CSCI4145 Cloud Computing

Term Assignment

Final Report

Name: Anantha Suraj Patnaikuni

Project: AWS-Powered Serverless Encryption and Decryption for messages

Banner: B00845171

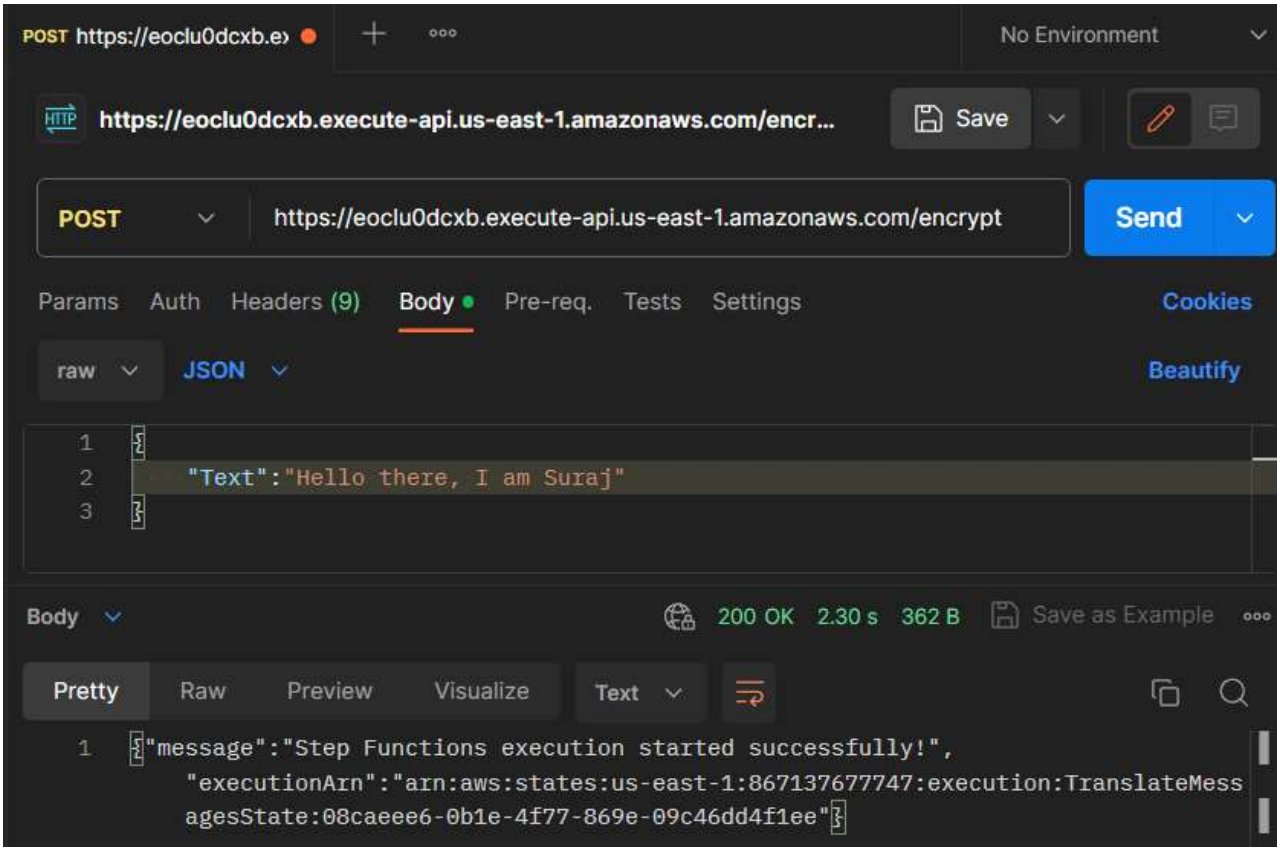
Email: an328201@dal.ca

What you built and what it is supposed to do

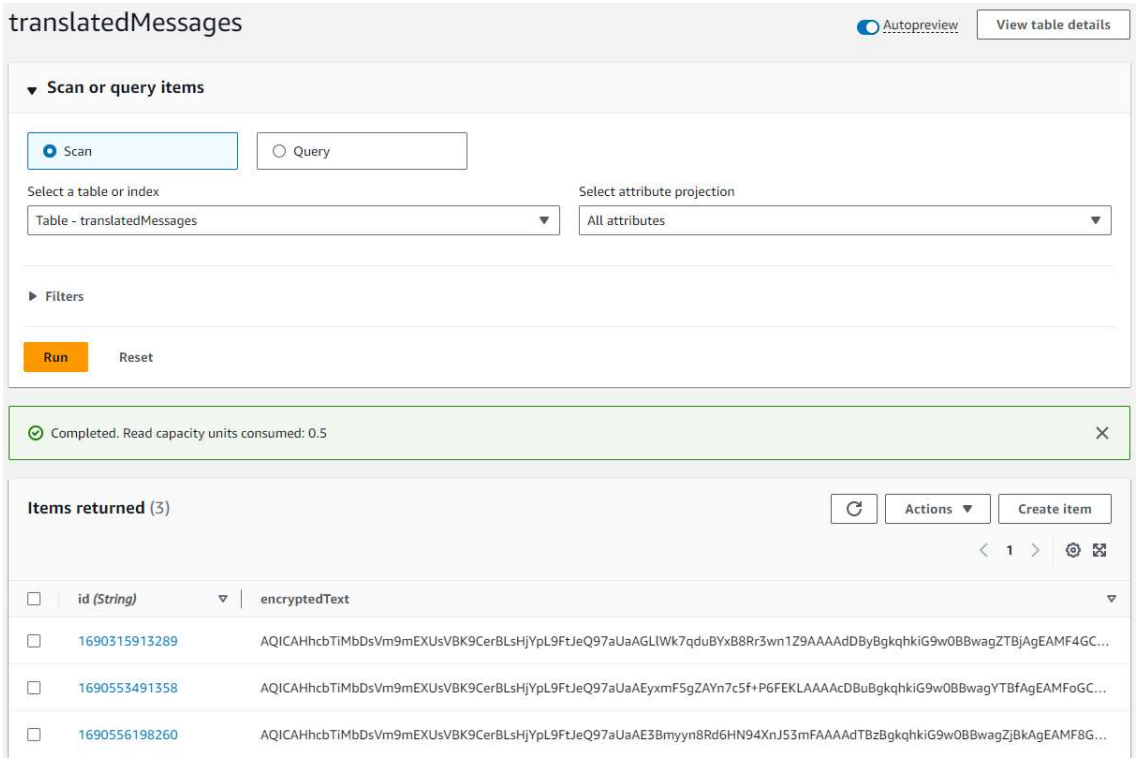
I have developed a serverless application that leverages various cloud services to encrypt and decrypt text messages in a secure and scalable way. In this architecture, I have utilized several AWS services to ensure a robust and reliable system. I used AWS services like Lambda functions for processing necessary actions, AWS DynamoDB for data storage, AWS API gateway for handling API request, AWS Step Functions to create a pathway for flow of operations, AWS Key Management Service for managing the cryptographic keys, and AWS Simple Notification Service (SNS) for getting notified on updates. Users can be anyone who wants to encrypt their text message and keep it safely in email.

The project's functionality involves four main lambda functions.

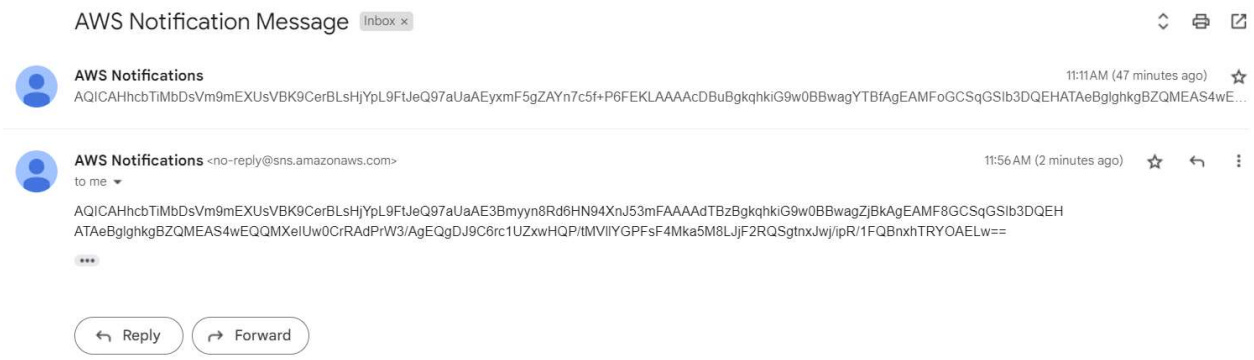
1. step-1EncryptMessage: This function is responsible for encrypting text messages received through the Postman when endpoint is invoked.

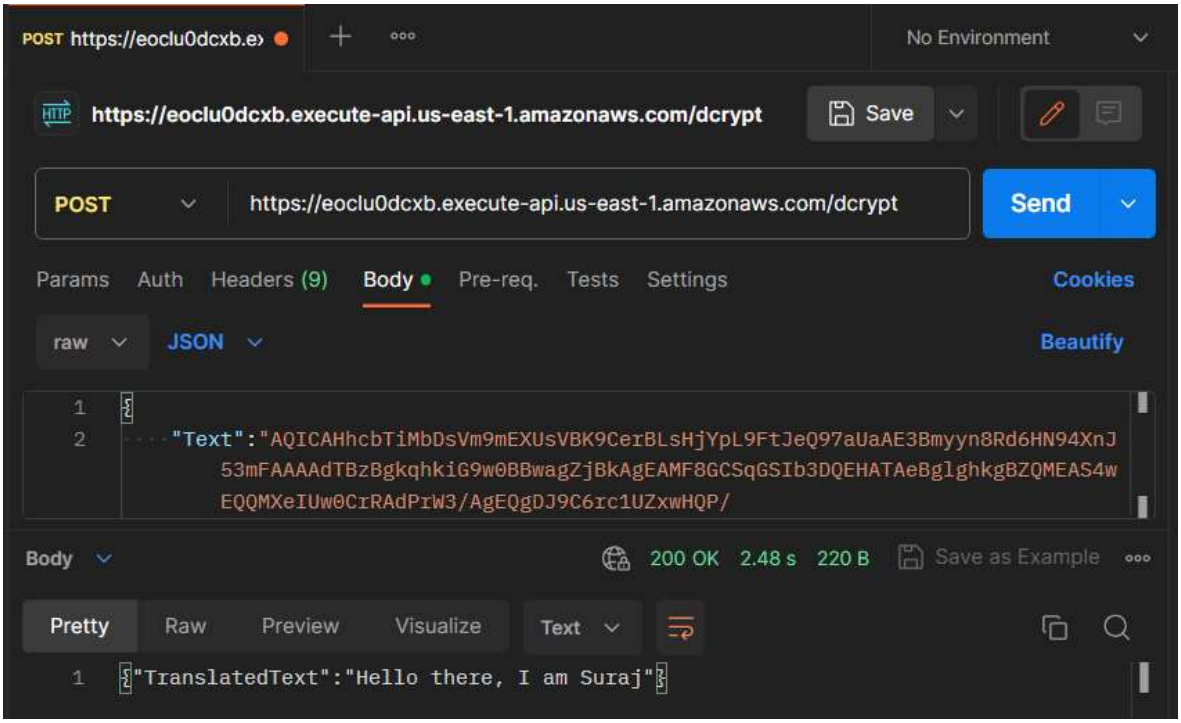


2. step-2-save-to-dynamodb: It stores encrypted text message in DynamoDB along with a unique identifier.

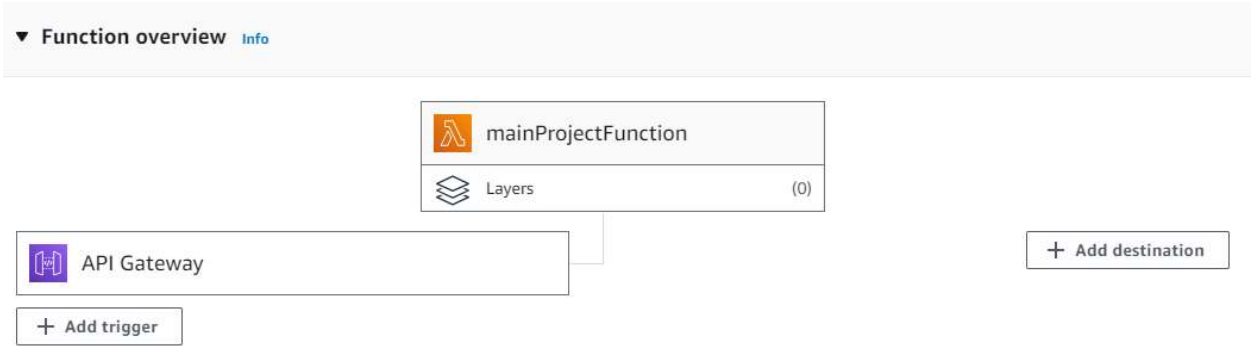


3. DcryptMessage: An email is sent to subscribed users with encrypted message. This encrypted message can be decrypted when we invoke decrypt route in API using Postman. It then shows the actual meaning of the sentence.





4. mainProjectFunction: This Lambda function acts as the orchestrator, validating the flow of actions and providing the success or error messages based on the execution of the other functions.



Overall, the process begins when a user sends a message through API gateway endpoint (encrypt). The API gateway then triggers a step function flow through the main function (mainProjectFunction) located in Lambda. Within the AWS Step Function, a lambda function is invoked (step1) to process/encrypt the incoming message. Thereafter, the DynamoDB function (step2) is triggered to save the encrypted data into the DynamoDB. Finally, the SNS publish sends a notification to user based on the communication protocol that customer chose. In my case, it is

email. The decrypted messages can be tracked back to readable format by invoking the “decrypt” API gateway endpoint in Postman.

How you met your menu item requirements: you will list the services you selected, and provide a comparison of alternative services, explaining why you chose the services in your system over the alternatives.

Compute

1. **AWS Lambda**

   There are lot of reasons for choosing AWS Lambda instead of other services

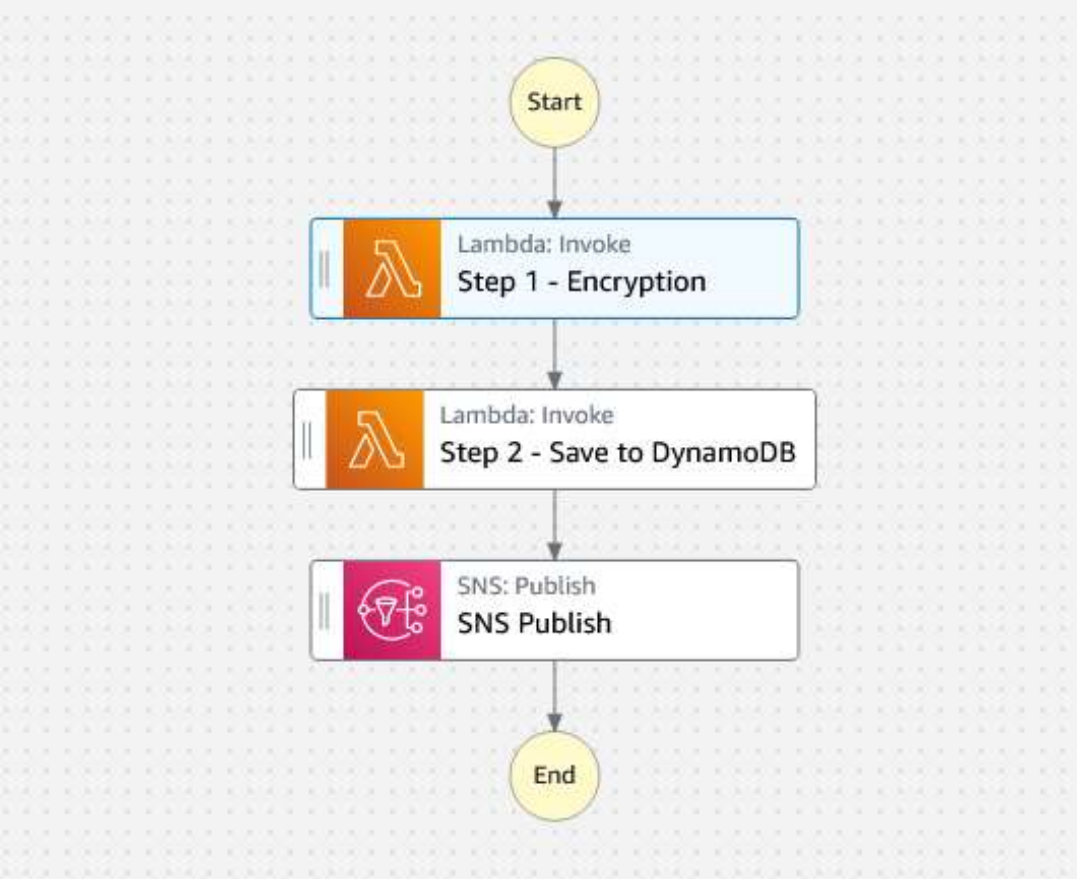
   - cost-effective: Lambda function is famously known for its significant cost saving methods. It just charges for the compute time that lambda function consumes.
   - Runs without servers: With Lambda functions in AWS, we do not need to worry about the server management, we just need to take care of writing and deploying the code. All the other functionalities like infrastructure and resource allocation is taken care by cloud providers.
   - Scalability: This feature has a smooth workload scalability. The automatic scaling functionality helps increase and decrease the payload based on the incoming traffic requests and allows the application to run efficiently.
   - For Instance, if there is an increase in incoming requests (high demand), AWS lambda creates more instances to handle the load. On the other hand, if there is low demand, the service automatically scales down the resources and reduces the cost. We cannot see this flexibility like EC2, EKS, Google Cloud, etc.

2. **AWS Step Function**

   The Step Function service is used to automate the flow of configured functionalities based on the conditions. It makes sure that application is running as per the configured logic by connecting multiple AWS services (Lambda, Database, EC2 SNS, etc) from start to end.

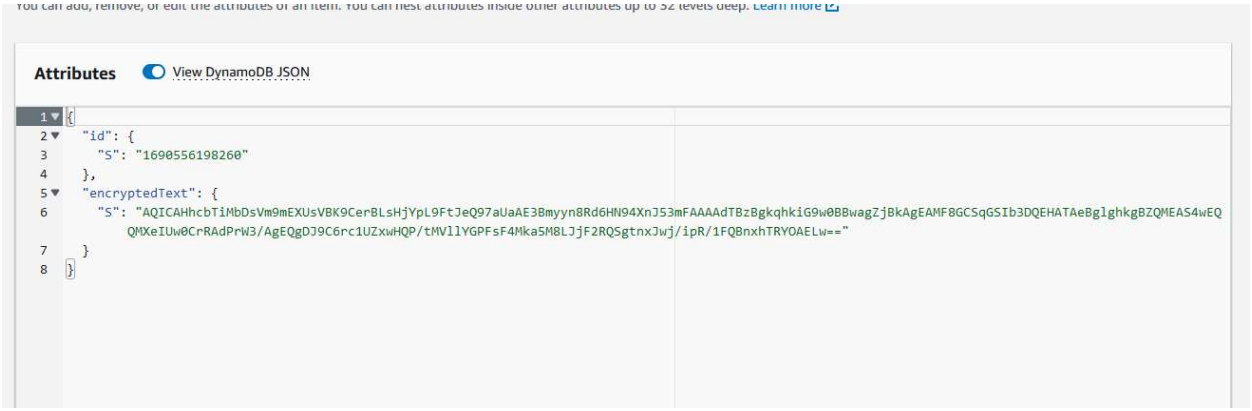
   The reason to choose this feature over other services are,

- Error handling: The flow of operations in the step function allows user to efficiently debug the application with its built-in error handling features. It has a continuous monitoring and logging feature which provides detailed steps on the status of workflows, the error its causing and the ways to resolve it.
- Workflow development: AWS Step function provides a service called “Workflow Studio”, helps user to create, design and visualize the workflow in a more user-friendly manner. No matte how complex the workflow is, we can easily drag and drop the built-in AWS service integrations for connecting various actions into one flow. This feature saves hours of time for people, as it eliminates the need for manually coding all the actions.
- We cannot see this type of flexibility in other AWS services like AWS SWF (Simple workflow service). We need to manually code each and every logic for these types of services. Hence AWS Step function is better choice.



Storage

1. **AWS DynamoDB -NoSQL database**
   - The reason for choosing DynamoDB over other services is it's a NoSQL database service which is completely managed by Amazon Web Services. We don't need to worry about table fields, and we can insert any number of fields we want.
   - Scalability and Storage: The AWS DynamoDB holds a record for higher scalability with the fastest response time at any complexity. Along with that it stores and maintains the backup data from time to time to recover from any data loss.
   - Why is this Service better? Because in service like Amazon RDS, we need to have well-defined, relational databases and require manual scaling. There are storage issues with Amazon S3 like it handles only limited queries. Hence DynamoDB is the better option.



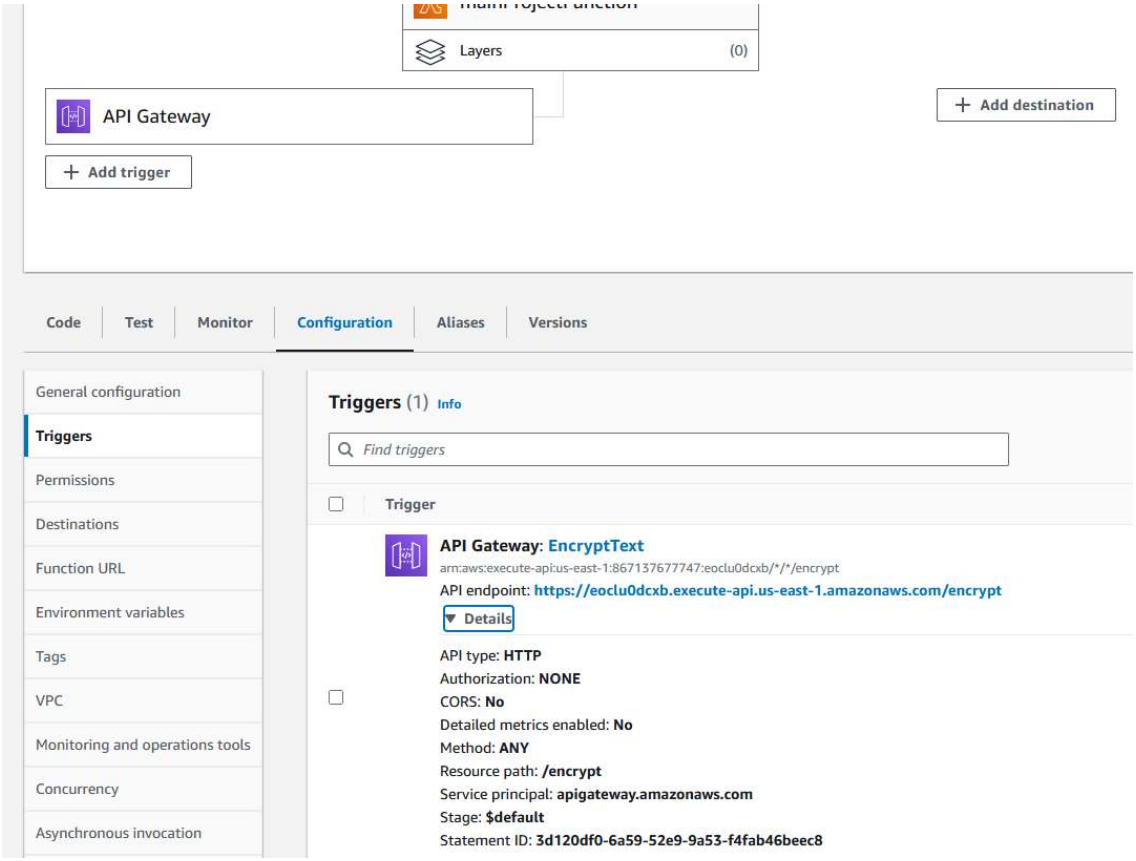
Network

1. **AWS API Gateway**

   Secure and route API requests to lambdas

   - The advantages of choosing API Gateway over other services are it is a fully managed AWS service. As a result, we do not need to worry about provisioning and maintenance.
   - We can easily create APIs, routes and invoke URL with this API Gateway Service.

- AWS CloudWatch is a feature used to monitor the incoming and outgoing traffic of API Gateway. Moreover, it also helps in troubleshooting issues if anything fails in the service.
- It consumes less cost when compared with services like CloudFront. CloudFront consumes more cost due to its availability, data transfer, requests, and scalability. In API Gateway, we do not need to worry about these modules. So, I chose AWS API gateway as my option [1].



General

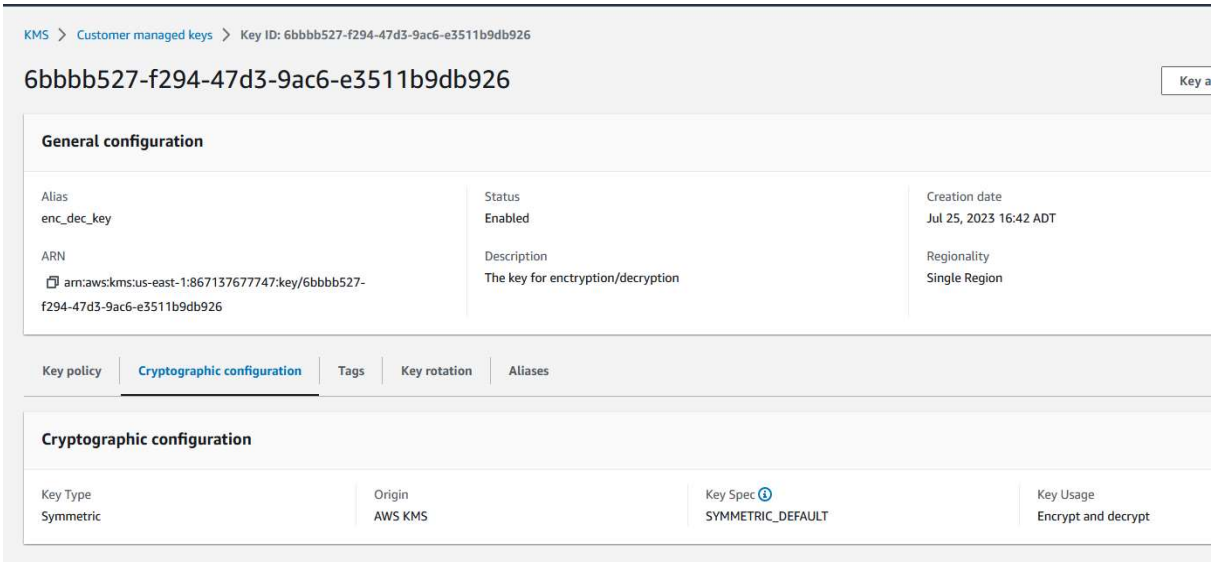
1. **AWS Key Management Service (KMS)**

   AWS KMS is popularly managed service in AWS, used for creating and managing the keys in a secure way.

   - The key is defined and stored in an encrypted format.
   - Cost-efficient. It's a pay-as-you-go model, charges only for the key you defined, stored, and used.

Things that made this service unique over other services are,

- The security of this AWS Key Management service controls access over the encryption keys from everyone except few users who are authorized to use it.
- The encrypted key is easily managed and used in difference services in AWS.
- Although, RDS provides simplified approach, the AWS KMS service provides fine-grained control over the encryption keys used in my application and services. Hence this is the appropriate option for my project.

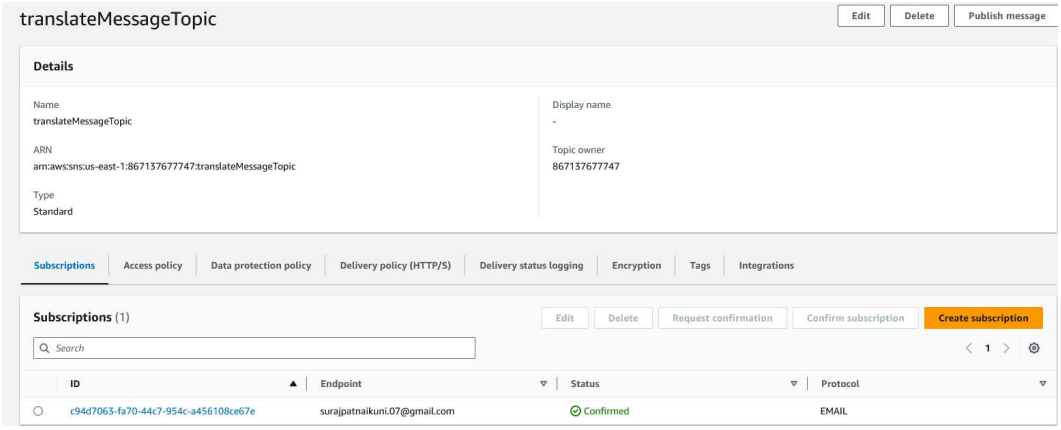


2. AWS Simple Notification Service (SNS)

- AWS SNS is notification service in AWS offers wide range of notification updates to the subscribed users via email, SMS or mobile.
- The owner has an option to restrict and public the content to the subscribers.
- This AWS SNS service has its own scalability feature, which handles and accommodates the services based on the message volume.
- Its automatic scalability helps reduce costs by scaling servers up and down based on the demand. As a result, it is cost effective.

Things that made this service unique over other services are,

- A message filter feature in SNS helps user to get notified for particular messages instead of all. This as a result helps protect privacy and also, they can avoid redundant notifications.
- Messages are securely stored, encrypted, and transferred.
- Services like AWS SQS is only dedicated to one-to-one model message delivery, which is not suitable for my project. Hence AWS SNS is better option.



A description of the deployment model you chose for your system, with reasoning for why you chose this deployment model.

The deployment model is public cloud. The main reason for this is because, the services I used in my project, like AWS Lambda, AWS Step Function, AWS DynamoDB, AWS API Gateway, Key Management Service, AWS SNS are all considered to be the part of public cloud infrastructure, which can be accessible over the internet by multiple users/customers from cloud providers like Amazon Web Services.

Along with that, my project mostly depends on serverless architecture features where I do not need to worry about the server management. I just need to focus on developing the application and hosting it into the cloud servers.

The reason why I chose this model is because

- Server Management: managed services reduces time on provisioning, infrastructure, and other configurations.

- Cost: Pay for the resources used
- Data availability: Wide range of data availability always makes the data available in different regions to not let server down in case of system failures.
- Scalability: Manages workloads efficiently

A description of the delivery model you chose for your system, with reasoning for why you chose this delivery model.

The delivery model I chose for my project is Function-as-a-Service (FaaS)

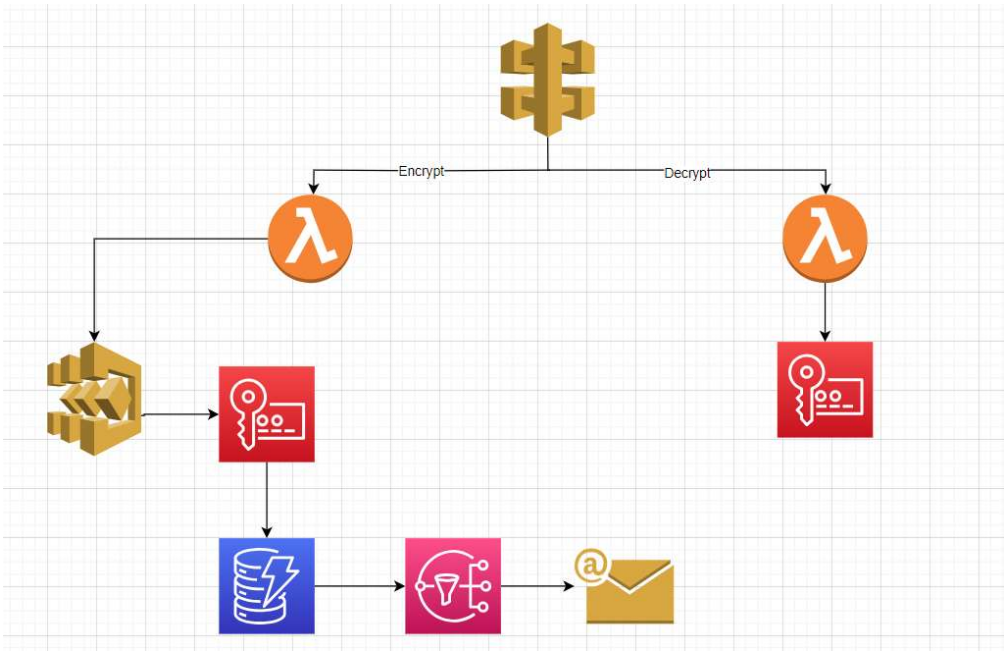
Function-as-a-Service (FaaS): Its one of the well-known cloud computing models helps developers to just focus on developing, running, and managing the application instead of worrying about the infrastructure and execution environments. This delivery model well suits my project because, my project is basically a serverless architecture which utilizes services like AWS Step functions, AWS API Gateway, AWS Lambda, etc.

The services I use like AWS API Gateway routes gateway requests to lambda through serverless approach without any need of server management. AWS Step Function builds state machines and the flow of tasks without any necessity of managing the underlying infrastructure. AWS Lambda helps develop the code and deploying it without any provisioning or other server managing configurations.

The reason for choosing Function as a Service (FAAS) is because

- Managing the servers: The cloud providers (AWS) handles all the parts of provisioning, resource allocations, server management, load balancing, security, etc.
- Stateless: The stateless feature helps maintain the application without any complexity. Along with it, the automatic scalability feature helps maintain a smooth in and out event triggers.
- Cost-efficient: The user is billed for the amount of resources they used and the execution time of the deployed function. The scaling feature assigns instances based on the demand and shuts down once the demand goes down. This as a result, can reduce the billing cost

The architecture of your final system. If your system does not match an architecture taught in the course you will describe why that is, and whether your choices are wise or potentially flawed.



1. *How do all of the cloud mechanisms fit together to deliver your application?*
   AWS API Gateway: It is one of the important services in my project, it acts as an access point for the users/customers to access the application. In my project, the API gateway is responsible for handling two routes: encrypt, decrypt.
   1. Encrypt Route: When client sends a POST request to the encrypt route, the API gateway triggers a Lambda function. A lambda function, a serverless compute service allows user to run the code on servers.
      Within the Lambda function, a step function is invoked, which has three main actions to fulfill:
      - The first step in this function is to execute the lambda function responsible for encrypting the text using AWS key management service (KMS).
      - After encryption, the step function executes the lambda function which saves encrypted data to DynamoDB database, a NoSQL service in AWS.

- The final step is to publish the topic using AWS Simple Notification Service (SNS). It delivers messages to multiple subscribers who subscribed to get updates on the service. In my case, I used emails as my communication method.

2. Decrypt Route: When client sends a POST request to the decrypt route with the encrypted text as a parameter, the API gateway triggers decrypt lambda function.
    - The encrypted text is decrypted with the help of AWS KMS and returns the original text to the user/customer.

In this way, all the cloud services fit together to deliver the application.

2. *Where is data stored?*

DynamoDB. The encrypted data is getting stored in DynamoDB with its unique id for every single execution.

3. *What programming languages did you use (and why) and what parts of your application required code?*

I used Node.js to build my application. There are four main functions in the application, where code implementation is needed in order to encrypt, decrypt, store and manage the data.

- **Encrypt Message:**

  The async lambda handler, a function that runs on AWS Lambda invokes a connection with AWS using a SDK module (The software development kit allows my application to interact with various AWS services). The main responsibility of using SDK in this function is to perform encryption using AWS KMS. An AWS KMS is used to list all the keys, thereby encrypting the data passed in the lambda function parameters. It first creates an id and a text parameter, encrypts the text, and saves it into readable format data. If everything goes as expected, it displays 200 status code or else it returns a server error with a status code of 500.

- **Save it to database:**

  This function invokes lambda handler by initializing the AWS DynamoDB client. A name is given to the DynamoDB table, “translatedMessages” to save the records with appropriate id and encrypted text. Here, AWS

DynamoDB client is interacting with my created DynamoDB table named, “translatedMessages”. A try catch method is used to save the data into the database with a promise method. It saves the encrypted text managed by AWS KMS and results in Success message if it is saved correctly or else and error message is displayed.

- **Decrypt Message:**
  The lambda handler in this function converts the encoded string to array (cipher text to readable format). It fetches the appropriate encryption key from AWS KMS to decrypt the data and results in decrypting the data into actual readable format. It returns 200 if the decryption is successful or else an error message with status code 500.

- **Main Function:**
  The main function acts as an entry point for the application. The lambda handler triggers state machine’s step function, thereby performs the set of actions in a logical order. It starts execution using the state machine’s ARN, input body with a promise method. With promise method, the function waits for the execution of state machine’s step function and returns the response. If the execution is successful, it displays a message of successful execution with the results or else, displays an error with the information to troubleshoot the error. (We can test this using postman).

4. *How is your system deployed to the cloud?*

This serverless application is deployed on AWS cloud, utilizing various services like Lambda, Step function, DynamoDB, API, KMS, SNS. Each has its own responsibility to fulfill the deployment tasks such as building and deploying code with lambda, creating task flows with Step, storing data in DynamoDB, API gateway to connect with lambda functions, managing encryption and decryption with KMS, sending notification via SNS to Gmail.

An analysis of your project's approach to security, particularly its approach to securing data through all stages of the system (in transit, at rest).

The list of security measures the application holds in the cloud during the implementation is

- Encryption (in transit): The API gateway routes API requests to the desired lambda function in a secure way. It supports SSL encryption to secure data in transit between end points.
- AWS Key Management Service (KMS): It stores the cryptographic keys outside the application. It securely encrypts and decrypts the data within my lambda function and avoids any sort of key exposures.
- Encryption in Data: The storage I chose is DynamoDB. The data has a feature which encrypts any sort of data stored in its environment. In my case, I am storing the encrypted data received from KMS. The encrypted data, which I received from AWS KMS is also encrypted in DynamoDB. Therefore, there is almost no chance of data leaks.
- AWS SNS: When the data is in transmit between AWS service to subscriber's endpoint, there might be chances of unauthorized access. The AWS SNS ensures that there is a secure transmission between SNS service to subscriber's endpoints.

What would your organization have to purchase to reproduce your architecture in a private cloud while providing relatively the same level of availability as your cloud implementation? Try to give a rough estimate of what it would cost, don't worry if you are far off. These systems are complicated, and you don't know all the exact equipment and software you would need to purchase. Just explore and try your best to figure out the combination of software and hardware you would need to buy to reproduce your app on-premise.

If we reproduce the architecture into private cloud, organizations' individual have to take care of everything like infrastructure, services, access, etc. The on-premise servers are needed to host the functions, for storing the data and for handling the incoming traffic, which costs a lot of bucks as physical servers are very costly. In order to handle the network traffic, routes, networking hardware, and handle load

balancing API requests, a proper internet connection with high performance is required. To create and manage the virtual machines, we need VM software like VMware to run virtual machines on physical server. This costs a lot if you need licenced VM software. For private cloud, the organization is fully responsible for its infrastructure. So, if anything fails, a backup is needed which costs money and also, if server goes down, they will be in loss.

“If a small business wat to purchase servers for their organization, it costs around $1000 - $3000 [4]”.

“Most of the organizations prefer Hyper-V licencing for VMs, so the rough estimation for this licence will be $3607 for 2 physical servers and $10,821 for 6 physical processors [5]”.

Therefore, these are the things we need to consider if we want to reproduce the architecture into private cloud.

Which cloud mechanism would be most important for you to add monitoring to in order to make sure costs do not escalate out of budget unexpectedly? In other words, which of your cloud mechanisms has the most potential to cost the most money?

The cloud mechanism that monitors and makes sure that costs do not escalate out of budget is AWS Lambda function. This is one of the main reasons for using this service in developing my project. It is cost effective, meaning that it charges for the invocation we made, it automatically scales the computing resources based on demand, ensuring limited resources are used. So, this service in my project is suitable for both deployment purposes and as well as scalability. EC2 can be used too, but Lambda is better option for start-ups or initial developments as its scalability feature can up and down the servers and saves the user’s cost.

An analysis of the cost metrics for operating your system. You will calculate the up-front, on-going, and additional costs to build this system in the real world. You will also explain alternative approaches that might have saved you money, or alternatively provide justification for a more expensive solution.

For the services I used in my project like, AWS Lambda, AWS Step Function, AWS DynamoDB, AWS API Gateway, AWS KMS and AWS SNS do not have any upfront costs as we do not have to worry about the investment in infrastructure or servers. It is like pay as you go process.

For ongoing costs, we only need to pay for the actual usages like number of requests and execution time for lambda, number of state transitions for step function, the amount of data getting stored in DynamoDB, API calls for API gateway, number of key requests for KMS and for the number of messages published for SNS.

Additional costs:

- There might be additional costs if there is an increase in storage data needs.
- The service SNS topic is free but for other protocols like SMS may incur additional costs for every request.
- The amount of data requested and transferred from different regions can incur additional costs.

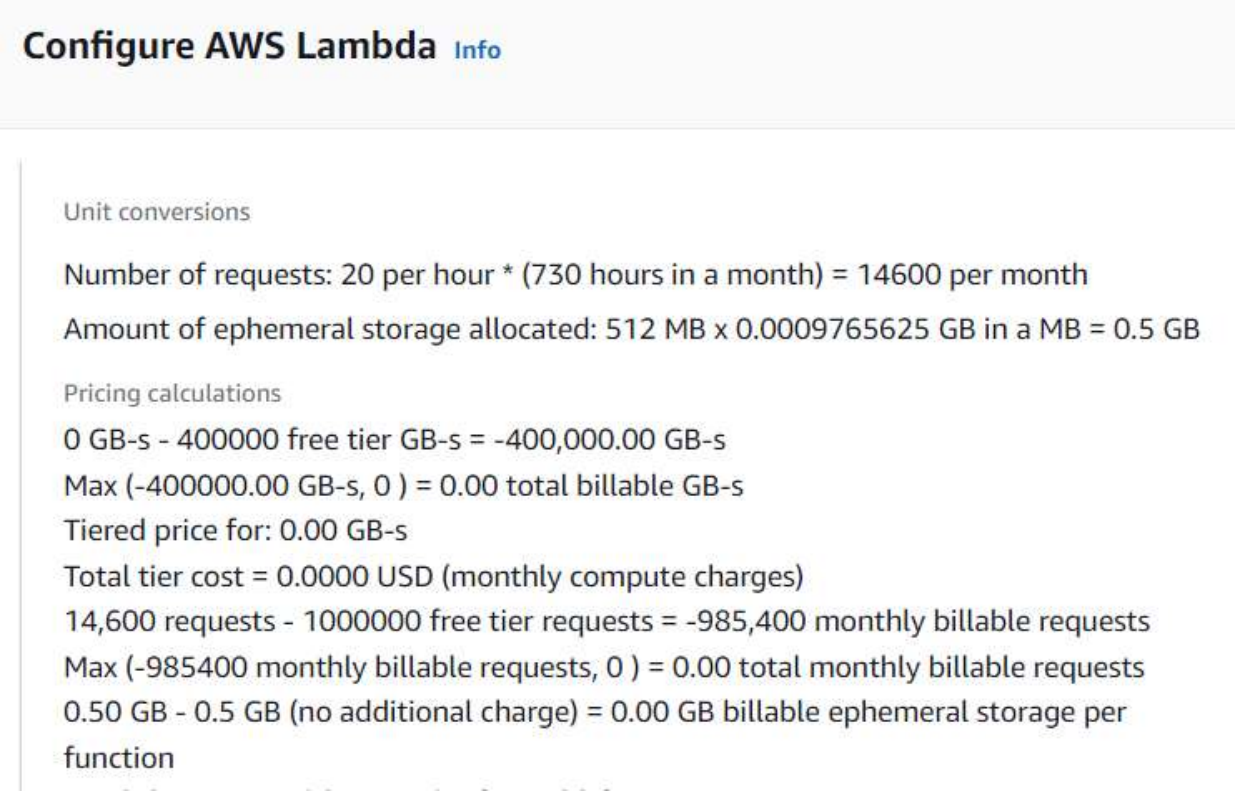
Some of the alternative approaches that could saved money are:

- Containerization: We can use containers instead of Lambda functions and use EC2 instances for running the application. This as a result could save money on execution process or for workloads.
- Data can be stored globally. If we do not want to replicate the data globally across multiple regions, we can just use single-region to save billing cost. Along with that, compressing the data is the best practice if it is compressible for saving storage costs.

provide justification for a more expensive solution:

- If we need high performance on low latency, we can utilize the services like Elastic beanstalk or EC2.
- If the application we build has predictable workloads, then we can use reserved instances that bills for the reserved amount of workload prediction we analyzed.

In general, AWS Lambda (to an extent) provides free tier that helps small business. This free tier include 1M free requests per moth and 4000,000 GB seconds to compute per month. Along with that, services like API gateway charge for the requests made in API and Step functions service charge for the flow of actions. (in cents)



DynamoDB is the one which costs a lot due to its specifications like storage, backup, provisioning, requests, etc.

How would your application evolve if you were to continue development? What features might you add next and which cloud mechanisms would you use to implement those features?

If I have to add new features to the current application, I will improve the security aspects by allowing subscribed users to access the encrypted text after authenticating the user with SMS OTP. This can be achieved through AWS SNS, AWS Cognito. The AWS Cognito helps user to configure the account for individuals and helps in authentication process. The mechanism is auto-scalable and takes care of its underlying infrastructure. On the other hand, the SMS OTP can be integrated with SNS for sending one time password to user's device. This can add

extra layer of security to the application. Although AWS Lambda can be used for this purpose, but there are lot of drawbacks with this service like security, it needs a lot of maintenance, which makes the service not eligible to use.

GitLab link:

https://git.cs.dal.ca/courses/2023-summer/csci4145-5409/patnaikuni/-/tree/main/TermProject

References:

1. *API gateway caching vs CloudFront: Which one to use?* (2023a) *Saturn Cloud Blog*. Available at: https://saturncloud.io/blog/api-gateway-caching-vs-cloudfront-which-one-to-use/ (Accessed: 27 July 2023).
2. *Draw.io - free flowchart maker and diagrams online* (no date) *Flowchart Maker & Online Diagram Software*. Available at: https://app.diagrams.net/ (Accessed: 27 July 2023).
3. MozDevNet (no date) *Uint8Array - JavaScript: MDN*, *JavaScript | MDN*. Available at: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Uint8Array (Accessed: 31 July 2023).
4. *How much does a server cost for a small business in 2023?* (no date) *What's the Cost of a Server for Small Business*. Available at: https://www.servermania.com/kb/articles/how-much-does-a-server-cost-for-a-small-business#:~:text=The%20average%20cost%20to%20rent,3000%20for%20a%20small%20business. (Accessed: 31 July 2023).
5. Leoni, K. (2021) *VMware or Hyper-V? part 3: Virtualization licensing costs*, *Longitude*. Available at: https://www.heroix.com/blog/virtualization-licensing/ (Accessed: 31 July 2023).
6. (No date) *AWS Pricing Calculator*. Available at: https://calculator.aws/#/ (Accessed: 31 July 2023).