COS20019 - Cloud Computing Architecture

Assignment 2

Highly Available File Manager web site

Due Date: Week 12 (26 November 2024, 11:59 PM)
Weighting: 15%

You must submit the completed template to have your assignment assessed and be eligible to attain a Credit grade or above in the unit.

To complete this Assignment, you will need to have…

- Successfully completed Assignment 1.
- Explored how to use the AWS PHP SDK and Lambda.
- Explored how to setup Elastic Load Balancing, Auto Scaling, and Network Access Control Lists.

Objectives

This assignment will extend / modify the infrastructure and web site you have developed in Assignment 1. It has the following objectives:

1. Use IAM role to enable EC2, Lambda, and S3 to interact with each other.
2. Create a Lambda function.
3. Create a custom AMI with preinstalled software and your website source files.
4. Create a Launch Configuration based on your custom AMI.
5. Create an Auto Scaling Group across two Availability Zones with policies for scaling up and down.
6. Create an Elastic Load Balancer to distribute service requests.
7. Access control and traffic limitations by using Network ACLs.

Important:
In your COS20019 assignments, all AWS resources you create (e.g. EC2 instances, Security groups, RDS database instances, etc.) should have the following additional tags added:

- StudentName (with a value of your name)
- StudentID (with a value of your id)

These tags are in addition to any other tags that are appropriate to add to the resource.
These tags will be used to assist in the assessment of your work

Functional requirements

The File Manager website is to be hosted on your EC2 web servers. The full source code has been provided (*Assignment 2.zip*). Modify the *constants.php* file in the provided code using details from your S3 bucket, RDS database, and Lambda function. The website should be accessible through your ELB at https://[your.elb.dns]/COS20019/file_manager/get_files.php

File Manager (get_files.php)

This page lists all the files 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.

File Uploading (upload.php)

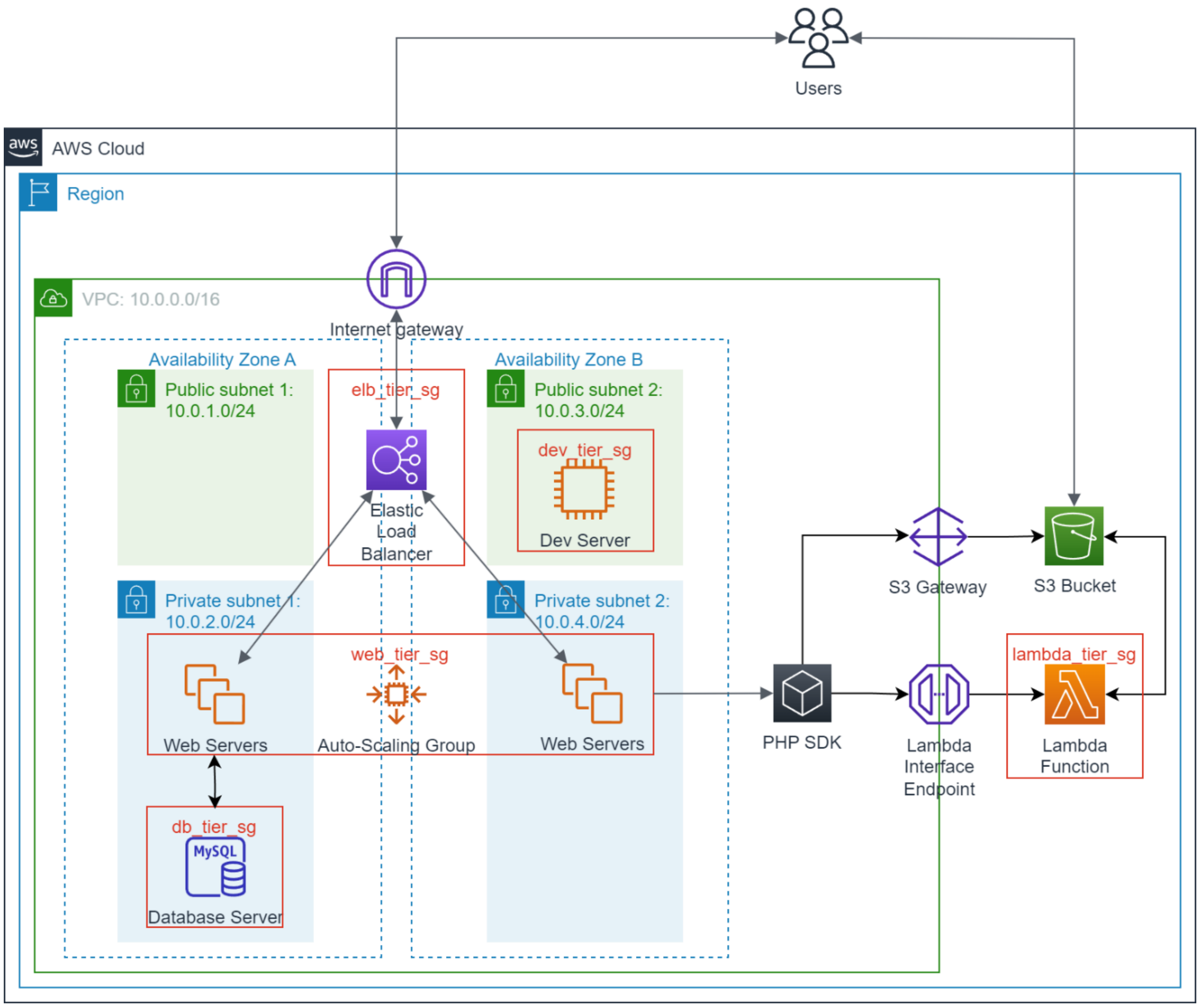
This page allows you to upload a file to an S3 bucket and insert its meta-data into the RDS database. In the meantime, a Lambda function called CreateZip will create a zipped version of the file that was just uploaded to S3. Programmatically, this page performs the following actions:

- When you click the "Upload" button on the page, the file will be uploaded from your local computer to an EC2 web server.
- The file is then uploaded from the EC2 web server to the S3 bucket.
- The EC2 web server inserts the file's meta-data (filename, description, creation date, and keywords) into the database in the RDS instance.
- The EC2 web server invokes the CreateZip Lambda function with the bucket name and the filename in the payload.
- The Lambda function downloads the file in the bucket specified in the payload sent above, zips it, and uploads the zipped file to the same S3 bucket. The zipped file is named "zipped-<nameOfTheOriginalFile>.zip".

For more details, please inspect the supplied source code.

Infrastructure requirements

You will set up a VPC with the structure and services as illustrated in the diagram below.



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

VPC

- The name of your VPC **must** be in the format [*StudentID*]VPC. For example, if your student id is 12345 your VPC would be named **12345VPC**.
- Region: us-east-1.
- Two availability zones, each with a private and public subnet with specified CIDR.
- Associate subnets with their appropriate route tables.
- Setup S3 Gateway to allow for connections from Web Servers in private subnets.
- Setup Lambda Interface Endpoint to allow for connections from Web Servers in private subnets.

Security Groups

Create the following security groups

SG Name	Protocols	Source	Assign To
dev_tier_sg	HTTP (80), HTTPS (443), SSH (22)	Anywhere	Dev Server
elb_sg	HTTP (80), HTTPS (443)	Anywhere	Elastic Load Balancer
web_tier_sg	HTTP (80), HTTPS (443)	elb_sg	Web Server
	All ICMP - IPv4	dev_tier_sg	Web Server
db_tier_sg	MySQL (3306)	web_tier_sg dev_tier_sg	Database Server
lambda_tier_sg	HTTP (80), HTTPS (443)	web_tier_sg dev_tier_sg	Lambda Interface Endpoint

Dev & Web Servers

The Dev Server can be used to develop the custom AMI (containing the web server software and website source code) to be used by the Auto Scaling Group. It can also be used to manage your Database Server.

- Amazon Machine Image: Amazon Linux 2 AMI (HVM)
- Instance type: t3.nano
- IAM instance profile: LabInstanceProfile
- Metadata version: V1 and V2 (token optional)
- User data: install Apache Web Server and PHP (as in Assignment 1a)

This project requires AWS SDK installation on the instance. To do this, SSH into the instance and follow these steps:

- Access the EC2 instance and install composer (command-line installation)
  - https://getcomposer.org/download/
- Install the AWS SDK (Add AWS SDK for PHP as a dependency via Composer)
  - https://docs.aws.amazon.com/sdk-for-php/v3/developer-guide/getting-started_installation.html
  - Run the command in the */var/www/htm/COS20019/file_manager* folder

The Web Server instances are based on the custom AMI and are automatically launched by the Auto Scaling Group. Incoming traffic is accepted from the load balancer. Once launched, the Web Servers can host the website without any further human intervention, meaning that no additional configuration in the instances is needed.

To ensure proper permissions are given to the servers to put objects into the S3 bucket and invoke the Lambda function, ensure that the “LabInstanceProfile” IAM instance profile is assigned to them, and metadata version is set to “V1 and V2 (token optional)”.

Database Server

Same configuration as Assignment 1.

- DB engine version: MariaDB 10.11.8
- Template: Free tier
- DB instance class: db.t4g.micro
- Storage autoscaling: disabled
- Public accessibility: No
- Backup retention period: 0 days

S3 Storage

Files are to be stored in an S3 bucket. Ensure that the bucket blocks all public access.

Lambda

Create a Lambda function with the following configuration:

- Name: CreateZip
- Runtime: Python 3.10
- Architecture: x86_64
- Execution role: LabRole
    - This is to allow the Lambda function to get objects from, and put objects into the S3 bucket

Once the Lambda function has been created, you can paste the script from "lambda_function.py" into the Code source.

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

{"bucketName":"your-bucket-name","fileName":"your-file.*"}

Load Balancer / Target Group

Web requests need to be distributed across web servers in the auto-scaling group. Ensure that your ELB is running health checks on all instances. You can use the Application Load Balancer.

The health check path must be correctly configured (e.g., "*/COS20019/file_manager/get_files.php*"). Otherwise, the health checks would fail.

Launch Template / Auto Scaling

The custom AMI from the Dev Server will be used for the launch template.

- Instance type: t3.nano
- IAM instance profile: LabInstanceProfile
- Metadata version: V1 and V2 (token optional)

You need to define a scaling policy for your auto-scaling group with 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.

The Auto Scaling Group should launch instances into the private subnets.

Network ACL

To add an additional layer of security to your web servers, update the default Network ACL to deny inbound and outbound ICMP traffic.

Testing

Using your Web application, upload a number of files (at least 5) along with their meta-data:

- Check the S3 bucket to see if files are actually uploaded and if their zipped versions are created.
- Check the database to see if their meta-data is recorded.
- The File Manager website is accessible through the load balancer only.
- Terminate servers then check to see if replacement EC2 instances are automatically deployed by the Auto Scaling Group. Thoroughly test the functionality of the website again once new instances have been launched. All EC2 targets should be healthy.
- Test direct access to your S3 files. They should not be publicly accessible.
- Test the Network ACL functionality by pinging a Web Server from the Dev Server. The ping should not be successful.

Marking Scheme

Infrastructure Requirements (10 marks)	**Marks**
VPC with 2 public and 2 private subnets	0.5
Security groups configured and working correctly	2
Network ACL configured and working correctly	1
Auto Scaling Group configured and working correctly	1
Elastic Load Balancer configured and working correctly	1
Lambda configured and working correctly	1
RDS configured and working correctly	1
Uploaded Files stored in S3 have restricted access. S3 bucket policy correct	1.5
S3 Gateway and Lambda Interface Endpoint configured and working correctly	1

Functional Requirements (5 marks)	**Marks**
Website only accessible via ELB	0.5
Uploaded files and their meta-data displayed on web page	1.5
Files and descriptions can be stored in DB and uploaded files stored by PHP in S3	1.5
Files are zipped by the Lambda function	1.5

Deductions	**Marks**
Documentation not as specified or poorly presented	-1 to -5
Serious misconfiguration of AWS services being used	-1 to -5
Plagiarism	-1 to -15