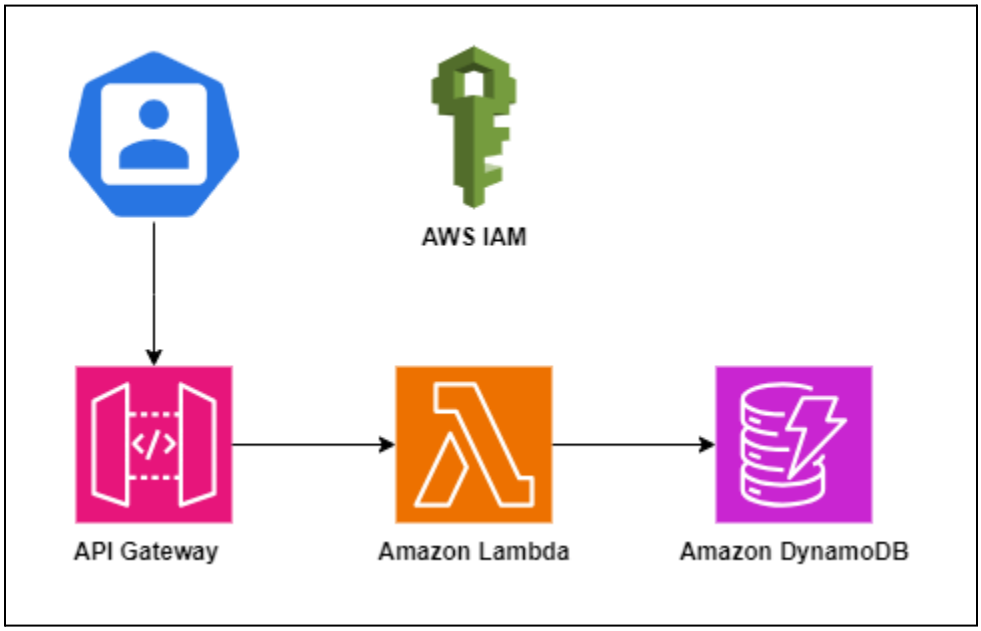
Cloud Project: Building a Serverless Registration WebApp with REST APIs

Table of content

Project flow

This project demonstrates the creation of a scalable, serverless web application for user registration using AWS services. The application allows users to register by submitting a form, which sends data to the API Gateway. The API Gateway triggers a Lambda function that processes the data and stores it in DynamoDB.



Tech Stack

WebApp: HTML, CSS, JavaScript
Services: Amazon DynamoDB, IAM, AWS Lambda, Amazon API Gateway

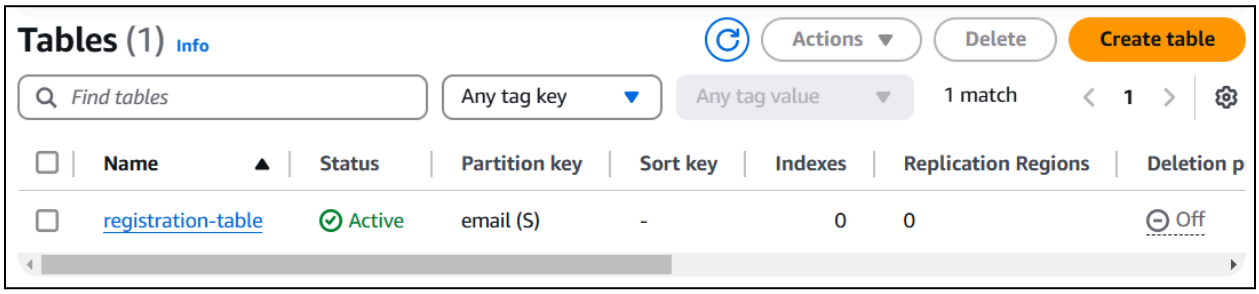
Lambda function written in Python

---

1. Create DynamoDB Table

Create a new table to start exploring DynamoDB

- Table name: registration-table
- Region: set it as ap-southeast-1
- Partition key: email (String type)
- Keep all table settings as default



2.. Create IAM Role for AWS Lambda

Create a new role for Lambda.

Attach permissions:
1. AmazonDynamoDBFullAccess
2. AWSLambdaFullAccess

Name the role RegistrationFormRole and create it.

---

3. Create AWS Lambda Function

- Function Name: registration-form-function

- Runtime: Python 3.9
- Architecture: x86_64

For the execution role, use an existing role as RegistrationFormRole as created before.

Function ARN:
arn:aws:lambda:ap-southeast-1:615299737603:function:registration-form-function

---

4. Write Lambda Function

In Lambda function registration-form-function, write the lambda function. Then choose Deploy code.


```
import json
```

```python
import boto3
dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('registration-table')
def lambda_handler(event, context):
    response = table.put_item(
        Item={
            'email': event['email'],
            'name': event['name'],
            'phone': event['phone'],
            'password': event['password']
        }
    )
    return {
        'statusCode': 200,
        'headers': {
            'Content-Type': 'application/json',
            'Access-Control-Allow-Origin': '*'
        },
        'body': json.dumps({'message': 'Registration Successful'})
    }
```

---

5. Provide HTML, CSS and JavaScript file:

Item.html

```html
<!DOCTYPE html>
<html>
<head>
    <title>Registration Form</title>
    <link rel="stylesheet" href="style.css">
</head>
<body>
    <div class="container">
        <h1>Registration Form</h1>
        <form>
            <label for="name">Name</label>
            <input type="text" id="name" name="name" required>
```

```
        <label for="email">Email</label>
        <input type="email" id="email" name="email" required>

        <label for="phone">Phone</label>
        <input type="tel" id="phone" name="phone" pattern="[0-9]{10}" required>

        <label for="password">Password</label>
        <input type="password" id="password" name="password" required>

        <input type="submit" value="Submit" onclick="submitForm()">
      </form>
    </div>
    <script src="script.js"></script>
</body>
</html>
```

script.js

```
function submitForm() {
    event.preventDefault();

    // Get form data
    const name = document.getElementById('name').value;
    const email = document.getElementById('email').value;
    const phone = document.getElementById('phone').value;
    const password = document.getElementById('password').value;

    // Create request object
    const xhr = new XMLHttpRequest();

    // Set up request
    xhr.open('POST', '<invokeurl>', true);
    xhr.setRequestHeader('Content-Type', 'application/json');

    // Set up response handler
    xhr.onreadystatechange = function () {
      if (xhr.readyState === XMLHttpRequest.DONE) {
```

```
        if (xhr.status === 200) {
            alert('Registration successful!');
            document.getElementById('name').value = '';
            document.getElementById('email').value = '';
            document.getElementById('phone').value = '';
            document.getElementById('password').value = '';
        } else {
            alert('Registration failed: ' + xhr.responseText);
        }
    }
};

// Send request
xhr.send(JSON.stringify({
    name: name,
    email: email,
    phone: phone,
    password: password
}));
}
```

style.css

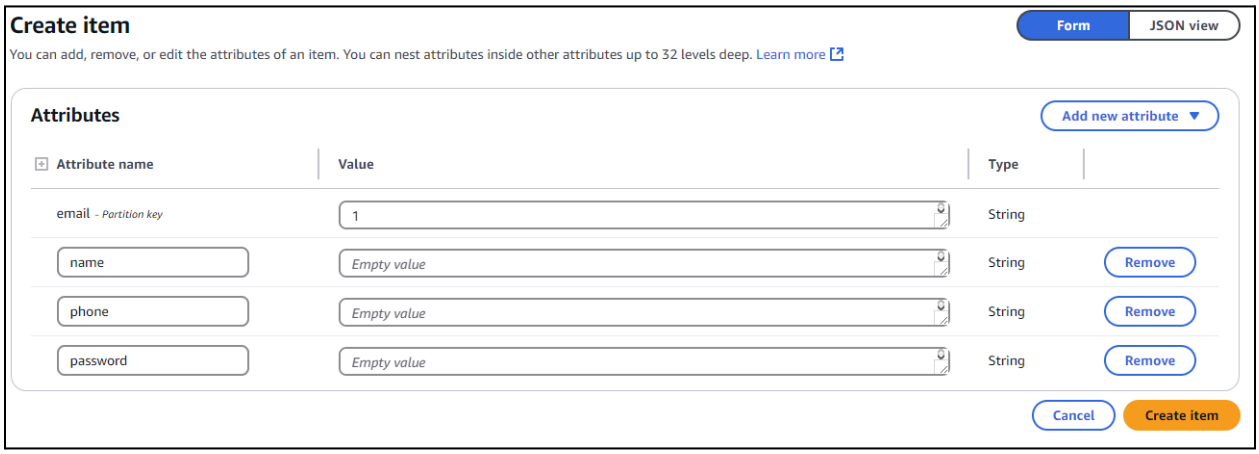
```
.container {
    max-width: 400px;
    margin: auto;
    padding: 10px;
}

form {
    display: flex;
    flex-direction: column;
}

label {
    margin-top: 10px;
}
```

```
input[type="submit"] {
    margin-top: 20px;
}
```

---

6. Create items returned in DynamoDB

In DynamoDB, edit an item in the table registration-table we created before. We will create 4 attributes for input as email (default value 1), name (default as Empty), phone (default as Empty), password (default as Empty).

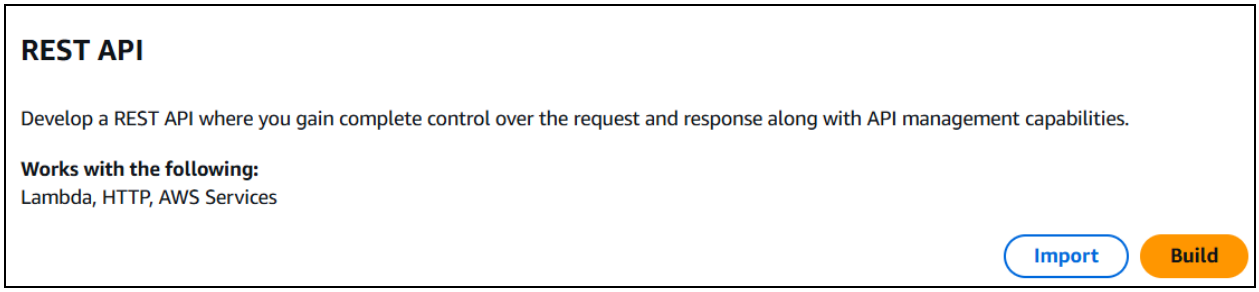


After create item, it will show format of items returned in format as:

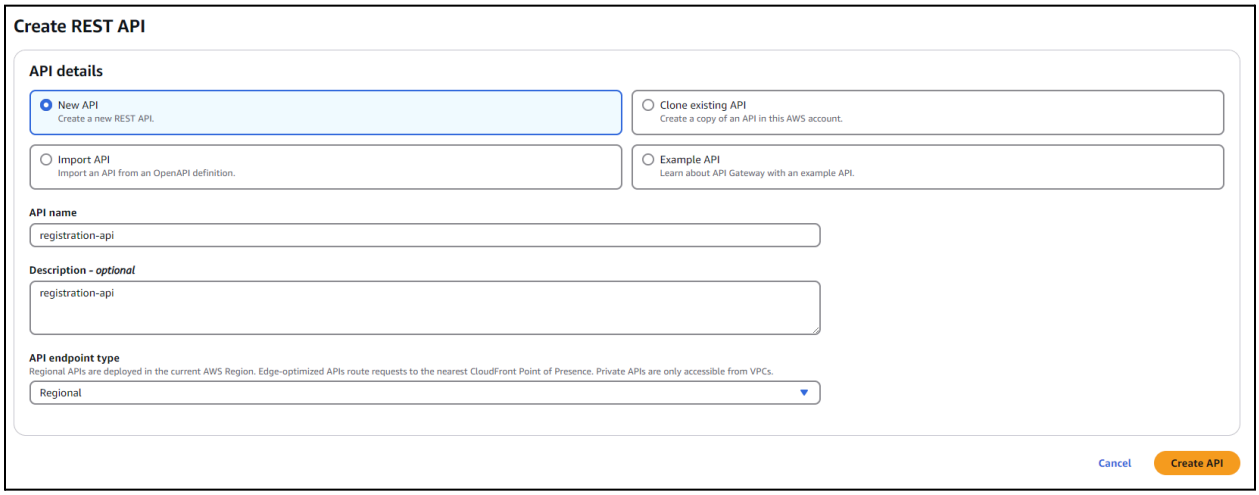
---

7. Create API Gateway and enable CORS

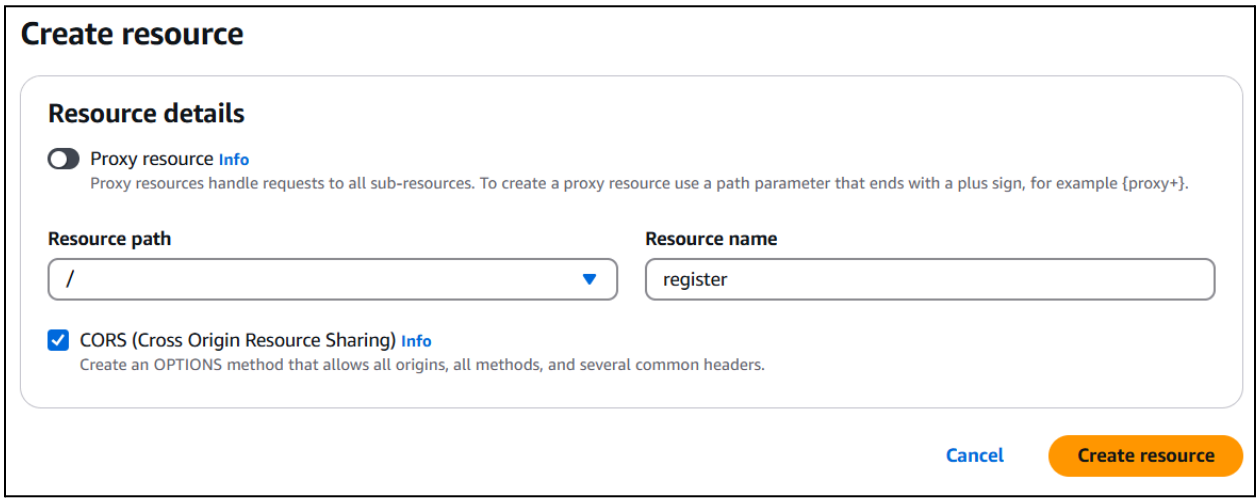
Next, reach out to API Gateway, choose to build REST API



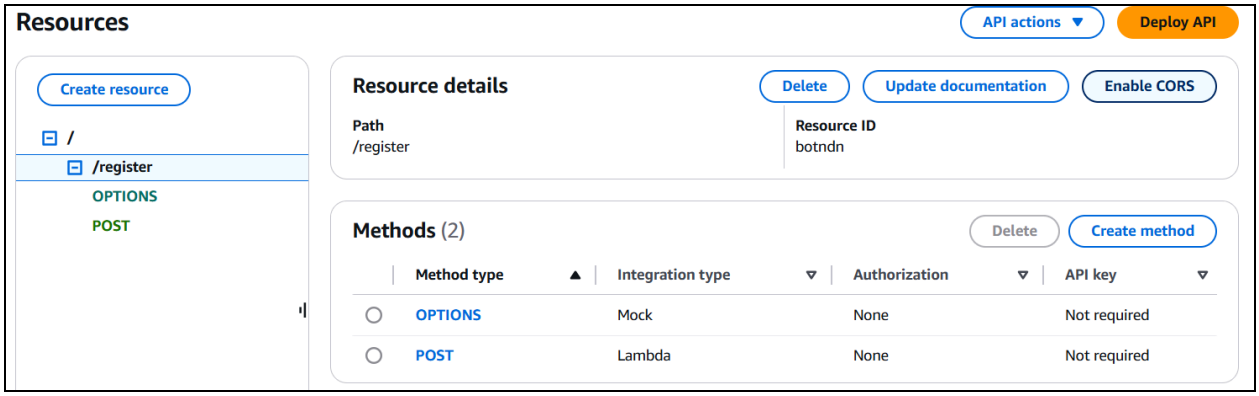
Create REST API named as registration-api



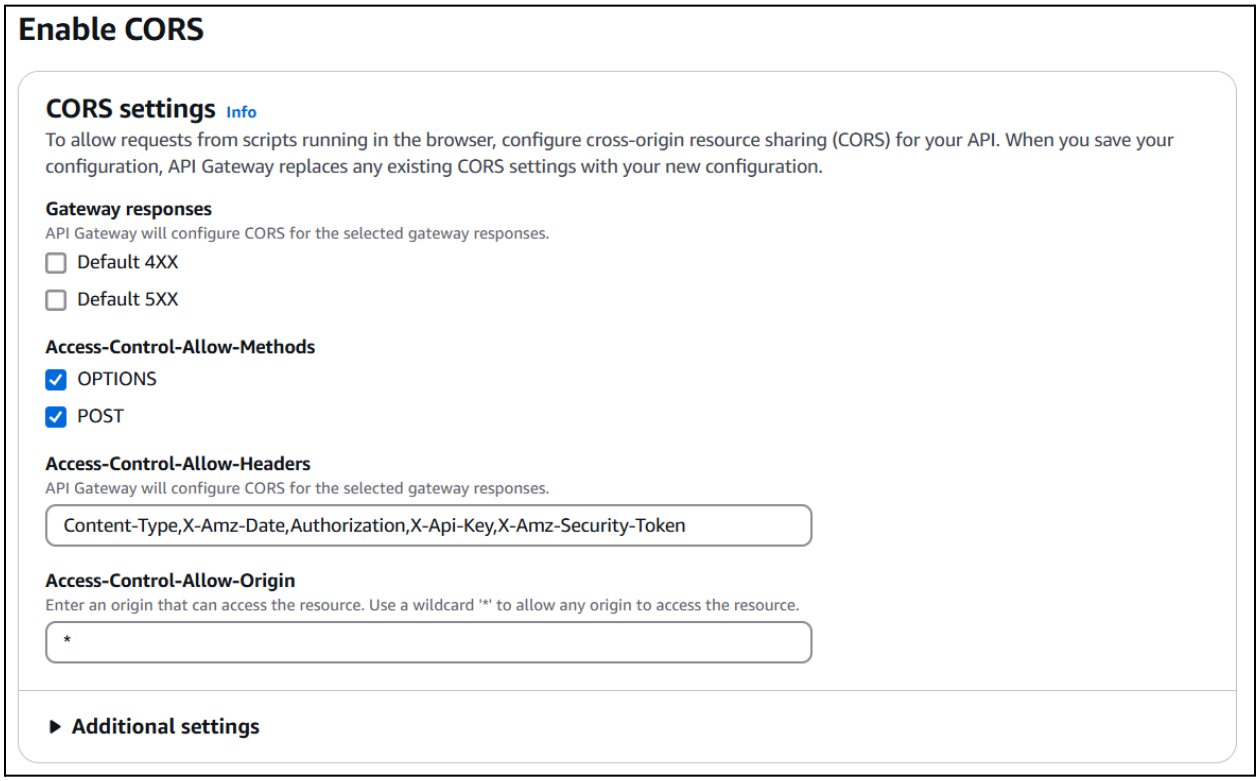
Create resource with resource name as register.



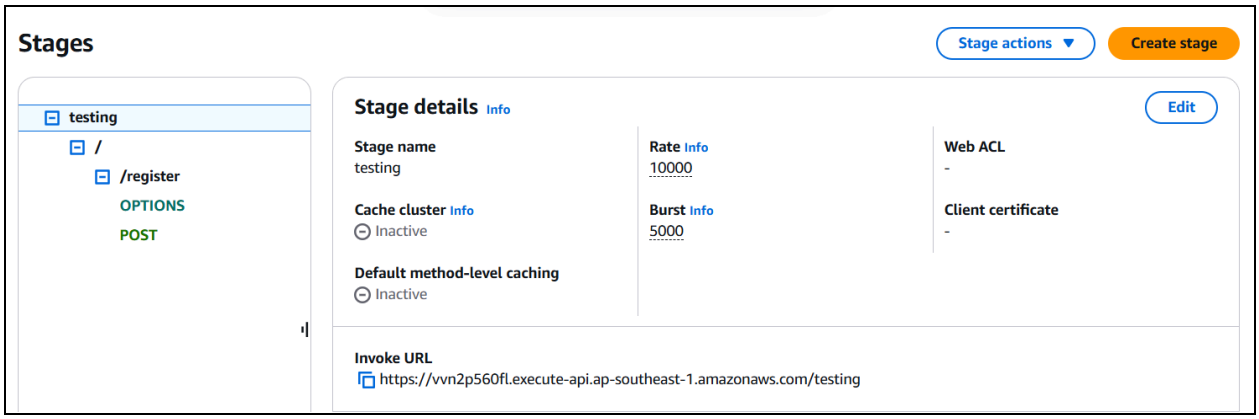
We will enable CORS for the resource register. Choose resource /register and choose Enable CORS.



In the Enable CORS console, select two Access-Control-Allow-Method as OPTIONS and POST.



Next, create new stage to deploy API. Name the stage as testing.



It will then genate an Invoke URL
https://vvn2p560fl.execute-api.ap-southeast-1.amazonaws.com/testing

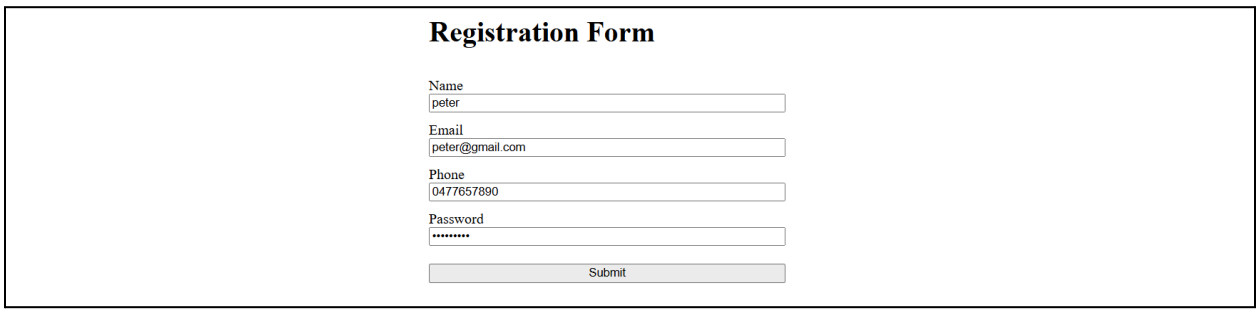
We will then include it in ***script.js*** so that when we post a request with attributes, it will update records into DynamoDB table.

```
// Set up request
xhr.open('POST', 'https://vvn2p560fl.execute-api.ap-southeast-1.amazonaws.com/testing/register', true);
xhr.setRequestHeader('Content-Type', 'application/json');
```

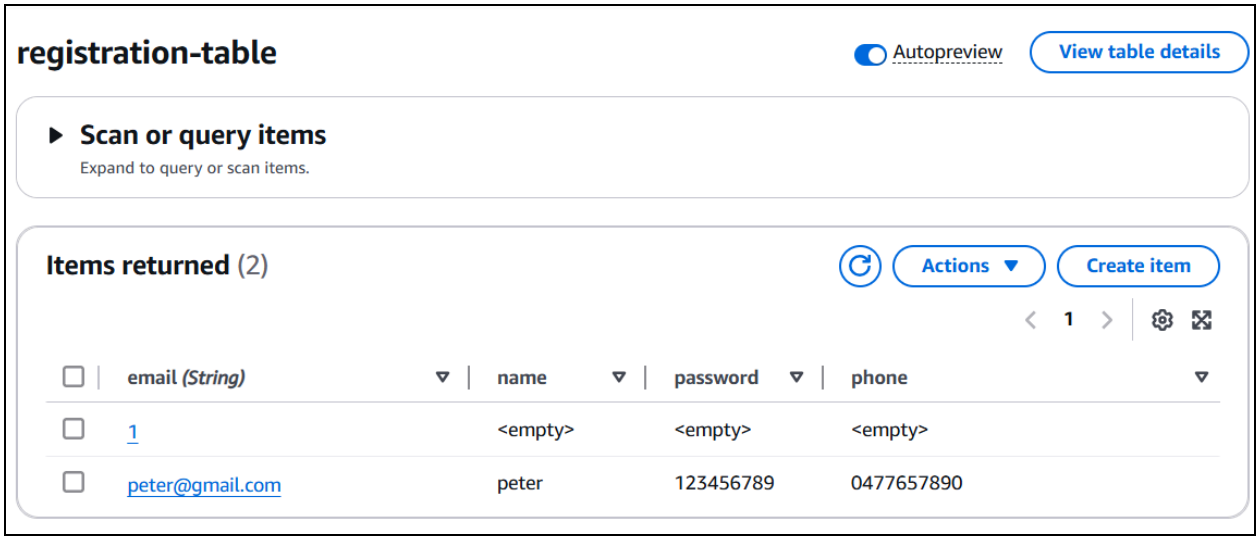
---

8. Test the project

We will then access to index.html. We will input information so that it will register into DynamoDB table. Then we click submit. Once it get successful, it will shows alert as Registration successful.



Then check on registration table in DynamoDB, we will see there is record of the new input.Meaning that we successfully retrieve record from registration form successfully.


registration-table
Autopreview
View table details
Scan or query items
Expand to query or scan items.
Items returned (2)
Actions
Create item
1
email (String)
name
password
phone
1
<empty>
<empty>
<empty>
peter@gmail.com
peter
123456789
0477657890