

TS-SDP4 PROJECT-3

190030746

k.pujitha

Design a Project on Database for a Mobile Application with Amazon DynamoDB Implementation:

The first screenshot shows the 'Method Execution' view for the POST method on the resource /postcustomerdetails. The flow is as follows:

- Client** sends a **Method Request** to the **API Gateway**.
- The **API Gateway** sends an **Integration Request** to the **Mock Endpoint**.
- The **Mock Endpoint** returns an **Integration Response** to the **API Gateway**.
- The **API Gateway** returns a **Method Response** to the **Client**.

The **Method Request** details are:

- Auth: NONE
- ARN: arn:aws:execute-api:us-east-1:190030746:api/pujitha/postcustomerdetails

The **Integration Request** details are:

- Type: MOCK

The **Method Response** details are:

- HTTP Status: 200
- Model: application/json => Empty

The **Integration Response** details are:

- HTTP status pattern: *
- Output passthrough: yes

The second screenshot shows the 'Integration Response' configuration for the POST method. It displays a table for mapping HTTP status regex to Method response status and Output model.

HTTP status regex	Method response status	Output model	Default mapping
.*	200	YES	YES

Below the table, there are sections for 'Header Mappings' and 'Mapping Templates'. The 'Mapping Templates' section shows a template for 'application/json' with a 'Generate template' button.

application/json
{ "StatusCode" : "200" }

The image shows two screenshots of the Amazon API Gateway console. The top screenshot displays the 'New Child Resource' page for the API 'PostCustomerDetails (postcustp)'. The 'Resource Name' is 'postCustomerDetails' and the 'Resource Path' is '/postcustomerdetails'. The 'Enable API Gateway CORS' checkbox is checked. The bottom screenshot shows the 'Method Test' page for the same API and resource. The 'Method' is 'POST'. The 'Request' tab is selected, showing the 'Request Headers' and 'Request Body'. The 'Request Headers' section shows 'Status: 200' and 'Response body'. The 'Request Body' section shows a JSON object: { "StatusCode": "200" }. The 'Log' section shows a list of log entries for the request.

New Child Resource

Use this page to create a new child resource for your resource.

Configure as (proxy) resource

☐

Resource Name
postCustomerDetails

Resource Path
/postcustomerdetails

You can add path parameters using brackets. For example, the resource path {username} represents a path parameter called 'username'. Configuring (proxy-) as a proxy resource catches all requests to its sub-resources. For example, it would for a GET request to /foo. To handle requests to /, add a new API method on the / resource.

Enable API Gateway CORS ☒

*** Required**

Cancel **Create Resource**

Method Test

Make a test call to your method. After you make a test call, API Gateway ships information and details to help you test your method.

Path
/postcustomerdetails

No path parameters exist for this resource. You can define path parameters by using the syntax {parameterName} in a resource path.

Query String
[postcustomerdetails]

Headers
[postcustomerdetails]

Test a method that accepts HTTP headers and values, and can return custom headers, cookies, and custom response bodies.

Stage Variables
No stage variables exist for this method.

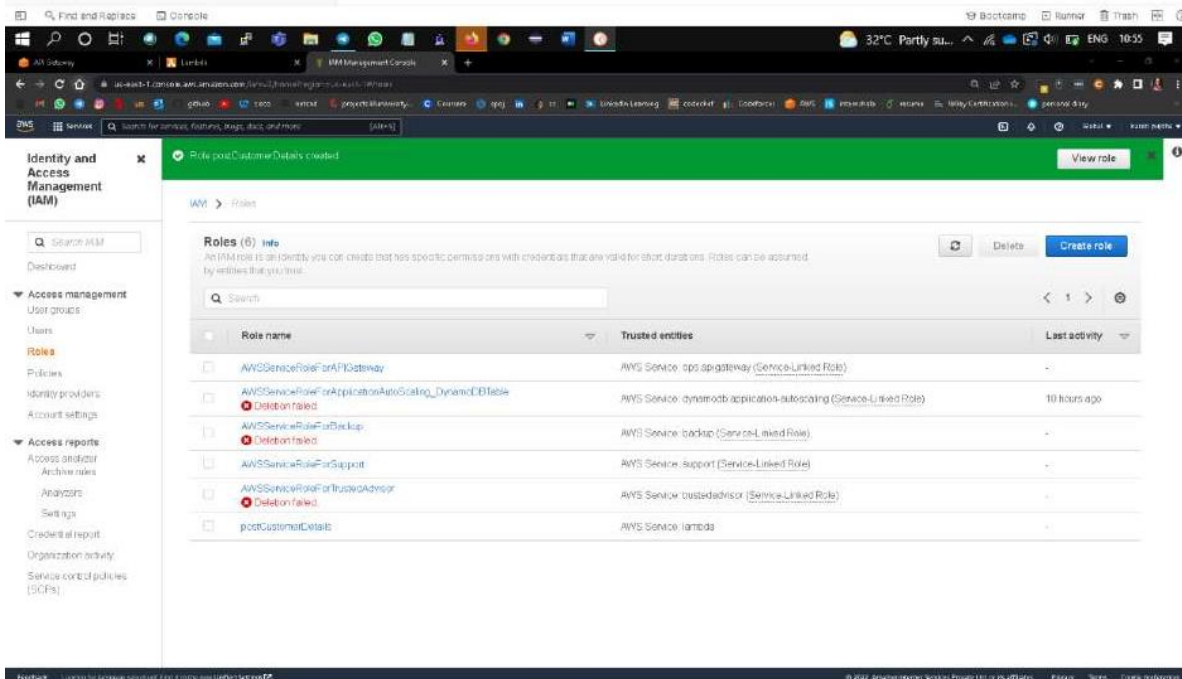
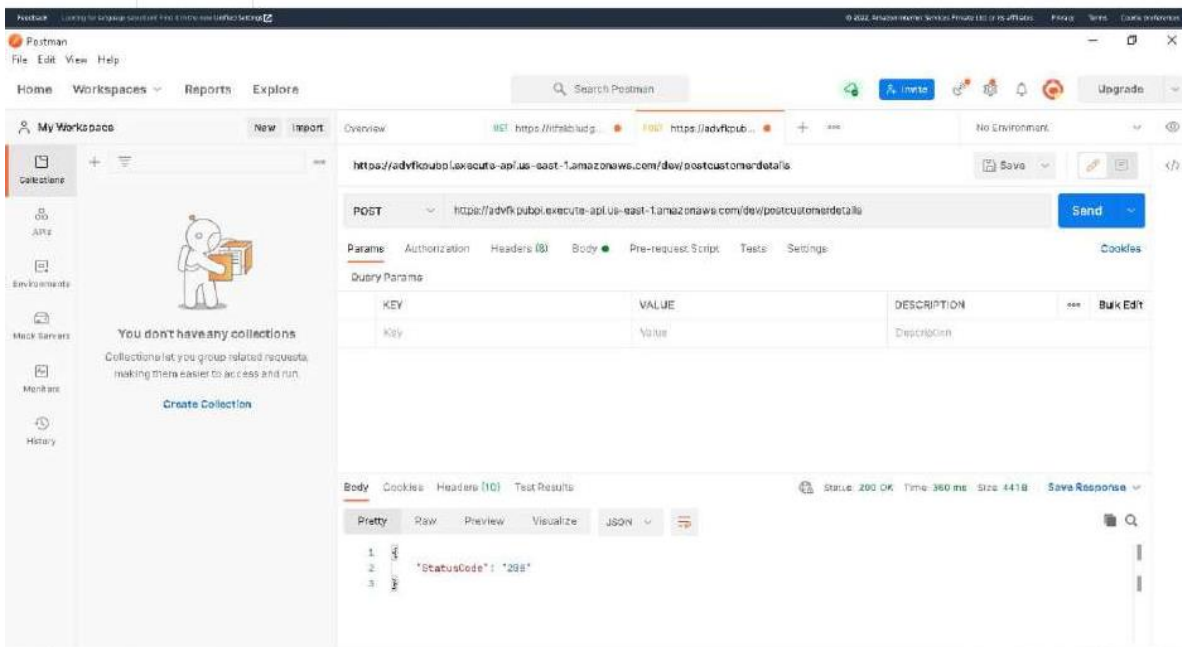
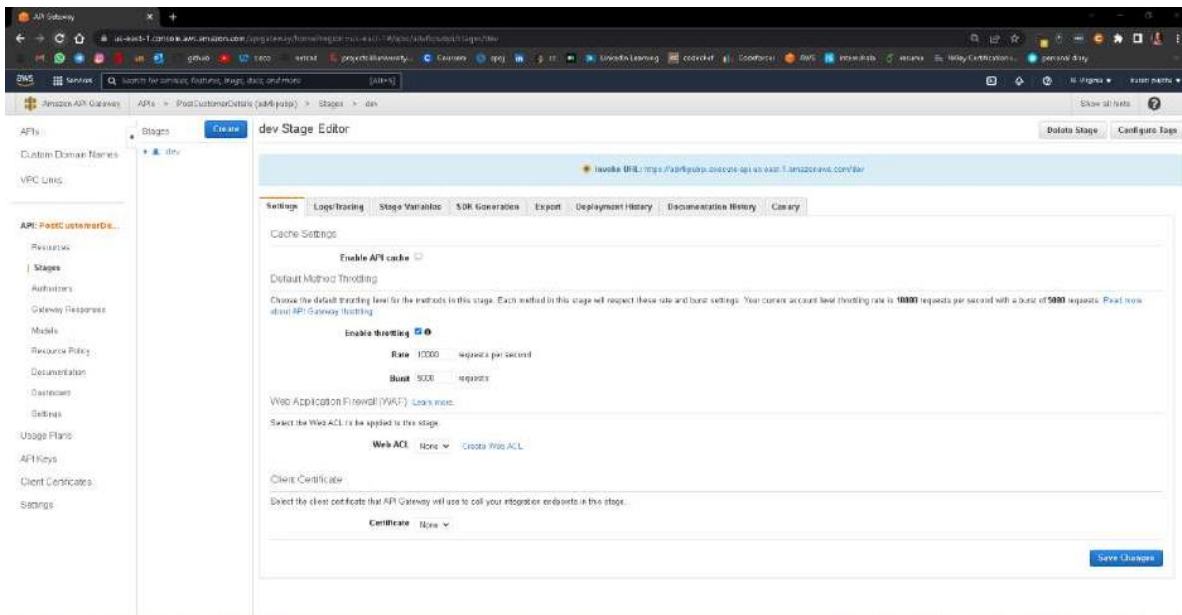
Client Certificate
No client certificate has been specified.

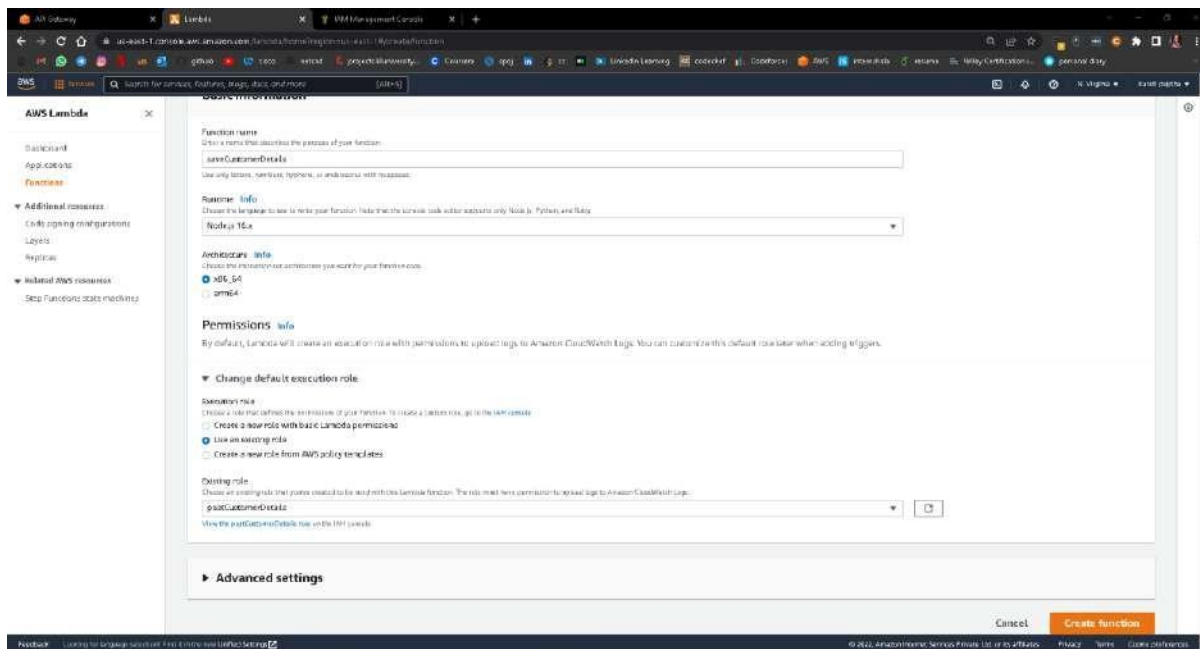
Request Body
[postcustomerdetails]

Request Headers
Status: 200
Response body

Response Headers
[postcustomerdetails]

Log
[postcustomerdetails]





- var

AWS = require("aws-sdk");

```
var docClient = new AWS.DynamoDB.DocumentClient();
exports.handler = (event, context, callback) => {
  var tableName = "CustomerDetails";
  var params = {
    TableName: tableName,
    Item: {
      "EmailID": event.EmailID,
      "FirstName": event.FirstName,
      "LastName": event.LastName
    }
  };
  docClient.put(params, function(err, data) {
    if(err) {
      callback(err)
    } else {
      callback(null, "Sucessfully Updated Data")
    }
  })
}
```

Partition key
The partition key is part of the table's primary key. It is a field value that is used to retrieve items from your table and allocate data across items for search by and available fields.

EmailID

Sort key - optional
You can set a sort key as the second part of a table's primary key. The sort key allows you to sort and search items within all items sharing the same partition key.

Settings

☒ **Default settings**
The Partition key is created from ID. It is recommended to use this setting for most of your tables. It is recommended to use this setting for most of your tables.

☐ **Custom settings**
Use these settings to create a table with a custom Partition key.

Default settings

Read/write capacity [Info](#)
Using provisioned capacity, read and write capacity are set to 5 units each with auto scaling enabled.

Secondary indexes [Info](#)
No secondary indexes have been created. Queries will be run by using the table's partition key and sort key only.

Key management for encryption at rest [Info](#)
Using the AWS-owned key. This key is managed by DynamoDB at no extra cost.

Table class
Using DynamoDB Standard table class. The default general purpose table class. Recommended for the vast majority of tables that store frequently accessed data, with throughput (reads and writes) as the dominant table cost.

Tags

CustomerDetails

Tables (1)

Any table name

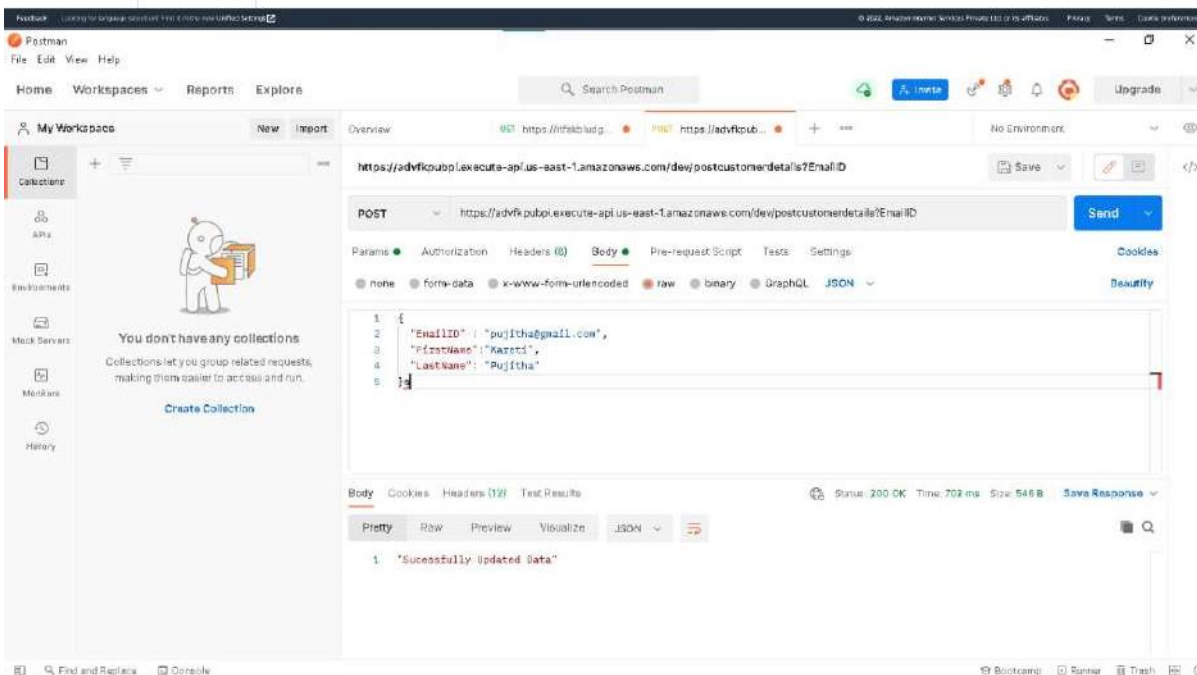
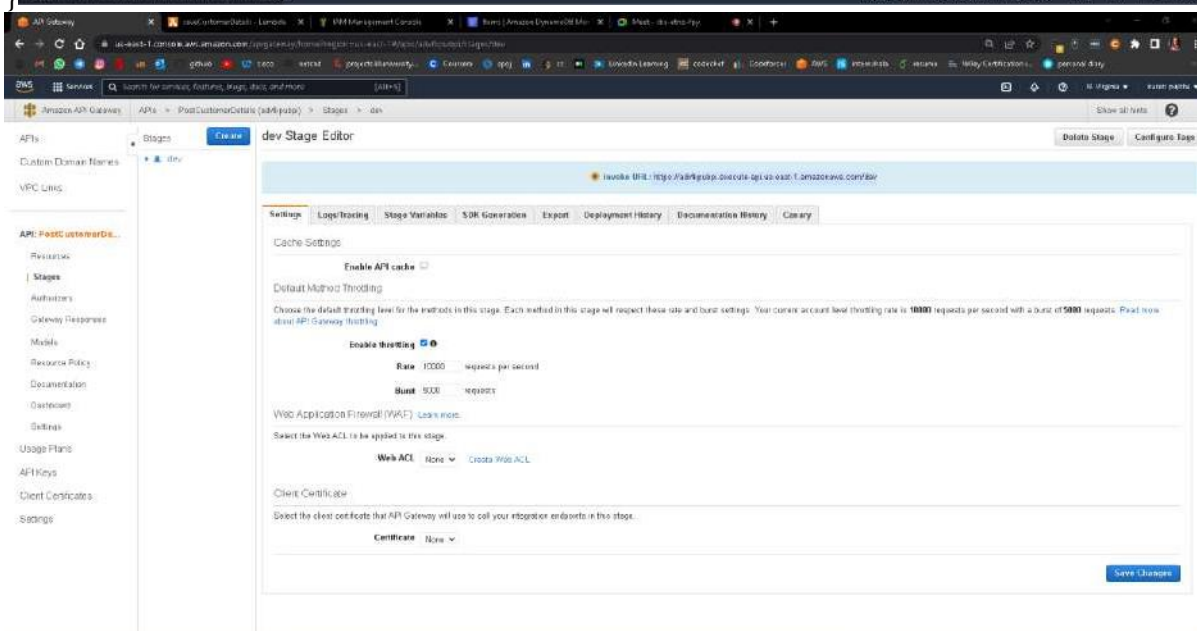
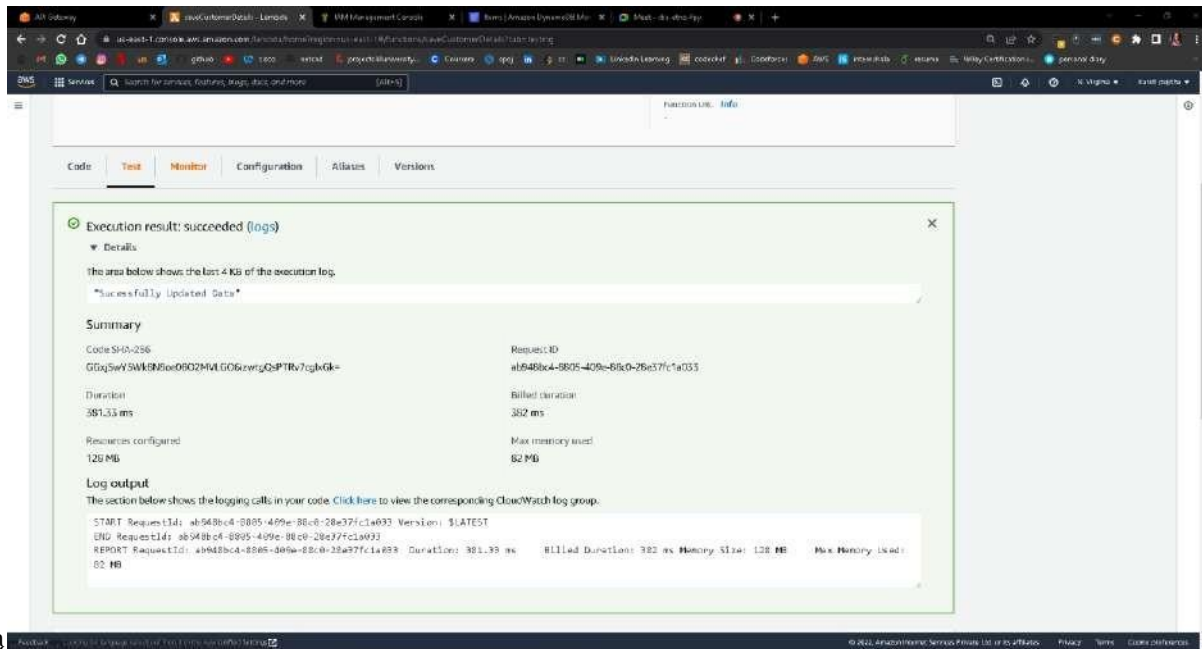
Table or index: CustomerDetails

Filters

Items returned (2)

EmailID	FirstName	LastName
email2@gmail.com	puja	pujitha
email3@gmail.com	puja	pujitha

"EmailID": "pujitha@gmail.com",
 "FirstName": "Kareti",
 "LastName": "Pujitha"



<https://advfkpubpi.execute-api.us-east-1.amazonaws.com/dev/postcustomerdetails>

```
index.html
<script>
    (function() {
        var CustomerDetailsModel = function() {
            var self = this;

            self.fullName = ko.observable("");
            self.firstName = ko.observable("");
            self.lastName = ko.observable("");
            self.userName = ko.observable("");

            self.newCustomerDetails = function() {
                var CustomerDetail = {
                    fullName: self.fullName(),
                    firstName: self.firstName(),
                    lastName: self.lastName()
                };

                $.ajax({
                    url: "https://cd4v3pubj-aws-ops-east-1.amazonaws.com/dev/newcustomerdetails",
                    cache: false,
                    type: "POST",
                    data: JSON.stringify(CustomerDetail),
                    success: function(data) {
                        self.userName(data.userName);
                        self.firstName(data.firstName);
                        self.lastName(data.lastName);
                    }
                });
            }
        };

        $(document).ready(function() {
            // ... (jQuery code for form handling) ...
        });
    })();

```



AWS Policy Generator

The AWS Policy Generator is a tool that enables you to create policies that control access to Amazon Web Services (AWS) products and resources. For more information about creating policies, see [Key Concepts in Using AWS Identity and Access Management](#). Here are sample policies.

Step 1: Select Policy Type

A policy is a container for permissions. The different types of policies you can create are an IAM Policy, an S3 Bucket Policy, an IAM Role Policy, a VPC Endpoint Policy, and an SQS Queue Policy.

Select Type of Policy: S3 Bucket Policy

Step 2: Add Statement(s)

A statement is the formal description of a single permission. See a description of elements that you can use in statements.

Effect: ☒ Allow ☐ Deny

Principal:

AWS Service: S3 ☐ All Services (**)

Actions: arn:aws:s3::* ☐ All Actions (**)

Amazon Resource Name (ARN): arn:aws:s3:::codesavecustomer

Step 3: Generate Policy

A policy is a document, written in the Access Policy Language, that acts as a container for one or more statements.

Add one or more statements above to generate a policy.

Amazon S3

- Buckets
- Access Points
- Object Lambda Access Points
- Multi-Region Access Points
- Batch Operations
- Access analyzer for S3
- Block Public Access settings for this account
- Storage Lens
- Documents
- AWS Organizations settings
- Feature sacRight
- AWS Marketplace for S3

Bucket policy

The bucket policy, written in JSON, provides access to the objects stored in the bucket. Bucket policies don't apply to objects owned by other accounts. [Learn more](#)

Bucket ARN: `arn:aws:s3:::codesavecustomer`

Policy

```
1 {
2   "Statement": [
3     {
4       "Sid": "Statement1",
5       "Effect": "Allow",
6       "Principal": "*",
7       "Action": "s3:*",
8       "Resource": "arn:aws:s3:::codesavecustomer/*"
9     }
10  ]
11 }
```

Edit statement **Stmt16524265274**

1. Add actions

Choose a service

Included

- S3
- Available
- AMP
- API Gateway
- API Gateway V2
- Access Analyzer
- Account
- Activate
- Alexa for Business

2. Add a resource

Feedback

Looking for something else? Visit [AWS IAM](#) in the [AWS IAM console](#)

© 2022 Amazon Web Services, Inc. or its affiliates. [Privacy](#) [Terms](#) [Cookie Preferences](#)

Upload succeeded
View details below

Summary

Destination <code>s3://codesavecustomer</code>	Succeeded 3 files, 145.8 KB (100.00%)	Failed 0 files, 0 B (0%)
---	--	-----------------------------

Files and folders

Configuration

Files and folders (5 Total, 145.8 KB)

Filter by name

Name	Folder	Type	Size	Status	Error
index.html	-	text/html	2.2 KB	Succeeded	-
jquery-5.1.1.min.js	-	text/javascript	84.7 KB	Succeeded	-
knockout-5.6.2.js	-	text/javascript	58.9 KB	Succeeded	-

Success! With static website hosting.

Learn how to effectively use the S3 Storage Classes.

Requester pays

When enabled, the requester pays for requests and data transfer costs, and anonymous access to this bucket is disabled. [Learn more](#)

Requester pays
Disabled

Static website hosting

Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting
Enabled

Hosting type
Bucket hosting

Bucket website endpoint

When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)

<http://codesavecustomer.s3-website-us-east-1.amazonaws.com>

DynamoDB

CustomerDetails

Tables (1)

Any table - eng

Find table by table name

CustomerDetails

CustomerDetails

Scan/Query items

Expand to query or scan items.

Items returned (2)

	CreatedAt	FirstName	LastName
<input type="checkbox"/>	pythonagg	Karen	Wright
<input type="checkbox"/>	123	Jim	J

1:03 PM | 0.9KB/s



codesavecustomer.s3-website



EmailID: swathi@gmail.com

FirstName: Swathika

LastName: Sakhamuri

Save



1:03 PM | 4.2KB/s



codesavecustomer.s3-website



EmailID:

FirstName:

LastName:

Sucessfully Updated Data

Save



Amazon DynamoDB

Tables

Update settings

Explore items

RunSQL editor

Backups

Export to CSV

General capacity

DAX

Clusters

Subnet groups

Parameter groups

Events

Tell us what you think

Access to the previous console experience

Display settings

DynamoDB

Items

CustomerDetails

Any table + key

Find items by table name

CustomerDetails

CustomerDetails

Scan/Query items

Expand to query or scan items

Items returned (3)

	EmailID	FirstNames	LastNames
<input type="checkbox"/>	prashantg...	Karan	Wadia
<input type="checkbox"/>	sanika.g...	Sanika	Siddhant
<input type="checkbox"/>	725	pa	j

Feedback

Looking for help? Search for errors and helpful content

© 2022 Amazon Web Services, Inc. or its affiliates. Privacy Terms Cloud Performance