Learning Objectives

check\_circleMeasure the Baseline Performance

Start by taking a baseline performance measurement of the application. We will compare this with the Redis performance after our changes.

1. Log in to the AWS Management Console using the instructions above.
2. Search for **RDS** using the search bar. Select **Databases** on the left-hand side menu.
3. Select the instance under the **DB identifier** column.
4. Under **Endpoint** copy the endpoint URL. It will be used momentarily.
5. Switch over to a terminal and login to the bastion host (aka Cloud Server) using the credentials above.
6. Download the application file (app.py) and its configuration files (database.ini and app.sql) from the [Git repository](https://github.com/linuxacademy/content-aws-database-specialty.git); This can be downloaded to the bastion host with the following command:

git clone https://github.com/linuxacademy/content-aws-database-specialty.git

1. Create the app directory:

mkdir /home/cloud\_user/app/

Copy these three files into a directory named app:

cp /home/cloud\_user/content-aws-database-specialty/S08\_Using\ Elasticache\ to\ Improve\ Database\ Performance/{app.py,app.sql,database.ini} /home/cloud\_user/app/

1. Edit the database configuration:

cd app

vim database.ini

Set the hostname to the endpoint (copied earlier in this video and also paste in the password to the bastion host:

host=<YourRDSEndpoint>

password=<BastionHostPW>

1. Save and quit with :wq.
2. Additionally, save the hostname in the PGHOST variable:

export PGHOST=<YourRDSEndpoint>

1. Ensure you can connect to the database with the command below and enter in the bastion password:

psql -U postgres -h $PGHOST

Quit with \q.

1. Install the SQL files for the application and, once again, input the bastion password:

psql -U postgres -h $PGHOST -f app.sql

1. Specify the Python version for the application:

python2.7 app.py

1. Switch to a browser and input the bastion host IP into the URL bar, while navigating to the app directory, like so: **bastionip/app**

If successful, you'll see the Postgres version displayed on the screen.

1. Head over to your terminal and open a new window or tab and save the application URL as an environmental variable:

export APP\_URL=http://<bastionIP>/app

1. Test the URL using the cURL command:

curl -L $APP\_URL

Note: Need to install cURL on Mac? Try [this](http://macappstore.org/curl/).

1. Create the benchmark by using a for loop:

time (for i in {1..5}; do curl -L $APP\_URL;echo \n;done)

1. Once completed, note the seconds it took for these requests. In our lesson, it took 25.462 seconds.

check\_circleDeploy the Redis Cluster

Deploy a new Redis cluster to use as our cache.

1. In the AWS Management Console, go back to the home screen, (select AWS logo) and type in the search bar **VPC**.
2. Click on **Security Groups** in the left-hand side menu.
3. Note the first security group under the **security group ID**, when clicked will show **EC2SecurityGroup**(under details).
4. Select **Create Security Group**.
5. Fill out the basic details of the security group:
   * **Security group name: CacheSG**
   * **Description: cache security group**
6. In the **Inbound rule** section, click **Add rule**. Change **Port range** to **6379**.
7. Select the security group notated in **step 3** for *the box next to* **Source**.
8. In the **Tags** section, add a tag with the **key** of **name** and **value** of **Cache SG**.
9. Click **Create Security Group**. A confirmation of creation will display on the screen.
10. Go back to the AWS console's home screen, and search the for **ElastiCache** service in the search bar.
11. Click **Get Started Now** button.
12. Set:
    * **Cluster Engine: Redis**
    * **Name:Cache**
    * **Description:cache aside for app**
    * **Node type: t2 --> cache.t2.micro**
    * Select **Save**.
    * **Number of replicas: 1**

Name of **Advanced Redis settings**:

* + **Name:cache**
  + **Description: cache subnet group**
  + **Subnets:** Select two subnets that are different Availability Zones.

Under the **Security** section:

* + **Security group:** change the default to **Cache SG**

Under the **Backup** section:

* + **Enable backups: Uncheck (Disable)**
  + Select the **Create** button.

1. The cluster will take some time to create (5 ~7mins, so take a stretch!). Once complete, click the dropdown arrow next to the cluster name.
2. Copy the **Primary Endpoint**.
3. Switch back to the bastion host terminal. Exit the Python app (ctrl + c) and clear the screen (type clear).
4. Check the connection to the cluster by using the Python REPL:

python2.7

>>> import redis

>>> client = redis.Redis.from\_url('redis://<PrimaryEndpoint>')

>>> client.ping()

Exit the Python REPL (ctrl +d).

check\_circleUpdate the Application and Measure its Performance

Lastly, we update the application to use the Redis cluster as a "cache-aside" and re-measure the application's performance.

1. In the bastion host terminal, open the app.py file.

vim app.py

We alter the fetch method in this file to utilize the cache. The full version of this snippet is [GitHub files](https://github.com/linuxacademy/content-aws-database-specialty/tree/master/S08_Using%20Elasticache%20to%20Improve%20Database%20Performance) we downloaded earlier, named redis-app.py.

...

def fetch(sql):

ttl = 10

try:

params = config(section='redis')

cache = redis.Redis.from\_url(params['redis\_url'])

results cache.get(sql)

if result:

return result

else:

# connect to database listed in datbase.ini

conn = connect()

cur = conn.cursor()

cur.execute(sql)

# fetch one row

result = cur.fetchone()

print('Closing connection to database...')

cur.close()

conn.close()

# cache results

cache.setex(sql, ttl, ''.join(results))

return result

except(Exception, psycopg2.DatabaseError) as error;

print(error)

...

Save the file with :wq.

1. Add the Redis URL to the bottom of the database.ini configuration file.

vim database.ini

[redis]

redis\_url=redis://<PrimaryEndpoint>

Save the file with :wq.

1. Run the app:

python2.7 app.py

1. Go back to the browser with the bastion host URL and refresh the screen multiple times, around five.
2. Head to the terminal on your local machine, and run the for loop once more:

time (for i in {1..5}; do curl -L $APP\_URL;echo \n;done)

Note the new total time. This will be significantly less time than the original total time.