Cloud Application Services: AWS

Christopher Herrera Magana
19127723
MSc in Cloud Computing
Link to the Cloud App http://3.248.209.41:5000/

I. Introduction

.

Over 130 years ago, data storage played a significant role in the communication. Despite contributions like the punched card and software-defined store, it remains an issue [1]. Forward to the digital era of smart devices, old solutions need to evolve to the current age. Digital data is exchanging at an exponential rate and large-scale amount. Moreover, the transmission of data relies on the computer foundation, which was an issue in the past.

Nowadays, Cloud computing is transforming the burden of physical architecture with the pay-as-you-go model. Likewise, another advantage relies on the numerous Cloud services, adjusting the client's needs at any time anywhere. Old services troubled by the lack of a flexible equipment lease, leading to higher capital expenditure. Big data comes to life as a result of the current problems with managing data. However, it also brings a laborious and time-consuming process to examine information. Unstructured and structured data resemble a picture, a video, a voice note, or text; Thus, the requirements to synthesize all this into a manageable service. From a technological aspect, Cloud computing value carries more than a solution or concept — goes beyond allowing endless opportunities and freedom either for businesses or individuals. In this report, we investigate the different methods in the design of a Cloud storage Infrastructure. Also, discuss a feasible implementation using Amazon Web services (AWS). Our focus is based on the most critical storage problems using Cloud computing providers, and to identify suitable Cloud deployment services to simplify those issues. Finally, we present recommendations, findings, and future directions for the Cloud storage and messaging infrastructure.

A. Contribution

This paper investigates the methods and design used for object storage in the Cloud. Besides, the different services and solutions, our focus leads towards the integration of an object storage messaging platform. Hence, using Amazon Web Services(AWS) to deploy our project. Finally, we introduce advice, findings, and future directions for the Cloud storage and messaging infrastructure.

This research is organized as follows: Section II, Background. Section III, Methodology, and Implementation. Section IV, Challenges, and Tests. Finally, Section V, Conclusions, Results, and Findings.

II. Background

A. Object Storage

The object-based storage architecture helps to analyze and process massive amounts of data with unstructured patterns. With the increase of technological devices, there is a necessity to transform the way to send and receive information in real-time. Nowadays, Big data technology contributes to reduce time in analysis and classification. However, it is required to use sophisticated machine hardware to satisfy those needs. Storage is relevant in every field to save, retrieve, update, and secure the information to make it accessible anywhere. There are many challenges and restrictions to store data; compliance regulations to guarantee that the data is encrypted and just accessible to specific users. Previous storage infrastructure involved complex installations and did have limits on the data capacity.

Cloud computing offers the possibility to deploy any business project in seconds. Some of the options to protect the information relies on using Private Cloud computing architectures. In addition to this, many cloud providers like Amazon Web Services (AWS), Microsoft Azure, Google (GCP) offer a diversity of services to adapt to any customer needs. The big challenge is to select an architecture that will reduce time-consuming operations and costs. The serverless solution became crucial to replace old hardware implementations and their prices; serverless is known as a function that will execute some code once its trigger to obtain or produce some results. Now it is more affordable to migrate services into the cloud that will facilitate the operations. Besides these advantages, one of the remaining obstacles with storing data in the cloud is the security and privacy. With the use of big data, it is easier to classify information; however, in the case that the data needs to be encrypted, this can become strenuous for the machines and engineers.

This era, automated processes are replacing manual processes, reducing significant time and operational costs. Using machine learning algorithms to simplify repetitive methods can lead to a productive environment for the businesses. Here the main challenges are to distinguish what type of information is private and which is not private. For instance, saving user credentials are confidential, and some people can have access to those values, but, with new regulations like the General Data Protection Regulation (GDPR) makes it even harder to process the customer data. Cloud computing replaced the hardware

to cloud services, making it easier to replicate and protect information from any part of the world in real-time.

III. Methodology

A. Cloud storage AND Infrastructure

With the different models of Cloud computing Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (Saas), Cloud storage becomes a software rather than a bare hardware. Using an application Programming Interface (API) facilitates the communication between end clients. Therefore, better infrastructure is employed to replace the lack of adaptability, scalability, and reliability. In the design of a better cloud Infrastructure for a service application, security, and data loss play a significant role in the development. Moreover, monitoring and controlling the flow of information brings challenges; hence cloud providers offer different services to monitor every request of their application. At the same time, it provides to adjust update information in a granular way.

B. Solution and Implementation

In this project, we use Amazon Web Services (AWS) to solve the existing challenges to process information and storage it safely on the Cloud. There are many ways to protect the information and make it accessible from any part of the World. Our approach is based on AWS best recommendations to build a reliable infrastructure. First, the components or services used need to be decoupled, which will reduce the affectation or failures on other elements. Also, Cloudwatch, which is a monitoring service, will allow us to set up alerts depending on the behavior of our server. Once an alarm is triggered, Cloudwatch can communicate with other services as Amazon Simple Notification Service (SNS) or Lambda, which is an event-driven service.

In our deployment, we use an EC2 instance, which is the service application, then using the Flask framework as a RESTful API allowing the communication with Amazon Simple Queue Service (SQS) integrated with Lambda as our client application. Once the user sends a request (input), the value is going to be analyzed by Lambda, which is triggered by the SQS delaying the messages for 15 seconds. Also, Lambda will communicate with Cloudwatch and SNS to notify them that a successful request has been received. Finally, using Dynamodb, which is a non-relational database to store the inputs and update their values or changes. One of the main criteria we followed in this deployment is the security and the cost involved. AWS Identity and Access Management allows us to create roles, groups, and attach different policies to users based on their identity and the access to the resources they have. In this case, we created a new user who will have the credentials to manipulate at least five services (Cloudwatch, Lambda, SNS, SQS, and Dynamodb) with that the user cannot interact with other services. If someone hacks the account, you only deleted that user.

Protecting the user and their information is imperative; using AWS as our cloud provider, it offers DDoS protection, and in case of an attack, it will notify us. Besides, AWS counts with Inspector service, which is a security assessment to check our EC2 vulnerabilities and improve security mechanisms. Furthermore, AWS offers encrypt data in transit or at rest; thus, our application and the results are protected from any malicious attack.

IV. Challenges and testing

The most challenging and time-consuming of this project was the interaction with other programming languages to communicate Flask back-end with HTML, Javascript, and Ajax front-end. Some libraries are not compatible, resulting in changing the architecture or programming in another way to avoid that bottleneck. Likewise, AWS offers the possibility to interact with their services in a programmatically way using Boto3 as their Software Development Kit, allowing interaction with Python language. Other complex challenges is to query the values in the correct way JSON or just string; it required some libraries to make it readable and manageable between the front-end and back-end. Despite the coding challenges, finding authentic and reputable guides to help us build or understand the interaction between different frameworks compatible with python and flask was another hurdle.

In this century, the majority of the solutions are becoming a service on the Cloud. Software solutions are becoming a life, and there is a lack of standardization to communicate with new solutions. Even though decoupling elements helps to improve service reliability, the truth is that programming in different languages with different libraries makes it even harder. Serverless is transforming start-ups or small companies paying just for the time they execute their function, without worrying about equipment. However, legacy equipment's struggle due to the incompatibility in their language or architecture. Therefore, the upcoming obstacles remain in to simplify a standard coding language to operate with old architecture and allow new architecture designs to become serverless without installing any software.

In this paper, we present a different design to integrate Cloud web services and make them interact with each other to build an effortless solution. The first stage of the project was to create the layout of the website; many JavaScript libraries lets you use a free template to adequate your needs. At first, we wanted to show dynamic and real-time CRUD (GET, POST, EDIT, DELETE) operations. However, challenges in the construction of the web page did not allow us to extract specific user-input values due to the programming language incompatibility. Hence, we had to adopt a different architecture design to allow the storage and retrieval of information: we carried-on Unit testing and Functional testing. At first, we validate that each module works independently, printing the results we want to have, then we integrate some cloud services functionalities to make them communicate with each other. Python requests to submit values and keep it on a queue AWS SQS, and Lambda is triggered by that message received. We tested and sent 50 petitions, and Cloudwatch showed messages received. We tried to execute a serverless function to input

items on the Dynamodb table. However, it did not work to call the other services; it blocked the operation. Hence, we use boto3 to call the client and provide the resources of input, retrieve, edit values. We post the results on a different page to make a comparison between the original values vs. modified values.

V. Conclusions

The main conclusion that can be drawn is that Cloud services provide different methodologies to solve data storage issues. Likewise, the facility to develop any RESTful API web server. The whole idea is to integrate Cloud services, which can communicate with other components but are decoupled, so there is not a dependency on each other to cause service affectation. There are many aspects to analyze before applying Cloud services, security and compatibility are essential. This project taught us that there are many programming languages which are not compatible and sometimes it is required to adapt the architecture to other design.

A. Future Direction work

There are pending obstacles to manage large amount of data and to store significant values. Some articles mention that Software-defined storage abstracts their physical layer avoiding the time on hardware and focusing more on programming. Privacy, security, and integrity will be indispensable for the coming years. At some point, the automated systems would evolve to facilitate the integration of services without the burden of different programming languages and time-consuming.

References

[1] C. H. , "Computer Data Storage History." https://www.computerhope.com/history/datastor.htm. Accessed on:April.3,2020.

[2] R. Buyya, S. N. Srirama, G. Casale, R. Calheiros, Y. Simmhan, B. Varghese, E. Gelenbe, B. Javadi, L. M. Vaquero, M. A. S. Netto, and et al., "A manifesto for future generation cloud computing: Research directions for the next decade," *ACM Comput. Surv.*, vol. 51, Nov. 2018. JCR Impact Factor: (6.131).

[3] C.-W. Tsai, C.-F. Lai, H.-C. Chao, and A. V. Vasilakos, "Big data analytics: a survey," *Journal of Big Data*, vol. 2, p. 21, Oct. 2015. JCR Impact Factor: (2.106).

[4] "What Is Big Data? | Oracle Ireland." https://www.oracle.com/ie/big-data/guide/what-is-big-data.html. Accessed on:March.21,2020.

[5] G. , "Analytics." https://www.gartner.com/en/information-technology/glossary/analytics. Accessed on:March.21,2020.

[6] M. D. Assunção, R. N. Calheiros, S. Bianchi, M. A. Netto, and R. Buyya, "Big data computing and clouds: Trends and future directions," *Journal of Parallel and Distributed Computing*, vol. 79-80, pp. 3 – 15, 2015. JCR Impact Factor: (1.819).

[7] A. M. Ghosh and K. Grolinger, "Deep learning: Edge-cloud data analytics for iot," in *2019 IEEE Canadian Conference of Electrical and Computer Engineering (CCECE)*, pp. 1–7, May 2019. CORE Ranking: C.

[8] I. , "The-Future-of-Object-Storage-From-a-Data-Dump-to-a-Data-Lake." {https://www.ibm.com/cloud/blog/the-future-of-object-storage-from-a-data-dump-to-a-data-lake}. Accessed on:April.1,2020.

[9] B. Ravandi and I. Papapanagiotou, "A Self-Learning Scheduling in Cloud Software Defined Block Storage," in *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, pp. 415–422, June 2017. ISSN: 2159-6190.

[10] "Boto 3 Documentation — Boto 3 Docs 1.12.39 documentation." {https://boto3.amazonaws.com/v1/documentation/api/latest/index.html}. Accessed on:April.1,2020.

[11] "Amazon DynamoDB - Overview." {https://aws.amazon.com/dynamodb/}. Accessed on:April.1,2020.

[12] "AWS Lambda – Serverless Compute - Amazon Web Services." {https://aws.amazon.com/lambda/}. Accessed on:April.1,2020.

[13] "Amazon EC2." {https://aws.amazon.com/ec2/}. Accessed on:April.1,2020.

[14] "Amazon Simple Queue Service (SQS) | Message Queuing for Messaging Applications | AWS." {https://aws.amazon.com/sqs/}. Accessed on:April.1,2020.

[15] "Amazon Simple Notification Service (SNS) | AWS." {https://aws.amazon.com/sns/}. Accessed on:April.1,2020.

[16] "Amazon CloudWatch - Application and Infrastructure Monitoring." {https://aws.amazon.com/cloudwatch/}. Accessed on:April.1,2020.