**Phase 2: Advanced AWS Service Implementation**

Vince Capistrano (2307321)

Madesh Manoharan (2323307)

Gaurav Dutt Meda (2313830)

Kohilagedara Yodinge Madini Pushpika Senevirathne (2312370)

University Canada West   
BUSI 653: Cloud Computing Technologies

Prof. Gholibeigian, Sarah

December 08, 2024.

Table of Contents

[Advanced AWS Service Implementation 3](#_Toc184565372)

[AWS Elastic Load Balancing (ELB) 3](#_Toc184565373)

[AWS Auto Scaling 5](#_Toc184565374)

[Amazon CloudWatch 9](#_Toc184565375)

[AWS Lambda 12](#_Toc184565376)

[Performance Analysis 13](#_Toc184565377)

[Impact on Platform Performance 13](#_Toc184565378)

[Scalability 14](#_Toc184565379)

[User Experience 15](#_Toc184565380)

[Challenge Review 15](#_Toc184565381)

[PHP and Apache Integration 16](#_Toc184565382)

[Database Connectivity via Local PC 16](#_Toc184565383)

[IAM Role Setup for Lambda 16](#_Toc184565384)

[Load Balancer Configuration Issues 16](#_Toc184565385)

[Network Latency During Testing 17](#_Toc184565386)

[Scaling Configuration Challenges 17](#_Toc184565387)

[Ensuring Security Compliance 17](#_Toc184565388)

[Insights and Lessons Learned 17](#_Toc184565389)

# Advanced AWS Service Implementation

## AWS Elastic Load Balancing (ELB)

**Figure 1**

*Load Balancers*

A screenshot of a computer

Description automatically generated

**Figure 2**

*Target groups*

A screenshot of a computer

Description automatically generated

**Figure 3**

*Configuration summery*

A screenshot of a computer

Description automatically generated

Go to EC2 dashboard, under Load Balancing, and select Load Balancer. Create a Load Balancer by choosing Load Balancer Type: Application Load Balancer. To configure the Load Balancer, Name: CoffeeApp-ALB , Scheme: Choose internet facing, IP address type: Choose IPv4, Listeners: Select HTTP (port 80).

Then configure Security Groups: Select existing security group and configure the routing by creating a new target group, Name the target group name: CoffeeAppTargets, Target type: Choose Instance, Protocol: Select HTTP, Port: Choose 80 (HTTP) and set the Health checks.

**Figure 4**

*Registered targets*

A screenshot of a computer

Description automatically generated

To Register target, select the EC2 instance to associate with the target group.

Review and Create to set up the load balancer.

## AWS Auto Scaling

**Figure 5**

*Auto scaling group details*

A screenshot of a software update

Description automatically generated

**Figure 6**

*Auto scaling group details*

A screenshot of a computer

Description automatically generated

**Figure 7**

*Auto scaling group details*

A screenshot of a white sheet

Description automatically generated

**Figure 8**

*Auto scaling group*

A screenshot of a computer

Description automatically generated

Go to the EC2 dashboard, under Auto Scaling, and Select Auto Scaling. Click Create Auto Scaling group, Name: CoffeeApp-ASG. Under Launch Template, create Launch Template, Choose AMI: Amazon Linux 2 (select the AMI used for EC2 Instance), Instance Type: t2.micro (free tier), Key Pair: CoffeePlatform.pem, Security group: Select the existing one CoffeeDBSG.

To configure Auto Scaling Group, Name: CoffeeApp-ASG, Network: Select the VPC and subnet created, Load Balancer: Select the Application Load Balancer. Set the Scaling Policies by the Desired Capacity: 2 instances, Minimum Capacity: 1, Maximum Capacity: 5. After reviewing Create an Auto Scaling Group.

## Amazon CloudWatch

**Figure 9**

*Target Tracking*

A screenshot of a computer

Description automatically generated

**Figure 10**

Create Alarms

A screenshot of a computer

Description automatically generated

**Figure 11**

Create alarms

A white rectangular object with a black border

Description automatically generated

**Figure 12**

Create alarms

A screenshot of a computer

Description automatically generated

**Figure 13**

Alarms

A screenshot of a computer

Description automatically generated

Go to CloudWatch and Go to Dashboards and Click Create Dashboard, Name: CoffeeApp-Dashboard. Add widgets for metrics.

To set Alarms, go to Alarms and Create Alarm, and select the metric as I chose EC2 instance CPU utilization. Set the threshold for CPU utilization > 80% for 5 minutes and send a notification via SNS (Simple Notification Service).

## AWS Lambda

* Create an IAM Role for Lambda by going to the IAM console and creating a new role, choosing the AWS service entity and selecting Lambda and Attach the AmazonRDSFullAccess.
* Create the Lambda Function by clicking Create function, Choose Author from scratch, Name the function and Choose the IAM role.
* Write the Lambda code

import boto3

from datetime import datetime

def lambda\_handler (event, context ) :

rds\_client = boto3.client ( ‘rds’ )

snapshot\_id = 'rds-snapshot-' + datetime.now().strftime('%Y-%m-%d-%H-%M-%S') response = rds\_client.create\_db\_snapshot(

DBSnapshotIdentifier=snapshot\_id,

DBInstanceIdentifier= ‘coffeeplatformdb’

)

return response

* Go to Trigger and Choose CloudWatch Events to schedule Lambda.
* Test the Lambda Function.

## Performance Analysis

### Impact on Platform Performance

*AWS Elastic Load Balancing ELB*

ELB is used for balancing the traffic in order to ensure that the requests are split equally among the multiple targets that can include EC2 instances or containers in order to avoid overloading and increasing the throughput through leveraging the remaining capacity of the resources. This also helps in reducing latency as it sends the traffic to the nearest instance and has the feature of optimized routing which is useful for web applications that use Application Load Balancers (ALB). Also, ELB performs health checks of the instances round the clock and sends the traffic only to the healthy instances and avoids delays that may occur due to faulty or unhealthy instances.

*Amazon CloudWatch*

CloudWatch helps increase performance by providing the real time metrics and logs which can be used to track down issues, identify the bottlenecks, monitor resource utilization and ensure effective application performance monitoring. For instance, it measures various metrics including CPU utilization, memory, and latency which facilitates problem solving before they affect the platform. By using CloudWatch Logs Insights for the detailed log analysis, debugging and troubleshooting can be done in the most effective manner.

*AWS Auto Scaling*

Auto Scaling maintains the performance by ensuring that there are the right number of instances at any given time in order to meet the demands of the application. It achieves this by automatically increasing the number of instances during high traffic and decreasing the number of instances during low traffic to avoid situations where there are shortages in performance. This may lead to delays or system slowness due to overloading or over subscription whereby there are redundant resources that are being used.

### Scalability

*Dynamic Scaling via Auto Scaling*

The platform can scale up during high traffic events, such as virtual coffee-tasting sessions, and scale down during off-peak hours, always keeping cost efficiency in mind. Use of t2.micro instances, combined with the scaling configuration of 1–5 instances, helps the platform grow incrementally with increased user demand.

*Data Storage and Processing*

Amazon RDS further features read replicas to handle the very high loads of queries efficiently without loss of performance. For data storage, solutions such as AWS S3 and intelligent tiering enable cost optimization in data storage while accommodating growth in regards to uploaded images and videos and user activity logs.

*Global Reach*

Use of ELB and further integration with CDNs ensures access with low latency for users worldwide, crucial for expanding the coffee community beyond local boundaries.

### User Experience

*Reliability*

The application is available and performs well all the time, even on the highest load, because of ELB and Auto Scaling. Due to RDS and IAM policy security setup, users can trust in the platform to handle their sensitive data.

*Interactivity*

The live chat and forums will have fully optimized backend infrastructure for real-time smooth interactions likewise, responsive templates will power highly usable interface designs to better engage the learners.

*Speed and Accessibility*

Fast content delivery with ELB will ensure users are able to access their videos, tutorials, and all other resources with no latency. Improved server response times translate into faster and more engaging browsing.

Advanced AWS services significantly improve the performance, scalability, and user experience of the platform. Dynamic resource allocation, robust monitoring, and seamless traffic management ensure that the platform is well-prepared to handle growth while maintaining a high-quality user experience. With proactive measures such as CloudWatch alarms and automated backups, the platform achieves operational efficiency and resilience.

## Challenge Review

When we were developing the Digital Community Platform for our coffee shop, we experienced many problems that needed to be solved through creative collaboration and adaptive strategies. We learned a lot from these challenges, including valuable lessons on cloud based development.

### PHP and Apache Integration

The first challenge, one of the firsts, was PHP scripts were failing to run on the EC2 instance because Apache was missing the integration. In order to solve this, we used **sudo yum install php**, and restarted the Apache server with (sudo systemctl restart httpd). What we learned from this was that it’s important to test server configurations on setup so that this doesn’t happen again.

### Database Connectivity via Local PC

The other issue related to not being able to connect to the RDS database from a local PC because the security group settings were simply restrictive. This was resolved by the team by configuring the EC2 instance as a secure bridge to the database. It provided security without reducing development efficiency and demonstrated the benefit of knowing AWS-specific security configurations.

### IAM Role Setup for Lambda

We did not have the permissions to configure IAM roles so that delayed setting up AWS Lambda. We looked through AWS tutorials for this as a resolution. In the meantime, we figured out the perfect Lambda function code to avoid implementation delays after permissions were given. This has shown the need for admins to factor in administration tasks in project timelines.

### Load Balancer Configuration Issues

The issue was incorrect mappings of security groups to the Application Load Balancer (ALB). We also fixed this by re-visiting the configurations of those EC2 instances and aligning them with the EC2 instance's settings. We learned that we have to be careful about how we manage complex AWS services. Settings and have attend to detail carefully

### Network Latency During Testing

When we were application testing, we had delays in queries to the database. To that end, we optimized the database design to have indexes on fields that are queried as such, as well as EC2 instance settings. It showed us the need and proved the importance of performing performance testing at many stages.

### Scaling Configuration Challenges

Firstly, our simulation of peak traffic showed under-provisioning as a result of incorrect auto-scaling policies. Using the right set of thresholds for variable traffic loads and revising scaling policies became possible. Indeed, this shows us the necessity of real-world testing to validate the scaling configurations we mentioned above.

### Ensuring Security Compliance

It takes lots of work to properly meet AWS security best practices, such as configuring IAM policies or encrypting data. We used AWS Key Management Service (KMS) for data encryption and AWS CloudTrail for data activity monitoring. The security highlighted above is very important when developing a platform.

## Insights and Lessons Learned

Solving these problems highlighted the importance of planning forward, Iterative testing, and, of course, teamwork. For each challenge, we got to enhance and challenge our problem-solving skills and deepen our knowledge of AWS services. These issues were addressed collectively to enable a successful implementation of the platform and make us ready for future projects.

Solving these problems showed the importance of planning forward and Iterative testing and teamwork. We got to work on enhancing and challenging our problem-solving skills while understanding AWS services better for each challenge. We discussed these issues together so as to allow for a successful implementation of the platform and still have us ready for future projects.