (false)	Boolean	Yes	Changes the way in which new connections requests are handled when the maximum connection limit is reached. If this parameter is set to 0 (zero), new connections are added to the backlog queue and will wait until other connections are closed. If the parameter is set to 1, ElastiCache sends an error to the client and immediately closes the connection.
slab_automove	0	integer	Yes	Adjust the slab automove algorithm: If this parameter is set to 0 (zero), the automove algorithm is disabled. If it is set to 1, ElastiCache takes a slow, conservative approach to automatically moving slabs. If it is set to 2, ElastiCache aggressively moves slabs whenever there is a cache eviction. (This mode is not recommended except for testing purposes.)
slab_reassign	0 (false)	Boolean	Yes	Enable or disable slab reassignment. If this parameter is set to 1, you can use the "slabs reassign" command to manually reassign memory.

Understanding and Tuning Memcached Connection Overhead

On each cache node, the memory made available for storing cache items is the total available memory on that cache 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 cache 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 cache 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 cache items and provide a larger buffer for connection overhead, while decreasing the value of the `memcached_connections_overhead` parameter will give you more memory to store cache 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` cache node type, the value for `memcached_connections_overhead` is determined as follows:

- If your cache cluster is using the default cache parameter group, ElastiCache will set the value for `memcached_connections_overhead` to 13MB.
- If your cache cluster is using a cache parameter group that you have created yourself, you can set the value for `memcached_connections_overhead` to a value of your choice.

Cache Node Type-Specific Parameters for Memcached

Although most parameters have a single value, some parameters have different values depending on the cache node type used. The following table shows the default values for the `max_cache_memory` and `num_threads` parameters for each cache node type.

Note
The `max_cache_memory` and `num_threads` parameters cannot be modified.

Cache Node Type	max_cache_memory	num_threads
cache.t1.micro	213	1
cache.m1.small	1300	1
cache.m1.medium	3350	1
cache.m1.large	7100	2
cache.m1.xlarge	14600	4
cache.m2.xlarge	16700	2
cache.m2.2xlarge	33800	4
cache.m2.4xlarge	68000	8
cache.m3.medium	2850	1
cache.m3.large	6200	2
cache.m3.xlarge	13600	4
cache.m3.2xlarge	28600	8
cache.c1.xlarge	6600	8
cache.r3.large	13800	2
cache.r3.xlarge	29100	4
cache.r3.2xlarge	59600	8
cache.r3.4xlarge	120600	16
cache.r3.8xlarge	242600	32

Parameters for Redis

Topics

If you do not specify a cache parameter group for your Redis cache cluster, then a default cache parameter group will be used (either `default.redis2.6` or `default.redis2.8`). You cannot change the values of any parameters in the default cache parameter group; however, you can always create a custom cache parameter group and assign it to your cache cluster at any time.

Redis 2.6.13 Parameters

The following table shows the Redis 2.6.13 parameters that ElastiCache supports.

Name	Default	Type	Modifiable	Description
activerehashing	yes	string	No	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. You can specify a different value for *activerehashing* when you create a cache cluster, but cannot change it once the cache cluster is up and running.
appendonly	no	string	Yes	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 cache node failures. The default value is *no*, meaning AOF is turned off. Set this parameter to *yes* to enable AOF. For more information, see Append-Only Files (AOF) (p. 40). Note that AOF is not supported for *cache.t1.micro* nodes. For cache nodes of this type, the *appendonly* parameter value is ignored.

Name	Default	Type	Modifiable	Description
appendfsync	everysec	string	Yes	Controls how often the AOF output buffer is written to disk: • *no* — the buffer is flushed to disk on an as-needed basis. • *everysec* — the buffer is flushed once per second. This is the default. • *always* — the buffer is flushed every time that data in the cache is modified.
client-output-buffer-limit-normal-hard-limit	0	integer	Yes	If a client's output buffer reaches the specified number of bytes, the client will be disconnected. The default is zero (no hard limit).
client-output-buffer-limit-normal-soft-limit	0	integer	Yes	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	0	integer	Yes	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	33554432	integer	Yes	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	8388608	integer	Yes	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	60	integer	Yes	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.

Name	Default	Type	Modifiable	Description
client-output-buffer-limit-slave-hard-limit	268435456	integer	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	67108864	integer	No	For Redis read replicas: If a client's output buffer reaches the specified number of bytes, the client will be disconnected, but only if this condition persists for *client-output-buffer-limit-slave-soft-seconds*.
client-output-buffer-limit-slave-soft-seconds	60	integer	No	For Redis read replicas: If a client's output buffer remains at *client-output-buffer-limit-slave-soft-limit* bytes for longer than this number of seconds, the client will be disconnected.
databases	16	integer	No	The number of databases. You can specify a different value for *databases* when you create a cache cluster, but cannot change it once the cache cluster is up and running.
hash-max-ziplist-entries	512	integer	Yes	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.
hash-max-ziplist-value	64	integer	Yes	Determines the amount of memory used for hashes. Hashes with entries that are smaller than the specified number of bytes are stored using a special encoding that saves space.
list-max-ziplist-entries	512	integer	Yes	Determines the amount of memory used for lists. Lists with fewer than the specified number of entries are stored using a special encoding that saves space.
list-max-ziplist-value	64	integer	Yes	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.

Name	Default	Type	Modifiable	Description
lua-time-limit	5000	integer	No	The maximum execution time for a Lua script, in milliseconds, before ElastiCache takes action to stop the script. If `lua-time-limit` is exceeded, all Redis commands will return an error of the form ____*-BUSY*. Since this state can cause interference with many essential Redis operations, ElastiCache will first issue a *SCRIPT KILL* command. If this is unsuccessful, ElastiCache will forcibly restart Redis.
maxclients	65000	integer	No	The maximum number of clients that can be connected at one time.
maxmemory-policy	volatile-lru	string	Yes	The eviction policy for keys when maximum memory usage is reached. Valid values are: `volatile-lru | allkeys-lru | volatile-random | allkeys-random | volatile-ttl | noeviction`
maxmemory-samples	3	integer	Yes	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.
set-max-intset-entries	512	integer	Yes	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.

Name	Default	Type	Modifiable	Description
reserved-memory	0	integer	Yes	The amount of memory, in bytes, reserved for non-cache usage. By default, the Redis cache will grow until it consumes the cache node's *maxmemory* (see Cache Node Type-Specific Parameters for Redis (p. 39)). If this occurs, then cache node performance will likely suffer due to excessive memory paging. If you use *reserved-memory*, you can set aside some of the available memory for non-Redis purposes to help reduce the amount of paging. For example, suppose you have a *cache.m1.small* node, with a maxmemory of 900MB bytes. If you set *reserved-memory* to 200 MB, then Redis will never consume this memory; instead, this 200 MB is reserved for the operating system and other background processes on the cache node. Consider increasing the reserved-memory parameter if you are using read replicas, append-only files (AOF), or other Redis features that consume more memory. **Note** This parameter is specific to ElastiCache, and is not part of the standard Redis distribution.
slave-allow-chaining	no	string	No	Determines whether a read replica in Redis can have read replicas of its own.
slowlog-log-slower-than	10000	integer	Yes	The maximum execution time, in microseconds, for commands to be logged by the Redis Slow Log feature.
slowlog-max-len	128	integer	Yes	The maximum length of the Redis Slow Log.

Name	Default	Type	Modifiable	Description
tcp-keepalive	0	integer	Yes	If this is set to a nonzero value (N), cache node clients are polled every N seconds to ensure that they are still connected. With the default setting of 0, no such polling will occur.
timeout	0	integer	Yes	If this is set to a nonzero value (N), the cache node will close a connection if the client is idle for N seconds. With the default setting of 0, the cache node does not disconnect idle clients.
zset-max-ziplist-entries	128	integer	Yes	Determines the amount of memory used for sorted sets. Sorted sets with fewer than the specified number of elements are stored using a special encoding that saves space.
zset-max-ziplist-value	64	integer	Yes	Determines the amount of memory used for sorted sets. Sorted sets with entries that are smaller than the specified number of bytes are stored using a special encoding that saves space.

Note
If you do not specify a cache parameter group for your Redis 2.6.13 cache cluster, then a default cache parameter group (`default.redis2.6`) will be used. You cannot change the values of any parameters in the default cache parameter group; however, you can always create a custom cache parameter group and assign it to your cache cluster at any time.

Redis 2.8.6 Parameters

For Redis 2.8.6, the following additional parameters are supported.

Name	Default	Type	Modifiable	Description
min-slaves-max-lag	10	integer	Yes	The number of seconds within which the primary node must receive a ping request from a read replica. If this amount of time passes and the primary does not receive a ping, then the replica is no longer considered available. If the number of available replicas drops below min-slaves-to-write, then the primary will stop accepting writes at that point. If either this parameter or min-slaves-to-write is 0, then the primary node will always accept writes requests, even if no replicas are available.
min-slaves-to-write	0	integer	Yes	The minimum number of read replicas which must be available in order for the primary node to accept writes from clients. If the number of available replicas falls below this number, then the primary node will no longer accept write requests. If either this parameter or min-slaves-max-lag is 0, then the primary node will always accept writes requests, even if no replicas are available.

Name	Default	Type	Modifiable	Description
notify-keyspace-events	(an empty string)	string	Yes	The types of keyspace events that Redis can notify clients of. Each event type is represented by a single letter: • **K** — Keyspace events, published with a prefix of *__keyspace@<db>__* • **E** — Key-event events, published with a prefix of *__keyevent@<db>__* • **g** — Generic, non-specific commands such as *DEL*, *EXPIRE*, *RENAME*, etc. • **$** — String commands • **l** — List commands • **s** — Set commands • **h** — Hash commands • **z** — Sorted set commands • **x** — Expired events (events generated every time a key expires) • **e** — Evicted events (events generated when a key is evicted for maxmemory) • **A** — An alias for *g$lshzxe* You can have any combination of these event types. For example, *AKE* means that Redis can publish notifications of all event types. Do not use any characters other than those listed above; attempts to do so will result in error messages. By default, this parameter is set to an empty string, meaning that keyspace event notification is disabled.

Name	Default	Type	Modifiable	Description
repl-backlog-size	1048576	integer	Yes	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.
repl-backlog-ttl	3600	integer	Yes	The number of seconds that the primary node will retain the backlog buffer. Starting from the time the last replica node disconnected, the data in the backlog will remain intact until repl-backlog-ttl expires. If the replica has not connected to the primary within this time, then the primary will release the backlog buffer. When the replica eventually reconnects, it will have to perform a full sync with the primary. If this parameter is set to 0, then the backlog buffer will never be released.
repl-timeout	60	integer	Yes	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

Cache Node Type-Specific Parameters for Redis

Although most parameters have a single value, some parameters have different values depending on the cache node type used. The following table shows the default values for the `maxmemory` parameter for each cache node type.

Note
The `maxmemory` parameter cannot be modified.

Cache Node Type	maxmemory
cache.t1.micro	142606336
cache.m1.small	943718400
cache.m1.medium	3093299200
cache.m1.large	7025459200
cache.m1.xlarge	14889779200
cache.m2.xlarge	17091788800
cache.m2.2xlarge	35022438400
cache.m2.4xlarge	70883737600
cache.m3.medium	2988441600
cache.m3.large	6501171200
cache.m3.xlarge	14260633600
cache.m3.2xlarge	29989273600
cache.c1.xlarge	6501171200
cache.r3.large	14470348800
cache.r3.xlarge	30513561600
cache.r3.2xlarge	62495129600
cache.r3.4xlarge	126458265600
cache.r3.8xlarge	254384537600

Append-Only Files (AOF)

Note
Append-only files (AOF) are not supported for *cache.t1.micro* nodes. For cache nodes of this type, the *appendonly* parameter value is ignored.

By default, the data in a Redis cache node on ElastiCache resides only in memory, and is not persistent. If a cache 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, the cache node writes all of the commands that change cache data to an append-only file. When a cache node is rebooted, the AOF is "replayed" when the Redis cache engine starts; the result is a warm Redis cache with all of the data intact.

AOF is disabled by default. To enable AOF for a cache cluster running Redis, you must create a cache parameter group with the `appendonly` parameter set to `yes`, and then assign that parameter group to your cache cluster. (You can also modify the `appendfsync` parameter to control how often Redis writes to the AOF file.) For more information, see the following sections:

- Cache Parameter Groups (p. 26)
- Parameters for Redis (p. 31)
- Managing Cache Parameter Groups (p. 104)

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

Tip
For maximum protection against data loss, you can create at least one read replica for your cache cluster, in addition to enabling AOF. For more information, see the following sections:

- Replication Groups and Read Replicas (p. 22)
- Managing Replication Groups (p. 85)

Cache Engine Version Management

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 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.

Since major version upgrades involve some compatibility risk, they will not occur automatically and must be initiated by you. Automatic patching works only for minor upgrades; for example, from Memcached 1.4.x to 1.4.y, or from Redis 2.6.x to 2.6.y. Automatic patching will not work for major upgrades (for example, from Memcached version 1.4.x to 1.5.y, or from Redis 2.6.x to 2.8.y). You can manually initiate major version upgrades to your cache cluster by modifying the cache cluster and specifying a new engine version.

Note
While cache 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 cache cluster on your behalf in the event of a critical security vulnerability in the system or cache software.

Automatic patching will occur during the scheduled maintenance window for your cache clusters, and will be announced on the ElastiCache discussion forum in advance. Automatic version upgrades are enabled by default on new cache clusters; if you wish to turn off automatic version upgrades, you can do so by selecting the “Auto Minor Version Upgrade” for your cache cluster to “No”.

Maintenance Window

Every cache 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 the cache cluster, ElastiCache assigns a 60-minute maintenance window on a randomly selected day of the week.

The 60-minute maintenance window is selected at random from an 8-hour block of time per region. The following table lists the time blocks for each region from which the default maintenance windows are assigned.

Region	Time Block
US East (Northern Virginia) Region	03:00-11:00 UTC
US West (Northern California) Region	06:00-14:00 UTC
US West (Oregon) Region	06:00-14:00 UTC
EU (Ireland) Region	22:00-06:00 UTC

Region	Time Block
Asia Pacific (Singapore) Region	14:00-22:00 UTC
Asia Pacific (Sydney) Region	01:00-09:00 UTC
Asia Pacific (Tokyo) Region	17:00-03:00 UTC
South America (Sao Paulo) Region	01:00-09:00 UTC

The maintenance window should fall at the time of lowest usage and thus might need modification from time to time. You can specify a time range of up to 24 hours in duration during which any maintenance activities you have requested should occur. For example, if you enable Auto Minor Version Upgrades for your cache cluster, non-critical Memcached or Redis software updates are applied during this time. Such updates occur infrequently (generally once every few months) and will be announced on the Amazon ElastiCache forum two weeks prior to being applied. Any deferred or pending cache cluster modifications you have requested would also occur during this time.

For more information about how to adjust the preferred maintenance window for your cache clusters, see Adjusting the Preferred Maintenance Window (p. 73).

ElastiCache and Amazon Virtual Private Cloud

Note
ElastiCache is fully integrated with Amazon Virtual Private Cloud (VPC). If your AWS account supports only the EC2-VPC platform, we always create your cache cluster in a virtual private cloud (VPC).

- If you haven't used ElastiCache before, your cache clusters will be deployed into a VPC. A default VPC will be created for you automatically.
- If you have a default VPC and don't specify a subnet when you launch a cache cluster, the cluster is launched into your default VPC.
- You can launch cache clusters into your default VPC without needing to know anything about Amazon VPC. Your experience with clusters that you launch is the same whether you have a default VPC or not.

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

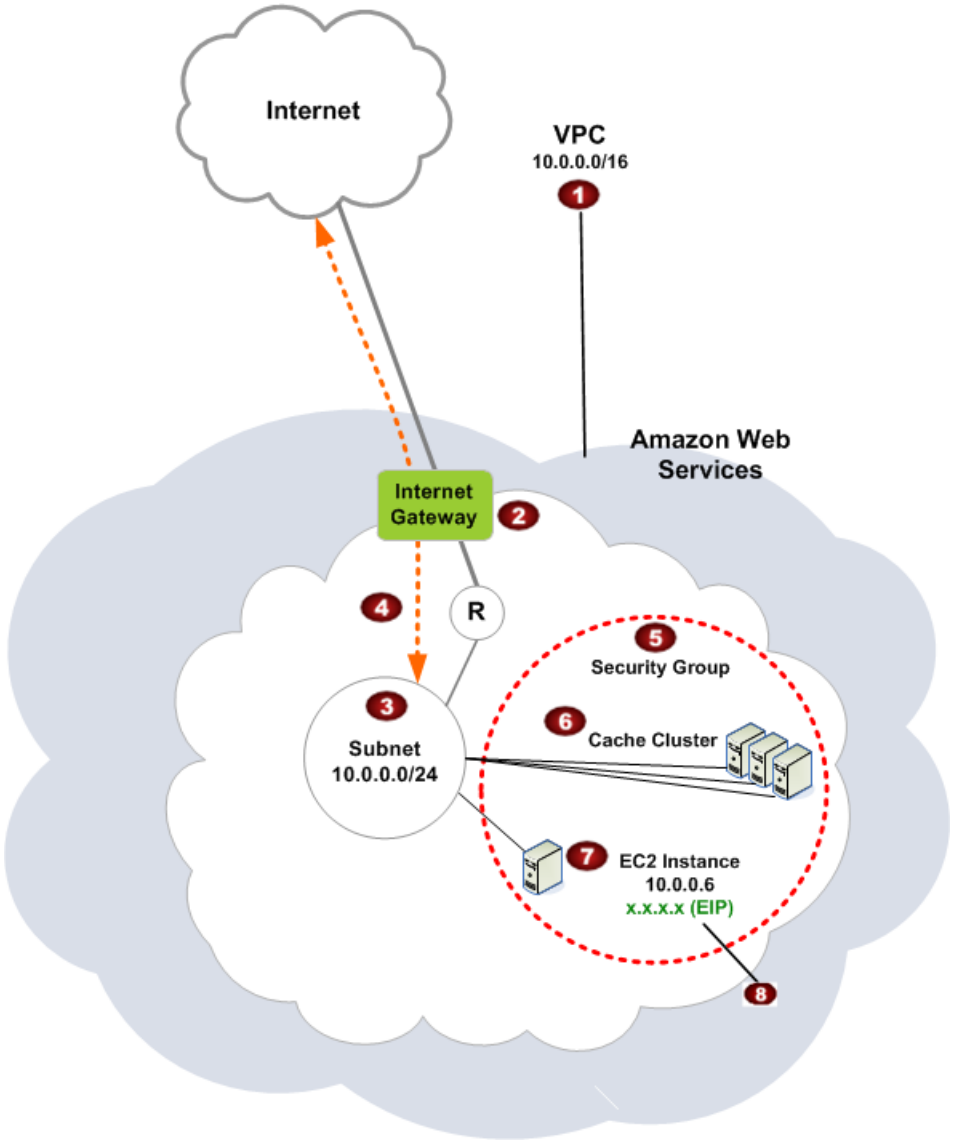
With Amazon Virtual Private Cloud you can create a virtual network in the AWS cloud that closely resembles a traditional data center. You can configure your VPC; you can select its IP address range, create subnets, and configure route tables, network gateways, and security settings.

The basic functionality of ElastiCache is the same in a virtual private cloud; ElastiCache manages software upgrades, patching, failure detection and recovery whether your cache clusters are deployed inside or outside a VPC.

ElastiCache cache nodes deployed outside a VPC are assigned an external IP address (to which the endpoint/DNS Name resolves), which provides connectivity from Amazon Elastic Compute Cloud (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 a VPC

The following diagram and table describe the VPC environment, along with ElastiCache clusters and EC2 instances that are launched in the VPC.



1	The VPC is an isolated portion of the AWS cloud that is assigned its own block of IP addresses.
2	An Internet gateway connects your 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 VPC.
3	An Amazon VPC subnet is a segment of the IP address range of a VPC where you can isolate AWS resources according to your security and operational needs.
4	A routing table in the VPC directs network traffic between the subnet and the Internet. The VPC has an implied router, which is symbolized in this diagram by the circle with the R.

5	A VPC security group controls inbound and outbound traffic for your ElastiCache clusters and EC2 instances.
6	You can launch an ElastiCache cluster in the subnet. The cache cluster nodes have private IP addresses from the subnet's range of addresses.
7	You can also launch EC2 instances in the subnet. Each EC2 instance has a private IP address from the subnet's range of addresses. The EC2 instance can connect to any cache cluster node in the same subnet.
8	For an EC2 instance in your VPC to be reachable from the Internet, you need to assign a static, public address called an Elastic IP address to the instance.

Prerequisites

In order to create an ElastiCache cluster within a VPC, your VPC must meet the following requirements:

- The VPC must allow non-dedicated EC2 instances. You cannot use ElastiCache in a VPC that is configured for dedicated instance tenancy.
- A cache subnet group must be defined for your 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 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 VPC to control where traffic flows (e.g., to the Internet gateway, virtual private gateway). With an Internet gateway, your VPC has direct access to other AWS resources that are not running in your 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 VPC security groups to help secure the ElastiCache clusters and EC2 instances in your VPC. Security groups act like a firewall at the instance level, not the subnet level.

Cache clusters in an Amazon VPC can be accessed by EC2 instances in the same VPC. If these EC2 instances are deployed in a public subnet with associated Elastic IPs, you can access the EC2 instances via the Internet.

> **Note**
> We strongly recommend that you use DNS names to connect to your cache nodes, as the underlying IP address can change if you reboot the cache node.

For more information about using ElastiCache with Amazon VPC, see Using ElastiCache with Amazon Virtual Private Cloud (VPC) (p. 113).

Amazon VPC Documentation

Amazon VPC has its own set of documentation to describe how to create and use your VPC. The following table gives links to the Amazon VPC guides.

Description	Documentation
How to get started using Amazon VPC	Amazon Virtual Private Cloud Getting Started Guide
How to use Amazon VPC through the AWS Management Console	Amazon Virtual Private Cloud User Guide
Complete descriptions of all the Amazon VPC commands	Amazon Elastic Compute Cloud Command Line Reference (the Amazon VPC commands are part of the Amazon EC2 reference)
Complete descriptions of the Amazon VPC API actions, data types, and errors	Amazon Elastic Compute Cloud API Reference (the Amazon VPC API actions are part of the Amazon EC2 reference)
Information for the network administrator who needs to configure the gateway at your end of an optional IPsec VPN connection	Amazon Virtual Private Cloud Network Administrator Guide

For more detailed information about Amazon Virtual Private Cloud, go to http://aws.amazon.com/vpc.

Cache Subnet Groups

A *cache subnet group* is a collection of subnets (typically private) that you may want to designate for your cache clusters running in an Amazon Virtual Private Cloud environment.

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

For more information about cache subnet group usage in an Amazon VPC environment, see ElastiCache and Amazon Virtual Private Cloud (p. 42).

Cache Security Groups

Note
Cache security groups are only applicable to cache clusters that are *not* running in an Amazon Virtual Private Cloud environment (VPC).

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

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

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

For more information about working with cache security groups, see Managing Cache Security Groups (p. 126).

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 several 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 (e.g. creating cache clusters) runs only in your current default region. The default region can be changed by setting the EC2_REGION environment variable, or be overridden by using the `--url` parameter with the command line interface. For more information, see *Common Options for API Tools*.

To create or work with a cache cluster in a specific region, use the corresponding regional service endpoint.

For a listing of ElastiCache endpoints, go to the "Regions and Endpoints" section of the Amazon Web Services General Reference.

Cache Node Auto Discovery

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

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. With Auto Discovery, your application does not need to manually connect to individual cache nodes; instead, your application connects to a *configuration endpoint*. The configuration endpoint DNS entry contains the CNAME entries for each of the cache node endpoints; thus, by connecting to the configuration endpoint, you application immediately "knows" about all of the nodes in the cluster and can connect to all of them. You do not need to hardcode 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.

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 marked as unavailable.
- A client program only needs to connect to the configuration endpoint. After that, the Auto Discovery library connects asynchronously to all of the other nodes in the cache 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. The client then connects to, or disconnects from, these nodes as needed.

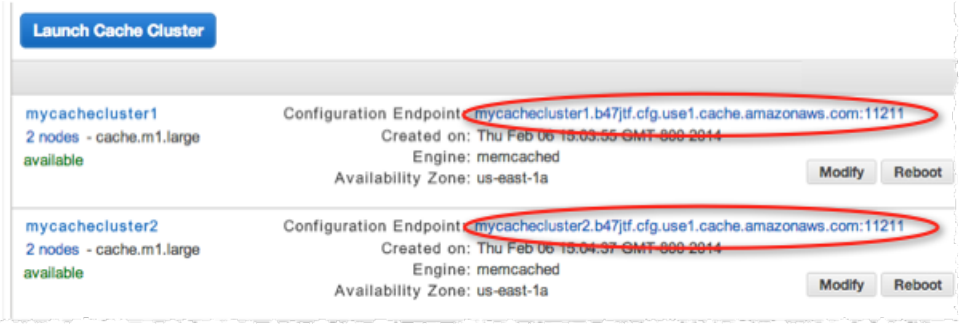
Auto Discovery is enabled on all ElastiCache Memcached cache clusters. You do not need to reboot any of your cache nodes to use this feature.

To begin using Auto Discovery, follow these steps:

- Step 1: Obtain the Configuration Endpoint (p. 48)
- Step 2: Download the ElastiCache Cluster Client (p. 48)
- Step 3: Modify Your Application Program (p. 49)

Step 1: Obtain the Configuration Endpoint

To connect to a cluster, client programs must know the cluster configuration endpoint. You can obtain the configuration endpoint using the AWS Management Console.



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

Example

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

CACHECLUSTER mycluster        mycluster.fnjyzo.cfg.use1.cache.amazonaws.com
       11211 https://console.aws.amazon.com/elasticache/home#client-download:
 2013-07-30T00:57:50.911Z cache.m1.small memcached available 2 us-east-1a 1.4.14

        SECGROUP default active PARAMGRP default.memcached1.4 in-sync
        NOTIFICATION arn:aws:sns:us-east-1:740835402826:autodiscovery active
  ...
```

Important
Please ensure that you are using the latest version of the ElastiCache Command Line Toolkit. To download the toolkit, go to http://aws.amazon.com/developertools/Amazon-ElastiCache.

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 and PHP, and contains all of the necessary logic for discovering and connecting to all of your cache nodes.

To download the ElastiCache Cluster Client

1. Sign in to the AWS Management Console and open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. From the ElastiCache console, click **\ElastiCache Cluster Client** then click **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. You are free to modify the source code as you see fit; you can even incorporate the code into other open source Memcached libraries, or into your own client code.

Note
To use the ElastiCache Cluster Client for PHP, you will first need to install it on your Amazon EC2 instance. For more information, see Appendix: Installing the ElastiCache Cluster Client for PHP (p. 151)

Step 3: Modify Your Application Program

You are now ready to modify your application program so that it uses Auto Discovery. The following sections show how to use the ElastiCache Cluster Client for Java and PHP.

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 will connect to all of the nodes in the cluster without any further intervention.

```
package com.amazon.elasticache;

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

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

public class AutoDiscoveryDemo {

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

       String configEndpoint = "mycluster.fnjyzo.cfg.use1.cache.amazonaws.com";

        Integer clusterPort = 11211;

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

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

    }
}
```

Using the ElastiCache Cluster Client for PHP

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

Note

To use the ElastiCache Cluster Client for PHP, you will first need to install it on your Amazon EC2 instance. For more information, see Appendix: Installing the ElastiCache Cluster Client for PHP (p. 151)

```
<?php

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

  /* Configuration endpoint to use to initialize memcached client. This is only
 an example. */
  $server_endpoint = "php-autodiscovery.1zvgtq.cfg.use1.cache.amazonaws.com";
  /* Port for connecting to the ElastiCache cluster. This is only an example
*/
  $server_port = 11211;

 /**
  * The following will initialize a Memcached client to utilize the Auto Discov
ery 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.
  */

  $dynamic_client = new Memcached();
  $dynamic_client->setOption(Memcached::OPT_CLIENT_MODE, Memcached::DYNAMIC_CLI
ENT_MODE);
  $dynamic_client->addServer($server_endpoint, $server_port);
  $dynamic_client->set('key', 'value', 60);  // Store the data for 60 seconds
in the cluster, the client will decide which node to store


 /**
  * 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();
  $static_client->setOption(Memcached::OPT_CLIENT_MODE, Memcached::STATIC_CLI
ENT_MODE);
  $static_client->addServer($server_endpoint, $server_port);
  $static_client->set('key', 'value');  // Store the data in the cluster without
 expiration

?>
```

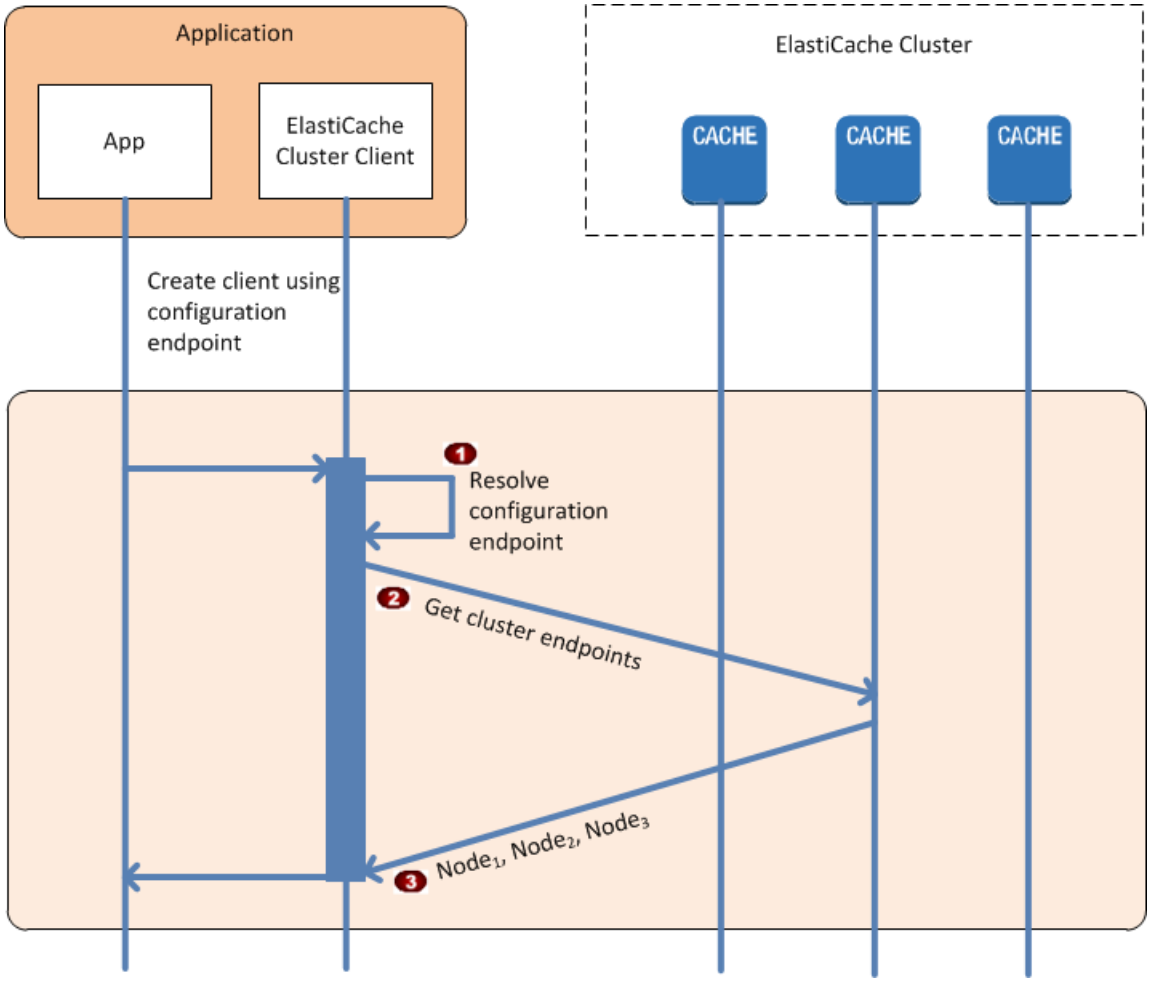
How Auto Discovery Works

Topics

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

Connecting to Cache Nodes

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



Process of Connecting to Cache Nodes

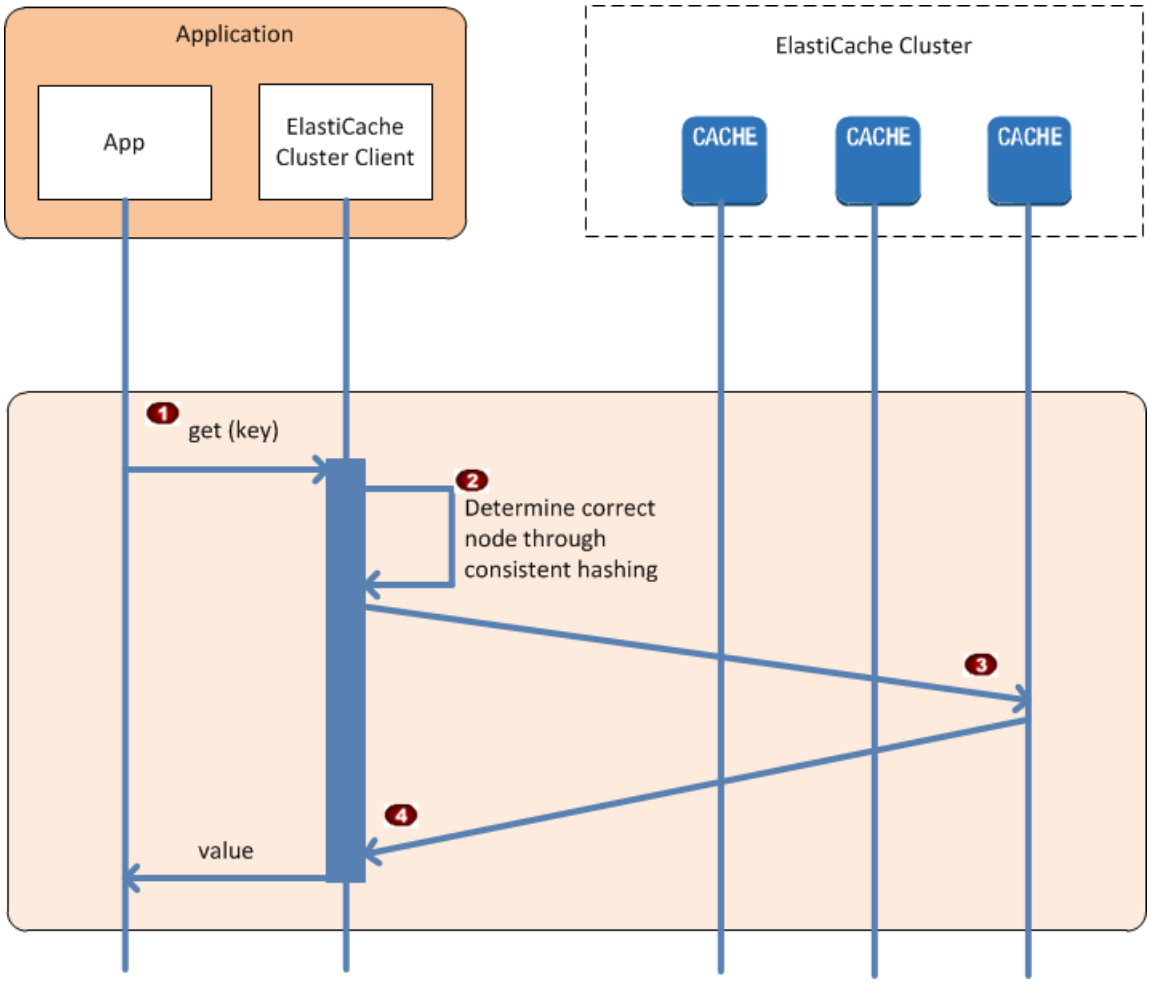
1	The application resolves the configuration endpoint's DNS name. Because the configuration endpoint maintains CNAME entries for all of the cache nodes, the DNS name resolves to one of the nodes; the client can then connect to that node.

2	The client requests the configuration information for all of the other nodes. Since each node maintains configuration information for all of the nodes in the cluster, any node can pass configuration information to the client upon request.
3	The client receives the current list of cache node hostnames and IP addresses. It can then connect to all of the other nodes in the cluster.

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

Normal Cluster Operations

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



Process of Normal Cluster Operations

1	The application issues a *get* request for a particular data item, identified by its key.

2	The client uses a hashing algorithm against the key to determine which cache node contains the data item.
3	The data item is requested from the appropriate node.
4	The data item is returned to the application.

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 `elasticache-describe-cache-clusters` command with the `--show-cache-node-info` parameter.

Example

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

```
...

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

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

...
```

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

Adding Auto Discovery To Your Client Library

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

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

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

The outputs from these two commands are identical, and are described in the Output Format (p. 54) section below.

Cache Engine Version 1.4.14 or Higher

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

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

Syntax

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

Options

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

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

```
config get cluster
```

Cache Engine Version Lower Than 1.4.14

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

```
get AmazonElastiCache:cluster
```

> **Note**
> We recommend that you 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.

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

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

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

Example

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

```
CONFIG cluster 0 147\r\n
12\r\n
myCluster.pc4ldq.0001.use1.cache.amazonaws.com|10.82.235.120|11211 myC
luster.pc4ldq.0002.use1.cache.amazonaws.com|10.80.249.27|11211\n\r\n
END\r\n
```

Note

The second line indicates that the configuration information has been modified twelve times so far.

In the third line, the list of nodes is in alphabetical order by hostname. This ordering might be in a different sequence from what you are currently using in your client application.

CloudWatch Metrics with ElastiCache

Topics

ElastiCache provides metrics that enable you to monitor your cache 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. 61)

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 using the `DescribeCacheClusters` API or **elasticache-describe-cache-clusters** command line utility.

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

See Also

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 Viewing Cache Cluster and Cache Node Metrics (p. 131).

Host-Level Metrics

The following table lists host-level metrics provided by ElastiCache for individual cache nodes.

See Also

- Metrics for Memcached (p. 57)
- Metrics for Redis (p. 60)

Metric	Description	Unit
CPUUtilization	The percentage of CPU utilization.	Percent
SwapUsage	The amount of swap used on the host.	Bytes
FreeableMemory	The amount of free memory available on the host.	Bytes
NetworkBytesIn	The number of bytes the host has read from the network.	Bytes
NetworkBytesOut	The number of bytes the host has written to the network.	Bytes

Metrics for Memcached

The following table lists the metrics provided by ElastiCache that are derived from the Memcached *stats* command. Each metric is calculated at the cache node level.

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

See Also

- Host-Level Metrics (p. 57)

Metric	Description	Unit
BytesUsedForCacheItems	The number of bytes used to store cache items.	Bytes
BytesReadIntoMemcached	The number of bytes that have been read from the network by the cache node.	Bytes
BytesWrittenOutFromMemcached	The number of bytes that have been written to the network by the cache node.	Bytes
CasBadval	The number of CAS (check and set) requests the cache has received where the Cas value did not match the Cas value stored.	Count
CasHits	The number of Cas requests the cache has received where the requested key was found and the Cas value matched.	Count
CasMisses	The number of Cas requests the cache has received where the key requested was not found.	Count
CmdFlush	The number of flush commands the cache has received.	Count
CmdGet	The number of get commands the cache has received.	Count
CmdSet	The number of set commands the cache has received.	Count
CurrConnections	A count of the number of connections connected to the cache at an instant in time. Note that due to the design of Memcached, this will always return a minimum count of 10.	Count
CurrItems	A count of the number of items currently stored in the cache.	Count
DecrHits	The number of decrement requests the cache has received where the requested key was found.	Count
DecrMisses	The number of decrement requests the cache has received where the requested key was not found.	Count
DeleteHits	The number of delete requests the cache has received where the requested key was found.	Count
DeleteMisses	The number of delete requests the cache has received where the requested key was not found.	Count
Evictions	The number of non-expired items the cache evicted to allow space for new writes.	Count
GetHits	The number of get requests the cache has received where the key requested was found.	Count
GetMisses	The number of get requests the cache has received where the key requested was not found.	Count
IncrHits	The number of increment requests the cache has received where the key requested was found.	Count

Metric	Description	Unit
IncrMisses	The number of increment requests the cache has received where the key requested was not found.	Count
Reclaimed	The number of expired items the cache evicted to allow space for new writes.	Count

For Memcached 1.4.14, the following additional metrics are provided.

Metric	Description	Unit
BytesUsedForHash	The number of bytes currently used by hash tables.	Bytes
CmdConfigGet	The cumulative number of "config get" requests.	Count
CmdConfigSet	The cumulative number of "config set" requests.	Count
CmdTouch	The cumulative number of "touch" requests.	Count
CurrConfig	The current number of configurations stored.	Count
EvictedUnfetched	The number of valid items evicted from the least recently used cache (LRU) which were never touched after being set.	Count
ExpiredUnfetched	The number of expired items reclaimed from the LRU which were never touched after being set.	Count
SlabsMoved	The total number of slab pages that have been moved.	Count
TouchHits	The number of keys that have been touched and were given a new expiration time.	Count
TouchMisses	The number of items that have been touched, but were not found.	Count

The following table describes the available calculated cache level metrics.

Metric	Description	Unit
NewConnections	The number of new connections the cache has received. This is derived from the memcached total_connections statistic by recording the change in total_connections across a period of time. This will always be at least 1, due to a connection reserved for a ElastiCache.	Count
NewItems	The number of new items the cache has stored. This is derived from the memcached total_items statistic by recording the change in total_items across a period of time.	Count
UnusedMemory	The amount of unused memory the cache can use to store items. This is derived from the memcached statistics limit_maxbytes and bytes by subtracting bytes from limit_maxbytes.	Bytes

Metrics for Redis

The following table lists the metrics provided by ElastiCache. With the exception of *ReplicationLag*, these metrics are derived from the Redis *info* command. Each metric is calculated at the cache node level.

For complete documentation of the Redis *info* command, go to http://redis.io/commands/info.

See Also

- Host-Level Metrics (p. 57)

Metric	Description	Unit
CurrConnections	The number of client connections, excluding connections from read replicas.	Count
Evictions	The number of keys that have been evicted due to the *maxmemory* limit.	Count
Reclaimed	The total number of key expiration events.	Count
NewConnections	The total number of connections that have been accepted by the server during this period.	Count
BytesUsedForCache	The total number of bytes allocated by Redis.	Bytes
CacheHits	The number of successful key lookups.	Count
CacheMisses	The number of unsuccessful key lookups.	Count
ReplicationLag	This metric is only applicable for a cache node running as a read replica. It represents how far behind, in seconds, the replica is in applying changes from the primary cache cluster.	Seconds

These are aggregations of certain kinds of commands, derived from *info commandstats*:

Metric	Description	Unit
GetTypeCmds	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.)	Count
SetTypeCmds	The total number of *set* types of commands. This is derived from the Redis commandstats statistic by summing all of the *set* types of commands (*set*, *hset*, etc.)	Count
KeyBasedCmds	The total number of commands that are key-based. This is derived from the Redis commandstats statistic by summing all of the commands that act upon one or more keys.	Count

Metric	Description	Unit
StringBasedCmds	The total number of commands that are string-based. This is derived from the Redis commandstats statistic by summing all of the commands that act upon one or more strings.	Count
HashBasedCmds	The total number of commands that are hash-based. This is derived from the Redis commandstats statistic by summing all of the commands that act upon one or more hashes.	Count
ListBasedCmds	The total number of commands that are list-based. This is derived from the Redis commandstats statistic by summing all of the commands that act upon one or more lists.	Count
SetBasedCmds	The total number of commands that are set-based. This is derived from the Redis commandstats statistic by summing all of the commands that act upon one or more sets.	Count
SortedSetBasedCmds	The total number of commands that are sorted set-based. This is derived from the Redis commandstats statistic by summing all of the commands that act upon one or more sorted sets.	Count
CurrItems	The number of items in the cache. This is derived from the Redis keyspace statistic, summing all of the keys in the entire keyspace.	Count

Which Metrics Should I Monitor?

The following CloudWatch metrics offer good insight into ElastiCache performance. In most cases, we recommend that you set CloudWatch alarms for these metrics so that you can take corrective action before performance issues occur.

CPUUtilization

This is a host-level metric.

- *Memcached:* Since Memcached is multi-threaded, this metric can be as high as 90%. If you exceed this threshold, scale you cache cluster up by using a larger cache node type, or scale out by adding more cache nodes.
- *Redis:* Since Redis is single-threaded, the threshold is calculated as (90 / number of processor cores). For example, suppose you are using a *cache.m1.xlarge* node, which has four cores. In this case, the threshold for *CPUUtilization* would be (90 / 4), or 22.5%.

  You will need to determine your own threshold, based on the number of cores in the cache node that you are using. If you exceed this threshold, and your main workload is from read requests, scale your cache cluster out by adding read replicas. If the main workload is from write requests, we recommend scaling up by using a larger cache instance type.

SwapUsage

This is a host-level metric.

- *Memcached:* This metric should not exceed 50 MB. If it does, we recommend that you increase the *ConnectionOverhead parameter value*.
- *Redis:* At this time, we have no recommendation for this parameter; you do not need to set a CloudWatch alarm for it.

Evictions

This is a cache engine metric, published for both Memcached and Redis cache clusters. We recommend that you determine your own alarm threshold for this metric based on your application needs.

- *Memcached:* If your exceed your chosen threshold, scale you cache cluster up by using a larger cache node type, or scale out by adding more cache nodes.
- *Redis:* If you exceed your chosen threshold, scale your cache cluster out by adding read replicas.

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.

Setting Up the ElastiCache Command Line Interface

Topics

This section describes the prerequisites for running the command line tools, where to get the command line tools, how to set up the tools and their environment, and includes a series of common examples of tool usage.

Prerequisites

This document assumes that you can work in a Linux/UNIX or Windows environment. The Amazon ElastiCache command line tools also work on Mac OS X, which is a UNIX-based environment; however, no specific Mac OS X instructions are included in this guide.

As a convention, all command line text is prefixed with a generic **`PROMPT>`** command line prompt. The actual command line prompt on your machine is likely to be different. We also use **`$`** to indicate a Linux/UNIX specific command and **`C:\>`** for a Windows specific command. The example output resulting from the command is shown immediately thereafter without any prefix.

The Java Runtime Environment

The command line tools used in this guide require Java version 5 or later to run. Either a JRE or JDK installation is acceptable. To view and download JREs for a range of platforms, including Linux/UNIX and Windows, go to Java SE Downloads.

Setting the Java Home Variable

The command line tools depend on an environment variable (`JAVA_HOME`) to locate the Java Runtime. This environment variable should be set to the full path of the directory that contains a subdirectory named

`bin` which in turn contains the executable `java` (on Linux and UNIX) or `java.exe` (on Windows) executable.

To set the Java Home variable

1. Set the Java Home variable.

   - On Linux and UNIX, enter the following command:

     ```
     $ export JAVA_HOME=<PATH>
     ```

   - On Windows, enter the following command:

     ```
     C:\> set JAVA_HOME=<PATH>
     ```

2. Confirm the path setting by running **`$JAVA_HOME/bin/java -version`** and checking the output.

   - On Linux/UNIX, you will see output similar to the following:

     ```
     $ $JAVA_HOME/bin/java -version
     java version "1.6.0_23"
     Java(TM) SE Runtime Environment (build 1.6.0_23-b05)
     Java HotSpot(TM) Client VM (build 19.0-b09, mixed mode, sharing)
     ```

   - On Windows, you will see output similar to the following:

     ```
     C:\> %JAVA_HOME%\bin\java -version
     java version "1.6.0_23"
     Java(TM) SE Runtime Environment (build 1.6.0_23-b05)
     Java HotSpot(TM) Client VM (build 19.0-b09, mixed mode, sharing)
     ```

Getting the Command Line Tools

The command line tools are available as a ZIP file on the ElastiCache Developer Tools web site. These tools are written in Java, and include shell scripts for Windows 2000/XP/Vista/Windows 7, Linux/UNIX, and Mac OSX. The ZIP file is self-contained and no installation is required; simply download the zip file and unzip it to a directory on your local machine.

Setting Up the Tools

The command line tools depend on an environment variable (AWS_ELASTICACHE_HOME) to locate supporting libraries. You need to set this environment variable before you can use the tools. Set it to the path of the directory you unzipped the command line tools into. This directory is named

ElastiCacheCli-A.B.nnnn (A, B and n are version/release numbers), and contains subdirectories named bin and lib.

To set the AWS_ELASTICACHE_HOME environment variable

- Open a command line window and enter one of the following commands to set the AWS_ELASTICACHE_HOME environment variable.
  - On Linux and UNIX, enter the following command:

```
$ export AWS_ELASTICACHE_HOME=<path-to-tools>
```

  - On Windows, enter the following command:

```
C:\> set AWS_ELASTICACHE_HOME=<path-to-tools>
```

To make the tools easier to use, we recommend that you add the tools' BIN directory to your system PATH. The rest of this guide assumes that the BIN directory is in your system path.

To add the tools' BIN directory to your system path

- Enter the following commands to add the tools' BIN directory to your system PATH.
  - On Linux and UNIX, enter the following command:

```
$ export PATH=$PATH:$AWS_ELASTICACHE_HOME/bin
```

  - On Windows, enter the following command:

```
C:\> set PATH=%PATH%;%AWS_ELASTICACHE_HOME%\bin
```

Note
The Windows environment variables are reset when you close the command window. You might want to set them permanently. Consult the documentation for your version of Windows for more information.

Note
Paths that contain a space must be wrapped in double quotes, for example:
"C:\Program Files\Java"

Providing Credentials for the Tools

The command line tools need the AWS Access Key and Secret Access Key provided with your AWS account. You can get them using the command line or from a credential file located on your local system.

The deployment includes a template file ${AWS_ELASTICACHE_HOME}/credential-file-path.template that you need to edit with your information. Following are the contents of the template file:

```
AWSAccessKeyId=<Write your AWS access ID>
AWSSecretKey=<Write your AWS secret key>
```

Important
On UNIX, limit permissions to the owner of the credential file:

```
$ chmod 600 <the file created above>
```

With the credentials file setup, you'll need to set the AWS_CREDENTIAL_FILE environment variable so that the ElastiCache tools can find your information.

To set the AWS_CREDENTIAL_FILE environment variable

1. Set the environment variable:

   - On Linux and UNIX, update the variable using the following command:

     ```
     $ export AWS_CREDENTIAL_FILE=<the file created above>
     ```

   - On Windows, set the variable using the following command:

     ```
     C:\> set AWS_CREDENTIAL_FILE=<the file created above>
     ```

2. Check that your setup works properly, run the following command:

   ```
   elasticache --help
   ```

   You should see the usage page for all ElastiCache commands.

Managing ElastiCache

Topics

This section covers the ElastiCache operations you are most likely to use, and provides procedural instruction and examples.

Managing Cache Clusters

This section covers operations on cache clusters.

Topics

Creating a Cache Cluster

When you create a new cache cluster, you need to name it, specify the cache engine software to use (Memcached or Redis), and choose the number and type of cache nodes.

> **Note**
> If you are creating a Redis cache cluster, you can seed it with data from ElastiCache for Redis snapshots or from your own RDB files stored in Amazon Simple Storage Service. For more information, see Migrate Your Redis Cache to ElastiCache (p. 100).

The following procedures show you how to create a cache cluster.

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Click **Launch Cache Cluster**.
3. On the **Cache Cluster Details** page, do the following. When all the settings are as you want them, click **Next**:

   a. In the **Name** box, type the name of your cache cluster.

   b. In the **Node Type** box, choose one of the available cache node types.

      For a complete listing of cache node types and specifications, see Cache Node Type-Specific Parameters for Memcached or Cache Node Type-Specific Parameters for Redis and Amazon ElastiCache Product Features and Details.

   c. In the **Number of Nodes** box, type the number of cache nodes that you want in your cache cluster.

      > **Note**
      > For a Redis cache cluster, the number of nodes must be 1. For a Memcached cache cluster, the number of nodes must be at least 1 and no more than 20.

   d. In the **Engine** box, choose the cache engine you want, *memcached* or *redis*.

   e. (Optional) If you are creating a Redis cache cluster and want to seed it with data from an RDB file, type the Amazon S3 location for that snapshot into the **S3 Snapshot Location** text box. For more information on placing an RDB file Amazon S3, see Migrate Your Redis Cache to ElastiCache (p. 100).

   f. To change the port that your cache nodes will use to accept connections, in the **Cache Port** box, type the new port number.

g. To send SNS notifications related to cache cluster events, in the **Topic for SNS notifications** box, click an existing Amazon SNS topic ID. If you have not configured Amazon SNS event notification, click **Disable Notifications**.

4. Leave the rest of the options on the **Cache Cluster Details** page at their default values, and click the **Next** button.
5. On the **Additional Configuration** page, leave the options at their default values, and click the **Next** button.
6. On the **Review** page, review the options for your cache cluster. If you need to correct any options, click the **Previous** button to return to previous panels and make corrections. If all your options are as you want them, click **Launch** to launch your new cache cluster.
7. When the message appears stating that your cache cluster is being created, click **Close**. Cluster creation can take a few minutes to complete.

On the ElastiCache console, your cache cluster is listed in the **Cache Clusters** panel. Its status is displayed as **creating** until it is ready for use.

CLI

To create a cache cluster, use the `elasticache-create-cache-cluster` command . The following example creates a Memcached cluster named *my-cache-cluster* that has three cache nodes.

```
PROMPT> elasticache-create-cache-cluster my-cache-cluster -n 3 -c cache.m1.large
 -e memcached -sg default
```

This command should produce output similar to the following:

```
CACHECLUSTER  my-cache-cluster  https://console.aws.amazon.com/elastic
ache/home#client-download:  cache.m1.large  memcached  creating  3  1.4.14
      CACHESECURITYGROUP  default  active
      CACHEPARAMETERGROUP  default.memcached1.4  in-sync
```

API

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

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?CacheNodeType=cache.m1.small
    &NumCacheNodes=1
    &CacheClusterId=my-redis-primary
    &Engine=redis
    &SnapshotArns.member.1=arn%3Aaws%3As3%3A%3A%3Amy-bucket%2Fdump.rdb
    &Version=2013-06-15
    &Action=CreateCacheCluster
    &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>
```

Adding or Removing Cache Nodes

By adding or removing cache nodes, you can accommodate changes in your application's capacity requirements.

> **Note**
> At this time, you can only add or remove cache nodes from cache clusters running Memcached.

The following procedures show you how to add and remove cache nodes.

Adding Nodes To A Cache Cluster

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the **Cache Clusters** list, click the cache cluster you want to modify in the **Cache Clusters** list. The detail panel appears.
3. Click the **Nodes** tab. The list of cache nodes for the cache cluster appears.
4. Click the **Add Node(s)** button at the top of the list. The **Add Node(s)** dialog box appears.
5. Type the number of nodes you want to add into the **Number of nodes to add** text box.
6. Select the **Apply Immediately** check box.
7. Click the **Yes, Add** button.

   After a few moments, the new cache nodes will show up in the nodes list with a status of "Creating".

CLI

Use the command `elasticache-modify-cache-cluster` with the following parameters:

```
PROMPT> elasticache-modify-cache-cluster my-cache-cluster --num-cache-nodes 5
--apply-immediately
```

This command produces output similar to the following:

```
CACHECLUSTER  my-cache-cluster  2013-07-06T23:34:09.756Z  cache.m1.large  mem
cached
modifying  3  us-east-1b  1.4.5  5
   SECGROUP  default  active
   PARAMGRP  default.memcached1.4  in-sync
```

API

- Call `ModifyCacheCluster` with the following parameters:
  - *`CacheClusterId`* = `my-cache-cluster`
  - *`NumCacheNodes`* = `5`
  - *`ApplyImmediately`* = `true`

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=true
    &NumCacheNodes=5
    &CacheClusterId=my-cache-cluster
    &Version=2013-06-15
    &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>
```

Removing Nodes From A Cache Cluster

AWS Management Console

The following procedures show how you can remove nodes from a cache 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 **Cache Clusters** list, click the cache cluster you want to modify in the **Cache Clusters** list. The detail panel appears.

3. Click the **Nodes** tab. The list of cache nodes for the cache cluster appears.
4. In the Nodes list, click the check box next to the cache nodes you want to remove from the cache cluster.
5. Click the **Remove Node(s)** button.

   The **Remove Node(s)** confirmation dialog box appears.
6. Click the **Apply Immediately** checkbox.
7. Click the **Yes, Remove** button.

   After a few moments, the new cache nodes will show up in the nodes list with a status of "deleting".

CLI

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

```
PROMPT> elasticache-describe-cache-clusters my-cache-cluster -sn
```

   This command produces output similar to the following:

```
CACHECLUSTER  my-cache-cluster  2013-07-06T23:34:09.756Z  cache.m1.large
memcached
available  5  us-east-1b  1.4.5
   SECGROUP  default  active
   PARAMGRP  default.memcached1.4  in-sync
   CACHENODE  0001  2013-07-14T23:39:51.273Z  available  my-cache-
cluster.m2st2p.f
sw4.use1qa.cache.amazonaws.com  11211  in-sync
   CACHENODE  0002  2013-07-14T23:39:51.276Z  available  my-cache-
cluster.m2st2p.f
sw7.use1qa.cache.amazonaws.com  11211  in-sync
   CACHENODE  0003  2013-07-06T23:34:09.756Z  available  my-cache-
cluster.m2st2p.f
swc.use1qa.cache.amazonaws.com  11211  in-sync
   CACHENODE  0004  2013-07-06T23:34:09.756Z  available  my-cache-
cluster.m2st2p.f
swd.use1qa.cache.amazonaws.com  11211  in-sync
   CACHENODE  0005  2013-07-06T23:34:09.756Z  available  my-cache-
cluster.m2st2p.f
swf.use1qa.cache.amazonaws.com  11211  in-sync
```

2. Use the command `elasticache-modify-cache-cluster` with a list of the cache nodes to remove, as in the following example.

```
PROMPT> elasticache-modify-cache-cluster my-cache-cluster --num-cache-nodes
 3 --nodes-to-remove 0004,0005 --apply-immediately
```

   This command produces output similar to the following:

```
CACHECLUSTER  my-cache-cluster  2013-07-06T23:34:09.756Z  cache.m1.large
memcached
modifying  3  us-east-1b  1.4.5  5
   SECGROUP  default  active
   PARAMGRP  default.memcached1.4  in-sync
```

API

Call `ModifyCacheCluster` with the cache cluster ID and a list of cache nodes to remove:

- *`CacheClusterId`* = `my-cache-cluster`
- *`NumCacheNodes`* = `3`
- *`CacheClusterNodeIdsToRemove.member.1`* = `0004`
- *`CacheClusterNodeIdsToRemove.member.2`* = `0005`
- *`ApplyImmediately`* = `true`

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=ModifyCacheCluster
    &CacheClusterId=my-cache-cluster
    &ApplyImmediately=true
    &CacheClusterNodeIdsToRemove.member.1=0004
    &CacheClusterNodeIdsToRemove.member.2=0005
    &NumCacheNodes=3
    &Version=2013-06-15
    &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>
```

Adjusting the Preferred Maintenance Window

Every cache 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 the cache cluster, ElastiCache assigns a one hour maintenance window on a randomly selected day of the week. The one hour maintenance window is selected at random from an 8-hour block of time per region.

The maintenance window should fall at the time of lowest usage and thus might need modification from time to time. You can specify the time range (UTC) during which any maintenance activities you have requested should occur. For example, if you enable Auto Minor Version Upgrades for your cache cluster, non-critical software updates for Memcached or Redis are applied during this time. Such updates occur infrequently (generally once every few months) and will be announced on the AWS ElastiCache forum

two weeks prior to being applied. Any deferred or pending cache cluster modifications you have requested would also occur during this time.

Adjusting the Maintenance Window

The following procedures show you how to adjust the preferred maintenance window for a cache cluster.

For the purpose of this example, we assume that the cache cluster named *my-cache-cluster* exists and has a preferred maintenance window of "Sun:05:00-Sun:06:00" UTC.

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Click on the **Cache Clusters** link in the navigation list on the left side of the console display.

   The **Cache Clusters** list appears.
3. Click the **Modify** link next to the cache cluster in the **Cache Clusters** list.

   The **Modify Cache Cluster** window appears.
4. Select the starting and ending time in the **Maintenance Window** section of the **Modify Cache Cluster** dialog box.
5. Select the **Apply Immediately** check box.
6. Click the **Yes, Modify** button.

CLI

To change the maintenance window for a cache cluster, use the `elasticache-modify-cache-cluster` command with the `--preferred-maintenance-window` parameter:

```
PROMPT> elasticache-modify-cache-cluster my-cache-cluster --preferred-mainten
ance-window Tue:04:00-Tue:05:00
```

This command produces output similar to the following.

```
CACHECLUSTER  my-cache-cluster  2013-07-06T23:34:09.756Z  cache.m1.large  mem
cached
available  3  us-east-1b  1.4.5
 SECGROUP  default  active
 PARAMGRP  default.memcached1.4  in-sync
```

API

Call `ModifyCacheCluster` with the following parameters:

- *`CacheClusterId`* = `my-cache-cluster`
- *`PreferredMaintenanceWindow`* = `Tue:04:00-Tue:05:00`

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=ModifyCacheCluster
    &CacheClusterId=my-cache-cluster
    &PreferredMaintenanceWindow=Tue:04:00-Tue:05:00
    &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>
```

Managing Reserved Cache Nodes

Reserved cache nodes let you make a one-time, up-front payment for a cache node, reserve the cache node for a one- or three-year term, and pay a significantly lower rate for each hour you run that cache node. Reserved cache nodes are available in three varieties - Heavy Utilization, Medium Utilization, and Light Utilization - that enable you to optimize your ElastiCache costs based on your expected utilization. You can use the command line or the API to list and purchase available reserved cache node offerings. Reserved cache node offerings are based on the cache node class and duration.

In this example, you see how to view available reserved cache node offerings, purchase an available reserved cache node offering, and list reserved cache nodes for your account.

Describing Available Reserved Cache Node Offerings

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

This example shows how to get pricing and information about available reserved cache node offerings.

CLI

Enter the following command at a command prompt:

```
PROMPT> elasticache-describe-reserved-cache-nodes-offerings  --headers
```

This call returns output similar to the following:

```
OFFERING  OfferingId                          Class           Duration  Fixed
 Price  Usage Price  Description  Offering Type
OFFERING  438012d3-4052-4cc7-b2e3-8d3372e0e706  cache.m1.large   1y
1820.00 USD  0.368 USD    memcached    Medium Utilization
OFFERING  649fd0c8-cf6d-47a0-bfa6-060f8e75e95f  cache.m1.small   1y
227.50 USD  0.046 USD    memcached    Medium Utilization
OFFERING  123456cd-ab1c-47a0-bfa6-12345667232f  cache.m1.small   1y
```

```
162.00 USD   0.00 USD    memcached    Heavy Utilization
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   0.123   USD       Hourly
OFFERING  123456cd-ab1c-37a0-bfa6-12345667232d  cache.m1.large   1y
700.00 USD   0.00 USD    memcached    Heavy Utilization
    Recurring Charges:   Amount  Currency  Frequency
    Recurring Charges:   1.25    USD       Hourly
OFFERING  123456cd-ab1c-17d0-bfa6-12345667234e  cache.m1.xlarge  1y
4242.00 USD   2.42 USD    memcached    Light Utilization
```

API

Call `DescribeReservedCacheNodesOfferings`.

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=DescribeReservedCacheNodesOfferings
    &Version=2013-06-15
    &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>
```

This call returns output similar to the following:

```
<DescribeReservedCacheNodesOfferingsResponse xmlns="http://elasticache.us-east-
1.amazonaws.com/doc/2013-06-15/">
  <DescribeReservedCacheNodesOfferingsResult>
    <ReservedCacheNodesOfferings>
      <ReservedCacheNodesOffering>
        <Duration>31536000</Duration>
        <OfferingType>Medium Utilization</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <FixedPrice>1820.0</FixedPrice>
        <ProductDescription>memcached</ProductDescription>
        <UsagePrice>0.368</UsagePrice>
        <ReservedCacheNodesOfferingId>438012d3-4052-4cc7-b2e3-
8d3372e0e706</ReservedCacheNodesOfferingId>
        <CacheNodeType>cache.m1.large</CacheNodeType>
      </ReservedCacheNodesOffering>
      <ReservedCacheNodesOffering>

      (...output omitted...)

      </ReservedCacheNodesOffering>
    </ReservedCacheNodesOfferings>
```

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

Some of the output has been omitted for brevity.

Purchasing a Reserved Cache Node

This example shows how to purchase a reserved cache node offering.

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

CLI

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

Enter the following command at a command prompt:

```
PROMPT> elasticache-purchase-reserved-cache-nodes-offering 649fd0c8-cf6d-47a0-
bfa6-060f8e75e95f -i myreservationID
```

The command returns output similar to the following:

```
RESERVATION  ReservationId      Class           Start Time                Dura
tion  Fixed Price  Usage Price  Count  State           Description     Offering
 Type
RESERVATION  myreservationid    cache.m1.small  2013-12-19T00:30:23.247Z  1y
      455.00 USD   0.092 USD    1      payment-pending  memcached        Medium
 Utilization
```

API

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

Call `PurchaseReservedCacheNodesOffering` with the following parameters:

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

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=PurchaseReservedCacheNodesOffering
    &ReservedCacheNodesOfferingId=649fd0c8-cf6d-47a0-bfa6-060f8e75e95f
    &ReservedCacheNodeID=myreservationID
    &CacheNodeCount=1
    &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>
```

This call returns output similar to the following:

```
<PurchaseReservedCacheNodesOfferingResponse xmlns="http://elasticache.us-east-
1.amazonaws.com/doc/2013-06-15/">
  <PurchaseReservedCacheNodesOfferingResult>
    <ReservedCacheNode>
      <OfferingType>Medium Utilization</OfferingType>
      <CurrencyCode>USD</CurrencyCode>
      <RecurringCharges/>
      <ProductDescription>memcached</ProductDescription>
      <ReservedCacheNodesOfferingId>649fd0c8-cf6d-47a0-bfa6-060f8e75e95f</Re
servedCacheNodesOfferingId>
      <State>payment-pending</State>
      <ReservedCacheNodeId>myreservationID</ReservedCacheNodeId>
      <CacheNodeCount>10</CacheNodeCount>
      <StartTime>2013-07-18T23:24:56.577Z</StartTime>
      <Duration>31536000</Duration>
      <FixedPrice>123.0</FixedPrice>
      <UsagePrice>0.123</UsagePrice>
      <CacheNodeType>cache.m1.small</CacheNodeType>
    </ReservedCacheNode>
  </PurchaseReservedCacheNodesOfferingResult>
  <ResponseMetadata>
    <RequestId>7f099901-29cf-11e1-bd06-6fe008f046c3</RequestId>
  </ResponseMetadata>
</PurchaseReservedCacheNodesOfferingResponse>
```

Describing Reserved Cache Nodes

You can get information about reserved cache nodes for your AWS account.

CLI

Enter the following command at a command prompt:

```
PROMPT> elasticache-describe-reserved-cache-nodes --headers
```

This command should return output similar to the following:

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

API

Call `DescribeReservedCacheNodes`.

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=DescribeReservedCacheNodes
    &Version=2013-06-15
    &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>
```

The API returns output similar to the following:

```
<DescribeReservedCacheNodesResponse xmlns="http://elasticache.us-east-
1.amazonaws.com/doc/2013-06-15/">
  <DescribeReservedCacheNodesResult>
    <ReservedCacheNodes>
      <ReservedCacheNode>
        <OfferingType>Medium Utilization</OfferingType>
        <CurrencyCode>USD</CurrencyCode>
        <RecurringCharges/>
        <ProductDescription>memcached</ProductDescription>
        <ReservedCacheNodesOfferingId>649fd0c8-cf6d-47a0-bfa6-
060f8e75e95f</ReservedCacheNodesOfferingId>
        <State>payment-failed</State>
        <ReservedCacheNodeId>myreservationid</ReservedCacheNodeId>
        <CacheNodeCount>1</CacheNodeCount>
        <StartTime>2010-12-15T00:25:14.131Z</StartTime>
        <Duration>31536000</Duration>
        <FixedPrice>227.5</FixedPrice>
        <UsagePrice>0.046</UsagePrice>
        <CacheNodeType>cache.m1.small</CacheNodeType>
      </ReservedCacheNode>
      <ReservedCacheNode>

      (...output omitted...)

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

Some of the output has been omitted for brevity.

Using Amazon SNS Notifications with ElastiCache

You can configure ElastiCache to send notifications for important cache cluster events using Amazon Simple Notification Service (Amazon SNS). In these examples, you will configure a cache 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, please see the Amazon Simple Notification Service Developer Guide.

Adding an Amazon SNS Topic

The following procedures show you how to add an Amazon SNS topic for a cache cluster.

Note
This process can also be used to modify the Amazon SNS topic.

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the **Cache Clusters** list, click the **Modify** link next to the cache cluster to which you want to add an Amazon SNS topic ARN .

   The **Modify Cache Cluster** window appears.
3. Add the ARN of the Amazon SNS topic to the **Notification Topic ARN** text box.
4. Click the **Apply Immediately** check box.
5. Click the **Yes, Modify** button.

CLI

Use the command `elasticache-modify-cache-cluster` with the following parameters:

```
PROMPT> elasticache-modify-cache-cluster my-cache-cluster -t arn:aws:sns:us-
east-1:565419523791:ElastiCacheNotifications
```

This command produces output similar to the following:

```
CACHECLUSTER  my-cache-cluster  2013-07-26T01:21:46.607Z  cache.m1.large  mem
cached
available  3  us-east-1d  1.4.5
   SECGROUP  default  active
   PARAMGRP  default.memcached1.4  in-sync
   NOTIFICATION  arn:aws:sns:us-east-1:565419523791:ElastiCacheNotifications
   active
```

API

Call `ModifyCacheCluster` with the following parameters:

- *CacheClusterId* = my-cache-cluster
- *TopicArn* = arn:aws:sns:us-east-1:565419523791:ElastiCacheNotifications

Example

```
https://elasticache.amazon.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=false
    &CacheClusterId=my-cache-cluster
   &NotificationTopicArn=arn%3Aaws%3Asns%3Aus-east-1%3A565419523791%3AElastiC
acheNotifications
    &Version=2013-06-15
    &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>
```

Enabling and Disabling Amazon SNS Notifications

You can turn notifications on or off for a cache cluster. The following procedures show you how to disable Amazon SNS notifications.

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the **Cache Clusters** list, click the **Modify** link next to the cache cluster to which you want to add an Amazon SNS topic ARN .

   The **Modify Cache Cluster** window appears.
3. In the **Notification Topic Status** drop-down list box, select **inactive**.
4. Select the **Apply Immediately** check box.
5. Click the **Yes, Modify** button.

CLI

Use the command elasticache-modify-cache-cluster with the following parameters:

```
PROMPT> elasticache-modify-cache-cluster my-cache-cluster -ts inactive
```

This command produces output similar to the following:

```
CACHECLUSTER  my-cache-cluster  2013-07-26T01:21:46.607Z  cache.m1.large  mem
```

```
cached
available  3  us-east-1d  1.4.5
   SECGROUP  default  active
   PARAMGRP  default.memcached1.4  in-sync
   NOTIFICATION  arn:aws:sns:us-east-1:565419523791:ElastiCacheNotifications
   inactive
```

API

Call `ModifyCacheCluster` with the following parameters:

- *`CacheClusterId`* = `my-cache-cluster`
- *`NotificationTopicStatus`* = `inactive`

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=ModifyCacheCluster
    &ApplyImmediately=false
    &CacheClusterId=my-cache-cluster
    &NotificationTopicStatus=inactive
    &Version=2013-06-15
    &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>
```

Deleting a Cache Cluster

As soon as your cache cluster becomes available, you're billed for each hour or partial hour that you keep the cache cluster running (even if the cache cluster is idle). Once you've decided that you no longer need the cache cluster, you can delete it. Deleting a cache cluster requires you to identify the cache cluster you want to remove.

The following procedures show you how to delete a cache cluster.

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the **Cache Clusters** list, click the **Delete** link next to the cache cluster you wish to delete.

   The **Delete Cache Cluster** confirmation dialog box appears.
3. Click the **Yes, Delete** button.

CLI

Use the command `elasticache-delete-cache-cluster` to delete a cache cluster.

```
PROMPT> elasticache-delete-cache-cluster my-cache-cluster --force
```

This command will produce output similar to the following:

```
CACHECLUSTER  my-cache-cluster     https://console.aws.amazon.com/elastic
ache/home#client-download:   my-cache-cluster.q68zge.cfg.use1.cache.amazonaws.com
  11211    2013-07-22T20:29:54.663Z  cache.m1.large  memcached  deleting  3
us-east-1a  1.4.14
      CACHESECURITYGROUP  default  active
      CACHEPARAMETERGROUP  default.memcached1.4  in-sync
```

API

Call `DeleteCacheCluster` with the following parameter:

- *`CacheClusterId`* = `my-cache-cluster`

Example

```
https://elasticache.us-east-1.amazonaws.com/
 ?Action=DeleteCacheCluster
 &CacheClusterId=my-cache-cluster
    &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>
```

Managing Replication Groups

Topics

This section covers operations on creating and managing replication groups.

> **Note**
> At this time, replication groups are supported only for cache clusters that are running Redis.

Creating a Replication Group

A replication group begins with a primary cluster, a single-node read/write cache cluster that provides data to read replicas. You create the primary cluster as you would any other cache cluster. For more information, see Creating a Cache Cluster (p. 68).

> **Important**
> You cannot remove the primary cache node from a replication group. Although you can add or delete read replicas, or promote a read replica to the primary role, you cannot remove the current primary cache node without deleting the entire replication group.

After you have created the primary cache cluster, you can create the replication group.

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the **Replication Groups** list, click **Create Replication Group**.
3. On the **Create Replication Group** page, do the following. When all the settings are as you want them, click **Create:**

   - In the **Primary Cluster ID** box, click the ID of an existing cache cluster.
   - In the **Replication Group ID** box, type a name for the replication group.
   - In the **Replication Group Description** box, type a description for the replication group.

On the ElastiCache console, your new replication group is listed in the **Replication Groups** panel.

CLI

To create a replication group, use the `elasticache-create-replication-group` command. The following example creates a replication group named *my-repgroup* with a primary cache cluster named *my-redis-primary*.

```
PROMPT> elasticache-create-replication-group my-repgroup --primary-cluster-id
my-redis-primary --description "My replication group"
```

This command will produce output similar to the following:

```
REPLICATIONGROUP  my-repgroup  My replication group  creating
      CLUSTERID  my-redis-primary
```

API

To create a replication group, use the `CreateReplicationGroup` action with the following parameters:

- *`ReplicationGroupId`*
- *`ReplicationGroupDescription`*
- *`PrimaryClusterId`*

The following example creates a replication group named *my-repgroup* with a primary cache cluster named *my-redis-primary*.

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=CreateReplicationGroup
    &ReplicationGroupDescription=My%20replication%20group
    &ReplicationGroupId=my-repgroup
    &PrimaryClusterId=my-redis-primary
    &Version=2013-06-15
    &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>
```

Adding a Read Replica To A Replication Group

When the replication group has been created, you can add a read replica cache cluster to it. ElastiCache supports up to five read replicas per replication group, so you can repeat these steps as necessary.

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the **Replication Groups** list, click the replication group that you want to modify.
3. Click **Add Read Replica**.
4. In the **Add Read Replica to Replication Group** dialog box, do the following. When all the settings are as you want them, click **Add:**

    a. In the **Read Replica ID** box, type a name for your read replica.

    b. (Optional) In the **Availability Zone** box, click an Availability Zone. By default, ElastiCache will choose an Availability Zone for you.

ElastiCache will create your read replica cache cluster and then add it to the replication group. This process will take a few minutes. When the process is complete, the cache cluster replica will be displayed in the **Replication Groups** panel. Its current role will be listed as *read replica*.

CLI

To add a read replica cache cluster, use the `elasticache-create-cache-cluster` command with the *`--replication-group-id`* option. The following example creates a new cache cluster named *my-replica-1* and adds it to the *my-repgroup* replication group.

```
PROMPT> elasticache-create-cache-cluster my-replica-1 --replication-group-id
my-repgroup
```

This command will produce output similar to the following:

```
CACHECLUSTER  my-replica-1  https://console.aws.amazon.com/elasticache/home#cli
ent-download:  cache.m1.small  redis  creating  1  2.6.13  my-repgroup
      CACHESECURITYGROUP  default  active
      CACHEPARAMETERGROUP  default.redis2.6  in-sync
```

API

To add a read replica cache cluster, use the `CreateCacheCluster` action with the *`ReplicationGroupId`* parameter. The following example creates a new cache cluster named *my-replica-1* and adds it to the *my-repgroup* replication group.

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=CreateCacheCluster
    &CacheClusterId=my-replica-1
    &ReplicationGroupId=my-repgroup
    &Version=2013-06-15
    &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>
```

Promoting a Read Replica to the Primary Role

In a replication group, you can designate any read replica as the new primary cache cluster. The existing primary cluster becomes a read replica, and the read replica that you designate is promoted to the primary role. You might decide to do this for performance tuning reasons. For example, with a web application with heavy write activity, you can choose the node that has the lowest network latency or is "closest" to your application.

There is always a lag between the time that data is written to the primary cache cluster and when that data is written to a read replica cache cluster. In most cases, this lag time will be close to zero seconds; however, network latency can cause some read replicas in a replication group to fall farther behind. If you want to promote a read replica to the primary role, we recommend that you choose the read replica with the shortest lag time.

The AWS Management Console displays replication lag times for read replicas. Lag times are also published by CloudWatch using the *ReplicationLag* metric.

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the **Replication Groups** list, click **Promote** next to the read replica that you want to promote.
3. In the **Promote Read Replica** dialog box, click **Promote**.

CLI

To promote a read replica to the primary role, use the `elasticache-modify-replication-group` command with the `--primary-cluster-id` option. The following example promotes *my-replica-1* to primary status, and *my-redis-primary* becomes a read replica:

```
PROMPT> elasticache-modify-replication-group my-repgroup --primary-cluster-id
my-replica-1
```

API

To promote a read replica cache cluster to the primary role, use the `ModifyReplicationGroup` action with the `PrimaryClusterId` parameter. The following example promotes *my-replica-1* to primary status, and *my-redis-primary* becomes a read replica:

Example

```
https://elasticache.us-east-1.amazonaws.com/
   ?Action=ModifyReplicationGroup
   &ReplicationGroupId=my-repgroup
   &PrimaryClusterId=my-replica-1
   &Version=2013-06-15
    &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>
```

Deleting a Read Replica

If you no longer need a read replica cache cluster, you can delete it from the replication group. For instructions on deleting cache clusters, see Deleting a Cache Cluster (p. 83).

Note
You can add and delete read replicas; however, you cannot delete the primary cache cluster in a replication group.

Deleting a Replication Group

If you no longer need a replication group, you can delete it. When you delete a replication group, ElastiCache deletes all of the cache clusters in that group, including the primary cache cluster and any read replicas.

Once you have begun this operation, it cannot be interrupted.

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the **Replication Groups** list, click the replication group you want to delete.
3. Click **Delete**.
4. In the **Delete Replication Group** dialog box, click **Yes, Delete**.

CLI

Use the command `elasticache-delete-replication-group` to delete a replication group.

```
PROMPT> elasticache-delete-replication-group my-repgroup
```

You will be asked to confirm your decision; if you enter *y* (yes), the operation will begin immediately.

```
    Once you begin deleting this replication group, all of its clusters will
be
    deleted as well.
    Are you sure you want to delete this replication group? [Ny]y

REPLICATIONGROUP  my-repgroup  My replication group  deleting
```

API

Call `DeleteReplicationGroup` with the *`ReplicationGroup`* parameter.

Example

```
https://elasticache.us-east-1.amazonaws.com/
 ?Action=DeleteReplicationGroup
 &ReplicationGroupId=my-repgroup
    &Version=2013-06-15
    &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>
```

Note

If you set the *`RetainPrimaryCluster`* parameter to `true`, all of the read replicas will be deleted, but the primary cache cluster will be retained.

Managing Backup and Restore

Topics

This section covers backup and restore operations using cache cluster snapshots.

> **Note**
> At this time, backup and restore is supported only for cache clusters running Redis.

Enabling Automatic Snapshots on a New Cache Cluster

The following sections contain procedures showing you how to enable automatic snapshots when you create a new cache cluster.

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Click **Launch Cache Cluster**.
3. On the **Cache Cluster Details** page, do the following. When all the settings are as you want them, click **Next**:

   a. In the **Name** box, type the name of your cache cluster.

   b. In the **Node Type** box, choose one of the available cache node types, but not `cache.t1.micro`. (Backup and restore is not supported on `cache.t1.micro` cache nodes.)

      For a complete listing of cache node types and specifications, see Cache Node Type-Specific Parameters for Memcached or Cache Node Type-Specific Parameters for Redis and Amazon ElastiCache Product Features and Details.

   c. In the **Number of Nodes** box, type 1.

   d. In the **Engine** box, choose *redis*.

   e. (Optional) If you are creating a Redis cache cluster and want to seed it with data from a Redis snapshot file, type the Amazon S3 location for that snapshot into the **S3 Snapshot Location** text box. For more information on placing a Redis snapshot in Amazon S3, see Migrate Your Redis Cache to ElastiCache (p. 100)

   f. To change the port that your cache nodes will use to accept connections, in the **Cache Port** box, type the new port number.

   g. To send SNS notifications related to cache cluster events, in the **Topic for SNS notifications** box, click an existing Amazon SNS topic ID. If you have not configured Amazon SNS event notification, click **Disable Notifications**.

4. Leave the rest of the options on the **Cache Cluster Details** page at their default values, and click the **Next** button.
5. On the **Additional Configuration** page, leave the options at their default values, and click the **Next** button.

   On the next page, set **Enable Automatic Backups** to **Yes** button.

   In the **Backup Retention Period** list, choose the number of days that you want to retain your daily snapshots.

   If you want to specify a particular backup window, click **Select Window** and choose a time range.

   When you are ready to proceed, click **Next**
6. On the **Review** page, review the options for your cache cluster. If you need to correct any options, click the **Previous** button to return to previous panels and make corrections. If all your options are as you want them, click **Launch** to launch your new cache cluster.

CLI

To create a new cache cluster with automatic snapshots, use the `elasticache-create-cache-cluster` command. The following example creates a new Redis cache cluster named *my-redis-primary*. Automatic snapshots are enabled by setting `snapshot-retention-limit` to a non-zero value—in this example, the retention period is 5 days.

```
PROMPT> elasticache-create-cache-cluster --cache-cluster-id my-redis-primary --snapshot-retention-limit 5 --engine redis --cache-node-type cache.m1.small --num-cache-nodes 1
```

API

To create a new cache cluster with automatic snapshots, use the `CreateCacheCluster` action with the following parameters:

- *`CacheClusterId`*
- *`SnapshotRetentionLimit`*
- *`Engine`*
- *`CacheNodeType`*
- *`NumCacheNodes`*

The following example creates a new Redis cache cluster named *my-redis-primary*. Automatic snapshots are enabled by setting `SnapshotRetentionLimit` to a non-zero value—in this example, the retention period is 5 days.

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=CreateCacheCluster
    &CacheNodeType=cache.m1.small
    &SnapshotRetentionLimit=5
    &NumCacheNodes=1
    &CacheClusterId=my-redis-primary
    &Engine=redis
    &Version=2014-03-24
    &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>
```

Enabling Automatic Snapshots on an Existing Cache Cluster

The following sections contain procedures showing you how to enable automatic snapshots on an existing cache cluster.

AWS Management 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 ElastiCache Console Dashboard, click **Cache Clusters**.
3. In the list of cache clusters, find the cluster that you want to change and click **Modify**.
4. In the **Modify Replication Group** window, set **Enable Automatic Backups** to **Yes**.

   In the **Backup Retention Period** list, choose the number of days that you want to retain your daily snapshots.

   If you want to specify a particular backup window, click **Select Window** and choose a time range.
5. Click the **Modify** button.

CLI

To enable automatic snapshots on an existing cache cluster, use the `elasticache-modify-cache-cluster` command. The following example enables automatic snapshots by setting `snapshot-retention-limit` to a non-zero value—in this example, the retention period is 3 days.

```
PROMPT> elasticache-modify-cache-cluster --cache-cluster-id my-redis-primary --snapshot-retention-limit 3
```

API

To enable automatic snapshots on an existing cache cluster, use the `ModifyCacheCluster` action with the following parameters:

- *`CacheClusterId`*
- *`SnapshotRetentionLimit`*—must be set to a non-zero value to enable automatic snapshots.

The following example enables automatic snapshots by setting `SnapshotRetentionLimit` to a non-zero value—in this example, the retention period is 3 days.

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=ModifyCacheCluster
    &SnapshotRetentionLimit=3
    &CacheClusterId=my-redis-primary
    &Version=2014-03-24
    &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>
```

Enabling Automatic Snapshots on a Replication Group

For replication groups, you can designate one cache cluster in the group as the *snapshotting cluster*. ElastiCache will perform snapshots at this cluster only.

> **Note**
> In a replication group, we recommend that you do not take snapshots at the primary cache cluster. Instead, you should designate one of the read replicas as the snapshotting cluster. For more information, see Performance Impact of Snapshots (p. 25).

The following sections contain procedures showing you how to enable automatic snapshots on a read replica in a replication group.

AWS Management 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 ElastiCache Console Dashboard, click **Replication Groups**.
3. In the **Modify Replication Group** window, do the following:
   a. Set **Enable Automatic Backups** to **Yes**.
   b. In the **Cluster Id** list, choose one of the read replicas to act as the snapshotting cluster.

c. In the **Backup Retention Period** list, choose the number of days that you want to retain your daily snapshots.
d. If you want to specify a particular backup window, click **Select Window** and choose a time range.

When the settings are as you want them, click **Modify**.

CLI

To enable automatic snapshots on read replica in a replication group, use the `elasticache-modify-replication-group` command. The following example enables automatic snapshots by setting `snapshotting-cluster-id` to a read replica cache cluster name, and by setting `snapshot-retention-limit` to a non-zero value—in this example, the retention period is 2 days.

```
PROMPT> elasticache-modify-replication-group --replication-group-id my-regroup
 --snapshotting-cluster-id my-replica-1 --snapshot-retention-limit 2
```

API

To enable automatic snapshots on read replica in a replication group, use the `ModifyReplicationGroup` action with the following parameters:

- *`ReplicationGroupId`*
- *`SnapshottingClusterId`*
- *`SnapshotRetentionLimit`*—must be set to a non-zero value to enable automatic snapshots.

The following example enables automatic snapshots by setting `snapshotting-cluster-id` to a read replica cache cluster name, and by setting `snapshot-retention-limit` to a non-zero value—in this example, the retention period is 2 days.

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=ModifyReplicationGroup
    &ApplyImmediately=false
    &SnapshotRetentionLimit=2
    &SnapshottingClusterId=my-replica-1
    &ReplicationGroupId=myreplgroup
    &Version=2014-03-24
    &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>
```

Creating a Manual Snapshot

You can create a manual snapshot at any time. You must provide the name of the source cache cluster and a name for the snapshot.

The following procedures show you how to create a manual snapshot.

AWS Management 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 ElastiCache Console Dashboard, click **Snapshots**.
3. At the top of the screen, click **Create Snapshot**.
4. In the wizard's **Cache Cluster** field, choose the name of the source cache cluster from the drop-down list.
5. In the **Snapshot Name** field, type a name for the snapshot.
6. Click **Yes, Create**.

CLI

To create a manual snapshot, use the `elasticache-create-snapshot` command. The following example creates a snapshot of a cache cluster named *my-redis-primary*.

```
PROMPT> elasticache-create-snapshot my-manual-snapshot --cache-cluster-id my-redis-primary
```

API

To create a manual snapshot, use the `CreateSnapshot` action with the following parameters:

- *`CacheClusterId`*
- *`SnapshotName`*

The following example creates a manual snapshot named `my-manual-snapshot` from a cache cluster named `my-redis-primary`.

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=CreateSnapshot
    &CacheClusterId=my-redis-primary
    &SnapshotName=my-manual-snapshot
    &Version=2014-03-24
    &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>
```

Describing Snapshots

The following procedures show you how to describe your snapshots.

AWS Management 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 ElastiCache Console Dashboard, click **Snapshots**.
3. Use the **Filter** field to display manual, automatic, or all snapshots.

CLI

To describe snapshots, use the `elasticache-describe-snapshots` command. The following example describes all of the snapshots in the current AWS account.

```
PROMPT> elasticache-describe-snapshots
```

API

To describe snapshots, use the `DescribeSnapshots` action

The following example describes all of the snapshots in the current AWS account.

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=DescribeSnapshots
    &Version=2014-03-24
    &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>
```

Copying a Snapshot

You can make a copy of any snapshot, whether it was created automatically or manually. The following procedures show you how to copy a snapshot.

AWS Management 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 ElastiCache Console Dashboard, click **Snapshots**.
3. In the list of snapshots, choose the one that you want to copy and then click **Copy Snapshot**.
4. In the **New Cache Snapshot Identifier** field, type a name for your new snapshot and then click **Yes, Copy**.

CLI

To copy a snapshot, use the `elasticache-copy-snapshot` command. The following example makes a copy of an automatic snapshot.

```
PROMPT> elasticache-copy-snapshot automatic.my-redis-primary-2014-03-27-03-15
my-snapshot-copy
```

API

To copy a snapshot, use the `CopySnapshot` action with the following parameters:

- *`SourceSnapshotName`*
- *`TargetSnapshotName`*

The following example makes a copy of an automatic snapshot.

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=CopySnapshot
    &TargetSnapshotName=my-snapshot-copy
    &SourceSnapshotName=automatic.my-redis-primary-2014-03-27-03-15
    &Version=2014-03-24
    &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>
```

Restoring a Snapshot To a New Cache Cluster

To restore your data from a snapshot, ElastiCache must first create a new cache cluster, and then copy the data from the snapshot into the new cluster's Redis cache. When this process is complete, your applications can access the data in the new cluster.

The following procedures show you how to restore a snapshot to a new cache cluster.

AWS Management 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 ElastiCache Console Dashboard, click **Snapshots**.
3. In the list of snapshots, choose the one that you want to restore and then click **Restore Snapshot**.
4. In the **Restore Cache Cluster** window, go to the **Cache Cluster Id** field and type a name for the new cache cluster.
5. (Optional) You can customize cache cluster by choosing new values for **Instance Type**, **Auto Minor Version Upgrade**, **Cache Port**, and other properties.
6. When the settings are as you want them, click **Launch Cache Cluster**.

CLI

To restore data from a snapshot into a new cache cluster, use the `elasticache-create-cache-cluster` command with the `--snapshot-name` parameter. The following example creates a new cache cluster named *my-restored-redis* and restores the data from *my-manual-snapshot* into it.

```
PROMPT> elasticache-create-cache-cluster my-restored-redis --snapshot-name my-manual-snapshot
```

API

To restored data from a snapshot into a new cache cluster, use the `CreateSnapshot` action with the following parameter:

- *`SnapshotName`*

The following example creates a new cache cluster named *my-restored-redis* and restores the data from *my-manual-snapshot* into it.

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=CreateCacheCluster
    &SnapshotName=my-snapshot-copy
    &CacheClusterId=my-restored-redis
    &Version=2014-03-24
    &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>
```

Migrate Your Redis Cache to ElastiCache

When you create a new cache cluster, you can seed the cache with data from a Redis RDB snapshot file. This can be useful if you currently manage a Redis instance outside of ElastiCache and want to populate your new ElastiCache cache cluster with your existing Redis data.

> **Important**
> You must ensure that your Redis snapshot data does not exceed the resources of the cache node. For example, you would not be able to upload an RDS file with 2 GB of Redis data to a *cache.m1.small* node that has 1.3 GB of memory.
> If the snapshot is too large, the resulting cache cluster will have a status of `restore-failed`. You must delete the cache cluster and start over.
> For a complete listing of cache node types and specifications, see Cache Node Type-Specific Parameters for Memcached or Cache Node Type-Specific Parameters for Redis and Amazon ElastiCache Product Features and Details.

Create a Redis Snapshot

To create the Redis snapshot from which you will seed your ElastiCache Redis instance:

1. Connect to your existing Redis instance.
2. Run either the BGSAVE or SAVE command to create the snapshot. The snapshot will be written locally to a Redis database file (.RDB).

   BGSAVE is asynchronous and does not block other clients while processing. See http://redis.io/commands/bgsave.

   SAVE is synchronous and blocks other processes until finished. See http://redis.io/commands/save.

For additional information on creating a snapshot, see http://redis.io/topics/persistence.

Upload your Snapshot to Amazon S3

Once you have created the snapshot file, you will need to upload it into an Amazon S3 bucket. For more information on this task, see the Amazon Simple Storage Service Getting Started Guide.

The name of your Amazon S3 bucket must be DNS compliant; otherwise ElastiCache will not be able to access your snapshot file. The rules for DNS compliance are the following:

- 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:
  - Must start with a lowercase letter or a number.
  - Must end with a lowercase letter or a number.
  - Must contain only lowercase letters, numbers, and dashes.
- Names cannot be formatted as an IP address (e.g., 192.0.2.0).

For additional information, see Bucket Restrictions and Limitations in the *Amazon Simple Storage Service Developer Guide*.

Important
We strongly recommend that you use an Amazon S3 bucket that is in the same region as your ElastiCache cluster. This will ensure the highest data transfer speed between when ElastiCache reads your Redis snapshot from Amazon S3.
It is also important that you note the Amazon Resource Name (ARN) of where you saved the file in Amazon S3. You need this path to seed the new cluster with the data in this snapshot.

Grant ElastiCache Read Access to the RDB File

The next step is to grant ElastiCache read access to the snapshot file you copied to Amazon S3.

1. Sign in to the AWS Management Console and open the Amazon S3 console at https://console.aws.amazon.com/s3/.
2. Click **All Buckets**, then click the name of the Amazon S3 bucket that contains your RDB file.
3. Click the name of the folder that contains your RDB file.
4. Click the name of your RDB file, click the **Actions** drop-down menu, and then select **Properties**.
5. Click **Permissions** and then click **Add more permissions**.
6. In the **Grantee** field, type this email address: `aws-scs-s3-readonly@amazon.com`
7. Click **Open/Download** and then click **Save**.

Note
The aws-scs-s3-readonly account is used exclusively for customers uploading Redis snapshot data from Amazon S3.

Seed a Cache Cluster With RDB File Data

Now you are ready to create your new cache cluster. To do this, follow the instructions at Creating a Cache Cluster (p. 68). Pay special attention to steps 3d and 3e. In step 3d, be sure you select *redis* as the cache engine. In step 3e, which is not optional in our current scenario, enter the ARN for the snapshot you copied to your Amazon S3 bucket in the previous section. The ARN will look something like `arn:aws:s3:::my-bucket/my-folder/my-snapshot-filename`, where `my-bucket` is the name of the Amazon S3 bucket you have access to, `my-folder` is the folder you saved the .rdb file in, and `my-snapshot-filename` is the name of the .rdb backup/snapshot file. How you tell ElastiCache where

to find the Redis snapshot that you uploaded to Amazon S3 depends on the method you use to create the cache cluster:

- **AWS Management Console**

  If you use the AWS Management Console, enter the ARN (`arn:aws:s3:::my-bucket/my-folder/my-snapshot-filename`) in the **S3 Snapshot Location** textbox. The ARN must resolve to the snapshot file that you stored in Amazon S3.
- **CLI**

  If you use the `create-cache-cluster` command, use the `--snapshot-arns` parameter to specify a fully qualified ARN, such as `arn:aws:s3:::my-bucket/my-folder/my-snapshot-filename`. The ARN must resolve to the snapshot file that you stored in Amazon S3.
- **API**

  If you use the `CreateCacheCluster` API, use the `SnapshotArns` parameter to specify a fully qualified ARN such as `arn:aws:s3:::my-bucket/my-folder/my-snapshot-filename`. The ARN must resolve to the snapshot file that you stored in Amazon S3.

During the process of creating your cache cluster, the data in your Redis snapshot will be written to the cache cluster. You can monitor the progress of this operation by viewing the ElastiCache event messages. To do this, go to the ElastiCache console and click **Cache Events**. You can also use the ElastiCache command line interface or API to obtain event messages. For more information, see Viewing ElastiCache Events (p. 133).

Deleting Snapshots

You can delete any of your snapshots, automatic or manual, at any time. The following procedures show you how to delete a snapshot.

AWS Management 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 ElastiCache Console Dashboard, click **Snapshots**.
3. In the list of **Cache Snapshot Identifiers**, choose the one that you want to delete and then click **Delete Snapshot**.
4. In the **Delete Cache Snapshot** confirmation dialog box, click **Yes. Delete** to delete the snapshot, otherwise click **Cancel**.

CLI

To delete a snapshot, use the `elasticache-delete-snapshot` command. The following example deletes a snapshot.

```
PROMPT> elasticache-delete-snapshot my-manual-snapshot
```

Note
The `elasticache-delete-snapshot` command will prompt you for confirmation; enter *y* if you indeed want to delete the snapshot.

API

To delete a snapshot, use the `DeleteSnapshot` action with the following parameter:

- *`SnapshotName`*

The following example deletes a snapshot.

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=DeleteSnapshot
    &SnapshotName=my-manual-snapshot
    &Version=2014-03-24
    &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>
```

Managing Cache Parameter Groups

Topics

A cache parameter group is initially created with the default parameters for the cache engine used by the cache cluster. To provide custom values for any of the parameters, you must modify the group after creating it.

In this example, you create, list, modify, and examine cache parameter groups.

> **Caution**
> Improperly setting parameters in a cache parameter group can have unintended adverse effects, including degraded performance and system instability. Always exercise caution when you are modifying cache parameters or your cache parameter group.

> **Note**
> Some cache engine parameters are constrained or disabled in the context of an ElastiCache cache cluster. For more information, please see Cache Parameter Groups (p. 26).

Creating a Cache Parameter Group

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

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Click **Cache Parameter Groups** in the navigation list on the left side of the window.
3. Click the **Create Cache Parameter Group** button.

   The **Create Cache Parameter Group** window appears.
4. In the **Cache Parameter Group Family** drop-down list box, choose a family that is compatible with your cache engine. For example, a cache parameter group with a family of `memcached1.4` can only be used on cache clusters running Memcached.
5. Type the name of the new cache parameter group in the **Cache Parameter Group** text box.
6. Type a description for the new cache parameter group in the **Description** text box.
7. Click the **Yes, Create** button.

CLI

Use the command `elasticache-create-cache-parameter-group` with the following parameters:

```
PROMPT> elasticache-create-cache-parameter-group myCacheParameterGroup -fm
memecached1.4 -d "My new parameter group"
```

This command should produce output similar to the following:

```
CACHEPARAMETERGROUP  mycacheparametergroup  memcached1.4  My new parameter group
```

API

Call `CreateCacheParameterGroup` with the following parameters:

- *`CacheParameterGroupName`* = `myCacheparamgroup`
- *`Description`* = `"My new parameter group"`

Example

```
https://elasticache.us-east-1.amazonaws.com/
 ?Action=CreateCacheParameterGroup
    &Description=My%20new%20parameter%20group
 &CacheParameterGroupFamily=memcached1.4
 &CacheParameterGroupName=myCacheParameterGroup
 &Version=2013-06-15
 &Action=CreateCacheParameterGroup
    &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>
```

This command should return a response similar to the following:

```
<CreateCacheParameterGroupResponse xmlns="http://elasticache.amazon
aws.com/doc/2013-06-15/">
  <CreateCacheParameterGroupResult>
    <CacheParameterGroup>
      <CacheParameterGroupName>mycacheparametergroup</CacheParameterGroupName>

      <CacheParameterGroupFamily>memcached1.4</CacheParameterGroupFamily>
      <Description>My new parameter group</Description>
    </CacheParameterGroup>
  </CreateCacheParameterGroupResult>
  <ResponseMetadata>
    <RequestId>d8465952-af48-11e0-8d36-859edca6f4b8</RequestId>
  </ResponseMetadata>
</CreateCacheParameterGroupResponse>
```

Listing Available Cache Parameter Groups

You can see which cache parameter groups you've created for your AWS account by listing them. The following procedures show you how to list all available cache parameter groups.

Note
The `default.memcached1.4` parameter group family is automatically created the first time you create a cache cluster without specifying a custom cache parameter group.

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Click **Cache Parameter Groups** in the navigation list on the left side of the window.

   The available cache parameter groups appears in the **Cache Parameter Groups** list.

CLI

Use the command `elasticache-describe-cache-parameter-groups` to list all available cache parameter groups for your AWS account:

```
PROMPT> elasticache-describe-cache-parameter-groups
```

This command should produce output similar to the following:

```
CACHEPARAMETERGROUP  default.memcached1.4  memcached1.4  Default parameter group
for memcached1.4
```

API

Call `DescribeCacheParameterGroups` with no parameters.

Example

```
https://elasticache.us-east-1.amazonaws.com/
 ?Action=DescribeCacheParameterGroups
 &MaxRecords=100
 &Version=2013-06-15
    &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>
```

This command should return a response similar to the following:

```
<DescribeCacheParameterGroupsResponse xmlns="http://elasticache.amazon
aws.com/doc/2013-06-15/">
  <DescribeCacheParameterGroupsResult>
    <CacheParameterGroups>
      <CacheParameterGroup>
       <CacheParameterGroupName>default.memcached1.4</CacheParameterGroupName>

        <CacheParameterGroupFamily>memcached1.4</CacheParameterGroupFamily>
        <Description>Default parameter group for memcached1.4</Description>
      </CacheParameterGroup>
    </CacheParameterGroups>
  </DescribeCacheParameterGroupsResult>
  <ResponseMetadata>
    <RequestId>3540cc3d-af48-11e0-97f9-279771c4477e</RequestId>
  </ResponseMetadata>
</DescribeCacheParameterGroupsResponse>
```

Viewing Parameter Values for a Cache Parameter Group

You can get a detailed listing of all parameters in a cache parameter group and their values. The following procedures show you how to view the parameter values for a specific cache parameter group.

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Click **Cache Parameter Groups** in the navigation list on the left side of the window.

   The available cache parameter groups appears in the **Cache Parameter Groups** list.
3. Click on the cache parameter group that you want to see details for.

Details about the parameters in the selected cache parameter group are shown at the bottom of the console window. You can use the navigation buttons at the top right of the detail panel to scroll through the available parameters.

You can filter the parameters shown by selecting an option from the **Viewing** drop-down list box.

CLI

Use the command `elasticache-describe-cache-parameters` to view the parameter values for a specific cache parameter group:

```
PROMPT> elasticache-describe-cache-parameters myCacheParameterGroup --headers
```

This command should produce output similar to the following (the following example has been truncated):

```
CACHEPARAMETER  Parameter Name                 Parameter Value  Source  Data
Type  Is Modifiable  Minimum Version
CACHEPARAMETER  backlog_queue_limit           1024             system  integer
    false          1.4.5
CACHEPARAMETER  binding_protocol               auto             system  string
     false          1.4.5
CACHEPARAMETER  cas_disabled                  0                system  boolean
    true           1.4.5
CACHEPARAMETER  chunk_size                    48               system  integer
    true           1.4.5
CACHEPARAMETER  chunk_size_growth_factor       1.25             system  float
      true           1.4.5
CACHEPARAMETER  error_on_memory_exhausted      0                system  boolean
    true           1.4.5
CACHEPARAMETER  large_memory_pages            0                system  boolean
    false          1.4.5
(...sample truncated...)
```

API

Call `DescribeCacheParameters` with the following parameter:

- *`CacheParameterGroupName`* = `myCacheParameterGroup`

Example

```
https://elasticache.us-east-1.amazonaws.com/
 ?Action=DescribeCacheParameters
 &CacheParameterGroupName=mycacheparametergroup
 &MaxRecords=100
 &Version=2013-06-15
    &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>
```

This command should return a response similar to the following (the following example has been truncated):

```
<DescribeCacheParametersResponse xmlns="http://elasticache.amazon
aws.com/doc/2013-06-15/">
  <DescribeCacheParametersResult>
    <CacheClusterClassSpecificParameters>
      <CacheNodeTypeSpecificParameter>
        <DataType>integer</DataType>
        <Source>system</Source>
        <IsModifiable>false</IsModifiable>
        <Description>The maximum configurable amount of memory to use to store
 items, in megabytes.</Description>
        <CacheNodeTypeSpecificValues>
          <CacheNodeTypeSpecificValue>
            <Value>1000</Value>
            <CacheClusterClass>cache.c1.medium</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>6000</Value>
            <CacheClusterClass>cache.c1.xlarge</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>7100</Value>
            <CacheClusterClass>cache.m1.large</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>1300</Value>
            <CacheClusterClass>cache.m1.small</CacheClusterClass>
          </CacheNodeTypeSpecificValue>

...output omitted...

        </CacheNodeTypeSpecificValues>
        <AllowedValues>1-100000</AllowedValues>
        <ParameterName>max_cache_memory</ParameterName>
        <MinimumEngineVersion>1.4.5</MinimumEngineVersion>
      </CacheNodeTypeSpecificParameter>
      <CacheNodeTypeSpecificParameter>
```

```
        <DataType>integer</DataType>
        <Source>system</Source>
        <IsModifiable>false</IsModifiable>
        <Description>The number of memcached threads to use.</Description>
        <CacheNodeTypeSpecificValues>
          <CacheNodeTypeSpecificValue>
            <Value>2</Value>
            <CacheClusterClass>cache.c1.medium</CacheClusterClass>
          </CacheNodeTypeSpecificValue>
          <CacheNodeTypeSpecificValue>
            <Value>8</Value>
            <CacheClusterClass>cache.c1.xlarge</CacheClusterClass>
          </CacheNodeTypeSpecificValue>

...output omitted...

        </CacheNodeTypeSpecificValues>
        <AllowedValues>1-8</AllowedValues>
        <ParameterName>num_threads</ParameterName>
        <MinimumEngineVersion>1.4.5</MinimumEngineVersion>
      </CacheNodeTypeSpecificParameter>
    </CacheClusterClassSpecificParameters>
    <Parameters>
      <Parameter>
        <ParameterValue>1024</ParameterValue>
        <DataType>integer</DataType>
        <Source>system</Source>
        <IsModifiable>false</IsModifiable>
        <Description>The backlog queue limit.</Description>
        <AllowedValues>1-10000</AllowedValues>
        <ParameterName>backlog_queue_limit</ParameterName>
        <MinimumEngineVersion>1.4.5</MinimumEngineVersion>
      </Parameter>
      <Parameter>
        <ParameterValue>auto</ParameterValue>
        <DataType>string</DataType>
        <Source>system</Source>
        <IsModifiable>false</IsModifiable>
        <Description>Binding protocol.</Description>
        <AllowedValues>auto,binary,ascii</AllowedValues>
        <ParameterName>binding_protocol</ParameterName>
        <MinimumEngineVersion>1.4.5</MinimumEngineVersion>
      </Parameter>

...output omitted...

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

Modifying a Cache Parameter Group

You can modify parameters in a cache parameter group. These parameters are applied to cache clusters associated with the cache parameter group when the cache cluster is rebooted.

> **Note**
> The `default.memcached1.4` parameter group family is automatically created the first time you create a cache cluster without specifying a custom cache parameter group. You cannot modify the `default.memcached1.4` parameter group.

In this example, you modify a parameter in a cache parameter group.

AWS Management Console

The AWS Management Console does not support this functionality. Please refer to the command line interface example.

CLI

Use the command `elasticache-modify-cache-parameter-group` to modify a cache parameter group.

```
PROMPT> elasticache-modify-cache-parameter-group myCacheParameterGroup --para
meters "name=chunk_size, value=64,name=chunk_size_growth_factor,value=1.02"
```

API

Call `ModifyingCacheParameterGroup` with the following parameters:

- *`CacheParameterGroupName`* = `mycacheparametergroup`
- *`Parameters.member.1.ParameterName`* = `chunk_size`
- *`Parameters.member.1.ParameterValue`* = `64`
- *`Parameters.member.2.ParameterName`* = `chunk_size_growth_factor`
- *`Parameters.member.2.ParameterValue`* = `1.02`

Example

```
https://elasticache.us-east-1.amazonaws.com/
   ?Action=ModifyCacheParameterGroup
   &ParameterNameValues.member.1.ParameterName=chunk_size
   &ParameterNameValues.member.1.ParameterValue=64
   &ParameterNameValues.member.2.ParameterName=chunk_size_growth_factor
   &ParameterNameValues.member.2.ParameterValue=1.02
   &CacheParameterGroupName=mycacheparametergroup
   &Version=2013-06-15
    &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>
```

This command should return a response similar to the following:

```
<ModifyCacheParameterGroupResponse xmlns="http://elasticache.amazon
aws.com/doc/2013-06-15/">
  <ModifyCacheParameterGroupResult>
    <CacheParameterGroupName>mycacheparametergroup</CacheParameterGroupName>
  </ModifyCacheParameterGroupResult>
  <ResponseMetadata>
    <RequestId>fcedeef2-b7ff-11e0-9326-b7275b9d4a6c</RequestId>
  </ResponseMetadata>
</ModifyCacheParameterGroupResponse>
```

Using ElastiCache with Amazon Virtual Private Cloud (VPC)

The Amazon Virtual Private Cloud (VPC) service defines a virtual network that closely resembles a traditional data center. You can configure your VPC; you can select its IP address range, create subnets, and configure route tables, network gateways, and security settings. You can add a cache cluster to the virtual network, and you can control access to the cache cluster by using VPC security groups.

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

Note
ElastiCache is fully integrated with Amazon Virtual Private Cloud (VPC). If your AWS account supports only the EC2-VPC platform, we always create your cache cluster in a virtual private cloud (VPC).

- If you haven't used ElastiCache before, your cache clusters will be deployed into a VPC. A default VPC will be created for you automatically.
- If you have a default VPC and don't specify a subnet when you launch a cache cluster, the cluster is launched into your default VPC.
- You can launch cache clusters into your default VPC without needing to know anything about Amazon VPC. Your experience with clusters that you launch is the same whether you have a default VPC or not.

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

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

Creating a Virtual Private Cloud (VPC)

In this example, you will create a VPC with a private subnet for each Availability Zone.

AWS Management 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 VPC by using the Amazon Virtual Private Cloud wizard:
   a. In the navigation list, click **VPC Dashboard**.
   b. Click **Start VPC Wizard**.
   c. In the VPC wizard, click **VPC with Public and Private Subnets** and then click **Next**.

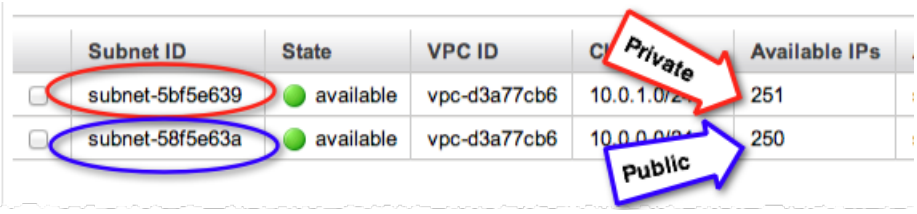
d. On the **VPC with Public and Private Subnets** page, keep the default options, and then click **Create VPC**.

e. In the confirmation message that appears, click **Close**.

3. Confirm that there are two subnets in your VPC, a public subnet and a private subnet. These subnets are created automatically.

   a. In the navigation list, click **Subnets**.

   b. In the list of subnets, find the two subnets that are in your VPC:



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

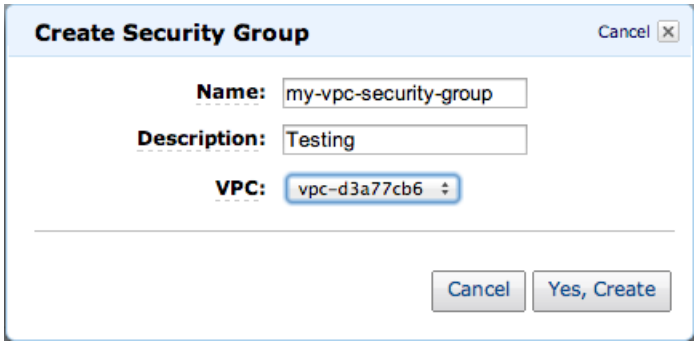
   Tip
   Make a note of your two subnet IDs, and which ones are public and private. You will need this information later when you launch your cache clusters and add an EC2 instance to your VPC.

4. Create a VPC security group. You will use this group for your cache cluster and your EC2 instance.

   a. In the navigation list, click **Security Groups**.

   b. Click **Create Security Group**.

   c. Enter a name and a description for your security group in the corresponding boxes. In the **VPC** box, click the identifier for your VPC.



   d. When the settings are as you want them, click **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 SSH.

   a. In the navigation list, click **Security Groups**.

   b. Find your security group in the list, and then click it.

c. Under **Security Group**, click the **Inbound** tab. In the **Create a new rule** box, click **SSH**, and then click **Add Rule**.
d. Click **Apply Rule Changes**.

Now you are ready to create a cache subnet group and launch a cache cluster in your VPC.

Creating a Cache Subnet Group

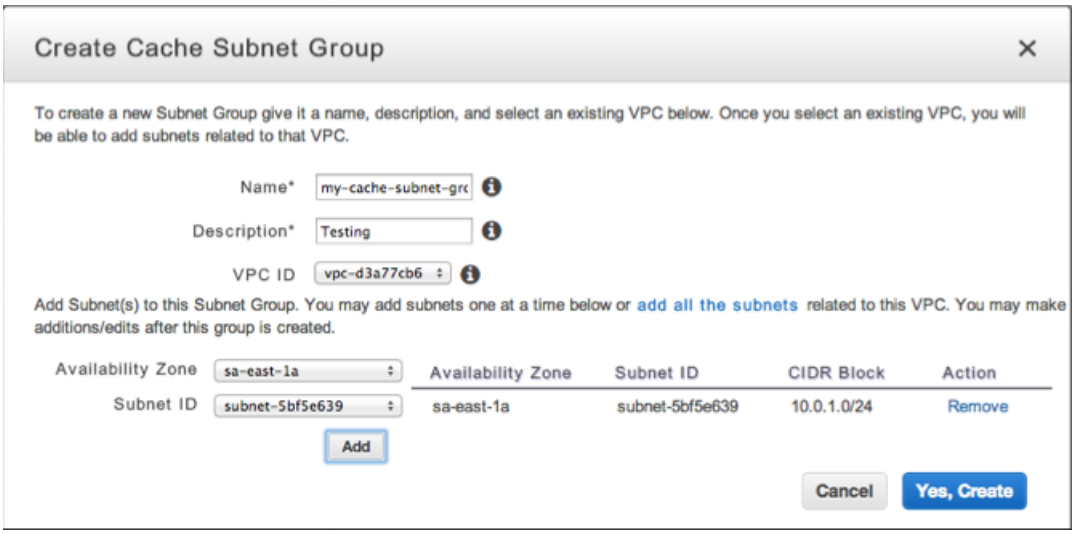
A *cache subnet group* is a collection of subnets that you may want to designate for your cache clusters in a VPC. When launching a cache cluster in a VPC, you will need to select a cache subnet group. ElastiCache then uses that cache subnet group to assign IP addresses within that subnet to each cache node in the cluster.

In this example, you will create a cache subnet group and add the private subnets from your VPC.

AWS Management Console

Create a Cache Subnet Group

1. Open the ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. In the navigation list, click **Cache Subnet Groups**.
3. Click **Create Cache Subnet Group**.
4. In the Create Cache Subnet Group, do the following. When all the settings are as you want them, click **Yes, Create**.
   a. In the **Name** box, type a name for your cache subnet group.
   b. In the **Description** box, type a description for your cache subnet group.
   c. In the **VPC ID** box, click the VPC that you created.
   d. In the **Availability Zone** and **Subnet ID** drop-down list boxes, click the Availability Zone and ID of your private subnet, and click **Add**.



5. In the confirmation message that appears, click **Close**.

Your new cache subnet group appears in the Cache Subnet Groups list of the ElastiCache console. You can click on the subnet group to see details, such as all of the subnets associated with this group, at the bottom of the window.

Now that you've created a cache subnet group, you can launch a cache cluster to run in your VPC.

Creating a Cache Cluster in a VPC

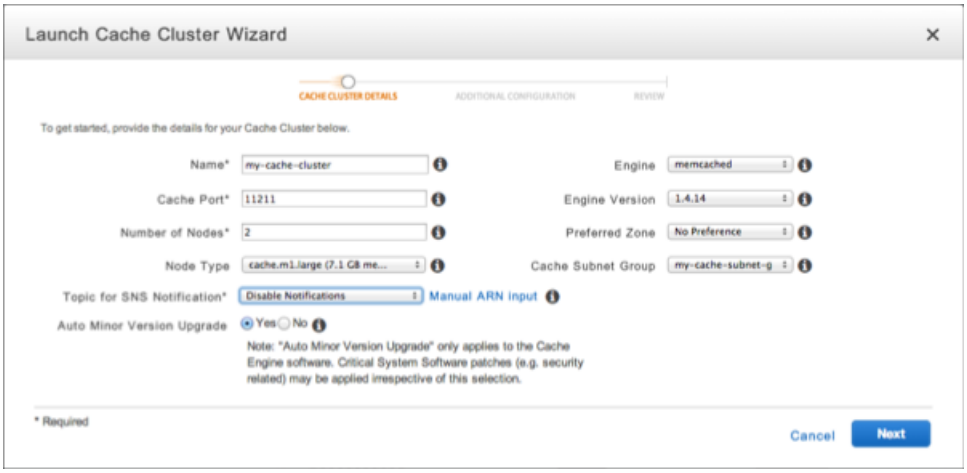
In this example, you will create a cache cluster in your VPC.

AWS Management Console

Create a Cache Cluster Inside an Amazon VPC

1. In the navigation list, click **Cache Clusters**.
2. Click **Launch Cache Cluster**.
3. On the **Cache Cluster Details** page, enter the following information:

   a. **Name**—Enter a name of your choice for the cluster.
   b. **Number of Nodes**—Enter the number of nodes the cluster will contain. For this example, use 2.
   c. **Cache Subnet Group**—Click the ID of your cache subnet group.
   d. **Topic for SNS notifications**—If your subnet group will send notifications, type the ARN for the topic where the message will be routed. For this example, choose **Disable Notifications**



4. When the settings are as you want them, click **Next**.
5. On the **Additional Configuration** page, ensure that your VPC security group appears in the list, and then click **Next**.
6. On the **Review** page, click **Launch Cache Cluster**.
7. In the confirmation message that appears, click **Close**.
8. When your cache cluster is up and running, take note of the node endpoints in the details pane; you will use these endpoints to connect to your cache nodes.

You have now launched a cache cluster inside a VPC. For an example of one way to connect to your new cache cluster running in the VPC, continue to Connecting to a Cache Cluster Running in a VPC (p. 117).

Connecting to a Cache Cluster Running in a VPC

This example shows how to launch an Amazon EC2 instance in your VPC. You can then log in to this instance and access the ElastiCache cluster that is running in the VPC.

AWS Management Console

In this example, you create an EC2 instance in your VPC. You can use this EC2 instance to connect to cache cluster nodes running in the VPC.

> **Note**
> For information about using Amazon EC2, go to the Amazon Elastic Compute Cloud Getting Started Guide in the Amazon EC2 documentation.

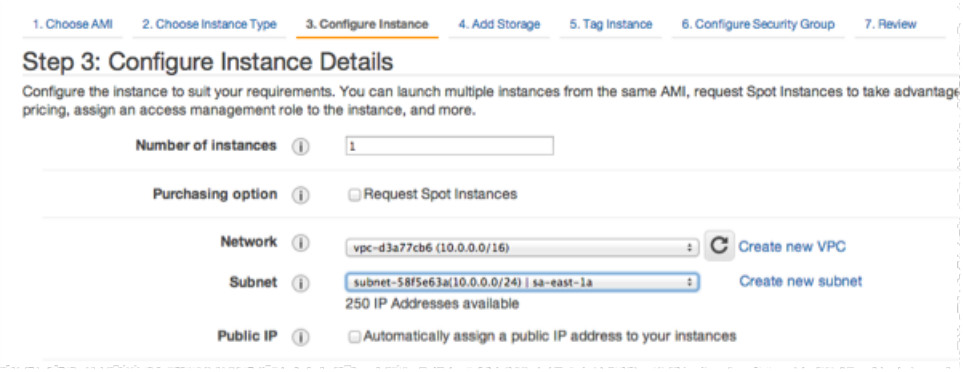
To create an EC2 instance in your VPC

Sign in to the AWS Management Console and open the Amazon EC2 console at https://console.aws.amazon.com/ec2/.

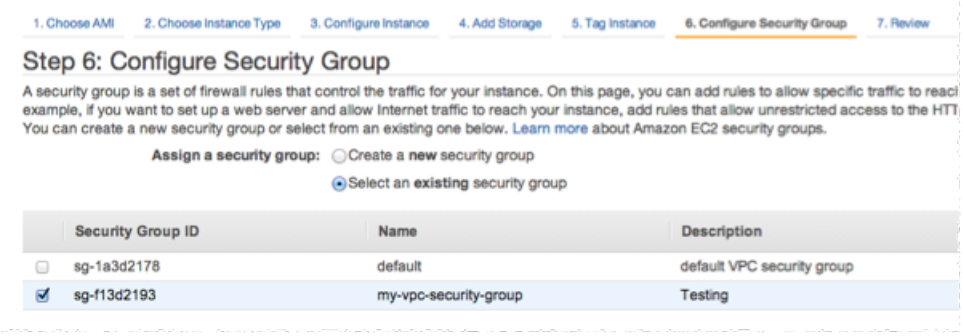
In the console, click **Launch Instance** and follow these steps:

1. On the **Choose an Amazon Machine Image (AMI)** page, choose the 64-bit Amazon Linux AMI, and then click **Select.**
2. On the **Choose an Instance Type** page, click **Next: Configure Instance Details**.
3. On the **Configure Instance Details** page, make the following selections:

   a. In the **Network** list, choose your VPC.
   b. In the **Subnet** list, choose your public subnet.



   When the settings are as you want them, click **Next: Add Storage**.
4. On the **Add Storage** Page, click **Next: Tag Instance**.
5. On the **Tag Instance** page, type a name for your Amazon EC2 instance, and then click **Next: Configure Security Group**.
6. On the **Configure Security Group** page, click **Select an existing security group**.



Select the name of your VPC security group, and then click **Review and Launch**.

7. On the **Review Instance and Launch** page, click **Launch**.

A new window appears: **Select an existing key pair or create a new key pair**. In this window, specify a key pair that you want to use with this instance.

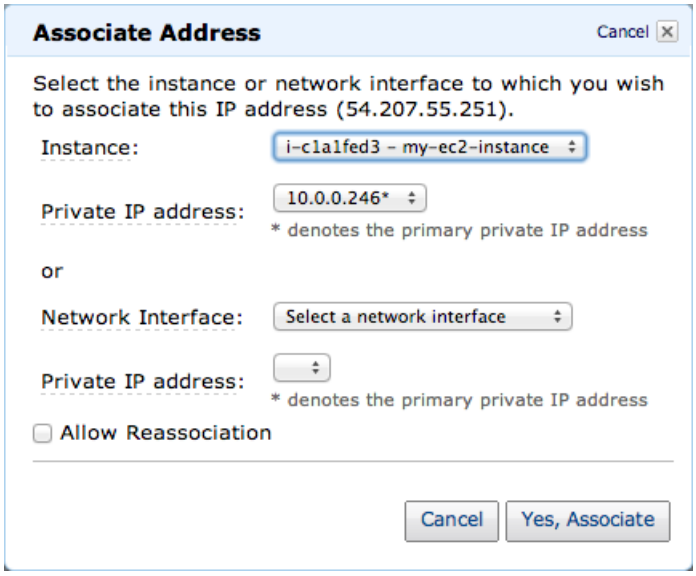
> **Note**
> For information about managing key pairs, go to the Amazon Elastic Compute Cloud Getting Started Guide.

When you are ready to launch your EC2 instance, click **Launch Instances**.

You can now assign an Elastic IP address to the EC2 instance that you just created. You will need to use this IP address to connect to the EC2 instance.

To assign an Elastic IP address

1. Open the Amazon VPC console at https://console.aws.amazon.com/vpc/.
2. In the navigation list, click **Elastic IPs**.
3. Click **Allocate New Address**.
4. In the **Allocate New Address** dialog box, in the **EIP used in** box, click **VPC**, and then click **Yes, Allocate**.
5. Select the Elastic IP address that you just allocated from the list and click **Associate Address**.
6. In the **Associate Address** dialog box, in the **Instance** box, click the ID of the EC2 instance that you launched, and then click **Yes, Associate**.



You can now use SSH to connect to the EC2 instance using the Elastic IP address that you created.

Tip
For instructions about using SSH to connect to a Linux/UNIX instance, go to Connect to Your Linux/UNIX Instance in the Amazon Elastic Compute Cloud Getting Started Guide.

To connect to your EC2 instance

1. Open a command window. At the command prompt, issue the following command. Replace *mykeypair.pem* with the name of your key pair file, and replace *54.207.55.251* with your Elastic IP address.

```
ssh -i mykeypair.pem ec2-user@54.207.55.251
```

2. Do not log out of your EC2 instance yet.

You are now ready to interact with your ElastiCache cluster. Before you can do that, you will need to install the *telnet* utility if you haven't already done so.

To install *telnet* and interact with your cache cluster

1. Open a command window. At the command prompt, issue the following command. At the confirmation prompt, enter *y*.

```
sudo yum install telnet

Loaded plugins: priorities, security, update-motd, upgrade-helper
Setting up Install Process
Resolving Dependencies
--> Running transaction check

...(output omitted)...
```

```
Total download size: 63 k
Installed size: 109 k
Is this ok [y/N]: y
Downloading Packages:
telnet-0.17-47.7.amzn1.x86_64.rpm                          |  63 kB     00:00


...(output omitted)...

Complete!
```

2. Go to the ElastiCache console at https://console.aws.amazon.com/elasticache/ and obtain the endpoint for one of the nodes in your cache cluster.
3. Use *telnet* to connect to your cache node endpoint over port 11211. Replace the hostname shown below with the hostname of your cache node.

```
telnet my-cache-cluster.7wufxa.0001.use1.cache.amazonaws.com 11211
```

You are now connected to the cache engine and can issue commands. In this example, you will add a data item to the cache and then get it immediately afterward. Finally, you'll disconnect from the cache node.

To store a key and a value, type the following two lines:

```
add mykey 0 3600 28
This is the value for my key
```

The cache engine responds with the following:

```
STORED
```

To retrieve the value for "mykey", type the following:

```
get mykey
```

The cache engine responds with the following:

```
VALUE mykey 0 28
This is the value for my key
END
```

To disconnect from the cache engine, type the following:

```
quit
```

Important
To avoid incurring additional charges on your AWS account, be sure to delete any AWS resources you no longer wish to use after trying these examples.

Managing Cache Subnet Groups

This section covers operations on cache subnet groups. Cache subnet groups are required for cache clusters running inside of an Amazon Virtual Private Cloud (VPC) environment.

Topics

Creating a Cache Subnet Group

The following procedures show you how to create a cache subnet group called *mycachesubnetgroup*.

AWS Management Console

1. From the ElastiCache Console, select **Cache Subnet Groups** from the navigation list on the left side of the console window.
2. Click **Create Cache Subnet Group**.
3. In the **Name** field, enter a name for your cache subnet group.
4. In the **Description** field, enter a short description for your cache subnet group.
5. In the **VPC ID** field, select the identifier for the VPC to be associated with this cache subnet group.
6. In the **Availability Zone** field, choose the Availability Zone to associate with this cache subnet group.
7. In the **Subnet ID** field, choose one of the subnets in your VPC and then click **Add**. You can repeat this step as many times as necessary to add more subnets to your cache subnet group.
8. Click the **Yes, Create** button. Your cache subnet group is now ready to use.

CLI

Use the command `elasticache-create-cache-subnet` to create a cache subnet group.

```
PROMPT> elasticache-create-cache-subnet-group mycachesubnetgroup --description
 "Testing" --subnet-ID-list subnet-53df9c3a
```

This command should produce output similar to the following:

```
SUBNETGROUP  mycachesubnetgroup  Testing  vpc-5a2e4c35
 SUBNET  subnet-53df9c3a  us-east-1b
```

API

Call `CreateCacheSubnetGroup` with the following parameters:

- *`CacheSubnetGroupName`* = `mycachesubnetgroup`
- *`CacheSubnetGroupDescription`* = `Testing`

- *SubnetIds.member.1* = subnet-53df9c3a

Example

```
https://elasticache.us-east-1.amazonaws.com/
    ?Action=CreateCacheSubnetGroup
    &CacheSubnetGroupName=mycachesubnetgroup
    &CacheSubnetGroupDescription=Testing
    &SubnetIds.member.1=subnet-53df9c3a
    &Version=2013-06-15
    &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>
```

Note
When you create a new cache subnet group, you should take note of the number of available IP addresses. If the subnet has very few free IP addresses, then you might be constrained as to how many more cache nodes you can add to the cache cluster. To resolve this issue, you can assign one or more subnets to a cache subnet group so that you have a sufficient number of IP addresses in your cluster's Availability Zone. After that, you will be able to add more cache nodes to your cluster.

Assigning a Cache Subnet Group to a Cache Cluster

Once you have created a cache subnet group, you can launch a cache cluster in an Amazon VPC. For more information, go to Creating a Cache Cluster in a VPC (p. 116)

Modifying a Cache Subnet Group

You can modify a cache subnet group's description, or modify the list of subnet IDs associated with the cache subnet group. You cannot delete a subnet ID from a cache subnet group if a cache cluster is currently using that subnet.

The following procedures show you how to modify a cache subnet group.

AWS Management Console

1. From the ElastiCache Console, select **Cache Security Groups** from the navigation list on the left side of the console window.
2. In the list of cache subnet groups, select the one you want to modify.
3. In the lower portion of the ElastiCache Console, make any changes to the description or the list of subnet IDs for the cache subnet group. To save your changes, click **Save**.

CLI

Use the command `elasticache-modify-cache-subnet-group` to modify a cache subnet group.

```
PROMPT> elasticache-modify-cache-subnet-group mycachesubnetgroup --description
 "New description" --subnet-ID-list subnet-42df9c3a,subnet-48fc21a9
```

This command should produce output similar to the following:

```
SUBNETGROUP  mycachesubnetgroup  Testing  vpc-5a2e4c35
 SUBNET  subnet-42df9c3a  us-east-1b
 SUBNET  subnet-48fc21a9  us-east-1b
```

API

Call `ModifyCacheSubnetGroup` with the following parameter:

- *`CacheSubnetGroupName`* = `mycachesubnetgroup`

Example

```
https://elasticache.us-east-1.amazonaws.com/
 ?Action=DeleteCacheSubnetGroup
 &CacheSubnetGroupName=mycachesubnetgroup
 &CacheSubnetGroupDescription=New%20description
 &SubnetIds.member.1=ssubnet-42df9c3a
 &SubnetIds.member.2=subnet-48fc21a9
 &Version=2013-06-15
    &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>
```

Note
When you create a new cache subnet group, you should take note of the number of available IP addresses. If the subnet has very few free IP addresses, then you might be constrained as to how many more cache nodes you can add to the cache cluster. To resolve this issue, you can assign one or more subnets to a cache subnet group so that you have a sufficient number of IP addresses in your cluster's Availability Zone. After that, you will be able to add more cache nodes to your cluster.

Deleting a Cache Subnet Group

If you decide that you no longer need your cache subnet group, you can delete it. You cannot delete a cache subnet group if it is currently in use by a cache cluster.

The following procedures show you how to delete a cache subnet group.

AWS Management Console

1. From the ElastiCache Console, select **Cache Security Groups** from the navigation list on the left side of the console window.
2. In the list of cache subnet groups, select the one you want to delete and then click **Delete**.
3. When you are asked to confirm this operation, click **Yes, Delete**.

CLI

Use the command `elasticache-delete-cache-subnet-group` to delete a cache subnet group.

```
PROMPT> elasticache-delete-cache-subnet-group mycachesubnetgroup
```

This command produces no output.

API

Call `DeleteCacheSubnetGroup` with the following parameter:

- *`CacheSubnetGroupName`* = `mycachesubnetgroup`

Example

```
https://elasticache.us-east-1.amazonaws.com/
 ?Action=DeleteCacheSubnetGroup
 &CacheSubnetGroupName=mycachesubnetgroup
 &Version=2013-06-15
    &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>
```

Managing Cache Security Groups

Note
Cache security groups are only applicable to cache clusters that are *not* running in an Amazon Virtual Private Cloud environment (VPC).

A cache security group allows you to control access to your cache clusters. A cache security group acts like a firewall controlling network access to your cache cluster. By default, network access is disabled for a new cache security group; you must specifically authorize access to an Amazon EC2 security group after the cache security group is created.

Note that Amazon EC2 instances running in a VPC are unable to connect to ElastiCache cache clusters in the AWS public cloud.

Creating a Cache Security Group

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

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

AWS Management Console

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Click **Cache Security Groups** in the navigation list on the left side of the window.
3. Click the **Create Cache Security Group** button.

   The **Create Cache Security Group** window appears.
4. Type the name of the new cache security group in the **Cache Security Group** text box.
5. Type a description for the new cache security group in the **Description** text box.
6. Click the **OK** button.

CLI

Use the command `elasticache-create-cache-security-group` with the following parameters:

```
PROMPT> elasticache-create-cache-security-group mycachesecuritygroup -d "My new
 security group"
```

API

Call `CreateCacheSecurityGroup` with the following parameters:

- *`CacheSecurityGroupName`* = `mycachesecuritygroup`
- *`Description`* = `"My new security group"`

Example

```
https://elasticache.us-east-1.amazonaws.com /
    ?Action=CreateCacheSecurityGroup
    &CacheSecurityGroupName=mycachesecuritygroup
    &Description=My%20cache%20security%20group
    &Version=2013-06-15
    &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>
```

Listing Available Cache Security Groups

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

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

AWS Management Console

1. Launch the AWS Management Console.

   a. Go to the AWS Management Console web page.
   b. Select **ElastiCache** from the drop-down list under the **Sign in to the AWS Console** button, and then click the **Sign in to the AWS Console** button.

2. Click **Cache Security Groups** in the navigation list on the left side of the window.

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

CLI

Use the command `elasticache-describe-cache-security-groups` to list all available cache security groups for your AWS account.

```
PROMPT> elasticache-describe-cache-security-groups
```

API

Call `DescribeCacheSecurityGroups` with no parameters.

Example

```
https://elasticache.us-east-1.amazonaws.com/
 ?Action=DescribeCacheSecurityGroups
 &MaxRecords=100
 &Version=2013-06-15
    &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>
```

Viewing a Cache Security Group

You can view detailed information about your cache security group.

The following procedures show you how to view the properties of a cache security group.

AWS Management Console

1. Launch the AWS Management Console.
   a. Go to the AWS Management Console web page.
   b. Select **ElastiCache** from the drop-down list under the **Sign in to the AWS Console** button, and then click the **Sign in to the AWS Console** button.
2. Click **Cache Security Groups** in the navigation list on the left side of the window.

   The available cache security groups appear in the **Cache Security Groups** list.
3. Select a cache security group from the **Cache Security Groups** list.

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

CLI

Use the `elasticache-describe-cache-security-groups` to view a cache security group.

```
PROMPT> elasticache-describe-cache-security-groups mycachesecuritygroup
```

API

Call `DescribeCacheSecurityGroups` with the following parameter:

- *`CacheSecurityGroupName`* = `mycachesecuritygroup`

Example

```
https://elasticache.amazonaws.com/
 ?Action=DescribeCacheSecurityGroups
 &CacheParameterGroupName=mycachesecuritygroup
 &Version=2013-06-15
    &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>
```

Authorizing Network Access to an Amazon EC2 Security Group

If you want to access your cache cluster from an 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 cache clusters from the EC2 instances belonging to the Amazon EC2 security group.

AWS Management Console

1. Launch the AWS Management Console.
   a. Go to the AWS Management Console web page.
   b. Select **ElastiCache** from the drop-down list under the **Sign in to the AWS Console** button, and then click the **Sign in to the AWS Console** button.
2. Select **Cache Security Groups** from the navigation list on the left side of the console window.
3. In the **Cache Security Groups** list, select the check box next to the cache security group that you want to grant access to.
4. At the bottom of the window, in the **EC2 Security Group Name** list, select your Amazon EC2 security group.
5. Click the **Add** button.

   Note
   It takes approximately one minute for changes to access permissions to take effect.

CLI

Use the command `elasticache-authorize-cache-security-group-ingress` to grant access to an Amazon EC2 security group

```
PROMPT> elasticache-authorize-cache-security-group-ingress default  --ec2-secur
ity-group-name myec2group --ec2-security-group-owner-id 987654321021
```

The command should produce output similar to the following:

```
SECGROUP  Name     Description
SECGROUP  default  default
      EC2-SECGROUP  myec2group  987654321021  authorizing
```

API

Call `AuthorizeCacheSecurityGroupIngress` with the following parameters:

- *`EC2SecurityGroupName`* = `myec2group`
- *`EC2SecurityGroupOwnerId`* = `987654321021`

Example

```
https://elasticache.us-east-1.amazonaws.com/
 ?Action=AuthorizeCacheSecurityGroupIngress
 &EC2SecurityGroupOwnerId=987654321021
 &EC2SecurityGroupName=myec2group
 &Version=2013-06-15
    &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>
```

Viewing 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.

AWS Management Console

To gather CPU utilization statistics for a cache cluster

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Select the cache nodes you want to view metrics for.
   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.

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 end end times are shown as examples only; you will need to substitute your own appropriate start and end times):

```
PROMPT> mon-get-stats CPUUtilization --dimensions="CacheClusterId=my
cachecluster,CacheNodeId=0002" --statistics=Average
--namespace="AWS/ElastiCache" --start-time 2013-07-05T00:00:00 --end-time
2013-07-06T00:00:00 --period=60
```

API

To gather CPU utilization statistics for a cache cluster

- Call the CloudWatch API `GetMetricStatistics` with the following parameters (note that the start end 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=2013-06-15
 &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>
```

Viewing ElastiCache Events

ElastiCache logs events that relate to your cache instances, cache security groups, and cache 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-describe-events command` or the `DescribeEvents` API.

The following procedures show you how to view all ElastiCache events for the past 24 hours (specified in seconds).

AWS Management Console

To view all ElastiCache instance events for the past 24 hours

1. Sign in to the AWS Management Console and open the Amazon ElastiCache console at https://console.aws.amazon.com/elasticache/.
2. Click **Cache Events** in the navigation list on the left side of the window.

   The available events appear in the **Cache Events** list.

   Note
   You can use the **Viewing** drop-down list box to filter the events by type.

CLI

To view all ElastiCache instance events for the past 24 hours

- Use the command `elasticache-describe-events` with the following parameters to view all ElastiCache events for the past 24 hours.

```
PROMPT> elasticache-describe-events --duration 1440
```

API

To view all ElastiCache instance events for the past 24 hours

- Call `DescribeEvents` with the following parameters:
  - *`Duration`* = `1440`

Example

```
https://elasticache.amazonaws.com/
 ?Action=DescribeEvents
 &Duration=1440
 &MaxRecords=100
 &Version=2013-06-15
    &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>
```

Configuring ElastiCache Clients

Topics

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

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

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`

Cache Node Endpoints and Port Numbers

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

AWS Management Console

To determine cache node endpoints and port numbers

1. Sign in to the Amazon ElastiCache Management Console and click **Cache Clusters**.
2. Click the name of your cache cluster.
3. Click the **Nodes** tab. All of the nodes in the cache cluster are displayed, along with the fully qualified DNS names and port numbers.

CLI

To determine cache node endpoints and port numbers

- Use the command `elasticache-describe-cache-nodes` with the following parameter:

```
PROMPT> elasticache-describe-cache-nodes --show-cache-node-info
```

This command should produce output similar to the following:

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

The fully qualified DNS names and port numbers are in the CACHENODE lines in the output.

API

To determine cache node endpoints and port numbers

- Call `DescribeCacheClusters` with the following parameter:

- *ShowCacheNodeInfo* = `true`

Example

```
https://elasticache.us-east-1.amazonaws.com /
    ?Action=DescribeCacheClusters
    &ShowCacheNodeInfo=true
    &Version=2013-06-15
    &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>
```

AutoDiscovery

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 Cache Node Auto Discovery (p. 47).

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

Connecting to Nodes in a Replication Group

Note
At this time, replication groups and read replicas are only supported for cache clusters running Redis.

For replication groups, 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 replication group. However, for write activity, we recommend that your applications connect to the primary endpoint for the replication group instead of connecting directly to the primary node. This will ensure that your applications can always find the current primary node, even if you decide to reconfigure your replication group by promoting a read replica to the primary role.

AWS Management Console

To determine endpoints and port numbers

1. Sign in to the Amazon ElastiCache Management Console and click **Cache Clusters**.
2. Click **Replication Group** and choose your replication group.
3. Click the **Node Groups** tab. All of the read replicas and the node group endpoint are displayed, with fully qualified DNS names and port numbers for each.

CLI

To determine cache node endpoints and port numbers

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

```
PROMPT> elasticache-describe-replication-groups my-repgroup
```

  This command should produce output similar to the following:

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

API

To determine cache node endpoints and port numbers

- Call `DescribeReplicationGroups` with the following parameter:
  - *`ReplicationGroupId`* = the name of your replication group.

  Example

```
https://elasticache.us-east-1.amazonaws.com /
    ?Action=DescribeCacheClusters
    &ReplicationGroupId=repgroup01
    &Version=2013-06-15
    &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 elasticache-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.

ElastiCache will ensure 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, and 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.

Using the ElastiCache API

Topics

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

For information about ElastiCache regions and endpoints, go to Regions and Endpoints in the Amazon Web Services General Reference.

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.

For example, the following is a sample request (linebreaks added for clarity).

```
https://elasticache.us-east-1.amazonaws.com/
?Action=DescribeCacheClusters
   &CacheClusterIdentifier=myCacheCluster
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-03-24
```

For the preceding query string, you would calculate the HMAC signature over the following string.

```
GET\n
elasticache.amazonaws.com\n
Action=DescribeCacheClusters
&CacheClusterIdentifier=myCacheCluster
&SignatureMethod=HmacSHA256
&SignatureVersion=4
&Version=2014-03-24
&X-Amz-Algorithm=AWS4-HMAC-SHA256
&X-Amz-Credential=AKIADQKE4SARGYLE%2F20140523%2Fus-east-1%2Felasticache%2Faws4_re
quest
&X-Amz-Date=20140520T223649Z
&X-Amz-SignedHeaders=content-type%3Bhost%3Buser-agent%3Bx-amz-content-sha256%3Bx-
amz-date
content-type:
host:elasticache.us-east-1.amazonaws.com
user-agent:CacheServicesAPICommand_Client
x-amz-content-sha256:
```

```
x-amz-date:
```

The result is the following signed request.

```
https://elasticache.us-east-1.amazonaws.com/
   ?Action=DescribeCacheClusters
   &CacheClusterIdentifier=myCacheCluster
   &SignatureMethod=HmacSHA256
   &SignatureVersion=4
   &Version=2014-03-24
   &X-Amz-Algorithm=AWS4-HMAC-SHA256
  &X-Amz-Credential=AKIADQKE4SARGYLE/20140520/us-east-1/elasticache/aws4_request

   &X-Amz-Date=20140520T223649Z
   &X-Amz-SignedHeaders=content-type;host;user-agent;x-amz-content-sha256;x-
amz-date
   &X-Amz-Signa
ture=2877960fced9040b41b4feaca835fd5cfeb9264f768e6a0236c9143f915ffa56
```

For detailed information on the signing process and calculating the request signature, see the topic Signature Version 4 Signing Process and its subtopics.

Available Libraries

AWS provides software development kits (SDKs) for software developers who prefer to build applications using language-specific APIs instead of the Query API. These SDKs provide basic functions (not included in the APIs), such as request authentication, request retries, and error handling so that it is easier to get started. SDKs and additional resources are available for the following programming languages:

- Java
- Windows and .NET
- PHP
- Python
- Ruby

For information about other languages, go to Sample Code & Libraries.

Troubleshooting Applications

ElastiCache provides specific and descriptive errors to help you troubleshoot problems while interacting with the ElastiCache API.

Retrieving Errors

Typically, you want your application to check whether a request generated an error before you spend any time processing results. The easiest way to find out if an error occurred is to look for an *Error* node in the response from the ElastiCache API.

XPath syntax provides a simple way to search for the presence of an *Error* node, as well as an easy way to retrieve the error code and message. The following code snippet uses Perl and the XML::XPath module to determine if an error occurred during a request. If an error occurred, the code prints the first error code and message in the response.

```
use XML::XPath;
                my $xp = XML::XPath->new(xml =>$response);
                if ( $xp->find("//Error") )
               {print "There was an error processing your request:\n", " Error
 code: ",
                $xp->findvalue("//Error[1]/Code"), "\n", " ",
                $xp->findvalue("//Error[1]/Message"), "\n\n"; }
```

Troubleshooting Tips

We recommend the following processes to diagnose and resolve problems with the ElastiCache API.

- Verify that ElastiCache is running correctly

  To do this, simply open a browser window and submit a query request to the ElastiCache service (such as https://elasticache.amazonaws.com). A MissingAuthenticationTokenException or 500 Internal Server Error confirms that the service is available and responding to requests.
- Check the structure of your request

  Each ElastiCache operation has a reference page in the *ElastiCache API Reference*. Double-check that you are using parameters correctly. In order to give you ideas regarding what might be wrong, look at the sample requests or user scenarios to see if those examples are doing similar operations.
- Check the forum

  ElastiCache has a discussion forum where you can search for solutions to problems others have experienced along the way. To view the forum, go to

  https://forums.aws.amazon.com/ .

Event Notifications and Amazon SNS

Topics

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*. Only one topic can be configured for ElastiCache notifications, and the AWS account that owns the Amazon SNS topic must be the same account that owns the cache cluster on which notifications are enabled.

The following ElastiCache events trigger Amazon SNS notifications:

Event Name	Description
ElastiCache:AddCacheNodeComplete	A cache node has been added to the cache cluster and is ready for use.
ElastiCache:AddCacheNodeFailed due to insufficient free IP addresses	A cache node could not be added because there are not enough available IP addresses.
ElastiCache:CacheClusterParametersChanged	One or more cache cluster parameters have been changed.
ElastiCache:CacheClusterProvisioningComplete	The provisioning of a cache cluster is completed, and the cache nodes in the cache cluster are ready to use.
ElastiCache:CacheClusterProvisioningFailed due to incompatible network state	An attempt was made to launch a new cache cluster into a nonexistent virtual private cloud (VPC).

Event Name	Description
ElastiCache:CacheClusterRestoreFailed	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 Migrate Your Redis Cache to ElastiCache (p. 100).
ElastiCache:CacheClusterSecurityGroupModified	One of the following events has occurred: • The list of cache security groups authorized for the cache cluster has been modified. • One or more new EC2 security groups have been authorized on any of the cache security groups associated with the cache cluster. • One or more EC2 security groups have been revoked from any of the cache security groups associated with the cache cluster.
ElastiCache:CacheNodeReplaceComplete	ElastiCache has detected that the host running a cache node is degraded or unreachable and has completed replacing the cache node. **Note** The DNS entry for the replaced cache node is not changed. In most instances, you do not need to refresh the server-list for your clients when this event occurs. However, some cache client libraries may stop using the cache node even after ElastiCache has replaced the cache node; in this case, the application should refresh the server-list when this event occurs.
ElastiCache:CacheNodesRebooted	One or more cache nodes has been rebooted.
ElastiCache:CustomerPendingSuspension	An operation could not be completed because the AWS account has been suspended.
ElastiCache:DeleteCacheClusterComplete	The deletion of a cache cluster and all associated cache nodes has completed.
ElastiCache:RemoveCacheNodeComplete	A cache node has been removed from the cache cluster.

For more information on using Amazon SNS notifications with ElastiCache, see Using Amazon SNS Notifications with ElastiCache (p. 81).

Controlling ElastiCache Access with IAM

Topics

ElastiCache allows you to control access to your cache clusters using cache security groups. A cache security group acts like a firewall controlling network access to your cache cluster.

> **Important**
> ElastiCache uses cache security groups to control who has access to specific ElastiCache cache clusters. There's no way in the IAM system to allow or deny access to a specific cache cluster.

For more information about using security groups with ElastiCache, refer to the Amazon ElastiCache User Guide.

About IAM

Amazon ElastiCache integrates with AWS Identity and Access Management (IAM), a service that enables you to do the following:

- Create users and groups under your AWS account
- Easily share your AWS resources between the users in your AWS account
- Assign unique security credentials to each user
- Control each user's access to services and resources
- Get a single bill for all users in your AWS account

For example, you can use IAM with ElastiCache to control which Users in your AWS Account can create or modify cache clusters for your AWS Account.

For more information about IAM, see the following:

- Identity and Access Management (IAM)
- IAM Getting Started Guide
- Using IAM

For more information on using IAM with ElastiCache, see Controlling ElastiCache Access with IAM (p. 146).

ElastiCache Security Groups and IAM

Using IAM with ElastiCache doesn't change how you use ElastiCache cache security groups to grant access to cache clusters. However, you can use IAM policies to specify which ElastiCache actions a User in your AWS Account can use with ElastiCache resources in general. Because you can't specify a particular cache cluster in the policy, you must specify * as the resource to indicate all cache clusters in the AWS Account.

Example

You could create a policy that gives the Developers group permission to use only these APIs: `CreateCacheCluster`, `DescribeCacheClusters`, `ModifyCacheCluster`, `RebootCacheCluster`, `DeleteCacheCluster`, `DescribeEvents`. They could then use those APIs with any cache cluster that belongs to your AWS Account.

For examples of IAM policies that cover ElastiCache actions, see Example Policies for ElastiCache (p. 148).

No ElastiCache ARNs

Because you can't specify a particular ElastiCache resource in an IAM policy, ElastiCache has no ARNs. When writing a policy to control access to ElastiCache actions, you use * as the resource. For more information about ARNs, go to ARNs in the Using IAM documentation.

ElastiCache Actions

In an IAM policy, you can specify any and all actions that ElastiCache offers. Each action name must be prefixed with the lowercase string `elastiCache:`. For example: `elastiCache:ModifyCacheCluster`, `elastiCache:DescribeCacheCluster`, or `elastiCache:*` (for all ElastiCache actions). For a list of the actions, refer to the Query API action names in the Amazon ElastiCache API Reference.

ElastiCache Keys

ElastiCache implements the following policy keys, but no others. For more information about policy keys, go to Condition in the in the Using IAM documentation.

AWS-Wide Policy Keys

- `aws:CurrentTime`–To check for date/time conditions.

- `aws:EpochTime`–To check for date/time conditions using a date in epoch or UNIX time.
- `aws:principaltype`–To check the type of principal (user, account, federated user, etc.) for the current request.
- `aws:SecureTransport`–To check whether the request was sent using SSL. For services that use only SSL, such as Amazon RDS and Amazon Route 53, the `aws:SecureTransport` key has no meaning.
- `aws:SourceArn`–To check the source of the request, using the Amazon Resource Name (ARN) of the source. (This value is available for only some services. For more information, see Amazon Resource Name (ARN) under "Element Descriptions" in the *Amazon Simple Queue Service Developer Guide*.)
- `aws:SourceIp`–To check the IP address of the requester. Note that if you use `aws:SourceIp`, and the request comes from an Amazon EC2 instance, the public IP address of the instance is evaluated.
- `aws:UserAgent`–To check the client application that made the request.
- `aws:userid`–To check the user ID of the requester.
- `aws:username`–To check the user name of the requester, if available.

Note
Key names are case sensitive.

Example Policies for ElastiCache

This section shows a few simple policies for controlling user access to Amazon ElastiCache.

Note
In the future, ElastiCache might add new actions that should logically be included in one of the following policies, based on the policy's stated goals.

Example 1: Allow a Network Admin group to only be able to access the APIs related to ElastiCache security groups

In this example, we create a policy that gives access to the relevant actions and attach it to the group. The resource is stated as "*", because you can't specify a particular ElastiCache resource in an IAM policy.

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

Example 2: Allow managers to only be able to list the current ElastiCache resources in the AWS Account

In this example, we create a policy that lets managers use the ElastiCache actions with `Describe` in the name.

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

Example 3: Allow a system administrator to access a select set of ElastiCache actions

In this example, we create a policy that gives access to the relevant actions for system administrators and attach it to the group. As with the other examples, the resource is stated as "*", because you can't specify a particular ElastiCache resource in an IAM policy.

```
{
   "Version": "2012-10-17",
   "Statement":[{
      "Effect":"Allow",
      "Action":["elasticache:ModifyCacheCluster",
                "elasticache:RebootCacheCluster",
                "elasticache:DescribeCacheClusters",
                "elasticache:DescribeEvents",
                "elasticache:ModifyCacheParameterGroup",
                "elasticache:DescribeCacheParameterGroups",
                "elasticache:DescribeCacheParameters",
                "elasticache:ResetCacheParameterGroup",
                "elasticache:DescribeEngineDefaultParameters"],
      "Resource":"*"
      }
   ]
}
```

Failure to Retrieve Account Attributes

Recent changes to ElastiCache may cause an error for some IAM users that were set up with permissions based on the ElastiCache Full Access policy templates. The error may display *"Failed to retrieve account attributes, certain console functions may be impaired."* shown at the top of the page or *"Error calling EC2.DescribeSecurityGroups"*. The error is caused by the console invoking actions that have not explicitly been given permissions in the ElastiCache Full Access policies.

In order to resolve this issue, your IAM administrator must update the IAM user's policy document to allow two additional Amazon EC2 actions: `ec2:DescribeAccountAttributes` and `ec2:DescribeSecurityGroups`. You must make this change for any IAM user or group that was assigned a policy that was based on the ElastiCache Full Access policy templates.

For example, the following code is the default policy document for the ElastiCache Full Access policy template.

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Action": [
                "elasticache:*",
                "ec2:DescribeAvailabilityZones",
                "ec2:DescribeVpcs",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:DescribeAlarms",
                "sns:ListTopics",
                "sns:ListSubscriptions"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

Add the two additional actions stated above to get the following policy document that will give permission to the console to invoke the needed actions.

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

For information about updating IAM policies, see Managing IAM Policies.

Appendix: Installing the ElastiCache Cluster Client for PHP

Topics

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 Cache Node Auto Discovery (p. 47). For sample PHP code to use the client, see Using the ElastiCache Cluster Client for PHP (p. 49).

Downloading the Installation Package

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

To determine the PHP version installed on your Amazon EC2 instance

- At the command prompt, run the following command:

```
php -v
```

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

```
PHP 5.4.10 (cli) (built: Jan 11 2013 14:48:57)
Copyright (c) 1997-2012 The PHP Group
Zend Engine v2.4.0, Copyright (c) 1998-2012 Zend Technologies
```

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

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

You are now ready to download the ElastiCache Cluster Client.

To download the ElastiCache Cluster Client for PHP

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

Installation Steps for New Users

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

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

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

3. Download the correct `php-memcached` package for your Amazon EC2 instance and PHP version. For more information, see Downloading the Installation Package (p. 151).
4. Install `php-memcached`. The URI should be the download path for the installation package:

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

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

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

   Note
   Please use the latest version of the install artifact.

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

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

To install on a Red Hat Enterprise Linux 6.5 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. 151).
4. Install `php-memcached`. The URI should be the download path for the installation package:

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

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

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

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

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

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

3. Download the correct `php-memcached` package for your Amazon EC2 instance and PHP version. For more information, see Downloading the Installation Package (p. 151).
4. Install `php-memcached`. The URI should be the download path for the installation package.

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

> **Note**
> This installation step installs the build artifact `amazon-elasticache-cluster-client.so` into the `/usr/lib/php5/20121212*` directory. Please verify the absolute path of the build artifact because it is needed by the next step.

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

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

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

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

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

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

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

3. Download the correct `php-memcached` package for your Amazon EC2 instance and PHP version. For more information, see Downloading the Installation Package (p. 151).
4. Install `php-memcached`. The URI should be the download path for the installation package.

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

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

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

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

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

To update the `php-memcached` installation

1. Remove the previous installation of the Memcached extension for PHP.
2. Install the new ElastiCache `php-memcached` extension as described previously in Installation Steps for New Users (p. 152).

Removing the PHP Cluster Client

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.

Document History

The following table describes the important changes to the documentation since the last release of the *Amazon ElastiCache User Guide*.

- **API version:** 2014-03-24
- **Latest documentation update:** June 23, 2014

Change	Description	Date Changed
New instance sizes supported	ElastiCache added support for additional General Purpose (M3) instances and Memory Optimized (R3) instances. For more information, see Cache Parameter Groups (p. 26).	July 1, 2014
PHP Auto Discovery	Added support for PHP version 5.5 Auto Discovery. For more information, see Appendix: Installing the ElastiCache Cluster Client for PHP (p. 151).	May 13, 2014
Backup and Restore for Redis Clusters	In this release, ElastiCache allows customers to create snapshots of their Redis clusters, and create new clusters using these snapshots. A snapshot is a backup copy of the cluster at a specific moment in time, and consists of cluster metadata and all of the data in the Redis cache. Snapshots are stored in Amazon S3, and customers can restore the data from a snapshot into a new cluster at any time. For more information, see Backup and Restore for Cache Clusters (p. 24) and Managing Backup and Restore (p. 91).	April 24, 2014
Redis 2.8.6	ElastiCache supports Redis 2.8.6, in addition to Redis 2.6.13. With Redis 2.8.6, customers can improve the resiliency and fault tolerance of read replicas, with support for partial resynchronization, and a user-defined minimum number of read replicas that must be available at all times. Redis 2.8.6 also offers full support for publish-and-subscribe, where clients can be notified of events that occur on the server.	March 13, 2014

Change	Description	Date Changed
Redis Cache Engine	ElastiCache offers Redis cache engine software, in addition to Memcached. Customers who currently use Redis can "seed" a new ElastiCache Redis cache cluster with their existing data from a Redis snapshot file, easing migration to a managed ElastiCache environment. To support Redis replication capabilities, the ElastiCache API now supports replication groups. Customers can create a replication group with a primary Redis cache node, and add one or more read replica nodes that automatically stay synchronized with cache data in the primary node. Read-intensive applications can be offloaded to a read replica, reducing the load on the primary node. Read replicas can also guard against data loss in the event of a primary cache node failure.	September 3, 2013
Support for Default Amazon Virtual Private Cloud (VPC)	In this release, ElastiCache is fully integrated with Amazon Virtual Private Cloud (VPC). For new customers, cache clusters are created in a VPC by default. For more information, see ElastiCache and Amazon Virtual Private Cloud (p. 42).	January 8, 2013
PHP Support for Cache Node Auto Discovery	The initial release of cache node auto discovery provided support for Java programs. In this release, ElastiCache brings cache node auto discovery support to PHP.	January 2, 2013
Support for Amazon Virtual Private Cloud (VPC)	In this release, ElastiCache clusters can be launched in Amazon Virtual Private Cloud (VPC). By default, new customers' cache clusters are created in a VPC automatically; existing customers can migrate to VPC at their own pace. For more information, see ElastiCache and Amazon Virtual Private Cloud (p. 42) and Using ElastiCache with Amazon Virtual Private Cloud (VPC) (p. 113).	December 20, 2012
Cache Node Auto Discovery and New Cache Engine Version	ElastiCache provides cache node auto discovery—the ability for client programs to automatically determine all of the cache nodes in a cluster, and to initiate and maintain connections to all of these nodes. This release also offers a new cache engine version: Memcached version 1.4.14. This new cache engine provides enhanced slab rebalancing capability, significant performance and scalability improvements, and several bug fixes. There are several new cache parameters that can be configured; for more information, see Cache Parameter Groups (p. 26).	November 28, 2012
New Cache Node Types	This release provides four additional cache node types.	November 13, 2012
Reserved Cache Nodes	This release adds support for reserved cache nodes.	April 5, 2012
New Guide	This is the first release of *Amazon ElastiCache User Guide*.	August 22, 2011

AWS Glossary

For the latest AWS terminology, see the AWS Glossary in the *AWS General Reference*.