# SERVERLESS WEB APPLICATION DEPLOYMENT ON AWS
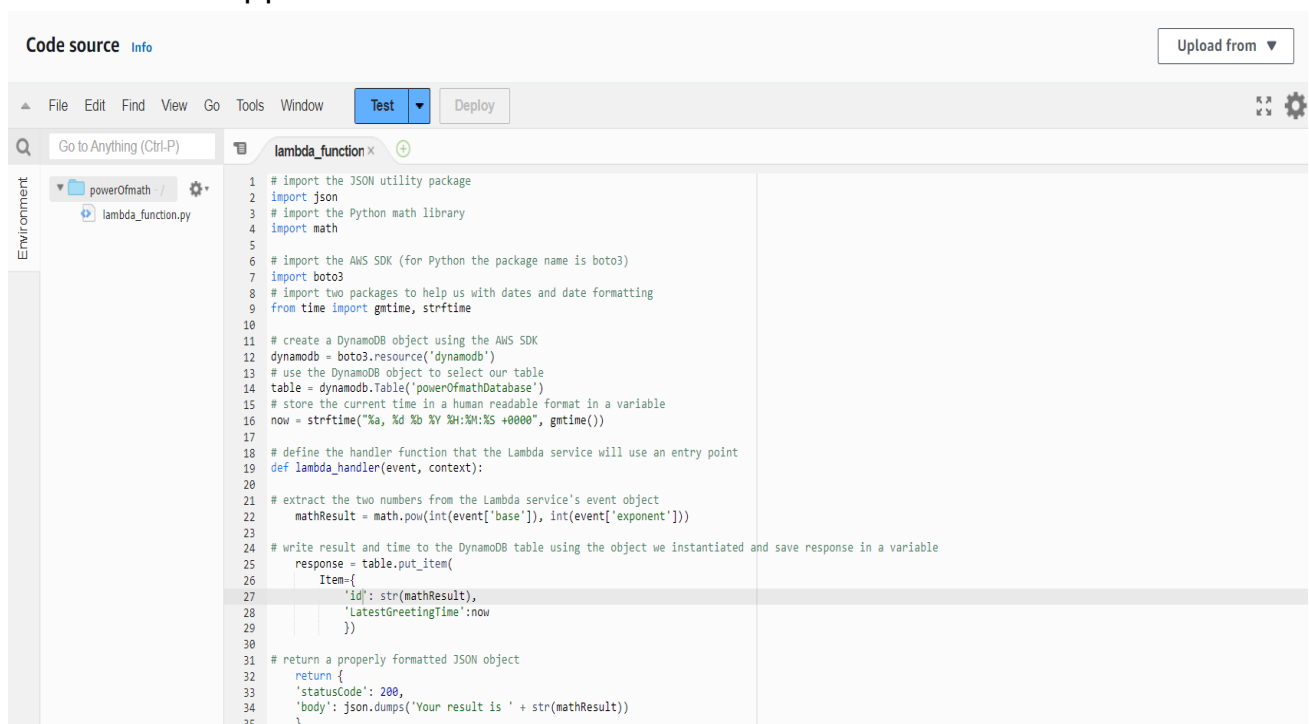
Abstract:

In this project I have deployed a simple web application using serverless architecture on AWS. I have used different AWS services through out this project like AWS Amplify, AWS lambda, AWS API Gateway and AWS DynamoDB. This serverless architecture a software design model that lets developers build and manage applications without having to manage the underlying infrastructure, and it reduces the cost.

Process of Deploying:

Step-1: Create an app on AWS Amplify and deploy the sample web application on it.
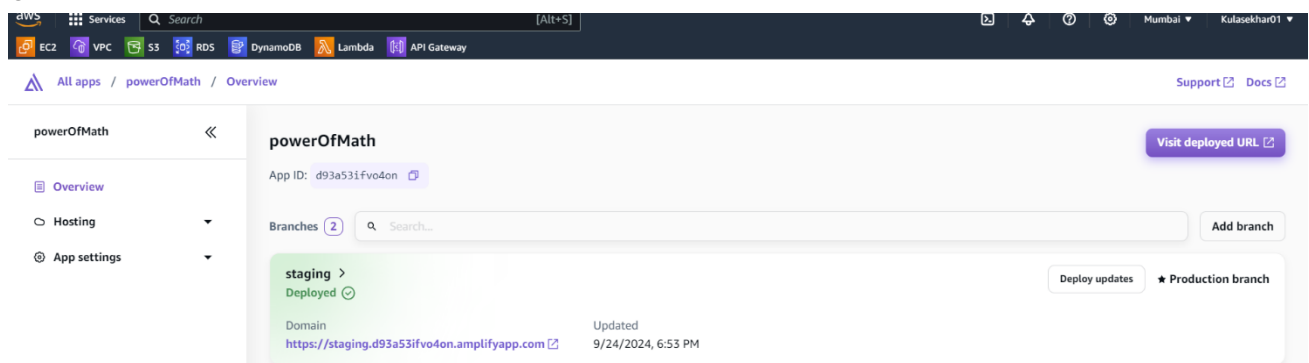

Step-2: Now create the lambda function to process the request, which will take the base and exponent values, calculates the power and stores on DynamoDB and return to application.


```python
# import the JSON utility package
import json
# import the Python math library
import math

# import the AWS SDK (for Python the package name is boto3)
import boto3
# import two packages to help us with dates and date formatting
from time import gmtime, strftime

# create a DynamoDB object using the AWS SDK
dynamodb = boto3.resource('dynamodb')
# use the DynamoDB object to select our table
table = dynamodb.Table('powerOfmathDatabase')
# store the current time in a human readable format in a variable
now = strftime("%a, %d %b %Y %H:%M:%S +0000", gmtime())

# define the handler function that the Lambda service will use an entry point
def lambda_handler(event, context):

    # extract the two numbers from the Lambda service's event object
    mathResult = math.pow(int(event['base']), int(event['exponent']))

    # write result and time to the DynamoDB table using the object we instantiated and save response in a variable
    response = table.put_item(
        Item={
            'id': str(mathResult),
            'LatestGreetingTime':now
        })

    # return a properly formatted JSON object
    return {
    'statusCode': 200,
    'body': json.dumps('Your result is ' + str(mathResult))
    }
```

Step-3: If we want to execute the lambda function, we need to add trigger (API Gateway). Which will trigger the lambda function.
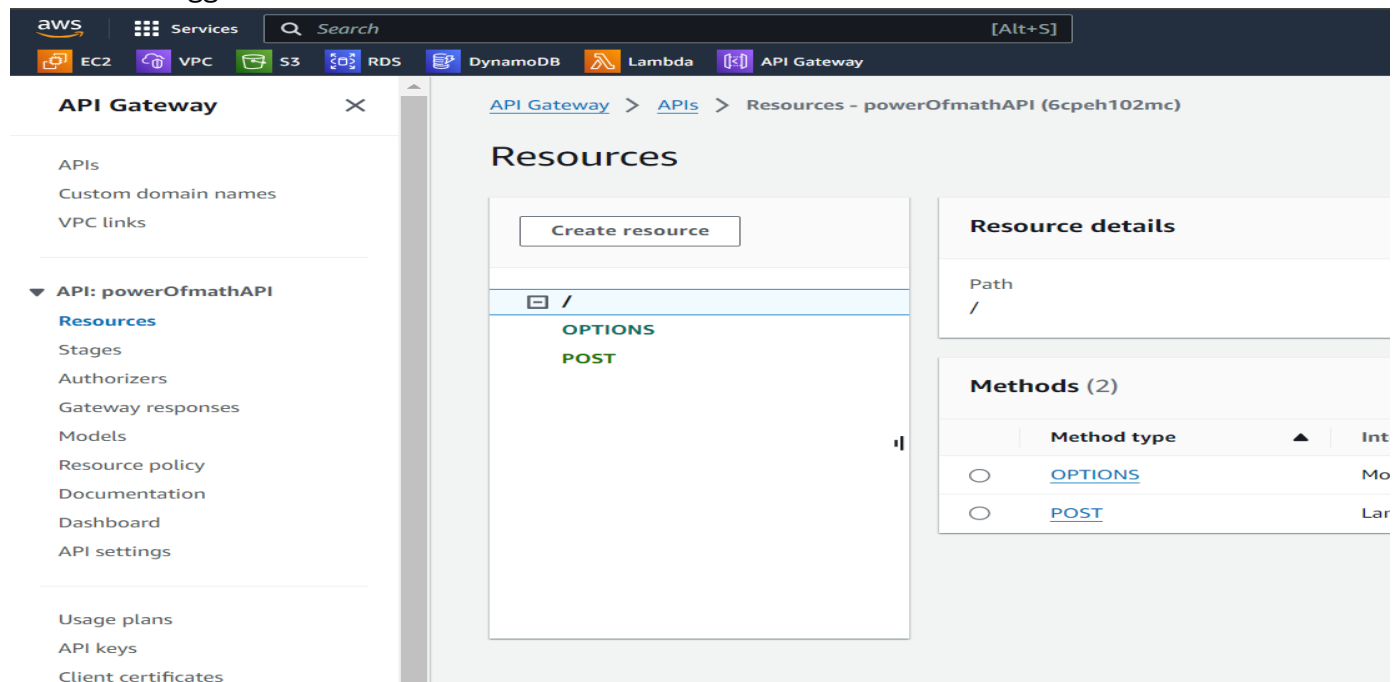
Step-4: Then create DynamoDB table for storing those data and give appropriate IAM permissions that lambda function access the DynamoDB.
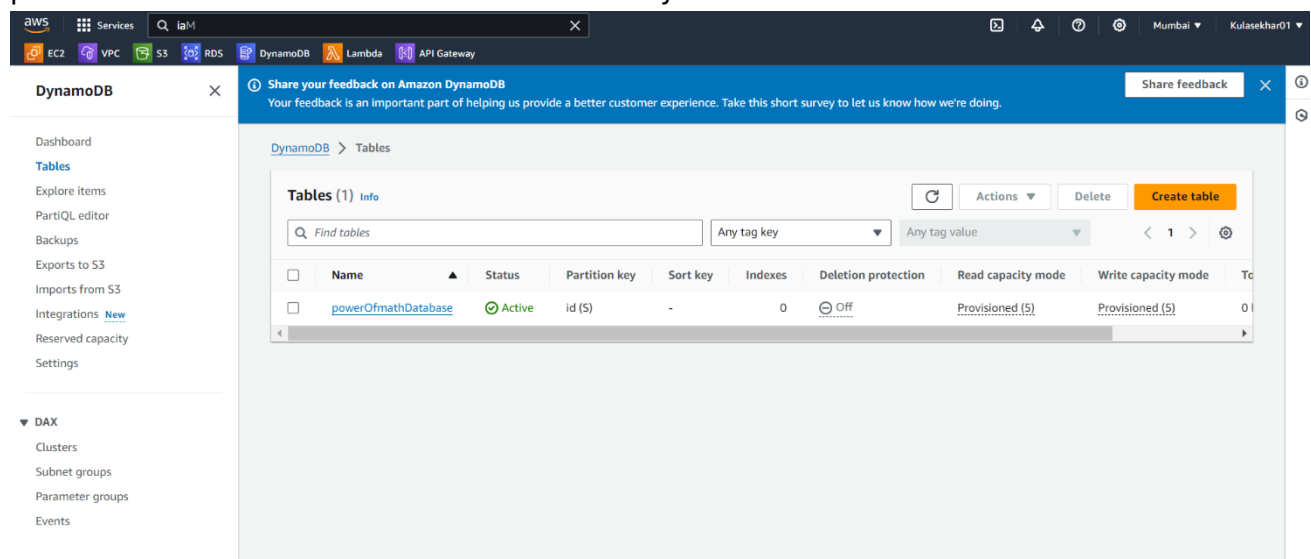
Step-5: Finally, we setup all the services on cloud now we need to change the web application JavaScript code for appropriate result.

```
<script>

    // callAPI function that takes the base and exponent numbers as parameters

    var callAPI = (base,exponent)=>{

        // instantiate a headers object

        var myHeaders = new Headers();
```

```javascript
    // add content type header to object

    myHeaders.append("Content-Type", "application/json");

    // using built in JSON utility package turn object to string and store in a variable

    var raw = JSON.stringify({"base":base,"exponent":exponent});

    // create a JSON object with parameters for API call and store in a variable

    var requestOptions = {

      method: 'POST',

      headers: myHeaders,

      body: raw,

      redirect: 'follow'

    };

    // make API call with parameters and use promises to get response

    fetch("YOUR API GATEWAY ENDPOINT", requestOptions)

    .then(response => response.text())

    .then(result => alert(JSON.parse(result).body))

    .catch(error => console.log('error', error));

  }
</script>
```

API CALL:

```html
<button type="button"
onclick="callAPI(document.getElementById('base').value,document.getElementById('expone
nt').value)">CALCULATE</button>
```