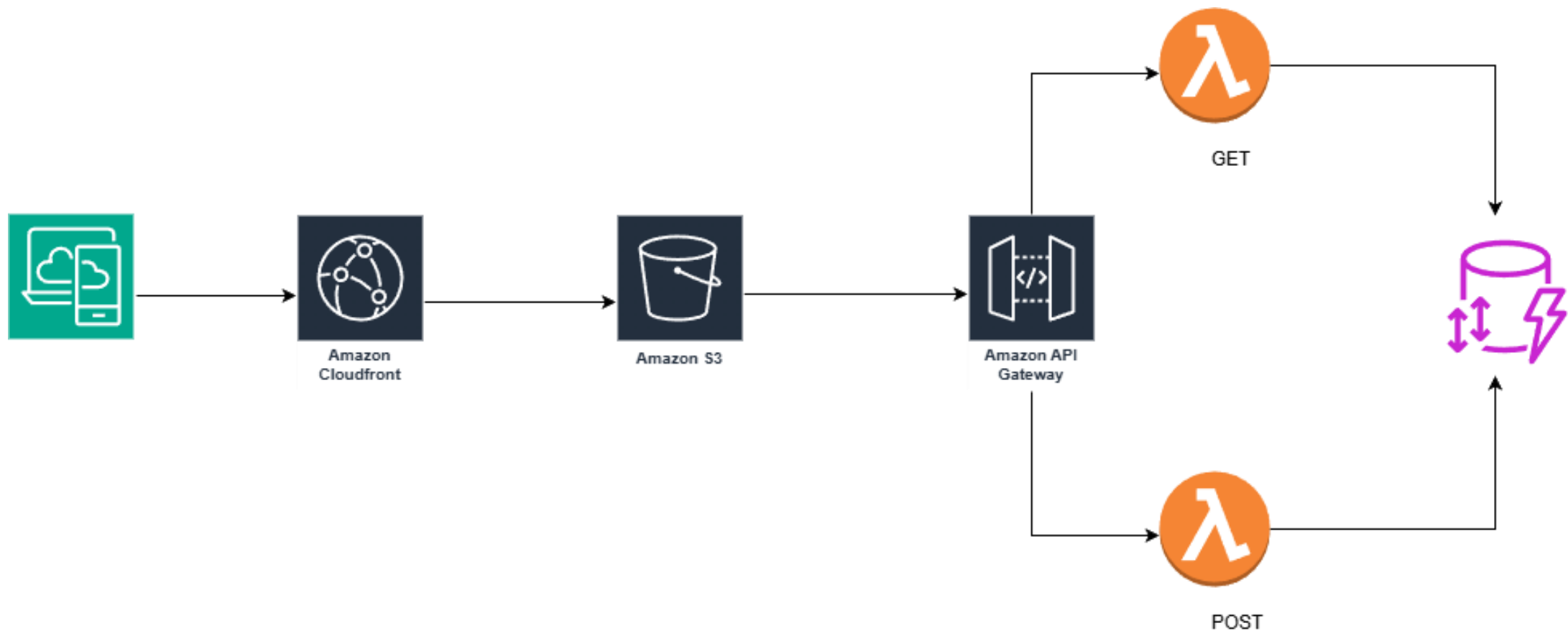


# Deploying a Serverless Web Application on AWS Using S3, CloudFront, API Gateway, Lambda, DynamoDB, and Custom Domain Integration



# AGENDA

- Deploy Lambda Functions for GET and POST
- Set Up DynamoDB Table
- Configure API Gateway
- Host Frontend on S3
- Set Up CloudFront CDN
- Troubleshooting & Common Issues



# Set Up DynamoDB Table

- Open AWS Console → search “DynamoDB”
- Click “Create table”
- Enter Table Name
- Set Partition Key (e.g., EmployeeID – String)
- Leave Sort Key OFF (optional)
- In Table settings, keep Default settings
- Keep Table class as “DynamoDB Standard”
- Click “Create table”

# Set Up DynamoDB Table

## Create table

### Table details [Info](#)

DynamoDB is a schemaless database that requires only a table name and a primary key when you create the table.

#### Table name

This will be used to identify your table.

Between 3 and 255 characters, containing only letters, numbers, underscores (`_`), hyphens (`-`), and periods (`.`).

#### Partition key

The partition key is part of the table's primary key. It is a hash value that is used to retrieve items from your table and allocate data across hosts for scalability and availability.



1 to 255 characters and case sensitive.

#### Sort key - *optional*

You can use a sort key as the second part of a table's primary key. The sort key allows you to sort or search among all items sharing the same partition key.



1 to 255 characters and case sensitive.

# Set Up DynamoDB Table

## Table settings

### ☒ Default settings

The fastest way to create your table. You can modify most of these settings after your table has been created. To modify these settings now, choose 'Customize settings'.

### ☐ Customize settings

Use these advanced features to make DynamoDB work better for your needs.

## Default table settings

These are the default settings for your new table. You can change some of these settings after creating the table.


Setting	Value	Editable after creation
Table class	DynamoDB Standard	Yes
Capacity mode	On-demand	Yes
Maximum read capacity units	-	Yes
Maximum write capacity units	-	Yes
Local secondary indexes	-	No
Global secondary indexes	-	Yes
Encryption key management	AWS owned key	Yes
Deletion protection	Off	Yes
Resource-based policy	Not active	Yes

# Set Up DynamoDB Table

- DynamoDB**
- Dashboard
- Tables**
- Explore items
- PartiQL editor
- Backups
- Exports to S3
- Imports from S3
- Integrations
- Reserved capacity
- Settings

✓ The EmployeeData table was created successfully. ✕

**Tables (1)** [Info](#) Last updated November 20, 2025, 16:50 (UTC+8:00)  Actions ▾ Delete Create table

Filter by tag Any tag key ▾ Filter by tag value Any tag value ▾ < 1 > 

<input type="checkbox"/>	Name	Status	Partition key	Sort key	Indexes	Replication Regions	Deletion protection	Favorite	Read capacity mode	Write capacity mode	Total size	Table class
<input type="checkbox"/>	<a href="#">EmployeeData</a>	✓ Active	EmployeeID (S)	-	0	0	⊖ Off	☆	On-demand	On-demand	0 bytes	Standard

# Deploy Lambda Functions

- Go to AWS Console → Open Lambda
- Click Create function
- Select Author from scratch
- Enter function name
- Choose runtime (Python/Node.js, etc.)
- Select or create an execution role
- Click Create function
- Add code → Deploy → Test



# Deploy Lambda Functions for GET and POST

## Creating a Lambda Function – getEmployeeData

- **Function Name:** getEmployeeData
- **Runtime:** Python 3.9
- **Architecture:** x86\_64
- **Execution Role:** Custom IAM Role
  - Create role in IAM console
  - Attach the role to Lambda
  - Ensure the role has permissions for CloudWatch logging and DynamoDB access
- **Click on Create Functions**

# Deploy Lambda Functions for GET and POST

## Basic information

### Function name

Enter a name that describes the purpose of your function.

getEmployeeData

Function name must be 1 to 64 characters, must be unique to the Region, and can't include spaces. Valid characters are a-z, A-Z, 0-9, hyphens (-), and underscores (\_).

### Runtime [Info](#)

Choose the language to use to write your function. Note that the console code editor supports only Node.js, Python, and Ruby.

Python 3.9



### Architecture [Info](#)

Choose the instruction set architecture you want for your function code.

☐ arm64

☒ x86\_64

### Permissions [Info](#)

By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

## ▼ Change default execution role

### Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions

☒ Use an existing role

☐ Create a new role from AWS policy templates

### Existing role

Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

LambdaDynamoDBRole



[View the LambdaDynamoDBRole role](#) on the IAM console.

# Deploy Lambda Functions for GET and POST

## Deploying Python Code on AWS Lambda (Serverless Deployment)

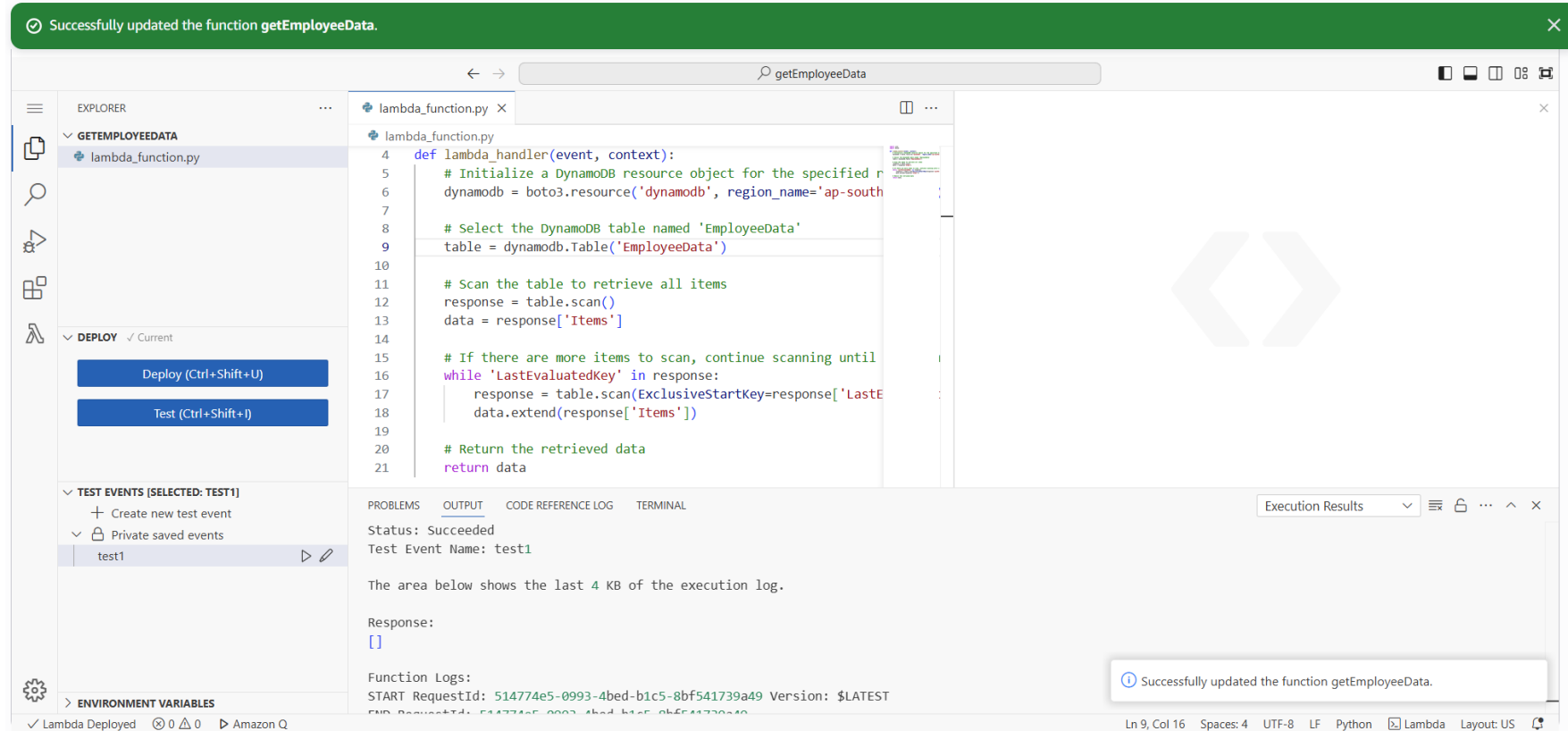
The screenshot displays the AWS Lambda console for a function named 'getEmployeeData'. The top section, 'Function overview', includes buttons for 'Throttle', 'Copy ARN', 'Actions', 'Export to Infrastructure Composer', and 'Download'. It shows the function's description, last modified time (2 minutes ago), and its ARN. Below this, the 'Code source' section is active, showing a code editor with the following Python code:

```
1 import json
2 import boto3
3
4 def lambda_handler(event, context):
5     # Initialize a DynamoDB resource object for the specified
6     dynamodb = boto3.resource('dynamodb', region_name='ap-southeast-1')
7
8     # Select the DynamoDB table named 'EmployeeData'
9     table = dynamodb.Table('EmployeeData')
10
11     # Scan the table to retrieve all items
12     response = table.scan()
13     data = response['Items']
14
15     # If there are more items to scan, continue scanning until
16     while 'LastEvaluatedKey' in response:
17         response = table.scan(ExclusiveStartKey=response['LastEvaluatedKey'])
18         data.extend(response['Items'])
19
20     return data
```

The console also shows the 'EXPLORER' sidebar with the file 'lambda\_function.py' selected, and the 'DEPLOY' section with buttons for 'Deploy (Ctrl+Shift+U)' and 'Test (Ctrl+Shift+I)'.

# Deploy Lambda Functions for GET and POST

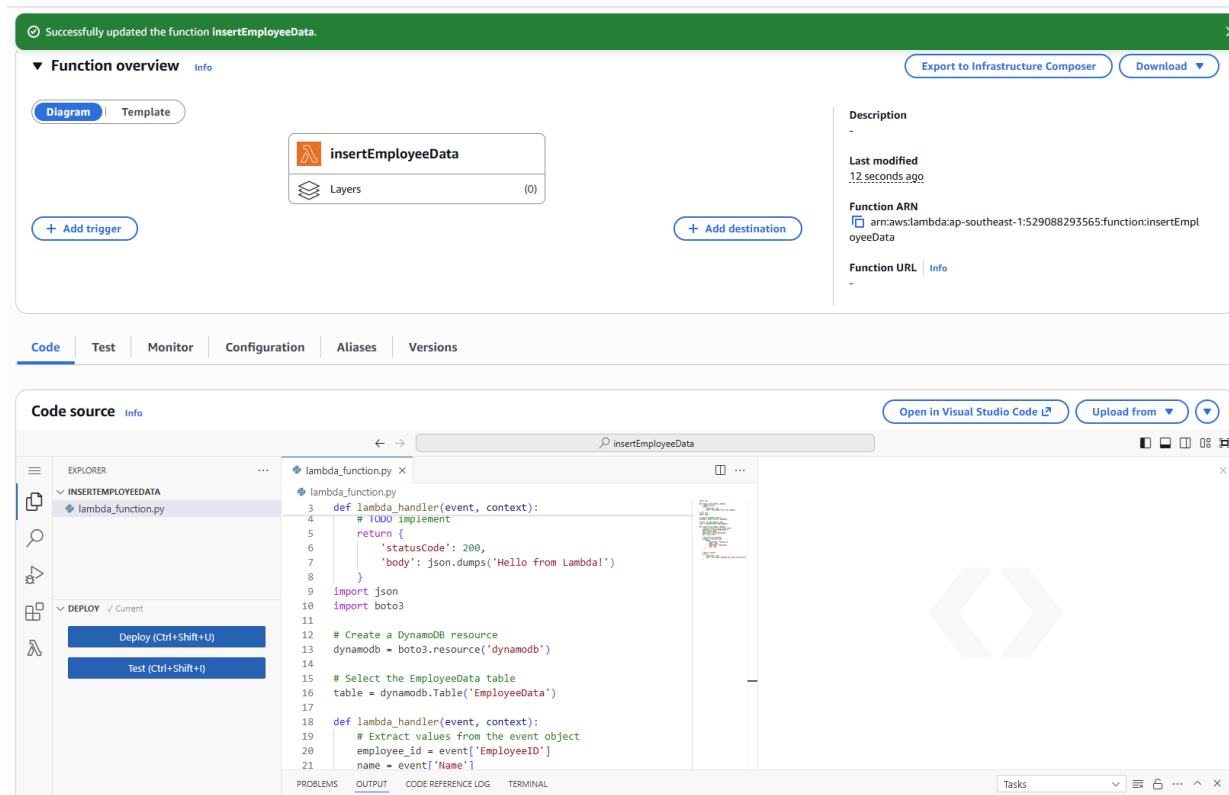
## Lamda Test Execution



# Deploy Lambda Functions for GET and POST

## Creating a Lambda Function – insertEmployeeData

*Remark: Follow the steps on pages 8–11 to create the insertEmployeeData Lambda function*



# Configure API Gateway

- Open AWS Console → search “API Gateway”
- Click “Create API”
- Choose “REST API (Build)”
- Select “New API” → Give API Name
- Click “Create API”
- Create a Resource
- Create a Method (GET or POST)
- Choose Integration Type: Lambda Function
- Select your Lambda → Click “Save”
- Deploy API → Create a new Stage (e.g., prod)
- Copy the Invoke URL and test your REST API

# Configure API Gateway

- Create an API and select REST API

Create REST API [Info](#)

**API details**

☒ **New API**  
Create a new REST API.

☐ **Clone existing API**  
Create a copy of an API in this AWS account.

☐ **Import API**  
Import an API from an OpenAPI definition.

☐ **Example API**  
Learn about API Gateway with an example API.

**API name**

EmployeeData

**Description - optional**

**API endpoint type**  
Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence. Private APIs are only accessible from VPCs.

Edge-optimized ▼

**IP address type** [Info](#)  
Select the type of IP addresses that can invoke the default endpoint for your API.

☒ **IPv4**  
Supports only edge-optimized and Regional API endpoint types.

☐ **Dualstack**  
Supports all API endpoint types.

[Cancel](#) [Create API](#)

# Configure API Gateway


- Create a GET method


**Create method**


**Method details**


Method type  
GET


**Integration type**

☒ **Lambda function**  
Integrate your API with a Lambda function.  


☐ **HTTP**  
Integrate with an existing HTTP endpoint.  


☐ **Mock**  
Generate a response based on API Gateway mappings and transformations.  


☐ **AWS service**  
Integrate with an AWS Service.  


☐ **VPC link**  
Integrate with a resource that isn't accessible over the public internet.  


☒ **Lambda proxy integration**  
Send the request to your Lambda function as a structured event.

**Response transfer mode** | [Info](#)

☒ **Buffered**  
Wait to receive the complete response before beginning transmission.

☐ **Stream**  
Send portions of the response without waiting for the complete response.

**Lambda function**  
Provide the Lambda function name or alias. You can also provide an ARN from another account.

ap-southeast-1

**Grant API Gateway permission to invoke your Lambda function**  
When you save your changes, API Gateway updates your Lambda function's resource-based policy to allow this API to invoke it.

**Integration timeout** | [Info](#)  
By default, you can enter an integration timeout of 50 - 29,000 milliseconds. You can use Service Quotas to raise the integration timeout to greater than 29,000 ms

29000



# Configure API Gateway


- Create a PUT method


**Create method**


**Method details**


Method type  
PUT


**Integration type**

☒ **Lambda function**  
Integrate your API with a Lambda function.  


☐ **HTTP**  
Integrate with an existing HTTP endpoint.  


☐ **Mock**  
Generate a response based on API Gateway mappings and transformations.  


☐ **AWS service**  
Integrate with an AWS Service.  


☐ **VPC link**  
Integrate with a resource that isn't accessible over the public internet.  


☒ **Lambda proxy integration**  
Send the request to your Lambda function as a structured event.

**Response transfer mode** | [Info](#)

☒ **Buffered**  
Wait to receive the complete response before beginning transmission.

☐ **Stream**  
Send portions of the response without waiting for the complete response.

**Lambda function**  
Provide the Lambda function name or alias. You can also provide an ARN from another account.

ap-southeast-1

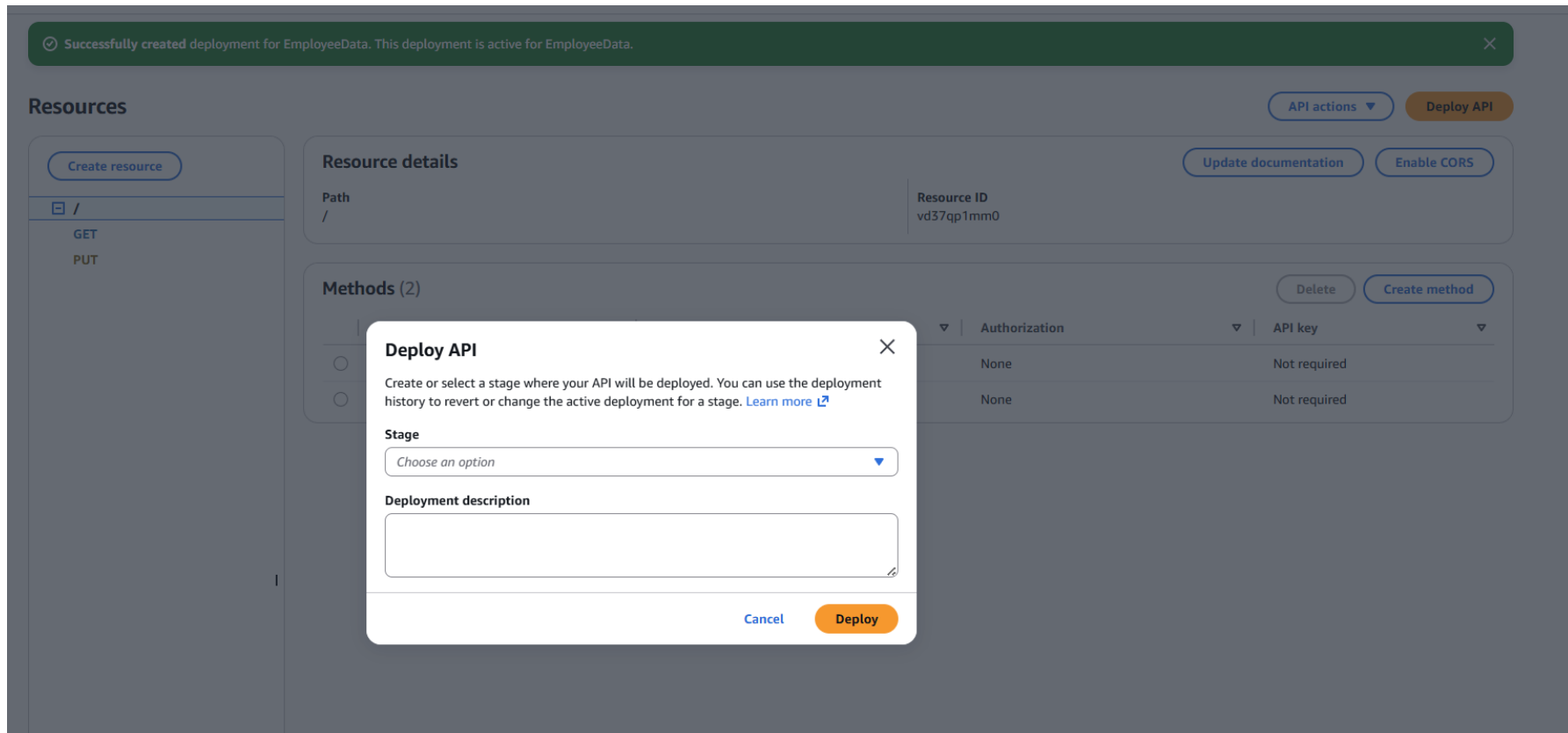
**Grant API Gateway permission to invoke your Lambda function**  
When you save your changes, API Gateway updates your Lambda function's resource-based policy to allow this API to invoke it.

**Integration timeout** | [Info](#)  
By default, you can enter an integration timeout of 50 - 29,000 milliseconds. You can use Service Quotas to raise the integration timeout to greater than 29,000 ms

29000

# Configure API Gateway

- Deploy the API to a stage



# Configure API Gateway

- API Deployed

Stages

EmployeeData

Stage details

Stage name

EmployeeData

Cache cluster

Inactive

Default method-level caching

Inactive

Invoke URL

<https://8rw6wh2h5k.execute-api.ap-southeast-1.amazonaws.com/EmployeeData>

Active deployment

412nwg on November 20, 2025, 18:16 (UTC+08:00)

Rate

10000

Burst

5000

Web ACL

-

Client certificate

-

Logs and tracing

CloudWatch logs

Inactive

X-Ray tracing

Inactive

Custom access logging

Inactive

Detailed metrics

Inactive

Data tracing

Inactive

Stage variables

Deployment history

Documentation history

Canary

Tags

Stage variables (0/0)

Find resources

Name

Value

19

# Host Frontend on S3

- Open AWS Console → search “S3”
- Create bucket → Uncheck “Block all public access”
- Upload your build files (index.html, JS)
- Go to Permissions → Add Bucket Policy to allow public read
- Go to Properties → Enable “Static website hosting”
- Set Index document = index.html
- Save changes
- Copy the S3 Website Endpoint URL and access your site

# Host Frontend on S3

- Create an S3 bucket

## Create bucket [Info](#)

Buckets are containers for data stored in S3.

### General configuration

#### AWS Region

Asia Pacific (Singapore) ap-southeast-1

#### Bucket type [Info](#)



##### General purpose

Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.



##### Directory

Recommended for specialized low-latency use cases supported by AWS Availability Zones or data residency use cases supported by AWS Local Zones.

#### Bucket name [Info](#)

employeeedata-529088293565

Bucket names must be 3 to 63 characters and unique within the global namespace. Bucket names must also begin and end with a letter or number. Valid characters are a-z, 0-9, periods (.), and hyphens (-). [Learn more](#) [↗](#)

#### Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: s3://bucket/prefix

## Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

### Object Ownership



##### ACLs disabled (recommended)

All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.



##### ACLs enabled

Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

#### Object Ownership

Bucket owner enforced

## Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#) [↗](#)



##### Block all public access

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.



##### Block public access to buckets and objects granted through new access control lists (ACLs)

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

# Host Frontend on S3

- Upload the index.html and frontend JavaScript files to the S3 bucket

**Upload: status**Close

After you navigate away from this page, the following information is no longer available.

**Summary**

**Destination**  
s3://employeedata-529088293565

**Succeeded**  
✔ 2 files, 5.3 KB (100.00%)

**Failed**  
⋯ 0 files, 0 B (0%)

Files and folders

Configuration

**Files and folders** (2 total, 5.3 KB)

< 1 >

Name	Folder	Type	Size	Status	Error
<a href="#">scripts.js</a>	-	text/javascript	1.9 KB	✔ Succeeded	-
<a href="#">index (1).html</a>	-	text/html	3.3 KB	✔ Succeeded	-

# Host Frontend on S3

- Enable Static website hosting

### Edit static website hosting [Info](#)

#### Static website hosting

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

**Static website hosting**

☐ Disable

☒ Enable

**Hosting type**

☒ Host a static website  
Use the bucket endpoint as the web address. [Learn more](#)

☐ Redirect requests for an object  
Redirect requests to another bucket or domain. [Learn more](#)

For your customers to access content at the website endpoint, you must make all your content publicly readable. To do so, you can edit the S3 Block Public Access settings for the bucket. For more information, see [Using Amazon S3 Block Public Access](#)

**Index document**  
Specify the home or default page of the website.

index (1).html

**Error document - optional**  
This is returned when an error occurs.

error.html

**Redirection rules - optional**  
Redirection rules, written in JSON, automatically redirect webpage requests for specific content. [Learn more](#)

1		

# Host Frontend on S3

- Edit the bucket policy under Permissions

Successfully edited bucket policy.

Block public access (bucket settings)

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to all your S3 buckets and objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to your buckets or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

Block *all* public access  
Off

► Individual Block Public Access settings for this bucket

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](#)

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "Statement1",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::employee-data-529088293565/*"
    }
  ]
}
```

Copy



# Set Up CloudFront CDN

- Go to CloudFront → Create Distribution
- Choose Distribution Type: Single Page Web Application (or Web)
- Select your S3 bucket as the Origin
- Set Origin Access = OAC to secure S3
- Click Create OAC → Copy policy → Paste into S3 bucket policy
- Set Default Root Object = index.html
- Leave other settings as default and Create Distribution
- Wait for deployment, then copy the CloudFront Domain Name

# Set Up CloudFront CDN

- Create a CloudFront distribution.

[CloudFront](#) > [Distributions](#) > Create distribution

Step 1

Choose a plan

Step 2

**Get started**

Step 3

Specify origin

Step 4

Enable security

Step 5

Review and create

## Get started

Connect your websites, apps, files, video streams, and other content to CloudFront. We optimize the performance, reliability, and security for your web traffic.

### Distribution options [Info](#)

**Distribution name**  
Name will be stored as a tag on the resource. You can change the name, or more tags, later.

**Description - optional**

**Distribution type**

☒ **Single website or app**  
Choose if each website or application will have a unique configuration.

☐ **Multi-tenant architecture - New**  
Choose when you have multiple domains that need to share configurations. This is a common architecture for SaaS providers.

### Domain [Info](#)

**Route 53 managed domain - optional**  
Enter a domain that's already registered with Route 53 in your AWS account. CloudFront will provision a TLS certificate for you. If you have a domain from a different DNS provider, skip this step and configure your domain later.

► **Tags - optional**

# Set Up CloudFront CDN

- Selected S3 bucket as the *origin* for CloudFront

### Specify origin

#### Origin type

Your origin is where your content (such as a website or app) lives. CloudFront works with AWS-based origins and origins hosted on other cloud providers.

Origin type

☒ Amazon S3  
Deliver static assets like files and images, statically generated websites or single page applications (SPA).

☐ Elastic Load Balancer  
Deliver applications hosted behind ELB such as dynamic websites, web services, and APIs.

☐ API Gateway  
Deliver API endpoints for REST APIs hosted on API Gateway.

☐ Elemental MediaPackage  
Deliver end-to-end live events or video on demand (VOD).


☐ VPC origin  
Deliver applications and content hosted within private VPCs, such as EC2 instances and Application Load Balancers.

☐ Other  
Refer to any AWS or non-AWS origin through its publicly resolvable URL.

#### Origin

**S3 origin**  
Choose an AWS origin, or enter your origin's domain name. [Learn more](#)

[Browse S3](#)

 This S3 bucket has static web hosting enabled. If you plan to use this distribution as a website, we recommend using the S3 website endpoint rather than the bucket endpoint.

[Use website endpoint](#)

**Origin path - optional**  
The directory path within your origin where your content is stored. [Learn more](#)

# Set Up CloudFront CDN

- Create a new CloudFront Origin Access Control (OAC) to allow CloudFront to securely access my S3 bucket.

⚠ This S3 bucket has static web hosting enabled. If you plan to use this distribution as a website, we recommend using the S3 website endpoint rather than the bucket endpoint. [Use website endpoint](#)

**Origin path - optional**  
Enter a URL path to append to the origin domain name for origin requests.

**Name**  
Enter a name for this origin.

**Origin access** | [Info](#)

☐ **Public**  
Bucket must allow public access.

☒ **Origin access control settings (recommended)**  
Bucket can restrict access to only CloudFront.

☐ **Legacy access identities**  
Use a CloudFront origin access identity (OAI) to access the S3 bucket.

**Origin access control**  
Select an existing origin access control (recommended) or create a new control.

[Create new OAC](#)

📄 You must allow access to CloudFront using this policy statement. Learn more about [giving CloudFront permission to access the S3 bucket](#). [Copy policy](#)

📄 [Go to S3 bucket permissions](#)

**Add custom header - optional**  
CloudFront includes this header in all requests that it sends to your origin.

[Add header](#)

# Set Up CloudFront CDN

- Copy the generated OAC bucket policy and paste it into the S3 bucket policy to allow CloudFront access

### 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](#)

```
{
  "Version": "2008-10-17",
  "Id": "PolicyForCloudFrontPrivateContent",
  "Statement": [
    {
      "Sid": "AllowCloudFrontServicePrincipal",
      "Effect": "Allow",
      "Principal": {
        "Service": "cloudfront.amazonaws.com"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::employee-data-529088293565/*",
      "Condition": {
        "StringEquals": {
          "AWS:SourceArn": "arn:aws:cloudfront::529088293565:distribution/E2T3VRRQ39QHMS"
        }
      }
    }
  ]
}
```

[Edit](#) [Delete](#) [Copy](#)

# Set Up CloudFront CDN

- Set Default Root Object = index.html

**AWS WAF web ACL - *optional***  
Choose the web ACL in AWS WAF to associate with this distribution.

CreatedByCloudFront-faad3351 ▼

**Alternate domain name (CNAME) - *optional***  
Add the custom domain names that you use in URLs for the files served by this distribution.

[Add item](#)

ⓘ To add a list of items, use the [bulk editor](#).

**Custom SSL certificate - *optional***  
Associate a certificate from AWS Certificate Manager. The certificate must be in the US East (N. Virginia) Region (us-east-1).

Choose certificate ▼

[Request certificate](#) ↗

**Supported HTTP versions**  
Add support for additional HTTP versions. HTTP/1.0 and HTTP/1.1 are supported by default

☒ HTTP/2

☐ HTTP/3

**Default root object - *optional***  
The object (file name) to return when a viewer requests the root URL (/) instead of a specific object.

index.html

**Description - *optional***

EmployeeData

# Set Up CloudFront CDN

- CloudFront is deployed. Copy the domain name and open it in your browser.

**EmployeeData** Free plan View metrics

**Details**

**Distribution domain name**  
d3pcth9n5u5yo.cloudfront.net

**Billing**  
Free plan (\$0/month)  
Manage plan

**ARN**  
arn:aws:cloudfront::529088293565:distribution/  
E2T3VRRQ39QHMS

**Last modified**  
November 21, 2025 at 1:33:38 PM UTC

General

Security

Origins

Behaviors

Error pages

Invalidations

Logging

Tags

**Settings** Edit

**Name**  
EmployeeData

**Description**  
EmployeeData

**Price class**  
Use all edge locations (best performance)

**Supported HTTP versions**  
HTTP/2, HTTP/1.1, HTTP/1.0

**Alternate domain names**  
-  
Add domain

**Standard logging**  
Available with the Pro plan

**Default root object**  
index.html

# Set Up CloudFront CDN

- CloudFront is deployed. Copy the domain name and open it in your browser.

*API stores employee data in DynamoDB and fetches all records on “View All Employees.”*

## Save and View Employee Data

Employee ID:

Name:

Department:

Age:

Save Employee Data

View all Employees

Employee ID	Name	Department	Age
-------------	------	------------	-----



# Troubleshooting & Common Issues

## ❑ 403 Forbidden

- OAC not configured correctly
- S3 bucket policy misconfigured
- Lambda IAM role missing required permissions
- API CORS Misconfiguration

## ❑ 404 Not Found

- Default Root Object was not configured in CDN
- API called with wrong resource path → Endpoint not found → 404 error.