# Part 1: AWS Lambda (4 marks)

## Question 1

**IaaS** consumers require to create a Linux operating system, install and configure Apache, MySQL and PHP. All steps are performed inside the Linux instance via the command line.

**PaaS** consumers launch and configure AWS Elastic Beanstalk, MySQL database in Amazon RDS and Beanstalk and RDS security groups/environment properties. Elastic Beanstalk automatically handles the code deploying, capacity provisioning and web application scaling.

**SaaS** consumers create and access WordPress directly via its official website. Consumers are only responsible to manage their data.

**FaaS** consumers create an AWS lambda function to render template html, install Serverless and deploy the function.

The table below shows the difference and similarities between service models.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **OS** | **Database** | **Application** | **Data** |
| **IaaS** | Yes. Create a new instance. | Yes. Install and configure MySQL. | Yes. Install and configure Apache and PHP. | Yes |
| **PaaS** | No | Yes. MySQL via RDS | Yes. Via Beanstalk | Yes |
| **SaaS** | No | No | No | Yes |
| **FaaS** | No | No | No | Yes |

## Question 2

Provisioning using auto-scaling and elastic load balancing monitors allows consumers to build scaling plans for resources to optimize performance. This approach can be scaled on **Amazon EC2 instances** and scaled for multiple resources across multiple services. On the other hand, AWS lambda creates an **instance of the function** to run the handler method when a new event comes. Once more events come in, the lambda function uses the available instances or create new instances, if needed, to process events concurrently. The lambda concurrency quotas of each region are the fixed numbers. If the function reaches the maximum concurrency, the additional requests fail and return the error.

# Part 2: AWS Lambda – Hello Word Enhanced

## Question 3

Lambda function

Graphical user interface, text, application

Description automatically generated

Test event

Graphical user interface, application

Description automatically generated

Log output

Graphical user interface, text, application, email

Description automatically generated

## Question 4

Graphical user interface, text, application, email

Description automatically generated

## Question 5

Code

Graphical user interface, text, application

Description automatically generated

Permission

A screenshot of a computer

Description automatically generated

## Question 6

Graphical user interface, text, application

Description automatically generated

## Question 7

The event notification is created under the S3 bucket to trigger the lambda function when a file is uploaded to the bucket. The function handler is set as *'lambda\_function.lambda\_handler'* which tells the lambda runtime to invoke the *‘lambda\_handler’* method. The code inside the method reads the data of the uploaded file and print the data to CloudWatch Logs.

When the lambda function (my-s3-function) is created, the role (my-s3-function-role) and the default policies are automatically created. Two additional policies, AmazonS3FullAccess and CloudWatchFullAccess, are added to enable the permission for S3 to trigger the lambda function and for CloudWatch to store logs generated by the lambda function.

# Part 3: AWS EMR (8 marks)

## Question 8

What are the benefits of using AWS EMR instead of using a local cluster?

Compare running MapReduce on PaaS EMR versus running Hadoop on IaaS. (2 marks)

## Question 9

Step 1: Import relevant packages. Add an array list of string allowedWords as a private class variable of the WordCount class. The array list of string are the list of word in dict.txt.

Text

Description automatically generated

Step 2: Create the setup function which gets bucketName and objectKey (path of dict.txt) of the S3 bucket in N. Virginia region. The function reads words in dict.txt line by line and stores the words in the array list of string allowedWords.

Text

Description automatically generated

Step 3: Remain the map function and the reduce class as same as the original file.

Text

Description automatically generated

Step 4: In the main function,

* Update the if condition from otherArgs.length!=2 to otherArgs.length!=3.
* Set bucketName as the third argument, the bucket where dict.txt is placed.
* Set objectKey as the path of dict.txt. In this setting, the file is placed in the dict folder.

Text

Description automatically generated

### Add a step in a cluster

The JAR file takes an input from the input folder (Argument 1) and a dictionary file from the S3 bucket (Argument 3). Then it generates an output to the output folder (Argument 2).

Graphical user interface, text, application, email

Description automatically generated

### The output folder

Once the cluster has completed, the output folder is generated. There are three different output files, part-r-00000 part-r-00001 part-r-00002, inside the folder.

Graphical user interface, application, email

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

part-r-00000 does not contain any result, but part-r-00001 and part-r-00002 contain the number of words listed in the dictionary file.

Text, table

Description automatically generated Graphical user interface, text, application, chat or text message

Description automatically generated