aws CERTIFIED Developer Associate

aws CERTIFIED Solutions Architect Associate

aws CERTIFIED SysOps Administrator Associate

lab

lab title

Programming AWS ElastiCache Redis using the Redis API Client

V1.03

Course title

BackSpace Academy
AWS Certification Preparation

BackSpace

Table of Contents

Contents

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 lastest 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

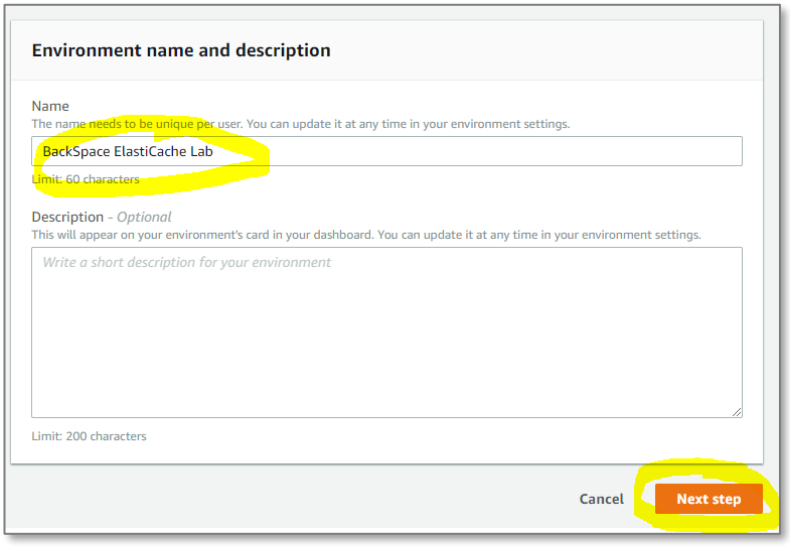
Please note these lab notes have been updated for the Cloud9 IDE due to problems with the Atom Remote-Edit package. You can still use an IDE such as Atom if you like but, you will need to upload files to your EC2 instance using FileZilla.

Go to *Services - Cloud9* from the console

Click *Create environment*

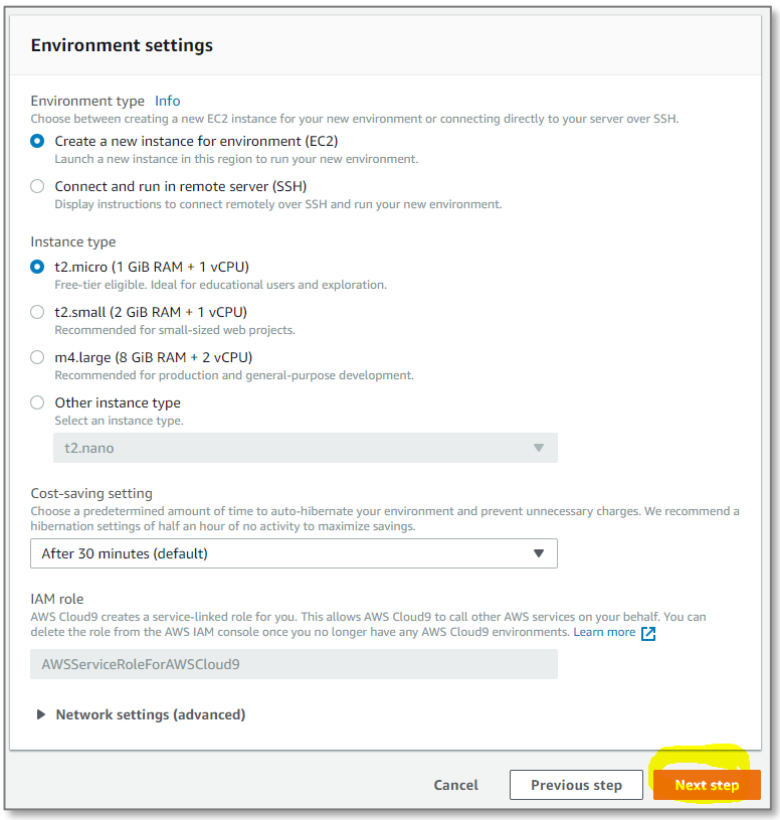
Give your environment a name

Click *Next step*

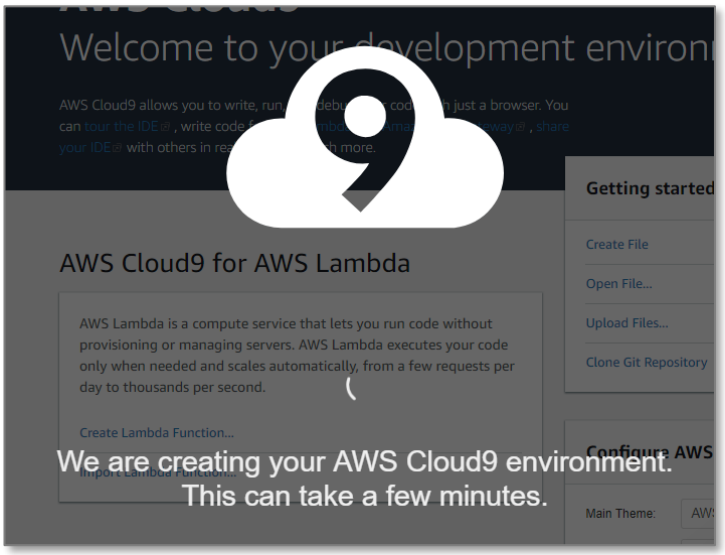


Leave default settings

Click *Next step*



Click *Create environment*

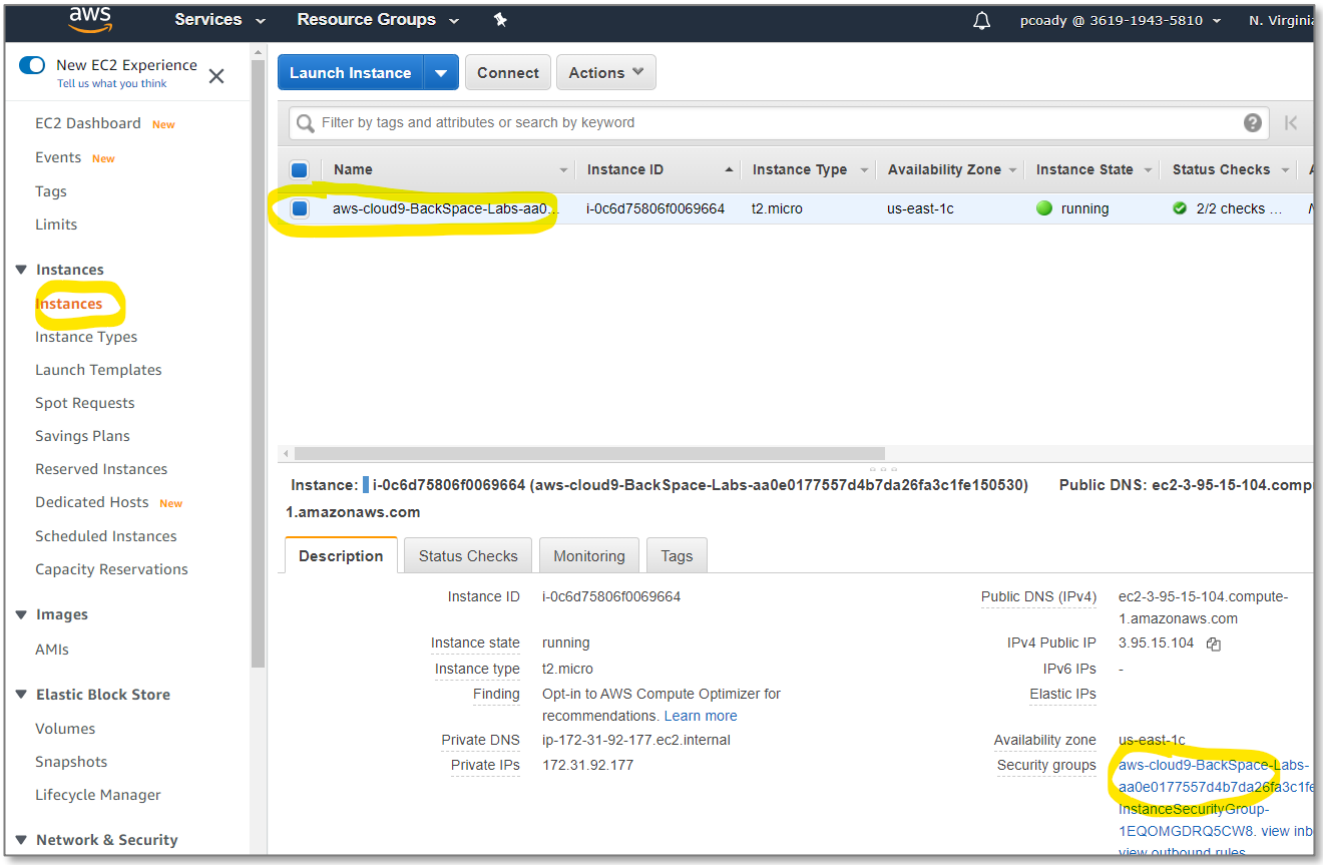


Creating the Redis Security Group

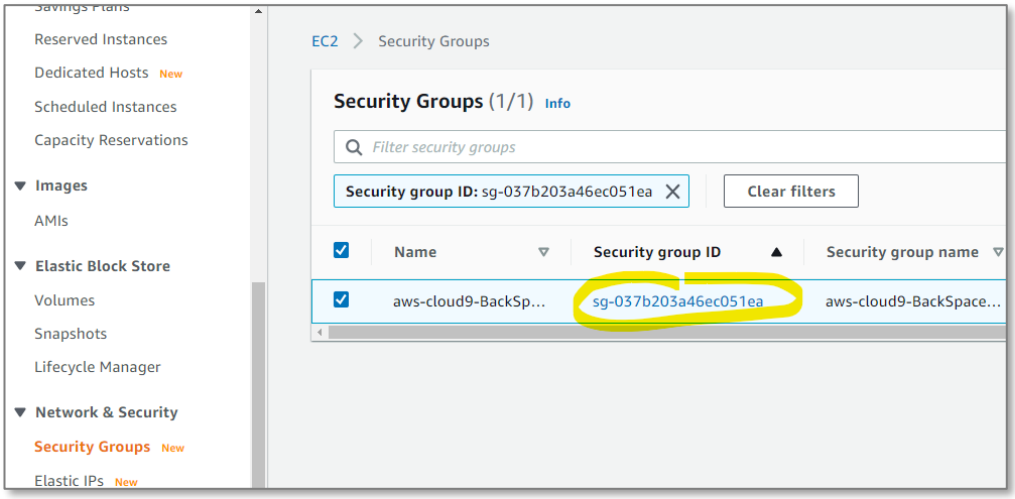
Go to the EC2 console

Select the Cloud9 IDE instance

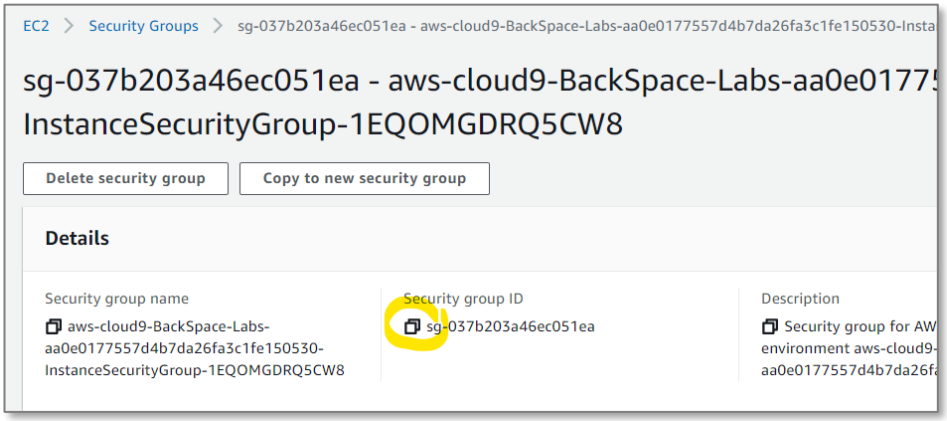
Click on the security group to view



Click on the *Security group ID*

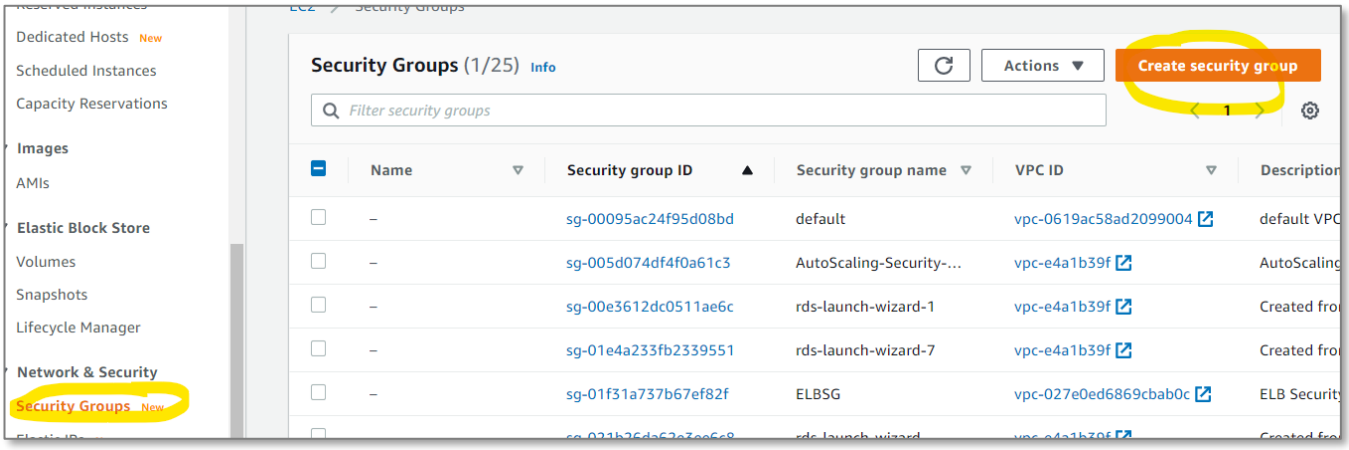


Copy the *Security group ID*

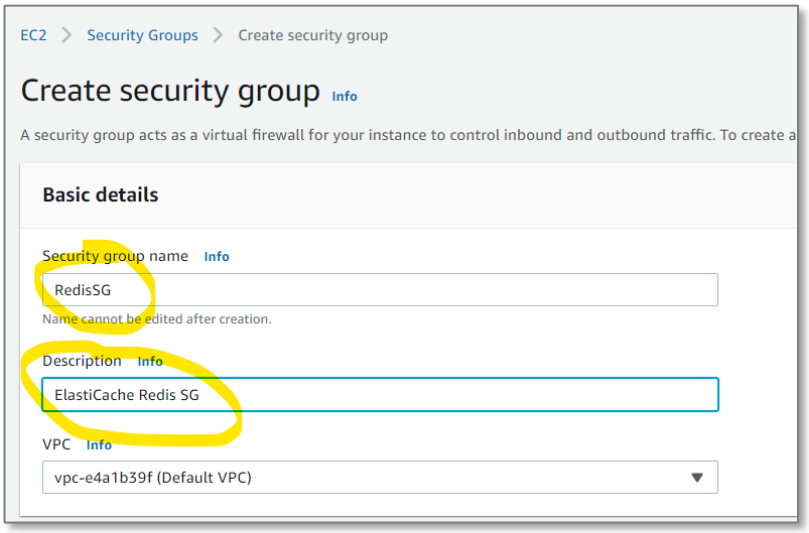


Select *Security Groups*

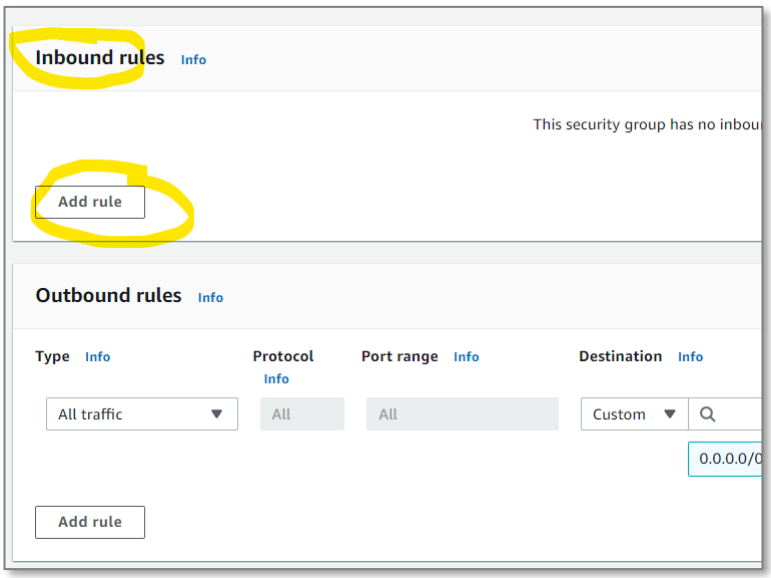
Click *Create security group*



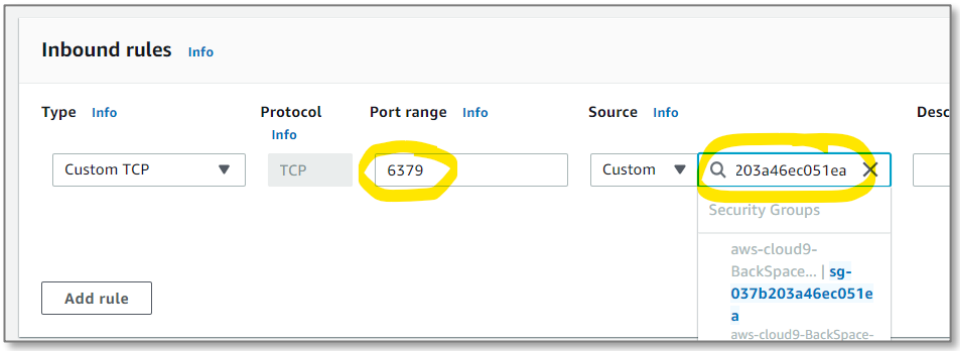
Create a new security group in the default VPC and call it *RedisSG*

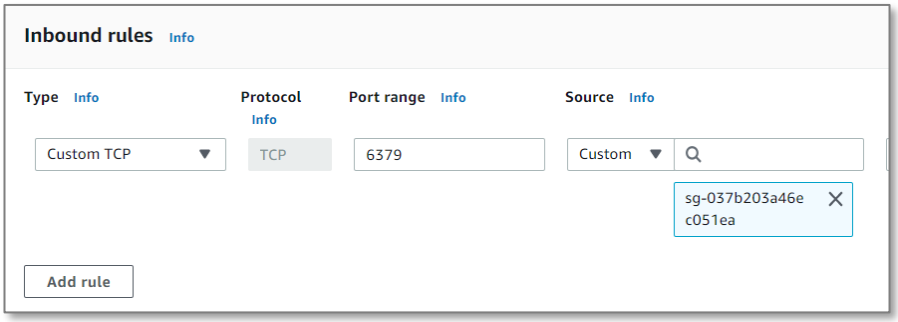


Click *Add Rule* to Inbound rules

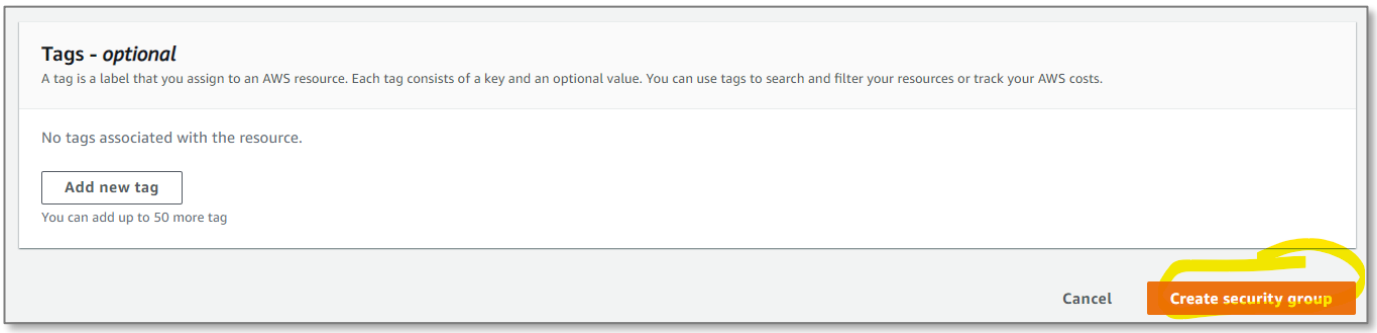


Create a custom TCP rule for the ElastiCache Redis port 6379 with source the Cloud9 Security Group for the environment you created.





Click *Create security group*

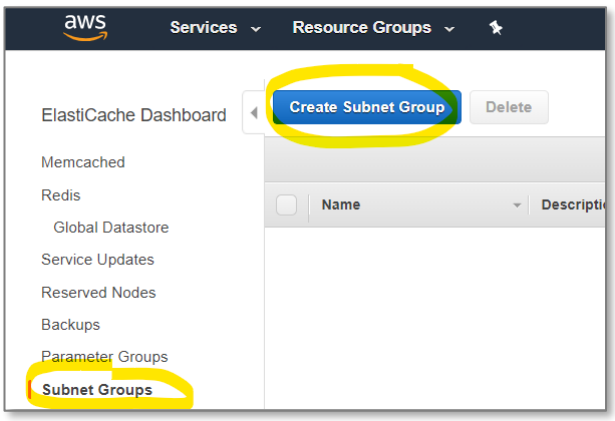


Creating an ElastiCache Subnet Group

Go to the ElastiCache console.

Click on Subnet Groups

Click Create Cache Subnet Group

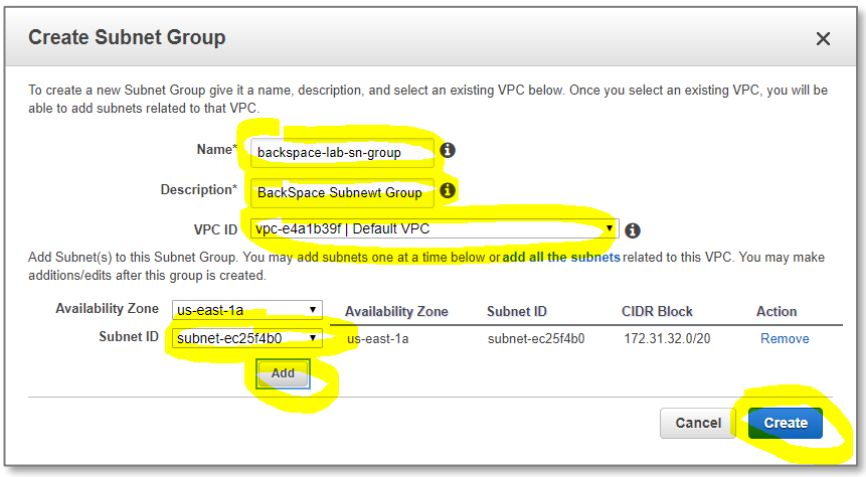


Give it a name *backspace-lab-sn-group*

Select the default VPC and an AZ and subnet.

Click Add

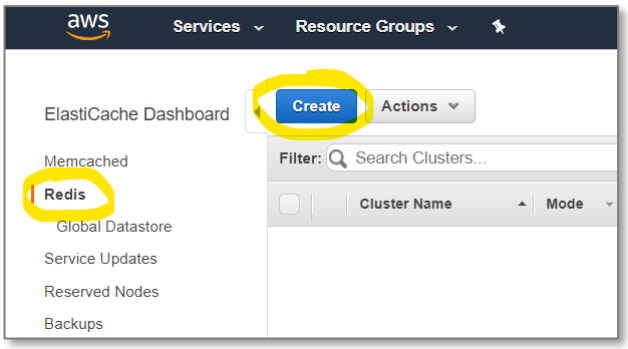
Click Create



Launching an ElastiCache Cluster

Select *Redis*

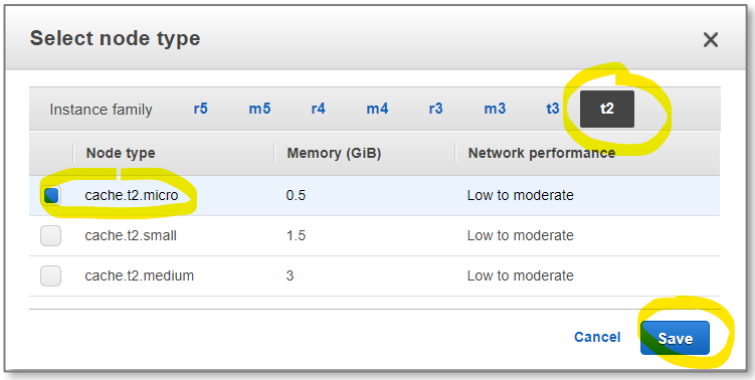
Click *Create*



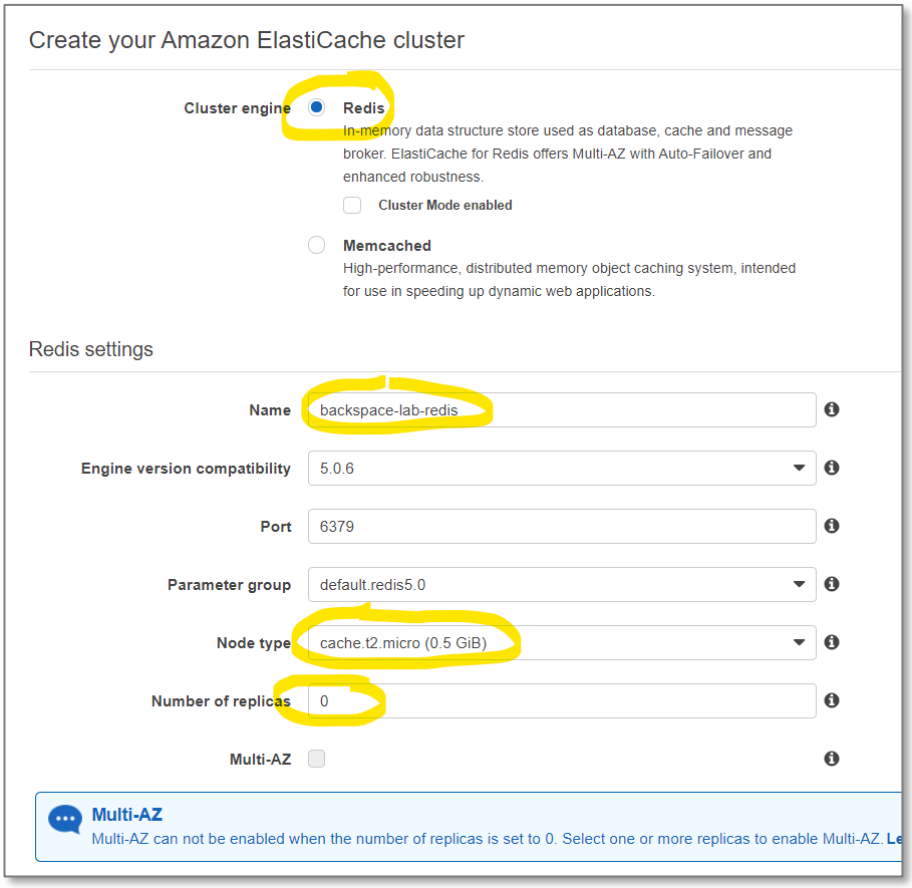
Call the cluster *backspace-lab-redis*

Select the t2 micro node type

Click *Save*

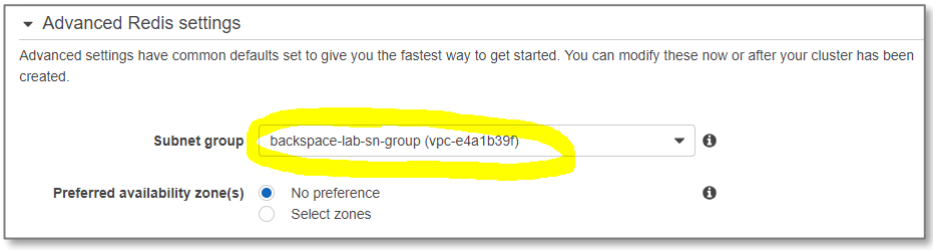


Number of replicas *0* for the lab

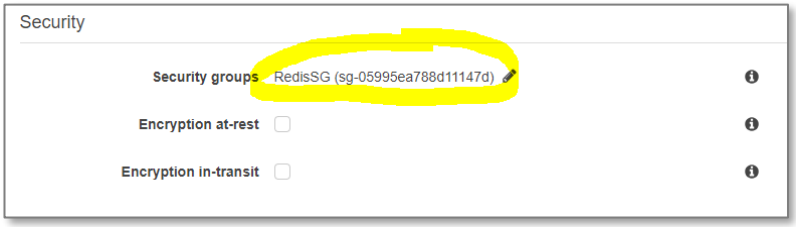


Click *Advanced Redis settings*

Select your Subnet Group created previously

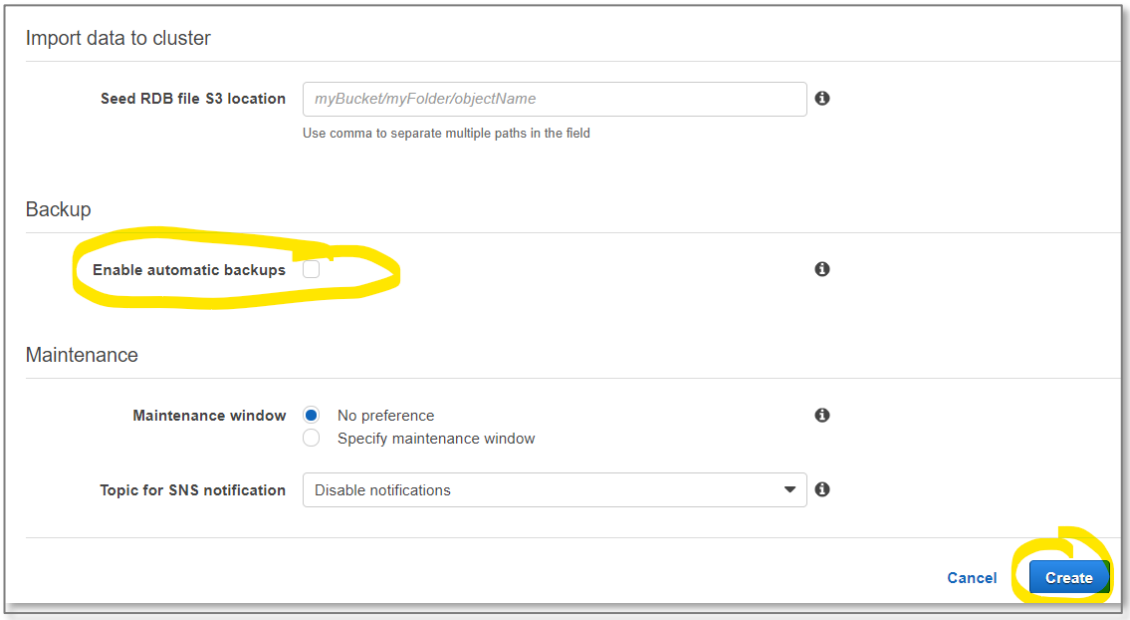


Select your *RedisSG* Security Group created previously (When searching for it remember names are case sensitive)

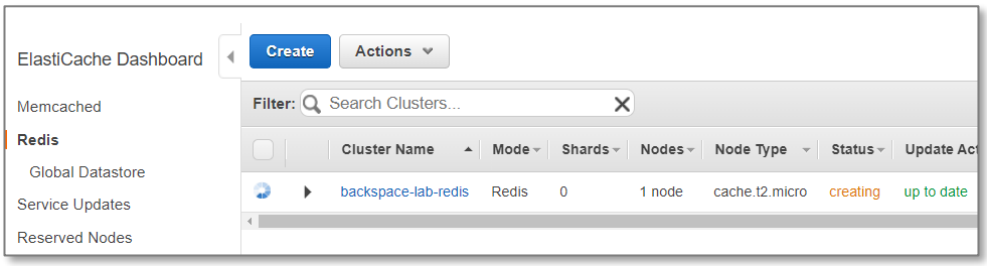


Uncheck *Enable automatic backups*

Click *Create*



Your Redis cluster will be creating

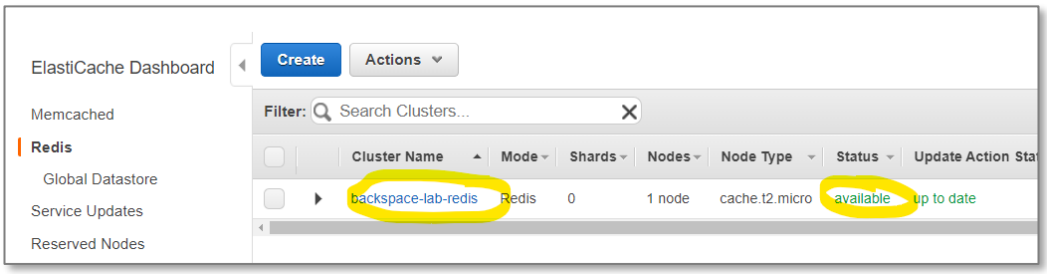


Connect to an ElastiCache Redis Cluster using NodeJS

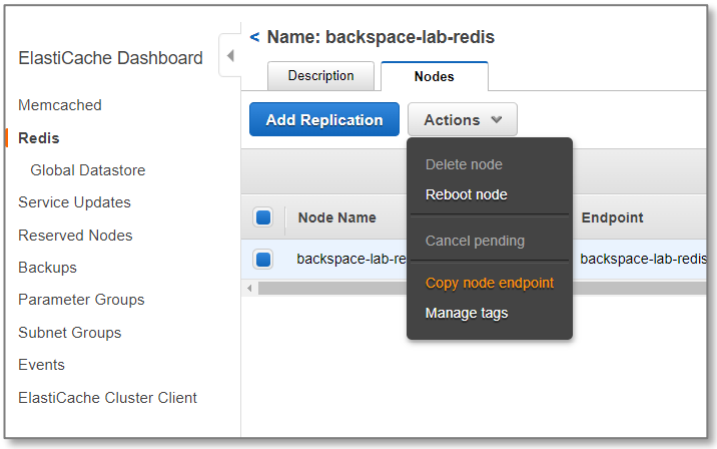
In this section we will connect to an ElastiCache Redis cluster using NodeJS.

Wait for your Cluster status to be *available*

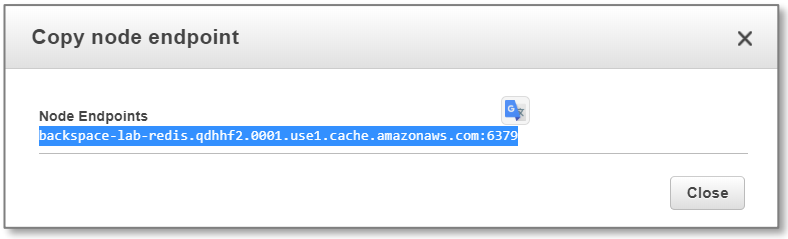
Click on the Cluster



Select *Actions* > *Copy node endpoint*



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



Go to the Cloud9 IDE

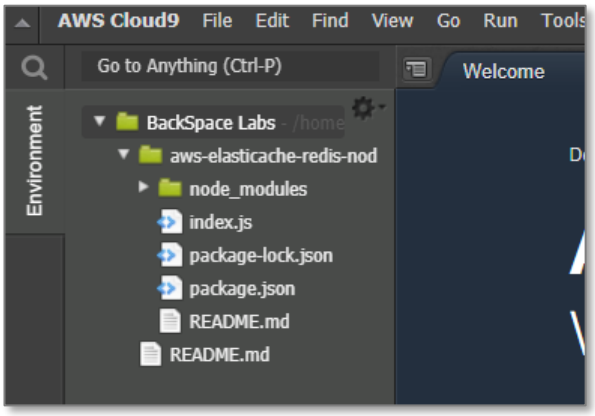
Clone the Git repository for the code template

```
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 Ctrl + S to save the file

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

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

var redisEndpoint = 'backspace-lab-redis.qdhhf2.0001.use1.cache.amazonaws.com:6379';
// 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);
});
```

```
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

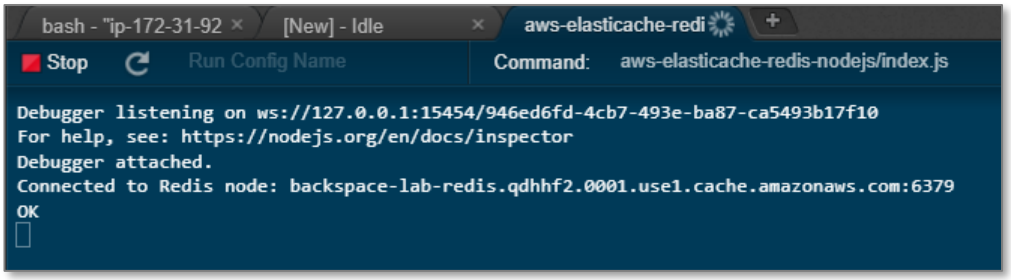
Create the new function which stores the high score for a game:

```
// 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 Ctrl + S to save the file

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



Stop the application.

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

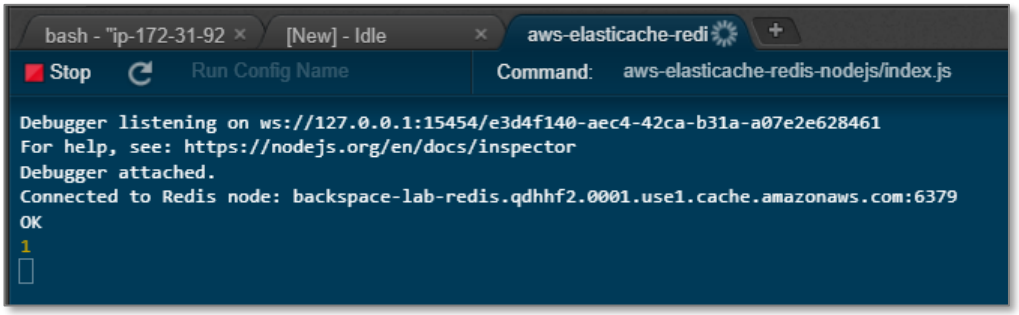
Create the new function which sets an expire time of 30 seconds for the key.

```
// 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 Ctrl + S to save the file

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



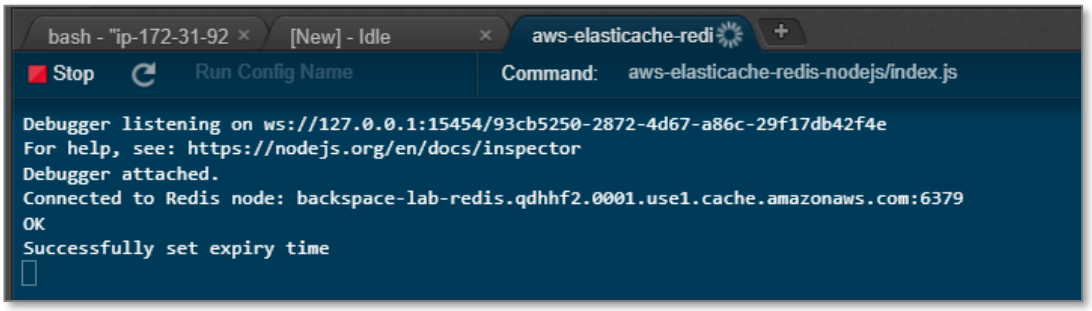
Stop the application.

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 Ctrl + S to save the file

Now run your application



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

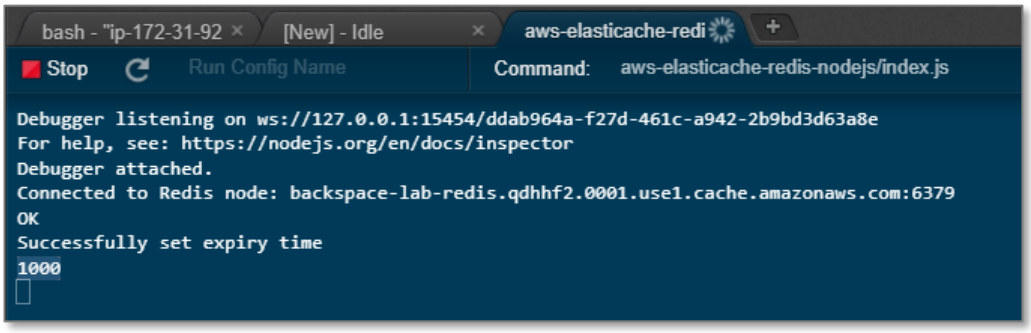
Create the new function which returns current the high score for a game.

```
// 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 Ctrl + S to save the file

Now run your application.



Stop the application.

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

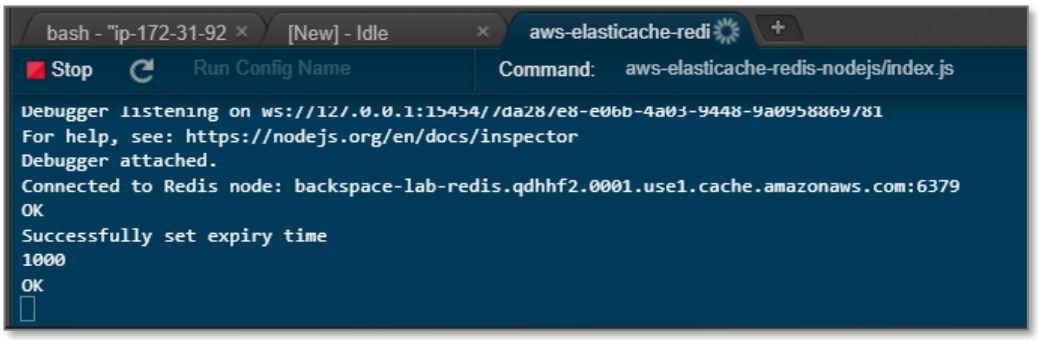
Create the new function which will create and save an object of keys.

```
// 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 Ctrl + S to save the file

Now run your application.



Stop the application.

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

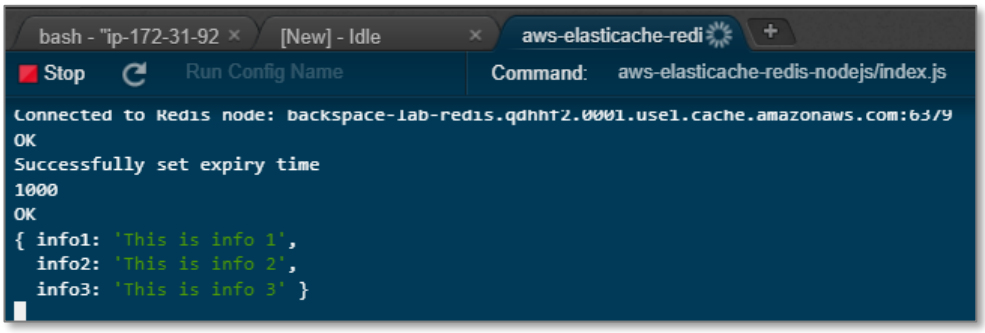
Create the new function which will read an object of keys.

```
// 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 Ctrl + S to save the file

Now run your application.



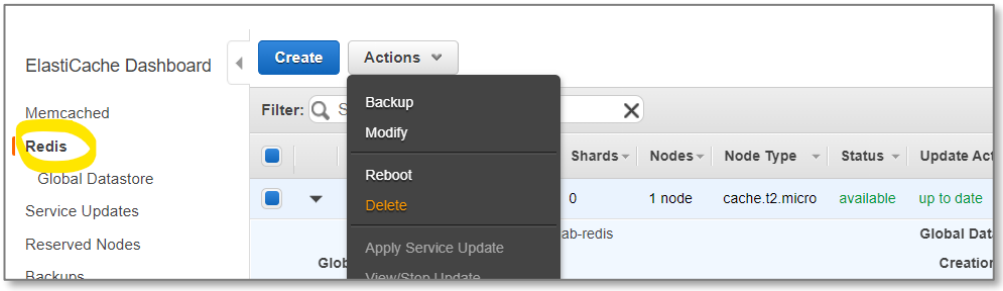
Stop the application.

Cleaning Up

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

Go to *Redis* and select your cluster

Click *Delete*

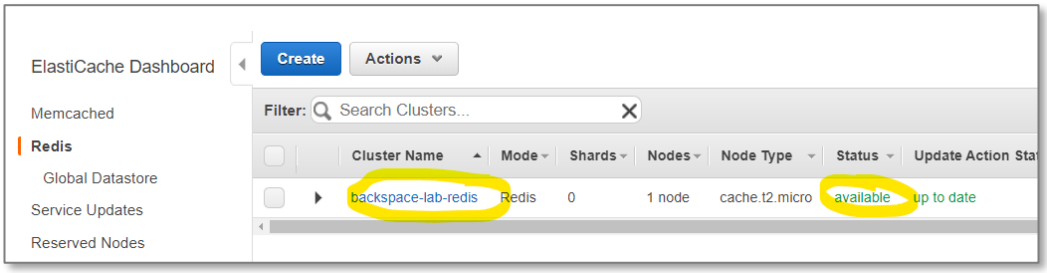


Connect to an ElastiCache Redis Cluster using Python

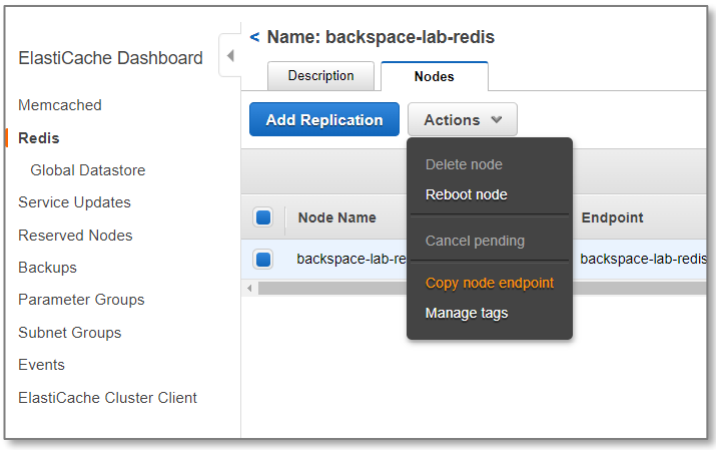
In this section we will connect to an ElastiCache Redis cluster using Python.

Wait for your Cluster status to be *available*

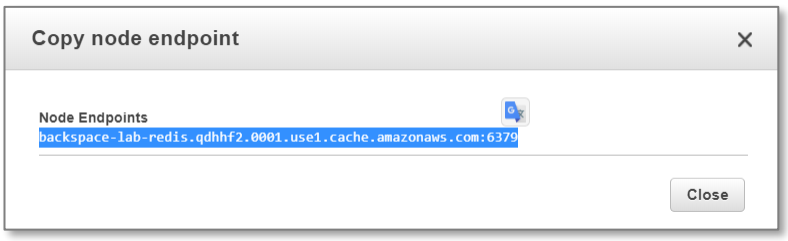
Click on the Cluster



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
```

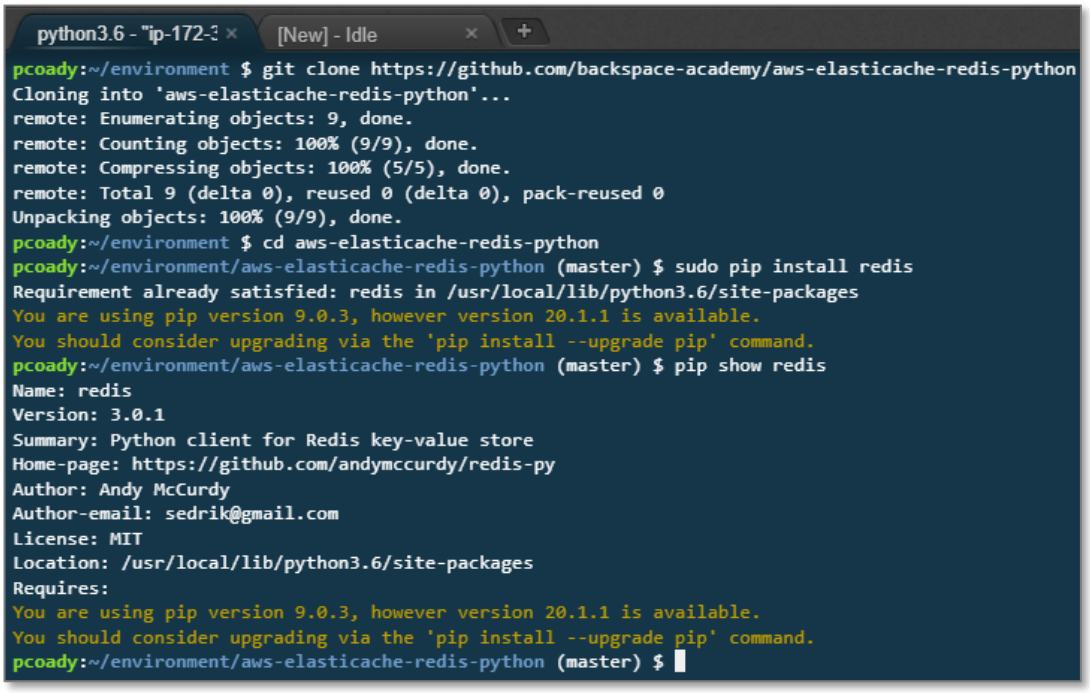
Install the dependencies

```
cd aws-elasticache-redis-python

sudo pip install redis
```

Check Redis client installed ok

```
pip show redis
```



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 Ctrl + S to save the file

Now run your application

Welcome | elasticache.py

```python
# Import the Redis package
import redis

redisEndpoint = 'backspace-lab-redis.qdhhf2.0001.use1.cache.amazonaws.com:6379'
# 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()
```

bash - "ip-172-31-92 | [New] - Idle | aws-elasticache-redi | aws-elasticache-redi

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>>>
```

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 × | [New] - Idle × | aws-elasticache-redi × | aws-elasticache-redi ×

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 Ctrl + S 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
```

bash - "ip-172-31-92 × | [New] - Idle × | aws-elasticache-redi × | aws-elasticache-redi ×

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


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 Ctrl + S 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 × [New] - Idle × aws-elasticache-redi × aws-elasticache-redi ×

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 Ctrl + S to save the file

Now run your application.

```python
# 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()
```

bash - "ip-172-31-92 × [New] - Idle × aws-elasticache-redi × aws-elasticache-redi ×

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
```

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.

```python
# 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 Ctrl + S to save the file

Now run your application.

```python
# 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()
```

```
bash - "ip-172-31-92 ×   [New] - Idle ×   aws-elasticache-redi ×   aws-elasticache-redi ×
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*