OVERVIEW:-

This read me is to deploy and run the serverless micro service application with the help of API gateway and lambda function:-

After the deployment this application will consist of Api gateway and a lambda function. This will behave like server less micro service which will take inputs from user and give a response based on that
The sample end point looks like:-

```
https://{SAMPLE API ENDPOINT/<employee>
```

CODE STRUCTURE:-

icelerio (Base directory)
—> template.yaml (Artificat or instructions to deploy serverless application)
src (source folder)
—>employeeDetails.py (source code for implementation)
Authorization(folder)
—> authorization.py
—> sessionToken.py

INSTRUCTIONS TO RUN:-

Note: I am assuming the reviewer knows AWS and has valid AWS account.

1. First install AWS cli using here. Verify the installation in the command line

```
aws - -version
```

2. Please input your AWS CLI credentials for configuring aws cli in your machine
aws configure --profile <your profile name(it can be any)>

```
$ aws configure --profile user2
AWS Access Key ID [None]: AKIAI44QH8DHBEXAMPLE
AWS Secret Access Key [None]: je7MtGbClwBF/2Zp9Utk/h3yCo8nvbEXAMPLEKEY
Default region name [None]: us-east-1
Default output format [None]: text
```

** The font highlighted in red you should replace with your credentials

The detailed AWS documentation on how to set CLI can be found here.

3. Install SAM cli in your machine using link here.

4. Initialize the application using following command.

```
sam init --runtime python3.6
```

5. To test the application locally. Start the API Gateway endpoint locally. You must run the following command from the directory that contains the `template.yaml` file.

```
sam local start-api
```

6. The command returns an API Gateway endpoint, which you can send requests to for local testing.Copy the API Gateway endpoint URL, paste it in the browser, and choose Enter. An example API Gateway endpoint URL is http://127.0.0.1:3000/employee.

API Gateway locally invokes the Lambda function that the endpoint is mapped to. The Lambda function executes in the local Docker container and returns hello world. API Gateway returns a response to the browser that contains the text.

7. Create an S3 bucket in the location where you want to save the packaged code. If you want to use an existing S3 bucket, skip this step.

```
aws s3 mb s3://bucketname  --profile <your profile name>
```

8. Create the Lambda function deployment package by running the following `package` AWS SAM CLI command at the command prompt.

```
sam package --output-template-file packaged.yaml --s3-bucket <bucketname>
--profile <your profile name>
```

9. To deploy the serverless application to the AWS Cloud use the deploy command to deploy all of the resources defined on the template.

```
aws cloudformation deploy —template-file packaged.yaml -- stack-name <your
stack name> \
--capabilities CAPABILITY_IAM  --region <your region> --profile <profile
```

10. To test the serverless application in the AWS Cloud

1. Open the AWS Lambda console at https://console.aws.amazon.com/lambda/.
2. In the navigation column on the left, choose Applications. In the list shown, choose the application that you created in the preceding step.
3. Under Resources, expand the Serverless RestApi item, and copy the API endpoint URL.
4. Append /employee at the end of url and select appropriate http methods.

SECURITY OVERVIEW:-

1. The api's are protected with custom authorisers. The custom authorisers are using JWT token to verify the authorisation.

2. The api is configured to have a separate session end point which will give you the JWT token when all the requirements are met

3. The session end point will take a payload as input and the secret key to generate token as header

https://{SAMPLE API ENDPOINT}/session

Mandatory Header:

The value for jwt_secret should be ‘my-secret’ for the time being. If other values are used then you won’t be able to authorize the api

{ “jwt_secret” : “my-secret" }

Payload:

```
{       "user_id":"test",
        "password":"test1"

}
```

Output:-


```
{
   "token":
"b'eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MjQ2LCJmaXJzdF9uYW1lIjoiYmhhcmF0aCIsImNvbXBhbnkiOiJ3aXBybyIsImxhc3RfbmFtZSI6InN1YiIsImFnZSI6MzB9.1VLPe9JiZ_04G9x_eXlp5gEG7T9dpRmxDjwJIeRVdCM'"
}
```


Copy the token and pass it in the header to the following end points

{“**Authorization**”:
“eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZCI6MjQ2LCJmaXJzdF9uYW1lIjoiYmhhcmF0aCIsImNvbXBhbnkiOiJ3aXBybyIsImxhc3RfbmFtZSI6InN1YiIsImFnZSI6MzB9.1VLPe9JiZ_04G9x_eXlp5gEG7T9dpRmxDjwJIeRVdCM”}

sample end points for CRUD operations:-

- **Endpoint for GET employee details:-**

https://{SAMPLE API ENDPOINT}/employee?id=<employee no>

Sample output:-

If no records exist in db:

```
{
    "details": []
}
```

If record exists in db:

```
{
    "details": [
        {
            "last_name": "sub",
            "company": "wipro",
            "id": "246",
            "first_name": "bharath",
            "age": "30"
        }]
}
```

Endpoint for POST employee details:-

https://{SAMPLE API ENDPOINT}/employee

Input:

```
{   "id":246,
    "first_name":"bharath",
    "company":"wipro",
    "last_name":"sub",
    "age":30
}
```

Sample output:-

```
{
    "response": "successfully created new entry"
}
```

Endpoint for PUT employee details:-

https://{SAMPLE API ENDPOINT}/employee

For updating the record please send the payload similar to the following.

Input:-

```
{    "id":246,
     "first_name":"bharath",
     "company":"wipro",
     "last_name":"sub",
     "age":40
}
```

Sample output:-

```
{
     "response": "successfully updated the record"
}
```

Endpoint for DELETE employee details:-

https://{SAMPLE API ENDPOINT}/employee?id=<employee no>

Sample output:-

```
{
     "response": "successfully Deleted the record"
}
```