Practical: ElastiCache and memcached with Python (optional)

Table of Contents

ElastiCache is AWS's managed service for memcached and Redis. In this practical we will see how to set up a memcached managed instance and use it to implement a basic cache for an external resource.

This practical is optional

Prerequisites and references

- ElastiCache docs (https://docs.aws.amazon.com/elasticache/)
- Memcached homepage (https://memcached.org/)
- memcached npm module (https://www.npmjs.com/package/memcached)
- memcachedDemo.zip (https://canvas.qut.edu.au/courses/20367/files/6583686/download)

Creating an elasticache instance

- Find ElastiCache on the AWS console
- Go to the *Memcached caches* list (on left side panel)
- *Create Memcached cache*
- Choose *Design your own cache* and *Easy create*
- Under Configuration choose *Demo*
- Under *Connectivity Subnet group* choose *CAB432-subnets*
- Add tags for your qut-username as usual
- Create the cache

It will take a few minutes to create the cache.

- In the sidebar, click *Memcached caches*
- In the list of memcached instances click on your instance. You may need to wait a few minutes before the cache is available for you to modify.
- Click on the *Network and security* tab:
  - Under *Security groups* click *Modify* on the right
  - Under *Security* click *Manage* on the right
  - Tick the box beside *CAB432MemcachedSG*

    - Click *Choose*
    - Click *Preview changes*
    - Click *Modify*
- You will be back in the list of memcached caches. Click on your cache again
- Copy the *Configuration endpoint* value. You will need this for connecting. You might need to wait a minute or two for this to be populated. Go back to the list and click on your instance again to refresh

Please note:

- The *CAB432MemcachedSG* security group is configured to allow access to your memcached cache from EC2 instances in the *CAB432SG* security group. If the cache and EC2 instance don't have their security groups configured properly then you will not be able to connect to the cache from your EC2 instance
- Memcached caches are not accessible from outside the AWS environment. You can only access them from EC2 instances (or later, containers and Lambdas)

Caching external resources

For Python we'll use the `pymemcache` library. To begin, we'll need to get some things ready:

- Create an EC2 instance. Make sure that it is in the `CAB432SG` security group.
- Log in to your EC2 instance and run the following commands

```
sudo apt install python3.12-venv
python3 -m venv venv
source venv/bin/activate
pip install pymemcache requests
```

- Create a file called `memcached.py` on your EC2 instance with the following contents:

```
import time
import requests
from pymemcache.client.base import Client

MEMCACHED_ENDPOINT = "past your endpoint address here, including the
:11211 at the end"
URL = "https://pymemcache.readthedocs.io/en/latest/getting_started.ht
ml"
CACHE_TTL = 60  # Cache time-to-live in seconds

# Global memcached client
memcached_client = Client(MEMCACHED_ENDPOINT)

def cached_fetch(url):
    print(f"Fetching {url}")
    value = memcached_client.get(url)
    if value:
        print("Cache hit")
        return value.decode('utf-8')
```

```
    print("Cache miss.  Fetching from URL")
    response = requests.get(url)
    # We need to encode the string, as pymemcache expects ASCII only
    fetched_value = response.text.encode()
    print(f"Fetched {len(fetched_value)} bytes")
    memcached_client.set(url, fetched_value, expire=CACHE_TTL)
    print("Stored in cache")
    return fetched_value

for i in range(10):
    start = time.time()
    print(f"Fetch {i}:")
    try:
        res = cached_fetch(URL)
        print(f"cached: {len(res)} bytes")
        cached_fetch(URL)
        print(f"Fetch {i} completed")
    except Exception as e:
        print("Error occurred")
        print(e)
    if i == 0:
        print(f"not cached: {time.time() - start:.2f}s")
    else:
        print(f"cached: {time.time() - start:.2f}s")
```

- Edit the `MEMCACHE_ENDPOINT` variable to set it to your *Configuration endpoint* value from earlier
- Run `python3 memcache.py`

You should see several fetches to the external URL, with the first being a cache miss and the later ones obtained from the cache. Note the difference in speed.

Delete your ElastiCache instance

If you are done with your ElastiCache instance, please delete it. You are also welcome to keep it around if you will be using it with your assessment, in which case please update the `purpose` tag to indicate this.