INTERNATIONAL JOURNAL OF ADVANCE RESEARCH, IDEAS AND INNOVATIONS IN TECHNOLOGY

ISSN: 2454-132X
Impact Factor: 6.078

(Volume 8, Issue 4 - V8I4-1162)
Available online at: https://www.ijariit.com

Messenger service using AWS and sprinboot framework

Shivani
shivanip.sit20@rvce.edu.in
R.V. College of Engineering, Bengaluru, Karnataka

Rekha B. S.
rekhabs@rvce.edu.in
R.V. College of Engineering, Bengaluru, Karnataka

ABSTRACT

Cloud computing is a technology for delivering pervasive, appropriate, sought-after networking access to shared computing resources such as storage, networks, servers, applications, and services that can be quickly equipped and supplied with minimal maintenance or service provider engagement. Most organizations have chosen cloud computing because it is believed to be safer and more trustworthy, especially in inventory tracking. Amazon, with its Amazon Web Services (AWS) business, is at the forefront when it comes to providing cloud computing services around the world. The cloud-based email service uses a serverless an architecture which is microservice type which will be enacted using Springboot Framework and Amazon Web Services including AWS Lambda, Simple Notification Service, Simple Queue Service (SQS), Simple Email Service (SES), Simple Storage Service (S3), Elastic Compute Cloud (EC2) and CloudWatch. The proposed work is to implement a cloud-based email service using AWS to send bulk emails with less time using email templates and configure services to monitor the email sends deliveries, failures, and complaints.

Keywords: *AWS, SQS, SES, Email Service*

1. INTRODUCTION

In recent years, cloud computing has become a popular and well-established method for administering and offering valuable services over the internet. Serverless computing is a development of cloud-based programming methodologies and a demonstration of the widespread adoption of cloud ideas. In this era of rapid technological advancements, massive internet companies such as Amazon, Netflix, and LinkedIn build big multistage cloud apps that can be designed, verified, deployed, extended, operated and upgraded independently. Infrastructure costs, in terms of gaining flexibility and agility, are a major roadblock for companies following this model, due to higher server traffic and the Emails could be stored and retrieved from anywhere else in the world because they have become the primary means of communication among users and are used in almost every aspect of our daily lives. Everyone now has an email account, and if using Hotmail, Gmail, or Yahoo, may have a cloud-based email account that can be accessed and shared from anywhere in the world. One disadvantage of cloud-based email is that there will be no idea where email messages and sensitive information are stored.

In recent years, Amazon Web Services (AWS) is one of the most popular cloud platform providers in the industry. It provides over 200 cloud service providers to build and manage applications. They are easy to use, and the user should not worry about the servers, security, and databases. AWS's flexible IP deployment and mail authentication options help to boost deliverability and protect sender reputation while sending analytics monitor the impact of each email.

The previous email cloud-based solutions are entirely adequate when having to send a couple of dozen emails manually but don't work when need to send out thousands of emails over the same time. It takes more time to send emails manually and couldn't be tracked whether the email is delivered or bounced back.

The scenario is such that if a message must be sent to many users with the specific formatted template, it will take more time to send manually. To effectively use a cloud-based email service for bulk email communication, email templates are created to store specific content formatted templates, so the emails are sent in less time. The AWS services help to promote products and services such as customers should be able to receive emails automatically from the cloud monthly, so they have a better insight into their product performance with customized content and email templates.

Another scenario is such that the email service must trigger automatically based on the sudden events like rash driving, accidents, fault occurrence, etc found in the vehicle. This should notify the customer immediately of a suitable solution for the problem and send immediate, trigger-based communications from applications to customers.

To prevent unauthorized access to send emails from any domain or email address, the AWS service provides email authentication options flexible deployment choices, including pooled, dedicated, and customer-owned IPs, which aid in influencing sending reputation. Monitor and alert the sender if emails bounce, cause a complaint, or are delivered successfully to the recipient's email system.

2. OBJECTIVES

Three main objectives are suggested based on the characteristics of the problem

1. AWS services are to be utilized in the messenger service to send customized mass emails in the form of plain text, formatted with/without attachments using email templates securely.
2. Implement the email service such that it is triggered automatically based on the events like rash driving, accidents, fault occurrence, etc found in the vehicle. This should notify the customer of a suitable solution for the problem.
3. These emails will be received by the customers automatically from the AWS Cloud on a day-to-day basis such that the users have a better insight into their vehicle performance.

3. LITERATURE REVIEW.

The authors in [1] explore the concept of serverless cloud computing and services like AWS Lambda along with other existing AWS services. In this paper, Serverless Microservice architecture was implemented to build a serverless Chat Application that supports scalability without the addition of new servers. Their study provides an analysis to understand the value and significance of serverless computing and 'Function as a Service' for hosting dynamic apps with a lot of user interaction.
In [2] the authors have designed a Serverless messenger chatbot application. The chatbot uses a Serverless Microservice architecture that was implemented using Amazon Web Services (AWS) including API Gateway, Lambda, DynamoDB, SNS, and CloudWatch. This research examines the architecture, which consists of several components calling one another with low latencies. One of the concerns was that if each module contributed n milliseconds towards the query time, the chatbot would become too slow to react to users.

In [3] authors examine the boundaries, vulnerabilities, privacy, and varying legislations of cloud-based emails, as well as how to mitigate them in order to provide users and organizations with reliable and secure cloud-based email services. Intelligent Cloud Based Email Encryption and Decryption System (ICLEEDS) is a new framework proposed to improve the security of cloud-based email messages. The idea is to encrypt the content of email messages before they are sent from users' mailboxes. This intelligent machine learning encryption system protects users from email eavesdropping, re-construction, phishing emails, relaying of previous messages, spoofing, and eavesdropping while also providing a high level of privacy.

The authors of [4] gave an examination of many functionalities, primarily supplied by cloud storage providers, that could be misused by attackers if the provided email address is not verified, as well as a generic approach to espionage, malware distribution, and incrimination operations. Because of unverified email addresses, security for cloud storage services is a major concern.

The authors of the [5] described and compared the capabilities of AWS, Azure, and Google's cloud computing to assist enterprises and customers in selecting the appropriate features that will meet their long-term needs. Their study provides a thorough review of some of the tools offered by AWS, Azure, and Google's cloud computing such as performance, storage space management, and compute offered which are the top three market leaders in cloud computing technology.

The authors of the study [6] describe a novel performance-oriented serverless computing platform written in.NET and deployed in Microsoft Azure, with Windows containers serving as function execution environments. They've also provided metrics for evaluating serverless platform execution performance and conducted testing on their prototype, as well as Google Cloud Functions, Azure Functions, AWS Lambda, and IBM's Apache OpenWhisk deployment. At most concurrent levels, the prototype outperforms competing platforms in terms of throughput.

The authors in [7] have examined the performance of many HPC benchmarks on the Azure cloud and AWS platforms, with an emphasis on the compute oriented H16r and c4.8xlarge instance types. These benchmarks measure processing speed, memory bandwidth, and network bandwidth, among other elements of computer system performance. However, the most cost-effective cloud platform for a given use case is determined by the application's processing and communication habits. The AWS c4.8xlarge instance was relatively cheaper of raw computation at the time the tests were done, but Azure's H16r offered cheaper bandwidth, according to this study.

4. METHODOLOGY

The proposed cloud-based email service will be built using Amazon web service, spring boot framework, and Programming Language Java. Figure 4.1 shows the methodology.

Figure 4.1 Messenger Service design

Approach:

The publisher will send a message scheme over SQS for an email to be sent. The templates inside the SES bucket will be transferred through Lambda to the S3 payload bucket. The specific template will be formatted with the message. Then the SQS event will trigger Lambda. The lambda will start fetching messages from SQS and start formatting inside the Lambda, once the format is done it will start publishing to SES. Further, SES will transfer the email to the cloud as well as CloudWatch and SNS to keep track of email sends, deliveries, bounces, and complaints.

5.SYSTEM ARCHITECTURE

The architecture diagram comprises of a CICD pipeline that deploys the code and lambda modules for uploading actions and alerts script components into the cloud. The strategy employed uses a separate state file that is maintained in the cloud which keeps atrack of changes and removals/additions made to the module components across each cloud. This is implemented using a storage account or an amazon EC2 instance. The

architecture employed is depicted in figure 5.1. The resource governance makes sure of the boundaries or constraints within which the messenger service package shall operate.

Figure 5.1 Messenger Service Architecture

5.1 Module Description

5.1.1 Simple Notification Service: A fully managed messaging service for application-to-application (A2A) and application-to-person (A2P) communication, Amazon Simple Notification Service (Amazon SNS) is available.

Between distributed systems, microservices, and event-driven serverless applications, the A2A pub/sub feature enables topics for high-throughput, push-based, many-to-many messaging. Your publisher systems can fanout messages to numerous subscriber systems, such as HTTPS endpoints, Amazon SQS queues, AWS Lambda functions, and Amazon Kinesis Data Firehose, for parallel processing using Amazon SNS topics.

5.1.2 S3 Buckets: An object storage service called Amazon Simple Storage Service (Amazon S3) provides performance, security, and scalability that are unmatched in the market. Any quantity of data can be stored and protected by customers of all sizes and sectors for practically any use case, including data lakes, cloud-native applications, and mobile apps. You may reduce expenses, organise data, and set up precise access controls to satisfy unique business, organisational, and compliance requirements using cost-effective storage classes and simple administration tools.

5.1.3 SES Configuration: AWS Simple Email Service (SES) will need some configuration before we can use it to send emails of Production workloads. The SES by default is tagged as Sandbox and is capped at 200 emails per day limit.
The below configurations will be done in SES for it to be production ready.

Sender Email Verification:
Amazon SES requires to verify the email before sending emails. In current email sending flows, Api.info.cs@sender.com is used. This email address must be Verified before using them as Sender Email address.

5.1.4 Configuration Sets: These are required to track the email sends, deliveries, bounces etc. We will have 2 different configuration sets with the success written to the cloud watch and failures (Bounces, Complaints) published to SNS Topic, for further analysis. The publishing component must analyse the hard bounces as SES adds those recipients to a Suppression List and will not send emails to the Suppressed email ID's in subsequent tries.

5.2 Estimating the Model Parameters
The cost optimization package used here is deployed across the cloud platform Amazon Web Services. While the deployment strategy involves using Lambda Architecture, CICD Pipeline, SpringBoot Framework and JavaScript software on windows, the sole working of the messenger service is restricted within the AWS cloud alone. The Lambda is the approach of deploying the runbooks and other functions into the cloud and the software is helpful as it helps us achieve the building of required infrastructure as code templates needed to provision and destroy runbooks and lambda functions on the cloud platforms.

The messenger service follows a modular approach, and the code is built following the rules of Object-oriented programming. The package is driven by a main program which is the master file, and this takes care of the execution including deployment of the entire messenger service structure. The coding conventions follow an all-small case definition when it comes to declaring parameters and the entire execution, deployment and monitoring is carried out using CloudWatch in AWS platform. The codes are built with references to each other inside the modules as well as at the level of main program execution. The variables are all declared in small case letters to avoid confusion and the values as well as keys pertaining to tags attached to the resources.

The flexibility provided by the approach is important to support robustness in the application. Once the optimization package is deployed in the cloud, the results of the optimization can be seen on the cost analysis section and the insights thus obtained are shared to the client for useful interpretation and further enhancement to encourage the most efficient levels of optimization services.

6. MODEL VALIDATION

In this section the validation of the model is explained.

6.1 DataDog Validation
The testing involves including all the runbooks at once, creating the resources in the cloud and executing the messenger service all together at once. This is important as it gives the insights into the system integration testing and is generally carried out using automation tools like datadog. This is done in this manner as there are usually more than fifty different modules to be tested simultaneously. The testing framework is also a part of the messenger service, and it is made available as a self-diagnosis tool post provisioning.

Figure 6.1 shows where 4k messages are sent into the messenger queue to test.

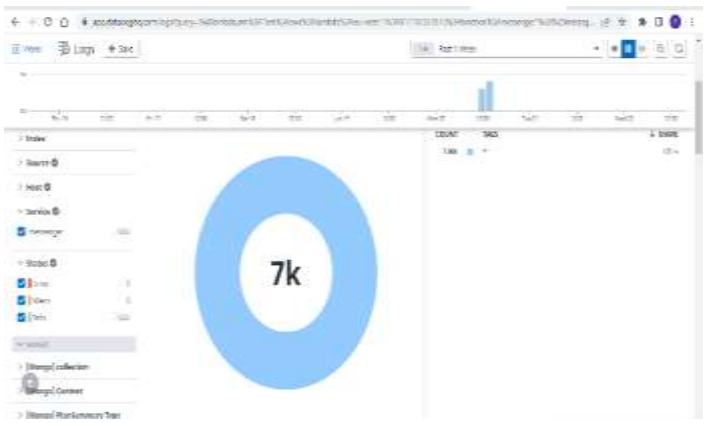

Figure 6.1 Load Test Result in Datadog

7.EXPERIMENTAL RESULTS/ SNAPSHOTS

7.2.1 Templated Email without Attachment
When the payload received does not have any attachments, the email body is formed using the json and email is generated and sent to the recipients in figure 7.2.1.

Figure 7.2.1 Templated Email without attachment

7.2.2 Templated Email with Attachment having link to S3: When the payload received has attachment, check the type of the attachment. If the Attachment type is s3link, then based on the details in json, fetch the attachment from s3 bucket and attach that as part of the email generated as shown in figure 7.2.2.

Figure 7.2.2 Templated Email with attachment

7.2.3 Templated Email with Attachment having small payload: When the payload is less than 256kb in size, the whole payload is received as part of the json. The details required for constructing the email will be present in the json received as shown in figure 7.2.3.

Figure 7.2.3 Templated Email with small payload attachment

7.2.4 Templated Email with Attachment having large payload: When the payload is too large to be sent via SQS, i.e, if the size is greater than 256KB, then the "isPayloadLarge" attribute is set to true. And the Json received will contain the details of the s3 bucket where the Json will be placed. First, the Json has to be fetched from s3 bucket and email has to be generated based on the payload as shown in figure 7.2.4.

Figure 7.2.4 Templated Email with large payload attachment

7.2.5 Templated Email with Multiple Attachments: Email can contain more than one attachment. To support this, the payload json is of the structure where multiple attachments can be added. Each attachment can be of type S3 link or base64encodedstring. If the type if of S3 link, the attachment is fetched from S3 bucket and if the type is of base64encodedstring, the file is generated using the base64encodedstring value. These Attachments are then added as part of the generated email as shown in figure 7.2.5.

Figure 7.2.5 Templated Email with multiple attachment

8. CONCLUSION AND FUTURE WORK

Cloud computing is the latest technological advancement that can make a huge impact on the world. This brings multiple benefits to users and businesses. For example, one of the benefits it offers to businesses is that they can reduce operational costs by reducing investment in management and software upgrades and allowing them to focus more on their core business.

Cloud-based email services are based on a serverless microservices architecture built on the Springboot Framework and Amazon Web Services (AWS). This includes AWS Lambda, Simple Queue Service (SQS), Simple Notification Service (SQS), Simple Email Service (SES), and more. Simple Storage Service (S3), Elastic Compute Cloud (EC2), and CloudWatch.
This project aims to provide an email service that uses contextual email templates to send large numbers of emails in a short amount of time, and configures email broadcasts, fault management, and complaint monitoring services. AWS's flexible IP provisioning and email authentication options help improve deliverability and protect the sender's reputation, and delivery analysis monitors the impact of each email.

9. REFERENCES

[1] Brijesh Choudhary, Aditya Gutte, Ankit Dani, Shilpa Sonawani, “Case Study: Use of AWS Lambda for Building a Serverless Chat Application”, Proceeding of International Conference on Computational Science and Applications,2020, pp.237-244

[2] Dr. Manish Saraswat1 and Dr. R.C. Tripathi, "Cloud Computing: Comparison and Analysis of Cloud Service Providers—AWS, Microsoft and Google", 9th International Conference on System Modeling & Advancement in Research Trends,2020.

[3] Niko Mäkitalo, Tommi Mikkonen,"Case Study: Building a Serverless Messenger Chatbot", Current Trends in Web Engineering,2018, pp.75-86.

[4] Godson Michael D'silva, Sanket Thakare , Sharddha More, and Jeril Kuriakose," Real-World Smart Chatbot for Customer Care using a Software as a Service (SaaS) Architecture", International conference on I-SMAC (IoT in Social, Mobile, Analytics, and Cloud),2017 IEEE.

[5] Garrett McGrath, Paul R.Brenner," Serverless Computing: Design, Implementation and performance", IEEE 37th International Conference on Distributed Computing Systems Workshops (ICDCSW),2017.

[6] C. Kotas, T. Naughton, and N. Imam, "A comparison of Amazon Web Services and Microsoft Azure cloud platforms for high-performance computing," 2018 International Conference on Consumer Electronics, Las Vegas, NV, 2018, pp. 1-4.

[7] Alalawi, A. Mohsin and A. Jassim, "A survey for AWS cloud development tools and services," 3rd Smart Cities Symposium (SCS 2020), 2020, pp. 17-23.

[8] T. Hahn, T. Kunz, M. Schneider and S. Vowé, "Vulnerabilities through Usability Pitfalls in Cloud Services: Security Problems due to Unverified Email Addresses," 2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications, 2012, pp. 850-856.

[9] T. Ayodele and D. Adeegbe, "Cloud based emails boundaries and vulnerabilities," 2013 Science and Information Conference, 2013, pp. 912-914.

[10] M. Villamizar et al., "Infrastructure Cost Comparison of Running Web Applications in the Cloud Using AWS Lambda and Monolithic and Microservice Architectures," 2016 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid).

[11] "Amazon Serverless computing or Google Function as a Service (FaaS) vs Microservices and Container Technologies," https://www.yenlo.com/blog/ amazon-serverless-computing-or-google-function-as-a-service--faas-vs-microservices-and-container-technologies [Online; accessed 1-Nov-2016].

[12] S. Hendrickson, S. Sturdevant, T. Harter, V. Venkataramani, A. C. Arpaci-Dusseau, and R. H. Arpaci-Dusseau, "Serverless computation with openlambda," in Proceedings of the 8th USENIX Conference on Hot Topics in Cloud Computing, ser. HotCloud'16, 2016.

[13] "AWS Lambda," https://aws.amazon.com/lambda/, [Online; accessed 15-Nov-2016].

[14] "AWS Lambda Pricing," https://aws.amazon.com/lambda/pricing/, [Online; accessed 14-September-2017]

[15] Amazon Web Services, "AWS DynamoDB Streams Best Practices," "http://docs.aws.amazon.com/amazondynamodb/latest/developer" "guide/Streams.Lambda.BestPracticesWithDynamoDB.html" Accessed 18-Sep-2017

[16] "Serverless Framework," https://serverless.com/, [Online; accessed 11-September-2017].

[17] "Docker-lambda," https://github.com/lambci/docker-lambda, [Online; accessed 11-September-2017].

[18] Amazon Web Services, "AWS X-Ray Segment Documents," "http://docs.aws.amazon.com/xray/latest/devguide/xray-apisegmentdocuments.html" Accessed 18-Sep-2017

[19] Wei-Tsung Lin, Chandra Krintz, Rich Wolski, and Michael Zhang," Tracking Causal Order in AWS Lambda Applications" [2018 IEEE]

[20] "AWS SDK for Python (Boto3)," https://aws.amazon.com/sdk-for-python/, [Online; accessed 14-September-2017

