

Project : Fortune Teller Application 🧙‍♂️

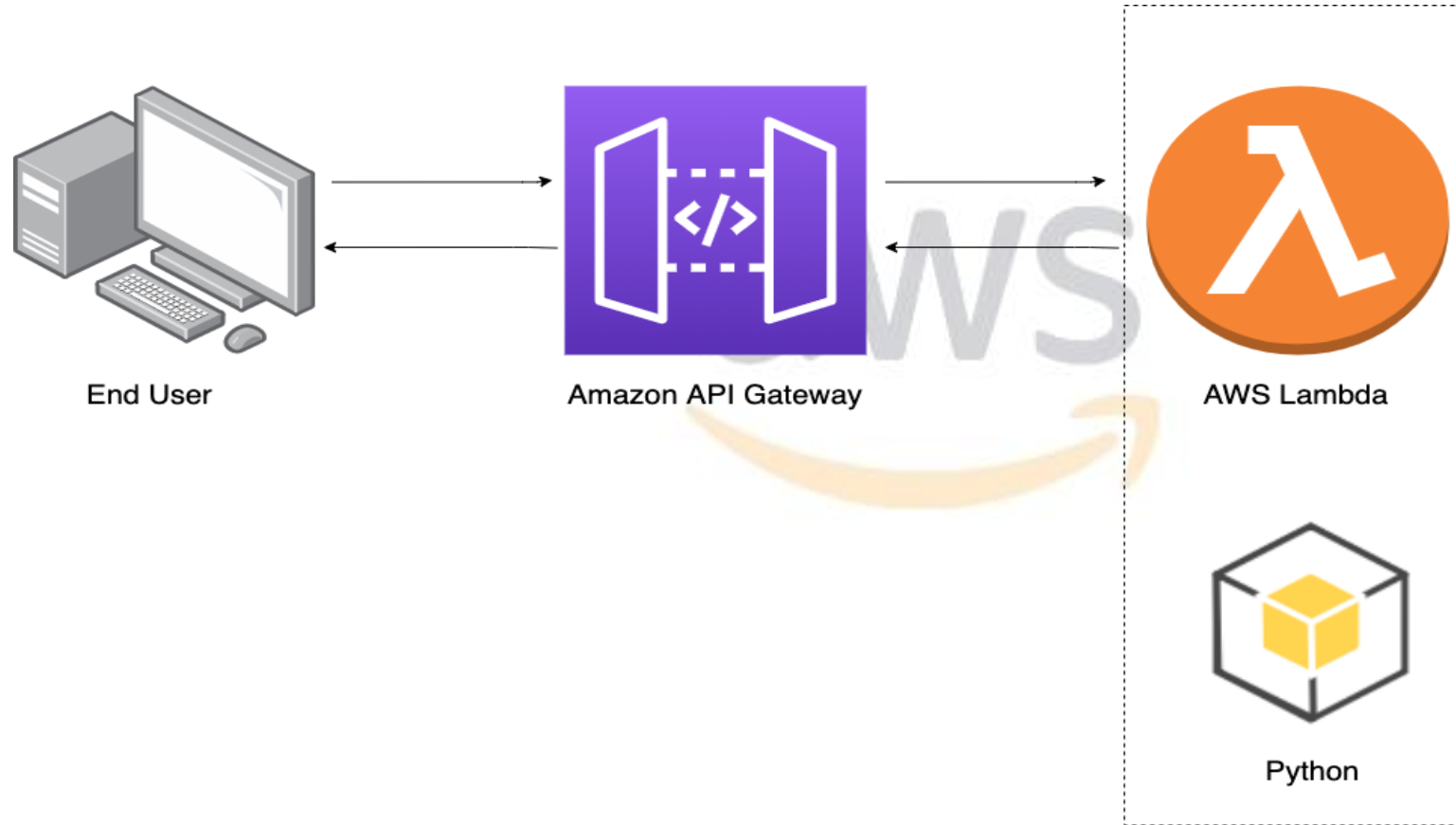
Project Overview:-

- The Fortune Teller application is designed to provide users with entertaining and personalized insights into their future, based on various mystical and fun elements such as astrology.
- The Fortune Teller application is a simple and entertaining service that provides users with random fortunes based on a generated integer.
- The core functionality is implemented as an AWS Lambda function, which generates a random integer and maps it to different fortune responses using conditional statements.

Services used:-

AWS Lambda, API Gateway , AWS IAM , AWS Cloud Formation.

Architecture diagram 🏗️ :-



Explanation:-

Step #1: Set Up an AWS Lambda Function

- Open the AWS Management Console and navigate to the Lambda service.
- Click "Create function" and select "Author from scratch."
- Enter a name for your function and select a runtime (e.g., Python 3.8).
- Under "Function code", write or upload the code for generating random responses.
- Configure the function settings, such as memory, timeout, and execution role.

Step #2: Write the Fortune-Telling Code

- Import the random module in Python.
- Define a function that generates a random integer between 1 and 3.
- Use conditional statements to map the random integer to a response (e.g., 1 = "yes", 2 = "no", 3 = "maybe").
Return the response as the output of the Lambda function.

Step #3: Set Up an API Gateway

- Open the API Gateway service in the AWS console.
- Click "Create API" and choose "HTTP API."
- Define the route, method (e.g., GET), and integration type (Lambda function).
- Select the Lambda function you created earlier.
- Configure the request and response settings as needed.



Step #4: Deploy Your Code

- In the API Gateway console, click "Deployments" and then "Create."
- Select the stage (e.g., "prod") and deploy the API.
- Copy the API endpoint URL, which users will use to access the fortune teller.

Step #5: Test Your Code

- Use a tool like Postman or simply your browser to send a GET request to the API endpoint.
 - Verify that you receive a "yes", "no", or "maybe" response to each question.
 - Test with different questions to ensure the randomness of the responses.
-
- **Cloud Formation** :- Manages the infrastructure resources for the entire Fortune Teller application.



Sample code for fortune-Teller 🧙 in python Code:-

```
import json
import random
def generate_fortune():
    random_integer = random.randint(1, 6)
    if random_integer == 1:
        return " You will have a great day!"
    elif random_integer == 2:
        return " Be patient. Good things come to those who wait."
    elif random_integer == 3:
        return "Adventure awaits you in the near future."
    elif random_integer == 4:
        return "Your creativity will shine this week."
    elif random_integer == 5:
        return "A financial opportunity will present itself."
    else:
        return " your fortune says: Adventure awaits you in the near future."
def lambda_handler(event, context):
    fortune = generate_fortune()
    print("Generated Fortune:", fortune)
    response = {
        'statusCode': 200,
        'body': json.dumps({'fortune': fortune})
    }
    return response
```

Challenges faced during the creation for this project:-


1. Configure IAM roles with the principle of least privilege, use HTTPS or HTTP for API Gateway.
2. Configuring API Gateway, setting up correct request and response mappings.
3. If you any doubt regarding the integration of any AWS services with the API Gateway we can check it in the resource section under TEST Icon. Where it gives the response code about result of the test.

 Link:-

<https://wc24kx88ba.execute-api.us-east-1.amazonaws.com/Fortune-Teller>

This is the link for the Fortune-Teller application which I was created.

Conclusion:-

-  The “Fortune Teller application” built on AWS Lambda and API Gateway showcases the power and flexibility of serverless architecture for developing lightweight and scalable applications.
-  The application seamlessly integrates AWS Lambda, API Gateway, and IAM roles to provide a secure and robust solution. AWS services work together to handle HTTP requests, generate random fortunes, and respond to users in real-time.

