

Introduction

Messaging has become one of the most essential forms of communication. Sending emails through conventional software such as Gmail is restricted to a few thousand, which is insufficient to meet business requirements. Managing the insights of sent emails is essential for businesses to boost their approach.

Serverless computing refers to any computing platform that obfuscates server usage from developers, runs code on-demand that is automatically scaled, and only bills for the time the code is running. This decreases the amount of time and expertise necessary to deploy software while providing significant cost savings compared to hosting applications on dedicated hardware. Numerous cloud systems, like Amazon Web Services, Microsoft Azure, and Google Cloud Platform, have adapted to this way of thinking, making it readily available.

Messaging is a method of using technology to bring people and ideas “together” despite of the geographical barriers. The technology has been available for years but the acceptance it was quit recent. Our project is an example of a multiple client chat server. It is made up of 2 applications the client application, which runs on the user’s Pc and server application, which runs on the any Pc on the network. To start chatting client should get connected to server. We will focus on TCP and UDP socket connections which are a fundamental part of socket programming.

Keywords: sockets, client-server, Java network programming-socket functions, Multicasting etc.

Sends And Receives Message from Client

Socket programming is a way of connecting two nodes on a network to communicate with each other. One socket(node) listens on a particular port at an IP, while the other socket reaches out to the other to form a connection. The server forms the listener socket while the client reaches out to the server.

Socket programming is started by importing the socket library and making a simple socket.

import socket

s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

The first parameter is `AF_INET` and the second one is `SOCK_STREAM`. `AF_INET` refers to the address family `ipv4`. The `SOCK_STREAM` means connection-oriented TCP protocol.

Note: For more information, refer to [Socket Programming in Python](#)

A Server is a program that provides service to other computers on the network or Internet. Similarly, a client is a program that receives services from the server. When a server wants to communicate with a client, there is a need for a socket. A socket is a point of connection between the server and the client.

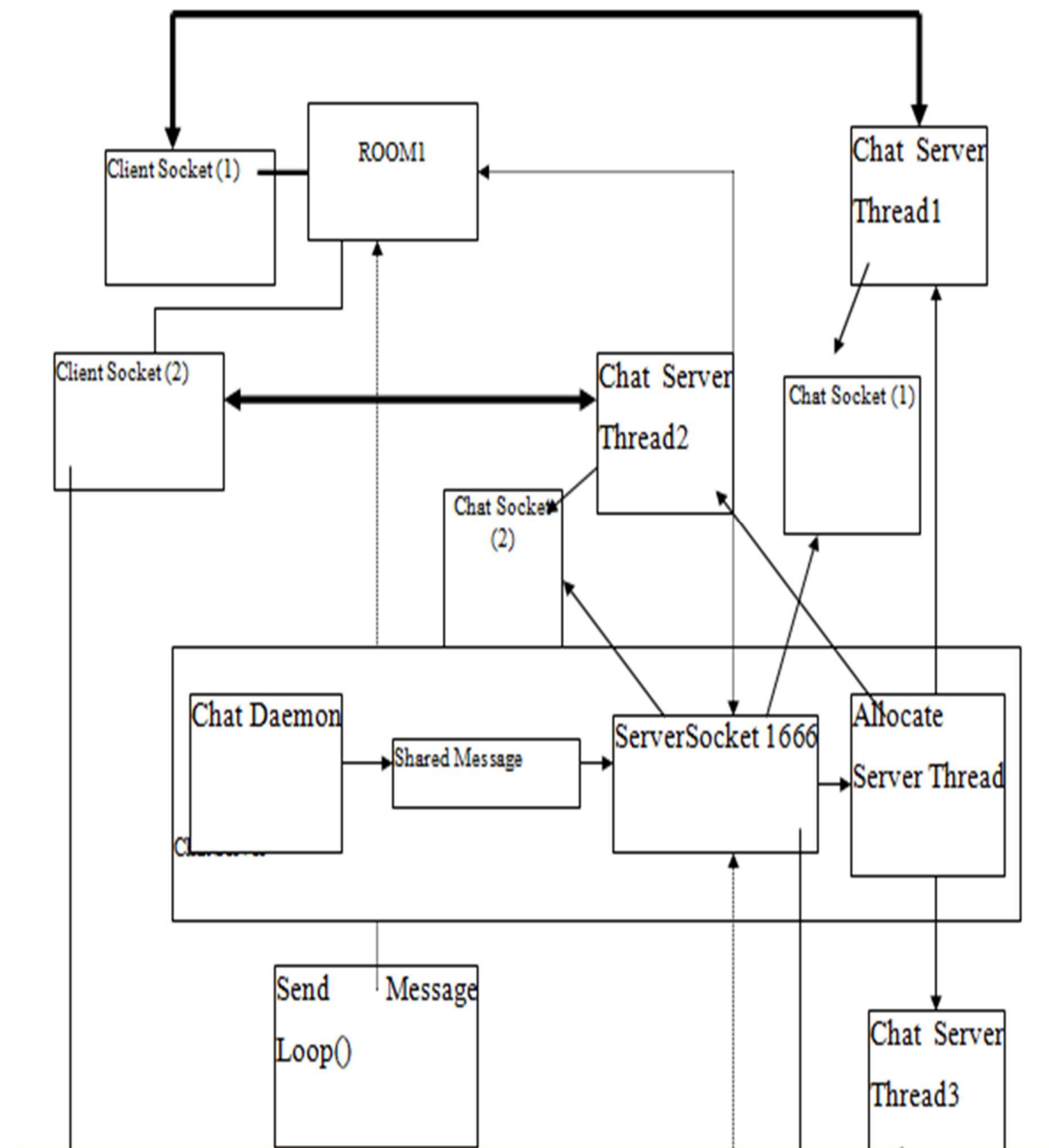
Challenges such as being able to track down bounced emails and compile statistics for emails sent. The prototype is constructed utilizing the React framework and underlying AWS services. These tightly connected services provide the most effective method to replicate the architecture and assist in sending tailored emails to each recipient based on email templates, as well as storing bounce emails.

With serverless technology, the cloud provider abstracts server management, supplying servers with fine-grained granularity, on-demand, and a pay-per-use model.

This report presents an architecture for sending and managing emails via a serverless AWS application. The study describes the precision and dependability of the design, as well as future considerations for this architecture.

Modules of the Project

DATA FLOW DIAGRAM



1. Amazon SES(Simple Email Service)
2. AWS Lambda
3. Serverless Computing
4. Sign Up
5. Login
6. Users or groups
7. Chat window
8. Settings
9. Software Required
10. Hardware Required
 - a. Hard Disk – 2 GB.
 - b. RAM – 1 GB.
 - c. Processor – Dual Core or Above.
 - d. Mouse.
 - e. Keyboard.
 - f. Monitor.
 - g. Printer.
11. Code
12. Output

Module Description and Screenshots

Amazon SES(Simple Email Service)

Amazon Simple Email Service (SES) enables you to confidently reach customers without a local Simple Mail Transfer Protocol (SMTP) system.

Amazon SES is a cloud-based email service that can be integrated with any application for sending mass emails. Whether you are sending transactional or promotional emails, you just pay for what you send. Additionally, Amazon SES supports a range of configurations, including dedicated, shared, and owned IP addresses. Reports on sender statistics and a deliverability dashboard allow organizations to maximize the effectiveness of every email.

Amazon SES delivers massive email campaigns using a service that sends hundreds of billions of emails per year. Using secure email authentication, you can reach customers' inboxes as a reliable sender. Transparent pricing built for bulk email will increase your bottom line.



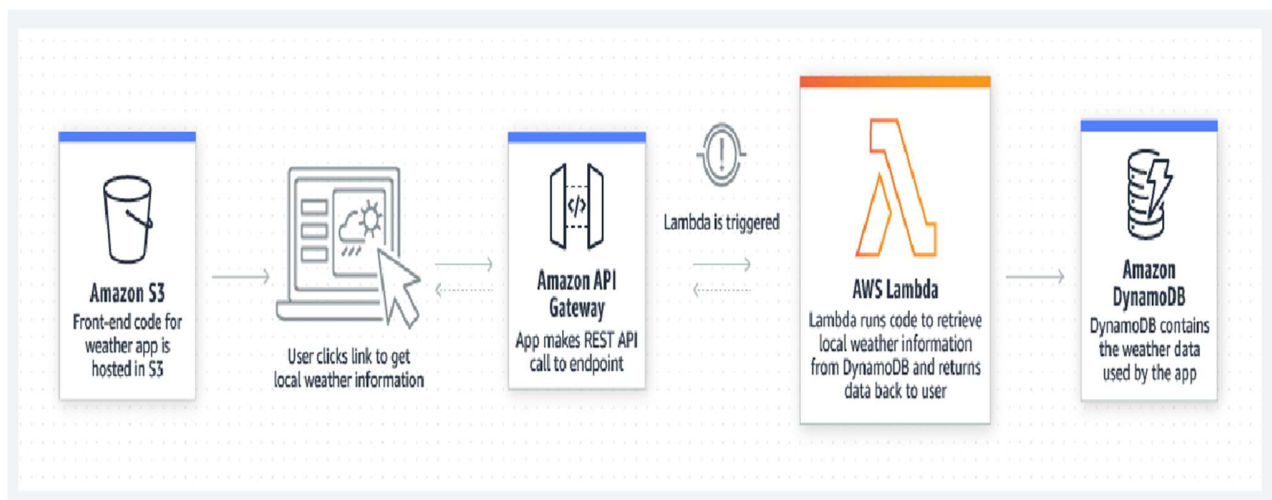
AWS Lambda

AWS Lambda is a serverless, event-driven compute solution that allows you to execute code for almost any application or backend service without creating or managing servers. Lambda can be triggered by more than 200 AWS services and software-as-a-service (SaaS) applications, and you only pay for what you use.

AWS Lambda supports executing code without the need to provision or manage infrastructure. Simply compose and upload code as a.zip or container image.

It responds automatically to code execution requests at any scale, ranging from a few occurrences per day to hundreds of thousands per second.

Use Amazon Simple Storage Service (Amazon S3) to initiate AWS Lambda data processing in real time following an upload, or connect to an existing Amazon EFS file system to facilitate massively parallel shared access for large-scale file processing.



Serverless Computing

Serverless computing is a means of offering as-needed backend services. A corporation that receives backend services from a serverless provider is not charged for a fixed amount of bandwidth or number of servers, but rather based on consumption.

AWS provides tools for executing code, maintaining data, and integrating applications without requiring the administration of servers. Serverless solutions include automatic scalability, built-in high availability, and a pay-as-you-go payment approach to enhance agility and save expenses. These solutions also eliminate infrastructure administration responsibilities like as capacity provisioning and patching, allowing you to focus on building customer-focused applications. AWS Lambda, an event-driven compute solution natively linked with over 200 AWS services and software as a service (SaaS) applications, is the foundation for serverless applications.

The benefits of serverless computing are:

Reduce expenses

Simplified scalability

streamlined back-end code

Cost benefits of Serverless

Existing System

When internet was not there, people were not able to communicate as a group. Text messages were the most common mechanism but it did not provide the facility of multiple communications. There are many occasions when communication has to happen in a group only like an official offline meeting or a fun conversation among a group of friends. Moreover, the mechanism of text messages was not cost effective and was relatively slow.

Proposed System

Proposed system although provides multiple conversations which have made the life of so many people easier and interesting. People are more than happy to get such a facility. The Client Server Messenger is a socket application which is multi-threaded in nature, it means multiple users can access the single server application. This is best used in professional environments. This system is very fast and the communication happens very smoothly.

Sign up- For using the instant messenger, each user must have an account on it. This is the very first step. The user will have to provide all necessary details like name, age, picture, date of birth etc., along with the most important details, username and password. If all the details are correct, the user is successfully registered.

Login –Every time the user wants to access the instant messenger, he will have to provide valid login id and password. Through this login id and password, users will be redirected to their own home page.

Users or groups – This module will have a list of users or groups which a user can add. Before beginning the chatting with one or more users, it is important to add the particular user. In this list, it will be visible that a user is online or offline.

Chat window– In this module, the conversations would happen and all the users in the group will be able to view the conversation and participate in it.

Settings– In this module, the user can modify his profile details like picture, password and other privacy settings.

Software Requirements

1. TCP/IP network layer
2. Jdk 1.5
3. Netbeans 5.0

Hardware Requirements

1. Hard Disk – 2 GB.
2. RAM – 1 GB.
3. Processor – Dual Core or Above.
4. Mouse.
5. Keyboard.
6. Monitor.
7. Printer.

MAIN CODE:

TCP/IP server program that sends message to the client.:

```
import socket

# take the server name and port name
host = 'local host'
port = 5000

# create a socket at server side
# using TCP / IP protocol
s = socket.socket(socket.AF_INET,
                  socket.SOCK_STREAM)

# bind the socket with server
# and port number
s.bind(("", port))

# allow maximum 1 connection to
# the socket
s.listen(1)

# wait till a client accept
# connection
c, addr = s.accept()

# display client address
print("CONNECTION FROM:", str(addr))

# send message to the client after
# encoding into binary string
c.send(b"HELLO, How are you ? \Welcome to Akash hacking World")
msg = "Bye....."
```

```
c.send(msg.encode())

# disconnect the server
c.close()
```

TCP/IP server program that receive message from server.:

```
import socket

# take the server name and port name

host = 'local host'
port = 5000

# create a socket at client side
# using TCP / IP protocol
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

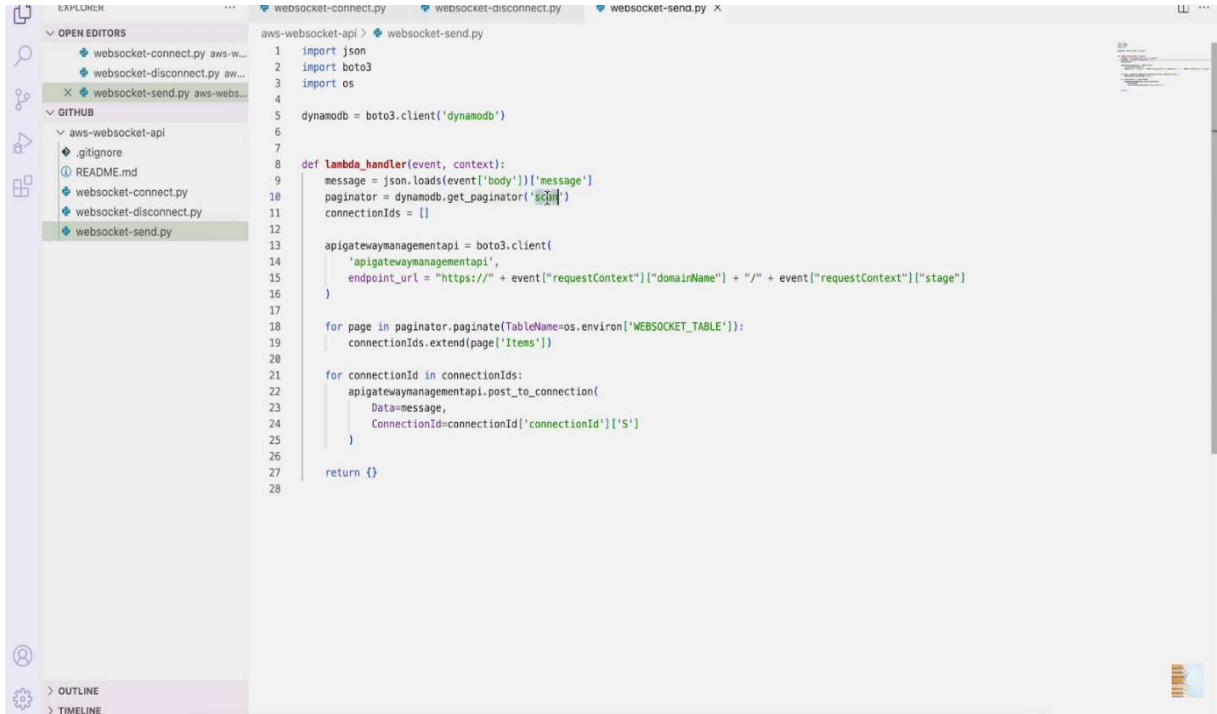
# connect it to server and port
# number on local computer.
s.connect(('127.0.0.1', port))

# receive message string from
# server, at a time 1024 B
msg = s.recv(1024)

# repeat as long as message
# string are not empty
while msg:
    print('Received:' + msg.decode())
    msg = s.recv(1024)

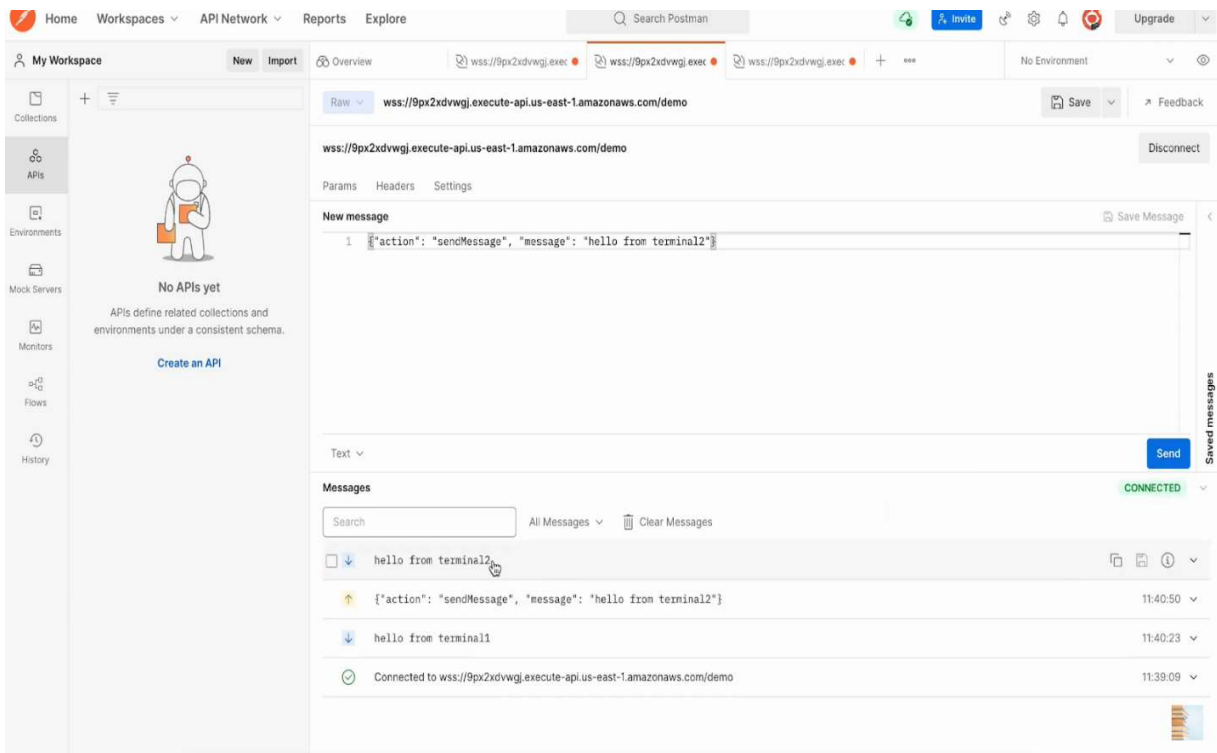
# disconnect the client
s.close()
```

Output



The screenshot shows a code editor with three files open: `websocket-connect.py`, `websocket-disconnect.py`, and `websocket-send.py`. The `websocket-send.py` file is the active one, showing the following Python code:

```
1 import json
2 import boto3
3 import os
4
5 dynamodb = boto3.client('dynamodb')
6
7
8 def lambda_handler(event, context):
9     message = json.loads(event['body'])['message']
10    paginator = dynamodb.get_paginator('scan')
11    connectionIds = []
12
13    apigatewaymanagementapi = boto3.client(
14        'apigatewaymanagementapi',
15        endpoint_url = "https://" + event["requestContext"]["domainName"] + "/" + event["requestContext"]["stage"]
16    )
17
18    for page in paginator.paginate(Table=os.environ['WEBSOCKET_TABLE']):
19        connectionIds.extend(page['Items'])
20
21    for connectionId in connectionIds:
22        apigatewaymanagementapi.post_to_connection(
23            Data=message,
24            ConnectionId=connectionId['connectionId']['S']
25        )
26
27    return {}
28
```



← → ↻

us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#item-explorer?initialTagKey=&table=websocket-connections

☆ Incognito

aws

Services

Search for services, features, blogs, docs, and more

[Option+S]

N. Virginia

listentolearnUser @ 7622-4479-1499

DynamoDB

×

Dashboard

Tables

Update settings

Explore items

PartiQL editor New

Backups

Exports to S3

Reserved capacity

▼ DAX

Clusters

Subnet groups

Parameter groups

Events

Tell us what you think

Return to the previous console experience

Density settings

The new DynamoDB console is now complete, and becomes your default experience

Following the preview phase in which we analyzed and incorporated your feedback, we have completed the new DynamoDB console, making it even easier for you to manage your data and resources. Let us know what you think. You can still choose to return to the previous console from the navigation pane.

×

ⓘ

DynamoDB > Items > websocket-connections

Tables (1)

×

Any table tag

▼

Find tables by table name

Q

< 1 >

ⓘ

websocket-connections

ⓘ

websocket-connections

Autopreview

ⓘ

Actions

▼

Create item

Update table settings

▶ Scan/Query items

Expand to query or scan items.

Items returned (2)

< 1 >

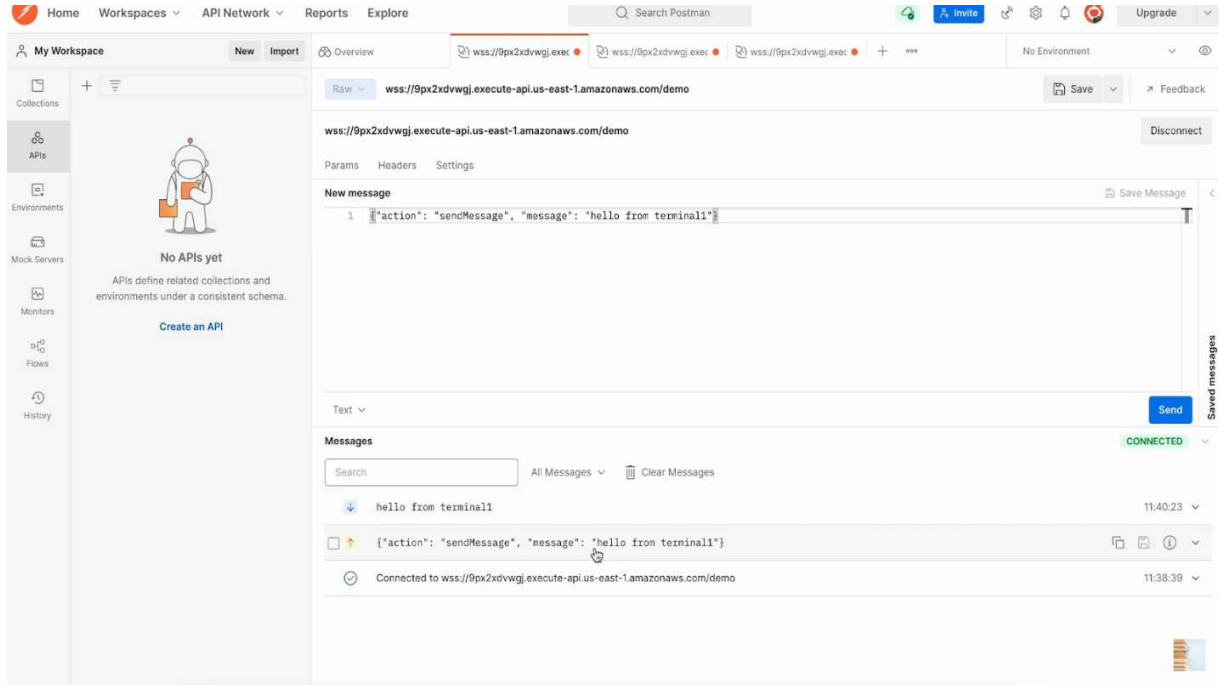
ⓘ

ⓘ

<input type="checkbox"/>	connectionId	▼
<input type="checkbox"/>	TMVnTcCbIAMCLGA=	
<input type="checkbox"/>	TMVsNfwFoAMCFkg=	

17

Inferences from Result:



- This project makes use of the fundamental capabilities offered by Amazon Web Services (AWS), AWS Identity and Management Access (IAM), and AWS Lambda (based on Serverless Architecture).
- A Python script that stores the message content in the form of an HTML template is executed by an AWS Lambda function after the function has been built.
- This AWS Lambda function has an IAM Role linked to it, which grants it full authorization to access AWS.

Conclusion

With Serverless Computing Architecture and Amazon Web Services, We Developed network applications in Python by using sockets, threads, and Web services.

This software is portable, efficient, and easily maintainable for large number of clients. Our developed web-based chatting software is unique in its features and more importantly easily customizable.

This Model enables us to design a model with variable pricing that is scalable, hence reducing our overall expenses.

By reducing network latency, this architecture also enables us to reach clients from various locations of the world faster.

References

Bazaru, Priyatham. (2021). Serverless Architecture For Bulk Email Management. 10.14293/S2199-1006.1.SOR-.PPTDBKS.v1.

S. Eismann, J. Scheuner, E. Van Eyk, M. Schwinger, J. Grohmann, N. Herbst, C. L. Abad, and A. Iosup, “Serverless applications: Why, when, and how?” IEEE Software, vol. 38, no. 1, pp. 32–39, 2020

P. Castro, V. Ishakian, V. Muthusamy, and A. Slominski, “The Rise of Serverless Computing”, Communications of the ACM, vol. 62, no. 12, pp. 44–54, Nov. 2019

Aws lambda. [Online]. Available: <https://aws.amazon.com/lambda/>

Amazon simple email service. [Online]. Available: <https://aws.amazon.com/ses/>

Amazon simple notification service. [Online]. Available: <https://aws.amazon.com/sns/>

Thank You