# 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.