



lab



lab title

# Programming AWS ElastiCache Redis using the Redis API Client

V1.03



Course title

BackSpace Academy  
AWS Certification Preparation



# Table of Contents

## Contents

Table of Contents .....	1
About the Lab .....	2
Launch an ElastiCache Redis Cluster .....	3
Setting up AWS Cloud9 .....	3
Creating the Redis Security Group .....	4
Creating an ElastiCache Subnet Group .....	8
Launching an ElastiCache Cluster .....	9
Connect to an ElastiCache Redis Cluster using NodeJS .....	12
Using ElastiCache Redis with NodeJS .....	15
Cleaning Up .....	20
Connect to an ElastiCache Redis Cluster using Python .....	22
Using ElastiCache Redis with Python .....	26
Cleaning Up .....	32

## About the Lab

**Please note that not all AWS services are supported in all regions. Please use the US-East-1 (North Virginia) region for this lab.**

These lab notes are to support the instructional videos on Programming AWS ElastiCache Redis using NodeJS in the BackSpace AWS Certified Developer course.

In this lab we will:

- Create an ElastiCache Redis cluster using the console.
- Connect to an ElastiCache Redis cluster using the AWS NodeJS SDK.
- Read and Write to an ElastiCache Redis cluster using the AWS NodeJS SDK.

Please refer to the Redis documentation at:

<https://redis.io/commands>

<https://github.com/NodeRedis/node-redis>

<https://github.com/andymccurdy/redis-py>

**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..

## Setting up AWS Cloud9

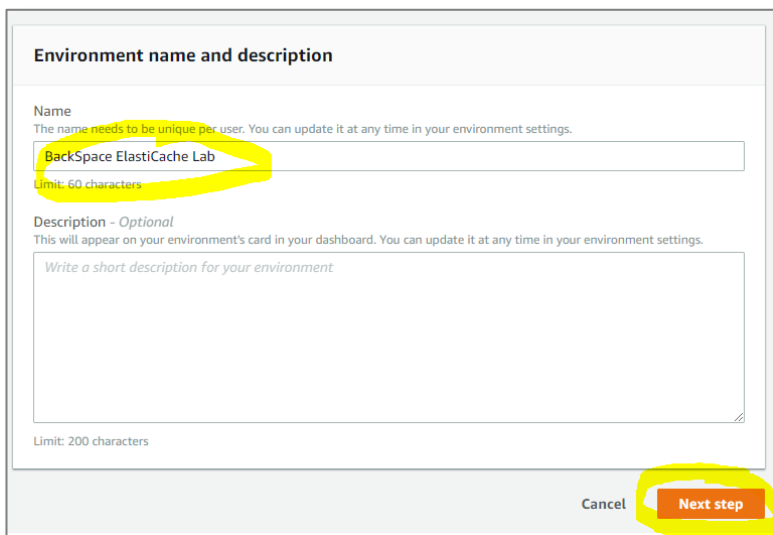
Please note these lab notes have been updated for the Cloud9 IDE due to problems with the Atom Remote-Edit package. You can still use an IDE such as Atom if you like but, you will need to upload files to your EC2 instance using FileZilla.

Go to *Services - Cloud9* from the console

Click *Create environment*

Give your environment a name

Click *Next step*



**Environment name and description**

**Name**  
The name needs to be unique per user. You can update it at any time in your environment settings.

BackSpace ElastiCache Lab  
Limit: 60 characters

**Description - Optional**  
This will appear on your environment's card in your dashboard. You can update it at any time in your environment settings.

Write a short description for your environment

Limit: 200 characters

Cancel **Next step**

Leave default settings

Click *Next step*

**Environment settings**

**Environment type** [Info](#)  
Choose between creating a new EC2 instance for your new environment or connecting directly to your server over SSH.

☒ **Create a new instance for environment (EC2)**  
Launch a new instance in this region to run your new environment.

☐ **Connect and run in remote server (SSH)**  
Display instructions to connect remotely over SSH and run your new environment.

**Instance type**

☒ **t2.micro (1 GiB RAM + 1 vCPU)**  
Free-tier eligible. Ideal for educational users and exploration.

☐ **t2.small (2 GiB RAM + 1 vCPU)**  
Recommended for small-sized web projects.

☐ **m4.large (8 GiB RAM + 2 vCPU)**  
Recommended for production and general-purpose development.

☐ **Other instance type**  
Select an instance type.

t2.nano

**Cost-saving setting**  
Choose a predetermined amount of time to auto-hibernate your environment and prevent unnecessary charges. We recommend a hibernation settings of half an hour of no activity to maximize savings.

After 30 minutes (default)

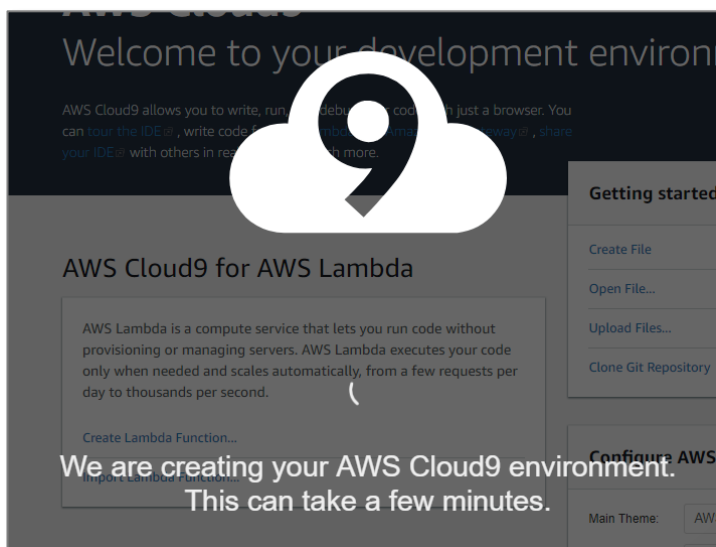
**IAM role**  
AWS Cloud9 creates a service-linked role for you. This allows AWS Cloud9 to call other AWS services on your behalf. You can delete the role from the AWS IAM console once you no longer have any AWS Cloud9 environments. [Learn more](#)

AWSServiceRoleForAWSCloud9

► **Network settings (advanced)**

Cancel Previous step **Next step**

Click *Create environment*



## Creating the Redis Security Group

Go to the EC2 console

Select the Cloud9 IDE instance

Click on the security group to view

The screenshot shows the AWS Management Console interface. On the left sidebar, the 'Instances' link is highlighted. The main content area displays a list of EC2 instances. The first instance, 'aws-cloud9-BackSpace-Labs-aa0...', is selected and highlighted. Below the list, the details for this instance are shown. The 'Security groups' section is highlighted, showing the security group 'aws-cloud9-BackSpace-Labs-aa0e0177557d4b7da26fa3c1fe1EQOMGDRQ5CW8'.

Name	Instance ID	Instance Type	Availability Zone	Instance State	Status Checks
aws-cloud9-BackSpace-Labs-aa0...	i-0c6d75806f0069664	t2.micro	us-east-1c	running	2/2 checks ...

Instance: i-0c6d75806f0069664 (aws-cloud9-BackSpace-Labs-aa0e0177557d4b7da26fa3c1fe150530) Public DNS: ec2-3-95-15-104.compute-1.amazonaws.com

Description		Status Checks	Monitoring	Tags
Instance ID	i-0c6d75806f0069664	Public DNS (IPv4)	ec2-3-95-15-104.compute-1.amazonaws.com	
Instance state	running	IPv4 Public IP	3.95.15.104	
Instance type	t2.micro	IPv6 IPs	-	
Finding	Opt-in to AWS Compute Optimizer for recommendations. <a href="#">Learn more</a>	Elastic IPs		
Private DNS	ip-172-31-92-177.ec2.internal	Availability zone	us-east-1c	
Private IPs	172.31.92.177	Security groups	aws-cloud9-BackSpace-Labs-aa0e0177557d4b7da26fa3c1fe1EQOMGDRQ5CW8. <a href="#">view inbound rules</a>	

Click on the Security group ID

The screenshot shows the AWS Management Console interface for Security Groups. The 'Security Groups (1/1)' section is active. The security group ID 'sg-037b203a46ec051ea' is highlighted in the table.

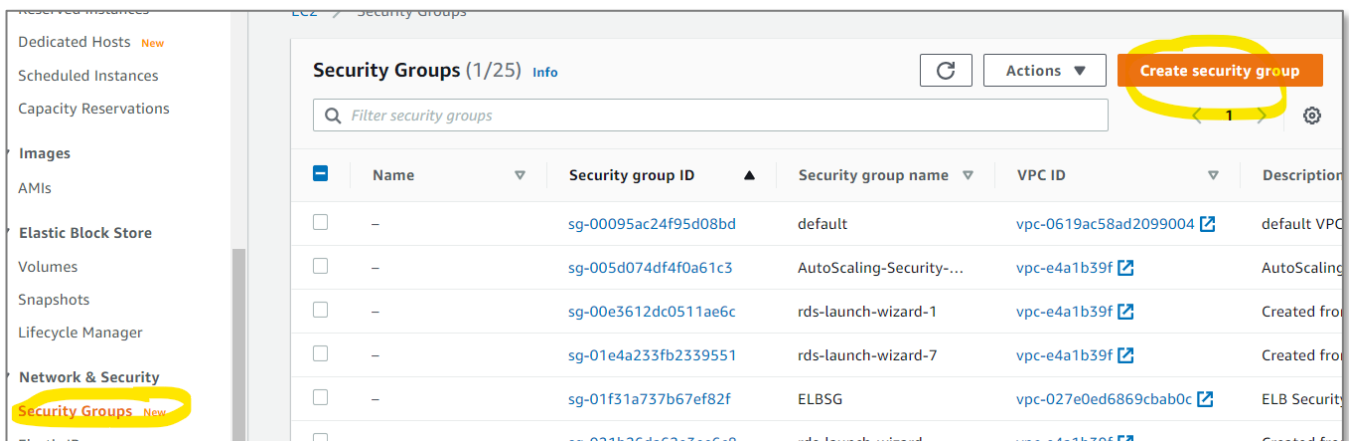
Name	Security group ID	Security group name
aws-cloud9-BackSp...	sg-037b203a46ec051ea	aws-cloud9-BackSpace...

Copy the Security group ID

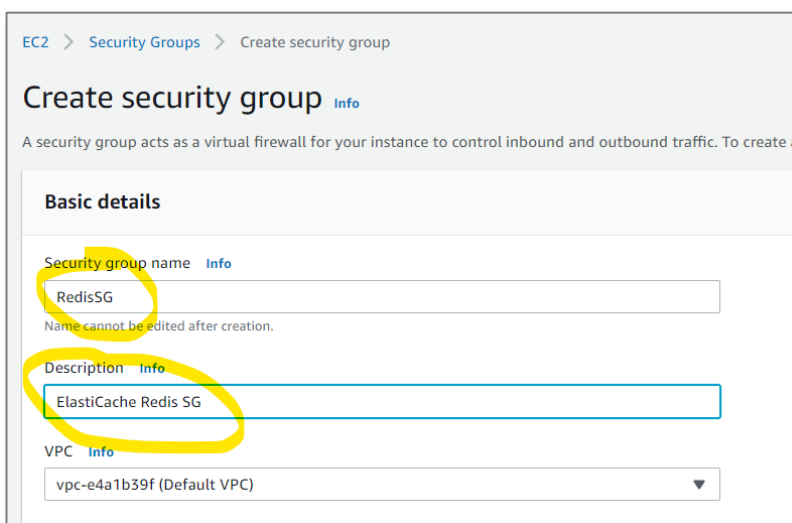


Select *Security Groups*

Click *Create security group*



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



Click *Add Rule* to Inbound rules

**Inbound rules** Info

This security group has no inbound rules.

**Add rule**

**Outbound rules** Info

Type Info Protocol Info Port range Info Destination Info

All traffic All All Custom 0.0.0.0/0

**Add rule**

Create a custom TCP rule for the ElastiCache Redis port 6379 with source the Cloud9 Security Group for the environment you created.

**Inbound rules** Info

Type Info Protocol Info Port range Info Source Info Description

Custom TCP TCP 6379 Custom 203a46ec051ea

**Add rule**

Security Groups

- aws-cloud9-BackSpace... | sg-037b203a46ec051ea
- aws-cloud9-BackSpace...

**Inbound rules** Info

Type Info Protocol Info Port range Info Source Info

Custom TCP TCP 6379 Custom sg-037b203a46ec051ea

**Add rule**

Click *Create security group*



**Tags - optional**  
A tag is a label that you assign to an AWS resource. Each tag consists of a key and an optional value. You can use tags to search and filter your resources or track your AWS costs.

No tags associated with the resource.

[Add new tag](#)

You can add up to 50 more tag

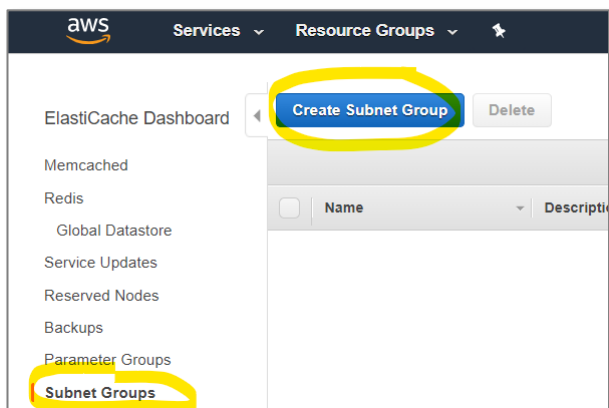
Cancel [Create security group](#)

## Creating an ElastiCache Subnet Group

Go to the ElastiCache console.

Click on Subnet Groups

Click Create Cache Subnet Group



Give it a name *backspace-lab-sn-group*

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

## Launching an ElastiCache Cluster

Select *Redis*

Click *Create*

**aws** Services Resource Groups

ElastiCache Dashboard  Actions

Memcached

**Redis**

Global Datastore

Service Updates

Reserved Nodes

Backups

Filter: Search Clusters...

☐ Cluster Name Mode

Call the cluster *backspace-lab-redis*

Select the t2 micro node type

Click *Save*

**Select node type**

Instance family **r5** **m5** **r4** **m4** **r3** **m3** **t3** **t2**

Node type	Memory (GiB)	Network performance
<input checked="" type="checkbox"/> cache.t2.micro	0.5	Low to moderate
<input type="checkbox"/> cache.t2.small	1.5	Low to moderate
<input type="checkbox"/> cache.t2.medium	3	Low to moderate

Number of replicas 0 for the lab

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.

**Redis settings**

**Name** backspace-lab-redis ⓘ

**Engine version compatibility** 5.0.6 ⓘ

**Port** 6379 ⓘ

**Parameter group** default.redis5.0 ⓘ

**Node type** cache.t2.micro (0.5 GiB) ⓘ

**Number of replicas** 0 ⓘ

**Multi-AZ** ☐ ⓘ

**Multi-AZ**  
 Multi-AZ can not be enabled when the number of replicas is set to 0. Select one or more replicas to enable Multi-AZ. Learn more

Click *Advanced Redis settings*

Select your Subnet 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-sn-group (vpc-e4a1b39f) ⓘ

**Preferred availability zone(s)** ☒ No preference ⓘ  
☐ Select zones

Select your *RedisSG* Security Group created previously (When searching for it remember names are case sensitive)

**Security**

**Security groups** RedisSG (sg-05995ea788d11147d) ⓘ

**Encryption at-rest** ☐ ⓘ

**Encryption in-transit** ☐ ⓘ

Uncheck *Enable automatic backups*

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](#)

Your Redis cluster will be creating

ElastiCache Dashboard

Memcached

Redis

Global Datastore

Service Updates

Reserved Nodes

Filter:

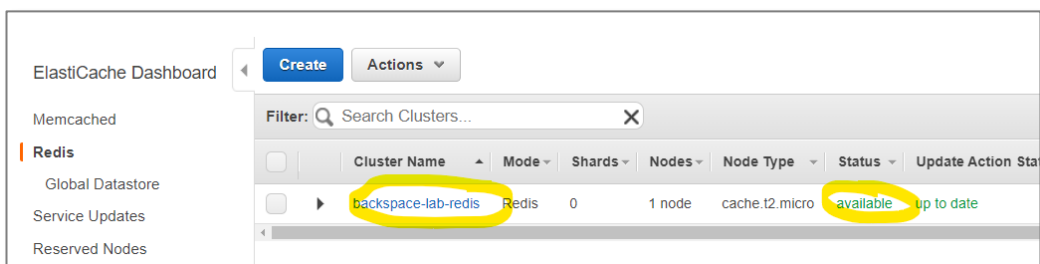
	Cluster Name	Mode	Shards	Nodes	Node Type	Status	Update Ac
	backspace-lab-redis	Redis	0	1 node	cache.t2.micro	creating	up to date

# Connect to an ElastiCache Redis Cluster using NodeJS

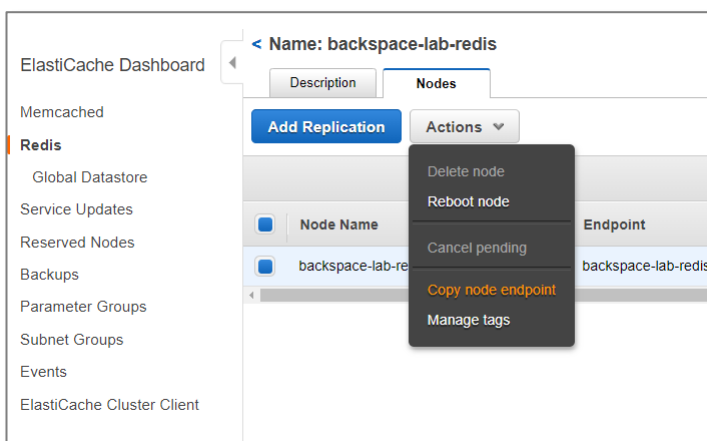
In this section we will connect to an ElastiCache Redis cluster using NodeJS.

Wait for your Cluster status to be *available*

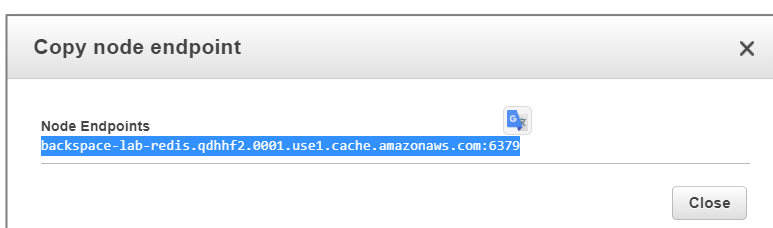
Click on the Cluster



Select **Actions** > **Copy node endpoint**



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



Go to the Cloud9 IDE

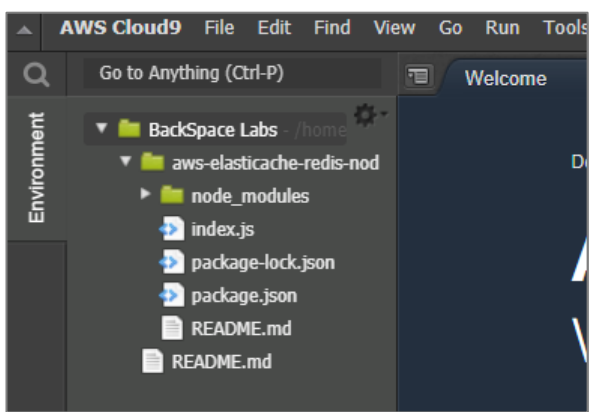
Clone the Git repository for the code template

```
git clone https://github.com/backspace-academy/aws-elasticache-redis-nodejs
```

Install the dependencies

```
npm install
```

Your file tree should now look like this:





Open index.js

Paste your ElastiCache Redis cluster endpoint into *YOUR\_REDIS\_ENDPOINT*

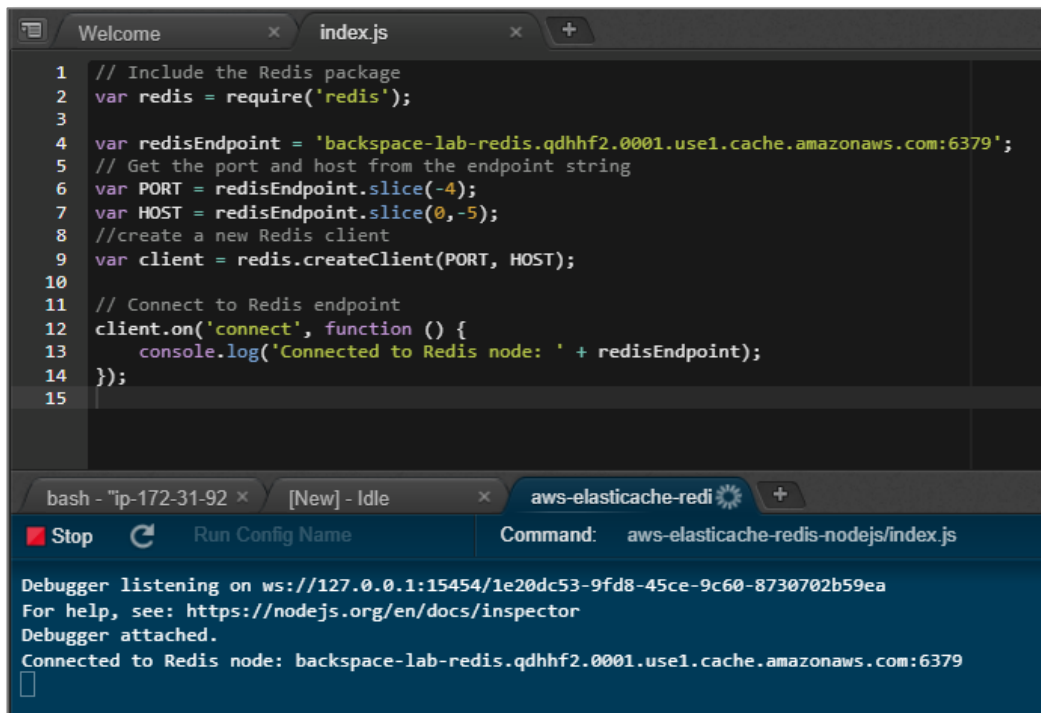
```
// Include the Redis package
var redis = require('redis');

var redisEndpoint = 'YOUR_REDIS_ENDPOINT';
// Get the port and host from the endpoint string
var PORT = redisEndpoint.slice(-4);
var HOST = redisEndpoint.slice(0,-5);
//create a new Redis client
var client = redis.createClient(PORT, HOST);


// Connect to Redis endpoint
client.on('connect', function () {
  console.log('Connected to Redis node: ' + redisEndpoint);
});
```



Click  +  to save the file


Now run your application and it should connect to your Redis node:



```
1 // Include the Redis package
2 var redis = require('redis');
3
4 var redisEndpoint = 'backspace-lab-redis.qdhhf2.0001.use1.cache.amazonaws.com:6379';
5 // Get the port and host from the endpoint string
6 var PORT = redisEndpoint.slice(-4);
7 var HOST = redisEndpoint.slice(0,-5);
8 //create a new Redis client
9 var client = redis.createClient(PORT, HOST);
10
11 // Connect to Redis endpoint
12 client.on('connect', function () {
13   console.log('Connected to Redis node: ' + redisEndpoint);
14 });
15
```

bash - "ip-172-31-92" x [New] - Idle x aws-elasticache-redis  +

 Stop  Run Config Name Command: aws-elasticache-redis-nodejs/index.js

Debugger listening on ws://127.0.0.1:15454/1e20dc53-9fd8-45ce-9c60-8730702b59ea  
For help, see: <https://nodejs.org/en/docs/inspector>  
Debugger attached.  
Connected to Redis node: backspace-lab-redis.qdhhf2.0001.use1.cache.amazonaws.com:6379  


Stop the application.

# Using ElastiCache Redis with NodeJS



In this section we will read and write to an ElastiCache Redis cluster using NodeJS.

Add a call to a function called *writeRedisKey* in the connect callback

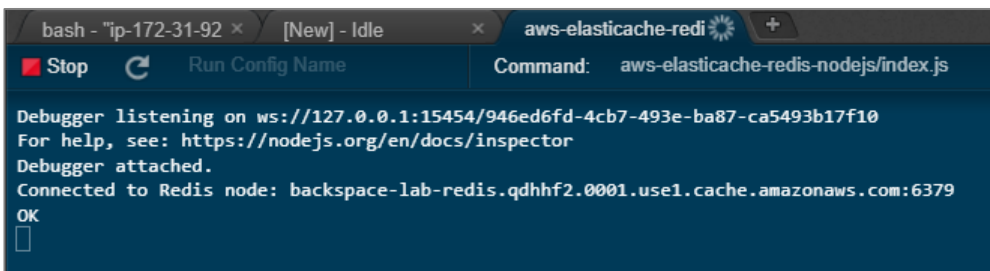
Create the new function which stores the high score for a game:

```
// Connect to Redis endpoint
client.on('connect', function () {
  console.log('Connected to Redis node: ' + redisEndpoint);
  writeRedisKey("myHighScore", "1000");
});

// Write to Redis
function writeRedisKey(keyRedis, value) {
  client.set(keyRedis, value, function (err, response) {
    if (err) console.log(err, err.stack); // an error occurred
    else {
      console.log(response);
    }
  });
}
```

Click  +  to save the file

Now run your application and it should create and save the key to your Redis node.



```
bash - "ip-172-31-92" x [New] - Idle x aws-elasticache-redis
Stop Run Config Name Command: aws-elasticache-redis-nodejs/index.js
Debugger listening on ws://127.0.0.1:15454/946ed6fd-4cb7-493e-ba87-ca5493b17f10
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
Connected to Redis node: backspace-lab-redis.qdhf2.0001.use1.cache.amazonaws.com:6379
OK
```

Stop the application.



Add a call to a function called *expireRedisKey* in the *writeRedisKey* callback



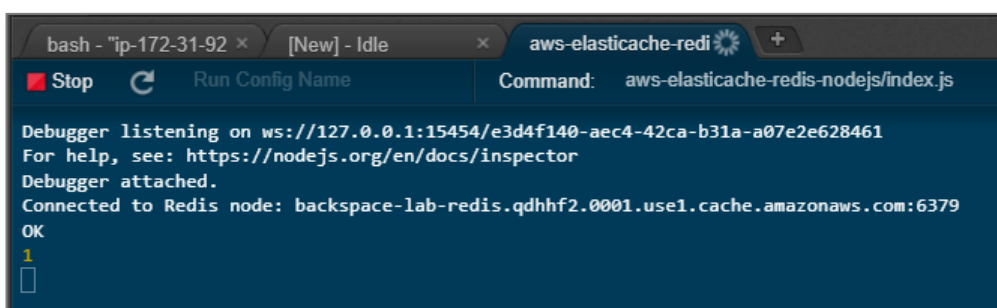
Create the new function which sets an expire time of 30 seconds for the key.

```
// Write to Redis
function writeRedisKey(keyRedis, value) {
  client.set(keyRedis, value, function (err, response) {
    if (err) console.log(err, err.stack); // an error occurred
    else {
      console.log(response);
      expireRedisKey(keyRedis, 30);
    }
  });
}

// Set key expiry time
function expireRedisKey(keyRedis, value) {
  client.expire(keyRedis, value, function (err, response) {
    if (err) console.log(err, err.stack); // an error occurred
    else {
      console.log(response);
    }
  });
}
```

Click  +  to save the file



Now run your application and it should create and save the key with an expiry time to your Redis node.



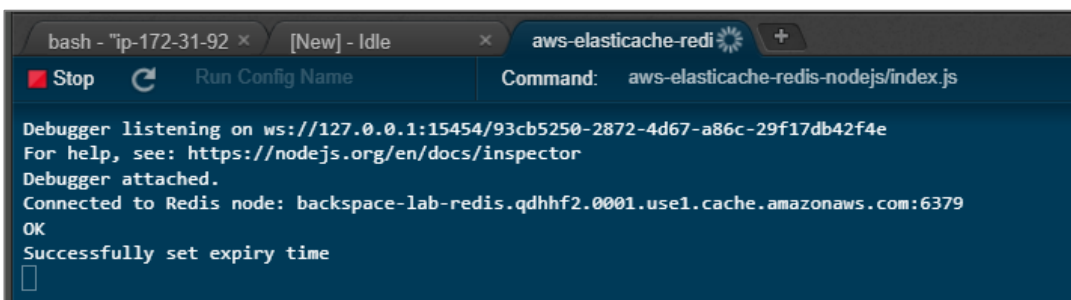
Stop the application.

Update the code to report when the key does not exist.

```
// Set key expiry time
function expireRedisKey(keyRedis, value) {
  client.expire(keyRedis, value, function (err, response) {
    if (err) console.log(err, err.stack); // an error occurred
    else {
      if (response){
        console.log('Successfully set expiry time');
      }
      else
        console.log('Unsuccessful. Key ' + keyRedis + ' does not exist!');
    }
  });
}
```

Click  +  to save the file

Now run your application



```
bash - "ip-172-31-92" x [New] - Idle x aws-elasticache-redis
Stop Run Config Name Command: aws-elasticache-redis-nodejs/index.js
Debugger listening on ws://127.0.0.1:15454/93cb5250-2872-4d67-a86c-29f17db42f4e
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
Connected to Redis node: backspace-lab-redis.qdhhf2.0001.use1.cache.amazonaws.com:6379
OK
Successfully set expiry time
█
```

Add a call to a function called *readRedisKey* in the *expireRedisKey* callback



Create the new function which returns current the high score for a game.

```

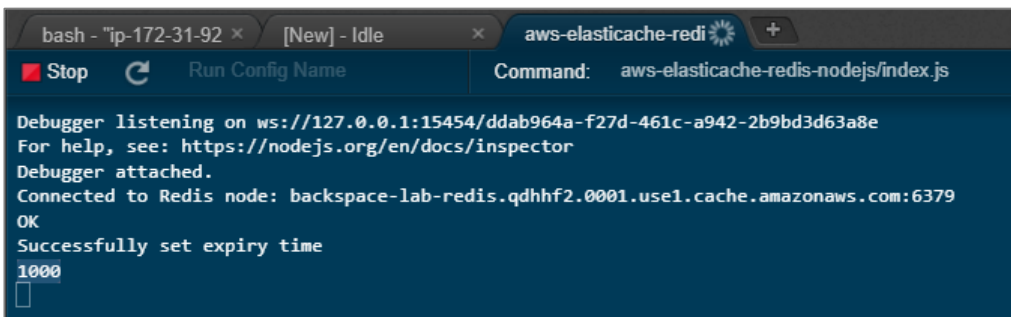
// Set key expiry time
function expireRedisKey(keyRedis, value) {
  client.expire(keyRedis, value, function (err, response) {
    if (err) console.log(err, err.stack); // an error occurred
    else {
      if (response){
        console.log('Successfully set expiry time');
        readRedisKey(keyRedis);
      }
      else
        console.log('Unsuccessful. Key ' + keyRedis + 'does not exist!');
    }
  });
}

// Read from Redis
function readRedisKey(keyRedis) {
  client.get(keyRedis, function (err, response) {
    if (err) console.log(err, err.stack); // an error occurred
    else {
      console.log(response);
    }
  });
}

```

Click  +  to save the file

Now run your application.



```

bash - "ip-172-31-92" x [New] - Idle x aws-elasticache-redis
Stop Run Config Name Command: aws-elasticache-redis-nodejs/index.js
Debugger listening on ws://127.0.0.1:15454/ddab964a-f27d-461c-a942-2b9bd3d63a8e
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
Connected to Redis node: backspace-lab-redis.qdhhf2.0001.use1.cache.amazonaws.com:6379
OK
Successfully set expiry time
1000

```

Stop the application.

Add a call to a function called writeRedisObject in the readRedisKey callback

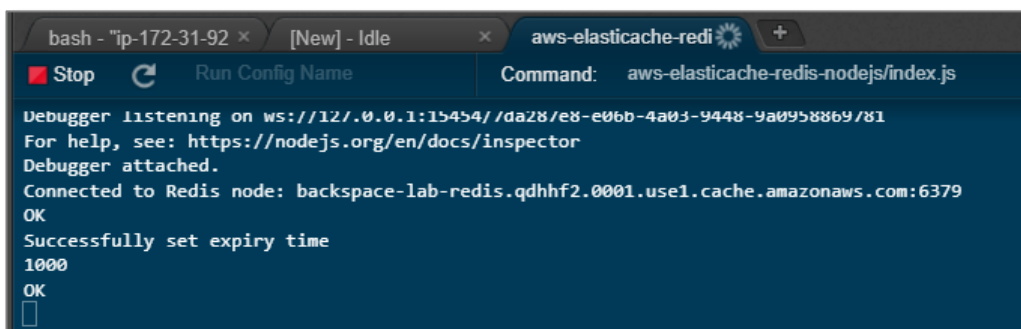
Create the new function which will create and save an object of keys.

```
// Read from Redis
function readRedisKey(keyRedis) {
  client.get(keyRedis, function (err, response) {
    if (err) console.log(err, err.stack); // an error occurred
    else {
      console.log(response);
      var objInfo = {
        info1: "This is info 1",
        info2: "This is info 2",
        info3: "This is info 3"
      };
      writeRedisObject("myInfo", objInfo);
    }
  });
}

// Write Redis Object of keys
function writeRedisObject(objRedis, value) {
  client.hmset(objRedis, value, function (err, response) {
    if (err) console.log(err, err.stack); // an error occurred
    else {
      console.log(response);
    }
  });
}
```

Click  +  to save the file

Now run your application.




Stop the application.

Add a call to a function called readRedisObject in the writeRedisObject callback

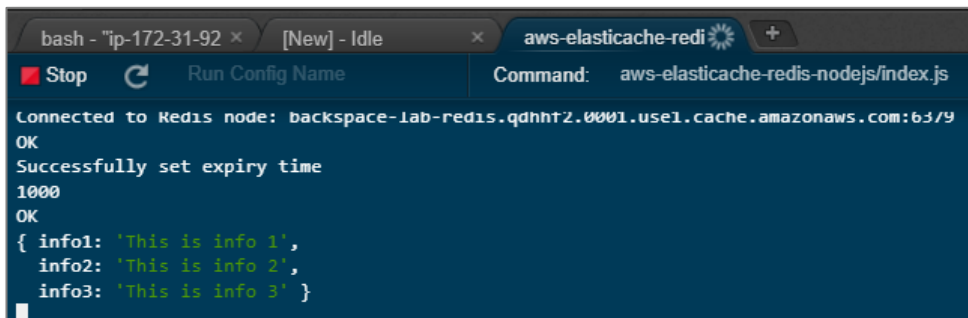
Create the new function which will read an object of keys.

```
// Write Redis Object of keys
function writeRedisObject(objRedis, value) {
  client.hmset(objRedis, value, function (err, response) {
    if (err) console.log(err, err.stack); // an error occurred
    else {
      console.log(response);
      readRedisObject(objRedis);
    }
  });
}

// Read Redis Object of keys
function readRedisObject(objRedis) {
  client.hgetall(objRedis, function (err, response) {
    if (err) console.log(err, err.stack); // an error occurred
    else {
      console.log(response);
    }
  });
}
```

Click  +  to save the file

Now run your application.



The terminal window shows the following output:

```
bash - "ip-172-31-92" x [New] - Idle x aws-elasticache-redis
Stop Run Config Name Command: aws-elasticache-redis-nodejs/index.js
Connected to Redis node: backspace-lab-redis.qdhtf2.0001.usel.cache.amazonaws.com:6379
OK
Successfully set expiry time
1000
OK
{ info1: 'This is info 1',
  info2: 'This is info 2',
  info3: 'This is info 3' }
```

Stop the application.

## Cleaning Up

Now clean up by deleting your cluster in the ElastiCache console.

Go to *Redis* and select your cluster

Click *Delete*

ElastiCache Dashboard

Create Actions

Memcached

**Redis**

Global Datastore

Service Updates

Reserved Nodes

Backups

Filter: Q S

Backup

Modify

Reboot

Delete

Apply Service Update

View/Stop Update

Shards	Nodes	Node Type	Status	Update Action
0	1 node	cache.t2.micro	available	up to date

ab-redis

Global Datastore

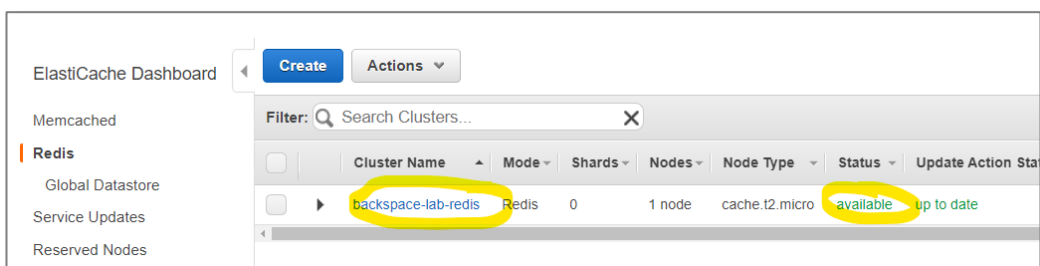
Creation

# Connect to an ElastiCache Redis Cluster using Python

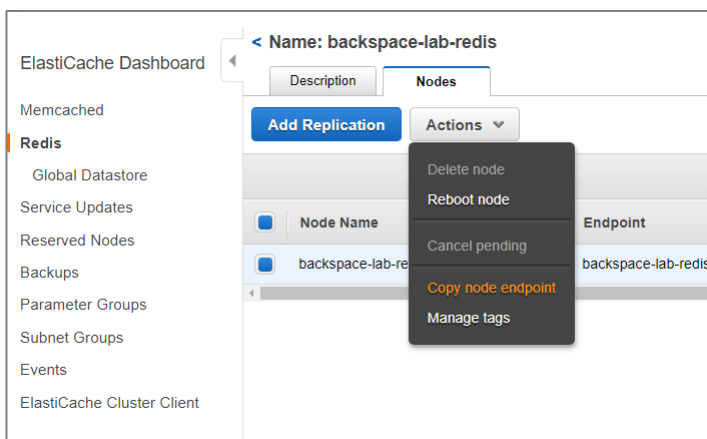
In this section we will connect to an ElastiCache Redis cluster using Python.

Wait for your Cluster status to be *available*

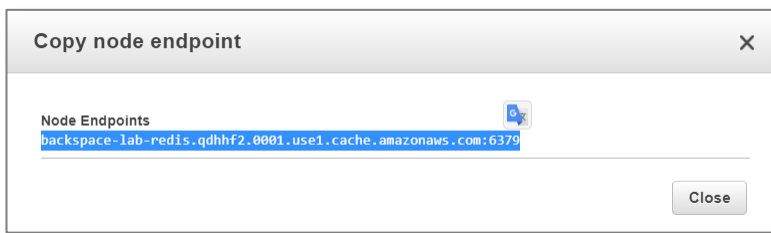
Click on the Cluster



Select **Actions** > **Copy node endpoint**



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



Go to the Cloud9 IDE

Clone the Git repository for the code template

```
git clone https://github.com/backspace-academy/aws-elasticache-redis-python
```

Install the dependencies

```
cd aws-elasticache-redis-python
sudo pip install redis
```

Check Redis client installed ok

```
pip show redis
```

```
python3.6 - "ip-172-31-172-31" x [New] - Idle x +
pcoady:~/environment $ git clone https://github.com/backspace-academy/aws-elasticache-redis-python
Cloning into 'aws-elasticache-redis-python'...
remote: Enumerating objects: 9, done.
remote: Counting objects: 100% (9/9), done.
remote: Compressing objects: 100% (5/5), done.
remote: Total 9 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (9/9), done.
pcoady:~/environment $ cd aws-elasticache-redis-python
pcoady:~/environment/aws-elasticache-redis-python (master) $ sudo pip install redis
Requirement already satisfied: redis in /usr/local/lib/python3.6/site-packages
You are using pip version 9.0.3, however version 20.1.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
pcoady:~/environment/aws-elasticache-redis-python (master) $ pip show redis
Name: redis
Version: 3.0.1
Summary: Python client for Redis key-value store
Home-page: https://github.com/andymccurdy/redis-py
Author: Andy McCurdy
Author-email: sedrik@gmail.com
License: MIT
Location: /usr/local/lib/python3.6/site-packages
Requires:
You are using pip version 9.0.3, however version 20.1.1 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
pcoady:~/environment/aws-elasticache-redis-python (master) $
```



Open elasticache.py



Paste your ElastiCache Redis cluster endpoint into *YOUR\_REDIS\_ENDPOINT*

```
# Import the Redis package
import redis

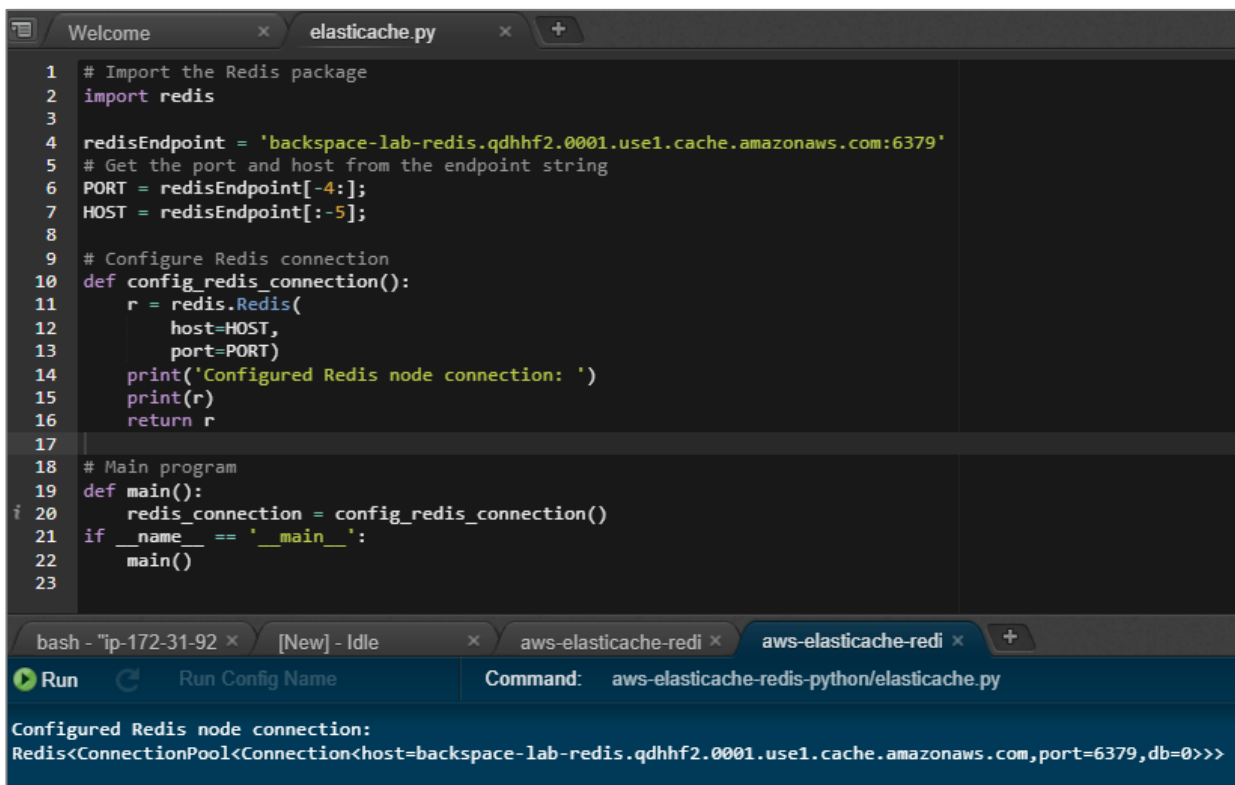
redisEndpoint = 'YOUR_REDIS_ENDPOINT'
# Get the port and host from the endpoint string
PORT = redisEndpoint[-4:];
HOST = redisEndpoint[:-5];

# Configure Redis connection
def config_redis_connection():
    r = redis.Redis(
        host=HOST,
        port=PORT)
    print('Configured Redis node connection: ')
    print(r)
    return r

# Main program
def main():
    redis_connection = config_redis_connection()
if __name__ == '__main__':
    main()
```

Click  +  to save the file

Now run your application



The screenshot shows an IDE with a Python script named `elasticache.py` and its execution output. The script imports the `redis` package, defines a Redis endpoint string, extracts the port and host, configures a Redis connection, and prints the connection details. The output shows the configured Redis node connection details.

```
1 # Import the Redis package
2 import redis
3
4 redisEndpoint = 'backspace-lab-redis.qdhhf2.0001.use1.cache.amazonaws.com:6379'
5 # Get the port and host from the endpoint string
6 PORT = redisEndpoint[-4:];
7 HOST = redisEndpoint[:-5];
8
9 # Configure Redis connection
10 def config_redis_connection():
11     r = redis.Redis(
12         host=HOST,
13         port=PORT)
14     print('Configured Redis node connection: ')
15     print(r)
16     return r
17
18 # Main program
19 def main():
20     redis_connection = config_redis_connection()
21 if __name__ == '__main__':
22     main()
23
```

Run Command: `aws-elasticache-redis-python/elasticache.py`

Configured Redis node connection:  
Redis<ConnectionPool<Connection<host=backspace-lab-redis.qdhhf2.0001.use1.cache.amazonaws.com,port=6379,db=0>>>

# Using ElastiCache Redis with Python



In this section we will read and write to an ElastiCache Redis cluster using Python.

Add a call to a function called `write_redis_key` in the main program

Create the new function which stores the high score for a game:

```
# Write to Redis cluster
def write_redis_key(r, new_key, new_value):
    temp_success = r.set(new_key, new_value)
    if temp_success:
        print('Successfully wrote to Redis')

# Main program
def main():
    redis_connection = config_redis_connection()
    write_redis_key(redis_connection, 'myHighScore', 1000)
if __name__ == '__main__':
    main()
```

Click  +  to save the file

Now run your application and it should create and save the key to your Redis node.

```

18 # Write to Redis cluster
19 def write_redis_key(r, new_key, new_value):
20     temp_success = r.set(new_key, new_value)
21     if temp_success:
22         print('Successfully wrote to Redis')
23
24 # Main program
25 def main():
26     # Connect to Redis endpoint
27     redis_connection = config_redis_connection()
28     write_redis_key(redis_connection, 'myHighScore', 1000)
29 if __name__ == '__main__':
30     main()
31

```

bash - "ip-172-31-92" x [New] - Idle x aws-elasticache-redis x aws-elasticache-redis x +

Run Run Config Name Command: aws-elasticache-redis-python/elasticache.py



Configured Redis node connection:  
 Redis<ConnectionPool<Connection<host=backspace-lab-redis.qdhhf2.0001.use1.cache.amazonaws.com,port=6379,db=0>>>  
 Successfully wrote to Redis

Now set an expire time of 30 seconds for the key.

```

# Write to Redis cluster
def write_redis_key(r, new_key, new_value):
    temp_success = r.set(new_key, new_value)
    if temp_success:
        print('Successfully wrote to Redis')
        # Set expiry time
        temp_success2 = r.expire(new_key, 30)
        if temp_success2:
            print('Successfully set expiry time')

```

Click  +  to save the file

Now run your application and it should create and save the key with an expiry time to your Redis node.

```

18 # Write to Redis cluster
19 def write_redis_key(r, new_key, new_value):
20     temp_success = r.set(new_key, new_value)
21     if temp_success:
22         print('Successfully wrote to Redis')
23         # Set expiry time
24         temp_success2 = r.expire(new_key, 30)
25         if temp_success2:
26             print('Successfully set expiry time')
27
28 # Main program
29 def main():
30     # Connect to Redis endpoint
31     redis_connection = config_redis_connection()
32     write_redis_key(redis_connection, 'myHighScore', 1000)
33 if __name__ == '__main__':
34     main()
35

```

aws-elasticache-redis-python/elasticache.py

Configured Redis node connection:  
 Redis<ConnectionPool<Connection<host=backspace-lab-redis.qdhhf2.0001.use1.cache.amazonaws.com,port=6379,db=0>>>  
 Successfully wrote to Redis  
 Successfully set expiry time

Process exited with code: 0

Add a call to a function called `read_redis_key` in the main program



Create the new function which returns current the high score for a game.

```

# Read from Redis cluster
def read_redis_key(r, new_key):
    temp_success = r.get(new_key)
    if temp_success:
        print('Value of ' + new_key + ' = ' + temp_success.decode("utf-8") )

# Main program
def main():
    redis_connection = config_redis_connection()
    write_redis_key(redis_connection, 'myHighScore', 1000)
    read_redis_key(redis_connection, 'myHighScore')
if __name__ == '__main__':
    main()

```

Click  +  to save the file

Now run your application.

```

27
28 # Read from Redis cluster
29 def read_redis_key(r, new_key):
30     temp_success = r.get(new_key)
31     if temp_success:
32         print('Value of ' + new_key + ' = ' + temp_success.decode("utf-8") )
33
34 # Main program
35 def main():
36     # Connect to Redis endpoint
37     redis_connection = config_redis_connection()
38     write_redis_key(redis_connection, 'myHighScore', 1000)
39     read_redis_key(redis_connection, 'myHighScore')
40 if __name__ == '__main__':
41     main()

```

bash - "ip-172-31-92" x [New] - Idle x aws-elasticache-redis x aws-elasticache-redis x +

Run Run Config Name Command: aws-elasticache-redis-python/elasticache.py

Configured Redis node connection:  
 Redis<ConnectionPool<Connection<host=backspace-lab-redis.qdhhf2.0001.use1.cache.amazonaws.com,port=6379,db=0>>>  
 Successfully wrote to Redis  
 Successfully set expiry time  
 Value of myHighScore = 1000

Add details for an object and add a call to a function called write\_redis\_object in the main program

Create the new function which will create and save an object of keys.



```

# Write object to Redis cluster
def write_redis_object(r, new_key, new_object):
    temp_success = r.hmset(new_key, new_object)
    if temp_success:
        print('Successfully wrote object to Redis')

# Main program
def main():
    # Connect to Redis endpoint
    redis_connection = config_redis_connection()
    write_redis_key(redis_connection, 'myHighScore', 1000)
    read_redis_key(redis_connection, 'myHighScore')
    temp_obj = {
        'info1': 'This is info 1',
        'info2': 'This is info 2',
        'info3': 'This is info 3'
    }
    write_redis_object(redis_connection, 'myInfo', temp_obj)

if __name__ == '__main__':
    main()

```

Click  +  to save the file

Now run your application.

```

34 # Write object to Redis cluster
35 def write_redis_object(r, new_key, new_object):
36     temp_success = r.hmset(new_key, new_object)
37     if temp_success:
38         print('Successfully wrote object to Redis')
39
40 # Main program
41 def main():
42     # Connect to Redis endpoint
43     redis_connection = config_redis_connection()
44     write_redis_key(redis_connection, 'myHighScore', 1000)
45     read_redis_key(redis_connection, 'myHighScore')
46     temp_obj = {
47         'info1': 'This is info 1',
48         'info2': 'This is info 2',
49         'info3': 'This is info 3'
50     }
51     write_redis_object(redis_connection, 'myInfo', temp_obj)
52
53 if __name__ == '__main__':
54     main()

```

aws-elasticache-redis-python/elasticache.py

```

Configured Redis node connection:
Redis<ConnectionPool<Connection<host=backspace-lab-redis.qdhhf2.0001.use1.cache.amazonaws.com,port=6379,db=0>>>
Successfully wrote to Redis
Successfully set expiry time
Value of myHighScore = 1000
Successfully wrote object to Redis

```

Stop the application.

Add a call to a function called read\_redis\_object in the main program

Create the new function which will read an object of keys.



```

# Read object from Redis cluster
def read_redis_object(r, new_key):
    temp_success = r.hgetall(new_key)
    if temp_success:
        print('Value of ' + new_key + ' :')
        print(temp_success)

# Main program
def main():
    # Connect to Redis endpoint
    redis_connection = config_redis_connection()
    write_redis_key(redis_connection, 'myHighScore', 1000)
    read_redis_key(redis_connection, 'myHighScore')
    temp_obj = {
        'info1': 'This is info 1',
        'info2': 'This is info 2',
        'info3': 'This is info 3'
    }
    write_redis_object(redis_connection, 'myInfo', temp_obj)
    read_redis_object(redis_connection, 'myInfo')

if __name__ == '__main__':
    main()

```

Click  +  to save the file

Now run your application.



```

40 # Read object from Redis cluster
41 def read_redis_object(r, new_key):
42     temp_success = r.hgetall(new_key)
43     if temp_success:
44         print('Value of ' + new_key + ' :')
45         print(temp_success)
46
47 # Main program
48 def main():
49     # Connect to Redis endpoint
50     redis_connection = config_redis_connection()
51     write_redis_key(redis_connection, 'myHighScore', 1000)
52     read_redis_key(redis_connection, 'myHighScore')
53     temp_obj = {
54         'info1': 'This is info 1',
55         'info2': 'This is info 2',
56         'info3': 'This is info 3'
57     }
58     write_redis_object(redis_connection, 'myInfo', temp_obj)
59     read_redis_object(redis_connection, 'myInfo')
60
61 if __name__ == '__main__':
62     main()
63

```

bash - "ip-172-31-92 x [New] - Idle x aws-elasticache-redis x aws-elasticache-redis x +

Run Run Config Name Command: aws-elasticache-redis-python/elasticache.py

Configured Redis node connection:  
 Redis<ConnectionPool<Connection<host=backspace-lab-redis.qdhhf2.0001.use1.cache.amazonaws.com,port=6379,db=0>>>  
 Successfully wrote to Redis  
 Successfully set expiry time  
 Value of myHighScore = 1000  
 Successfully wrote object to Redis  
 Value of myInfo :  
 {b'info1': b'This is info 1', b'info2': b'This is info 2', b'info3': b'This is info 3'}

Process exited with code: 0

## Cleaning Up

Now clean up by deleting your cluster in the ElastiCache console.

Go to *Redis* and select your cluster

Click *Delete*

