2020

Abrar Al-Sagheer, S3707180

Labiba Islam, S3694372

5/24/2020



Cloud Assignment 2 Report, Crypto Watch

Contents

[1. Links 2](#_Toc41249399)

[2. Summary 2](#_Toc41249400)

[3. Introduction 2](#_Toc41249401)

[4. Related work 3](#_Toc41249402)

[5. Architecture and Software Design 3](#_Toc41249403)

[5.1. General Software Design architecture 4](#_Toc41249404)

[5.2. Crypto Watch Web Application 5](#_Toc41249405)

[5.2.1. Amazon Elastic Beanstalk and Elastic Load Balancer 5](#_Toc41249406)

[5.2.2. Amazon DynamoDB 6](#_Toc41249407)

[5.2.3. Amazon Simple Storage Service (S3) 6](#_Toc41249408)

[5.2.4. Amazon Lambda Service 7](#_Toc41249409)

[5.2.5. Amazon Translate Service 7](#_Toc41249410)

[5.3. Crypto Watch Lambda function handler 7](#_Toc41249411)

[5.3.1. Amazon Simple Service 8](#_Toc41249412)

[5.4. AWS Supporting services 8](#_Toc41249413)

[5.4.1. AWS Identity and Access Management (IAM) 8](#_Toc41249414)

[5.4.2. Amazon CloudWatch 8](#_Toc41249415)

[5.4.3. AWS CodePipeline and CodeBuild 9](#_Toc41249416)

[6. Implementation 10](#_Toc41249417)

[6.1. Create the Elastic Beanstalk Environment 10](#_Toc41249418)

[6.2. Create the Lambda Function 11](#_Toc41249419)

[6.3. Create S3 Bucket 12](#_Toc41249420)

[6.4. Create DynamoDb Contact Table 12](#_Toc41249421)

[6.5. Create CodePipeline and CodeBuild 13](#_Toc41249422)

[6.6. Update permissions in IAM 14](#_Toc41249423)

[7. User Manual 16](#_Toc41249424)

[7.1. Crypto Watch Main page 16](#_Toc41249425)

[7.2. Crypto Watch Admin page 18](#_Toc41249426)

[7.3. Crypto Watch Debug page 18](#_Toc41249427)

[8. References 20](#_Toc41249428)

[9. Signed Contribution Agreement 21](#_Toc41249429)

# Links

Application Link: <http://cryptowatch-env.eba-76gwvtzy.us-east-1.elasticbeanstalk.com/>

GitHub Code repository link: <https://github.com/s3707180/cloudassignment2.git>

# Summary

Crypto Watch project provides on-demand information about the state of Crypto Currency market. It provides historical information of cryptocurrency performance and visuals it using web technologies.

The project is deployed into Amazon Web Services (AWS) and make efficient use of its services. In addition of web visualisation, user customised messages are delivered on-demand. The project uses best practices in building and deploying cloud native applications. It provides a simple and more accessible way to consume the information in a glance compared to existing portals like CoinMarketCap.

# Introduction

Cryptocurrencies are digital currencies that are secured by cryptography. The makes them nearly impossible to counterfeit or double-spend. Unlike Fiat currencies which are legal tender whose value is backed by the government that issued it, existing Cryptocurrencies are not backed by central governments or authorities. Many cryptocurrencies are decentralized networks based on blockchain technology with no single authority controlling or manipulating them. There are many cryptocurrencies created with varying degree of success and adoption in the market. These cryptocurrencies are typically traded on online exchange systems and can be exchanged for Fiat currencies as well as traded with each other.

There are a number of applications and websites that provide details about cryptocurrency exchange rates but they are too complex for easy consumption with details for 1000s of cryptocurrencies with limited comparison and visualisation.

The Crypto Watch system is created to inform users about the latest price changes in cryptocurrencies while using the scalability of Amazon Web Services (AWS). The system combines a number of AWS services to keep the users up to date with best and worst performing Crypto currencies in the market using Binance Cryptocurrency exchange as the source for information. It provides a simple user interface to summarise the details in a friendly and easy to consume manner. Crypto Watch utilises a number of AWS services to add value like sending personalised messages to the users with their chosen language as well as store retrieved data in scalable AWS storage.

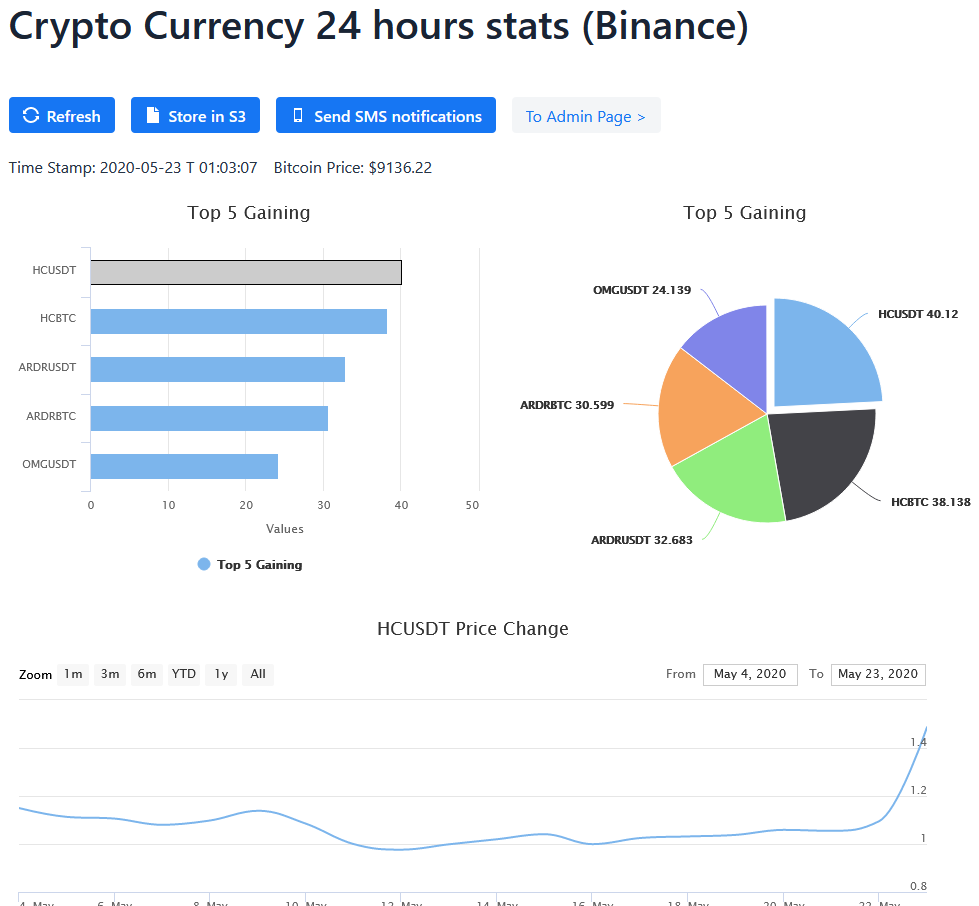


Figure : Screenshot from Crypto Watch home page

Crypto Watch can be further enhanced by adding user access control and having Realtime monitoring to notify users using a predefined or user customised conditions.

# Related work

` CoinMarketCap is accessible using the link below.

<https://coinmarketcap.com/>

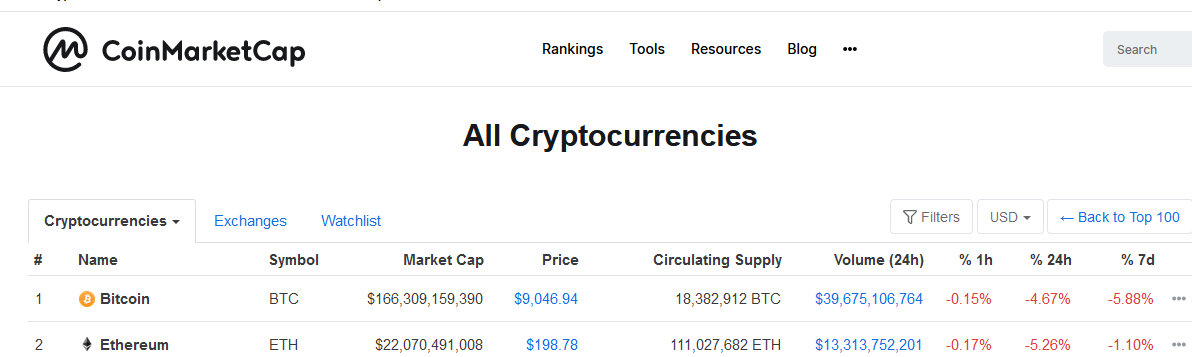
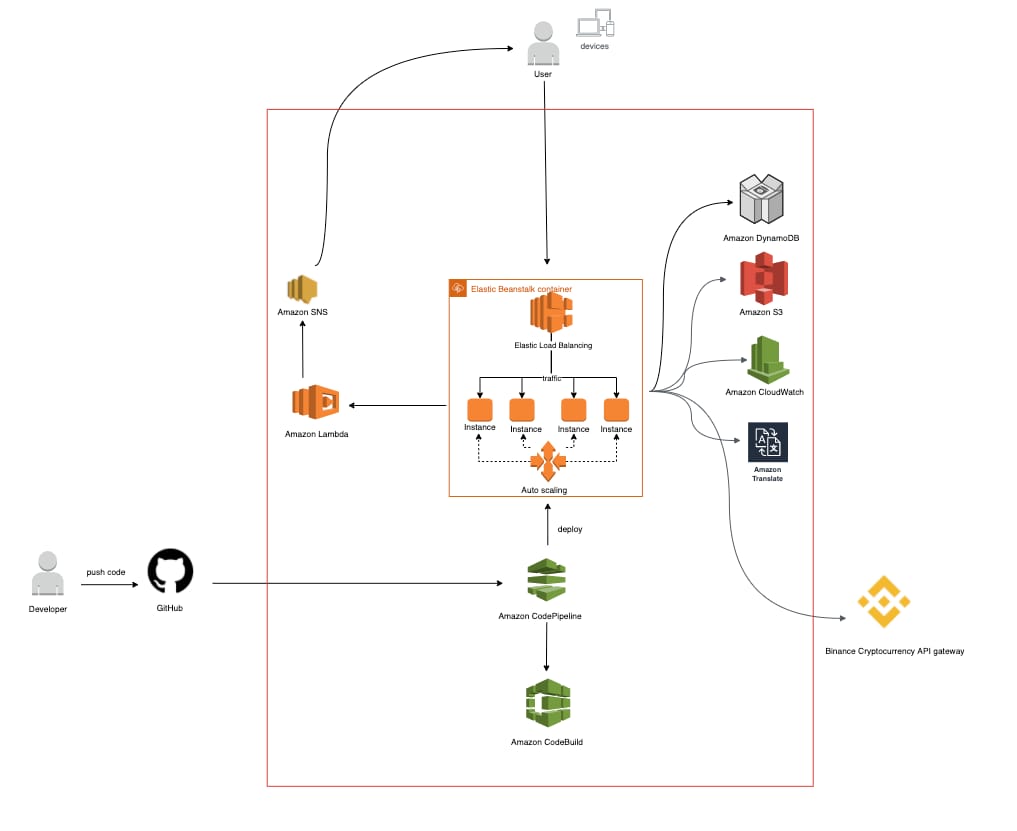


Figure : CoinMarketCap front page

# Architecture and Software Design

This section will describe Crypto Watch architecture as well as related technologies.

Crypto Watch uses a number of AWS services to serve the users as well as to cater for build and deployment of the application. The application is divided into 2 parts. Web application which is deployed to AWS Elastic Beanstalk service and a serverless component which is deployed to AWS Lambda server. See the diagram below for the overview: 

## General Software Design architecture

Crypto Watch application is written using Java programming language. We used Spring Boot (2.3) to create a stand-alone web application that can be deployed to AWS Elastic Beanstalk service. We used an embedded Apache Tomcat as the web server of the application. This made it easier to make a portable jar file from the application for simplified deployment. We could have used a different setup to have Tomcat setup and configured separately but we decided against this as the embedded version is more portable and consistent. This way we can run it on our local development environment and on AWS cloud the same way.

The application uses Vaadin java web framework for the presentation of the application. Vaadin makes it easy to build web applications all in Java without the need to write HTML or JavaScript. The framework works in a similar fashion to Java Swing in that the Graphical User Interface (GUI) components are placed using layouts and actions can be associated with the components all in Java. We have used Vaadin Charts for visualisation of graphs. Vaadin Charts are part of Vaadin Pro and need to be licenced beyond a trial period.

The application utilises Binance Cryptocurrency exchange data and uses the open source Binance Java API to consume its services.

The code for both Web application and Lambda Function is stored in GitHub. GitHub is linked to AWS CodePipeline Service so when new code is pushed, the application is built and deployed.

All application dependencies are defined in a pom.xml file which is managed by Apache Maven. This simplifies using various libraries as Maven can pull all the dependent libraries instead of manually downloading them and including them in the application.

We separated every Service into its own class. This mean Every Amazon Service (e.g. S3, DynamoDb) had its own class where it encapsulated related code. Similarly, every Web page was given its own class “view” class. The three views are Main, Administrator and debug.

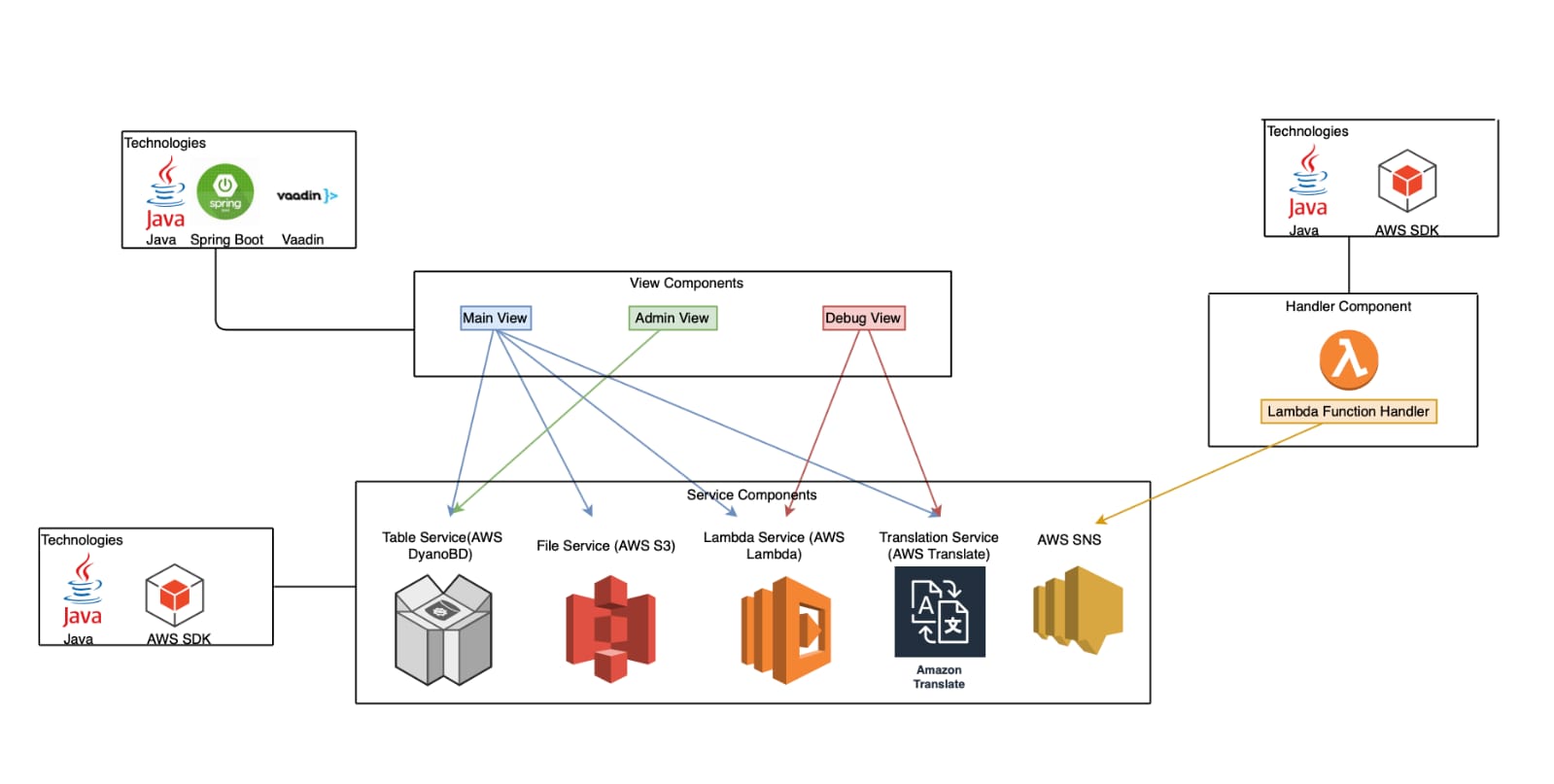


Figure : Software Components

## Crypto Watch Web Application

The web application is written using Java programming language and is deployed to AWS Elastic Beanstalk service. The application has 3 pages and views:

* **MainView** which is intended for end users and is accessible by going to the root application:  
  <http://cryptowatch-env.eba-76gwvtzy.us-east-1.elasticbeanstalk.com/>
* **AdminView** which is intended for Administrators and apart from the normal web flow, is directly accessible from the link below:  
  <http://cryptowatch-env.eba-76gwvtzy.us-east-1.elasticbeanstalk.com/admin>
* **DebugView** which is intended for development and troubleshooting and apart from the normal web flow, is directly accessible from the link below:  
  <http://cryptowatch-env.eba-76gwvtzy.us-east-1.elasticbeanstalk.com/debug>

The following services and technologies are used in the web application:

### Amazon Elastic Beanstalk and Elastic Load Balancer

The beanstalk is configured to be scalable with utilising an **Elastic Load Balancer** which is backed by 1-4 instances. The load balancer monitors the health and network load and if the application utilisation increases, deploys additional instances to serve user requests. As the utilisation decreases, the instances are terminated to decrease associated cost. This provides a great balance between serving large volumes while not paying for extra resources when not needed. We could have used AWS Elastic Compute Cloud (EC2) but Beanstalk was chosen as it simplified management of Web application and scalability. Using Beanstalk there was no need to provision and manage virtual machines and we were able to focus on application functionality.

### Amazon DynamoDB

Amazon DynamoDB is a key-value and document database that is used in Crypto Watch to store user related data records like email, phone number and language. The phone number and the language are used to distribute on-demand notifications regarding cryptocurrencies performance. The application is benefited from distributed nature of DynamoDb and the ease of development for a schema free database compared to traditional Relational Database Management Systems. Amazon Java SDK was used to create a client to interact with DynamoDB.

|  |  |  |
| --- | --- | --- |
| Table name | Primary Key | Columns |
| Contact | Email  E.g. s3@st.rmit.au | Email: String  SMS: String  Language: String |

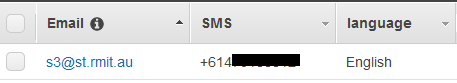


Figure : DynamoDB Contact Table

### Amazon Simple Storage Service (S3)

Amazon Simple Storage Service (Amazon S3) is object storage which provide a similar structure as a file system for cloud applications. Crypto Watch uses the service to store the cryptocurrencies exchange information a predefined bucket. The Object (File) is named Crypto. See the figure below. Amazon Java SDK was used to create a client to interact with S3.

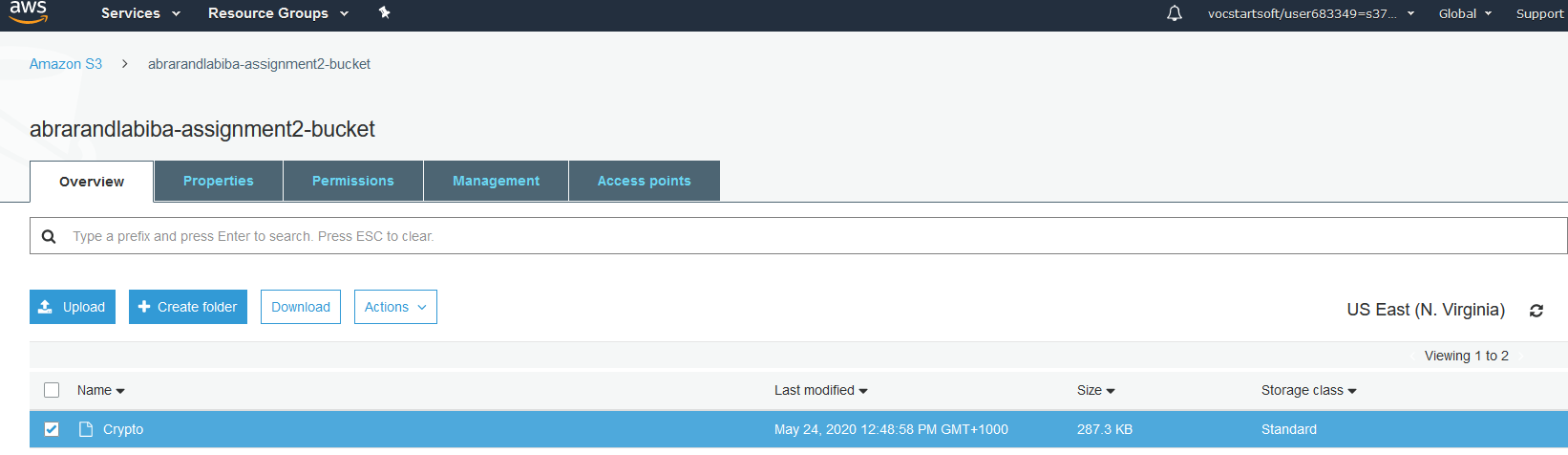


Figure : AWS S3 Bucket for Crypto Watch

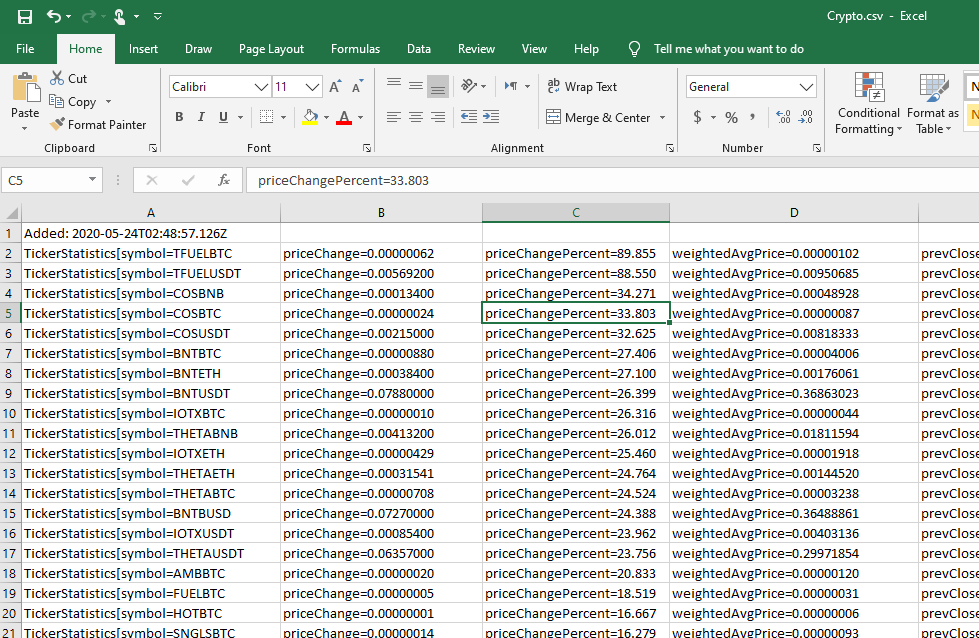


Figure : Sample content of Crypto Object (File)

### Amazon Lambda Service

AWS Lambda is an event-driven, serverless computing platform provided by Amazon. Crypto Watch web application uses the AWS lambda client to trigger the lambda function that is responsible for sending SMS messages to the users.

### Amazon Translate Service

Amazon Translate provides translation service which can simply receive a text and return its translation. The application uses Amazon Java SDK to create an AWS Translate client. It passes the text with expected language depending on the user’s preferred language to receive the appropriate text before sending SMS messages.

## Crypto Watch Lambda function handler

The application has a serverless component that is responsible for sending SMS messages to the users. It receives a list of recipients with the messages to deliver. Lambda functions are not active and do not consume resources until they are invoked. This makes the Lambda service great for processing scenarios like the one used here.

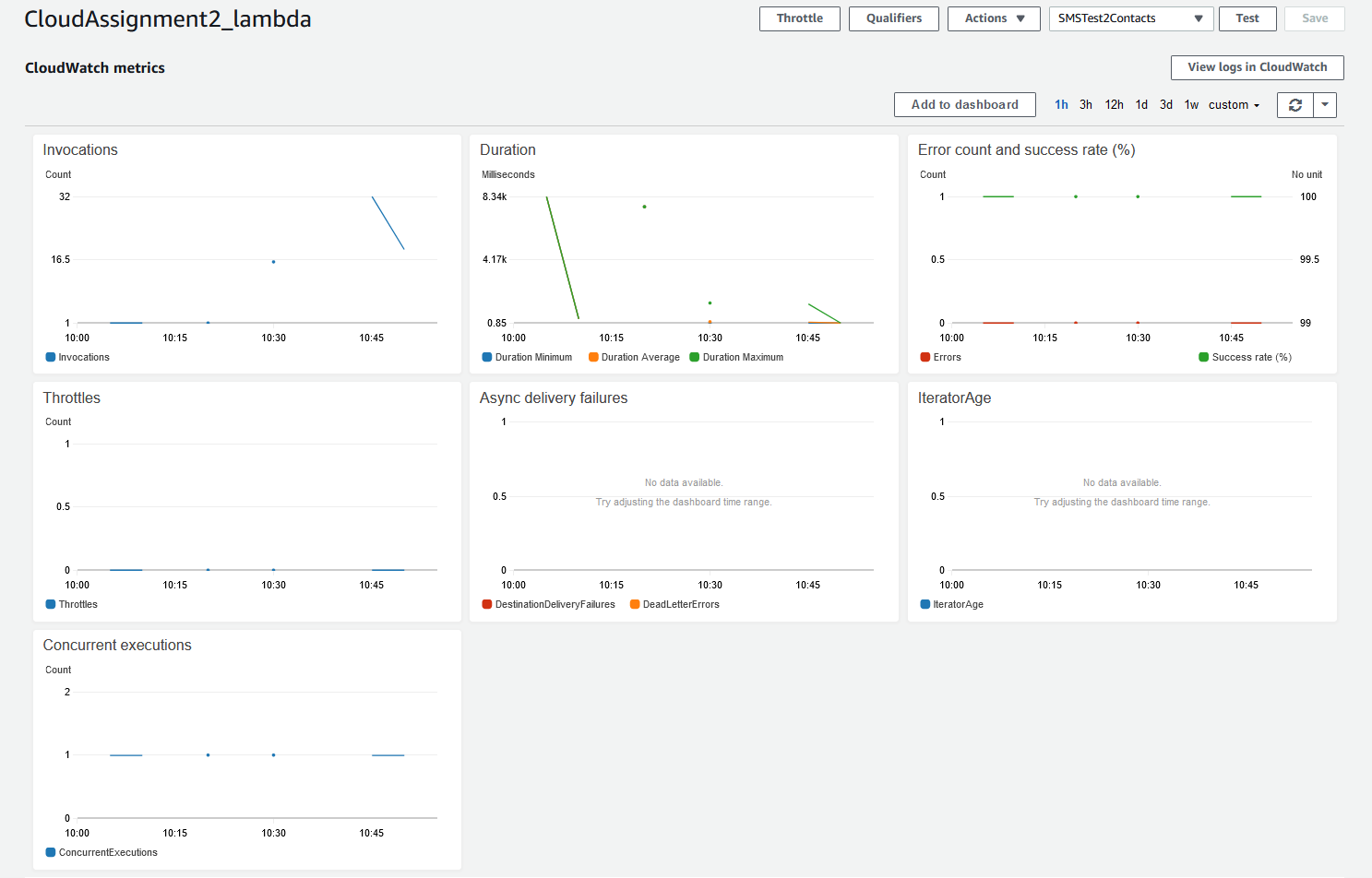


Figure : Amazon Lambda monitoring

### Amazon Simple Service

Amazon Simple Notification Service (SNS) is a highly scalable messaging service. Crypto Watch’s Lambda function handler uses this service to send SMS messages to the registered users. The function is invoked by the web application where the recipients and the relevant messages are passes to the function handler. The function handler uses the SNS service to distribute the messages. SNS messages cost ~5 USD per message. In order to reduce the cost during testing, an environmental variable is created “sendSms” is created. When this variable is not set to True, the service only logs the messages and does not send them.

## AWS Supporting services

There are a number of other AWS services used to support the application. These services include

### AWS Identity and Access Management (IAM)

AWS Identity and Access Management (IAM) allows securely managing access to AWS resources and services. In order for the various parts of the application to securely interact with each other, the proper IAM roles and policies need to be applied. For example, the Elastic Beanstalk role need to access S3 and DynamoDB services in order to be able to write to S3 and DynamoDB.

### Amazon CloudWatch

Amazon CloudWatch is a monitoring service that can retain log information for the application and used for debugging and troubleshooting issues. The logs for both Web application and Lambda function are monitored in CloudWatch. The Web application and the Lambda function both need to use logging to ensure the right details are captured in CloudWatch. In addition to the logs, CloudWatch monitors the application for usage and provide valuable information and insight.

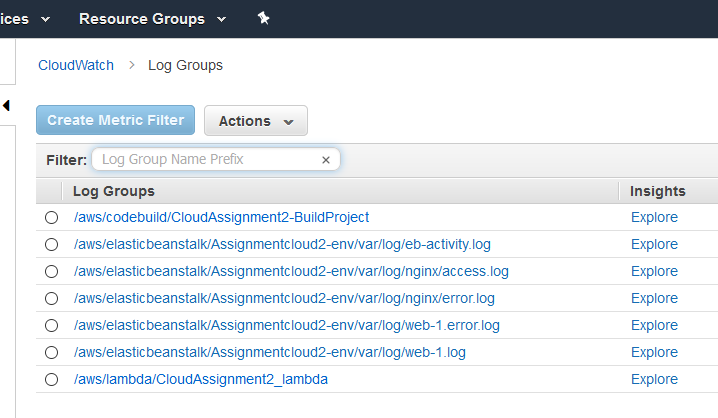


Figure : AWS CloudWatch logging information

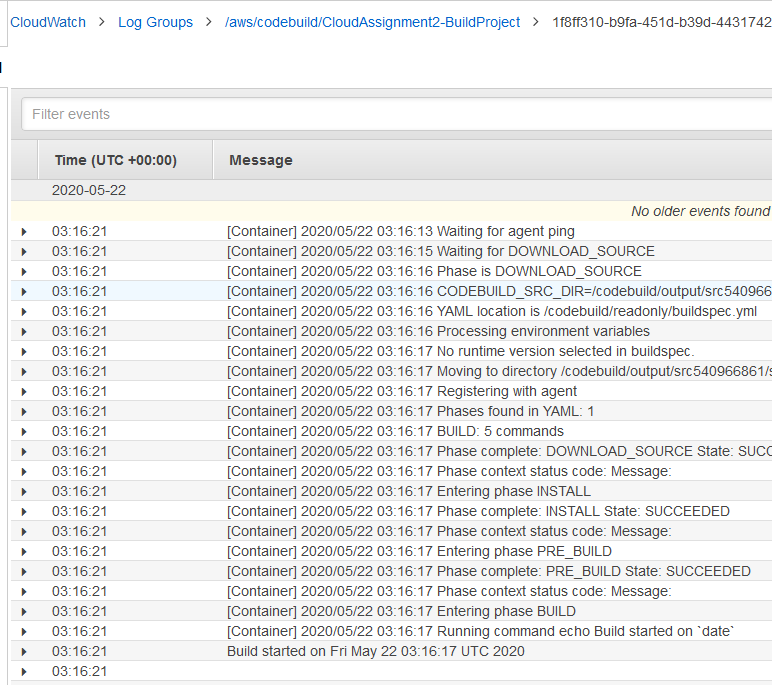


Figure : Sample CloudWatch BuildProject output

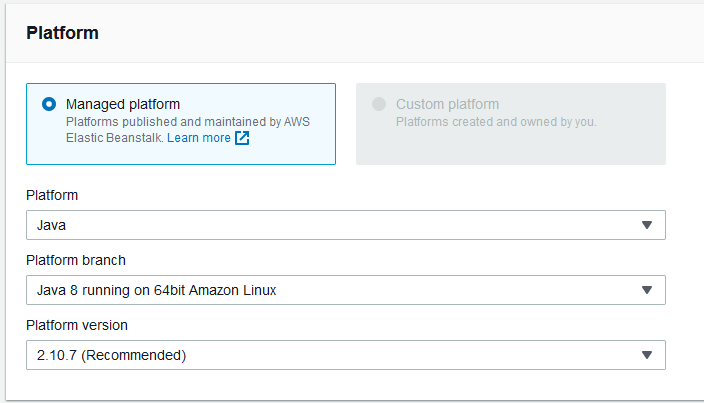
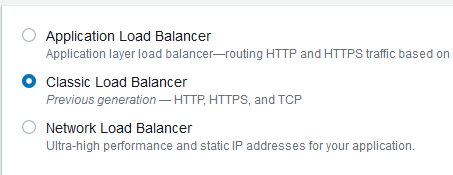
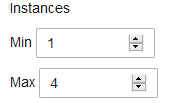
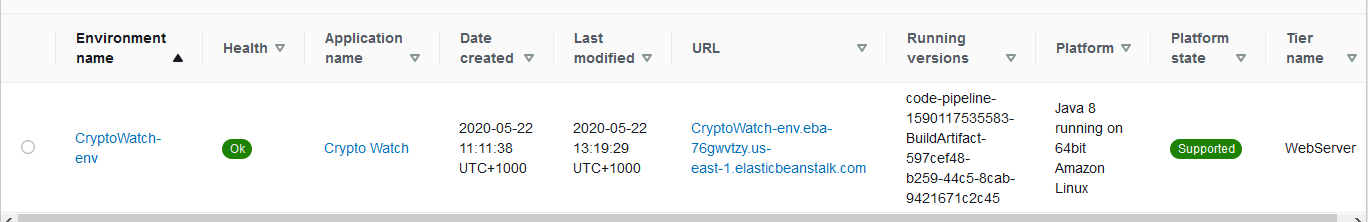
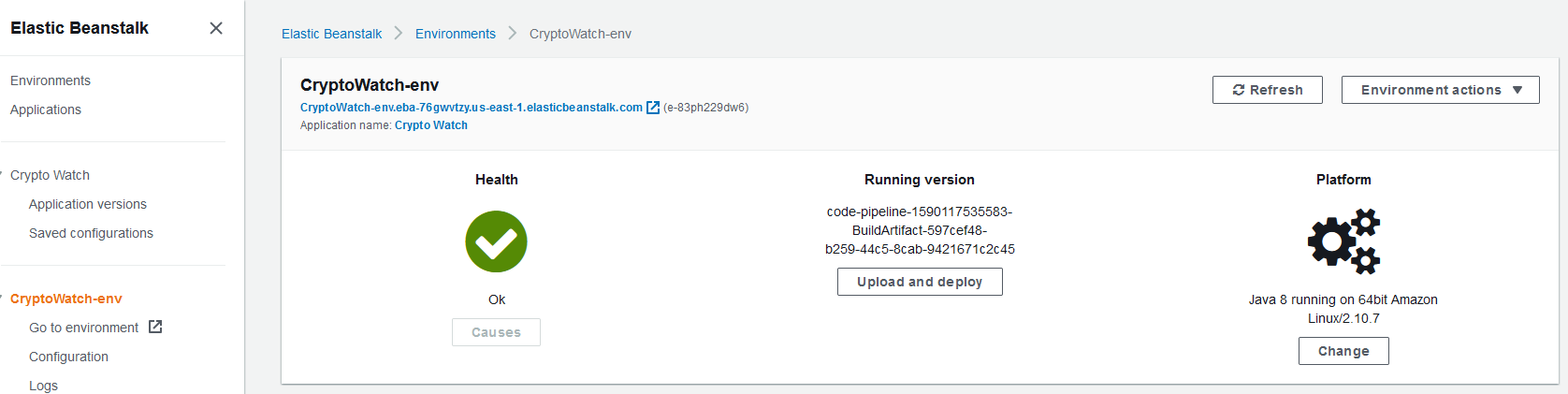
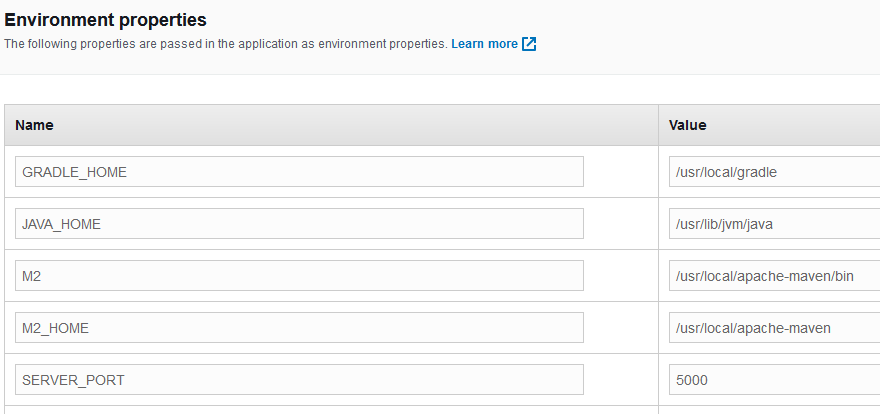
### AWS CodePipeline and CodeBuild

AWS CodePipeline is a managed delivery service that automates build, test and deployment. It uses CodeBuild service to build and test the application packages before deploying them. Using this service simplified the application deployment because as soon as the code was pushed to GitHub, CodePipeline would trigger another build and deploy to Amazon Elastic Beanstalk. This would cause the application to be updated automatically after a few minutes of a push unless there is an issue. Issues are highlighted in CodePipeline and can be resolved to continue the deployment.

# Implementation

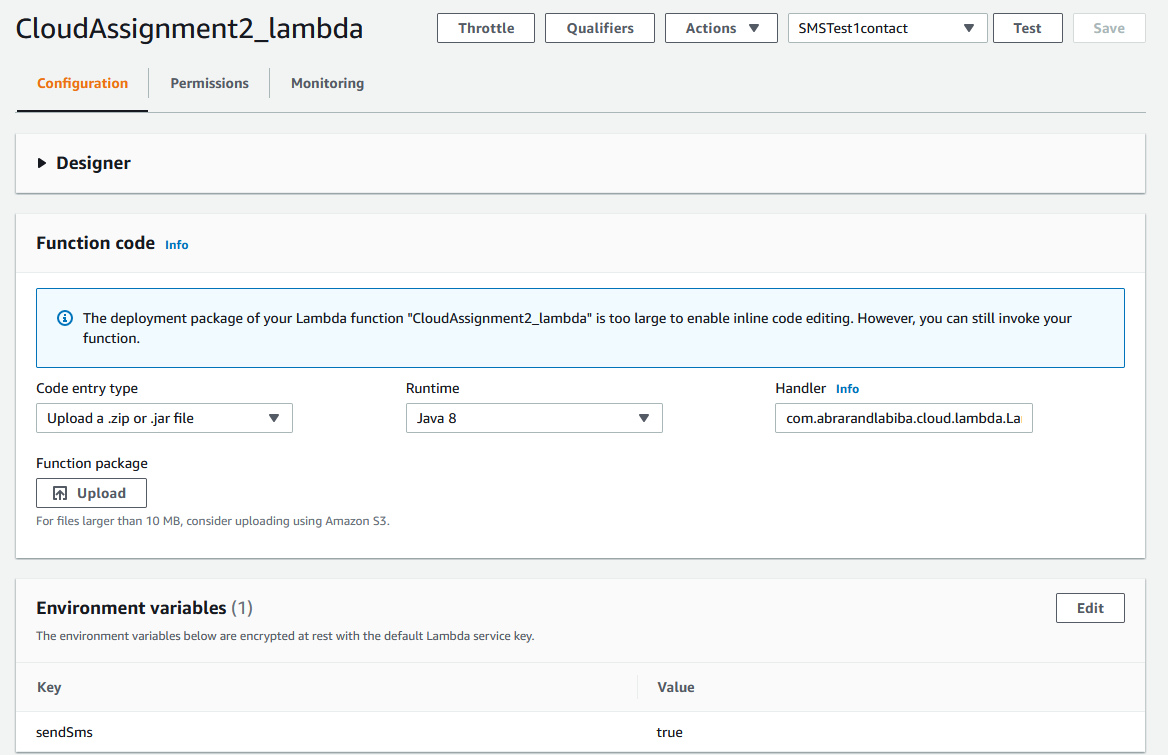
The following is a step by step guide to build the Web application and Lambda Function in Amazon Web Services.

## Create the Elastic Beanstalk Environment

1. Log in to AWS: https://console.aws.amazon.com
2. Search for **Beanstalk** in **Find Services**.
3. Click on **Create New Environment**
4. Select **Web server environment** and click on **Select**
5. TypeCrypto Watchfor **Application Name**
6. In **Platform** dropdown, select **Java**
7. In **Platform branch** dropdown, select **Java 8**
8. Click on **Configure More Options**
9. Under **Presets**, Select **High Availability** radio button
10. Under **Load Balancer**, click **Edit**
11. Select **Classic Load Balancer** and click on **Save**
12. Under **Capacity**, click Edit
13. Make sure the **Min** instances is **1** and **Max** is set to **4**
14. Set the instance type to **t2.micro**
15. Click on **Save**
16. Click on **Create environment**. After a few minutes the new environment is created.  
    
17. Click on the newly created Environment **Crypto Watch**.  
    
18. You can click on the **Go to environment** to see the sample application created
19. Click on **Configuration** to set environment variable
20. Click on the **Edit** button for **Software** category
21. Under Environment properties, type a variable name **SERVER\_PORT** and **5000** for value  
    
22. Click **Apply**.

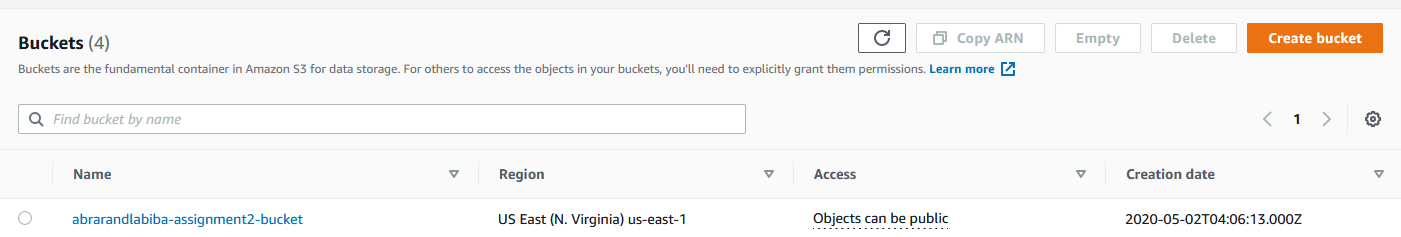
At this stage you have successfully created your Beanstalk environment

## Create the Lambda Function

1. Log in to AWS: <https://console.aws.amazon.com>
2. Search for **Lambda** in **Find Services**
3. Clickon **Create function**
4. For **Function name**, type **CloudAssignment2\_lambda**
5. In **Runtime** dropdown, select **Java 8**
6. Click on **Create Function** button
7. When the function is created, got **Environment variables** section and click on **Manage environment variables**
8. Click on **Add Environment Variable**
9. Type the following for key: **sendSms** and for Value: **true**
10. Under Function code, click on **Upload** button
11. Find the **demo-1.0.0.jar** and select it then click **Open**
12. Click on **Save** button to apply the changes. This can take a few minutes if you’re on a slow internet connection  
    

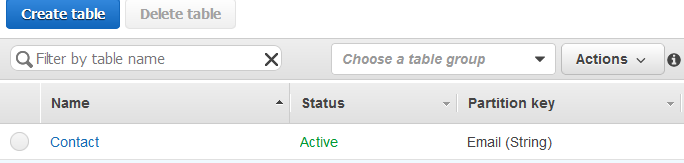
## Create S3 Bucket

1. Log in to AWS: <https://console.aws.amazon.com>
2. Search for **S3** in **Find Services**
3. Click on **create Bucket**
4. For **Bucket name**, type **abrarandlabiba-assignment2-bucket**
5. Leave the region and click on **Create Bucket**



## Create DynamoDb Contact Table

The **Contact** table can be created within the application from Admin page. (see details in User Manual) and can also be created manually using the following

1. Log in to AWS: <https://console.aws.amazon.com>
2. Search for **DynamoDb** in **Find Services**
3. Click on **Create Table**
4. For **Table name** type **Contact**
5. For **Primary key** type **Email**. Leave the type to be **String**
6. Click on **Create** to create the table

## Create CodePipeline and CodeBuild

1. Log in to AWS: <https://console.aws.amazon.com>
2. Search for **CodePipeline** in **Find Services**
3. Click on **Create** **Pipeline**
4. For **pipeline** name, type **cloudassignment2-pipeline**
5. Leave the **Role name** as is and click on **Next**
6. For **Source provider**, select **GitHub**
7. Click on **Connect to** **GitHub** button
8. For **Repository**, Select **s3707180/cloudassignment2**
9. For **Branch**, Select **master**
10. Click on **Next**
11. On **Add build stage** page, For **Build provider**, select **AWS CodeBuild**
12. Under **Project name**, click on **Create project**
13. For **Project name**, type **CloudAssignment2-BuildProject**
14. For **Operating System**, select **Ubuntu**
15. For **Runtime(s)** select **Standard**
16. For **Image**, select **aws/codebuild/standard:4.0**
17. Leave other settings as default
18. Under **Buildspec**, select **Insert build commands**
19. Click on **Switch to editor**
20. Copy paste the **buildspec.yaml** file from the **deploy\** folder
21. Click on **Continue to CodePipeline** button
22. Back on **Add build stage**, click on **Next** Button
23. On **Add deploy stage** page, for **Deploy provider**, select **AWS Elastic Beanstalk**
24. For **Application name**, select **Crypto Watch**
25. For **Environment name**, select **CryptoWatch-env**
26. Click on **Next** Button
27. Check the details on **Review** page and click on **Create Pipeline**. The pipeline will be created

This CodePipline will be triggered automatically when new code is pushed to the repository. You can run the CodePipeline manually by going to **cloudassignment2-pipeline** and clicking on **Release Change**. This will run the pipeline by getting the source code from GitHub and build the project using AWS CodeBuild and then deploy the updated application into AWS Elastic Beanstalk

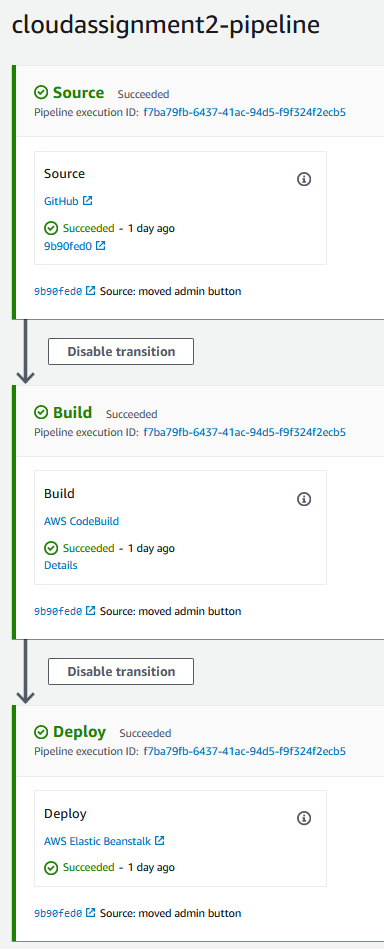
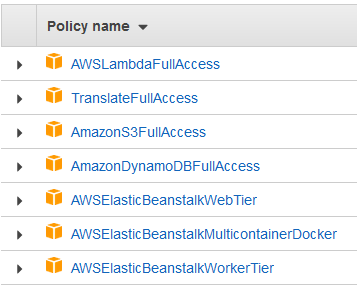
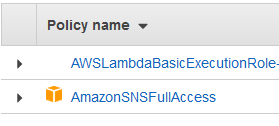


Figure : AWS CodePipeline Service

## Update permissions in IAM

1. Log in to AWS: <https://console.aws.amazon.com>
2. Search for **IAM** in **Find Services**
3. Under **Dashboard**, click on **Roles**
4. Search for **aws-elasticbeanstalk-ec2-role**
5. Attach the following policies to the role so the elastic beanstalk can use the services:  
   AWSLambdaFullAccess  
   TranslateFullAccess  
   AmazonS3FullAccess  
   AmazonDynamoDBFullAccess  
   
6. Search for **CloudAssignment2**
7. Attach the following policies to the role so the Elastic Beanstalk can use the services:  
   AmazonSNSFullAccess  
   

# User Manual

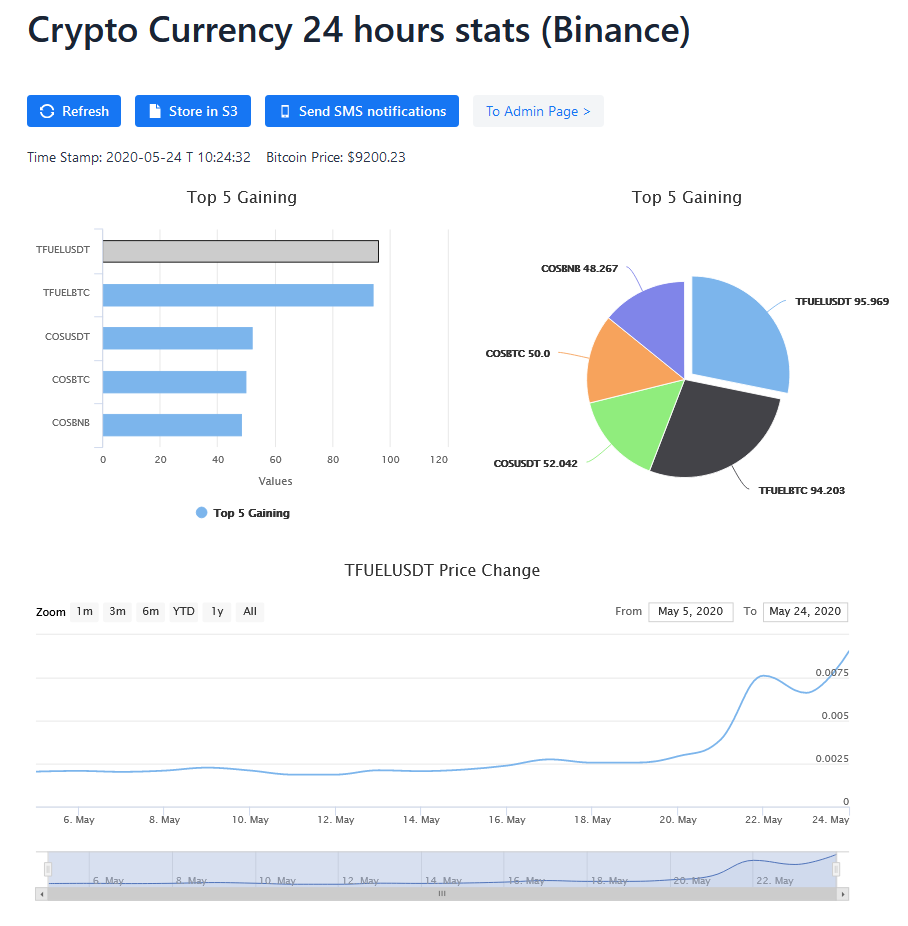
You can go to the Crypto Watch web application by using to the following link:

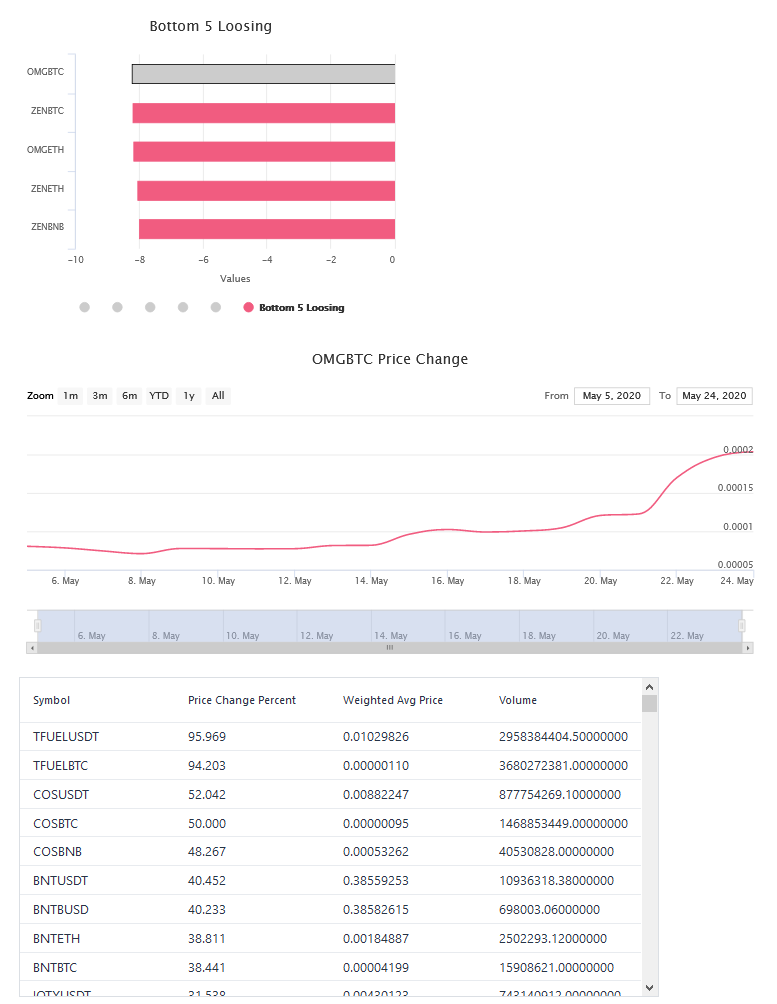
<http://cryptowatch-env.eba-76gwvtzy.us-east-1.elasticbeanstalk.com/>

You will land in the Main page of Crypto Watch web application.

## Crypto Watch Main page

The main page is shown below in two screenshots:





The main page has 4 buttons: ‘Refresh’, ‘Store in S3’, ‘Send SMS notifications’ and ‘To Admin Page’.

* “Refresh” button will refresh the main page and the current top 5 gaining and losing cryptocurrencies.
* “Store in S3” button will store all the information about the cryptocurrencies in S3 Bucket.
* “Send SMS notification” will send SMS to all the contacts added in the Admin Page.
* The “To Admin Page” button will take you to the admin page.

In the main page, below the pie chart, there is a date button which has a From and To date. Clicking on the boxes, the date can be changed, and you can see the resultant graph below. This allow the user to select the time frame in which they want to investigate the data.

From the main page, you can also view the Price Change Percent, Weighted Average price and Volume of other cryptocurrencies. You can scroll below to look for your desired cryptocurrency.

At the bottom of the page there is a table that lists all the availabvle crypto currnecies exchange information.

## Crypto Watch Admin page

A screenshot of a cell phone

Description automatically generatedThis is the Admin page is intended for administrators. The Admin can insert User Contact details like email address and phone number. Admin can choose any of the three supported languages of English, Chinses and Arabic, by clicking on the radio button and add the contact using the “Add Contact” button. The language is used for SMS notification. The messages are translated to the target language prior to transmission.

By clicking on “Create Table” and “Delete Table”, tables can be created and deleted from this web application, instead of DynamoDB. This allows the administrator to interacts with the DynamoDb tables without the need for logging to AWS Console.

At the top, there is the “to Debug Page >” button which will take the Admin to the debug page.

## Crypto Watch Debug page

The debug page is used mostly for development and troubleshooting. It provides a number of good to have functions that were very useful during development.

A screenshot of a social media post

Description automatically generated

In the debug page, you can type in any input sentence in the Input Sentence box in English. You have to select the language you want to translate to and click on the “Translate button”. The Translation text field will show the corresponding translation in Chinese or in Arabic depending on the selected language.

# References

* Spring Boot: <https://spring.io>
* Apache Tomcat: <http://tomcat.apache.org/>
* Apache Maven: <https://maven.apache.org/>
* Vaadin: <https://vaadin.com/>
* Spring.io. 2020. *Creating CRUD UI With Vaadin*. [online] Available at: <https://spring.io/guides/gs/crud-with-vaadin/> [Accessed 24 May 2020].
* GitHub. 2020. *Public Rest API For Binance*. [online] Available at: <https://github.com/binance-exchange/binance-official-api-docs/blob/master/rest-api.md> [Accessed 24 May 2020].
* Amazon Web Services, Inc. 2020. *How To Set Up A Continuous Deployment Pipeline - Amazon Web Services*. [online] Available at: <https://aws.amazon.com/getting-started/tutorials/continuous-deployment-pipeline/> [Accessed 24 May 2020].
* Amazon Web Services, Inc. 2020. *AWS SDK For Java*. [online] Available at: <https://aws.amazon.com/sdk-for-java/> [Accessed 24 May 2020].

# Signed Contribution Agreement

A screenshot of a cell phone

Description automatically generated