



lab



lab title

Using Amazon ElastiCache Redis V1.02



Course title

**AWS Certified Solutions Architect
Associate**



Table of Contents

Contents

Table of Contents.....	1
About the Lab	2
Launch an ElastiCache Redis Cluster	3
Create Security Groups.....	3
Launch ElastiCache Cluster	4
Creating an EC2 Instance	8
Using ElastiCache Redis with the Redis CLI.....	10
Cleaning Up.....	13

Please note that AWS services change on a weekly basis and it is extremely important you check the version number on this document to ensure you have the latest version with any updates or corrections.

About the Lab

These lab notes are to support the instructional videos on Using AWS ElastiCache Redis in the BackSpace AWS Certified Solutions Architect course.

In this lab we will:

- Create an ElastiCache Redis cluster using the console.
- Connect to an ElastiCache Redis cluster from EC2 using the Redis CLI.
- Read and Write to an ElastiCache Redis cluster.

Please refer to the AWS JavaScript SDK documentation at:

<http://docs.aws.amazon.com/AWSJavaScriptSDK/latest/AWS/ElastiCache.html>

Please refer to the Redis command documentation at:

<http://redis.io/commands>

Please note that AWS services change on a weekly basis and it is extremely important you check the version number on this document to ensure you have the latest version with any updates or corrections.

▶ Launch an ElastiCache Redis Cluster

In this section we will create an ElastiCache Redis cluster using the console.

Create Security Groups

Go to the VPC console

Create a new security group in the default VPC and call it *LocalServerSG*

Create an inbound rule for SSH access from source *0.0.0.0/0* (if you have a static IP address you can enter that instead for more security)

Click **Save**

Type	Protocol	Port Range	Source	Description	Remove
SSH (22)	TCP (6)	22	0.0.0.0/0		

Create a new security group in the default VPC and call it *RedisSG*

Create Security Group

Name tag: RedisSG

Group name: RedisSG

Description: ElastiCache Redis Security Group

VPC: vpc-e4a1b39f | Default VPC

Cancel Yes, Create

Create a custom TCP rule for the ElastiCache Redis port 6379 and source *LocalServerSG* security group (type sg to get a drop-down list of security groups)

sg-0dc5c92ba6b033820 | RedisSG

Summary Inbound Rules Outbound Rules Tags

Cancel Save

Type	Protocol	Port Range	Source	Description
Custom TCP Rule	TCP (6)	6379	sg-04a7bf65bbc81d286 LocalServerSG	

Add another rule

Click Save

Launch ElastiCache Cluster

Go to the ElastiCache console.

Select *Subnet Groups*

ElastiCache Dashboard

- Cache Clusters
- Replication Groups
- Reserved Cache Nodes
- Snapshots
- Cache Parameter Groups
- Cache Subnet Groups
- Cache Events
- ElastiCache Cluster Client

Click Create Subnet Group

Give it a name and description

Select the default VPC and an AZ and subnet.

Click Add

Click Create

Create Subnet Group

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

Name*

Description*

VPC ID

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

Availability Zone	Subnet ID	CIDR Block	Action
us-east-1a	subnet-ec25f4b0	172.31.32.0/20	Remove

Click ElastiCache Dashboard

Click “Get Started Now”

Select Redis

Create your Amazon ElastiCache cluster

Cluster engine ☒ **Redis**
 In-memory data structure store used as database, cache and message broker. ElastiCache for Redis offers Multi-AZ with Auto-Failover and enhanced robustness.
☐ Cluster Mode enabled

☐ **Memcached**
 High-performance, distributed memory object caching system, intended for use in speeding up dynamic web applications.

Call the cluster *backspace-redis-lab*

Select the t2 micro *Node type*

Set *Number of replicas* to zero

Redis settings

Name	backspace-redis-lab	i
Engine version compatibility	4.0.10	i
Port	6379	i
Parameter group	default.redis4.0	i
Node type	cache.t2.micro (0.5 GiB)	i
Number of replicas	None	i

Click Next

Select your Subnet Group created previously

Select your RedisSG Security Group created previously

▼ Advanced Redis settings

Advanced settings have common defaults set to give you the fastest way to get started. You can modify these now or after your cluster has been created.

Subnet group	backspace-lab-subnet-group (vpc-e4a1b39f)	i
Preferred availability zone(s)	<input checked="" type="radio"/> No preference <input type="radio"/> Select zones	i

Security

Security groups	RedisSG (sg-0dc5c92ba6b033820)	i
Encryption at-rest	<input type="checkbox"/>	i
Encryption in-transit	<input type="checkbox"/>	i

Click create

Import data to cluster

Seed RDB file S3 location ⓘ
Use comma to separate multiple paths in the field

Backup

Enable automatic backups ☐ ⓘ

Maintenance

Maintenance window ☒ No preference ⓘ
☐ Specify maintenance window

Topic for SNS notification ⓘ

Cancel **Create**

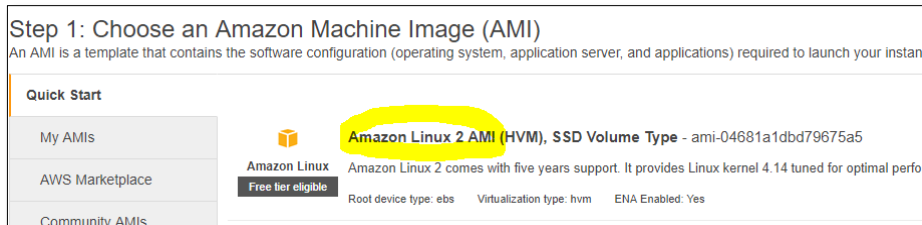
Refresh view and wait until status becomes *available*

Create	Backup	Reboot	Delete	Modify					
Filter: Search Clusters...					1 to 1 of 1 Clusters				
	Cluster Name	Mode	Shards	Nodes	Node Type	Status	Encryption in-transit	Encryption at-rest	
	backspace-redis-lab	Redis	0	1 node	cache.t2.micro	available	No	No	

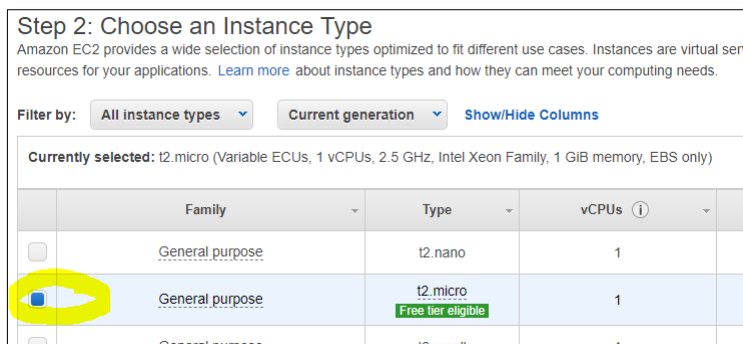
▶ Creating an EC2 Instance

In this section we will create an EC2 instance to connect to our ElastiCache cluster.

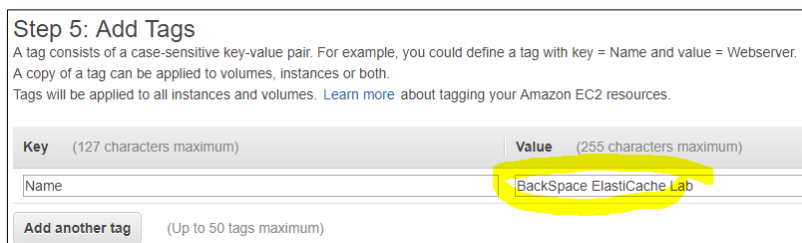
From the EC2 console create an Amazon Linux 2 instance



Select *t2.micro*



Add a Name tag



Select the *LocalServerSG* Security Group

Click *Review and Launch*

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server, you can allow HTTP and HTTPS traffic. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☐ Create a new security group

☒ Select an existing security group

Security Group ID	Name	Description
<input type="checkbox"/> sg-0e1da50c6ddd47869	aws-cloud9-BackSpace-ElastiCache-Lab-9d4ae658c56b401d9182c99daa1e655d-InstanceSecurityGroup-YDXBO0ZTWP13	Security group for AWS Cloud9 environment aw
<input type="checkbox"/> sg-7d1df536	default	default VPC security group
<input checked="" type="checkbox"/> sg-04a7bf65bbc81d286	LocalServerSG	Local EC2 Server Default VPC Security Group
<input type="checkbox"/> sg-0dc5c92ba6b033820	RedisSG	ElastiCache Redis Security Group

Click *Launch*

Select a key pair

Click *Launch instance*

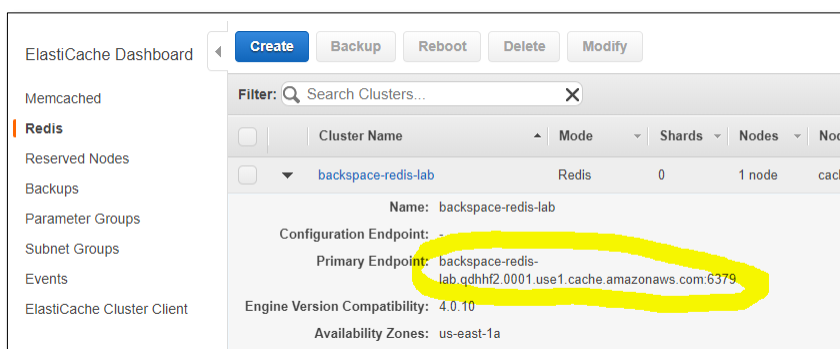
▶ Using ElastiCache Redis with the Redis CLI

In this section we will read and write to an ElastiCache Redis cluster using the Redis Command Line Interface (CLI).

From the console go to Cache Clusters

Click on the Cluster Node in your Cache Cluster

Copy the endpoint and the port, we will need this to connect to the node.



Connect into your Amazon Linux 2 EC2 instance using Bash (Windows) or Terminal (Mac)

```
ec2-user@ip-172-31-44-40:~
paulp@DESKTOP-7SKRN08 MINGW64 /d/OneDrive/Documents/KeyPairs/BackSpace-Labs
$ ssh -i "pcoady.pem" ec2-user@ec2-18-232-165-239.compute-1.amazonaws.com
The authenticity of host 'ec2-18-232-165-239.compute-1.amazonaws.com (18.232.165.239)' can't be established.
ECDSA key fingerprint is SHA256:87XTVbX670axRzREKlAMjXEIwipLmySGbMkQW24FRII.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-18-232-165-239.compute-1.amazonaws.com,18.232.165.239' (ECDSA) to the list of known hosts.

 _ _ | _ _ | )
 _ | ( _ /   Amazon Linux 2 AMI
 _ | \ _ | _ |

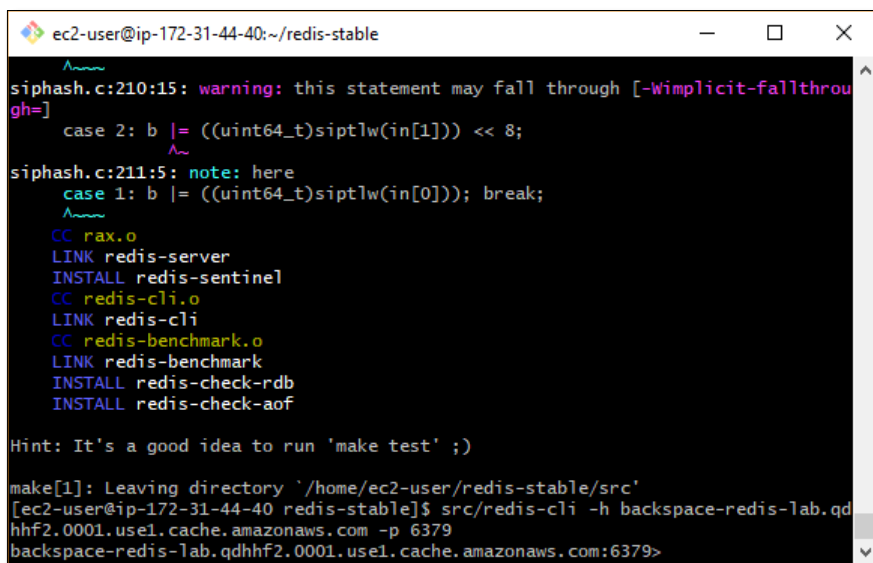
https://aws.amazon.com/amazon-linux-2/
No packages needed for security; 1 packages available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-44-40 ~]$
```

Input the following commands to install GCC and the Redis-CLI utility:

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

Now connect to your Redis cluster your endpoint domain (without the :6379)

```
src/redis-cli -h YOUR_ENDPOINT_GOES_HERE -p 6379
```



The screenshot shows a terminal window titled 'ec2-user@ip-172-31-44-40:~/redis-stable'. It displays the output of the 'make' command, which includes compilation warnings and notes from the siphash.c file, followed by the linking and installation of various Redis components: redis-server, redis-sentinel, redis-cli, redis-benchmark, redis-check-rdb, and redis-check-aof. A hint suggests running 'make test'. The terminal then shows the user running the command 'src/redis-cli -h backspace-redis-lab.qdhhf2.0001.usel.cache.amazonaws.com -p 6379', which results in a connection to the Redis cluster.

```
ec2-user@ip-172-31-44-40:~/redis-stable
siphash.c:210:15: warning: this statement may fall through [-Wimplicit-fallthrough]
gh=]
  case 2: b |= ((uint64_t)siptrw(in[1])) << 8;
  ^
siphash.c:211:5: note: here
  case 1: b |= ((uint64_t)siptrw(in[0])); break;
  ^
CC rax.o
LINK redis-server
INSTALL redis-sentinel
CC redis-cli.o
LINK redis-cli
CC redis-benchmark.o
LINK redis-benchmark
INSTALL redis-check-rdb
INSTALL redis-check-aof

Hint: It's a good idea to run 'make test' ;)

make[1]: Leaving directory '/home/ec2-user/redis-stable/src'
[ec2-user@ip-172-31-44-40 redis-stable]$ src/redis-cli -h backspace-redis-lab.qdhhf2.0001.usel.cache.amazonaws.com -p 6379
backspace-redis-lab.qdhhf2.0001.usel.cache.amazonaws.com:6379>
```

Now run a command to set a key myHighScore to 1000:

```
set myHighScore 1000
```

```

ec2-user@ip-172-31-44-40:~/redis-stable
case 2: b |= ((uint64_t)sip1w(in[1])) << 8;
^
siphash.c:211:5: note: here
case 1: b |= ((uint64_t)sip1w(in[0])); break;
^
CC rax.o
LINK redis-server
INSTALL redis-sentinel
CC redis-cli.o
LINK redis-cli
CC redis-benchmark.o
LINK redis-benchmark
INSTALL redis-check-rdb
INSTALL redis-check-aof

Hint: It's a good idea to run 'make test' ;)

make[1]: Leaving directory '/home/ec2-user/redis-stable/src'
[ec2-user@ip-172-31-44-40 redis-stable]$ src/redis-cli -h backspace-redis-lab.qd
hhf2.0001.use1.cache.amazonaws.com -p 6379
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> set myHighScore 1
000
OK
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379>

```

Now read the key:

```
get myHighScore
```

```

ec2-user@ip-172-31-44-40:~/redis-stable
siphash.c:211:5: note: here
case 1: b |= ((uint64_t)sip1w(in[0])); break;
^
CC rax.o
LINK redis-server
INSTALL redis-sentinel
CC redis-cli.o
LINK redis-cli
CC redis-benchmark.o
LINK redis-benchmark
INSTALL redis-check-rdb
INSTALL redis-check-aof

Hint: It's a good idea to run 'make test' ;)

make[1]: Leaving directory '/home/ec2-user/redis-stable/src'
[ec2-user@ip-172-31-44-40 redis-stable]$ src/redis-cli -h backspace-redis-lab.qd
hhf2.0001.use1.cache.amazonaws.com -p 6379
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> set myHighScore 1
000
OK
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> get myHighScore
"1000"
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> |

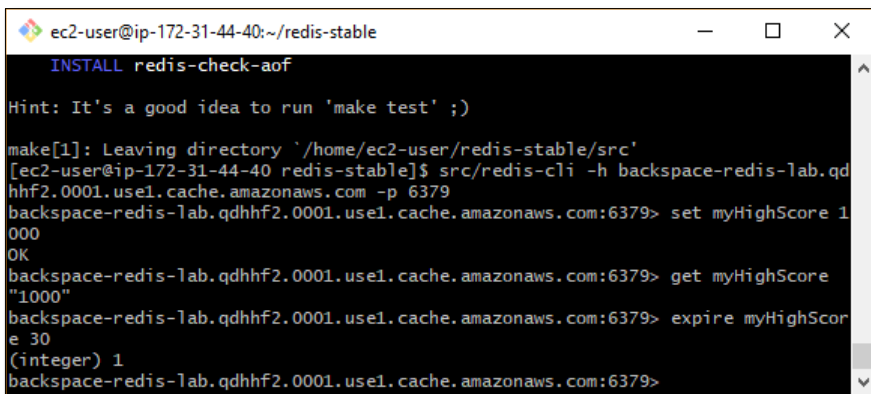
```

Now set an expiry time of 30s for the key:

```
expire myHighScore 30
```

It will return:

- 1 if the timeout was set.
- 0 if key does not exist or the timeout could not be set.



```

ec2-user@ip-172-31-44-40:~/redis-stable
INSTALL redis-check-aof

Hint: It's a good idea to run 'make test' ;)

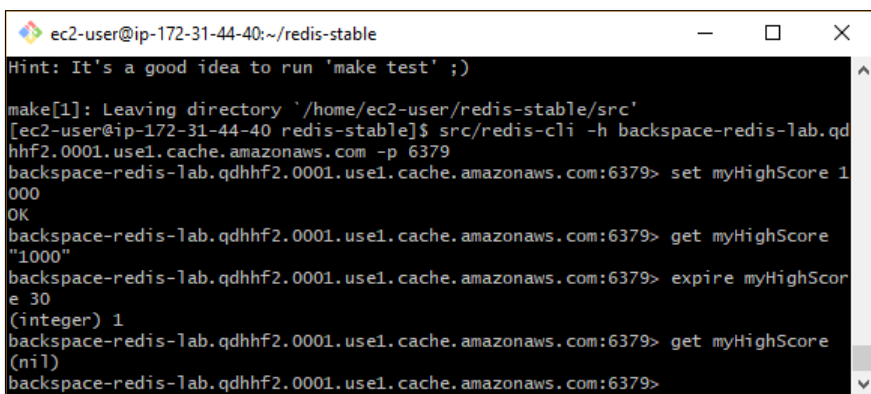
make[1]: Leaving directory '/home/ec2-user/redis-stable/src'
[ec2-user@ip-172-31-44-40 redis-stable]$ src/redis-cli -h backspace-redis-lab.qd
hhf2.0001.use1.cache.amazonaws.com -p 6379
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> set myHighScore 1
000
OK
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> get myHighScore
"1000"
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> expire myHighScor
e 30
(integer) 1
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379>

```

Now wait 30 s and read the key:

```
get myHighScore
```

If 30s has past it will return (nil)



```

ec2-user@ip-172-31-44-40:~/redis-stable
Hint: It's a good idea to run 'make test' ;)

make[1]: Leaving directory '/home/ec2-user/redis-stable/src'
[ec2-user@ip-172-31-44-40 redis-stable]$ src/redis-cli -h backspace-redis-lab.qd
hhf2.0001.use1.cache.amazonaws.com -p 6379
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> set myHighScore 1
000
OK
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> get myHighScore
"1000"
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> expire myHighScor
e 30
(integer) 1
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> get myHighScore
(nil)
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379>

```

Type *Exit* to disconnect



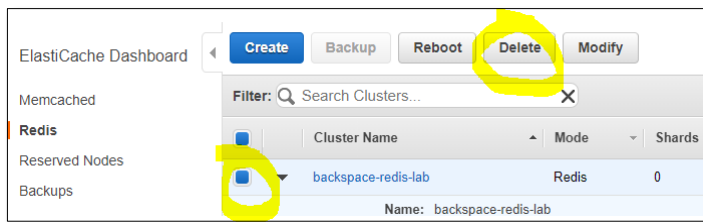
```

ec2-user@ip-172-31-44-40:~/redis-stable
make[1]: Leaving directory '/home/ec2-user/redis-stable/src'
[ec2-user@ip-172-31-44-40 redis-stable]$ src/redis-cli -h backspace-redis-lab.qd
hhf2.0001.use1.cache.amazonaws.com -p 6379
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> set myHighScore 1
000
OK
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> get myHighScore
"1000"
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> expire myHighScor
e 30
(integer) 1
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> get myHighScore
(nil)
backspace-redis-lab.qdhhf2.0001.use1.cache.amazonaws.com:6379> exit
[ec2-user@ip-172-31-44-40 redis-stable]$

```

Cleaning Up

Delete your cluster in the ElastiCache console.



Terminate your EC2 instance in the EC2 console

