Architecture Critical Analysis and Response

KRYPTON
BY STORMTROOPERS

CSCI5409: Advanced Cloud Computing

Group 40 (Team 3): Stormtroopers

Tanvi Pruthi – B00875949

Sidharth Mahant – B00899439

Mayank Sareen – B00899565

DALHOUSIE UNIVERSITY

Table of Contents

System Architecture

The Krypton website is a trading broker website where users can register/login and buy/sell Cryptocurrencies. The website lets users maintain their portfolio of cryptocurrencies and perform the transactions with a user-friendly and robust website hosted and supported on the AWS cloud platform. Whenever any unregistered user visits the website, the home page and login/register page are visible to the guest. However, when a user registers to the website, a personalized dashboard and trading page is accessible to that registered user.

The system architecture will help communicate the website front-end and the back end via a network. The website will deploy on the Elastic Beanstalk container developed with a docker image, which will contain the customizable EC2 instances and a load balancer that will redirect the website traffic to the scalable EC2 instances hosted in the Beanstalk container. Once the health check in the beanstalk passes, the environment created in the Beanstalk application will redirect the website to the provided URL in the python flask application. The root folder of the website code will contain a requirements text file, which holds all the package requirements of the website, which the Beanstalk environment will fulfil while running the health status of the website package. Also, the beanstalk will host on the AWS Virtual Private Network (VPC) to maintain the website and registered users' security.

The home page fetches the live data using API calls whose data gets updated in the AWS DynamoDB first, followed by a presentation on the home page from the DynamoDB for the faster execution of the data flow. Furthermore, the login/register page leverages the AWS Cognito service that helps the user to login/register with a customizable UI. Once a user does the registration to the website, the user table in DynamoDB is updated with the user email id to maintain the personalized data of that user. For example, when a user wants to add a cryptocurrency to the watchlist, the user DynamoDB table gets updated with those preferred ones to maintain the user session. Also, the AWS SNS service sends a welcome email to the first-time registered user, and one topic gets created for each registered user in the SNS service. The email id is added to that topic to publish the message to that email id added in the SNS topic.

Once a user logs in to the website, the DynamoDB transactions table will store all the buy/sell data of that user, and the open/completed orders tab will fetch the data from the same DynamoDB table to display all the "open" and "completed" orders of that user. Then, the SNS service will send an email to that logged in user for every successful transaction. Also, the dashboard for that user will fetch the data from the DynamoDB transactions and users table to update the Wishlist and Cryptocurrencies holdings, followed by profit and loss calculated for that user. Furthermore, the users' wallets will update the DynamoDB table after every successful transaction.

The S3 AWS service will contain all the website logs whenever anything happens or triggers on the website. Finally, the AWS lambda function will update the live data on the website home page by fetching the data from the DynamoDB exchange and market tables to update the website at regular intervals. Therefore, the Krypton website hosted on the AWS cloud will leverage the cloud

architecture to access the data, infrastructure, network, storage on an on-demand basis to leverage the power of the cloud resources.

Architecture Components

The Krypton website will use the following cloud components to deliver an architecture on the cloud platform:

1. **Docker and AWS Elastic Beanstalk:** Docker is an open-source platform that uses the docker file to create a source code image and then transmits that image to the Docker Hub, a cloud-based repository for storing source code images. We can use the docker-compose file to build and execute the docker image from the docker hub using AWS Elastic Beanstalk after our EC2 instance is up and running. In this way, Docker will take care of the infrastructure through docker images. For speedy deployment and maintenance of our application, we will run the krypton application deployment on Elastic Beanstalk in EC2.

2. **AWS EC2:** Amazon EC2 will host the containers launched by the AWS Elastic Container Service on various EC2 instances.

3. **Elastic Load Balancing:** Amazon's Elastic Load Balancer is a service that balances traffic towards the application. If there is a high volume of traffic to a specific EC2 instance, the load balancer will shift the traffic to another EC2 instance. If the available EC2 instances are depleted, auto-scaling will be used to increase the number of instances accessible to access the Krypton application.

4. **DynamoDB:** A NoSQL database service offered by Amazon, DynamoDB, is faster and more efficient than S3. It, like S3, saves data in key-value pairs in documents. Krypton will store all user-related data in DynamoDB, such as cryptocurrency trade data, account balances, user portfolios, and so on. Because there are numerous transactions, DynamoDB will handle them efficiently and keep track of their details.

5. **AWS S3:** AWS Simple Storage Service is a NoSQL database that stores data in buckets as objects. It stores the information as a key-value pair. S3 enables the creation of several versions of data, making restoration straightforward. The krypton program will use Amazon's S3 to store log data, images, and other data in buckets. These buckets will be created, deleted, and updated using the REST API.

6. **Amazon Virtual Private Cloud (VPC):** To ensure resource allocation, connection, and application security, the Krypton web application will be installed in an EC2 instance that is attached to a virtual private network. A virtual private network (VPC) is a private network built at a specified location to isolate it from public access. Launch EC2 instances in this private network's public or private subnets, and the network is connected to the internet via

the internet gateway. Depending on the requirements of an application, multiple VPCs can be constructed in a single region.

7. **Amazon Cognito:** Amazon Cognito will retain user data from the Krypton website, making it extremely simple to log in or register using email, Facebook, or Gmail. DynamoDB will have additional links to the user credentials and data to access the data for cryptocurrency transactions. Amazon Cognito is a user identification and data synchronization service that helps to reduce back-end development to save data when a user signs in or registers. It facilitates the integration of the user's identity with the user's multiple social networks and stores the data as key-value pairs.

8. **AWS SNS:** An email will be sent to the user's registered email address when he or she checks in or registers on the Krypton website. Furthermore, when a user buys or sells a cryptocurrency, the user will receive an email including the transaction details. These emails will be sent using AWS' SNS capabilities. Amazon Simple Notification Service is Amazon's messaging service, which allows you to send notifications by email, SMS, mobile push, and other channels.

9. **AWS Lambda:** AWS Lambda is a service that computes code without the use of a server. It is referred to as serverless computing. The code is executed in reaction to events in AWS services such as adding/removing files from an S3 bucket, updating Amazon DynamoDB tables, sending an HTTP request to Amazon API Gateway, and so on.

Architecture Important Aspects

The most important aspect of the system architecture is the Elastic Beanstalk and docker, which will deploy the application and govern the website performance by handling and provisioning the customizable and scalable EC2 instances to maintain the website load and traffic to the website. The beanstalk will maintain and update all the underlying infrastructure, therefore, helping to deliver high performance and scalable website with a pay-as-you-go advantage.
Along with the system infrastructure, the DynamoDB and Amazon Cognito are the other necessary aspects of the architecture. DynamoDB helps maintain the website database (user and live cryptocurrency data) with much faster and more flexible storage. Amazon Cognito helps store and fetch the user’s credentials and user profile integration with the social networks in a simple, secured manner.

Figure 1 demonstrates the data flow in the Krypton website by leveraging the AWS cloud services.

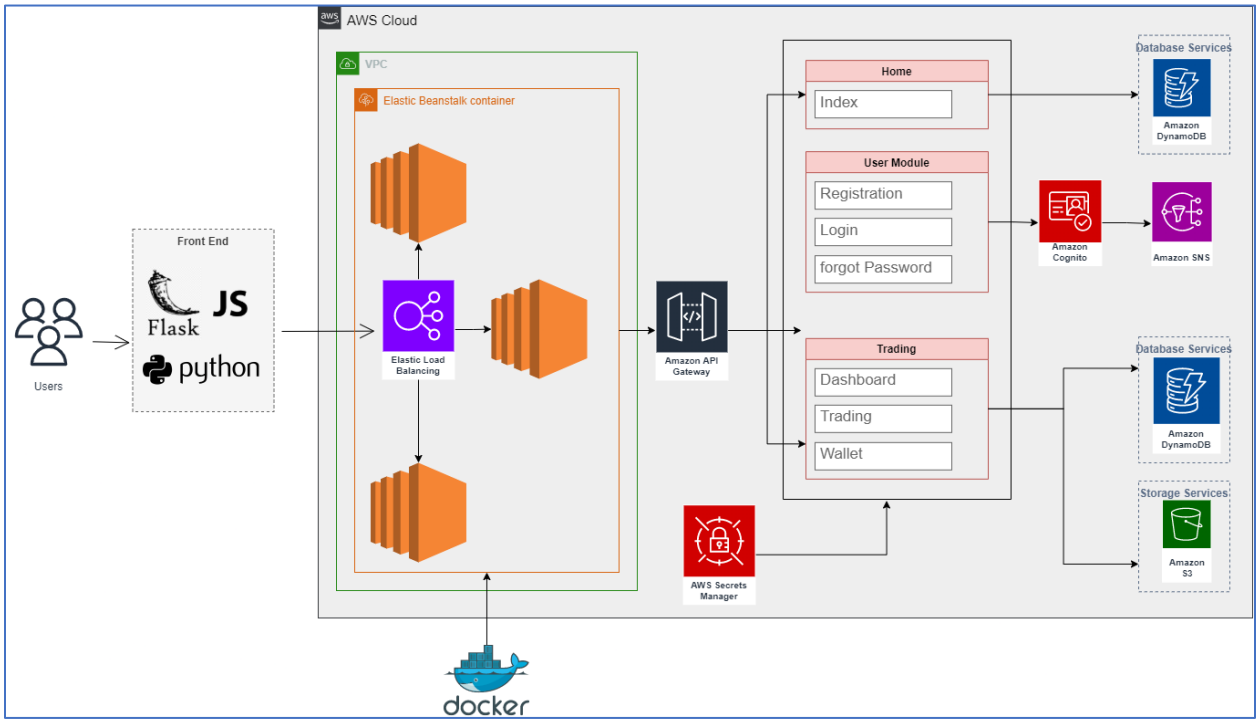

Figure 1: AWS Architecture

Suitable Architecture

The Krypton website will have to handle traffic and triggers from multiple users, fetch the live data from the Cryptocurrencies exchange APIs and perform transactions for many users. To fulfil all the website requirements at a much faster speed and better performance, we have decided to follow the **Service Load Balancing Architecture** taught in the lecture. The main aim is to deliver a website with robust performance, reduce response time and less wastage of IT resources. This architecture will help redirect the traffic to multiple EC2 instances as per the requirement, auto scaling the resources and balancing the load on the website efficiently. In addition to this, the resources would scale down to normal usage when the website receives less traffic. Since the on-demand use and pricing of the resources is available, this architecture is best suited at an affordable price. Also, the health check provided by the Service Load Balancing Architecture helps monitor the website's health and the resources available to operate the website.

We planned to use the AWS Elastic Beanstalk and the docker platform to deploy the website on the AWS cloud platform. Beanstalk acts as a container that contains the EC2 instances, Nginx load balancer and the secure network to run and monitor a web application. This design architecture is similar to the Service Load Balancing Architecture with one difference. We did not consider the multiple cloud systems for the website deployment. We only evaluated the public cloud in our design as a matter of fact. Furthermore, we did not ponder the state of the resources in case of failure detection and recovery management.

Architecture Alternatives

We believe that the Service Load Balancing Architecture best suits the requirements of the Krypton website, where we focus on the better scalability and utilization of the resources to enhance the performance of the application. The load balancer will distribute the workload and speed up the data integration and response time.

There are other architecture alternatives to consider if we did not select the Service Load Balancing Architecture, for example, Dynamic Scalability Architecture, Dynamic Data Normalization Architecture, and Cloud Balancing Architecture.

Below is the comparison of the Service Load Balancing Architecture with the other alternatives:

1. **Dynamic Scalability Architecture**
   Dynamic Scalability Architecture will scale up and down the resources as required, I.e., scale up when traffic on the website is huge and scale down when the website receives less traffic. However, the Elastic Beanstalk that we will use in the project will automatically perform the auto scaling to improve the performance of the website.

2. **Dynamic Data Normalization Architecture**
   The dynamic data normalization architecture assists in automatically normalizing the data stored in cloud storage devices, hence increasing the capacity of the cloud storage devices. This enhances the performance of the AWS Cloud-hosted application. However, the Krypton design does not normalize the redundancy of data saved in Amazon Cloud Service storage devices. Our architecture, on the other hand, auto-scales incoming traffic.

3. **Cloud Balancing Architecture**
   The Cloud Balancing Architecture helps in provisioning of the resources across multiple cloud services for better reliability and optimization of the resources. We will use Elastic Beanstalk to achieve the IT resources provisioning with a load balancer to redirect the traffic across multiple resources and utilize the resources when required.

References

[1] “Configure Amazon DynamoDB,” *AWS* [Online]. Available: https://calculator.aws/#/createCalculator/DynamoDB. [Accessed: March 19, 2022].

[2] “Start Building on AWS today” *AWS* [Online]. Available: https://aws.amazon.com [Accessed: March 19, 2022].

[3] “Fundamental Cloud Architectures” *Informit* [Online]. Available: https://www.informit.com/articles/article.aspx?p=2093407&seqNum=9 [Accessed: March 19, 2022].

[4] “AWS Architecture Explained: Function, Components, Deployment Models & Advantages” *upgrad* [Online]. Available: https://www.upgrad.com/blog/aws-architecture/ [Accessed: March 20, 2022].

[5] “Diagrams.net - free flowchart maker and diagrams online,” Flowchart Maker & Online Diagram Software. [Online]. Available: https://app.diagrams.net . [Accessed: March 20, 2022].