CloudStore

Andreas Lazarou

Computer Science Department
University of Bristol
Bristol, UK

fg18223@bristol.ac.uk

(fg18223)

***Abstract*— CloudStore is a cloud web application that provides the users with the ability to advertise their products both in a store containing products from other sellers, as well as a personal store displaying only their products. The seller can upload a picture of their product with accompanying information, which is directly uploaded to the cloud and displayed in CloudStore. The application was built using PHP server-side scripting language and uses AWS Elastic Beanstalk as an orchestration service. The application uses Amazon RDS for MySQL as a storage device and EC2 instances to run the application. The application can be accessed at http://cloudstore.eu-west-2.elasticbeanstalk.com.**

Keywords*— *Cloud Computing, CloudStore, Amazon Web Services (AWS), PHP, MySQL

I. INTRODUCTION

CloudStore brings the idea of a convenience store into the cloud. By allowing the users to be both vendors and buyers at the same time, CloudStore diminishes the need for middlemen in people's everyday purchases. CloudStore works by requiring the user to sign up for an account. The information asked include an email address which provides people with the ability to contact them for purchases, and a username which will be the displayed seller's name in the global store.

When the users are logged in, they have many choices on how to proceed with their store. On the website, links are visible which allow the user to visit the global store, personal store, or their favourites. When the user visits the global store, he can add products to his favourites list or visit another user's store to view their products. By visiting their personal store, the user can add or delete products to be displayed both on their personal, and on the global store. Visiting their Favourites, users can view all products that have been added to favourites and remove any that are no longer desirable. Figure 1 shows the homepage and some of the listed products in the global store.

Figure 1: Homepage & Global

Due to the high need of availability, using cloud services to deploy this application is very beneficial, as it ensures very high availability with minimal costs. AWS services provide Fault tolerance, as well as consistency, availability and partition tolerance. By running the application in multiple instances, we can ensure that all three properties of the CAP theorem are fulfilled for both our RDS database as well as the Server.

II. AWS SERVICES

Amazon Web Services (AWS) [1] is a secure platform that allows users to deploy applications on the cloud. It offers a variety of services such as compute power, database storage, content delivery and other functionalities to help developers deploy and scale their applications. AWS, unlike traditional solutions, allows the users to build sophisticated applications with increased flexibility, scalability and reliability. As the need for scalable applications is consistently growing, Amazon Web Services provide an easy solution for their development.

A. *Amazon Relatitonal Database Service(RDS)*

Amazon RDS [2] is a distributed relational database service that runs in the cloud. It was developed to ease setting up, operating, and scaling a relational database in the cloud. It automates time-consuming administration tasks, for example hardware provisioning, creating backups, and setting up the database. It provides the database with a resizable capacity, and allows developers to attain fast performance, high availability, security, and compatibility in their application.

Amazon RDS is available for six familiar database engines. CloudStore uses Amazon RDS for MySQL [3] but can be easily migrated to another service by using AWS Database Migration Service. Amazon RDS allows developers to deploy scalable MySQL servers with resizable hardware capacity. By integrating MySQL and Amazon RDS, backups, software, patching, monitoring, scaling, and replication become automated processes.

B. *Amazon Elastic Compute Cloud(EC2)*

Amazon EC2 [4] enables developers to launch instances (Virtual Computing Environments) with different operating systems and load applications to them. This allows developers to use EC2 instances as servers, providing scalability, as the developer can run as many or as few virtual servers as needed. EC2 allows the configuration of security and networking, as well as managing storage. It automates scaling up or down to handle changes in requirements or spikes in popularity. EC2 instances provide preconfigured templates, allowing the developer to deploy the application without thinking about the operating system, or additional software needed. EC2 allows the developer to

completely configure the instance, choosing configurations such as CPU/memory/storage configurations, regions and Availability Zones, a firewall that uses security groups to filter content from unauthorized sources, and others.

C. *AWS Elastic Beanstalk*

AWS Elastic Beanstalk [5] is an orchestration service that allows developers to deploy and manage applications in the Cloud, while it figures out the correct infrastructure to run those applications. It gives full control and choice to the user while reducing management complexity. Elastic Beanstalk can automatically handle capacity provisioning, load balancing, scaling, and health monitoring. It can support applications developed in many languages, and when the application is deployed, Elastic Beanstalk provisions one or more AWS resources, such as EC2 instances. Elastic Beanstalk works by allowing the user to create an application and upload a version of the application as a zip file containing the source code. It then configures the AWS resources needed and creates the environment. The version of the application can then be updated to add extra functionality, and Elastic beanstalk ensures all instances are updated. The lifecycle of Elastic Beanstalk can be seen in Figure 2.

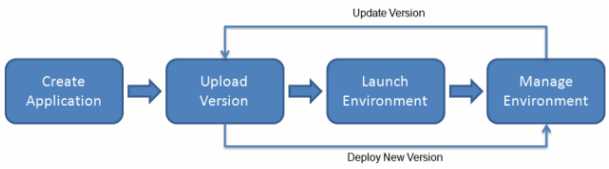

Figure 2: Elastic Beanstalk lifecycle

III. Architecture of the Application

CloudStore is built using PHP along with HTML and CSS to provide static content. Elastic Beanstalk is used to create an EC2 instance which acts as a server. The server can then communicate with an RDS for MySQL database to insert and fetch information for the users.

A. *Hypertext Preprocessor (PHP)*

PHP is a server-side scripting language which allows operations such as:

- Generating dynamic page content
- Handling file operations on the server
- Collecting form data
- Sending/Receiving cookies
- Accessing databases
- Controlling user access
- Encrypting data

The web application works by creating tables in a pre-defined RDS for MySQL database. When a new user registers, their credentials are added to the users table which contains all registered users and a new table is created specifically for them, which will contain any products they have added to their favourites. To login, the server needs to validate the credentials of a user. PHP fetches data from the users table by running MySQL queries, and compares them with the login form filled by the user. If the username and password are a match in the table, a session is created by PHP and the user is logged in.

Once the user is logged in, they can perform a variety of actions. A brief explanation of how to use the web application is displayed in the home page. The user can visit their personal store to view the items they are advertising. This is achieved by using the session created on the login page. When the session is created, a variable username is also created containing the username of the person that is logged in throughout the session. By using that variable in the MySQL query, the table only displays products advertised by the logged in user. By pressing on the Add Product button, the user can add information about a new product which will be appended in the table of the database containing all products. This is achieved by PHP handling the input of the form and creating a query to insert this product in the table. This product will be then displayed on the user's personal store as well as the global store. To remove a previously advertised product, the user can enter the product's ID and click on the remove product button. The user can select how to sort the products by selecting from a drop-down box and clicking on the sort button. The query fetching the items is then changed and the products are displayed sorted as the user selected. Figures 3 and 4 display how the pages work.

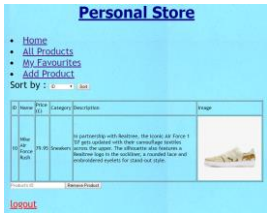

Figure 3: Personal Store

Figure 4: Add Product

The user can select to visit the All Products page, which redirects them to a page displaying all products users have uploaded to the server. On this page users can click on the seller of an item which redirects them to that user's personal store to view their products and contact information. By clicking on the id of an item, the user can add it to their favourites to review it later. This is achieved by inserting the product's id in the table created when the user signed up. The user can see their favourites by clicking on "My Favourites" link which redirects them to the favourites page. Any products added to their favourites are displayed, and the user has the option to remove a product from their favourites. As with the personal store, the user has the option to sort the list of products. Throughout the pages, the user can click on the logout button to destroy their session and logout. Figures 5 and 6 display the functionality of the pages.

Figure 5: All Products

Figure 6: My Favourites

B. *AWS Features*

The web application uses AWS features to integrate with the cloud such as EC2 instances and RDS. Elastic Beanstalk is used as an orchestration platform to handle capacity provisioning, load balancing and auto-scaling. Even though it automates many configurations, the developers retain full control of any resources being used. CloudStore uses elastic beanstalk to achieve scalability and to be able to handle higher traffic rates.

EC2 instances [6] have many different types. CloudStore uses the t2.micro type instances. These are general purpose burstable performance instances providing a baseline level of CPU performance but can also burst above the baseline. They have high frequency Intel Xeon processors and can balance compute, memory, and network resources.

CloudStore is configured, using Elastic Beanstalk, to have between 1 and 4 distances, depending on the load of the web application. The application uses the Classic Load Balancer [7] configuration of the AWS Elastic Load Balancing. This distributes the incoming traffic across multiple EC2 instances in multiple availability zones. By doing this, CloudStore becomes Fault Tolerant, as the elastic load balancing used detects unhealthy instances and routes the traffic only to healthy instances. It also increases the efficiency of the application, as the load is evenly balanced between the instances.

To increase the availability of the application, the load balancer acts as a single point of contact for clients. This allows the addition or removal of instances from the load balancer, as needed by the application. The Classic Load Balancer is preconfigured containing one listener, which checks for connection requests from clients, and forwards the requests to the instances. The Classic Load Balancer can have more than one listener. Figure 7 shows how the load balancer works.

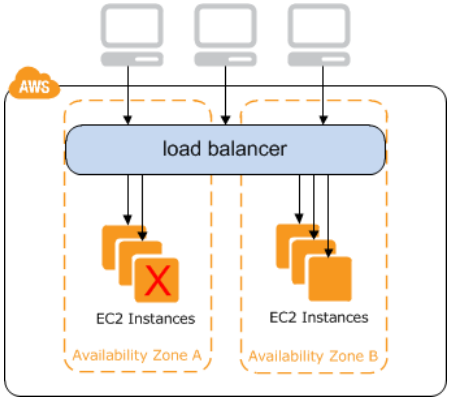

Figure 7: Load Balancer

To allow communications between the EC2 instances and the RDS database created, the security group must be configured. AWS security [8] groups are associated with EC2 instances and provide security for the protocol and port access level. A security group works very similarly to a firewall, filtering traffic targeted to or leaving from the EC2 instance. If a rule does not exist to allow a data packet through, then the packet is dropped.

Databases use similar security groups to allow read/write operations on the databases. To allow CloudStore to access the RDS database created, a rule was created allowing traffic between the EC2 instances and the Database instance. RDS allows for multiple availability zones deployment for the database instance, if the Free Tier limit is exceeded. Figure 8 shows how CloudStore uses security groups [9] , by allowing traffic to the database from the EC2 instance.

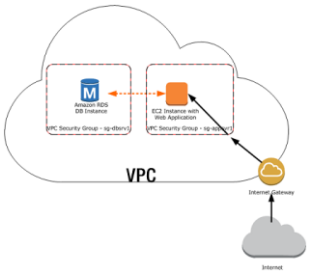

Figure 8: Security Group

IV. Scalability

As technology continues evolving, the creation of non-scalable applications is diminishing. Most technological companies have already began migrating to the cloud and building scalable applications, because the benefits outweigh the risks. In order for an application to be scalable, it must be able to perform at approximately the same level regardless of the load assigned to it. As the number of people using applications increases by the hour, and the majority of people use the same applications as their co-workers, friends, and family, creating an application that does not handle large increases in simultaneous users, data volume and other workloads can segregate the application from the rest, and create an unfavourable outcome.

Testing the scalability of an application is equally important as creating the application because once the application is deployed, the first impression of the users can make the difference on whether the application will be a success or a failure. There are many ways to test the scalability of a web application, for example the response time, screen transitions, throughput, network and memory usage, and time taken to execute tasks. The most common form of scalability testing is the load testing.

As discussed in section III CloudStore's load balancing and scalability is conveniently managed by Elastic Beanstalk. Elastic Beanstalk manages everything from load balancing, auto scaling, instances and capacity of the web application, to ensure availability and fault tolerance for CloudStore. To improve the vertical scalability of the web application, the type of EC2 instance needs to be changed, having more CPUs. While the RDS database is already scalable, its scalability could also be improved, both horizontally and vertically. To increase the vertical scalability of the database, the instance classes used could be changed to provide more virtual CPUs as well as more RAM. To increase the horizontal scalability of the database, more replicas could be used. These features are unfortunately outside of the Free Tier of Amazon Web Services.

V. Future Work

While CloudStore is already at a scalable and working condition, there are a lot of things that need improvement before deploying the application to the world. These are both in terms of modifying the software of the application, as well as integrating it with hardware. As CloudStore is a data warehouse, a physical warehouse could be integrated with the application where sellers could send their products for storage, ease of management, and delivery.

Most of the AWS features CloudStore currently uses are provided by the free tier. This forbids the application of reaching its full capabilities. Most of the restrictions in the free tier are revolved around vertical scalability, as Amazon requires monthly payment to increase the capacity of existing hardware. When CloudStore is deployed, the usage of AWS that are not on the free tier will be much more profitable, as the improvement of the application's scalability directly corelates with the users' ability to access the application, therefore increasing the profit margin of CloudStore. Once CloudStore is fully integrated in the cloud, scalability testing should take place, thinking about availability zones and replicas of both the databases as well as the EC2 instances.

As CloudStore is a prototype, the aesthetic features of the application were not focused as much as the functionality. Before deployment, the style of the application will be changed to be more attractive to the users. Further functionality could be added to the application such as allowing card/cryptocurrency payments, rating of products and sellers, and delivery services. Sellers should be able to customize their personal store, advertising specific products which are focused for a specific range of people. The database could be sharded by categories, allowing users to view only the products in a particular shard, creating a faster, more scalable store, with improved navigation.

VI. Conclusion

Amazon web services provide easy and cheap integration of applications to the cloud. As the needs of every application are different, AWS provides the developer with a variety of options to choose the one that fits the needs of their application. By providing reasonable pricing and flexible scalability configurations, AWS is one of the best cloud computing platforms.

CloudStore has achieved to utilise many of Amazon's Web Services to create a robust and scalable system, such as the Elastic Beanstalk, Relational Database Service, Elastic Compute Cloud, and Elastic Load Balancer. By using both Infrastructure and Platform as a service, CloudStore is heavily automated, increasing its fault tolerance and allowing the developer to focus on other parts of the development process.

The scalability of the CloudStore has been automated by Elastic Beanstalk, to ensure that any problems are automatically fixed, and the downtime of the application is minimal. By using a load balancer, CloudStore is ensured to distribute client requests efficiently across the instances of the servers.

To conclude, CloudStore is a well thought prototype which could be further developed to overtake the trading market. By using solutions discussed in the Future Work section, CloudStore could become a professional web application and could easily be deployed in a short period of time. Given that only one person developed the application, by increasing the number of developers working on the application CloudStore is guaranteed to be a success.

References

[1] Amazon, Amazon Web Services, Accessed January 2019. [Online]. Available: https://aws.amazon.com/what-is-aws/

[2] Amazon, Relational Database Service, Accessed January 2019. [Online]. Available: https://aws.amazon.com/rds/

[3] Amazon, Relational Database Service for MySQL, Accessed January 2019. [Online]. Available: https://aws.amazon.com/rds/mysql/

[4] Amazon, Amazon Elastic Compute Cloud, Accessed January 2019. [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/concepts.html

[5] Amazon, Amazon Elastic Beanstalk, Accessed January 2019. [Online]. Available : https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/Welcome.html

[6] Amazon, Amazon Instance Types, Accessed January 2019. [Online]. Available: https://aws.amazon.com/ec2/instance-types/

[7] Amazon, Elastic Load Balancing, Accessed January 2019. [Online]. Available: https://docs.aws.amazon.com/elasticloadbalancing/latest/classic/introduction.html

[8] Stuart Scott, AWS Security Groups: Instance Level Security, Accessed January 2019. [Online]. Available: https://cloudacademy.com/blog/aws-security-groups-instance-level-security

[9] Amazon, Relational Database Service, Accessed January 2019. [Online]. Available: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Overview.RDSSecurityGroups.html