SINGAPORE POLYTECHNIC | SP

AWS Cloud Foundations

EP0404

Assignment: **LAB3 PBIL Part 2**

Class: **EL/EP0404/FT/02**

Student Number	**Full Name**
P1904444	Chen Jieyang
P1904189	Lee Yi Terng
P1904428	Eve Chiang Yu Zhen

Submitted to: Mr Tan Chee Seng

Date: 14 Aug 2020

1. Introduction

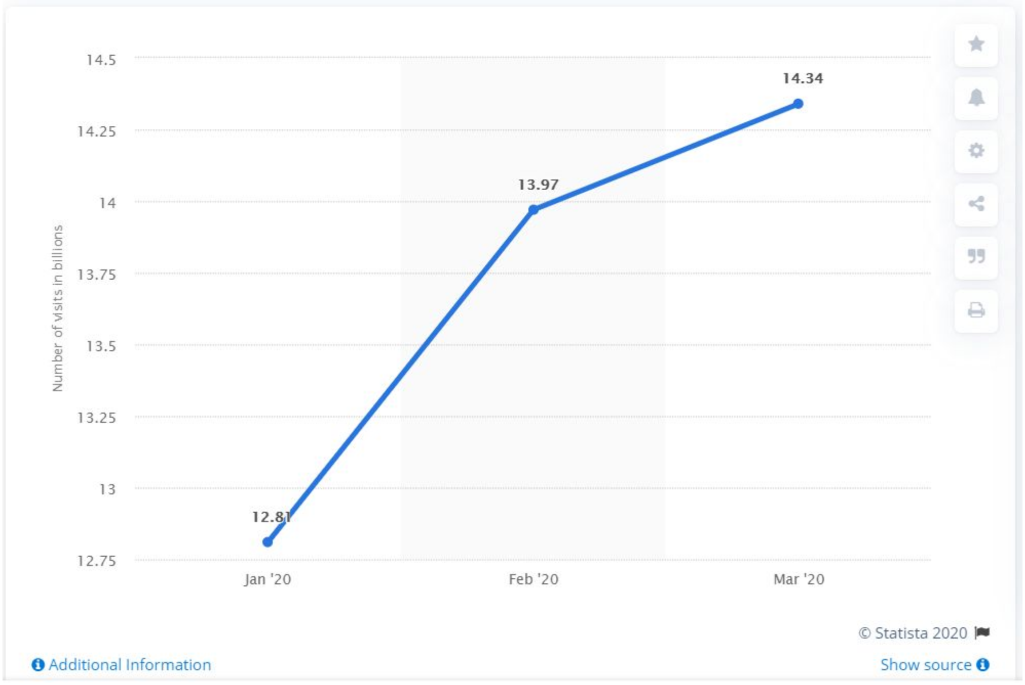
Online shopping is becoming a more popular choice for people to purchase products. The main reason is for ease of access and convenience. Due to the Covid-19 outbreak, it has caused people all over the world to be quarantined at home, unable to access traditional brick and mortar stores. This has restricted their options to online shopping, causing a massive surge in traffic to e-commerce sites. Not every company has the capability to handle such a high amount of traffic, especially smaller businesses who have limited computational capacity, which leads to poor performance. It may even cause the whole application/website to crash because too many users are trying to access them at once.

Companies that use on-premises infrastructures are not able to upgrade their systems fast enough to meet the needs and demands of their consumers. The logistical work and capital needed to improve the hardware of on-premises infrastructure lead to a long ramp-up time to catch the rising markets. Not only that, but the costs incurred by procuring additional resources is also very high. Furthermore, once this surge is over, the companies will have excess capacity and resources that are idle and wasted.

2. Main issues

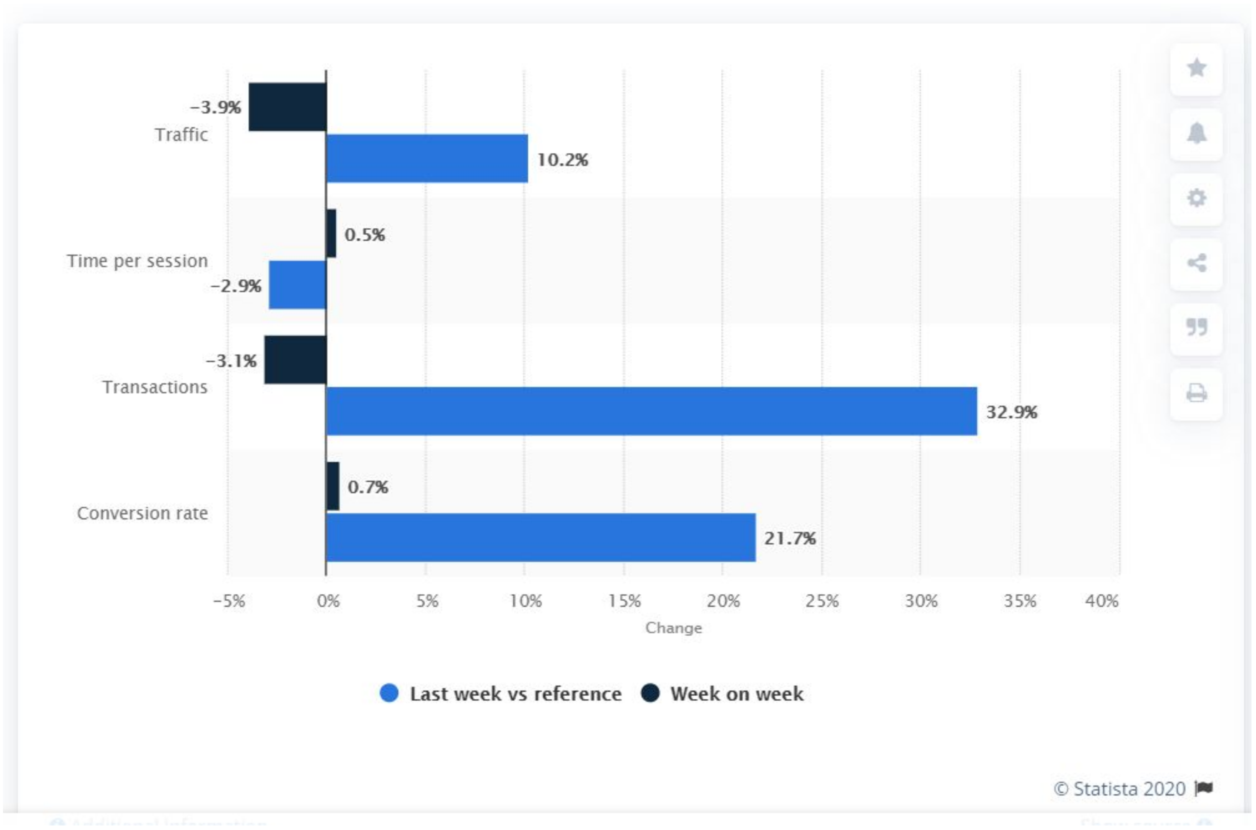
Problem statement:

An increase in traffic on E-commerce sites due to the pandemic, causing problems such as websites crashing and poor performance.



COVID-19 impact on global retail e-commerce site traffic 2020. Published by J. Clement, Jul 17, 2020

Retail platforms have undergone a 6% increase in global traffic between January and March 2020, and retail websites generated an estimate of 12.81 billion worldwide visits in January 2020 to 14.34 billion visits in march 2020. This is due to the global coronavirus pandemic, which leads to our problem statement. (shown in the diagram above)



Coronavirus global online traffic impact as of June 2020 Published by J. Clement, Jun 19, 2020

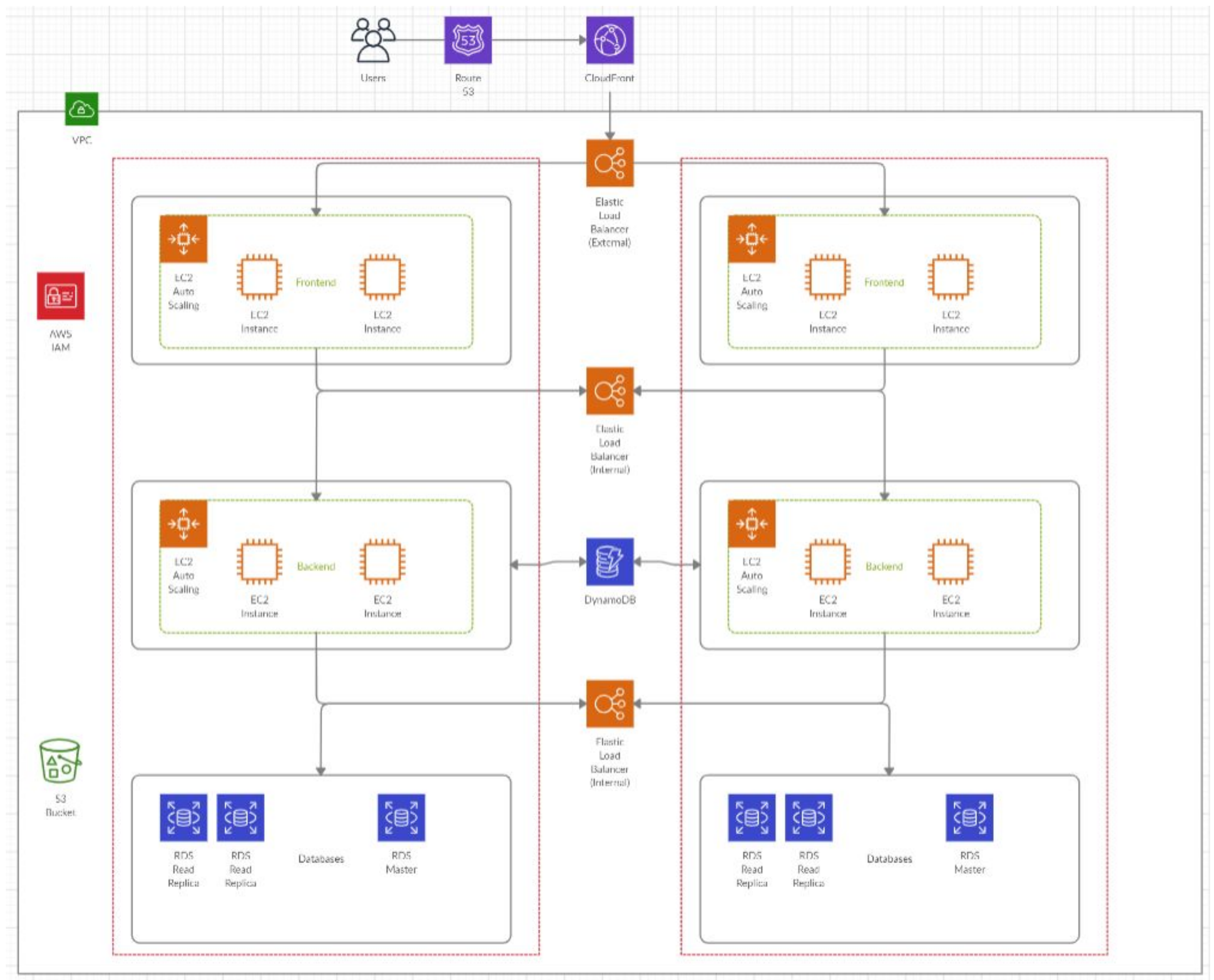
As of June 2020, the online traffic across 20 different countries has increased by 10.2% compared to January and February 2020 (shown in the diagram above)

Currently, to improve one's infrastructure to catch up with the demand of the consumers due to the pandemic, one must quickly decide to upgrade the hardware and allocate resources needed as fast as possible. However, once the coronavirus pandemic is over, and the demands of consumers drop back to normal, there would no longer be a need for all the resources. Furthermore, there is also a need to maintain the resources to keep it running, and it can cause a lot of maintenance costs.

Hence, the solution to tackle this problem is to build a cloud infrastructure that is scalable, elastic, cost-effective, highly available, fault-tolerant and secure. This will enable e-commerce platforms scale to meet high traffic demands and scale down to reduce wasted resources when traffic is low. We have turned to AWS to make this happen.

3. Alternative Solutions

3.1 Alternative Solution #1



3.1.1 AWS Services used in Alt. Solution 1

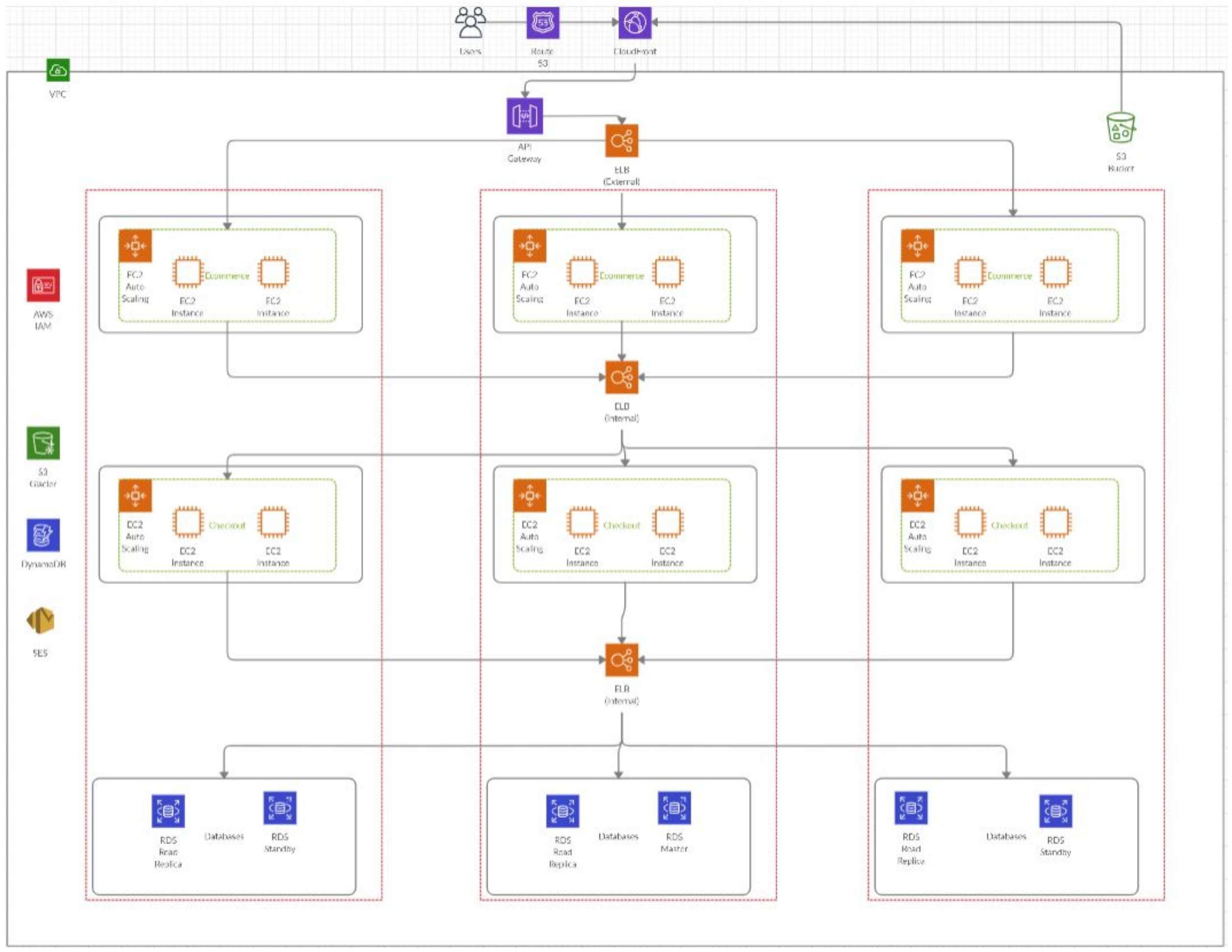
- Route 53
- Cloudfront
- Virtual Private Cloud (VPC)
- EC2
- S3 Bucket
- DynamoDB
- Relational Database Service (RDS)
- Identity and Access Management (IAM)

3.1.2 AWS Architecture in Alt. Solution 1

The architecture of alternative solution 1 consists of a 3 tier web application deployed across 2 Availability Zones (AZ). In the front-end web application tier, it comprises 2 EC2 instances in each Availability Zone (AZ) inside an Auto Scaling group that is behind an Elastic Load Balancer (ELB). In the back-end application tier, it comprises 2 EC2 instances in each Availability Zone inside an Auto Scaling group that is behind an Elastic Load Balancer (ELB). The EC2 instances in this application tier are linked to a central DynamoDB database service that is shared amongst the 2 AZs. In the database tier, one Amazon RDS instance is deployed in each Availability Zone (AZ) with 1 master user, and 2 read replicas configured on top of that.

AWS Identity and Access Management allow administrators to manage the infrastructure effectively and securely. Amazon Simple Storage Service (S3) bucket used to host the web application's data in all 2 Availability Zones.

3.2 Alternative Solution #2



3.2.1 AWS Services used in Alt. Solution 2

- Route 53
- CloudFront

- Virtual Private Cloud (VPC)
- API Gateway
- EC2
- Identity and Access Management (IAM)
- S3 Bucket
- S3 Glacier
- Simple Email Service (SES)
- Relational Database Service (RDS)
- DynamoDB

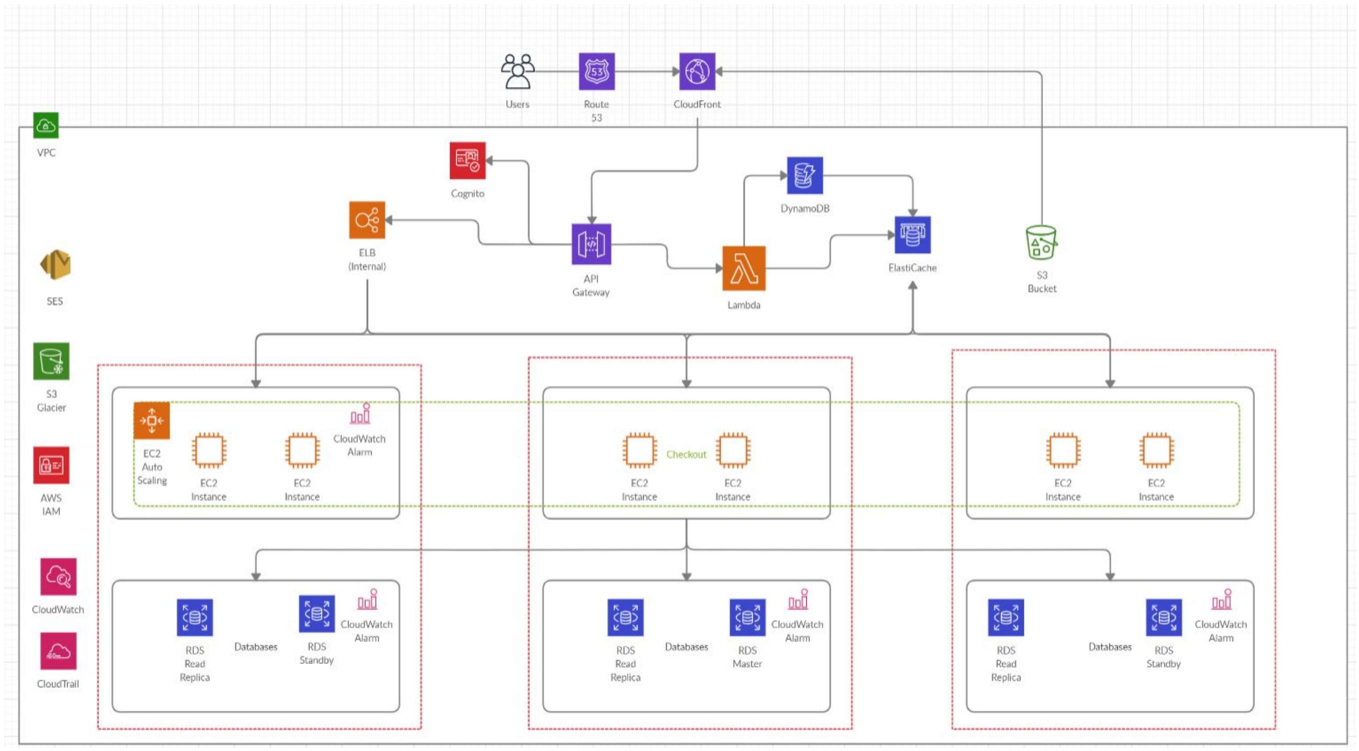
3.2.2 AWS Architecture in Alt. Solution 1

The architecture of alternative solution 2 consists of a 3 tier web application being deployed across 3 Availability Zones (AZ). In the front-end web application, it comprises 2 EC2 instances in each Availability Zone (AZ) inside an Auto Scaling group that is behind an Elastic Load Balancer (ELB). In the back-end application tier, it comprises 2 EC2 instances in each Availability Zone (AZ) inside an Auto Scaling group that is behind an Elastic Load Balancer (ELB). In the database tier, one Amazon RDS instance is deployed in each Availability Zone (AZ) with 1 master and 1 read replica being configured.

AWS Identity and Access Management allow administrators to manage the infrastructure effectively and securely. Amazon Simple Storage Service (S3) bucket is then used to host the web application's data. The S3 bucket also acts as an AWS route 53 failover in the event where the main application fails the route 53 health check and is unavailable. Amazon S3 Glacier is configured for data archiving and backups. Amazon Simple Email Service (SES) is used to enable email exchange on the web application.

4. Proposed Solution

4.1 Proposed Solution



The AWS architecture in the proposed solution has been designed with many factors in mind, such as fault-tolerance, stability, high availability, cost-effective, scalability, elasticity and security. The main objectives were to tackle the issues faced by e-commerce websites as explained in section 2.

4.2 AWS Services used in Proposed Solution

- Route 53
- CloudFront
- API Gateway
- Cognito
- Lambda
- DynamoDB
- ElastiCache
- EC2
- S3 Bucket
- S3 Glacier
- Identity and Access Management (IAM)
- Simple Email Service (SES)
- Relational Database Service (RDS)
- CloudWatch
- CloudTrail

4.3 Architecture of Proposed Solution

The architecture of the proposed solution is a hybrid design that consists of a serverless multi-tier web and database application that is being deployed across 3 Availability Zones (AZ). In the serverless tier, it comprises many AWS services such as API Gateway, Lambda, ElastiCache and DynamoDB. This tier handles most of the front-end web application functions. In the next tier, it comprises 6 EC2 instances that are divided among all 3 Availability Zones (AZ). The EC2 instances are grouped inside an Auto Scaling group that is configured together with an AWS CloudWatch Alarm. In the database tier, one Amazon RDS instance is deployed in each Availability Zone (AZ). However, there is only 1 central RDS master instance in AZ 2 while the other AZs have RDS read replicas configured and 1 RDS standby instance set up.

4.4 Flow of Proposed Solution

When a user connects to our website, Amazon Route 53 directs him to a CloudFront distribution which is serving our website that is hosted in Amazon S3. The user then logs in and authenticates using Amazon Cognito User Pools and is given access to APIs on the website with Identity Pools.

The APIs on the static S3 website are linked to the API Gateway, which handles the various requests that a user sends, such as searching for products, adding of items to cart and many others.

One example is the adding of items to cart, which is handled by a Lambda Function. The Lambda function retrieves cart information from DynamoDB and updates it as required by the user’s action. The function then stores it back into DynamoDB for future use. ElastiCache is used to cache DynamoDB results in its in-memory data stores in order to ensure lower latency and faster retrieval times, which will help enhance and boost our application.

EC2 instances handle the checkout process. When a user chooses to check out, the API Gateway sends a request to the Elastic Load Balancer, which will evenly distribute load between all the instances in all three availability zones using Cross-Zone Load Balancing.

The EC2 instances will process the checkout by retrieving cart information from ElastiCache and process payment. After which, it will send an email confirmation to the user using Amazon Simple Email Service.

The user’s transaction information is then stored in a RDS database. If a user wants to view his transaction information, the EC2 instance retrieves it from an RDS read replica as well.

Amazon CloudWatch and Amazon CloudTrail are used to monitor resources and log all API calls for auditing and security purposes. Monitoring resources also help us to see where we can reduce costs as well as when to scale our solution.

Server, access and monitoring logs are all stored inside S3 buckets and subsequently into Amazon Glacier for archival and storage purposes for future audits.

AWS IAM is used to limit access to resources and to create IAM users and groups for the developers and staff who will be managing the application.

4.5 Strengths of Proposed Solution

High Availability & Fault Tolerance:

The cloud infrastructure has been designed to offer high availability to all users by deploying a multi-AZ infrastructure.

Firstly, each AZ is a fully isolated partition of the AWS regional infrastructure. Each AZ may be housed in a different data centre that may be in a different geographical location. If any of the data centres experience issues such as redundant power, slow network connectivity or natural disasters, only the affected AZ will experience downtime while the other AZs will not be affected. In cases whereby the main AZ database is affected, there is a failover to another AZ. As we are using a triple-AZ architecture, the application is fault-tolerant of up to 2 AZs.

Secondly, RDS Multi-AZ deployment is used to set up the database tier in our architecture. It allows us to have a synchronous standby replica of the database that is provisioned in different AZs. This increases data availability across our infrastructure. In cases of a failover, the standby RDS instances will be promoted to the master instance to manage all RDS resources. Automated backups are also regularly created to help recover the database from high-level faults such as unintentional data modification or bugs in the system.

Thirdly, Amazon Route 53 is configured with Route 53 failover to a static website hosted on Amazon S3 and served through Amazon CloudFront. In the event the VPC goes down, users can still be redirected to the static website hosted on S3. This is for higher availability of the application and to ensure that the website is still available.

Furthermore, the usage of fully managed services like AWS Lambda, API Gateway, Amazon RDs and more means that those services will automatically scale themselves to meet greater demand.

Lastly, as we are using DynamoDB with ElastiCache to store user cart and session information, there is no need for Sticky Sessions in our EC2 instances. This means that even if an EC2 instance goes down while processing a payment, the cart information is not lost and another EC2 instance can retrieve that user's cart information from DynamoDB and complete the payment process in place of the other EC2 instance.

High Scalability:

The EC2 instances are configured with an Elastic Load Balancer (ELB) and an auto-scaling group. This allows us to launch multiple EC2 instances in our cloud infrastructure and evenly distribute traffic amongst the instances. If instances in the main AZ have a sudden surge in network traffic, it can route the traffic to an instance in another AZ to reduce network load. Autoscaling is also in place to automatically scale EC2 capacity up or down based on a set of defined rules and network traffic. If needed, auto-scaling will enable the addition of new instances and when it is no longer needed, it will terminate all extra resources. As auto-scaling can work across multiple AZs within an AWS region, managing all EC2 instances across the 3 AZs is much simpler and efficient.

Amazon RDS Read Replicas are also used to take read queries off the Master database. This helps to decrease the load on the primary database. Furthermore, a Read Replica can be promoted to a standalone database instance if needed.

Cost-Effective:

Firstly, we have incorporated a serverless tier in our infrastructure of the proposed solution. This serverless tier handles all client requests that are related to the front-end service of our web application. Such services that handle the requests are the AWS Lambda service that runs our web application’s code and AWS CloudFront. This reduces the need for more server-based services such as EC2 instances. Thus, reducing the overall cost of our AWS infrastructure.

Secondly, auto-scaling and the elastic load balancer are used to automatically increase or decrease the EC2 resources. During high network traffic, the EC2 will be scaled up to meet high user demands. When the network traffic goes down, the EC2 will be scaled back down. This helps to prevent using excessive resources to meet a low demand. Thus, reducing the overall cost of our AWS infrastructure.

Thirdly, the AWS CloudWatch service has been configured for the EC2 instances and RDS instances. CloudWatch enables the administrators to have detailed information about the resources, which allow them to find places where they can reduce costs. This can help administrators to audit and evaluate the infrastructure which can help the system run more efficiently and productively. CloudWatch Alarms have been set up to watch the instance's metrics and send notifications to the main server if the metrics fall outside a set of defined levels.

4.6 Challenges of Proposed Solution

Firstly, when the system is operating during high network traffic and demands, the overall cost of running the whole AWS cloud infrastructure will surge and increase dramatically. This will cause the Auto Scaling group to provision more EC2 instances to handle the load, which in turn will cause our costs to go up as well. This can be mitigated by using a fully serverless architecture with AWS Lambda and API Gateway. Serverless architecture is fully managed and much more cost-effective than provisioning EC2 instances.

Secondly, the web application may be prone to new and unknown web exploits, bugs and cyber-attacks that will compromise the system’s security and infrastructure. The infrastructure must be resistant to the compromisation of the whole system. Hence, both regular automatic backups and manual backups have been enabled to prevent huge data losses. AWS CloudFront has also been enabled to prevent common cyber attacks such as Denial of Service Attacks (DoS) with the help of Content Delivery Networks (CDN) and Virtual Private Cloud (VPC)

4.7 Video Demonstration of Proposed Solution

The link to the video demonstration: https://youtu.be/TwEuDIpT820