COS80001 - Cloud Computing Architecture

Assignment 2

Developing a highly available Photo Album website

Due date: **9 AM, Monday of Week 11**
Weighting: 15%. The assignment needs to be completed to attain a Credit grade or above in this unit.
Late submission penalty: 10% of total available marks per day.

Prerequisite requirements:

- Successfully completed Assignments 1A and 1B.
- Completed all ACF labs (1-6).
- Know how to use AWS PHP SDK and AWS Network ACLs.

All supporting materials mentioned in this document can be found in the corresponding assignment page on Canvas.

PHP source code has been provided for this assignment. However, you will need to understand how this code works and modify the missing parts. Each student is supposed to add their own specific information in this code; hence, you cannot copy someone else's code.

Objectives

This assignment will extend/modify the infrastructure and program you developed in Assignment 1b. It has the following additional objectives:

1. Create IAM roles to enable EC2, Lambda, and S3 to interact with each other.
2. Restrict access to S3 using S3 bucket policy.
3. Create a lambda function.
4. Create a custom AMI.
5. Create a launch configuration based on your custom AMI.
6. Create an auto scaling group across multiple Availability Zones with policies for scaling up and down.
7. Create an elastic load balancer to distribute service requests.
8. Access control and traffic limitations by using AWS NACLs.

1. Functional requirements of Photo Album website

The PhotoAlbum website is to be hosted on your EC2 web servers. The full source code has been provided to you (*photoalbum.zip*). 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 website 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.

2.1 – 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.

2.2 – 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 version of the photo that was just uploaded to S3. Programmatically, this page performs the following actions:

- When you click the "Upload" button on the page, the photo will be uploaded from your local computer to an EC2 web server.
- The 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, and keywords) 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 named "*resized-<nameOfTheOriginalPhoto>.png*".

For more details, please inspect the supplied source code.

2. Infrastructure deployment

You will set up a VPC with the structure and services as illustrated in Fig. 1. You can set it up on top of the infrastructure developed from Assignment 1b.

2.1 - VPC

The VPC is as per Assignment 1b. The following points should be noted:

- Name: [*FirstNameInitial*][*LastName*]*VPC*. For example, if your name is Bill Gates, your VPC would be named "*BGatesVPC"*.
- Region: us-east-1
- Two availability zones, each with a private and public subnet with suitable CIDR ranges.
- Associate public subnets with a route table that routes to an Internet Gateway
- Associate private subnets with a private route table that routes to the NAT instance. NAT instance is now required because EC2 instances need to upload photos to S3 bucket, which is outside the VPC. The NAT instance is an EC2 instance that is configured as follows:
    - AMI: amzn-ami-vpc-nat-2018.03.0.20210721.0-x86_64-ebs - ami-00a36856283d67c39
    - Source/destination check disabled.

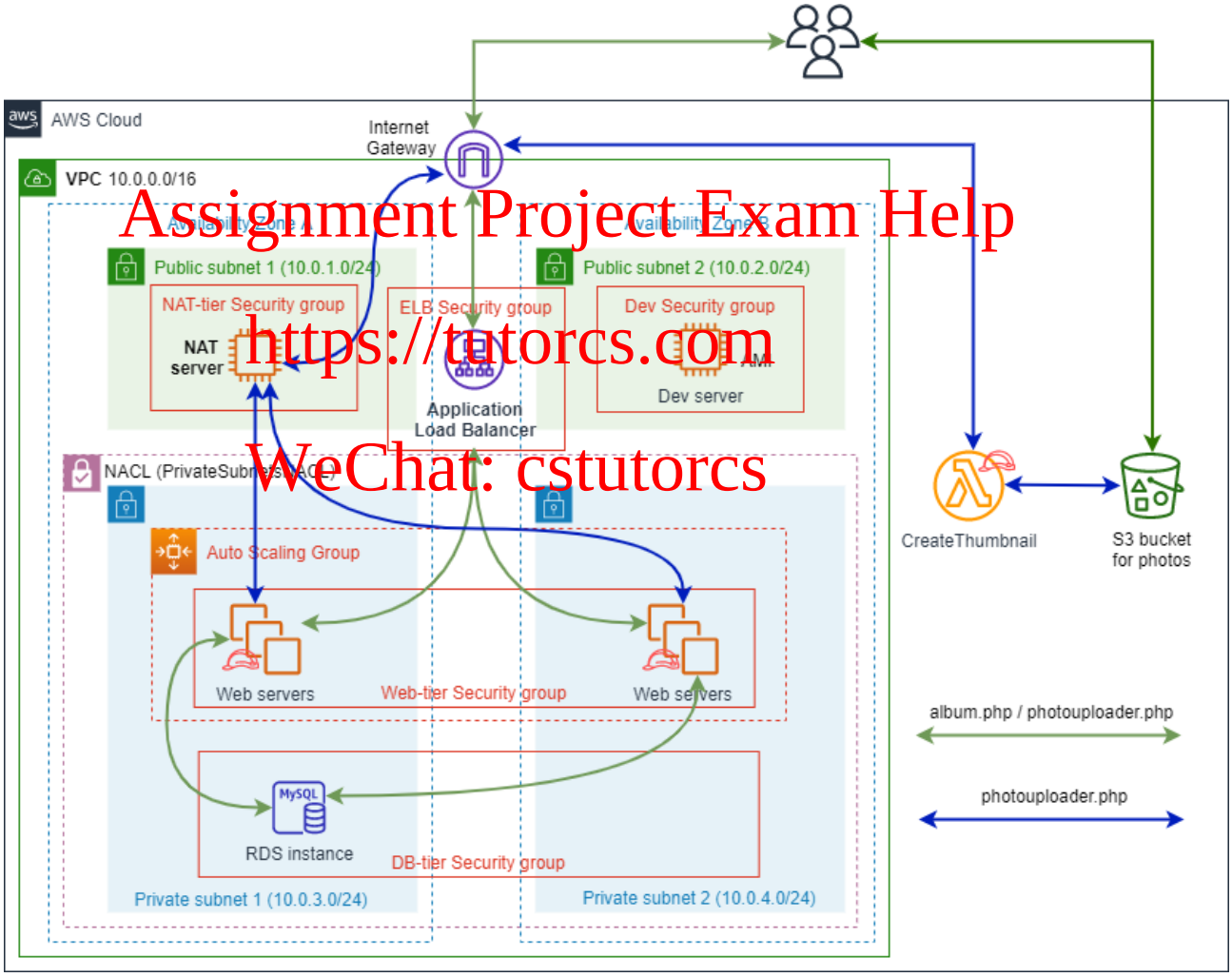

Figure 1 - Architecture diagram

2.2 – S3 photo storage

Photos are to be stored in an S3 bucket, which has been created from Assignment 1. However, in this assignment, the bucket must be **private** so that the photos are not publicly accessible. The photos should be accessible through your website only.

HINT: You can set up an S3 bucket policy that restricts access to a specific HTTP referrer (https://docs.aws.amazon.com/AmazonS3/latest/userguide/example-bucket-policies.html#example-bucket-policies-use-case-4).

2.3 – 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 ELB is running health checks on all instances.
NOTE: The health check path must be correctly configured (e.g., "*/photoalbum/album.php*"). Otherwise, the health checks would fail.

2.4 – Auto scaling

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 3.
- 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.

2.5 – EC2 instances

Your EC2 web server instances should be based on *Amazon Linux 2 AMI (HVM), SSD Volume Type* (similar to the previous assignments). 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. 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 configs once the instances have been launched.
HINT: An ASG can launch instances based on an AMI that has been customized by you.

The Dev server does not receive traffic from the ELB. The Dev server can be used to develop the custom AMI, which would contain everything needed to run the PhotoAlbum website (AWS PHP SDK, Apache web server, source code of the website, etc.). It can also be used to manage your database (through phpMyAdmin – similar to Assignment 1b).

2.6 – CreateThumbnail Lambda function

Create a Lambda function with the following configs:

- Name: *CreateThumbnail*
- Runtime: ***Python 3.7***
- Execution role: An IAM role with policies that allow this Lambda function to get objects from and put objects into the S3 bucket. The role must follow the ***least-privilege principle***.

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 (*lambda-deployment-package.zip*). This package contains the library and full source code to resize images and download/upload images to S3 (for best result, please use PNG images). The package is ready to work without any modification.

TIP: In order to test this function, you can create a test event with the following input:

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

You are encouraged to inspect the source code and understand the logic of this Lambda function.

2.7 – Database with RDS

Same RDS database created in Assignment 1b.

Since the web servers are now in private subnets, access to phpMyAdmin from those servers would require some further configs. This is NOT required. It is acceptable to manage your DB through the Dev server, which is in a public subnet.

2.8 - Security groups

Create five security groups, each is associated with a tier shown in the architecture diagram:

- ELBSG: for the ELB created above.
- WebServerSG: for all the web servers in private subnets.
- DBServerSG: for the RDS instance.
- NATServerSG: for the NAT server.
- DevServerSG: for the Dev server.

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

NOTE: Security groups are *stateful* (https://docs.aws.amazon.com/vpc/latest/userguide/VPC_SecurityGroups.html).

TIP: If unsure how to set up security groups, NACLs, 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) then tighten them later once your web app is fully functional[1].

2.9 – Network ACL

To add an additional layer of security to your web servers, you have been asked to design and deploy a Network ACL that limits ICMP traffic to the corresponding subnets.

- Create a network ACL (named "*PrivateSubnetsNACL*") to block bidirectional ICMP traffic to/from Dev server.

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

Testing

The PhotoAlbum website should be accessible through http://[your.elb.dns]/photoalbum/album.php Using your PhotoAlbum web app (http://[your.elb.dns]/photoalbum/photouploader.php), upload a few photos along with their metadata.

- 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 PhotoAlbum website 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 website again once new instances have been launched.
- All EC2 targets are healthy.
- Test direct access to your S3 photos, which should not be publicly accessible.
- Test the Network ACL bidierectional functionality by sending ICMP traffic between the web servers and Dev server.
- Double check all security groups and IAM roles, make sure they follow the least-privilege principle.

Submission

No demonstration is required. **Make sure your website is running from the due date - check you have started the webserver EC2 instance if you have stopped it.**

Submission is a single PDF document to Canvas. The document must contain the following:

1. Title page with your name, student ID, and tutorial class.
2. URL of your website (through ELB) so the marker can view your website from their browser.
3. *If your assignment is done in your personal AWS account instead of Vocareum*, you need to create an IAM user with proper permissions and provide us with the credentials so that the marker can access your AWS management console.
4. Screenshot of the data records in your database.

COS80001 CCA: Assignment 2 Checklist

Make sure all the following are completed.

Submission Checklist

Student Name: ……

Student Id: ……

Tutorial time: ……

Date of submission: ……

Submit to Canvas:

- ☐ A PDF document file as specified in the Submission section of the assignment specification.

Marking Scheme

Infrastructure Requirements (20 marks)		
VPC configured with 2 AZs both with public and private subnets. Public and private route tables route to IGW and NAT, respectively.	1	
Security groups created and properly configured.	4	
NACL correctly configured.	2	
IAM roles properly configured	3	
ASG configured and working correctly.	2	
ELB configured and working correctly.	2	
Photos stored in S3 have restricted access. S3 bucket policy correct.	3	
Lambda configured and working correctly.	2	
RDS configured and working correctly.	1	
Functional Requirements (10 marks)		
Website accessible via ELB.	1	
Photos and their meta-data displayed on album.php page	3	
Photos and their meta-data can be uploaded to the S3 bucket and RDS database, respectively.	3	
Photos are resized by the Lambda function.	3	
Deductions		
Documentation not as specified or poorly presented (up to minus 30)		
Serious misconfigurations of AWS services being used (up to minus 30)		

Comments