Marwadi University Marwadi Chandarana Group	**Marwadi University** **Faculty of Engineering and Technology** **Department of Information and Communication Technology**	
Subject: Cloud Developing (01CT0720)	**Aim: Caching application data with ElastiCache.**	
Experiment No: 09	**Date:**	**Enrolment No: 92200133030**

Aim :- Caching application data with ElastiCache.

Lab overview and objectives

In this lab, you will deploy an Amazon ElastiCache cluster. You will also test synchronizing the cache with the Amazon Aurora Serverless database by using Python scripts. Finally, you will update the coffee suppliers application with new Node.js code that will use data caching.

After completing this lab, you should be able to:

- Create a new ElastiCache for Memcached cluster
- Query the ElastiCache for Memcached cluster by using Python and the pymemcache client

AWS service restrictions

In this lab environment, access to AWS services and service actions might be restricted to the ones that are needed to complete the lab instructions. You might encounter errors if you attempt to access other services or perform actions beyond the ones that are described in this lab.

Scenario

Frank and Martha are excited that the bean inventory from the coffee suppliers application is integrated into the café website. Unfortunately, they are getting occasional negative feedback from customers who say that the bean inventory information takes too long to display on the page. Frank and Martha have asked Sofía if she can have the site display the information more quickly.

Luckily, one of the café's customers, Olivia, is an AWS consultant and has a lot of database experience. She has offered to help with the website. Olivia proposed adding database caching to improve the speed of the database query. This way, frequently used data would be stored in memory instead of having to be retrieved from the database, which could also require information to be read from database storage.

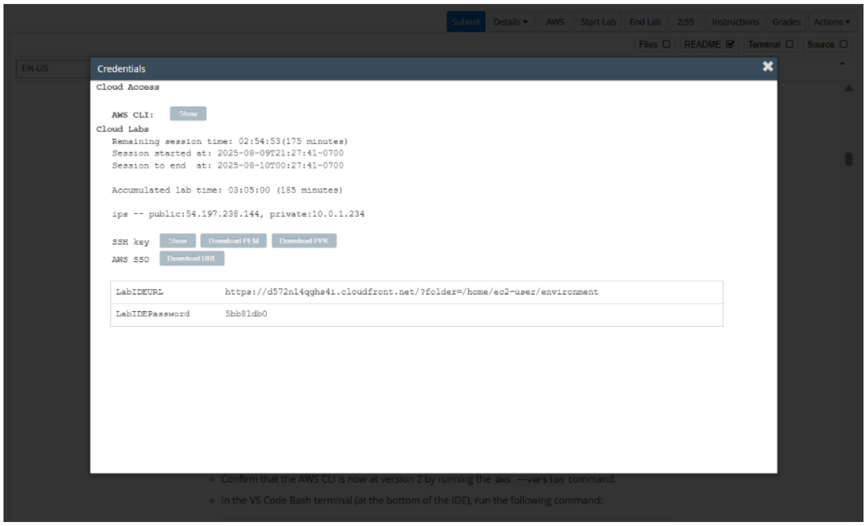
Olivia explained that you could cache data locally on the Amazon Elastic Compute Cloud (Amazon EC2) instance that is hosting the application containers. However, that architecture would not provide fault tolerance. This strategy would also require more administration because you would need to install and maintain the caching software yourself. Olivia recommended using the Amazon ElastiCache managed service. ElastiCache is similar to Aurora Serverless in that it simplifies the process to deploy and maintain a cache cluster.

Task 1: Preparing the lab

Connect to the VS Code IDE.

1. At the top of these instructions, choose Details followed by **AWS: Show**
2. Copy values from the table **similar** to the following and paste it into an editor of your choice for use later.
   a. **LabIDEURL**

b. **LabIDEPassword**

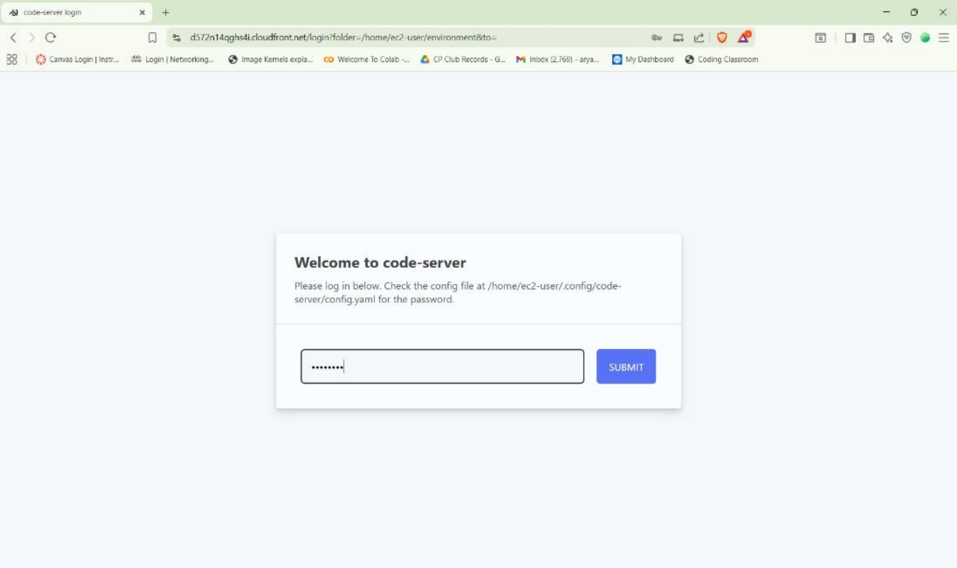


```
Cloud Access

   AWS CLI:
Cloud Labs
   Remaining session time: 02:54:53(175 minutes)
   Session started at: 2025-08-09T21:27:41-0700
   Session to end  at: 2025-08-10T00:27:41-0700

   Accumulated lab time: 03:05:00 (185 minutes)

   ips -- public:54.197.238.144, private:10.0.1.234

   SSH key
   AWS SSO
```

LabIDEURL	https://d572n14qghs4i.cloudfront.net/?folder=/home/ec2-user/environment
LabIDEPassword	5bb81db0

- Confirm that the AWS CLI is now at version 2 by running the aws --version command.
- In the VS Code Bash terminal (at the bottom of the IDE), run the following command:

3. In a new browser tab, paste the value for **LabIDEURL** to open the VS Code IDE.
4. On the prompt window **Welcome to code-server**, enter the value for **LabIDEPassword** you copied to the editor earlier, choose **Submit** to open the VS Code IDE.



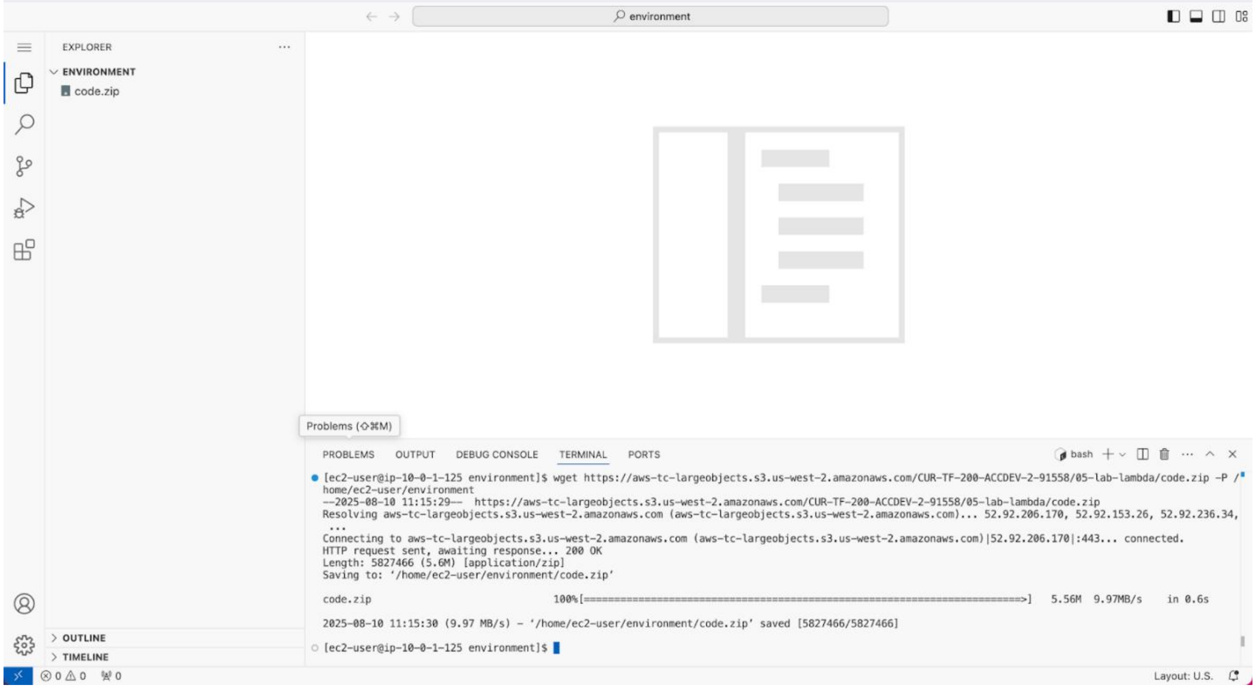
Welcome to code-server

Please log in below. Check the config file at /home/ec2-user/.config/code-server/config.yaml for the password.

••••••••

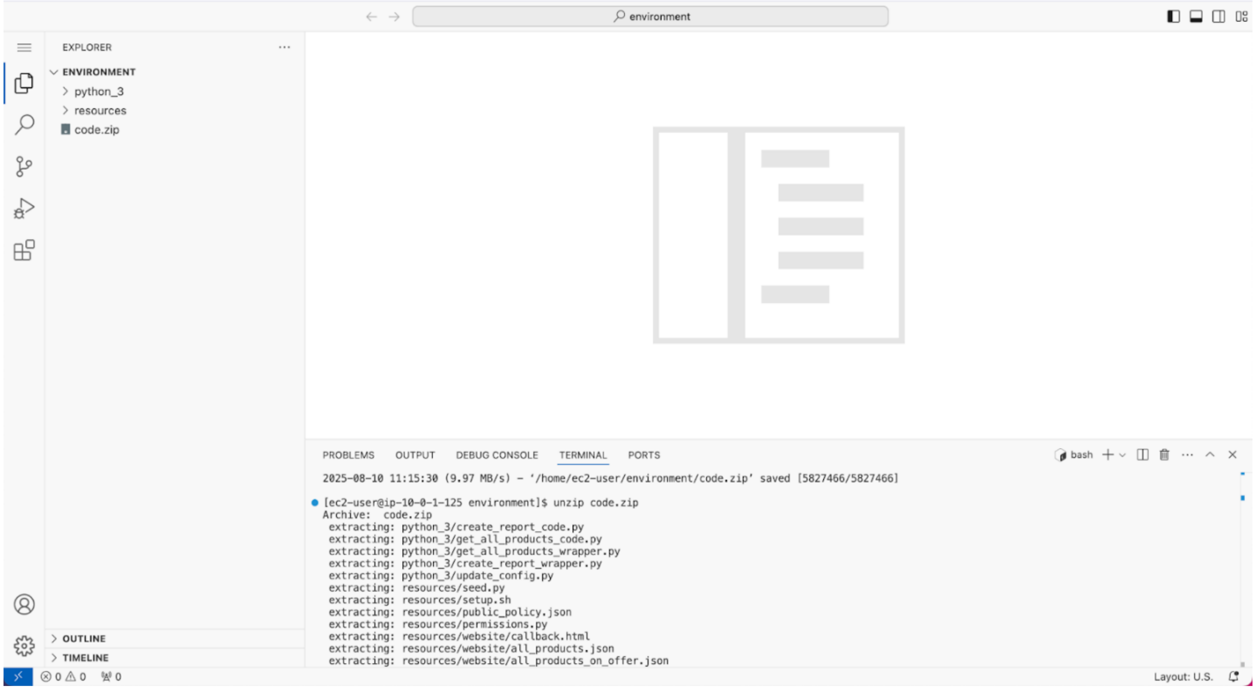
5. Download and extract the files that you need for this lab.
   - In the VS Code bash terminal (located at the bottom of the IDE), run the following commands:

```
wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-
ACCDEV-2-91558/08-lab-db-caching/code.zip -P /home/ec2-user/environment
```



```
● [ec2-user@ip-10-0-1-125 environment]$ wget https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/05-lab-lambda/code.zip -P /
home/ec2-user/environment
--2025-08-10 11:15:29--  https://aws-tc-largeobjects.s3.us-west-2.amazonaws.com/CUR-TF-200-ACCDEV-2-91558/05-lab-lambda/code.zip
Resolving aws-tc-largeobjects.s3.us-west-2.amazonaws.com (aws-tc-largeobjects.s3.us-west-2.amazonaws.com)... 52.92.206.170, 52.92.153.26, 52.92.236.34,
 ...
Connecting to aws-tc-largeobjects.s3.us-west-2.amazonaws.com (aws-tc-largeobjects.s3.us-west-2.amazonaws.com)|52.92.206.170|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 5827466 (5.6M) [application/zip]
Saving to: '/home/ec2-user/environment/code.zip'

code.zip                      100%[==========================================================================>]   5.56M  9.97MB/s    in 0.6s

2025-08-10 11:15:30 (9.97 MB/s) - '/home/ec2-user/environment/code.zip' saved [5827466/5827466]

○ [ec2-user@ip-10-0-1-125 environment]$ 
```

6. You should see that the **code.zip** file was downloaded to the VS Code IDE and is now in the left navigation pane.
    - Extract the file by running the following command:
      `unzip code.zip`

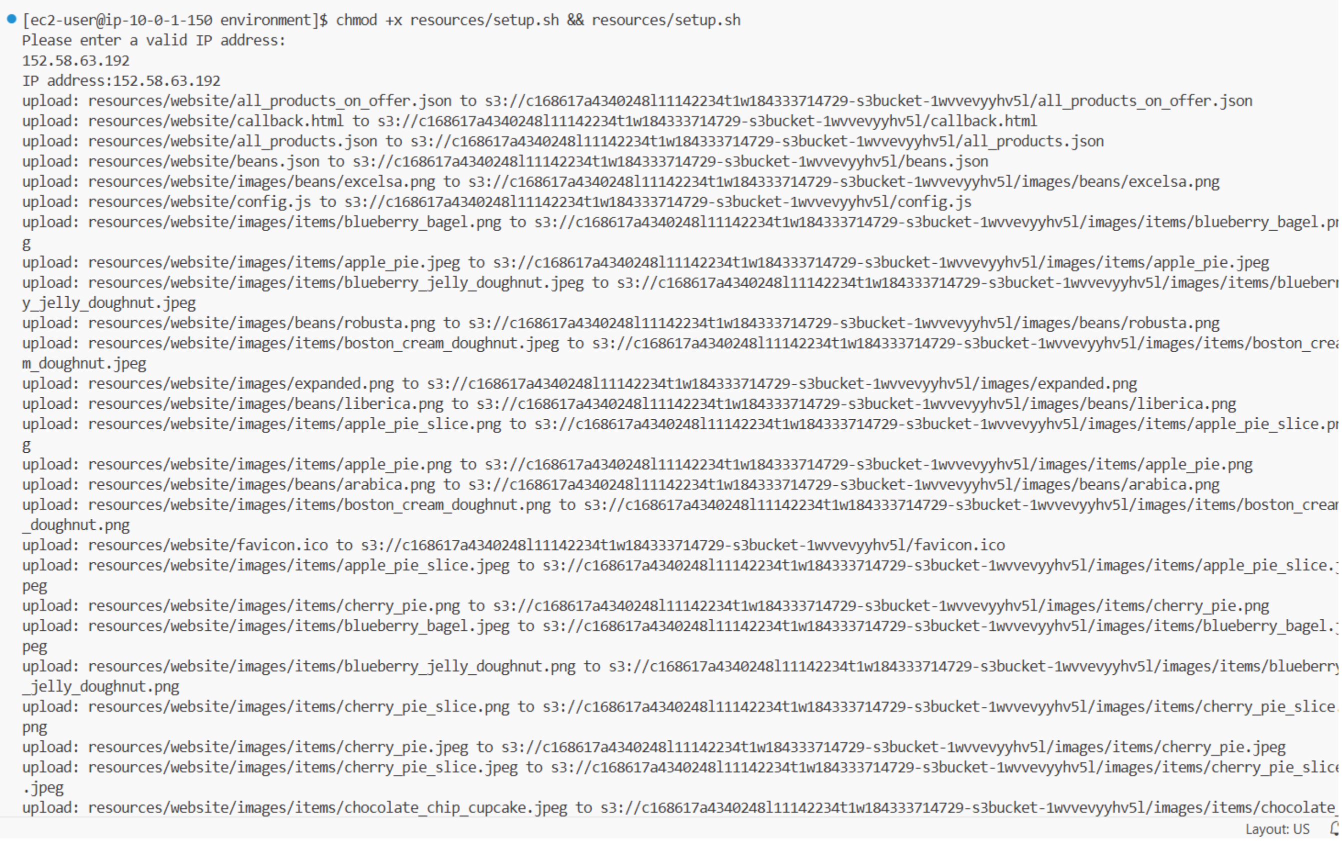


```
2025-08-10 11:15:30 (9.97 MB/s) - '/home/ec2-user/environment/code.zip' saved [5827466/5827466]

● [ec2-user@ip-10-0-1-125 environment]$ unzip code.zip
Archive:  code.zip
  extracting: python_3/create_report_code.py
  extracting: python_3/get_all_products_code.py
  extracting: python_3/get_all_products_wrapper.py
  extracting: python_3/create_report_wrapper.py
  extracting: python_3/update_config.py
  extracting: resources/seed.py
  extracting: resources/setup.sh
  extracting: resources/public_policy.json
  extracting: resources/permissions.py
  extracting: resources/website/callback.html
  extracting: resources/website/all_products.json
  extracting: resources/website/all_products_on_offer.json
```

7. Run a script that upgrades the version of the AWS CLI installed on the VS Code IDE.
    - To set permissions on the script and then run it, run the following commands in the Bash terminal:

```
chmod +x ./resources/setup.sh && ./resources/setup.sh
```

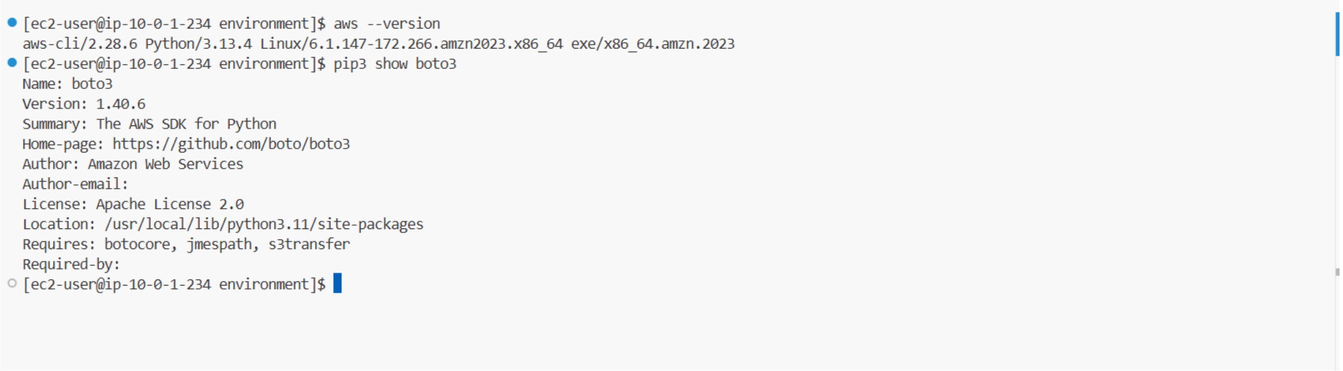
The script will prompt you for the **IP address** by which your computer is known to the internet.
Use www.whatismyip.com to discover this address and then paste the IPv4 address into the command prompt and finish running the script.



```
● [ec2-user@ip-10-0-1-150 environment]$ chmod +x resources/setup.sh && resources/setup.sh
  Please enter a valid IP address:
  152.58.63.192
  IP address:152.58.63.192
  upload: resources/website/all_products_on_offer.json to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/all_products_on_offer.json
  upload: resources/website/callback.html to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/callback.html
  upload: resources/website/all_products.json to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/all_products.json
  upload: resources/website/beans.json to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/beans.json
  upload: resources/website/images/beans/excelsa.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/beans/excelsa.png
  upload: resources/website/config.js to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/config.js
  upload: resources/website/images/items/blueberry_bagel.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/blueberry_bagel.pn
  g
  upload: resources/website/images/items/apple_pie.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/apple_pie.jpeg
  upload: resources/website/images/items/blueberry_jelly_doughnut.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/blueberr
  y_jelly_doughnut.jpeg
  upload: resources/website/images/beans/robusta.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/beans/robusta.png
  upload: resources/website/images/items/boston_cream_doughnut.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/boston_crea
  m_doughnut.jpeg
  upload: resources/website/images/expanded.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/expanded.png
  upload: resources/website/images/beans/liberica.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/beans/liberica.png
  upload: resources/website/images/items/apple_pie_slice.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/apple_pie_slice.pn
  g
  upload: resources/website/images/items/apple_pie.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/apple_pie.png
  upload: resources/website/images/beans/arabica.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/beans/arabica.png
  upload: resources/website/images/items/boston_cream_doughnut.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/boston_cream
  _doughnut.png
  upload: resources/website/favicon.ico to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/favicon.ico
  upload: resources/website/images/items/apple_pie_slice.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/apple_pie_slice.j
  peg
  upload: resources/website/images/items/cherry_pie.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/cherry_pie.png
  upload: resources/website/images/items/blueberry_bagel.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/blueberry_bagel.j
  peg
  upload: resources/website/images/items/blueberry_jelly_doughnut.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/blueberry
  _jelly_doughnut.png
  upload: resources/website/images/items/cherry_pie_slice.png to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/cherry_pie_slice.
  png
  upload: resources/website/images/items/cherry_pie.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/cherry_pie.jpeg
  upload: resources/website/images/items/cherry_pie_slice.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/cherry_pie_slice
  .jpeg
  upload: resources/website/images/items/chocolate_chip_cupcake.jpeg to s3://c168617a4340248l11142234t1w184333714729-s3bucket-1wvvevyyhv5l/images/items/chocolate_
```

8. Verify the AWS CLI version and also verify that the SDK for Python is installed.
    - Confirm that the AWS CLI is now at version 2 by running the **`aws --version`** command.
    - In the VS Code Bash terminal (at the bottom of the IDE), run the following command:

```
pip3 show boto3
```



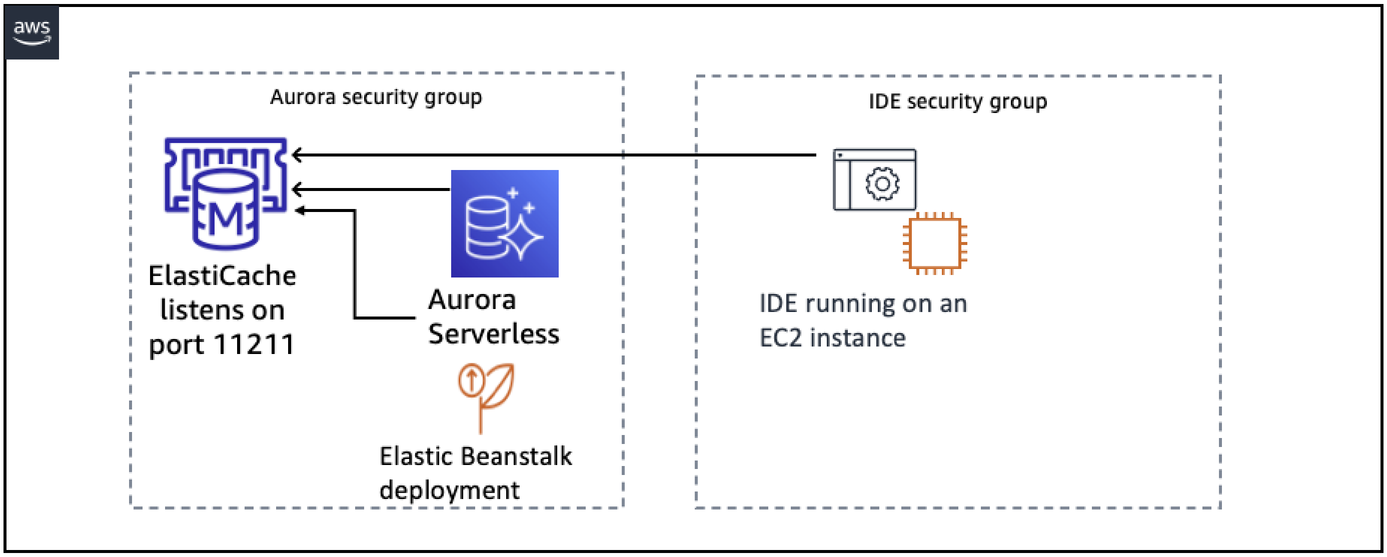
```
[ec2-user@ip-10-0-1-234 environment]$ aws --version
aws-cli/2.28.6 Python/3.13.4 Linux/6.1.147-172.266.amzn2023.x86_64 exe/x86_64.amzn.2023
[ec2-user@ip-10-0-1-234 environment]$ pip3 show boto3
Name: boto3
Version: 1.40.6
Summary: The AWS SDK for Python
Home-page: https://github.com/boto/boto3
Author: Amazon Web Services
Author-email:
License: Apache License 2.0
Location: /usr/local/lib/python3.11/site-packages
Requires: botocore, jmespath, s3transfer
Required-by:
[ec2-user@ip-10-0-1-234 environment]$
```

Task 2: Configuring the subnets for ElastiCache to use

The Memcached database must be able to communicate with the Aurora Serverless database. In addition, the application needs to be able to access both the Memcached database and the Aurora Serverless database.

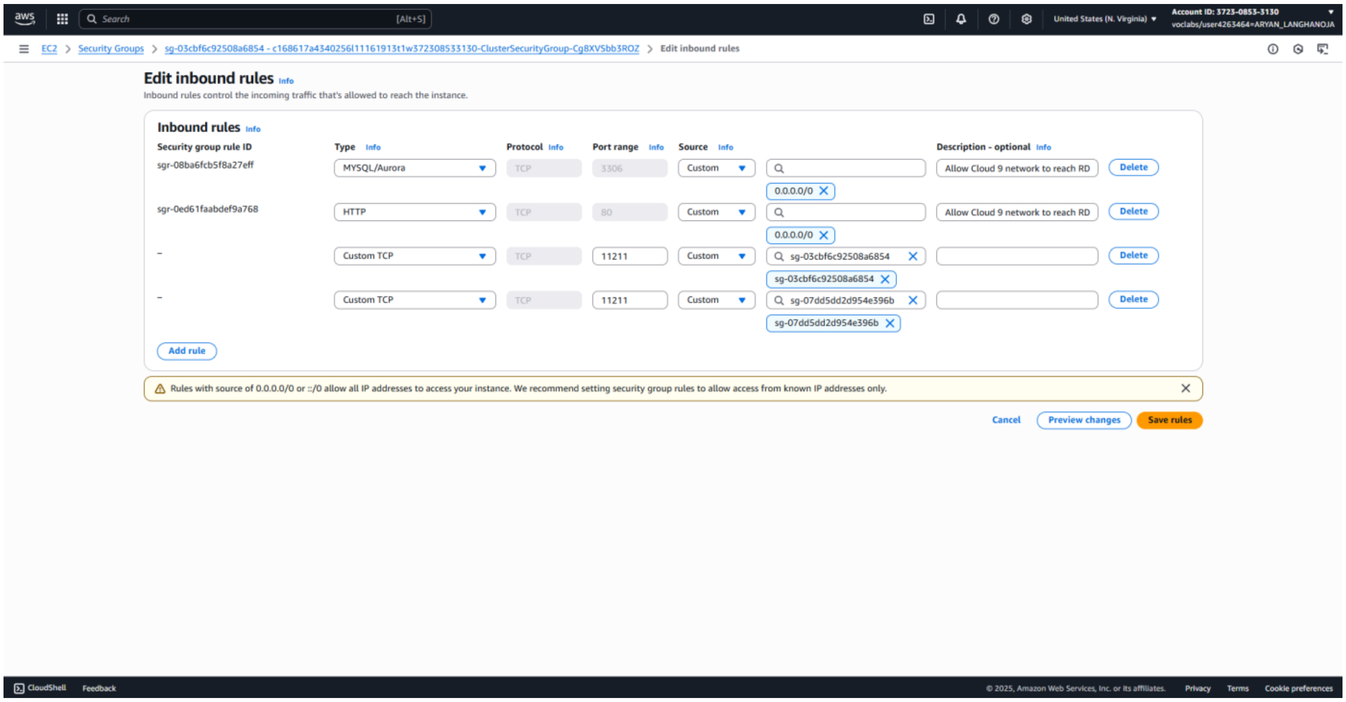
In this task, you will act as Olivia to configure a security group to enable the necessary communication between the application, database, and cache. You will use the security group that the Aurora Serverless cluster is already using rather than creating an additional security group.

9. Configure the security group that the Aurora Serverless instance will use, as shown in the following diagram.



aws
Aurora security group
ElastiCache listens on port 11211
Aurora Serverless
Elastic Beanstalk deployment
IDE security group
IDE running on an EC2 instance

- Return to the AWS Management Console browser tab.
- From the **Services** menu, choose **EC2**.
- In the left navigation pane, choose **Security Groups**.
- Select the checkbox for the security group that has *Aurora* in the name.
- In the lower pane, choose the **Inbound rules** tab.
- Choose **Edit inbound rules**.
  - Add an entry to enable access to the Memcached port from all sources that are assigned to the *Aurora* security group:
    - Choose **Add rule**
    - **Type:** Choose **Custom TCP**
    - **Port range:** Enter 11211
    - **Source:** Choose the security group with *Aurora* in the name. This is the same group that you are currently editing.
- Add an entry to enable access to the Memcached port from all sources that are assigned to the *Lab IDE SG* security group:
  - Choose **Add rule**
  - **Type:** Choose **Custom TCP**
  - **Port range:** Enter 11211
  - **Source:** Choose the security group with *Lab IDE SG* in the name
- Choose **Save rules**.



aws
Search
[Alt+S]
United States (N. Virginia) ▾
Account ID: 3723-0853-3130
voclabs/user4263464=ARYAN_LANGHANOJA
EC2 > Security Groups > sg-03cbf6c92508a6854 - c168617a4340256l11161913t1w372308533130-ClusterSecurityGroup-Cg8XVSbb3ROZ > Edit inbound rules
Info
Add rule
Cancel
Preview changes
Save rules
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Edit inbound rules

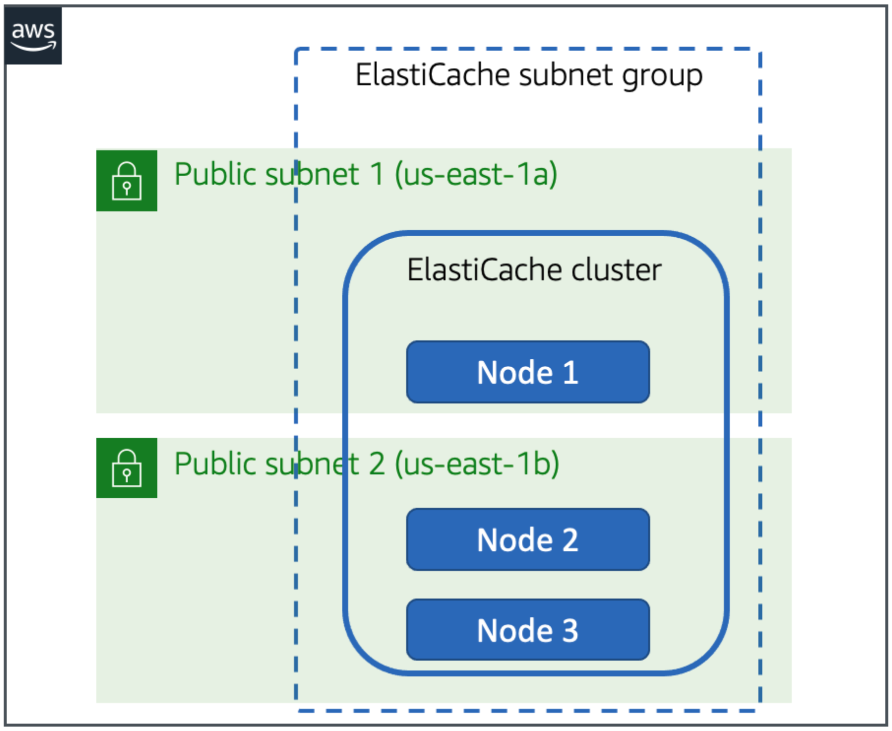
Inbound rules control the incoming traffic that's allowed to reach the instance.

Inbound rules

Security group rule ID	Type	Protocol	Port range	Source		Description - optional	
sgr-08ba6fcb5f8a27eff	MYSQL/Aurora	TCP	3306	Custom	0.0.0.0/0	Allow Cloud 9 network to reach RD	Delete
sgr-0ed61faabdef9a768	HTTP	TCP	80	Custom	0.0.0.0/0	Allow Cloud 9 network to reach RD	Delete
-	Custom TCP	TCP	11211	Custom	sg-03cbf6c92508a6854 sg-03cbf6c92508a6854		Delete
-	Custom TCP	TCP	11211	Custom	sg-07dd5dd2d954e396b sg-07dd5dd2d954e396b		Delete

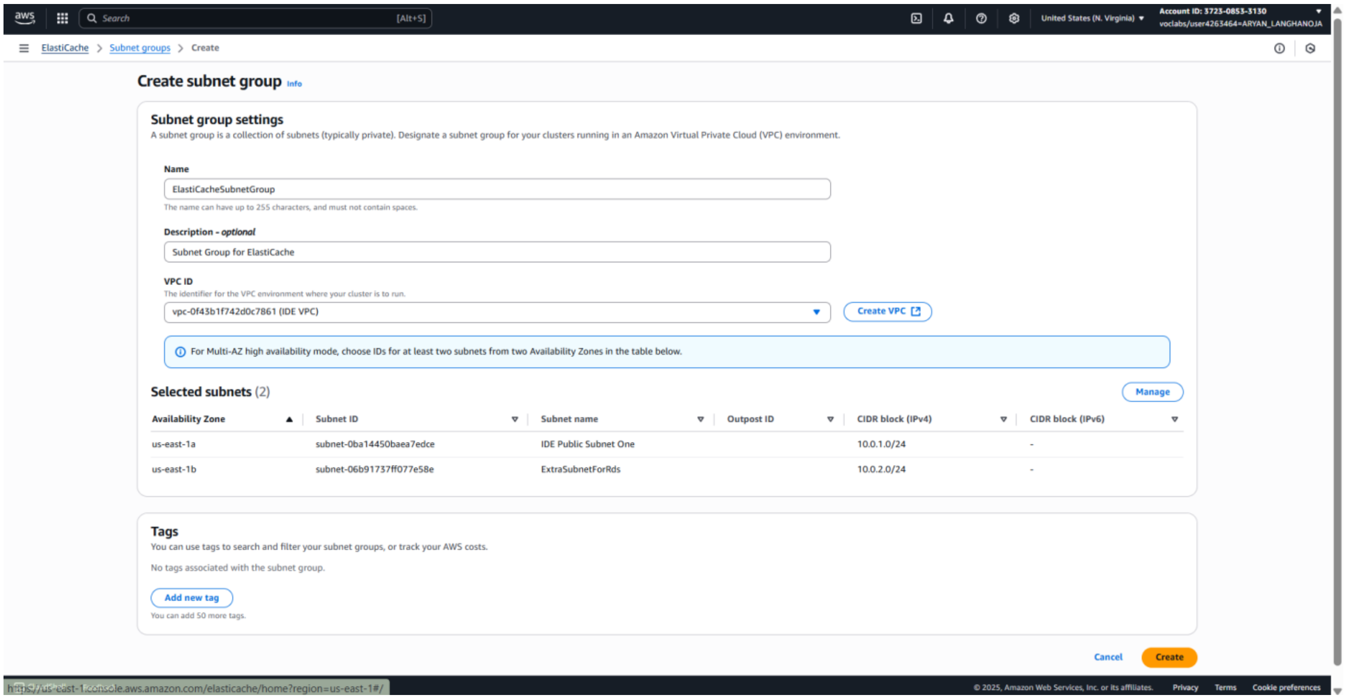
⚠ Rules with source of 0.0.0.0/0 or ::/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

10. Configure the database subnet group that the Aurora Serverless instance will use, as shown in the following diagram.



- In the search box at the top of the AWS management console, type and choose ElastiCache.
- From the left navigation pane, choose **Subnet groups**.
- Choose **Create subnet group**.
  - Configure the following settings:
    - **Name:** Enter ElastiCacheSubnetGroup
    - **Description:** Enter Subnet Group for ElastiCache
    - **VPC ID:** From the dropdown, choose **IDE VPC**
    - Notice that two subnets are already chosen for the cluster **Selected subnets (2)**
      - If you do not see this, choose **Manage** button on the right and select two subnets.
    - Choose **Create**.
      - Message is displayed stating that *The subnet group was created successfully.*

Next, you will deploy an ElastiCache cluster.



Search [Alt+S]

United States (N. Virginia) ▾ Account ID: 3723-0853-3130 voclabs/user4263464=ARYAN_LANGHANOJA

ElastiCache > Subnet groups > Create

Create subnet group Info

Subnet group settings

A subnet group is a collection of subnets (typically private). Designate a subnet group for your clusters running in an Amazon Virtual Private Cloud (VPC) environment.

Name

ElastiCacheSubnetGroup

The name can have up to 255 characters, and must not contain spaces.

Description - *optional*

Subnet Group for ElastiCache

VPC ID

The identifier for the VPC environment where your cluster is to run.

vpc-0f43b1f742d0c7861 (IDE VPC) ▾ Create VPC

ⓘ For Multi-AZ high availability mode, choose IDs for at least two subnets from two Availability Zones in the table below.

Selected subnets (2) Manage

Availability Zone	Subnet ID	Subnet name	Outpost ID	CIDR block (IPv4)	CIDR block (IPv6)
us-east-1a	subnet-0ba14450baea7edce	IDE Public Subnet One		10.0.1.0/24	-
us-east-1b	subnet-06b91737ff077e58e	ExtraSubnetForRds		10.0.2.0/24	-

Tags

You can use tags to search and filter your subnet groups, or track your AWS costs.

No tags associated with the subnet group.

Add new tag

You can add 50 more tags.

Cancel Create

https://us-east-1.console.aws.amazon.com/elasticache/home?region=us-east-1#/

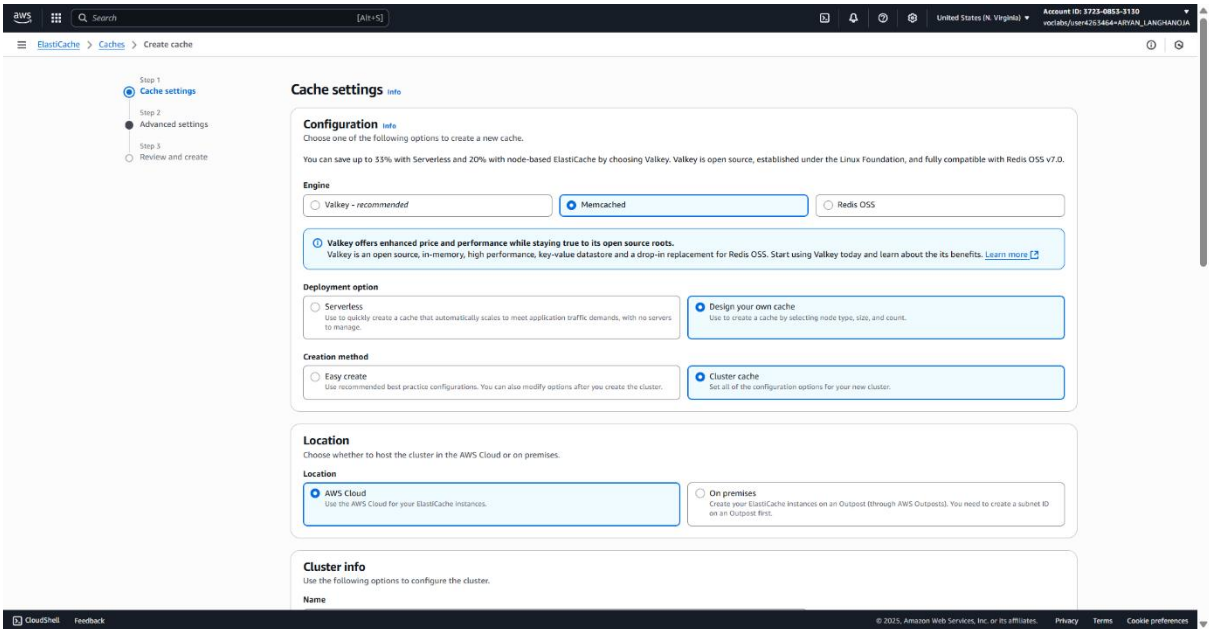
© 2025, Amazon Web Services, Inc. or its affiliates. Privacy Terms Cookie preferences

Task 3: Creating the ElastiCache cluster

As with the database deployment in a previous lab, the café has chosen to deploy a managed service instead of manually installing software in a container or on an EC2 instance. ElastiCache offers two engine options, Memcached and Redis. Olivia decided to use Memcached for the coffee suppliers application because the application does not require advanced features that Redis offers. Memcached will be a simple way to get started with caching, and the café could move to Redis later if needed.
In this task, you will again act as Olivia to create an ElastiCache for Memcached cluster.

12. Create an ElastiCache for Memcached instance.
    - Remain in the ElastiCache console.
    - In the left navigation pane, choose **Memcached caches**.
    - Choose **Create Memcached cache**.
    - Configure the ElastiCache cluster with the following settings:
        - In the **Choose a cluster creation method** section:
            - Choose: **Design your own cache**.
            - Choose: **Standard create**.
            - In the **Location** section, choose **AWS Cloud**.
        - In the **Cluster info** section:
            - **Name:** Enter MemcachedCache
        - In the **Cluster settings** section:
            - **Engine version:** Choose **1.6.6**
            - **Port:** Enter 11211 (If not already mentioned).
            - **Parameter group:** Choose **default.memcached.1.6**
            - **Node type:** Choose **cache.r6g.large (13.07GiB)**

  - **Number of nodes:** Enter 3
- Leave the **Connectivity** section at default values.
- Choose **Next**.
- On the **Advanced settings** page
  - For **Selected security groups (0)** choose **Manage**.
  - Select the Security group having *ClusterSecurityGroup* in the name.
- Choose **Next** .
- Choose **Create**.


aws
Search [Alt+S]
United States (N. Virginia)
Account ID: 3723-0853-3130
voclabs/user4263464=ARYAN_LANGHANOJA
ElastiCache > Caches > Create cache
Step 1
Cache settings
Step 2
Advanced settings
Step 3
Review and create
Info
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Cache settings

Configuration

Choose one of the following options to create a new cache.

You can save up to 33% with Serverless and 20% with node-based ElastiCache by choosing Valkey. Valkey is open source, established under the Linux Foundation, and fully compatible with Redis OSS v7.0.

Engine

○ Valkey - *recommended* ● Memcached ○ Redis OSS

Valkey offers enhanced price and performance while staying true to its open source roots.
Valkey is an open source, in-memory, high performance, key-value datastore and a drop-in replacement for Redis OSS. Start using Valkey today and learn about the its benefits. Learn more

Deployment option

○ Serverless
Use to quickly create a cache that automatically scales to meet application traffic demands, with no servers to manage.

● Design your own cache
Use to create a cache by selecting node type, size, and count.

Creation method

○ Easy create
Use recommended best practice configurations. You can also modify options after you create the cluster.

● Cluster cache
Set all of the configuration options for your new cluster.

Location

Choose whether to host the cluster in the AWS Cloud or on premises.

Location

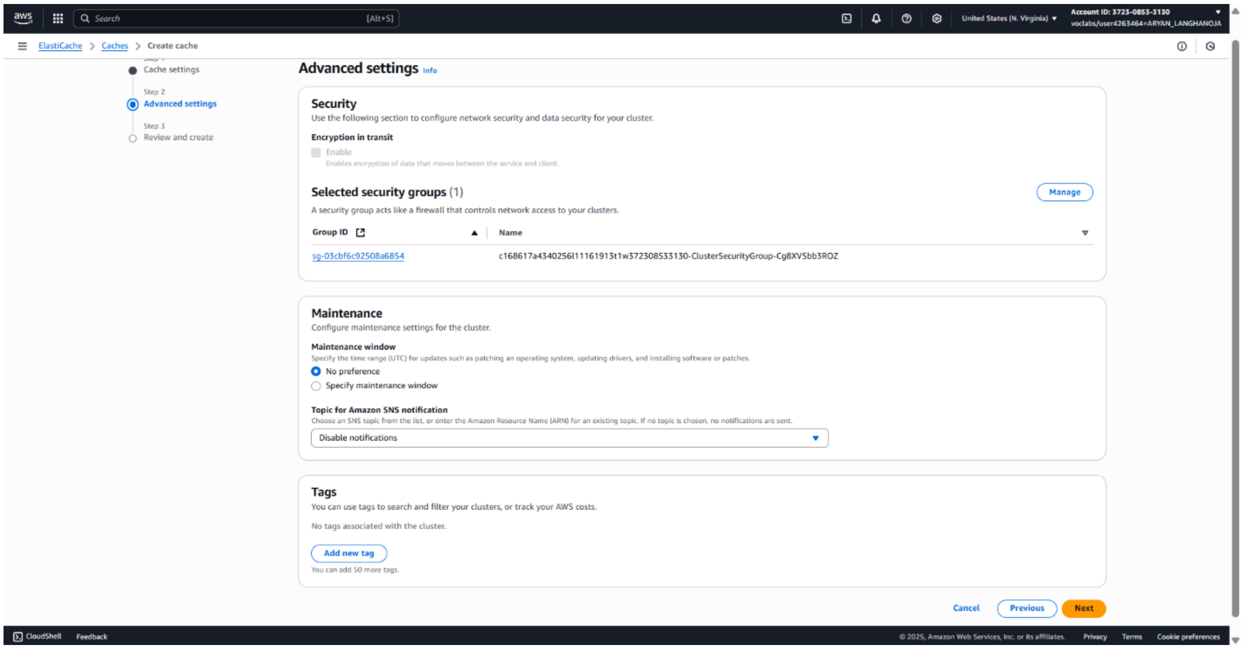
● AWS Cloud
Use the AWS Cloud for your ElastiCache instances.

○ On premises
Create your ElastiCache instances on an Outpost (through AWS Outposts). You need to create a subnet ID on an Outpost first.

Cluster info

Use the following options to configure the cluster.

Name


aws
Search [Alt+S]
United States (N. Virginia)
Account ID: 3723-0853-3130
voclabs/user4263464=ARYAN_LANGHANOJA
ElastiCache > Caches > Create cache
Cache settings
Step 2
Advanced settings
Step 3
Review and create
Info
Manage
Add new tag
Cancel
Previous
Next
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Advanced settings

Security

Use the following section to configure network security and data security for your cluster.

Encryption in transit

□ Enable
Enables encryption of data that moves between the service and client.

Selected security groups (1)

A security group acts like a firewall that controls network access to your clusters.

Group ID	Name
sg-03cbf6c92508a6854	c168617a4340256l11161913t1w372308533130-ClusterSecurityGroup-Cg8XVSbb3ROZ

Maintenance

Configure maintenance settings for the cluster.

Maintenance window

Specify the time range (UTC) for updates such as patching an operating system, updating drivers, and installing software or patches.

● No preference
○ Specify maintenance window

Topic for Amazon SNS notification

Choose an SNS topic from the list, or enter the Amazon Resource Name (ARN) for an existing topic. If no topic is chosen, no notifications are sent.

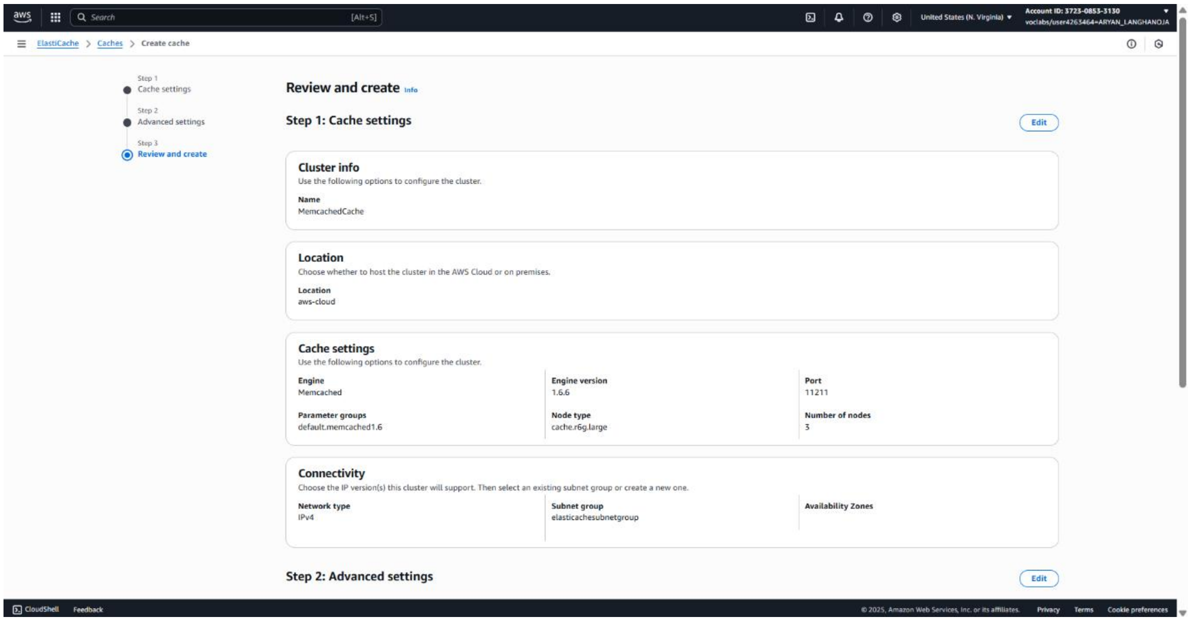
Disable notifications

Tags

You can use tags to search and filter your clusters, or track your AWS costs.

No tags associated with the cluster.

You can add 50 more tags.



ElastiCache > Caches > Create cache

Step 1 Cache settings
Step 2 Advanced settings
Step 3 Review and create

Review and create Info

Step 1: Cache settings

Cluster info
Use the following options to configure the cluster.
Name: MemcachedCache

Location
Choose whether to host the cluster in the AWS Cloud or on premises.
Location: aws-cloud

Cache settings
Use the following options to configure the cluster.

Engine	Engine version	Port
Memcached	1.6.6	11211
Parameter groups	**Node type**	**Number of nodes**
default.memcached1.6	cache.r6g.large	3

Connectivity
Choose the IP version(s) this cluster will support. Then select an existing subnet group or create a new one.

Network type	Subnet group	Availability Zones
IPv4	elasticachesubnetgroup	

Step 2: Advanced settings

Search [Alt+S]
United States (N. Virginia) ▾
Account ID: 3723-0853-3130
voclabs/user4263464=ARYAN_LANGHANOJA
Edit
Edit
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Task 4: Loading records into the cache from the database

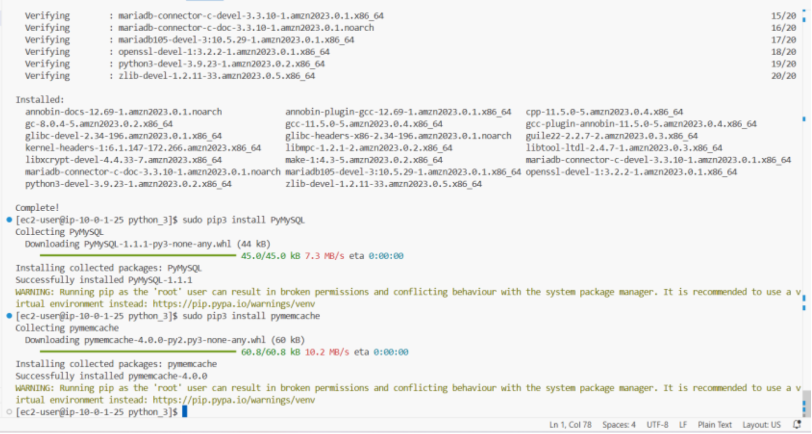
13. To change to the directory that holds the proof-of-concept scripts, return to the VS Code IDE Bash terminal, and run the following command:

```
cd ~/environment/python_3
```

14. To install the libraries that Python needs to interact with the database and the cache, run the following commands:

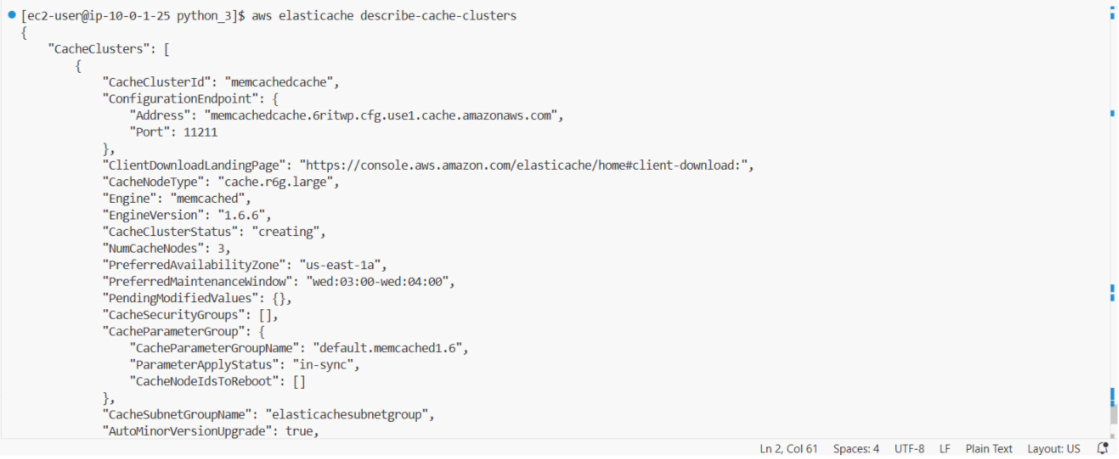
```
sudo dnf install -y mariadb105-devel gcc python3-devel
sudo pip3 install PyMySQL
sudo pip3 install pymemcache
```

In the scripts, you will need to define the connection to both the Aurora database and the Memcached cache. In the following steps, you find the endpoints for these resources.



```
  Verifying        : mariadb-connector-c-devel-3.3.10-1.amzn2023.0.1.x86_64                                       15/20
  Verifying        : mariadb-connector-c-doc-3.3.10-1.amzn2023.0.1.noarch                                          16/20
  Verifying        : mariadb105-devel-3:10.5.29-1.amzn2023.0.1.x86_64                                              17/20
  Verifying        : openssl-devel-1:3.2.2-1.amzn2023.0.1.x86_64                                                   18/20
  Verifying        : python3-devel-3.9.23-1.amzn2023.0.2.x86_64                                                    19/20
  Verifying        : zlib-devel-1.2.11-33.amzn2023.0.5.x86_64                                                      20/20

Installed:
  annobin-docs-12.69-1.amzn2023.0.1.noarch          annobin-plugin-gcc-12.69-1.amzn2023.0.1.x86_64   cpp-11.5.0-5.amzn2023.0.4.x86_64
  gc-8.0.4-5.amzn2023.0.2.x86_64                    gcc-11.5.0-5.amzn2023.0.4.x86_64                 gcc-plugin-annobin-11.5.0-5.amzn2023.0.4.x86_64
  glibc-devel-2.34-196.amzn2023.0.1.x86_64          glibc-headers-x86-2.34-196.amzn2023.0.1.noarch   guile22-2.2.7-2.amzn2023.0.3.x86_64
  kernel-headers-1:6.1.147-172.266.amzn2023.x86_64  libmpc-1.2.1-2.amzn2023.0.2.x86_64               libtool-ltdl-2.4.7-1.amzn2023.0.3.x86_64
  libxcrypt-devel-4.4.33-7.amzn2023.x86_64          make-1:4.3-5.amzn2023.0.2.x86_64                 mariadb-connector-c-devel-3.3.10-1.amzn2023.0.1.x86_64
  mariadb-connector-c-doc-3.3.10-1.amzn2023.0.1.noarch mariadb105-devel-3:10.5.29-1.amzn2023.0.1.x86_64 openssl-devel-1:3.2.2-1.amzn2023.0.1.x86_64
  python3-devel-3.9.23-1.amzn2023.0.2.x86_64        zlib-devel-1.2.11-33.amzn2023.0.5.x86_64

Complete!
[ec2-user@ip-10-0-1-25 python_3]$ sudo pip3 install PyMySQL
Collecting PyMySQL
  Downloading PyMySQL-1.1.1-py3-none-any.whl (44 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 45.0/45.0 kB 7.3 MB/s eta 0:00:00
Installing collected packages: PyMySQL
Successfully installed PyMySQL-1.1.1
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a v
irtual environment instead: https://pip.pypa.io/warnings/venv
[ec2-user@ip-10-0-1-25 python_3]$ sudo pip3 install pymemcache
Collecting pymemcache
  Downloading pymemcache-4.0.0-py2.py3-none-any.whl (60 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 60.8/60.8 kB 10.2 MB/s eta 0:00:00
Installing collected packages: pymemcache
Successfully installed pymemcache-4.0.0
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a v
irtual environment instead: https://pip.pypa.io/warnings/venv
[ec2-user@ip-10-0-1-25 python_3]$
```

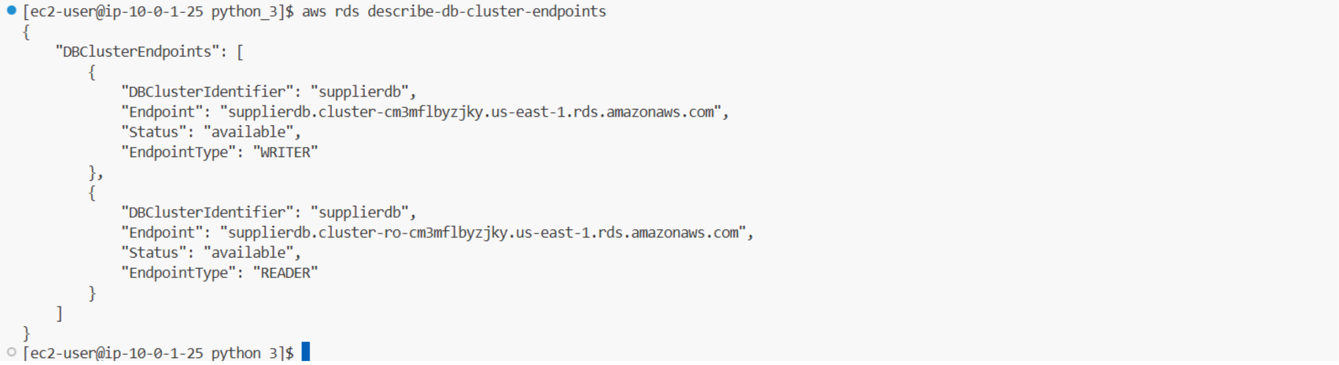
15. Find the ElastiCache for Memcached endpoint.
    - In the VS Code IDE Bash terminal, run the following command:
      `aws elasticache describe-cache-clusters`



```
[ec2-user@ip-10-0-1-25 python_3]$ aws elasticache describe-cache-clusters
{
    "CacheClusters": [
        {
            "CacheClusterId": "memcachedcache",
            "ConfigurationEndpoint": {
                "Address": "memcachedcache.6ritwp.cfg.use1.cache.amazonaws.com",
                "Port": 11211
            },
            "ClientDownloadLandingPage": "https://console.aws.amazon.com/elasticache/home#client-download:",
            "CacheNodeType": "cache.r6g.large",
            "Engine": "memcached",
            "EngineVersion": "1.6.6",
            "CacheClusterStatus": "creating",
            "NumCacheNodes": 3,
            "PreferredAvailabilityZone": "us-east-1a",
            "PreferredMaintenanceWindow": "wed:03:00-wed:04:00",
            "PendingModifiedValues": {},
            "CacheSecurityGroups": [],
            "CacheParameterGroup": {
                "CacheParameterGroupName": "default.memcached1.6",
                "ParameterApplyStatus": "in-sync",
                "CacheNodeIdsToReboot": []
            },
            "CacheSubnetGroupName": "elasticachesubnetgroup",
            "AutoMinorVersionUpgrade": true,
```

16. Find the Aurora Serverless cluster endpoint.

- In the VS Code IDE Bash terminal, run the following command:
  `aws rds describe-db-cluster-endpoints`



```
[ec2-user@ip-10-0-1-25 python_3]$ aws rds describe-db-cluster-endpoints
{
    "DBClusterEndpoints": [
        {
            "DBClusterIdentifier": "supplierdb",
            "Endpoint": "supplierdb.cluster-cm3mflbyzjky.us-east-1.rds.amazonaws.com",
            "Status": "available",
            "EndpointType": "WRITER"
        },
        {
            "DBClusterIdentifier": "supplierdb",
            "Endpoint": "supplierdb.cluster-ro-cm3mflbyzjky.us-east-1.rds.amazonaws.com",
            "Status": "available",
            "EndpointType": "READER"
        }
    ]
}
[ec2-user@ip-10-0-1-25 python 3]$
```

17. Update the script to define the database and cache connections.
    - In the *find_all.py* file, replace <*FMI_1*> with the ElastiCache endpoint address.

The updated code should be similar to the following:

```
memcached_client =
base.Client(('memcachedcache.xxxxxxx.cfg.use1.cache.amazonaws.com', 11211))
```

- Replace <*FMI_2*> with the Aurora endpoint.

The updated code should be similar to the following:

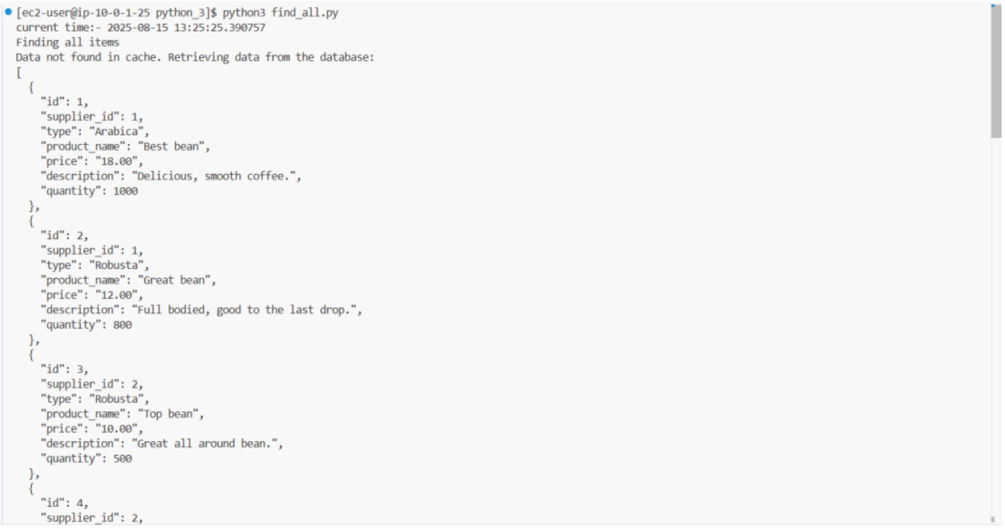
```
mydb = pymysql.connect(
  "supplierdb.cluster-xxxxxxxxxxx.us-east-1.rds.amazonaws.com",
  "nodeapp",
  "coffee",
  "COFFEE"
```

)

- From the navigation pane, choose menu, then choose **File > Save**.

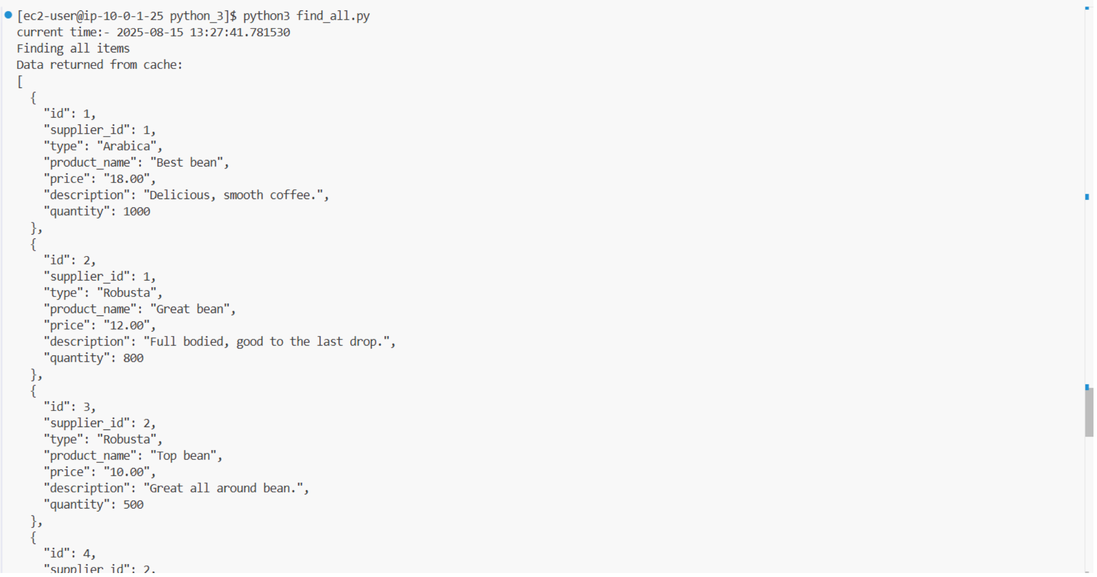
18. Test the *find_all.py* script.

- First, ensure that the status of the Memcached cluster is *available* before you continue.
- In the VS Code IDE Bash terminal, run the following command:
  `python3 find_all.py`



```
● [ec2-user@ip-10-0-1-25 python_3]$ python3 find_all.py
  current time:- 2025-08-15 13:25:25.390757
  Finding all items
  Data not found in cache. Retrieving data from the database:
  [
    {
      "id": 1,
      "supplier_id": 1,
      "type": "Arabica",
      "product_name": "Best bean",
      "price": "18.00",
      "description": "Delicious, smooth coffee.",
      "quantity": 1000
    },
    {
      "id": 2,
      "supplier_id": 1,
      "type": "Robusta",
      "product_name": "Great bean",
      "price": "12.00",
      "description": "Full bodied, good to the last drop.",
      "quantity": 800
    },
    {
      "id": 3,
      "supplier_id": 2,
      "type": "Robusta",
      "product_name": "Top bean",
      "price": "10.00",
      "description": "Great all around bean.",
      "quantity": 500
    },
    {
      "id": 4,
      "supplier_id": 2,
```

- In the VS Code IDE Bash terminal, run the command again:
  `python3 find_all.py`



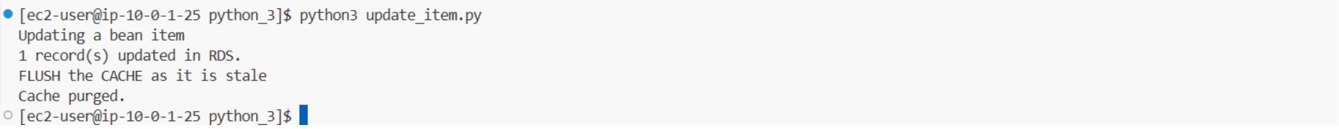
```
● [ec2-user@ip-10-0-1-25 python_3]$ python3 find_all.py
  current time:- 2025-08-15 13:27:41.781530
  Finding all items
  Data returned from cache:
  [
    {
      "id": 1,
      "supplier_id": 1,
      "type": "Arabica",
      "product_name": "Best bean",
      "price": "18.00",
      "description": "Delicious, smooth coffee.",
      "quantity": 1000
    },
    {
      "id": 2,
      "supplier_id": 1,
      "type": "Robusta",
      "product_name": "Great bean",
      "price": "12.00",
      "description": "Full bodied, good to the last drop.",
      "quantity": 800
    },
    {
      "id": 3,
      "supplier_id": 2,
      "type": "Robusta",
      "product_name": "Top bean",
      "price": "10.00",
      "description": "Great all around bean.",
      "quantity": 500
    },
    {
      "id": 4,
      "supplier_id": 2,
```

Task 5: Updating data

19. Test the *update_item.py* script.

- Update the *<FMI_1>* and *<FMI_2>* placeholders as you did previously.
- Run the *update.py* script:

```
python3 update_item.py
```

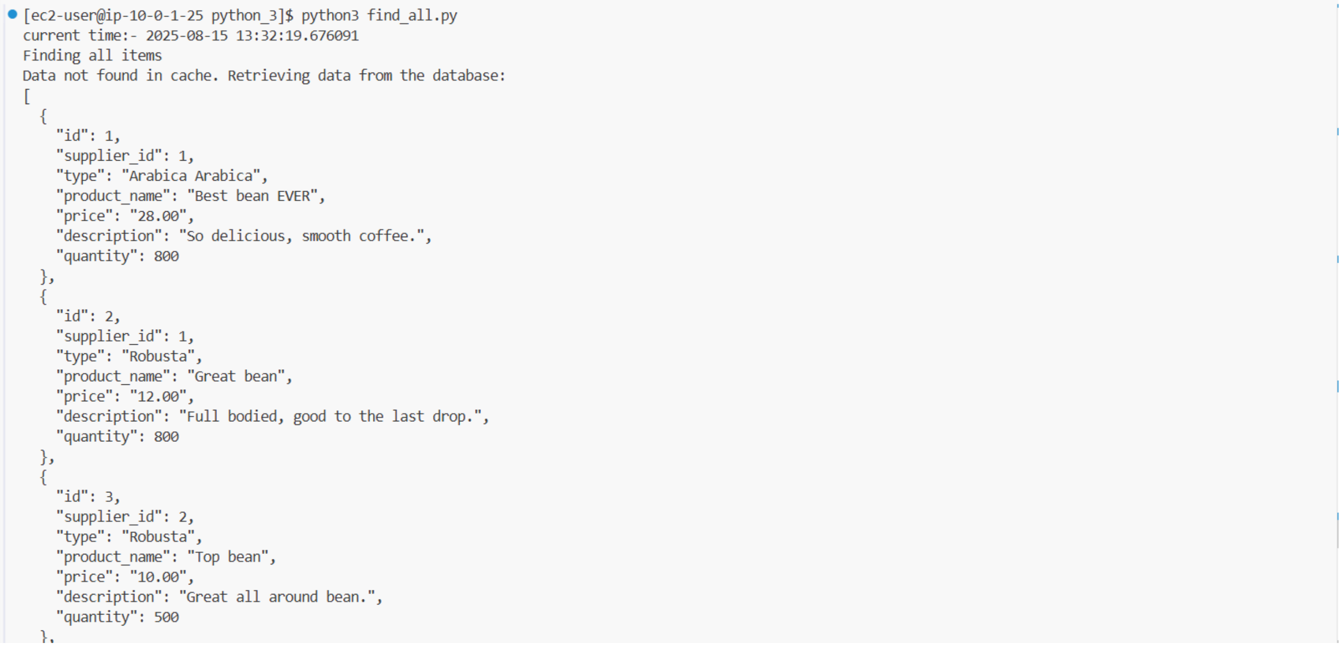


```
[ec2-user@ip-10-0-1-25 python_3]$ python3 update_item.py
Updating a bean item
1 record(s) updated in RDS.
FLUSH the CACHE as it is stale
Cache purged.
[ec2-user@ip-10-0-1-25 python_3]$
```

20. Now, to test the cache refresh.

- Run the *find_all.py* script:

```
python3 find_all.py
```



```
[ec2-user@ip-10-0-1-25 python_3]$ python3 find_all.py
current time:- 2025-08-15 13:32:19.676091
Finding all items
Data not found in cache. Retrieving data from the database:
[
  {
    "id": 1,
    "supplier_id": 1,
    "type": "Arabica Arabica",
    "product_name": "Best bean EVER",
    "price": "28.00",
    "description": "So delicious, smooth coffee.",
    "quantity": 800
  },
  {
    "id": 2,
    "supplier_id": 1,
    "type": "Robusta",
    "product_name": "Great bean",
    "price": "12.00",
    "description": "Full bodied, good to the last drop.",
    "quantity": 800
  },
  {
    "id": 3,
    "supplier_id": 2,
    "type": "Robusta",
    "product_name": "Top bean",
    "price": "10.00",
    "description": "Great all around bean.",
    "quantity": 500
  },
```

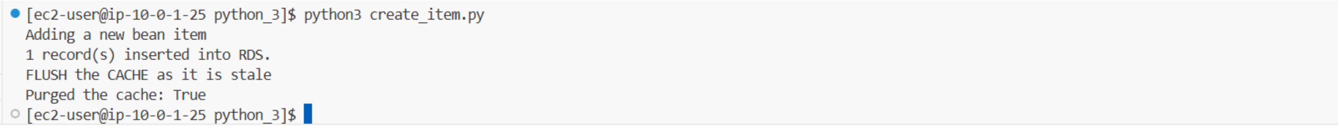
Task 6: Creating a new record

In this task, you will create a new item in the database. You will use the *write-through* caching strategy to prevent the cache from becoming stale.

21. Test the *create_item.py* script.

- Update the *<FMI_1>* and *<FMI_2>* placeholders as you did previously.
- Run the *update.py* script:

```
python3 create_item.py
```



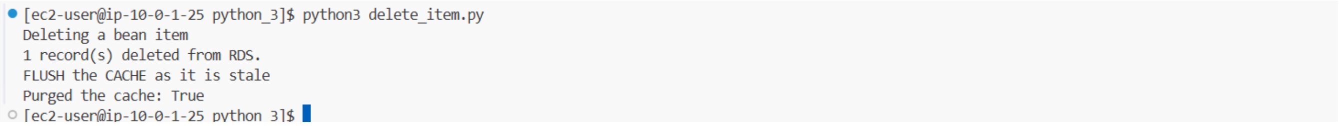
```
● [ec2-user@ip-10-0-1-25 python_3]$ python3 create_item.py
  Adding a new bean item
  1 record(s) inserted into RDS.
  FLUSH the CACHE as it is stale
  Purged the cache: True
○ [ec2-user@ip-10-0-1-25 python_3]$ 
```

Task 7: Deleting a record

The final proof-of-concept script tests deleting an item from the database. You will again use the *write-through* strategy to ensure the cache stays in sync with the database.

Test the *delete_item.py* script.

- Update the *<FMI_1>* and *<FMI_2>* placeholders as you did previously.
- Run the *delete_item.py* script:
  python3 delete_item.py



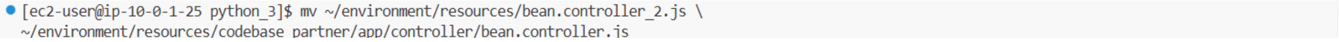
```
● [ec2-user@ip-10-0-1-25 python_3]$ python3 delete_item.py
  Deleting a bean item
  1 record(s) deleted from RDS.
  FLUSH the CACHE as it is stale
  Purged the cache: True
○ [ec2-user@ip-10-0-1-25 python_3]$ 
```

Task 8: Updating the application container

Nikhil quickly updated the coffee suppliers application. His code expanded on the examples from the proof of concept. He restructured the code, and now rather than having a script for each type of operation (SELECT, INSERT, UPDATE, DELETE), one script handles all database calls, and another interacts with the cache. He also modified the code to use an environment variable to identify the ElastiCache for Memcached endpoint.

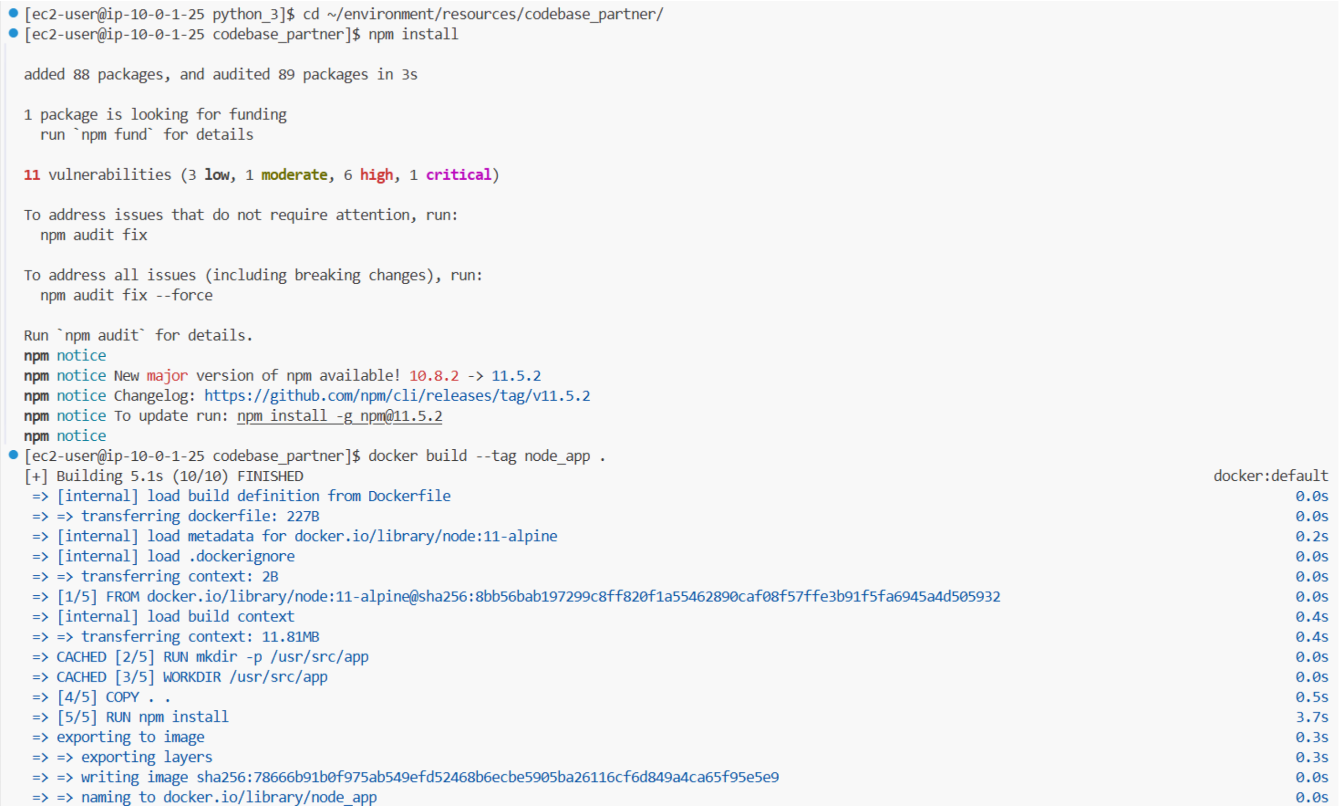
You will perform the following steps as Nikhil to review the application code that interacts with the cache. Then, you will update the application.

29. To update the application code, run the following command:
```
mv ~/environment/resources/bean.controller_2.js \
~/environment/resources/codebase_partner/app/controller/bean.controller
.js
```



```
● [ec2-user@ip-10-0-1-25 python_3]$ mv ~/environment/resources/bean.controller_2.js \
  ~/environment/resources/codebase_partner/app/controller/bean.controller.js
```

30. To rebuild and update the Docker image, run the following commands:
```
cd ~/environment/resources/codebase_partner/
npm install
docker build --tag node_app .
```



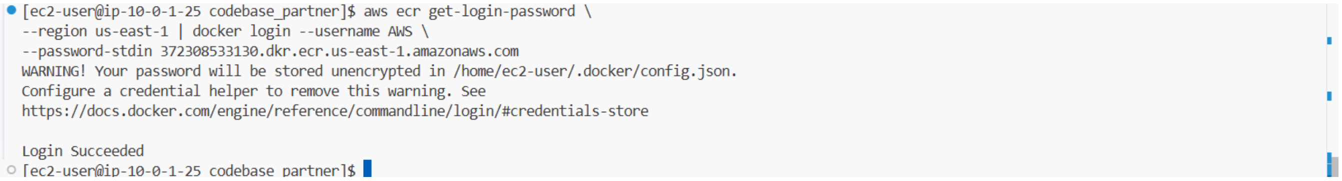
```
● [ec2-user@ip-10-0-1-25 python_3]$ cd ~/environment/resources/codebase_partner/
● [ec2-user@ip-10-0-1-25 codebase_partner]$ npm install

  added 88 packages, and audited 89 packages in 3s

  1 package is looking for funding
    run `npm fund` for details

  11 vulnerabilities (3 low, 1 moderate, 6 high, 1 critical)

  To address issues that do not require attention, run:
    npm audit fix

  To address all issues (including breaking changes), run:
    npm audit fix --force

  Run `npm audit` for details.
  npm notice
  npm notice New major version of npm available! 10.8.2 -> 11.5.2
  npm notice Changelog: https://github.com/npm/cli/releases/tag/v11.5.2
  npm notice To update run: npm install -g npm@11.5.2
  npm notice
● [ec2-user@ip-10-0-1-25 codebase_partner]$ docker build --tag node_app .
  [+] Building 5.1s (10/10) FINISHED                                                                        docker:default
   => [internal] load build definition from Dockerfile                                                                0.0s
   => => transferring dockerfile: 227B                                                                                0.0s
   => [internal] load metadata for docker.io/library/node:11-alpine                                                   0.2s
   => [internal] load .dockerignore                                                                                   0.0s
   => => transferring context: 2B                                                                                     0.0s
   => [1/5] FROM docker.io/library/node:11-alpine@sha256:8bb56bab197299c8ff820f1a55462890caf08f57ffe3b91f5fa6945a4d505932  0.0s
   => [internal] load build context                                                                                   0.4s
   => => transferring context: 11.81MB                                                                                0.4s
   => CACHED [2/5] RUN mkdir -p /usr/src/app                                                                          0.0s
   => CACHED [3/5] WORKDIR /usr/src/app                                                                               0.0s
   => [4/5] COPY . .                                                                                                  0.5s
   => [5/5] RUN npm install                                                                                           3.7s
   => exporting to image                                                                                              0.3s
   => => exporting layers                                                                                             0.3s
   => => writing image sha256:78666b91b0f975ab549efd52468b6ecbe5905ba26116cf6d849a4ca65f95e5e9                        0.0s
   => => naming to docker.io/library/node_app                                                                         0.0s
```

31. Authorize your Docker client to connect to the Amazon Elastic Container Registry (Amazon ECR) service.

- Discover your AWS account ID:
- In the AWS Management Console, in the upper-right corner, choose your user name, which begins with **voclabs/user**.
- Copy the **My Account** value from the menu. This is your AWS account ID.
- Return to the VS Code IDE Bash terminal.
- To authorize your VS Code IDE Docker client, run the following command. Replace with the actual account ID that you just found in the console:

```
aws ecr get-login-password \
--region us-east-1 | docker login --username AWS \
--password-stdin <account-id>.dkr.ecr.us-east-1.amazonaws.com
```

- A message indicates that the login succeeded.

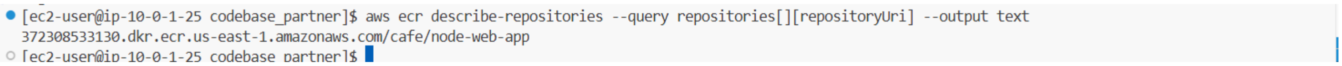


```
● [ec2-user@ip-10-0-1-25 codebase_partner]$ aws ecr get-login-password \
  --region us-east-1 | docker login --username AWS \
  --password-stdin 372308533130.dkr.ecr.us-east-1.amazonaws.com
  WARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
  Configure a credential helper to remove this warning. See
  https://docs.docker.com/engine/reference/commandline/login/#credentials-store

  Login Succeeded
○ [ec2-user@ip-10-0-1-25 codebase_partner]$ 
```

32. Push the updated container image to Amazon ECR.

- To find the URI for the repository, run the following command:

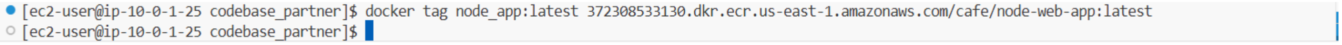
```
aws ecr describe-repositories --query
repositories[][repositoryUri] --output text
```



```
● [ec2-user@ip-10-0-1-25 codebase_partner]$ aws ecr describe-repositories --query repositories[][repositoryUri] --output text
  372308533130.dkr.ecr.us-east-1.amazonaws.com/cafe/node-web-app
○ [ec2-user@ip-10-0-1-25 codebase_partner]$
```

- Next, to tag the Amazon ECR image, run the following command. Replace <*FMI_1*> with the repository URI:

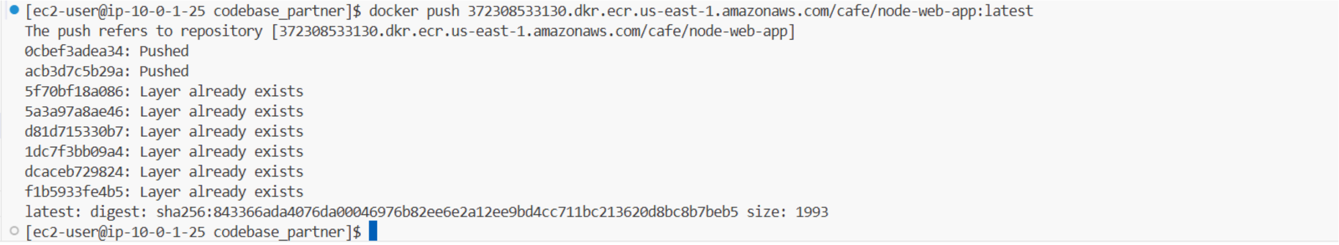
```
docker tag node_app:latest xxxxxxxxxxxx.dkr.ecr.us-east-
1.amazonaws.com/cafe/node-web-app:latest
```



```
● [ec2-user@ip-10-0-1-25 codebase_partner]$ docker tag node_app:latest 372308533130.dkr.ecr.us-east-1.amazonaws.com/cafe/node-web-app:latest
○ [ec2-user@ip-10-0-1-25 codebase_partner]$
```

- Now, to push the image to Amazon ECR, run the following command. Again, replace <*FMI_1*> with the repository URI:

```
docker push 372308533130.dkr.ecr.us-east-1.amazonaws.com/cafe/node-web-
app:latest
```



```
● [ec2-user@ip-10-0-1-25 codebase_partner]$ docker push 372308533130.dkr.ecr.us-east-1.amazonaws.com/cafe/node-web-app:latest
  The push refers to repository [372308533130.dkr.ecr.us-east-1.amazonaws.com/cafe/node-web-app]
  0cbef3adea34: Pushed
  acb3d7c5b29a: Pushed
  5f70bf18a086: Layer already exists
  5a3a97a8ae46: Layer already exists
  d81d715330b7: Layer already exists
  1dc7f3bb09a4: Layer already exists
  dcaceb729824: Layer already exists
  f1b5933fe4b5: Layer already exists
  latest: digest: sha256:843366ada4076da00046976b82ee6e2a12ee9bd4cc711bc213620d8bc8b7beb5 size: 1993
○ [ec2-user@ip-10-0-1-25 codebase_partner]$
```

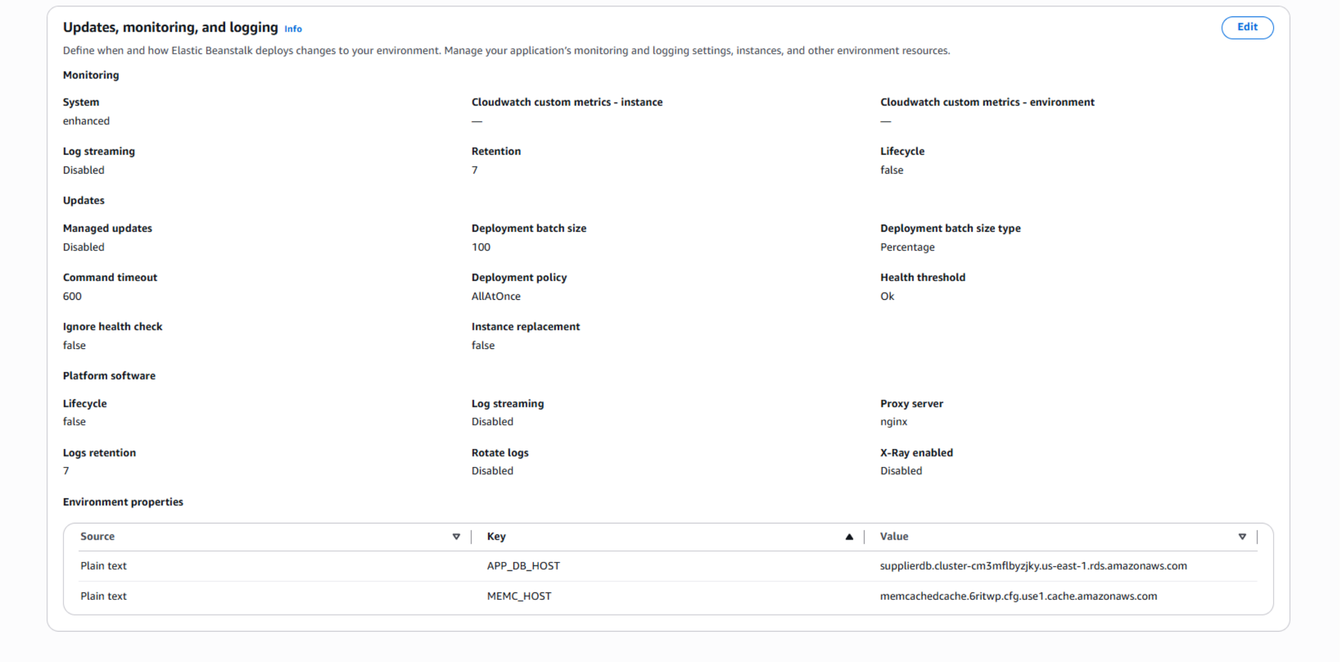
34. Configure the new Memcached environment variable, *MEMC_HOST*.
    - Return to the browser tab with the **MyEnv** page.
    - In the left navigation pane, choose **Configuration**.
    - In the **Updates, monitoring, and logging** category, choose **Edit**.
    - Scroll down to the **Environment properties** section, and add **Add environment property** as follows:
        - **Name:** Enter MEMC_HOST
        - **Value:** Enter the same Memcached endpoint that you used when testing the Python scripts
    - Choose **Apply**.

A message indicates that *Elastic Beanstalk is updating your environment.*

It takes a few minutes for the environment to be updated. Once the build has finished, you can continue.

- Revisit the browser tab with the coffee suppliers application, and refresh the page.

A note now appears under **All beans** to indicate when the results were returned from the database and when they were returned from the cache.



Updates, monitoring, and logging Info

Define when and how Elastic Beanstalk deploys changes to your environment. Manage your application's monitoring and logging settings, instances, and other environment resources.

Monitoring

System enhanced	Cloudwatch custom metrics - instance –	Cloudwatch custom metrics - environment –
Log streaming Disabled	Retention 7	Lifecycle false

Updates

Managed updates Disabled	Deployment batch size 100	Deployment batch size type Percentage
Command timeout 600	Deployment policy AllAtOnce	Health threshold Ok
Ignore health check false	Instance replacement false	

Platform software

Lifecycle false	Log streaming Disabled	Proxy server nginx
Logs retention 7	Rotate logs Disabled	X-Ray enabled Disabled

Environment properties

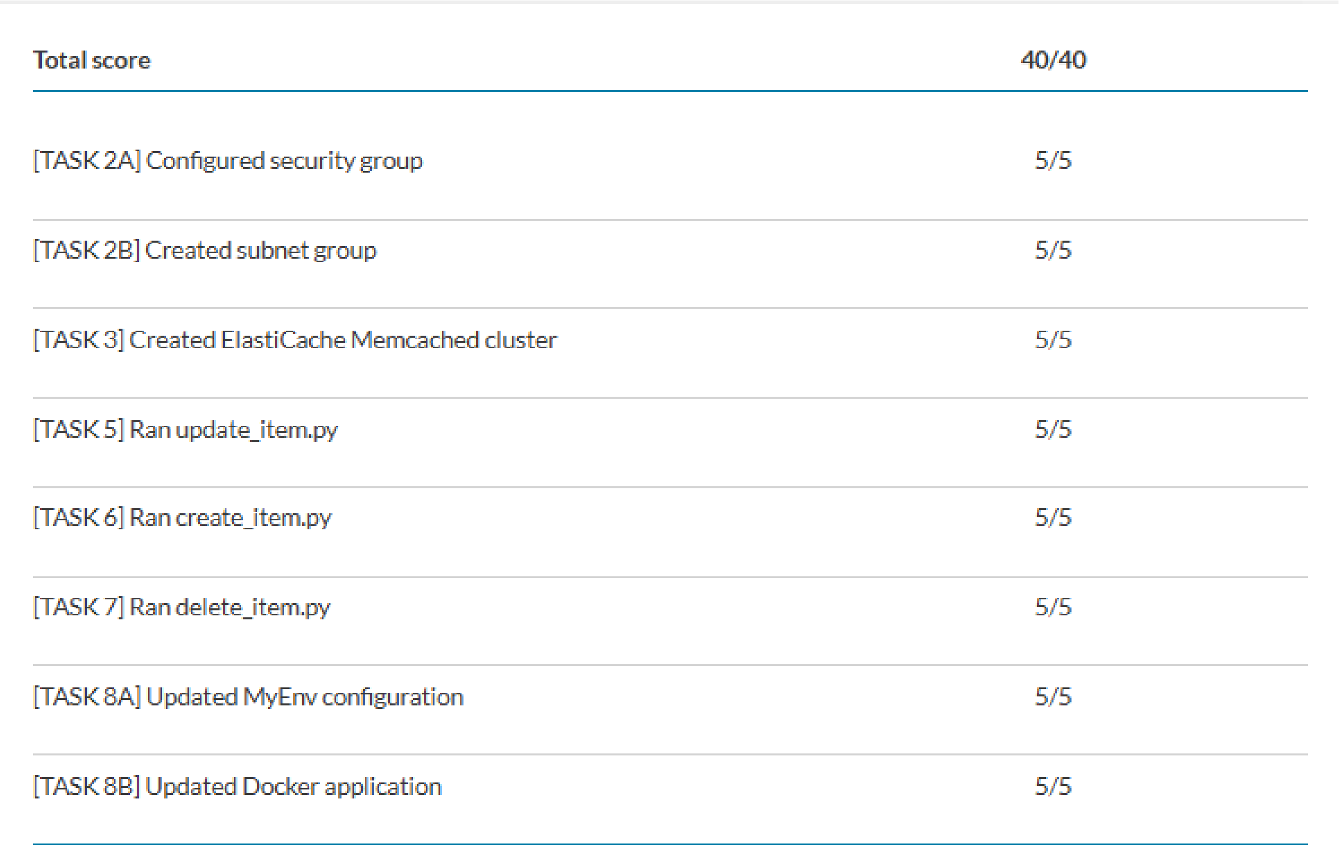
Source	Key	Value
Plain text	APP_DB_HOST	supplierdb.cluster-cm3mflbyzjky.us-east-1.rds.amazonaws.com
Plain text	MEMC_HOST	memcachedcache.6ritwp.cfg.use1.cache.amazonaws.com

Conclusion:-

- **Set up the lab environment –** connected to VS Code IDE, downloaded and extracted lab files.
  - wget <url> → download code
  - unzip code.zip → extract files
- **Configured AWS CLI & Python SDK –** upgraded AWS CLI and verified installation.
  - chmod +x ./resources/setup.sh && ./resources/setup.sh → run setup script
  - aws --version, pip3 show boto3 → verify installation
  - **Configured networking –** updated **Security Groups** and created **Subnet Group** for ElastiCache.
  - **Created ElastiCache Cluster –** deployed Memcached with required nodes and parameters via AWS Console.
- **Installed dependencies & connected Python scripts**
  - sudo dnf install -y mariadb105-devel gcc python3-devel
  - sudo pip3 install PyMySQL pymemcache
  - aws elasticache describe-cache-clusters → get cache endpoint
  - aws rds describe-db-cluster-endpoints → get DB endpoint
- **Tested Python scripts for CRUD operations**
  - python3 find_all.py → read from DB & cache

  - python3 update_item.py → update record & refresh cache
  - python3 create_item.py → insert record with write-through caching
  - python3 delete_item.py → delete record & sync cache
- **Updated Node.js application & Docker container**
  - mv bean.controller_2.js ... → update controller code
  - npm install → install dependencies
  - docker build --tag node_app . → build image
  - aws ecr get-login-password | docker login ... → authenticate
  - docker tag & docker push → push image to ECR
  - **Configured environment variable (MEMC_HOST)** in Elastic Beanstalk and tested the updated application.

Result :-



Total score	40/40
[TASK 2A] Configured security group	5/5
[TASK 2B] Created subnet group	5/5
[TASK 3] Created ElastiCache Memcached cluster	5/5
[TASK 5] Ran update_item.py	5/5
[TASK 6] Ran create_item.py	5/5
[TASK 7] Ran delete_item.py	5/5
[TASK 8A] Updated MyEnv configuration	5/5
[TASK 8B] Updated Docker application	5/5