COS20019 Cloud Computing Architecture - Assignment 02 Scalable Cloud Computing Architecture (30%)

Table of Co

Table 1. Modification History

Date (created / modified)	Purposes
2023-04-28	Created the assignment
2023-05-06	Finalize the assignment
2023-05-16	Revised as suggested by Jieying Wang
2024-02-07	Revised for 2024 Teaching
2024-04-27	Updated for new Amazon Linux 2023 AMI

Assignment Overview

This assignment has two components:

1. ACF and ACA Lab Exercises (5% total) - 5 Lab Exercises from ACF and ACA - worth 1% each
2. Cloud Project (22% total) - Show that you can integrate the skills learnt in the Lab Exercises to deploy a scalable Web application on the cloud

Class attendance worth of 3% has been incorporated in the following assignment marksheet.

Assignment Deadline: 27 May 2024, Monday (subject to change by your co-teacher)

Assignment Submission: Please discuss with your co-teacher about how and what to submit.

Assignment MarkSheet:

Component	Tasks	Total Marks	Student Marks	Date Checked
Class attendance	(A)	3%		
Lab Exercises	ACF Lab 06	1%		
	ACA Lab	1%		
	ACA Lab	1%		
	ACA Lab	1%		
	ACA Lab	1%		
	Sub-Total (1)	5%		
Cloud Project	Scalable Photo Album Project (2)	22%		
Total	= (A) + (1) + (2)	30%		

1. Lab Exercises (5% in Total, 5 Lab Exercises, 1% each)

1. Complete ACF and ACA Lab Exercises as per the instructions
2. Demo your work in the presence of your co-teacher for each Lab

Marking Criteria for Lab Exercises

Below is the marking criteria of each Lab exercise

a. Complete all work required (e.g. be able to show the relevant configuration information on the cloud environment and can explain how the relevant components work together) - 1%

b. Partially complete the work required (e.g. can show partial information but not fully understanding how the relevant components work together) - 0.5%

c. Cannot complete the work required (e.g. do not understand what to do) - 0%

2. Scalable Cloud Project - Highly Available Photo Album Web Application (22% in total)

In the cloud project of Assignment 1, you have learnt how to deploy a Photo Album Web Application on a VPC, actually in a EC2 running as a Web Server. However, this is not a highly available environment. In case there are some hardware failure related to that particular EC2 instance running the Web Server. The whole Web Applicaiton will be down and hence cannot be used by the public users.

This is what we want to overcome in this assignment. We will configure this solution to make it highly available. In other words, we try to make the Photo Album Web Application (1) highly available and (2) a bit scalable. Yes, we want a bit scalable because there are many levels of

scalability as you may have known by now. In this assignment, we can only experience one or two aspects of the scalability. Our approach in this assignment is to have multiple EC2 instances running the same Web Application so that together they share the load and hence can handle higher loading.

Furthermore, in this project, we allow users to upload photos using a web browser via the internet. When a photo is upload[illegible], the application will also create a thumb nail image for ease of reference.

Similar to the cloud pr[illegible]t 1, this project has two parts, namely infrastructure deployment and functio[illegible]s. They are designed to test whether you can integrate the skills learnt in the [illegible]xercises in this subject in deploying a highly available Web application on the cloud. You are not required to program the Web Application. You will be given the code of the Web Application and instructions to revise the code.

Since we are going to make the web application scalable by having more EC2 instances running the same web application at the same time, we need to have a working system first and then we `replicate` the system to different EC2 instances. Hence, in this assignment, we need to deal with the functional requirements first. Once we have a working system, we can then configure our cloud infrastructure to make it highly available.

Unlike the cloud project in Assignment 1, you will only be given the requirements of each part with some hints on the steps. You need to think about the steps carefully before you start to attempt this project. Probably a revision of the relevant ACF and ACA labs may help.

Prerequisite requirements

1. Completed ACF Labs 01 - 05 (Assignment 1 Part 1)
2. Completed ACF Lab 06 and ACA Modules 9 and 13 Guided Labs (included in Part 1 Lab Exercises)
3. Completed Assignment 1
4. Know how to use AWS PHP SDK

Possible AWS Accounts for cloud project

You have a choice of two accounts / environments you can use to complete the cloud project of this assignment.

1. **AWS Academy Learner Lab** (**recommended**): accessible through AWS Canvas. Note that this is NOT the sandbox in ACA/ACF courses that you use for your ACA/ACF Labs. This is a managed environment that allows your co-teacher to gain access to your AWS console so your work can be marked/troubleshooted. This class gives you **US$100 credit**. **Use it carefully**. This account is deleted at the end of the semester.
2. **Regular AWS account** (**NOT recommended**): new AWS accounts are eligible for a free tier. This gives you more freedom, but you need to be careful as you will be charged for the services if you go outside the free tier offering. Make sure to keep track of your AWS services usage (using Billing & Cost Management Dashboard) throughout the entire learning to avoid paying fees. This account is on-going, but some services are no longer free after 12 months. If you choose this

option, you will need to create a (read-only) IAM user and provide its credentials to your co-teacher so your assignment can be marked properly.

Getting the files ready

Download the following files from Cloud Campus

1. `EC2-setup-script-202` (the one in Assignment 1)
2. `Remote_Access_to_an_` ws) or `Remote_Access_to_an_EC2_from_a_Mac.pdf` (Mac OS or Linux) (same as th t 1)
3. `Install_phpMyAdmin_` (the one in Assignment 1)
4. `photoalbum.zip` (a tota ignment 2)

This zip file contains the full source code of the Photo Album Web Application. It is different from the one in Assignment 1. Hence the set up script will be different. You need to customize it yourself.

Furthermore, after you unzip this file, there is a folder named `photoalbum` by default. This may conflict with your Assignment 1's photoalbum folder. Be careful. Please be remember to put all files inside `/var/www/html/photoalbum`. This is different from `/var/www/html/cos20019/photoalbum` in Assignment 1.

Last, but not least, you need to read the file `constants.php` carefully. There are some extra steps in the file telling you to install the AWS SDK inside `/var/www/html/`.

5. `photoalbum-setup-script.txt` (same as the one in Assignment 1 but you need to modify this file for Assignment 2)

6. `lambda-deployment-package-0.1.zip` (a totally new file for Assignment 2)

This zip file contains the full source code of the CreateThumbnail Lambda function.

2.1. Part 1 Functional Requirements

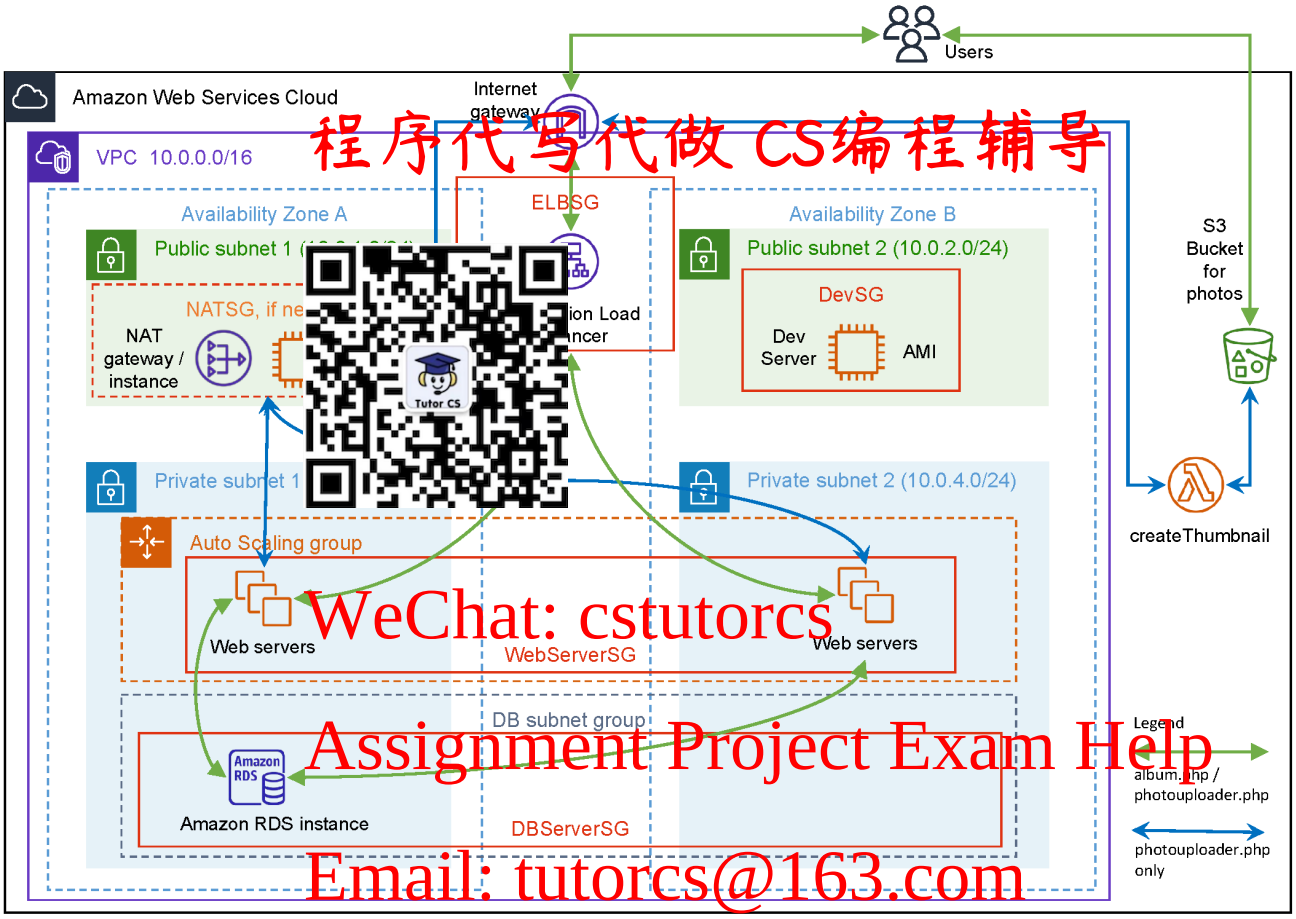

Figure 1. Cloud Architecture for Highly Available Photo Album Cloud Project

Figure 1 shows the cloud infrastructure and services of the Highly Available Photo Album Cloud Project. First, we need to make sure that we have a working system in the EC2 instance named `Dev Server`. This EC2 instance serves two purposes:

1. as a development machine - allows developers to ensure that a working system is ready (e.g. make changes to the code and test the code properly before putting it into production use)
2. as a template image - allows us to create new EC2 instances based on the image of this machine (e.g. software installed and programming code)

The Photo Album Web Application is to be hosted on your EC2 web servers. The full source code has been provided to you in *photoalbum.zip*.

You need to modify the *constants.php* file in the provided code (carefully read the comments in the file) using available information from your S3 bucket, RDS database, and Lambda function.

The web site should be accessible through `http://[your.elb.dns]/photoalbum/album.php` if the directory structure in your web server is as specified in the *constants.php* file.

Please be very careful about this. In Assignment 1, we use the public ip v4 address to access the Photo Album Web Application. However, after we have made the Auto Scaling Group available, we use the public ip v4 address of the Elastic Load Balancer to access the Photo Album Web Application. While we prepare the Web Application (the extra one is the Photo Uploading feature see Part 1 Requirement 2 below), we need to use the public ip v4 address of the `Dev Server` to test our Web

Application first. Once it is OK, we then use it as the machine image to launch the Web Servers in the Auto Scaling Group.

Furthermore, this project requires AWS SDK installation on your EC2 web server instances. Read and follow the provided instructions in the *constants.php* file carefully.

Below are the individual requirements of this Photo Album Web Application.

2.1.1. Requirement 1 Photo Album (album.php)

This page lists all the photos whose meta-data are stored in the database. Programmatically, this page performs the following actions:

- Establish a connection to the RDS instance
- Request to retrieve all the records in the database table in the RDS instance
- The RDS instance will then send back all the records to the EC2 server hosting this web page

2.1.2. Requirement 2 Photo Uploading (photouploader.php)

This page allows you to upload a photo to an S3 bucket and insert its meta-data into the RDS database. In the meantime, a Lambda function called *CreateThumbnail* will create a resized verion of the photo that was just uploaded to S3.

Programmatically, this page performs the following actions (assuming you have all information filled and photo been selected):

- When you click the [Upload] button on the page, the selected photo and its relevant meta-data will be uploaded from your local computer to an EC2 web server
- The selected photo is then uploaded from the EC2 web server to the S3 bucket
- The EC2 web server inserts the photo's meta-data (title, description, creation date, keywords and reference) into the database in the RDS instance
- The EC2 web server invokes the *CreateThumbnail* Lambda function with the bucket name and the photo name in the payload
- The Lambda function downloads the photo in the bucket specified in the payload sent above, resizes it, and uploads the resized image to the same S3 bucket. The resized image is name "*resized-<nameOfTheOriginalPhoto>.png*"

For more details, please inspect the supplied source code.

You need to set the S3 policy properly to allow write access to your S3 bucket before you can test this upload feature. Please see Part 2 Requirement 3 for some examples in setting up the S3 policy.

2.2. Part 2 Cloud Infrastructure

This part of the assignment is to set up the infrastructure, which involves the following infrastructure as shown in Figure 1:

1. A Virtual Private Cloud (VPC) with appropriate availability zones, public and private subnets
2. A Load Balancer whi
3. A Network Address T vice that routes traffic to the private subnets
4. Several Security Gro secure the access of the resources
5. An Auto Scaling Gro b Servers that run the Photo Album Web Application
6. An EC2 instance taking two roles:- (1) a Development Server and (2) a launch image for the EC2 instance in the Auto Scaling Group
7. An RDS database server instance
8. A S3 bucket that stores the photos uploaded by users

For this assignment, you have the following two options:

1. Modify your Assignment 1's VPC to suit the requirements for this assignment, or
2. Start from scratch and build your VPC accordingly

It is totally your choice. Either way, I advise you to make a plan first and enact your plan step by step.

2.2.1. Requirement 1 Your VPC

The VPC used in Highly Available Photo Album Cloud Project is similar to that in Assignment 1. The following points should be noted:

- Name: [FirstNameInitial][LastName]-vpc. For example, if your name is Man Lau, your VPC would be noted `mlau-vpc`, using all lower case is much easier for me. If you want, you can use a mix of the upper and lower case letters here, your choice.
- Region: `US East (N. Virginia) us-east-1`
- Two availability zones, each with a public and private subnet with suitable CIDR ranges (see Figure 1)
- Associate public subnets with a route table that routes to an Internet Gateway
- Associate private subnets with a private route table that routes to a NAT device (see Part 2 Requirement 2 for details)

1. Due to some incompatibility issues on AWS's management console, it is recommended that you create your VPC manually using the "**Create VPC**" button in the VPC tab. Please DO NOT use the "Start VPC Wizard" button in AWS dashboard.

2. Do not use the default VPC. All services should be in your custom VPC.

2.2.2. Requirement 2 Network Address Translation (NAT)

For private EC2 instances (i.e.\ those EC2 instances in private subnets) to be able to communicate with the public internet, their private IP addresses need to be translated by a NAT device. You need to create this NAT devic in your VPC. Please see Figure 1 for the location of this NAT device.

A NAT device is either a NAT instance (an EC2 instance taking the role of a NAT gateway).

So, you have the following two options

1. use a NAT gateway - A NAT gateway is easier to set up but very expensive. Please observe the routing from Figure 1 to configure your NAT gateway properly. Also, you need to check your costs on your Learner Labs.
2. use a NAT instance - A NAT instance is free but requires some set up. This means that you need to create an EC2 instance and configure it as a NAT server.

AWS has deprecated the relevant NAT AMI (Amazon Machine Image) for a NAT instance. Hence, you may need to create your own AMI if you choose this option.

Please see https://docs.aws.amazon.com/vpc/latest/userguide/VPC_NAT_Instance.html for the relevant instructions on setting up a NAT instance and the relevant permissions (e.g. Security Group required).

2.2.3. Requirement 3 S3 Photo Storage

Photos are to be stored in an S3 bucket, which has been created from Assignment 1, ensuring objects stored in this S3 bucket are correctly accessible by applying the appropriate permissions and policies.

In AWS Learner Lab environment, you may not be able to create your own IAM roles due to AWS Academy's restrictions. Nonetheless, an IAM role named "**LabRole**" or "**LabinstanceProfile**" with required permissions already exists in your management console that can be used for this assignment.

Since you will be uploading new photos to your S3 bucket, you need to set up appropriate S3 bucket policy to allow users to write to the S3 bucket.

You need to make sure that the uploading of new photos must be via your web application or done by you via the S3 Management Console, meaning that you cannot directly upload any photos to your S3 bucket using the URL of your S3 bucket.

Please see https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html for the appropriate examples. You may need to modify the examples to suit your needs. There are so many options in this part. Here are two options (choose one for your assignment, you do not

need to do both):

1. Allow write access via specific HTTP referrer

Please see https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html#example-bucket-policies-HTTP-HTTPS-2 for an example.

2. Allow write access (...om the IAM role "**LabRole**" or "**LabinstanceProfile**" to the S3 bucket

There ...ple in https://docs.aws.amazon.com/AmazonS3/latest/userg...t-policies.html that matches this requirement. You may need to read through some of them and modify them to suit your need.

2.2.4. Requirement 4 Load Balancing

Web request load needs to be distributed across the web servers in the auto-scaling group using an Application Load Balancer. Ensure that your Elastic Load Balancer (ELB) is running health checks on all instances.

Your ELB needs to be internet facing and routes the traffic to appropriate Web Servers, in the private subnets, running the Photo Album Web Application.

The health check path must be correctly configured (e.g. "/photoalbum/album.php", see the Functional Requirements for details). Otherwise, the health checks would fail.

2.2.5. Requirement 5 Auto Scaling

You need to create an Auto Scaling Group (ASG) that scales your Web Server's EC2 instances automatically.

You need to define a scaling policy for your auto scaling group with at least the following rules:

- The minimum number of servers is 2. The maximum number of servers is 4.
- Configure a target tracking scaling policy to keep the request count per target of your ELB target group at 30 for your Auto Scaling group.

The ASG should launch instances into the private subnets.

2.2.6. Requirement 6 EC2 Web Server instance

Your EC2 Web Server instances should be based on *Amazon Linux 2023 AMI*, which is similar to the one used in Assignment 1. They need to be given proper permissions through an IAM role to be able to put objects into the S3 bucket and invoke the *CreateThumbnail* Lambda function (see Part 2 Requirement 7). The role must follow the *least-privilege principle*.

These instances should be **automatically** launched by the auto scaling group, and only accept

incoming traffic from the load balancer. Once launched, they should be ready to serve Photo Album users without any further human intervention. In other words, you should not have to do any configurations once the instances have been launched.

An ASG can launch instances based on an AMI that has been customized by you.

The Development (Dev) [illegible]eive traffic from the ELB. The Dev server can be used to develop the custom AMI [illegible]ain everything needed to run the Photo Album Web Site (AWS PHP SDK, Apache [illegible]code of the Web Application, etc.). It can also be used to manage your database ([illegible]in - similar to Assignment 1).

2.2.7. Requirement[illegible]umbnail Lambda function to scale the image

The *CreateThumbnail* Lambda function is used to scale the image in this project. This Lambda function has the following configurations:

- Name: *CreateThumbnail*
- Runtime: ***Python 3.11***
- Architecture: ***arm64***
- Execution role: ***LabRole***

Since you cannot create your own role in the Learner Lab, we use the `LabRole`, which has been created for you in the Learner Lab environment. In real production environment, you need to create an IAM role with policies that allow this Lambda function to get objects from and put objects to the S3 bucket. The role must follow the ***least-privilege principle***.

- Timeout : ***30*** seconds

The default is 3 seconds. It is better to change it to 30 seconds to avoid unexpected timing error.

Once the Lambda function has been created, you can upload a deployment package to add functionality to this function. The deployment package has been provided to you in *lambda-deployment-package.zip*. This package contains the library and full source code to resize images and dowload/upload images to S3 (for best result, please use PNG images). The package is ready to work without any modification.

In order to test this function, you can create a test event with the following input in AWS Lambda Console (via the `Test` tab):

```
{"bucketName" : "your-photo-bucket", "fileName": "your-image.png"}
```

An example could be

```
{"bucketName" : "mlau-photos-bucket", "fileName": "swinburnelogo.png"}
```


Since we are putting new objects to the same S3 bucket, please **DO NOT** set the trigger of this lambda function to `S3 : All object create events`. Doing so will end up having

You are encourage to in ode and understand the logic of this Lambda function. However, you will not ver coding questions about Lambda functions. Having said that, general concep functions work will be asked in the exam.

2.2.8. Rquirement 8 Database with RDS

Same RDS database created in Assignment 1.

You may think that since the Web Server EC2 instances are now in private subnets, access to phpMyAdmin from these servers would require some further configurations. That is correct. However, it is not necessary to go through phpMyAdmin from these servers. It is because we still have a `Dev Server` in the public subnet. And, it is acceptable to manage your DB through the `Dev Server`. Hence, there is no need to further configure the EC2 Web Server instances. That is also the reason why we keep the `Dev Server` instance in the public subnet.

2.2.9. Requirement 9 Security Groups

You need to create four to five security groups, depending on whether you choose to use a NAT Gateway or a NAT instance. Each security group is associated with a tier shown in the architecture diagram as in Figure 1:

- ELBSG: for the ELB created above
- WebServerSG : for all the web server EC2 instances in private subnets
- DBServerSG : for the RDS instance
- DevSG : for the Dev server
- NATSG : for the NAT instance

You need this NATSG if you use a NAT instance. In other words, if you use NAT gateway, you do not need this security group.

ELBSG, WebServerSG, DBServerSG and NATSG, if any, must follow the ***least-privilege principle***, i.e., allowing all traffic from anywhere is NOT acceptable. DevSG does not have to follow the least-privilege principle.

Your RDS instance needs to be in a private subnet. Only WebServerSG security group can access it.

Security groups are *stateful*. See https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html for details.

If unsure about how to set up security groups and IAM roles, or unsure if your security groups and IAM roles are causing problems, you can make them wide open (allowing all traffic from anywhere, full permissions first), then tighten them later once your web application is fully functional.

Testing

The Photo Album Web Application should be accessible through http://[your.elb.dns]/photoalbum.

Using your Photo Album Web Application (http://[your.elb.dns]/photoalbum/photouploader.php), upload a few photos along with their meta-data

- Check the S3 bucket to see if photos are actually uploaded and if their resized versions are created
- Check the database to see if their meta-data is recorded
- The Photo Album Web Application is accessible through the load balancer only
- Terminate servers then check to see if replacement EC2 instances are automatically deployed by the ASG. Thoroughly test the functionality of the web application again once new instances have been launched
- All EC2 targets are healthy
- Test direct write access to your S3 bucket, which should not be publicly accessible
- Double check all security groups and IAM roles, make sure they follow the least-privilege principle.

Marking Criteria of Cloud Project

Table 2. Marking Criteria of Cloud Project

Requirements	Tasks	Total Marks	Student Marks
Part 1 Infrastructure	VPC (vailability Zones, both with publ ts	1%	
	Publ tables route to internet gatev (instance or gateway) resp	2%	
	Security groups properly configured and attached	2%	
	IAM roles properly configured (or used)	1%	
	ASG configured and working correctly	2%	
	ELB configured and working correctly with associated Elastic Public IP address	2%	
	Photos stored in S3 are correctly accessible. S3 bucket policies are correct	2%	
	Lambda configured and working correctly	2%	
	RDS configured and working correctly	1%	
	Sub-Total (a)	15%	
Part 2 Functionality	Web site accessible via ELB	1%	
	Photos and their meta-data displayed on album.php page	2%	
	Photos and their meta-data can be uploaded to the S3 bucket and RDS database, respectively	2%	
	Photos are resized by the Lambda function	2%	
	Sub-Total (b)	7%	
Deductions	Documentaion not as specified or poorly presented [between 0% to 22%] (c)	22%	
	Serious misconfigurations of AWS services being used [between 0% to 22%] (d)	22%	
	Sub-Total (e) = minimum of "(c)+(d)" and 22%	22%	
Total	= (a) + (b) - (e)	22%	

Assignment Submission

Make sure your web site is functional from the due date - check you have started the web server EC2 instance if you have stopped it. Your co-teacher will notify you to stop your web site once the marking is completed.

Submit a single pdf document to your co-teacher via Cloud Campus. No demonstration is required. The document must contain the following:

1. Title page with your name, student id and class
2. URL of your web application (through ELB) so your co-teacher can view your website from their browser (Elastic IP address to be used)
3. *If you assignment is [1] using personal AWS account instead of Academy Learner Lab,* you will need to create an IAM user with proper permissions and provide your co-teacher with the credentials so that they can login to AWS management console.
4. Well formatted Screenshots of records in your database, with appropriate titles.
5. Well formatted screenshots with brief explanation for each step that you have taken, problems that you faced and achievements during your deployment for this cloud project
6. Each screenshot must have your AWS Management Console username/student id visible

1. This assignment is to be completed in a managed AWS Lab environment (e.g. AWS Academy Learner Lab), which is accessible through AWS's Canvas page. For further information of how to access this environment, please refer to your "Accessing AWS Resources" on Cloud Campus.
2. This environment is time-limited until the end of the teaching and comes with US$100 credit. It is your responsibility to use and manage this credit correctly to ensure there will be enough remaining credits for all assignments.
3. Marks will be deducted if your assignment resources are not accessible due to insufficient credits.

That is all for Assignment 2.

[1] This is not a good practice. However, you can do this for learning purposes.