Overtime Lifecycle Environment



**Prepared By:**

Abdelrahman Mamdouh

Abdelrahman Hesham

Mirna Azmi

Nourelhoda Ashraf

Omar Hesham

Randa Osama

Reham Ashraf

Table of Contents

[Objective 2](#_Toc15378551)

[System Requirements 2](#_Toc15378552)

[Cloud Architecture 3](#_Toc15378553)

[Infrastructure Components 3](#_Toc15378554)

[VPC 3](#_Toc15378555)

[Bastion Server 3](#_Toc15378556)

[Orchestration Machine 3](#_Toc15378557)

[Routing Tables 3](#_Toc15378558)

[API Gateway 3](#_Toc15378559)

[Subnets 3](#_Toc15378560)

[NACLS 3](#_Toc15378561)

[Security Groups 3](#_Toc15378562)

[NAT 3](#_Toc15378563)

[Cloud infrastructure diagram 4](#_Toc15378564)

[Bonus 5](#_Toc15378565)

[User Automated Creation 5](#_Toc15378566)

[Infrastructure as a code 6](#_Toc15378567)

[Cost 6](#_Toc15378568)

[Cloud Development 9](#_Toc15378569)

[Used Services/Tools 9](#_Toc15378570)

[AWS Lambda Service 10](#_Toc15378571)

[Relation Database Services (RDS) 10](#_Toc15378572)

[Simple Email Service (SES) 11](#_Toc15378573)

[API Gateway (Postman) 11](#_Toc15378574)

[Service Diagram 11](#_Toc15378575)

# [Github Repo](https://github.com/omarhesham1997/OverTime-Lifecycle-Cloud-Native-Application)

# Objective

We are required to develop a -Cloud Native- system that manages the overtime lifecycle according to the role of each system user.

The current system depends on two sub-systems “CATW” and “EES”, the users should update and insert the data in the two sub-systems, so we are required to create a single point data entry to decrease effort, time also minimize data error.

The new system is cloud native applied on AWS amazon web services trying to gain the max benefit from the AWS services such as RDS, Lambda………etc.

# System Requirements

* Orchestration Machine to manage RDS.
* Complete and secured cloud architecture.
* Service to handle single point data entry.
* Service to handle email approvals.
* Service for storage and archiving.

# Cloud Architecture

The infrastructure is built according to system requirements -security and networking- also to be easily using tunnels to access secured RDS.

## Infrastructure Components

### VPC

The virtual private cloud is created with CIDR range for IPV4 10.0.0.0/24.

All the infrastructure and the services are on the same region “us-east-1” USA Virginia.

### Bastion Server

The bastion server is the only component that exits in a public subnet and can be reached directly from the IGW.

A tunnel is created from the bastion to access RDS for development and testing.

### Orchestration Machine

The orchestration machine is the main machine used to handle RDSs also the machine can host other applications if needed.

The machine exits in a private subnet that can be only accessed using bastion server where the private key is installed.

SQL client is installed on the machine to manage the RDS.

### Routing Tables

### API Gateway

### Subnets

### NACLS

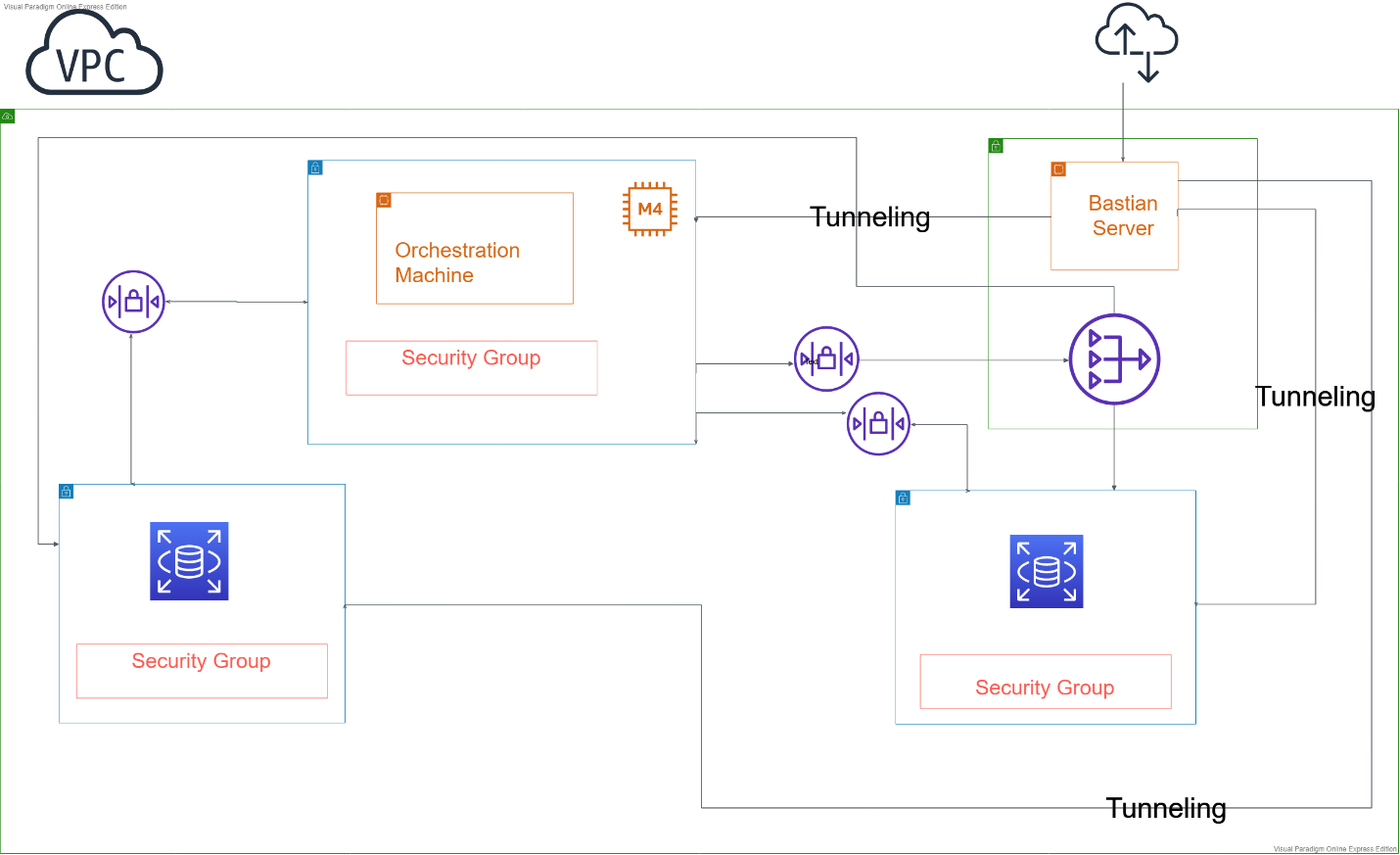
### Security Groups

### NAT

## Difficulties

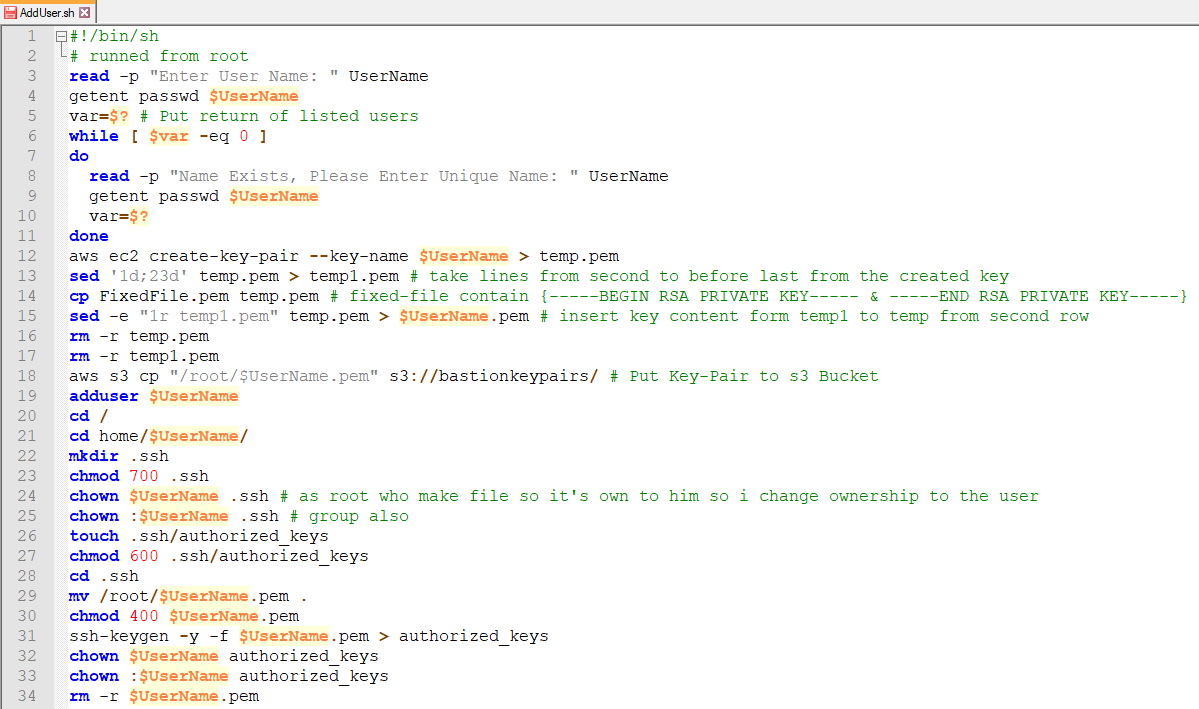
* Putting Lambda in Private subnet.
* Lambda can’t reach RDS.
* Database Connection due to firewall.
* Errors in Lambda environment variables.
* Missing NAT gateway.
* Tunneling to RDS through bastion.
* Syntax of terraform .
* Access users in creation.

## Cloud infrastructure diagram



## Bonus

### User Automated Creation



### 

### Infrastructure as a code

Terraform tool is used in creating infrastructure, the tool uses AWS CLI inserted credentials to verify the account and specify the user creating infrastructure

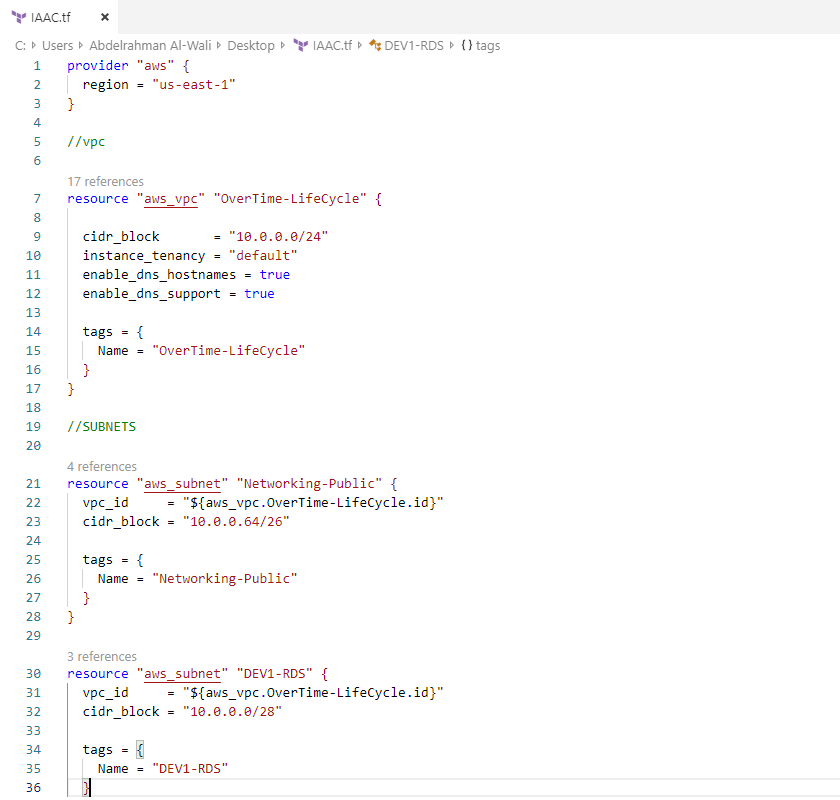


Figure Sample from code

For full code: [Github/Terraform](https://github.com/omarhesham1997/OverTime-Lifecycle-Cloud-Native-Application/tree/master/Infrastructure%20as%20a%20code)

## Cost

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Your Estimate |  |  |  |  |  |
| Service Type | Components | Region | Component Price | Service Price |  |
| Amazon EC2 Service (US East (N. Virginia)) |  |  |  | $20.98 |  |
|  | Compute: | US East (N. Virginia) | $17 |  |  |
|  | EBS Volumes: | US East (N. Virginia) | $3.80 |  |  |
|  | EBS IOPS: | US East (N. Virginia) | $0 |  |  |
|  | EBS Snapshots: | US East (N. Virginia) | $0.10 |  |  |
|  | Elastic IPs: | US East (N. Virginia) | $0.08 |  |  |
| Amazon S3 Service (US East (N. Virginia)) |  |  |  | $0.46 |  |
|  | S3 Standard Storage: | US East (N. Virginia) | $0.23 |  |  |
|  | S3 Standard Put Requests: | US East (N. Virginia) | $0.03 |  |  |
|  | S3 Standard Select Data Returned: | US East (N. Virginia) | $0.01 |  |  |
|  | S3 Standard Select Data Scanned: | US East (N. Virginia) | $0.01 |  |  |
|  | S3 Glacier - Storage: | US East (N. Virginia) | $0.02 |  |  |
|  | S3 Glacier - Storage Overhead: | US East (N. Virginia) | $0.01 |  |  |
|  | S3 Glacier - Data Retrieval (Standard): | US East (N. Virginia) | $0 |  |  |
|  | S3 Glacier - PUT/COPY Requests: | US East (N. Virginia) | $0.03 |  |  |
|  | S3 Glacier - Lifecycle Transitions: | US East (N. Virginia) | $0.03 |  |  |
|  | S3 Glacier - Data Returned by S3 Glacier Select (Standard): | US East (N. Virginia) | $0.01 |  |  |
|  | S3 Glacier - Data Returned by S3 Glacier Select (Expedited): | US East (N. Virginia) | $0.03 |  |  |
|  | S3 Glacier - Data Returned by S3 Glacier Select (Bulk): | US East (N. Virginia) | $0.01 |  |  |
|  | S3 Glacier - Data Scanned by S3 Glacier Select (Standard): | US East (N. Virginia) | $0.01 |  |  |
|  | S3 Glacier - Data Scanned by S3 Glacier Select (Expedited): | US East (N. Virginia) | $0.02 |  |  |
|  | S3 Glacier - Data Scanned by S3 Glacier Select (Bulk): | US East (N. Virginia) | $0.01 |  |  |
| Amazon RDS Service (US East (N. Virginia)) |  |  |  | $408.60 |  |
|  | DB instances: | US East (N. Virginia) | $0 |  |  |
|  | Reserved DB instances (one-time fee): | US East (N. Virginia) | $404 |  |  |
|  | Storage: | US East (N. Virginia) | $4.60 |  |  |
| Amazon SES Service (US East (N. Virginia)) |  |  |  | $26.55 |  |
|  | Send Messages from email clients | US East (N. Virginia) | $1 |  |  |
|  | Attachment from email clients | US East (N. Virginia) | $0.60 |  |  |
|  | Dedicated IP Addresses: | US East (N. Virginia) | $24.95 |  |  |
| Amazon VPC Service (US East (N. Virginia)) |  |  |  | $48.78 |  |
|  | NAT Gateway | US East (N. Virginia) | $33.84 |  |  |
|  | Private Link: | US East (N. Virginia) | $14.74 |  |  |
|  | Intra Region data Transfer: | US East (N. Virginia) | $0.05 |  |  |
|  | VPC Peering Data Transfer: | US East (N. Virginia) | $0.05 |  |  |
|  | Inter - Region Data Transfer Out: | US East (N. Virginia) | $0.10 |  |  |
| AWS Data Transfer In |  |  |  | $0 |  |
|  | US East (N. Virginia) Region: | Global | $0 |  |  |
| AWS Data Transfer Out |  |  |  | $0.36 |  |
|  | US East (N. Virginia) Region: | Global | $0.36 |  |  |
| AWS Support (Basic) |  |  |  | $0 |  |
|  | Support for all AWS services: |  | $0 |  |  |
|  |  | Total One-Time Payment: |  | $404 |  |
|  |  | Total Monthly Payment: |  | $101.73 |  |
|  |  |  |  |  |  |

# Cloud Development

## Used Services/Tools

  -AWS-Lambda Service

  -Relation Database Services (RDS)

  -Simple Email Service (SES)

  -Simple Storage Service (S3)

  -API Gateway (Postman)

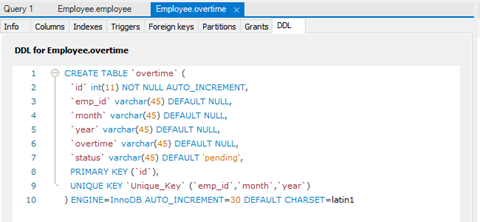
  -Eclipse IDE

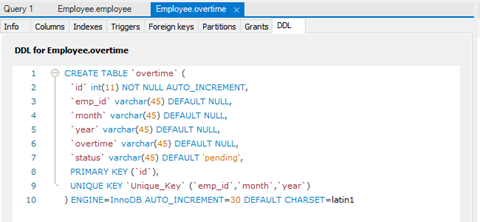
### AWS Lambda Service

Three lambdas functions have been developed to satisfy the core requirements of the project. ***The First Lambda***inserts the overtime information in the overtime table in the database. It also decodes the email content being received as base64 string and upload such on S3 in .msg format with a file name that concatenates the employee ID, the month and the year. ***The Second Lambda***listens to the creation of an object in S3 Bucket(trigger) to get invoked. It uses the employee ID to fetch the data in the database and relates it to their manager’s ID to get the manager email. It also uses ***SES***to send emails with the sender being (assuming) the employee and recipient being the manager with content that contains information about the employee and related overtime information (I.e. Employee A will have X overtime in month Y in Year Z) as well as making the (the employee ID, the month and the year.msg) file attachable to the email sent ***The Third Lambda*** updates the status of approval in the database depending on the response (Approved/Rejected) .

### Relation Database Services (RDS)

MySQL database was used as all the team have knowledge as well as it is included in the free tier. And we made 2 tables, one for employee’s info and the other for the overtime. The employee table contains the (employee ID-name-mail-the manager ID). The overtime table contains (employee ID- month- year- overtime).





### Simple Email Service (SES)

SES was used to send emails because it is a reliable, cost-effective service for businesses of all sizes that use email to keep in contact with their customers. And SNS wasn’t used because Troubleshooting can be very difficult when encountering an issue with SNS and often many AWS services.

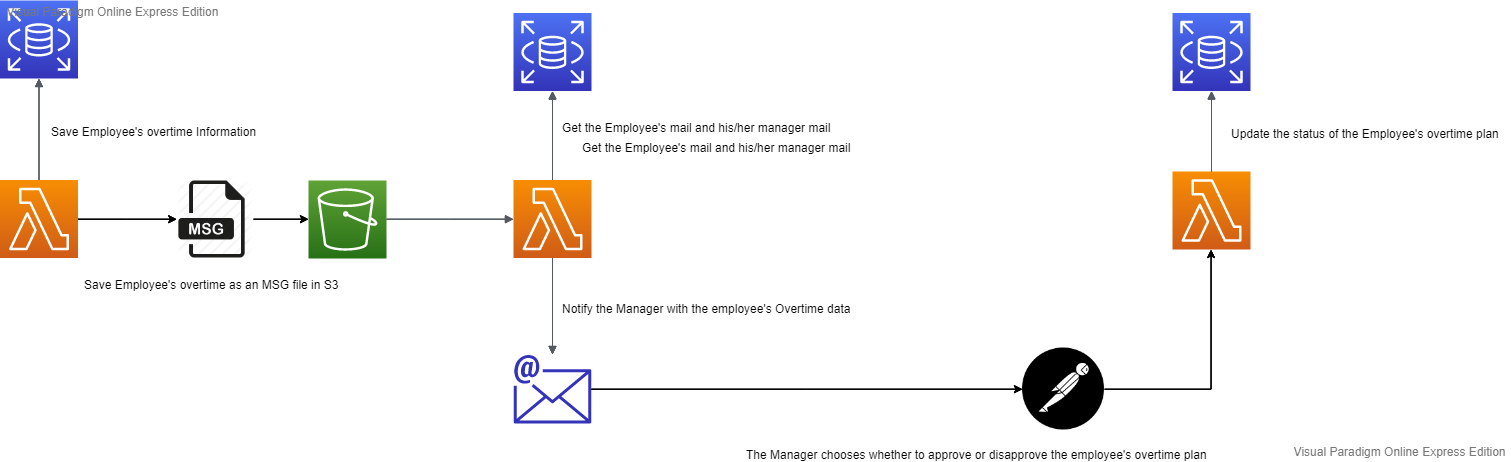
### API Gateway (Postman)

Postman was used to stimulate and test the functions without front-end.

Issues and Difficulties

-The network firewall didn’t allow us to connect to the RDS.

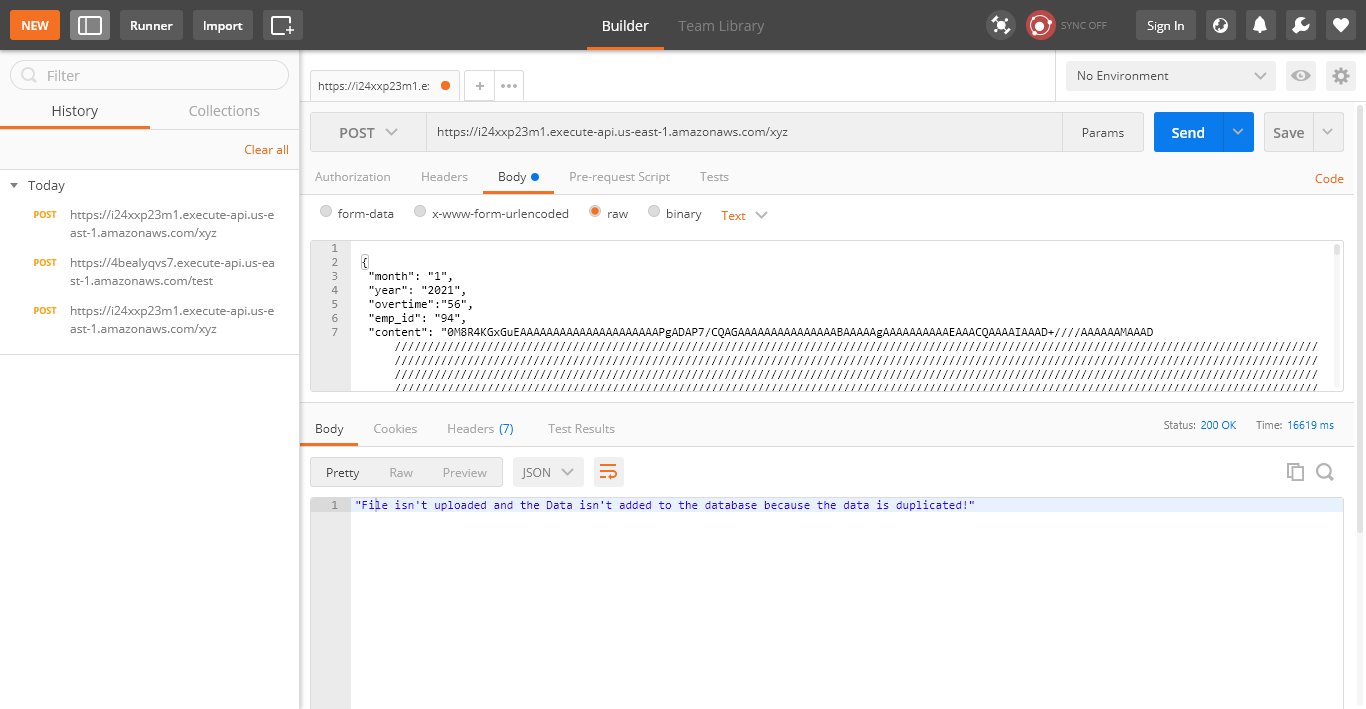
## Service Diagram



Test Cases

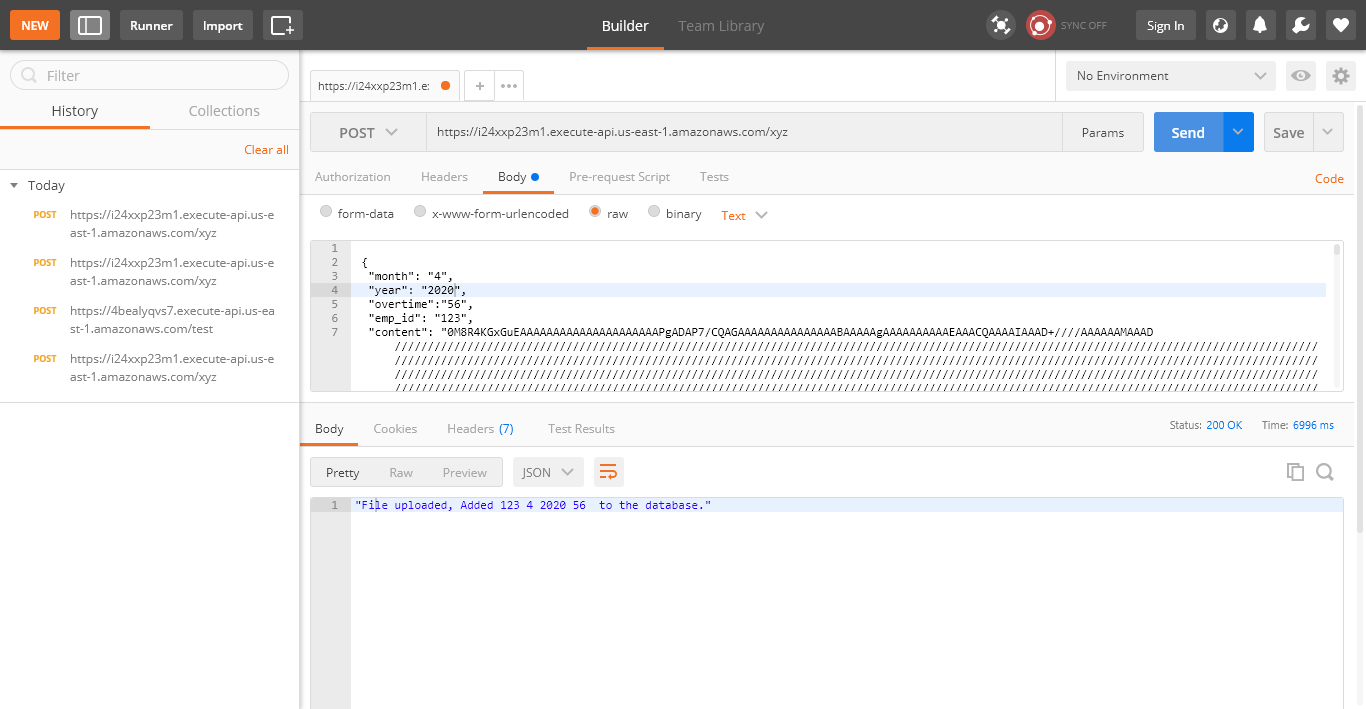
1. Test Case (1): Try to add values that is already entered previously and were saved in the database.

Result: The data won’t be entered again and an error message will be shown.



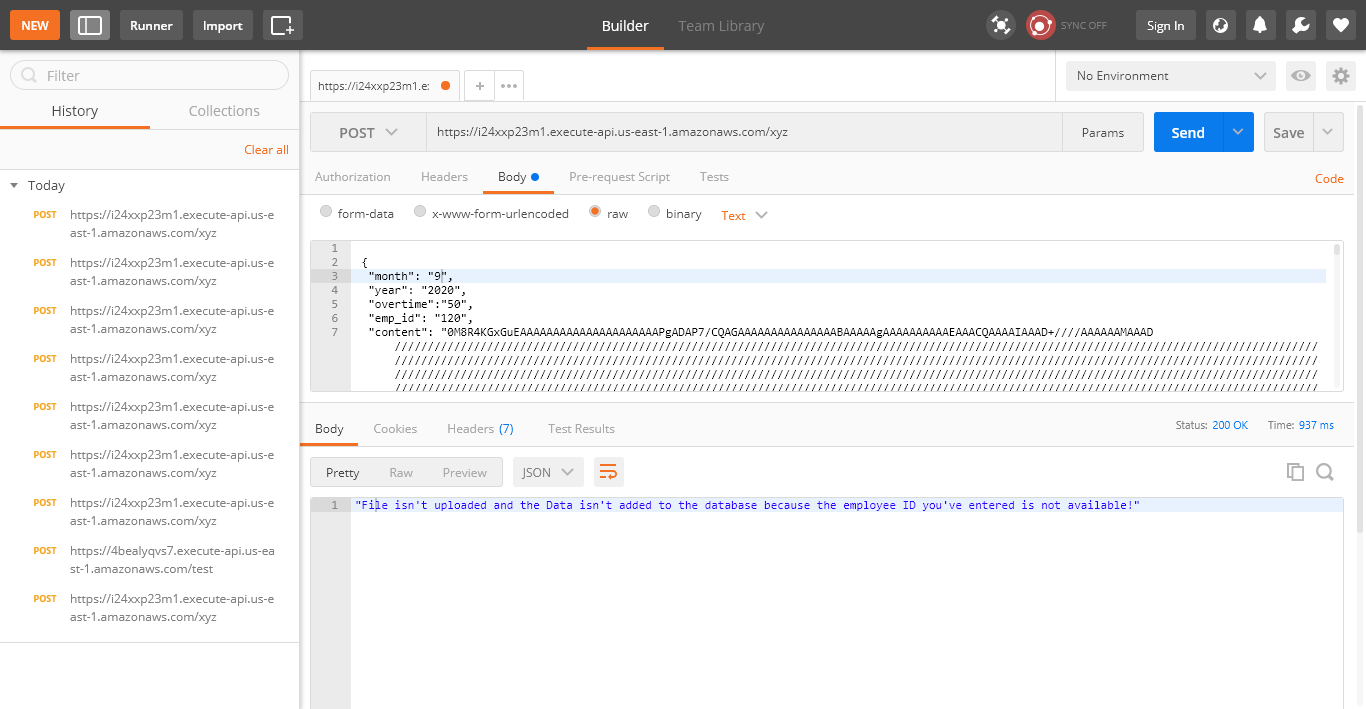
1. Test Case (2): Add unique values.

Result: The data will be saved in the database and an msg file will be created in the S3.



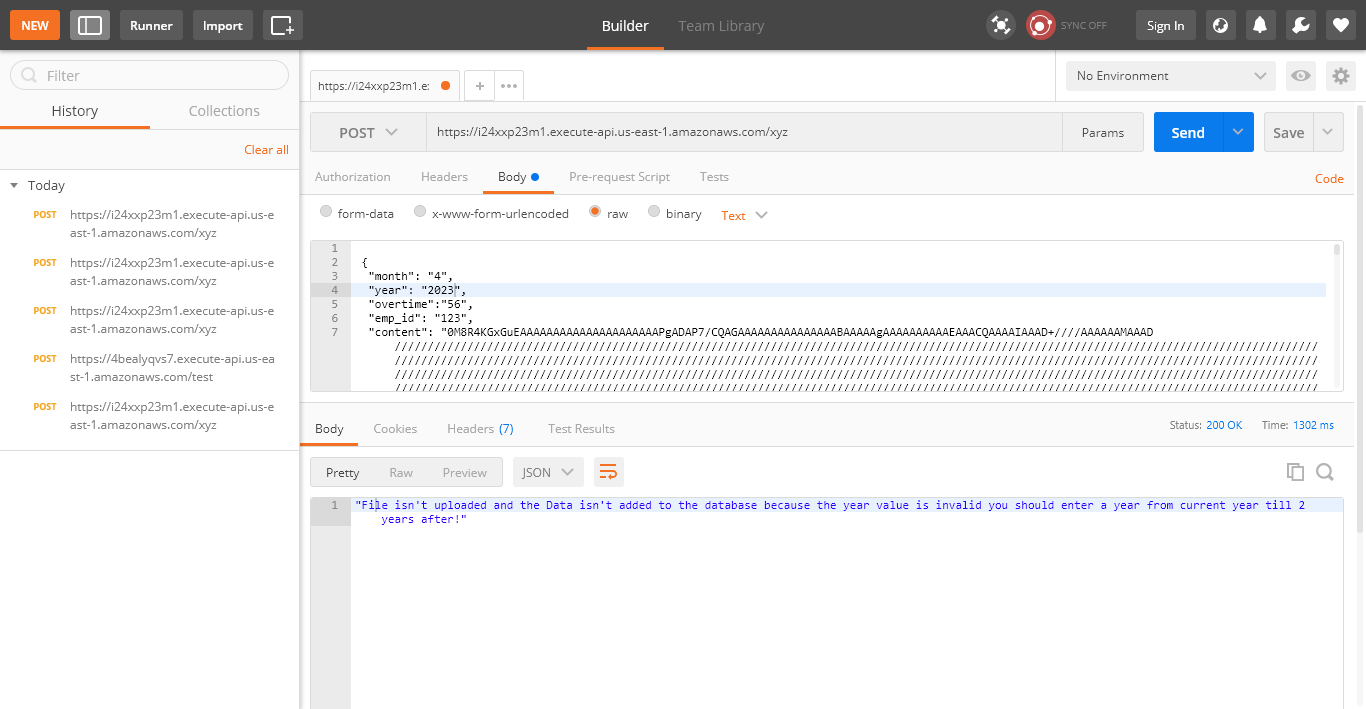
1. Test Case (3): Try to add the data with an employee’s ID that is not found in the Employee’s database.

Result: The data won’t be entered again and an error message will be shown.



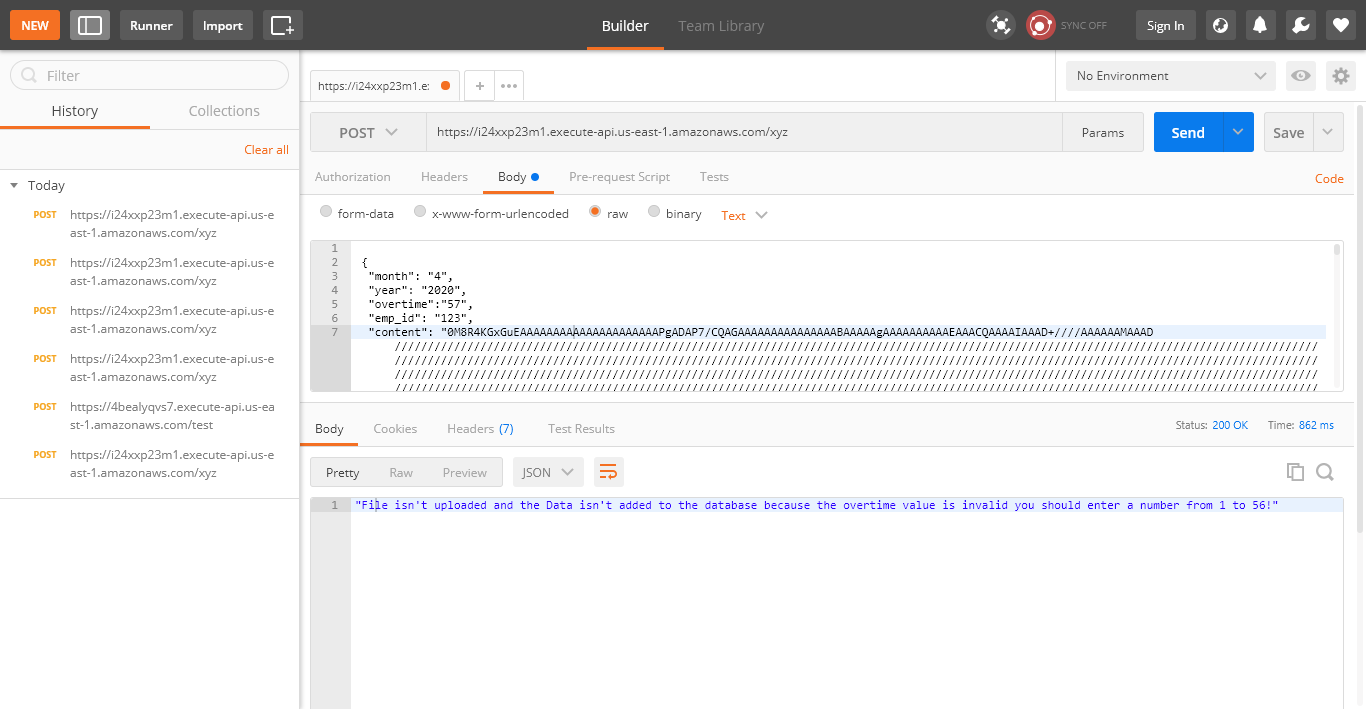
1. Test Case (4): Try to add a year that is not the current year, next year or 2 years after.

Result: the data won’t be entered again and an error message will be shown.



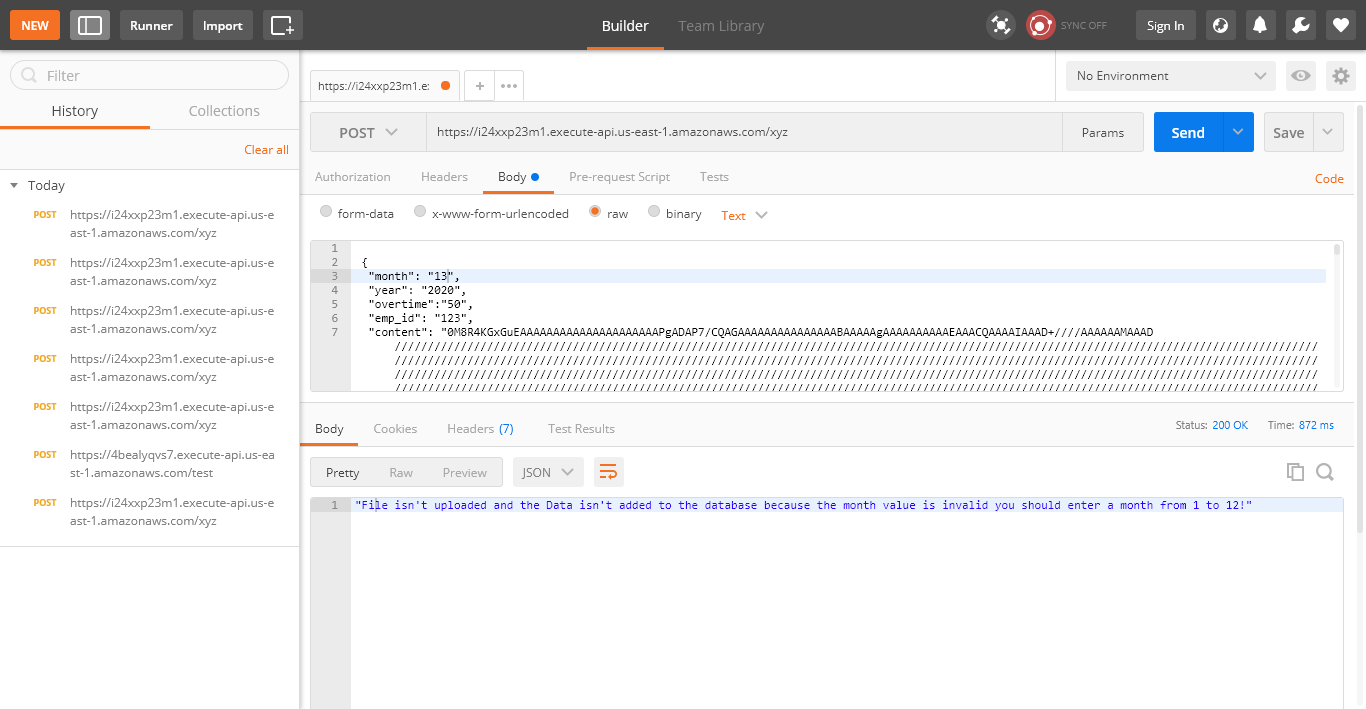
1. Test Case (5): Try to add an overtime that is greater/less than the range from 1 to 56.

Result: the data won’t be entered again and an error message will be shown.



1. Test Case (6): Try to add a month that is greater/less than the range from 1 to 12.

Result: the data won’t be entered again and an error message will be shown.



**-GUI**

**We thought of a solution to create a front-end like to be a better option than using Postman….so we have implemented two methods**

**-1st screen (1st lambda)**

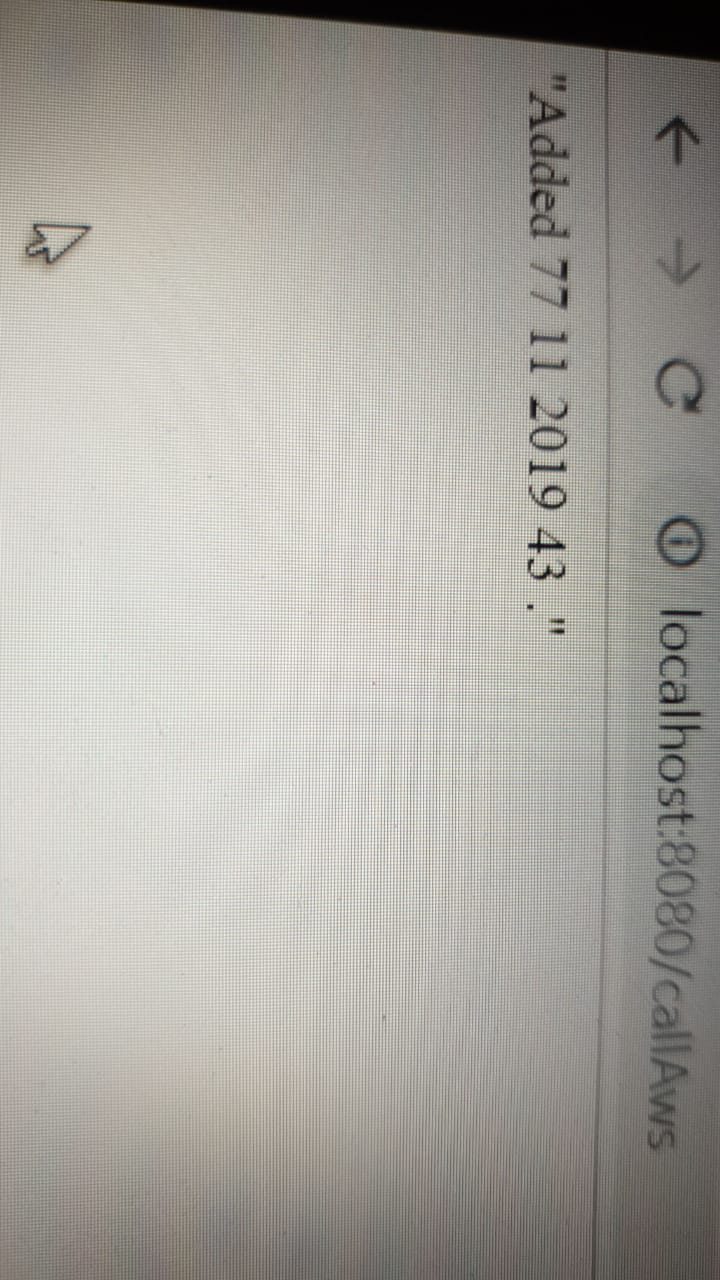
**We used Spring Boot to invoke lambda from an html file on local host**

**A screenshot of a cell phone

Description automatically generated**

**Instead of using the file content (String 64 based) we upload the actual.msg file and then the java code in the WebController class (Spring boot) converts it into string 64 based and then send the content to the lambda to revert it back to the .msg file**

**Result**

****

**-Another method for the GUI is using Angular, we invoked our lambda from a html file that includes angular code in which it calls the api gateway of such lambda and send the data to such. We used this approach in the 3rd lambda.We could either use the browser to run and invoke such or we could use amazon S3 static website to host such and invoke from there.**

**A screenshot of a cell phone

Description automatically generated**

**A screenshot of a social media post

Description automatically generated**

**After invoking…(was rejected)**

****

**Note: we could implement the both of them in a just unified way…..however we wanted to experiment try different approaches in order to create a user friendly presentation instead of postman**