# Celebrity Face Recognition

Stefan Klas
sk15962@bristol.ac.uk
University of Bristol

***Abstract*— This paper presents a celebrity face recognition application which has been developed using Amazon Web Services(AWS). It allows the user to upload images containing famous celebrities they may not know and identify who they actually are. The application can be run at: *http://loadbalancer-1617859900.us-east-1.elb.amazonaws.com/***

## I. INTRODUCTION

With today's digital age, there is a constant overload of celebrities to keep up with and know about. Whether it is actors, politicians, athletes or musicians, it is easy to start forgetting the names of who they actually are. The phrase "I've seen him before but can't quite remember his name" is common. To solve this problem, this application has been developed to allow users to upload images they have containing famous people and help identifying them. Furthermore, for those who want to just test their knowledge of celebrities, this application can also act as a game, challenging themselves against what others have uploaded. The project has been implemented by making use of fundamental cloud computing knowledge with the aid of Amazon Web Services platform.

## II. CLOUD COMPUTING

Cloud Computing is the delivery of computing services over the Internet. This means that instead of having to purchase the required resources to host the service yourself, third-party cloud infastructure can be made use of through a URL. With this comes many benefits including instant availability, large amounts of flexibility to increase and decrease resources when required and at a far lower cost due to economies of scale. Furthermore maintenance for the resources can also be kept at a minimum which helps many organisations to cut costs and focus primarily on their core business.

Multiple third party cloud platforms exist which has meant that anyone can launch an application in no time with minimal effort. The provider may provide many different types of cloud computing service models, of which the standard three shown in Fig 1 are Infrastructure as a Service, Platform as a Service and Software as a Service. Each is offering increasing levels of abstraction for the developer and therefore an appropriate selection has to be made to suit the use case. Infrastructure as a Service provides the low level access to cloud computing resources, including networking, virtual machine selection, security and storage. Platform as a Service provides less flexibility as management of the underlying infrastructure is left for the cloud provider. Instead application developers can develop their software solutions through toolkits and standards provided. Software as a Service means users gain access to the application software and databases. This can often follow the "on-demand" principle where costs are based on a pay-per-use basis .

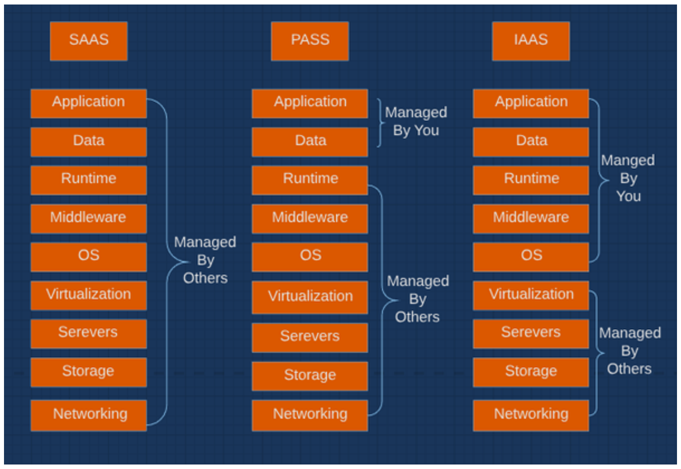

Fig. 1. Types of Model Services (image by Krishna Srinivasan )

## III. ARCHITECTURE OF APPLICATION

The application implemented makes use of two different service models. The first, Infrastructure as a Service(IaaS), occurs through the use of Amazon Elastic Compute (EC2) instances. The second, Software as a Service(SaaS), comes from using Amazon Rekognition.

### *A. EC2*

EC2 instances [1] are compute nodes that provide secure, resizable compute capacity through a virtual machine. It has the main advantage of being very inexpensive or completely free to run. Multiple different Amazon Machine Images are available but the default linux os kernal 4.14 was selected. Two EC2 instances where started for the application, with each placed in a different availability zone. This would be essential to provide some fault tolerance [5] if a particular availability zone was affected by an outage as the remaining instance would continue to operate.

### B. Amazon Rekognition

Amazon Rekognition [2] is a SaaS tailored to image and video analysis through the Rekognition API. It comprises of a large array of different deep learning visual analysis tools that can readily be used. One of these is the functionality to detect celebrity faces against a large stored database of famous people. Making use of this API has two main benefits. Firstly, having this database means there is no requirement to conglomerate the database yourself. Secondly, there is also no need to write the machine learning algorithm for facial detection. Both algorithm and database will constantly be updated and improved in the background, resulting in a better functioning application with no extra work involved. This in essence demonstrates the real advantage of SaaS.

The architecture shown in Fig 2 also consists of two other components which are the Amazon Simple Storage Service (S3) and Amazon Elastic Load Balancer

### C. S3

S3 [3] is a very cheap storage space offered by Amazon that can be used to store any kind of objects inside buckets. Having this available means that EC2 instances can have minimal storage as the images are not stored locally which minimises the cost. Furthermore, uploaded images are then available for all instances to download and view, which is exactly the functionality needed.

### D. Elastic Load Balancer

The Elastic Load Balancer serves the functionality of ensuring some fault tolerance, distributing the traffic load between the EC2 instances and allowing for easy scale up. The fault tolerance means that if one instance stops running due to a hardware or software failure, the application will remain available if the fault has not affected the other availability zone. The load distribution also helps in keeping the application as responsive as possible for the user using the least outstanding requests routing algorithm. Health checks on the instances are made every 30 seconds and a healthy instance with the least pending requests is chosen. Scaling the application is as simple as creating more EC2 instances and notifying the load balancer of their existance. This is very little effort but currently has to be done manually.

## IV. The Workings of Application

The application has been implemented using Flask as the webserver. This is a framework that uses Python and Jinja2 in order to deploy both the front end and back end components. This has then been containerized using Docker and run on each EC2 instance. All necessary installation requirements are found within the Docker file which makes the application very portable. It can easily be shipped onto other cloud providers if necessary or installed on EC2 instances running different OS image versions. Containerization also allows for exploration of more advanced infastructure scaling techniques using Amazon Elastic Container Service and Amazon Elastic Container Service for Kubernetes which is discussed below in section V.

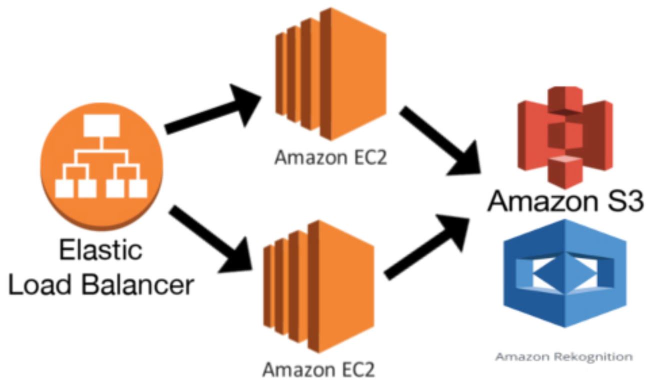

Fig. 2. Architecture of Application

When accessing the application, the user first arrives at the Domain Name Service(DNS) for the load balancer, accessed through port 80. The load balancer will scan the available instances within the Virtual Private Cloud (VPC) and redirect traffic along an individual subnet to a particular instance. Each instance works with a simple security group that allows all inbound traffic through port 80 and redirects this to the exposed port within the docker container. Having the security group open to all traffic makes it easy to test and debug whether each instances is operating correctly. However, this could be a security issue, making the application vulnerable to a DDOS attack and should therefore be restricted before deployment. The flask webserver will then make a retrieval request from S3 of all uploaded images which are downloaded to a local volume within the Docker container.

The images can be clicked on to be expanded and an identification request is made. A JQuery ajax request is sent to the webserver to retrieve celebrity identification using the Rekognition API. If the response has been successful, the returned value is inserted into the caption below. Otherwise, if unsuccessful, the string ”No Name” is returned.

Uploading the image, involves first storing the image locally within a volume inside the Docker container. The PIL python package is then used to compress it to a thumbnail before it is sent to a bucket in S3. The local copy is then deleted as is not needed. Once it has successfully uploaded to S3, the image is available to be downloaded for all instances in the VPC.

## V. Scaling

There are some significant limitations to the current scalability of the architecture. The most pronounced issue is that it lacks automation. CPU, memory and network usage can be observed on Amazon CloudWatch but manual creation and termination of instances is still required. To solve this problem, the most flexible options available would be to transfer the application over to Amazon Elastic Container Service (ECS) or Amazon Elastic Container Service for Kubernetes (EKS).

Both are systems for automating, deploying, scaling and managing containerized applications. It creates a cluster of nodes that can run Docker containers. As a result, the application instances launch must faster on demand as there is no requirement to set up a virtual machine.

Initially a cluster is creating containing a master node, with the role of communicating and monitoring worker nodes. ECS eliminates the need to operate your own cluster management or worrying about scaling whilst EKS gives this extra flexibility. In EKS, an application service can be created using Kubernetes [6] which is described using a yaml file. This can include the number of nodes, the pods on each node and the CPU/RAM usage required before more nodes are instantiated. Once the service has started, constant pooling of worker node conditions are made to ensure traffic load is spread and sufficient resources are available. This can be closely monitored using the Kubernetes dashboard, which prints logs of each node in operation. If worker nodes are being overloaded, the master node will instruct the creation of more nodes to ensure the application remains responsive.

Both systems also has a big advantage when software updates for the application are required. By specifying a rollingUpdate, the cluster ensures that there will always be instances running whilst others are being updated and therefore the application does not need to be put offline at any point.

ECS has the big advantage over EKS that it is free to use whilst EKS costs around $0.20 per hour. This is important to consider as the extra flexibility provided by EKS may not be justified by the extra cost. Particularlly for this application ECS would be the prefered option available, requiring little extra development time to be deployed.

## VI. CONCLUSIONS

This paper has explored some of the basic cloud computing techniques through the implementation of a celebrity facial recognition application using Amazon Web Service. Different architectures have been discussed to serve the best cost effectiveness and scaling flexiblity that would be required for the deployment of the application.

## REFERENCES

[1] Amazon Elastic Compute Cloud API https://docs.aws.amazon.com/AWSEC2/latest/APIReference/Welcome.html

[2] Amazon Rekognition API https://docs.aws.amazon.com/rekognition/latest/dg/API_Reference.html

[3] Amazon Simple Cloud Storage Service API https://docs.aws.amazon.com/AmazonS3/latest/API/Welcome.html

[4] Flask Webserver http://flask.pocoo.org/

[5] Designing Fault Tolerance Applications on AWS https://media.amazonwebservices.com/AWS_Building_Fault_Tolerant_Applications.pdf

[6] Kubernetes Basics https://kubernetes.io/docs/tutorials/kubernetes-basics/