SCHOOL OF INFOCOMM TECHNOLOGY

Diploma in IT, FI, CSF

Cloud Architecture and Technologies (CAT)

April 2020 Semester

Assignment

30% of CAT Module – (Team: 70%, Individual 30%)

6 July – 16 August 2020 (Weeks 12 & 16)

Deadline for submission:

SOFTCOPY: Submit in MeL by 16 August 2020, 23:59

Tutorial Group:	**P01**	**Team Number:**	**1**
Members	**Student No.**	**Student Name**	**Grade**
	S10185214C	**Ethan Leong**	
	S10198464	**Elvan Sim**	
	S10189579	**Azzi Izzuan**	

1. Introduction

This report talks about the solutions we have implemented for all of the mandatory and optional requirements of the second assignment. For each individual solution, we will give a detailed description of what we have done and explain the steps we took to construct the solution. This report will also go through the problems we have encountered while constructing solutions and how we managed to overcome our problems, if any. Also, we decided to do this assignment as a group on a single AWS account because Azzi had the most credit balance in his Amazon Web Service (AWS) account. We decided to use his account throughout the whole assignment. Azzi would usually screen share through Discord, a free messenger / voice /video call application, to discuss what to do. We decided to use discord as all three of us were most familiar with the application as we normally use it for voice calls during gaming sessions. If Azzi is busy, he will leave his desktop on and allow us to take control of his desktop remotely via teamviewer to get our distributed work done. This way, we can have teammates doing research while someone is doing the work.

2. Requirements

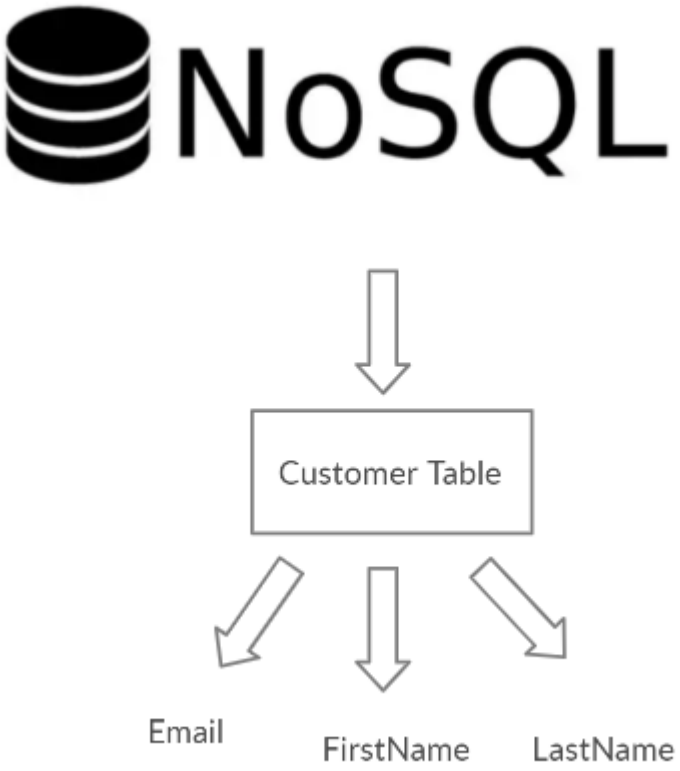
This section will talk about all the requirements we have tackled and includes detailed explanations on how we have tackled the 6 requirements handed to us in the Assignment.

2.1. First Mandatory Requirement

The first mandatory requirement requires us to deploy a scalable, resilient and load-balanced web server to host the website of the fictitious company we were assigned to. The specifications we were told to follow was to deploy General Purpose servers running an Operating System (OS) of our choice which was Windows Server 2016 Core. Another specification we had to follow is to constantly run at least 2 web servers when in low demand and it must be able to scale up to 4 servers in times of high demand. The web servers are used to host our simple web application that we have created with ASP.NET knowledge from our Web Application Development (WEB) module. The application will be used to capture customer data (E-mail, First Name, Last Name) and both of the servers are hosting the same web application.

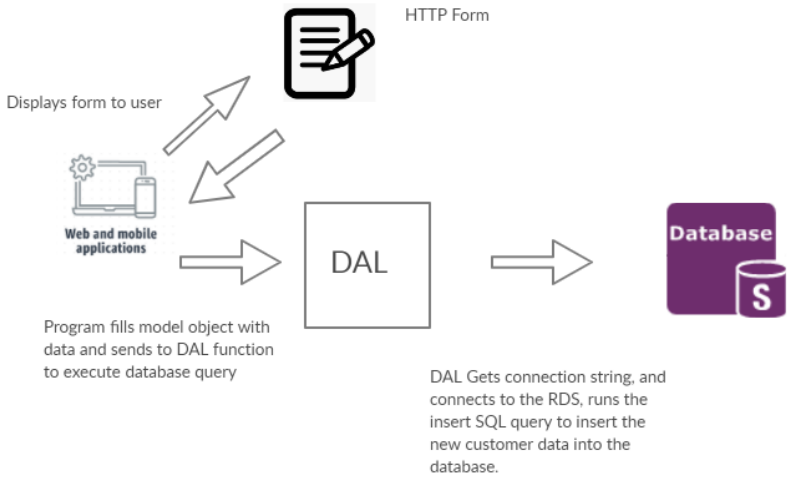
2.2. Second Mandatory Requirement

The second mandatory requirement we had to implement was to create a NoSQL database that contains fields to store customer information (E-mail, First Name and Last Name). As for the database language, we decided to use Microsoft SQL Server 2017. The fields we entered into the database customer table are: Email as a primary key, First Name and Last Name. The NoSQL Database is located in the private subnet we have created (10.0.1.0, 10.0.3.0) as we have to make it secure such that the general public will not be able to access the database. The database requires username and password authentication to connect and access the data inside. For the username and password, we have decided to follow the practical example in Week 12 with the username : sqladmin and password: password123.



2.3. Third Mandatory Requirement

The third mandatory requirement is to create a serverless function that will post the data into the created NoSQL database. For this requirement, we have decided to use ASP.NET programming language to create the program as we have learnt about it from our Web Application Development (WEB) module. The program consists of a HTML form which takes in the required fields, which are the customer's email address, their first name and their last name. The program would then store these variables into a model object, before sending the data to the Data Access Layer (DAL). The DAL is in charge of communicating to and from the database, sending and receiving data when needed. To establish a connection, a connection string is required. This has to be added to the application.json file in the ASP.NET program such that the program knows where to look for the database and to authenticate its connection. Once the user submits the form filled with the customer's data, the program will then trigger the DAL to execute the sql query along with its data, in this case inserting the data into the database fields for a new customer object.



2.4. Fourth Mandatory Requirement

The fourth mandatory requirement is to create a notification system that will notify the administrator when new customer data is added into the NoSQL database and / or new pictures have been uploaded into the S3 bucket. For the notification type, we had 2 options. We could receive either notifications via E-Mail or Phone Message. We went with the latter as the E-Mail method required extra account privileges, which we did not have. We started off by using Lambda to trigger when someone has uploaded a new picture in the s3 bucket or uploaded new customer information into DynamoDB. The base language is Python and it is written in the Lambda. Permission policies used are Cloud Watch Full Access, Amazon DynamoDb Full Access and Amazon Simple Notification Service (SNS) Full Access. Cloud Watch Full Access is basically used for monitoring and viewing logs, Amazon DynamoDB Full Access is used for letting the lambda function access the DynamoDb and finally Amazon SNS Full Access is for Lambda getting sms features.

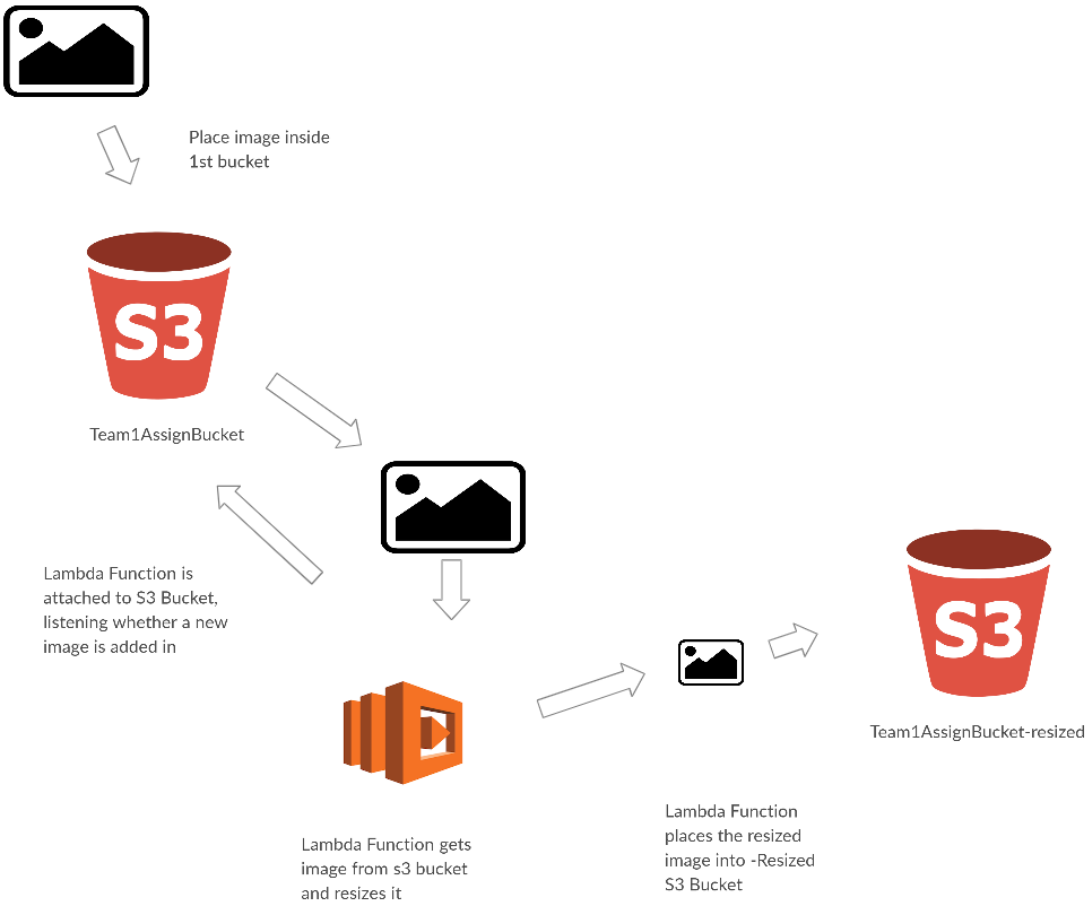
2.5. First Optional Requirement

The first optional requirement is to create a form (which we reused from the mandatory requirement 3) to capture the customer data and post it into the NoSQL database using HTTP (in this case the same

ASP.NET program). For this requirement, we had to implement an API using AWS API Gateway. This would handle the requests from ASP.net as we had to run a AWS Lambda script to insert the customer's inputted data in the HTTP form. We decided to implement it seamlessly along with the 3rd requirement using the same form, as there was already validation in place where if the customer's entered email already exists in the RDS, the data will not be inserted into the database. Therefore, when the customer fills up the HTTP form and submits it, the new entry will be inserted into both RDS and DynamoDB at the same time.

2.6. Second Optional Requirement

The Second optional requirement is to create a thumbnail for pictures uploaded to the s3 bucket, for this requirement, we had to implement a serverless function using Lambda which will trigger a python code when a picture is uploaded to the s3 bucket. The python code will resize the original image into a thumbnail and will post the resized picture to another s3 bucket which specifically contains only resized images. Upon an image upload, a notification will also be sent to the administrator's handphone number via a lambda function.



3. Problems encountered and resolutions implemented

This section talks about the problems we have encountered while implementing the assignment requirements and how we overcame the problems faced.

3.1. 1st Mandatory Requirements problems encountered

We did not face any difficulties for this requirement as most of it was taught and we could easily reference our learning materials for help.

3.2. 2nd Mandatory Requirements problems encountered

The first problem I encountered for this part of the project was how

laggy the EC2 was. This was because I followed the practical example, and used the t.2 micro free instance. This meant that the EC2 instance only had a mere 1GB of RAM. At first, it was tolerable, copying files from my computer to google drive to the EC2, however when it came to installing visual studio and SSMS, it took way too long, and was even unresponsive at most times. I would have to wait 5-10 minutes for the EC2 to become responsive again, at this point I felt that creating a EC2 with more RAM was the only solution, however I would have to reinstall and transfer all the files over again. For the new EC2 I decided to go with 8GB of ram, such that the same issue won't occur again. I made sure that whatever time I spent on the EC2 was used wisely as I was worried that the balance in the AWS account would drop significantly quicker. I also made sure to stop the EC2 and RDS Instance when not in use.

3.3. **3rd Mandatory Requirements problems encountered:**

For this part of the project, I had issues with posting the data through to AWS RDS, It kept giving me errors that the request had timed out. I had to go through my code ensuring that everything ran smoothly one by one trying to figure out why the program could not connect to the database. I checked the connection string and made sure that it matched the one in Visual Studio Server Explorer. However after checking it again, i realized that the username and password were blanked off (******** like this), I suspected that the authentication had failed when connecting to the RDS, thus i replaced the username and the password with the correct fields and it finally was able to run and push data to the database. This was new to me as i always thought that as long as the connection string is copied over correctly, that the program would automatically connect to the database with no errors.

3.4. **4th Mandatory Requirements problems encountered:**

For this part of this assignment, the problem I faced was at a later stage after the code for this part worked smoothly. There was a maximum of 50 messages that could be sent, as Rafael said in one of the lessons. I got worried that I would too hit the maximum number of notifications and that the function won't work anymore and would then have to find another solution. This was because my team and I still had to test and attempt the S3 bucket part of the assignment. The solution to this problem we faced was just to turn off the feature till we were complete with the S3 image resize part.

3.5. **1st Optional Requirements problems faced:**

For this part of the assignment, the first problem faced was figuring out how to actually submit data into the DynamoDB, for this part we had to ask our dear friend Rafael whose group had completed the assignment prior. He explained that we had to create an API gateway which would route to a AWS lambda function which would then insert the fields/data as a new item in the DynamoDB. We were confused with how the API Gateway and Lambda would link as for our WEB assignment, we were only tasked with implementing an API which was publicly accessible, however now we had to create our own API with AWS, which was foreign. However we then realized after watching/ reading through a few online tutorials, we found out that it was similar to WEB, where the API side was really simple, and was merely a redirect to a lambda function. Creating the lambda function was pretty ok, as the solution was given online, and we just had to understand it before putting it into our Lambda function. The code is just a 1. Get data from POST API, Format the data correctly using python code, lastly inserting it into the DynamoDB, there were only a few lines of code we had to change, that being the name of where the Lambda function would look for the DynamoDB to insert data in.

3.6. **2nd Optional Requirements problems faced:**

For this part of the assignment, it was very difficult to find a tutorial online that teaches us how to resize the pictures in the S3 Bucket. Also, we have not done this before during lessons. So, it was very difficult to start. The various tutorials we researched involved different methods. Most used Lambda however most used an external code editor, followed by installing multiple dependencies before actually starting to explain the code for the resizing of images. We did not understand why an external code editor was required and the worst part was we had no knowledge or any experience whatsoever with node.js. Most of the tutorials we searched for, we searched using the keyword “S3 Bucket Thumbnail resize” and we went through the search results tutorials one by one. Most of them required us to use an external code editor while connected to Lambda and running the console inside an external text editor. The first tutorial we found gave us easy to follow steps on how to start :

(https://docs.aws.amazon.com/lambda/latest/dg/with-s3-example.html). But, it uses Node.JS as a base programming language for Lambda. The Node.JS method requires packages to be installed and we did not know how to install packages or even open the Command Line Interface (CLI). We were extremely confused on how to actually attempt this part of the assignment. We decided to try again the next day, and this time we just searched “S3 Lambda Image resize”, we scrolled through multiple tutorials before finally finding a Lambda piece of code which uses python language, which did not require any dependencies to be installed. It was a big difference as python allowed us to not install any dependencies, which was the limiting factor of the first few tutorials we found. The python tutorial was so simple and easy to do and turns out it worked perfect for us and that is how we overcame our hurdle.

Lastly a last minute problem arised, we could not combine the 2 lambda functions (Simple notification service & Image resizer) this is because they both have different events (one being “put”, and another being “all objects create events”) this is because both of these events have the same permissions to the same s3 bucket and that is what caused the error/ mishap. So, to solve this problem we had to create another s3 bucket to demonstrate the notification system works.

4. Conclusion

In conclusion, we believe as a team that this final assignment was a test of our learnt knowledge and understanding of AWS as a whole. This assignment gave us a taste of what a real world configuration would be like, although we think it would be much harder if it was an actual real world project. It also tested our knowledge of how to launch a working VPC for a “real life” example where a HTTP form / data collection program would collect the data and be pushed to the back end side of the whole system. In a “real life” example, the back end portion of the system must not be accessible to the public, and the front end has to be able to support sudden influx of visitors, and be able to scale down during off peak hours. By using 2 availability zones, it makes sure that our system is less prone to failure if a natural disaster occurs. If one instance fails, there is still another instance as a backup so that our system will never crack under pressure. Moving back to the assignment, we felt that this assignment was a little off topic as quite a few things we had to implement were not really taught before such as the S3 bucket thumbnail generator. It was quite tedious to search up online for solutions, and I would rather have

implemented what we learnt into assignment and put to use more of what we actually learnt in the previous weeks. And also it was quite hard to get work done, as the work had to be on one computer such that when submitted, the marker can easily access all the done work and access what we have done instead of going around to each of our accounts figuring out where and who did what. This way we could only do work when all of us were free, where normally we would go on discord and instruct/help each other with the various parts of the assignment. Other times when Azzi was busy (we used his account for all of the work), we would use teamviewer to take over control of his desktop remotely so that we could continue doing our work even when he was busy. All in all we believe that we worked well as a team as usual, as we have been working together for most of our projects throughout this 1 and a half years. Another tough part was that we didn't have any chance to meet up in person to get work done, we are usually more productive when we stay back after school rushing to get work done. However due to this Covid-19 situation, nothing could be done in school. At the end of this assignment we are glad that we were able to complete all the specified required and optional extra requirements in time, cross this off our to-do list, and can now move on to studying for the Common test coming up and focusing on our other assignments.