**Tools Evaluation**

                          Functions: AWS Lambda & Google Cloud Functions & Microsoft Functions

|  |  |  |  |
| --- | --- | --- | --- |
|  | AWS Lambda | Google Cloud Functions | Microsoft Functions |
| **Scalability & availability** | Automatic scaling (transparently) | Automatic scaling | Manual or metered scaling (App Service Plan), or sub-second automatic scaling (Consumption Plan) |
| **Max # of functions** | Unlimited functions | 1000 functions per project | Unlimited functions |
| **In-browser code editor** | Yes | Only with Cloud Source Repositories | Functions environment, App Service editor |
| **Pricing** | 1M requests for free, then $0.20/1M invocations, plus $0.000016/GB-sec | 1M requests for free, then $0.40/1M invocations, plus $0.00000231/GB-sec | 1 million requests for free, then $0.20/1M invocations, plus $0.000016/GB-s |
| **HTTP(S) invocation** | API Gateway | HTTP trigger | HTTP trigger |
| **Supported languages** | JavaScript, Java, Ruby, C#, Go, Node.js  PHP and Python | Only JAVA Script | C#, JavaScript, F#, Python, Node.js |
| **Environment variables** | Yes | Not yet | App Settings and Connection Strings only from App Services |
| **Response Time** | =400ms | >400 ms |  |

As the above table shows that although all these three functions are auto-scaling,  AWS have several distinctive advantages. For instance: AWS Lambda functions support much more languages than the other two, so it’s much more advantageous for developers who use several programming languages. Also, it has the lowest pricing among these three functions. And the AWS has in-browser code editor, which enable developer develop in the browser and it’s convenient .

Platforms: AWS & Google Cloud Platform & Azure

|  |  |  |  |
| --- | --- | --- | --- |
|  | AWS | Google Cloud Platform | Azure |
| Storage Pricing | $0.03 (standard S3) | $0.03 (standard) | $0.044 (Average) |
| Infrastructure Deployment | Most extensive(more than 40 locations) | Fall behind outside U.S(33 locations) | Fall behind outside U.S(32 locations) |
| Instance Types | 38 | 18 | 33 |

As the above table shows that AWS platform has the lowest storage pricing and also the most extensive infrastructure deployment among the 3 platforms. In this case, the AWS platform enables major company like UPMC health insurance to deploy their service all over the world in more than 40 locations. Besides, because the related instance types of AWS is much more than the other two platforms, it would be much more convenient for the developers to develop related instance in the same platform. In a conclusion, AWS is the best choice of these three platforms so we choose AWS to develop our system.

APIS: Amazon Recognition & Google Vision & Azure Computer Vision

|  |  |  |  |
| --- | --- | --- | --- |
|  | Amazon Rekognition | Google Vision | Azure Computer Vision |
| Pricing | $1 / 1000 events | $1.5 / 1000 events | $1.5 / 1000 Events |
| Average Response Time  (*N. Virginia, 1000 files, 10 at a time*) | 1.1s | 0.98s | 2.1s |
| Accuracy | 2st accurate | Most accurate | Less accurate |
| Image Size Limit | 15Mb / Image from S3 | 4 MB / Image | 4 MB / Image |

As the above table shows that Amazon Rekognition has the highest image size limit which will totally expand our service range, because almost all the pictures in the Internet is less than 15 MB. But as for Google Vision and Azure Computer Vision, 4 MB’s limit sets a high requirement for the picture’s quality and size.  As for the accuracy, after numerous tests,we discovered that the Google Vision’s recognition is the most accurate and Google Vision has the most diverse and precise labels. Additionally, as for the pricing and average response time, there is not much differences between Amazon Rekognition and Google Vision. So in order to improve the accuracy and performance of our system, we decide to use Amazon Rekognition mainly. To be more precisely,  3 APIs have different weighting factor . For instance, for a picture which worth 1 point, GCP weights 0.8 points because it is the most accurate after numerous testing and the other two APIs only counts 0.1 each, because we use