International Journal of Engineering & Technology, 7 (2.21) (2018) 420-422

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Building micro service for user engagement

Devansh Sharma[1*], R. Anandan[2], A. Manikandan[3], Kumar Narayanan[4], C. Swaraj Paul[5]

[1] Department of Computer Science Engineering, Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai, India.
[2] Department of Computer Science Engineering, Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai, India.
[3]Department of Computer Science Engineering, Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai, India.
[4]Department of Computer Science Engineering, Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai, India.
[5]Department of Computer Science Engineering, Vels Institute of Science, Technology and Advanced Studies (VISTAS), Pallavaram, Chennai, India.
**Corresponding author E-mail: 75devanshsharma@gmail.com*

Abstract

The emails that are being sent today are done manually through the use of email providers like Gmail, Yahoo, Hotmail etc. with a copy paste mechanism of the same message. This requires more human resource and time. The system now being developed will be able to send the emails to thousands number of user without any copy-paste mechanism in very less time and with great accuracy handling all errors, if occurs, through the use of Amazon web services & Google Cloud services. Thus, the messages received by every user will be unique instead of the same message being sent to all. The email micro service will be responsible for handling out all the essential needs, verification & validation, automation, clean code and follows a proper SDLC as per current industry standards. The track ability feature will also be provided to track all the emails that are being sent to the users. Thus, the automation will be done and will save a lot of human resource and time.

Keywords: *Micro Service, Email Micro Service, Automation, Fast & Secured Email.*

1. Introduction

1.1. Current scenario

The application built are first built with small size. During the initial phase of developing application, the story is less. But slowly, the lines of code in the application increases over time and the application after some time will have grown into monstrous monolith.

- The development organization start getting problem once the application has brown large. The agile development and delivery will flounder. Application becomes complex to handle. It becomes difficult for developer to understand the whole application.
- As a result, it becomes difficult to fix bugs and also takes a lot of time. What's more, this tends to be a downwards spiral. Changes cannot be made correctly if the codebase is difficult to understand.
- It becomes obstacle to continuous development. Today, the state of the art for SaaS applications is to push changes into production many times a day. It becomes difficult to do with a complex monolith since we have to redeploy the whole application if we want to make change at any one part.
- Reliability is the another problem with monolithic application. A bug in any module, such as a memory leak can disrupt the entire process of application. Having all the instances of application as identical will result into the bug affecting the entire application.
- Micro services[1] Architecture pattern is used by many organizations, such as **Amazon, eBay, and Netflix,** to solve the problem of monolithic architecture. The idea is to split the application into set of smaller, interconnected services.
- A service can have many features or functionality, such as order management, customer management, etc. Each micro service is a mini-application and has its own hexagonal architecture consisting of business logic along with various adapters. So, by using this micro service architecture[1], we can have many services broken into isolated component.

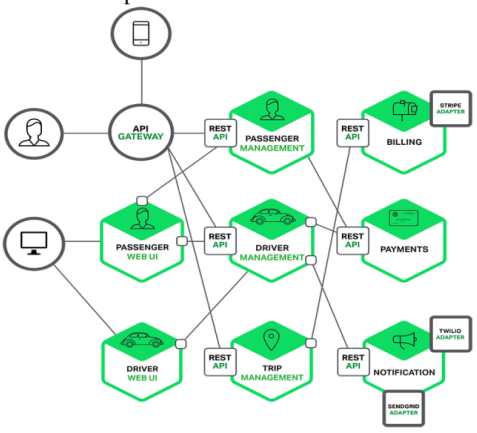
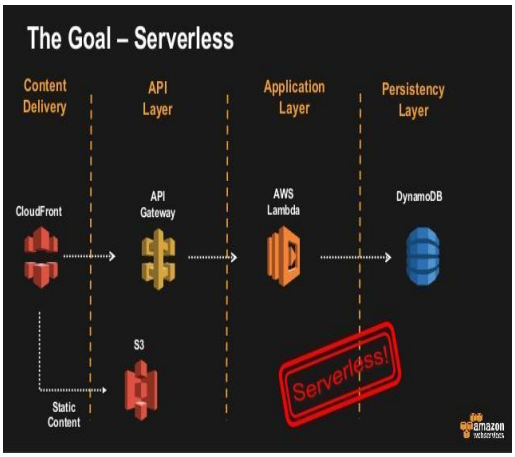


Figure. 1: Micro service example

The micro service architecture of **amazon** is show below:



Limitations of current email services

The email service used today requires manual interaction. By manual interaction, we mean that for sending emails to user, the person has the provide all the details for an email like to Address, subject, message, attachments(optional).
The email micro service used today is **SMTP based.**

Disadvantage

- Using SMTP based email service limits the flexibility of email sending.
- Also, the speed rate of sending email is little more than using API. There is no tracking of email status that could be done.
- It uses http which is not allowed by some firewalls these days.

Let's take a case if we don't use API based email service.
Suppose, if the person wants to send emails with same message to 100 numbers of user then he/she has to copy-paste the message and send it to the recipients. Thus, a lot of time and resource is wasted. Also, when the emails are received, it goes to the particular user who sent it. Thus, the emails received are distributed to many emails an no centralized records are found.
The other way it's done is by saving the template in database like MySQL, Oracle, PostGres etc. But this is not a centralized or conventional way in industry since instances are stored in every system.

2. Proposed system

The email micro service will be responsible for sending/receiving mails without much manual interaction.
The micro service[2] will be integrated and connected with other micro service via APIs (Application Programming Interfaces).
A common template will be stored in Amazon which can be accessed programmatically via Amazon APIs. The emails can be sent to 1000 numbers of user in 3-4 seconds with dynamic or different values for each users.
Thus, each user will be able to receive his/her own copy of email with unique or different message. So, the copy-paste mechanism of sending emails will be completely removed. Email sending and receiving being a separate micro service can be handled and configured independently without affecting the whole system. The emails can be tracked to know whether the user has opened the mail, clicked the link under the email etc.
Thus, time and resources will be saved. Also, infrastructure are not needed to setup the email server or database server for storing and accessing email templates.

2.1. Working of current system

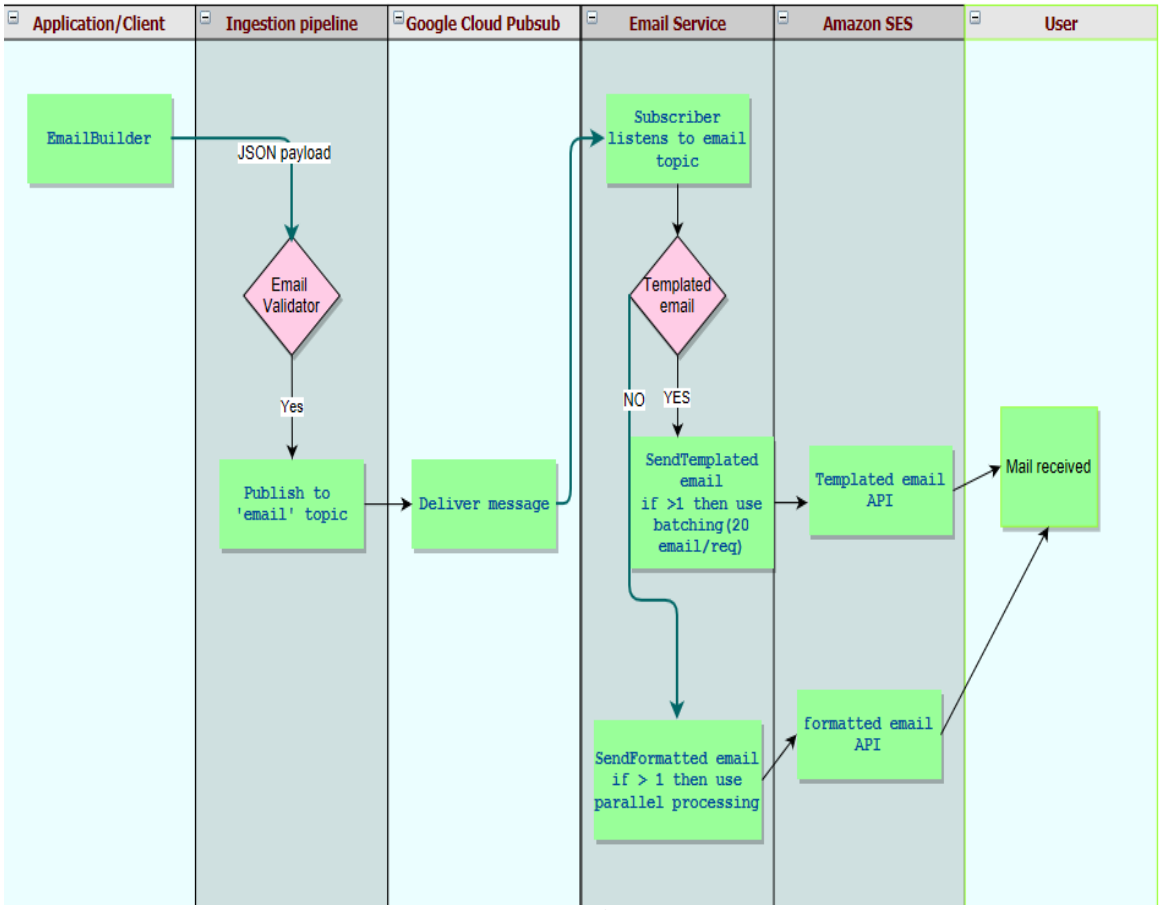


Figure. 2: Architecture of email micro service

1. Email service application

This is the main application class for the email service. This class will be responsible for running the email service application.
This class will call the PubSub Subscriber class for getting the subscriber from the Google Cloud[4] PubSub.

2. PubSub subscriber

This class will be responsible for getting the messages for the subscriber. This class will call the Messsage Receiver Impl class to get the message and will return the subscriber information back to the main application class.

3. Message receiver

This class will be responsible for receiving the message from the Cloud[3] PubSub in the form of JSON. The JSON message that is received will contain the to, from, from Name & template Id information.

4. Pay load

This class will be having getters and setter's method for to, from, from Name & template Id fields. The JSON received will be converted into Java Object for the above fields using this class.

5. Email sender

This is the class for sending the single email as well as bulk emails. This methods for sending email will receive information in form of Pay Load class objects. The class will have the method to count the number of Destination address and then decide whether to single email sending or bulk email sending.

6. Email validation service

This class will be responsible for validating the email format of the recipient and will report error if the format is wrong. It will use apache Email Validator class for validation of the email format.

7. Configuration service

Fetch Data from google data store with kind : credential, type : AwsSes Credential. Store Access Key Id and Secret Access Key into variables in Application class. Access those variables from Email Sender class and use them to build the client.

8. Amazon simple email service client

This class is used to set the AWS[6] Credentials (Access Key and Secret Key) for sending the email through Amazon SES.

3. Conclusion

The email micro service will be able to send 1000 mails with unique message in each of them in 8-10 seconds. This will reduce the time, make the service automated and having the micro service architecture will bring down the complexities and problems faced in monolithic architecture.
Using Rest APIs, the service can be used from anywhere and with any applications.

References

[1] Habl, A, Kipouridis, O & Fottner, J, "Deploying micro services for a cloud-based design of system-of-systems in intralogistics", *IEEE 15th International Conference on Industrial Informatics (INDIN)*, (2017), pp.861-866.

[2] Brüggemann ME, Vallon R, Parlak A & Grechenig T, "Modelling micro services in email-marketing concepts, implementation and experiences", IEEE *9th International Conference on Software Paradigm Trends (ICSOFT-PT)*, (2014), 67-71.

[3] Hyseni LN & Ibrahimi A. Comparison of the cloud computing platforms provided by Amazon and Google. *IEEE Computing Conference*, (2017), pp.236-243.

[4] Albertengo G, Debele FG, Hassan W & Stramandino D, "On the performance of web services and Google cloud messaging", *IEEE International Conference on Computer and Information Technology (CIT)*, (2017), pp.363-367.

[5] Madhumathi R, RadhaKrishnan R, Kumar SS, Abineshkumar K, Karthi M & ManojKrishna M, "Bidding application in Amazon web services for the sales of agricultural products", *IEEE International Conference on Recent Trends in Information Technology (ICRTIT)*, (2016), pp.1-6.

[6] Liao WH, Kuai SC & Leau YR, "Auto-scaling strategy for amazon web services in cloud computing", *IEEE International Conference on Smart City/Social Com/Sustain Com (Smart City)*, (2015), pp.1059-1064.