Solution Description

Prepared by| Ayo Salawu.

Table of Contents

Table of Figures

1.0 Introduction

This document describes a troubleshooting procedure, a solution guideline detailing both short and long term alternatives for Mixed Systems Web application deployment.

1.1 Overview of Problem Statement

The customer; “Mixed Systems” is not familiar with Amazon Web Services and is encountering issues when trying to launch a web application as a proof of concept.

Mixed Systems has launched an Elastic Load Balancer, and Elastic Compute Cloud instance (acting as the web server). Both are deployed in a Virtual Private Cloud on the Amazon web services console.

While the customer's initial deployment aims to present a static web page to its users (demo.html located in the document root of the web server), the end solution should continue to be suitable for generating dynamic responses (the customer is currently developing the application). The customer is not sure about their future direction or requirements and are looking to you to provide expert guidance despite the ambiguity.

1.2 Customer Requirements

The customer requirements are detailed as follows:

a) Troubleshoot the implementation by doing the minimum amount of work required to make the web site operational with detailed written troubleshooting instructions or scripts for the in-house team.

b) Propose solutions to improve the availability, security, reliability, cost and performance before the project goes into production. This should include business and technical benefits, including design or architecture document and diagrams.

c) Provide high-level alternative solution for the longer term as the web application becomes more successful.

2.0 Troubleshooting Guide

2.1 Initial (Existing) Architecture

As part of the initial analysis, the Cloud formation stack was launched using the provided yaml template. The launch was successful and gave the DNS name of the ELB as output. Below is a screenshot of the existing architecture observed.

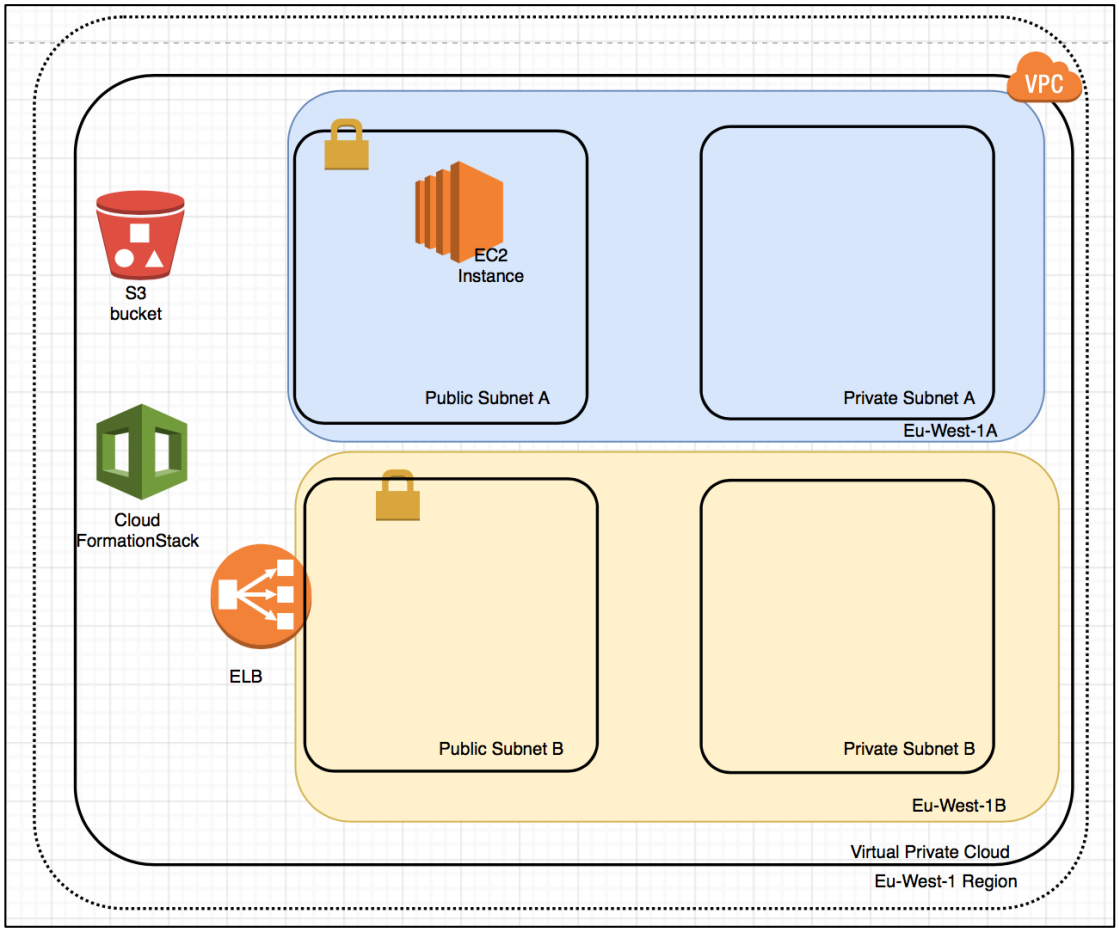

Figure 1 Initial Architecture

2.2 Issues Identified / Solution Steps

Issues uncovered while analyzing the entire architecture include:

I. The ELB and the EC2 instance were deployed into different availability zones and subnets please see screenshots below

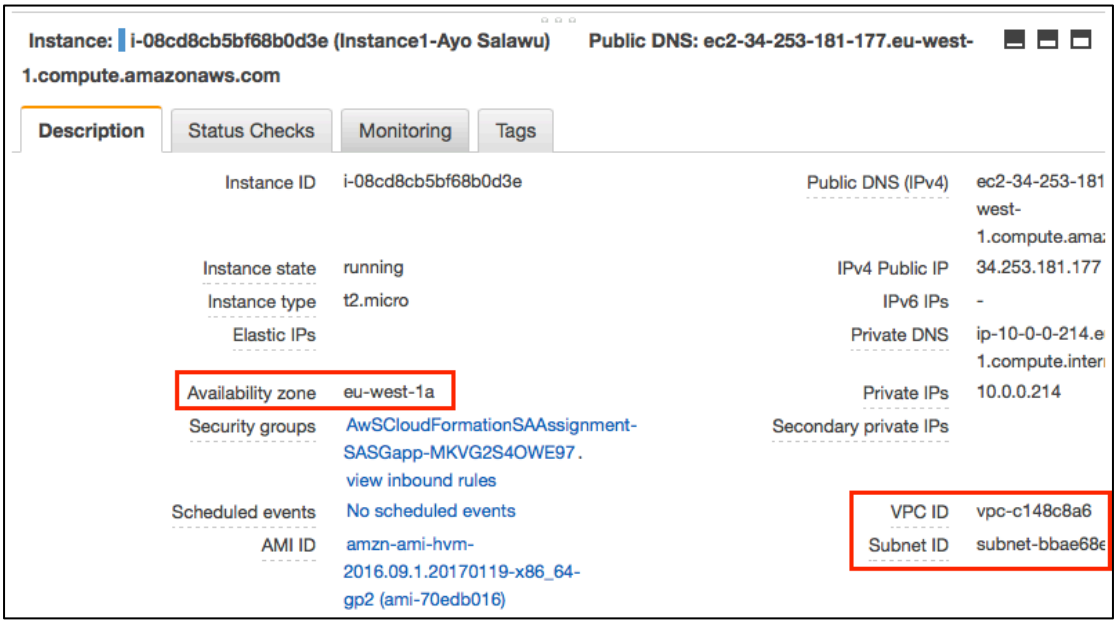

Figure 2 EC2 Instance AZ and Subnet Configuration

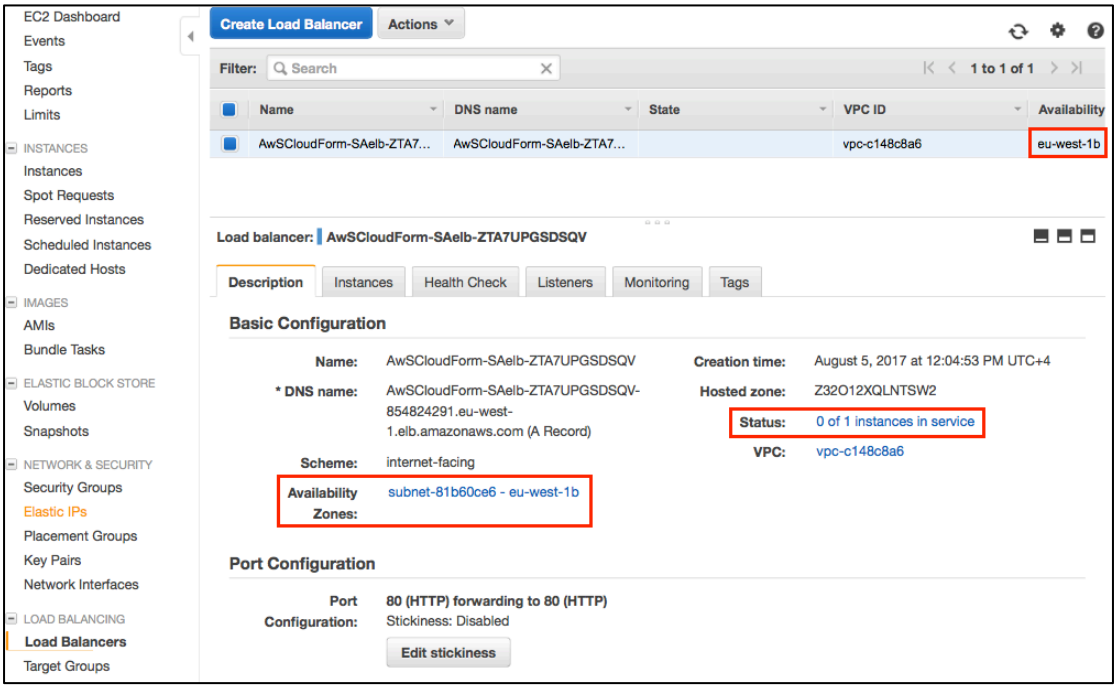

Figure 3 Load Balancer AZ and Subnet Configuration

The best practice for designing load balancers requires all subnets containing resources to be managed by a load balancer should be added to the load balancer configuration. It is stated in the load balancer guide as well that using additional availability zones and subnets expand the availability of your application (See section on Add an Availability Zone and Add a Subnet; Pages 26 & 28 of classic load balancer guide).

In our scenario, the ec2 instance is in the eu-west-1a Availability Zone and subnet as the instance is in a public subnet and availability zone different from those of the load balancer we can add the availability zone as follows.

Solution I: Add an Availability Zone

i. Open the EC2 console, on the navigation pane, under LOAD BALANCING, choose Load Balancers

ii. Select the load balancer named *AwSCloudForm-SAelb-ZTA7UPGSDSQV*

iii. On the instances tab, choose Edit Availability Zones.

iv. On the Add and Remove Subnets page, select the eu-west-1a availability zone which contains the subnet named *PublicSubnetA-Ayo Salawu* (this subnet contains the ec2 instance) and Choose Save. See Screenshot below showing the completed configuration.

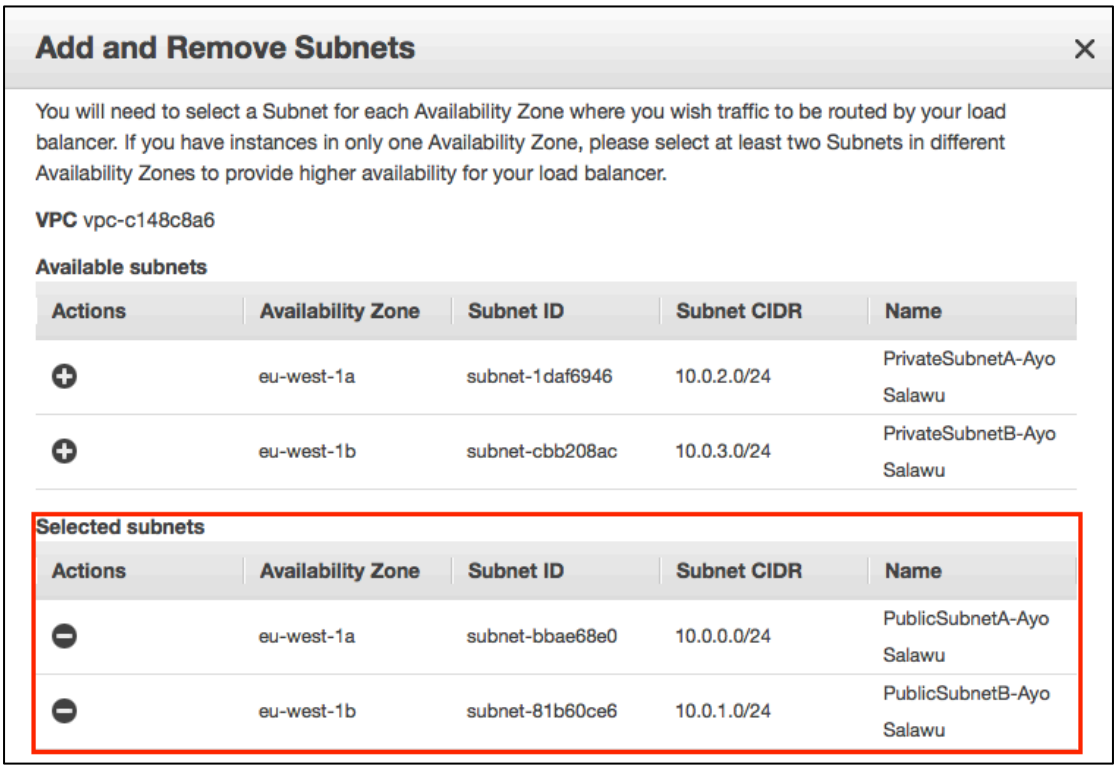

Figure 4 Subnet Configurations

II. The Load balancer health-check was failing.

From the below screen shot, we see the instance status as OutOfService. This was the case because though the instance was actually up and running and in good working condition, the health-check of the load balancer was not configured properly and was unable to reach the instance in its subnet.

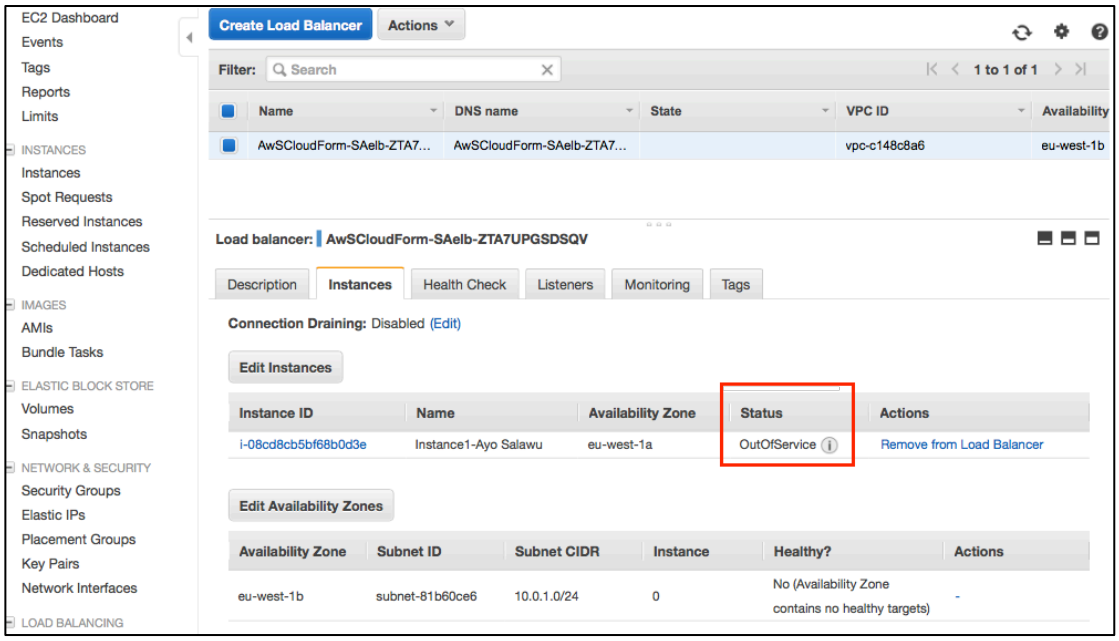

Figure 5 LB Health Check Status

It is stated on page 16 of the Classic Load Balancer User Guide (see references) a load balancer periodically sends pings, attempts connections, or sends requests to test the EC2 instances. These tests are called *health checks*. The status of the instances that are healthy at the time of the health check is InService. The status of any instances that are unhealthy at the time of the health check is OutOfService

Further observation revealed that the Health check configuration of the ELB was not consistent with the security group configurations for both the ELB (named *ELBSecurityGroup*) and the EC2 instance (*AppServerSecurityGroup*). These requires changes to the configuration.

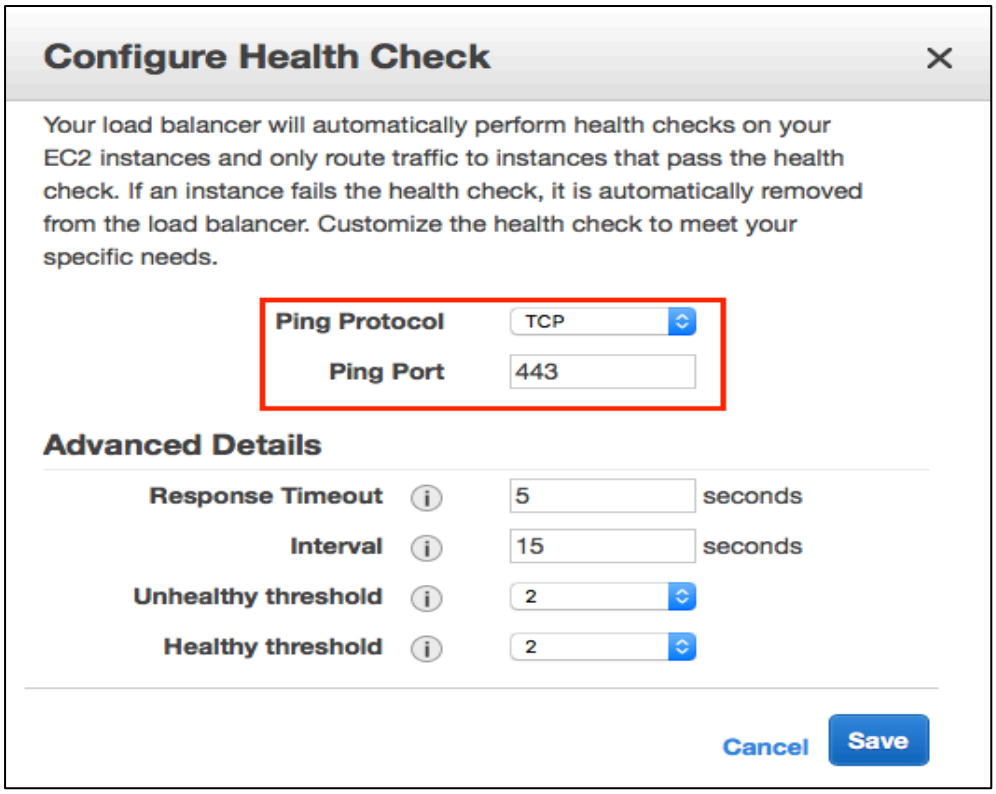

Figure 6 LB Health Check Configurations

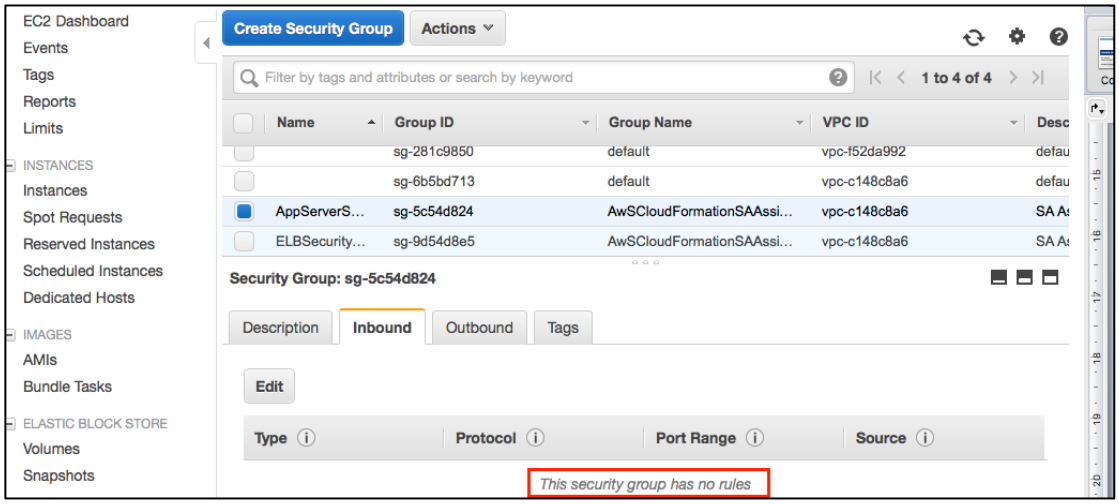

Figure 7 EC2 Security Group Configurations with NO RULES

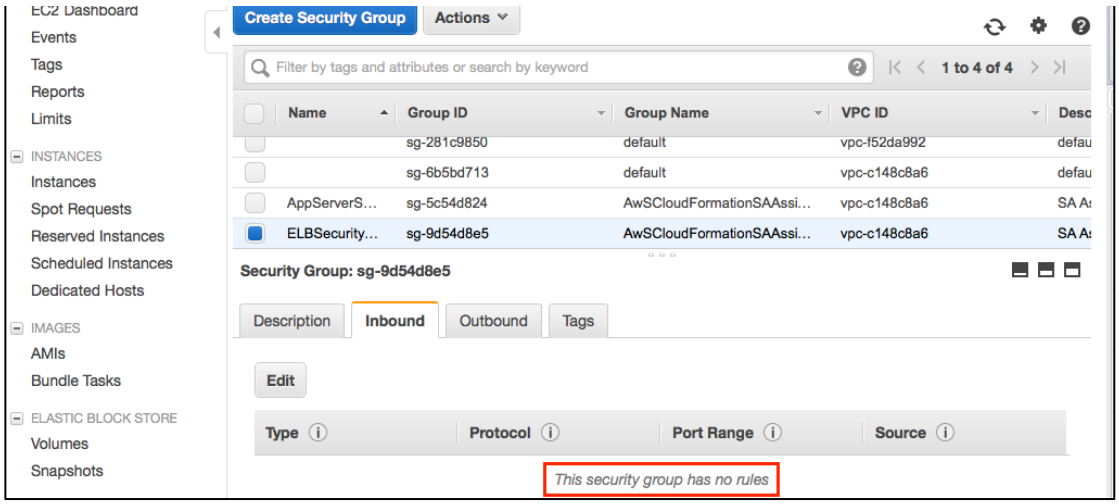

Figure 8 LB Security Group Configurations with NO RULES

Step 1: Configure Security Group Rules for EC2 Instance

i. Still within the EC2 Service, Look under NETWORK &SECURITY in the Navigation pane and select “Security Groups”.

ii. Select the security group for the EC2 instance named *AppServerSecurityGroup*

iii. Select the inbound tab underneath, click on edit, click on Add Rule and create the rules by making the below selections in the drop down menus.
– Under Type select “HTTP”, (Protocol and Port Range will be populated automatically with TCP and 80) under source select “Custom” and type in the name of the security group for the Load Balancer (sg-9d54d8e5)
- Add another rule for “HTTPS” similarly and click on save.

This will ensure HTTP(S) traffic to the EC2 instance on ports 80 and 443 will only come from the Load balancer

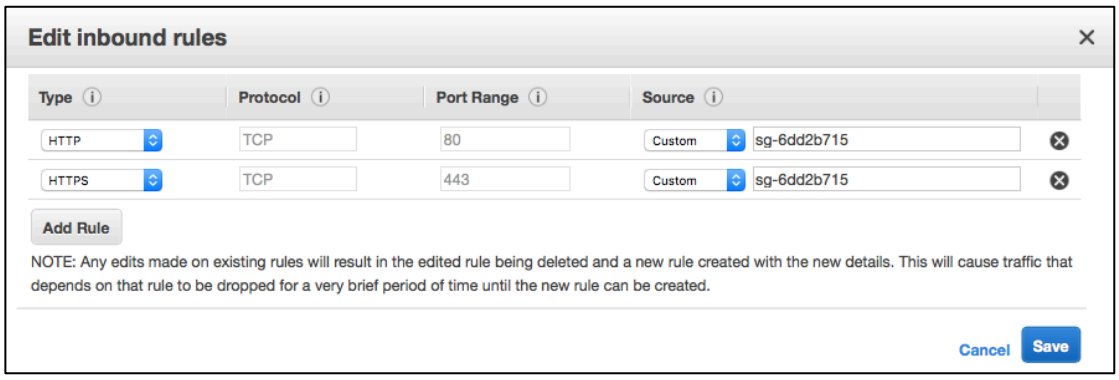

Figure 9 EC2 Instance Security Group Rules

1. Still within the Security Groups view, select the security group for the load balancer named *ELBSecurityGroup*

2. Select the inbound tab underneath, click on edit, click on Add Rule and create the rules by making the below selections in the drop down menus.

– Under Type select “HTTP”, (Protocol and Port Range will be populated automatically with TCP and 80) under source select “Custom” and type in 0.0.0.0/0 this will make the load balancer accessible over the Internet

- Add another rule for “HTTPS” similarly and click on save.

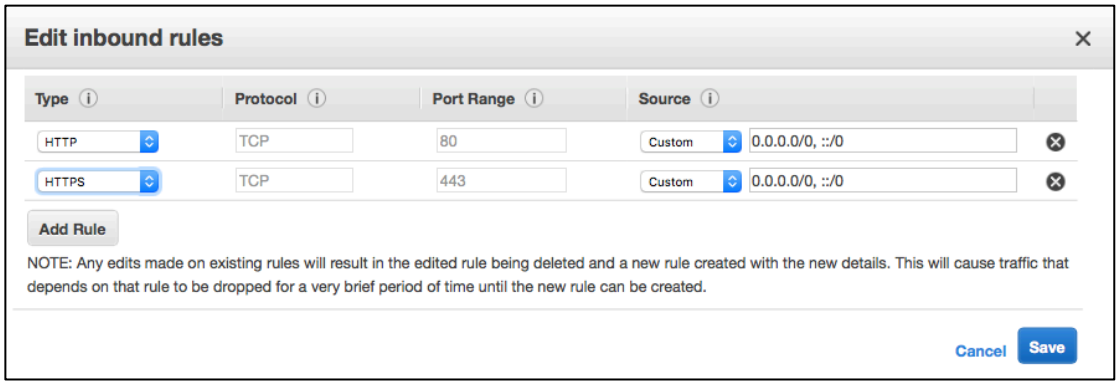

Figure 10 ELB Security Group Rules

Step III: Re-Configure Health Check

To re-configure the Health Check to align with the security group configurations, perform the following steps

From the Classic Load balancer guide, refer to the Tutorial: Create a Classic Load Balancer (Step 4: page 5) as follows.

a. From the AWS console, click on the EC2 service and select Load Balancers from the navigation pane on the left
b. Click on the Health-Check Tab and click the “edit health check”
c. Make the following changes to the configuration and click save.
   - Change Ping Port from 443 to 80
   - Change Protocol from TCP to HTTP
   - Define Ping path as /demo.html

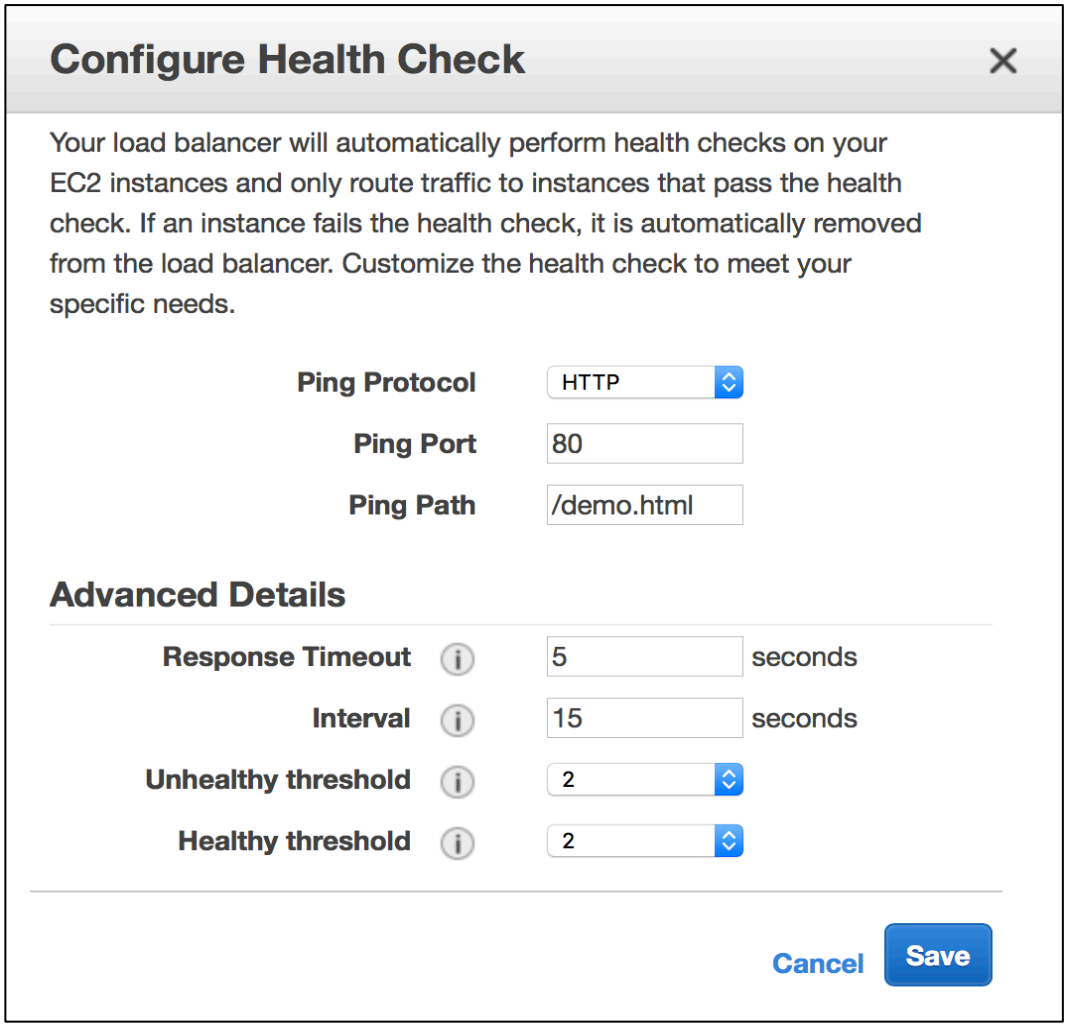

Figure 11 Health Check Re-configured

2.3 Launch The Website

After following all the steps above, re-attempt to launch the website. This time we will do this via the DNS name of the elastic load balancer as this is the requirement.
This can be achieved via one of 2 ways.

- From Cloud Formation
- From EC2

From Cloud Formation, go to the Output tab and copy the DNS name of the elastic load balancer.
Paste the name (as highlighted in the red box below) into a web browser and press enter.
The following should be the output of the url

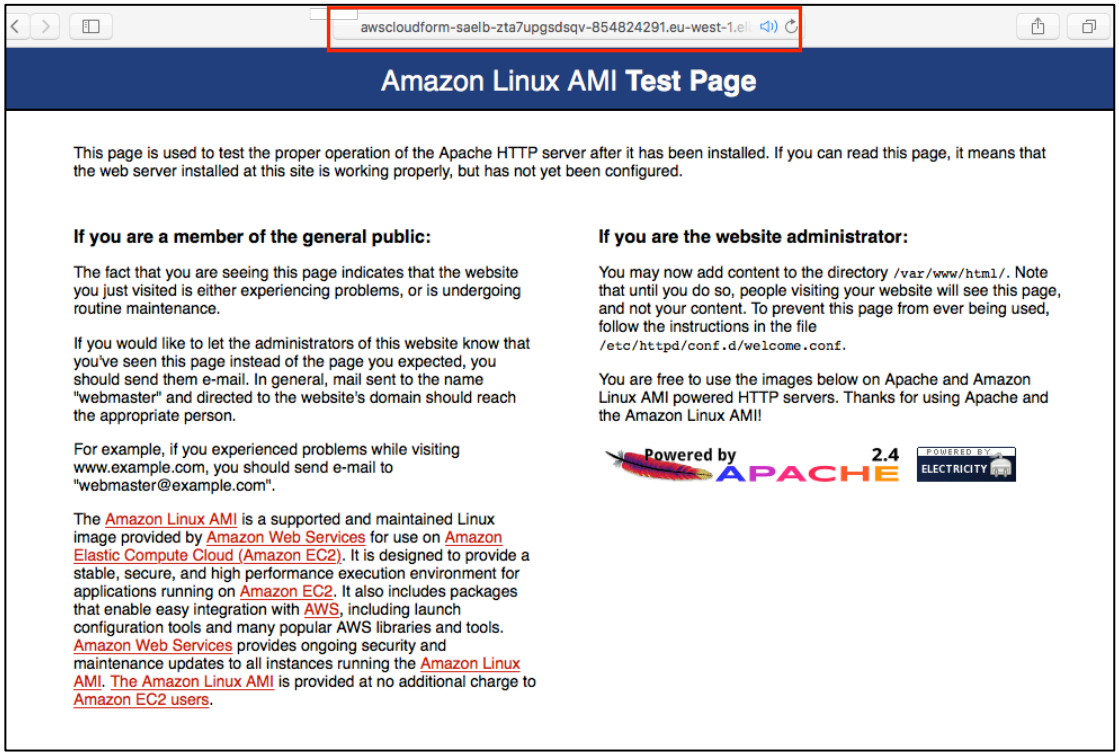

Figure 12 Screenshot of website launched via ELB

1. From the EC2 console, Go to LOAD BALANCING on the navigation pane and select Load Balancers.
2. Click on the Description tab and copy the DNS name, which is *AwSCloudForm-SAelb-ZTA7UPGSDSQV-854824291.eu-west-1.elb.amazonaws.com* (without the A record text in brackets)
3. Paste this into a web browser and the output should be the same as above.

3.0 Short Term Solution Proposal

Now the website is accessible we can move on to the Short-term changes that can be implemented to the entire architecture to improve availability, security, reliability cost and performance before production.

As the customer's initial deployment aims to present a static web page to users, the short term change will be to host the static website on S3, Amazon's simple storage service and also configure a domain name in Amazon Route 53 to point the actual http domain name for the website to the S3 bucket where the website content or assets are hosted. Lastly, cloudfront can be used to make the website content available globally. The short-term solution is represented diagrammatically below:

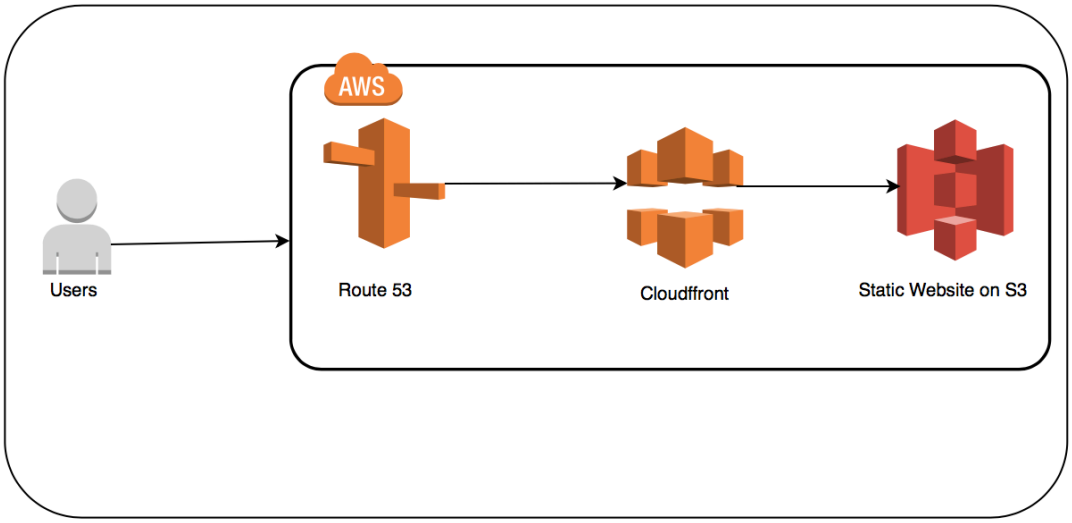

Figure 13 Short Term Change Architecture

This short-term change will make the website highly performing and scalable at a fraction of the cost of a web server.

This proposal aligns with the best practice for hosting static websites as stated in the AWS whitepaper: Hosting Static Websites on AWS (pages 5 and 6). It also makes the website ready to meet the real-world demands of a high-traffic website.

This short-term change is simple and quick to implement and will provide the required improvements to the architecture. The demo.html file will be placed in a bucket in S3 and also configured as the index file that will serve as the landing page that visitors to the web site will view. Also an error file (error.html) needs to be setup as well. Once these two files are in place, we can set permissions and made globally available. Also the static website hosting feature on AWS S3 is to be activated. This is done on the S3 console as shown in the screen shot below.

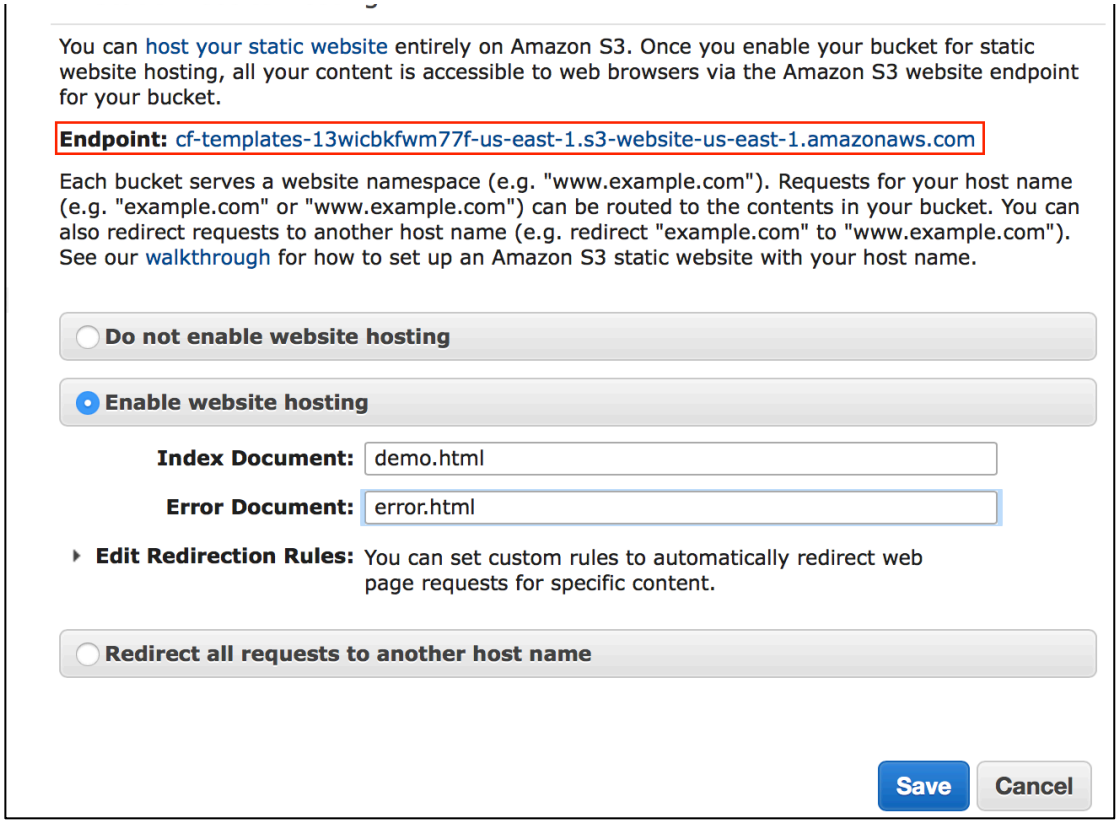

Figure 14 Enable Website Hosting

The field in the red box shows the endpoint url that will be mapped to the customers required domain name in Route 53 for reachability over the internet.
Also to cater for some level of dynamic content in the short term, the versioning feature can also be activated as well to have different versions of the demo.html web page.

Implementing these short-term changes will provide improvements in the following areas:

Availability, Reliability and Performance

Static websites hosted on S3 are highly available, this is because S3 is built with 99.99% availability as well as 999999999.99% of durability. S3 delivers these capabilities as objects stored on S3 are automatically replicated across multiple data centers globally. In the event of an outage of one AWS data center, the static website will still be running and available for customers to access. On S3, the website will also scale as well, this is due to S3's ability to scale its throughput seamlessly to serve thousands of HTTP requests per second without making any changes to the architecture.
To also improve performance S3 supports multi-part upload, S3 transfer acceleration as well as the ability to increase the GET requests per second made to the website. All these features all help to improve the performance of S3 to host websites. We also improve the reliability of the website as cloudfront provides us an easy way to cache and distribute content of the website to end users with low latency and high data transfer speeds by leveraging of cloudfront's global edge location distribution network.

Cost

Hosting a static website on S3, is very cost effective as you are only charged for actual S3 storage required to store the assets of the website. Essentially, you will be

charged for assets such as JavaScript files, html files, videos, audio files and any other downloadable files. Bandwidth charges will depend on the actual site traffic. From the aws whitepaper on website hosting it is stated that small sites with few visitors have minimal hosting costs. This matches our customers initial deployment of a static website for now while the application is being developed.

The monthly costs can be estimated by using the AWS simple monthly calculator and you can also track your monthly spend as well with the AWS cost explorer

Security

Security is of utmost priority at Amazon Web Services and the best practice for ensuring security for hosting websites on S3 involve encryption. While access to data in S3 is restricted by default, there are several ways to secure content on S3. S3 offers server side encryption for data at rest and in -transit, more specifically for a static website deployment, we can implement the entire HTTPS encryption and decryption process as well. S3 also allows for secure upload and download of data via SSL encrypted endpoints. These endpoints are accessible from the Internet.

3.1 Technical and Business Benefits

Some of the technical benefits of the short-term changes proposed include:

- Highly secure and available website
- Automated Scalability and reliability built into the system
- Use of managed services let you focus on your core business as AWS handles underlying management of the infrastructure for you.

Some business benefits of the short-term solution include:

- When using AWS services the Total Cost of Ownership for infrastructure significantly reduces and translates into huge cost savings that allow the customer to invest budgets on other areas of the business
- Ability to develop and focus on your core business without resourcing and building an entire organization to manage your infrastructure
- Increases your speed and agility to roll out new business offerings with a globally available infrastructure allowing your business to reach a global market
- Next to no upfront investment for your infrastructure, you basically pay for ONLY what you use allowing you to manage your infrastructure costs as well.

4.0 Long Term Solution Proposal

As the web application becomes successful a long-term solution architecture would be required to cater for the expected growth in user traffic to the website and web application.
This would be achieved by a combination of services to cater for static content as well as dynamic content. The entire architecture as shown below will be detailed in the following points

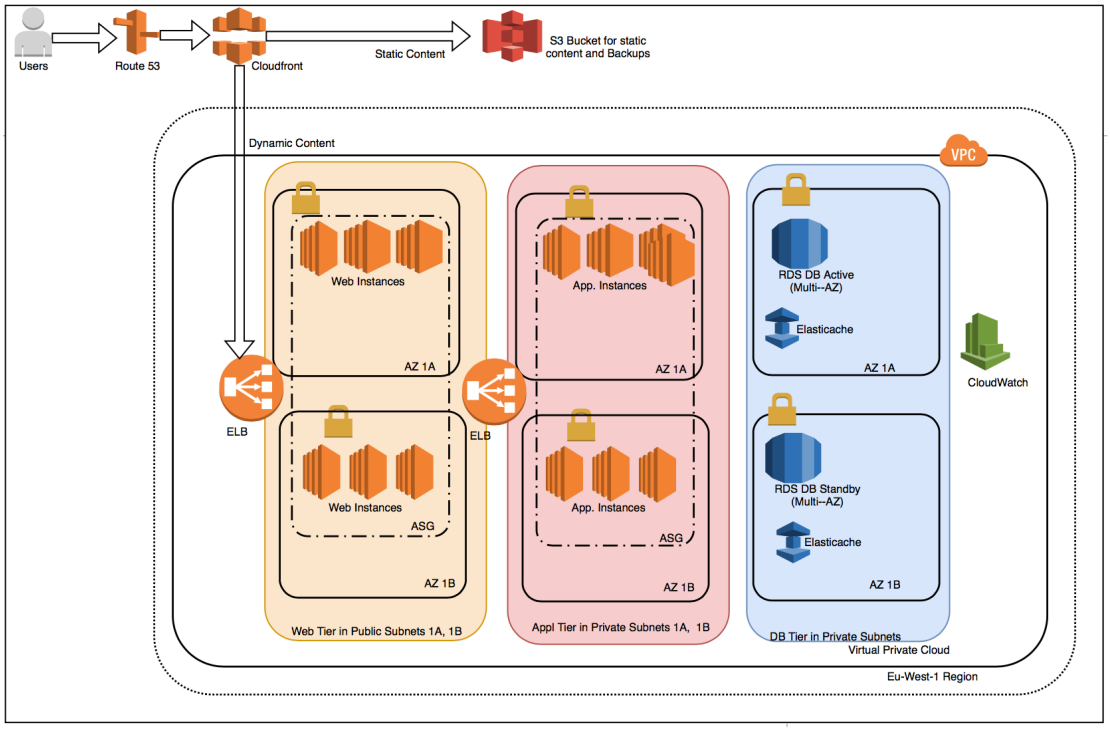

Figure 15 Long Term Architecture

1. As deployed in the short term changes we will maintain the Route 53, cloudfront and S3 services for the static content
2. HTTP User request for dynamic content would also be delivered by cloudfront as well but would be routed to an elastic load balancer (we will refer to this elb as the internet facing or external elb based on its location in the architecture)
3. The internet facing ELB will automatically distribute incoming web traffic to the main architecture spread across availability zones. This spread will enable greater fault tolerance for the architecture. The load balancer will also send frequent health checks to the instances and will stop sending requests to an instance if it becomes unavailable or unhealthy.
4. The main infrastructure for the dynamic content will be spread across three tiers for the Web Servers, Application Servers and the Database. This tiering will allow for a loosely coupled architecture allowing each tier to be scalable independently of each other as well as be upgraded independently.
5. The web server tier would be made of EC2 on-demand instances that will handle the web requests. This tier would be spread across 2 availability zones behind the elastic load balancer to provide high availability for the web traffic.
6. The application server tier would also be made of EC2 on-demand instances to handle the web application. This tier would also be spread across 2

availability for the application traffic.

7. The third tier will be the database tier made of RDS database instances. This tier will be placed in private subnets for security of user data. It will be deployed using the Multi-AZ feature of RDS for high availability at this tier having the master database in one zone and the standby in the next zone with synchronous replication between both databases
8. To achieve scalability, the web and application servers would be deployed in auto scaling groups to automatically adjust capacity up and down according to conditions defined in the scaling policy, with this, we ensure the number of EC2 instances increase seamlessly during demand spikes and decrease instance capacity during reduced capacity periods.

9. From a security perspective, the internet-facing ELB and the three tiers are to be placed inside a VPC (virtual private cloud). Each tier would have its security groups and network access control lists configured for added security. The security groups act as virtual firewalls to control VPC traffic to and from your instances at each tier.
10. As the web servers are internet facing and receiving requests from the internet, the web server security groups would be configured to give access to hosts over TCP on ports 80 and 443 only (HTTP and HTTPS) and instances in the application server security group on port 22 (SSH) for direct host management. The security group of the application server cluster would be configured to allow access from the Web Server security group for handling web requests and from customer corporate network over TCP on port 22 (SSH) for direct host management. This would allow customer engineers to log in directly to the application servers from the corporate network and then access other tiers from the application servers.
11. Also within the database tier, elasticache is introduced to provide caching services for the application tier and also remove load from the database tier
12. It is worth noting the S3 static website would also serve as a backup to the main site should there be an unexpected outage on the main site to ensure users are able to still access content and unaware of a major outage on the site. Also backups for the RDS database would be stored on S3
13. As the EC2 instances for the web server tier can have EBS (elastic block store) volumes attached to them to store web application data, this data would also be backed up to S3 as well
14. Cloudwatch is also included to create alarms or custom metrics to trigger scaling actions when certain thresholds are crossed
15. IAM users to be setup to allow authorized personnel manage and administer the AWS infrastructure as required. (This would include setting up roles and policies as required).
16. The autoscaling groups for both web and application tiers would span across the two availability zones for consistent and uniform scaling of the infrastructure but within the vpc.

The architecture adheres to best practice for website hosting as stated in the Web Application Hosting in the AWS Cloud: Best Practices reference as well as the AWS reference architecture for Web Application Hosting.

Another high-level alternative for the long term will be to deploy the web application on elastic beanstalk. If the web application is developed with either Java, .NET, PHP, Node.js, Python, Ruby or Go.

Once you upload your web application code, Elastic Beanstalk will automatically handle the deployment, capacity provisioning, load balancing, auto-scaling and application health monitoring of the infrastructure required to power your web application on AWS.

It is recommended to take a few actions frequently such as activating logging on various services to track and monitor activities on the services and pre-empt certain behaviors.

Cost optimization is an iterative process, and as the application and its performance evolve over time, it is good practice to constantly review the entire architecture to see where further cost reductions and improvements can be made. Improvements such as upgrading or vertically scaling the EC2 instances from t2 micro size to M4 large or large instance size (the M4 instances are General Purpose instances which provide a balance of compute, memory, and network resources, and is a good choice for many applications). Setting up billing alarms can also let you know when you exceed certain defined budgetary limits to alert you about costs incurred. Setting up SNS notifications as well to send notifications to pre-defined users as well helps to notify you of events as at when they occur.

In summary, the solution above provides zero upfront investments, efficient resource utilization and usage based costing and very fast time to market. Please contact me anytime, should you have any further questions. On behalf of Amazon Web Services I wish you all the very best with this and future cloud deployments.

References

Elastic Load Balancing: Classic Load Balancers
http://docs.aws.amazon.com/elasticloadbalancing/latest/classic/elb-classic.pdf#elb-healthchecks

Hosting Static Websites on AWS: Prescriptive Guidance
https://d0.awsstatic.com/whitepapers/Storage/Building%20Static%20Websites%20on%20AWS.pdf

AWS Storage Services Overview
https://d0.awsstatic.com/whitepapers/Storage/AWS%20Storage%20Services%20Whitepaper-v9.pdf

Architecting for the Cloud: Best Practices
https://d0.awsstatic.com/whitepapers/AWS_Cloud_Best_Practices.pdf

Web Application Hosting in the AWS Cloud
https://d0.awsstatic.com/whitepapers/aws-web-hosting-best-practices.pdf

AWS Reference Architecture for Web Application Hosting
http://media.amazonwebservices.com/architecturecenter/AWS_ac_ra_web_01.pdf

Draw.io Tool for Architectures