CAB432 Assignment 2 Individual Report

Name: Mark Burton
SN: 9801154
Partner: Mitchell Traynor

As part of assignment 2 a scaling cloud-based application was developed. The accompanying group report describes the design and implementation of the application and Figure 1 shows the overall architecture diagram.

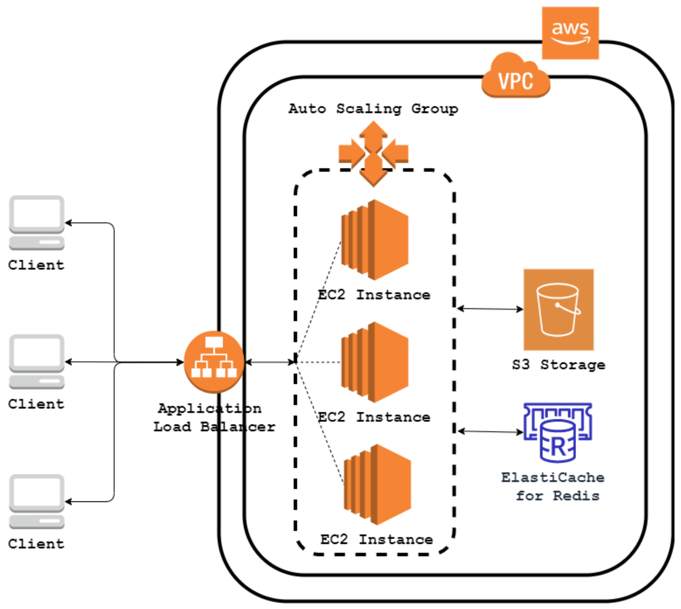

Figure 1 - Architecture diagram of the application developed for assignment 2.

The application is stateless in that server instances do not retain local resources that would be unrecoverable should the instance fail. Instead, all data is retained in cloud based S2 storage or ElastiCache for Redis in-memory cache which are shared between all instances. This ensures that should an instance fail it will not have any effect on required resources and future requests can be seamlessly handled by other instances in the scaling group.
There are no compromises from true statelessness since there is no reliance on transient caches. Even if there is a cache miss the application is still able to function correctly by instead requesting the required data from an API endpoint.
S3 was selected as a suitable long-term storage solution as it provides low latency due to its eventual consistency model and allows unlimited storage at a relatively cheap cost. Further, it can scale on demand making it resilient to potential lambda spikes and can be easily secured against public access. ElastiCache for Redis was selected for short-term caching as it provides an easy solution to a common cache shared between all instances. This eliminated the reliance of transient data being cached locally on instances and ensures that the application is truly stateless.
A potential alternative to S3 storage is DynamoDB which is a NoSQL database rather than an object storage solution. DynamoDB maintains many of the benefits as S3 in that it is a highly scalable and cost-effective solution, however, it has strong consistency compared to the

eventual consistency model of S3. The decision to use S3 storage over DynamoDB came down to the developers having better knowledge of S3.
For scaling an application load balancer was used to distribute incoming traffic equally across existing EC2 instances. The EC2 instances themselves exist within an auto scaling group which is used to create or terminate instances used to share the load when the CPU usage across the group exceeds or subceeds a specified threshold. For this assignment, a CPU usage threshold of just 5% was set for the group. The reason for this was to easily demonstrate application scaling and to ensure that the application performs optimally under a simulated load created by Postman. The number of instances allowed in the scaling group was set to a minimum of 1 and a maximum of 5 instances.
For a real-world deployment of the application cost becomes much more relevant, therefore, a more suitable CPU usage threshold would be 80-90% which would reduce the number of running instances, which each incur cost.

In a hypothetical situation where the application is deployed on a global scale, the architecture must be modified to handle the increased client demand. Figure 2 shows a revised architecture diagram of this global application.

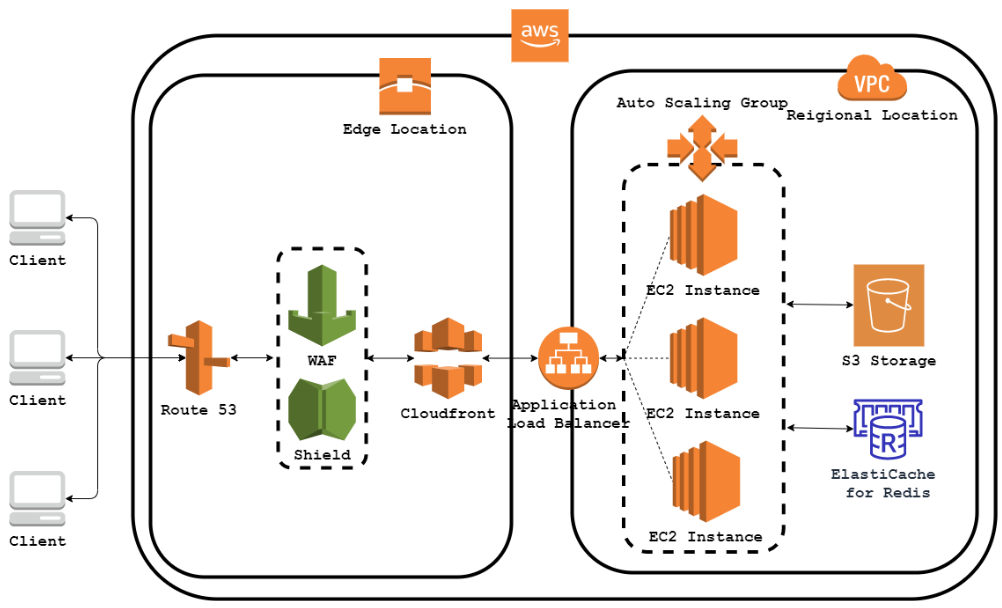

Figure 2 - Architecture diagram of a hypothetical global implementation of the application developed for assignment 2.

The persistence choices that were made for the application are appropriate for a global scale with S3 featuring cross-region replication and ElastiCache able to scale globally, across many shards of nodes, provided that cluster mode is enabled. They are each able to maintain statelessness at scale by ensuring that any and all EC2 instances are able to access them as shared resources and there is never a dependency on transient caches when managing the needs of users. Further, due to the cross-region distribution of the persistence choices, they are able perform to a consistent latency worldwide.
When scaling the application over large pools of EC3 instances, scaling based upon a requests per second threshold rather than CPU usage would offer a better representation of user demand resulting in more effective application scaling. Additionally, the use of

Microsoft Cognitive Services API in the application may require further design consideration as the standard pricing tiers only allow up to 250 calls per second.
The overall consistency of the application would remain eventual as the only component that could compromise this consistency status, ElastiCach for Redis cluster (not able to guarantee strong consistency), is complemented by a cache miss redundancy call to the relevant API.
The split between relational and entity-based storage in the application would be close to equal as the use of S3 storage, to store historical trend searches for future re-viewing, is relational-based. Whereas, the use of Elasticache for Redis, to cache the current image and news data for the trends of a searched location, is primarily entity-based. On a global scale, however, since cached data is available to all entities, those searching the same locations would be able to access the already cached data as if it was relational-based storage.
With respect to additional cache levels required, the main difference between figure 1 and 2 is the addition of an "edge location" comprising of three main components; Route 53, WAF/Shield and Cloudfront, which when used together, according to AWS, provide comprehensive availability protection against all known infrastructure (Layer 3 and 4) attacks. Route 53 is a scalable cloud domain name system web service which routes commonly used web addresses such as "www.aws.amazon.com" to their numeric IP addresses such as "192.168.0.1". WAF is an acronym for "Web Application Firewall" and helps to protect against common exploits that may affect user experience. It allows developers to create security rules that control how and what traffic is able to access the application and is capable of blocking SQL-injection and cross-site scripting attacks as well as some resistance to distributed denial of service (DDoS) attacks. To complement WAF, Shield should also be employed to further mitigate malicious attacks and provide AWS refunds when aggressively scaling due to DDoS related spikes. Cloudfront acts as an 'edge cache', or, 'content delivery network service' which is able to securely deliver application data to clients from a regionally local location with low-latency and high transfer speeds. This helps to ensure that clients are able to access results and content faster than potential competitors.
In the case of a globally popular web application one of the primary considerations is user experience. As such, it is vital that that the application be available at all times, and should be able to scale far beyond the anticipated requests per second threshold. Since AWS Shield will refund any costs incurred due to upscaling, as a result of legitimate DDOS attacks, the maximum EC2 allowance of the auto scaling group should be set as high as possible. Similarly, the minimum number of instances should be set to a reasonably high, cost permittable, level to ensure optimal user experience even when experiencing transient traffic spikes.

One final note to make which is relevant to both the current and hypothetical global version of the application is the importance of logging of cloud data. It is vital that data be available to help in identify the cause of any malicious attacks or unanticipated excessive traffic scenarios, therefore, all available data should be initially logged and retained based upon a hierarchy of relevance.