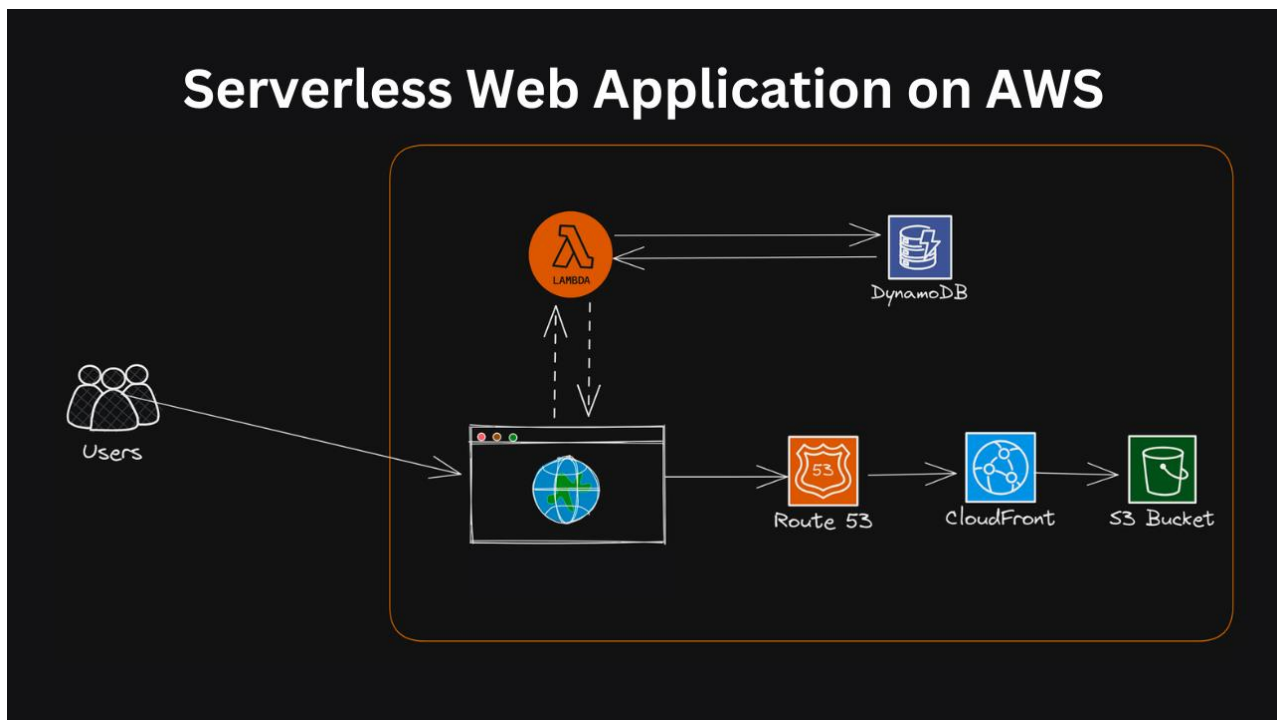# AWS PROJECT

SERVERLESS WEB APPLICATION USING ON AWS

Project Name: Serverless Web Application on AWS

Project Description:

In this project, I will build a serverless web application using AWS Lambda, DynamoDB, S3, CloudFront and Route 53. The application will allow users to create, read, update, and delete (CRUD) items from a DynamoDB table.

Project Architecture:

Steps to build the project:

1) Create a S3 bucket and to store the static web host application files (HTML, CSS, JAVA SCRIPTS)
2) Create a CloudFront distribution to serve the S3-hosted static files with low latency.
3) Create a Hosted zone using route 53 and attach the name server with domain.
4) Create a DynamoDB table to store the items
5) Create an AWS Lambda function to handle the CRUD operation on the DynamoDB table.
6) Test the project.

Detailed explain the steps:

Step 1: Create a S3 bucket and to store the static web host application files (HTML, CSS, JAVA SCRIPTS)

Go to S3 bucket → create bucket → to enter general configuration with give permission all public access→ create bucket → select the object in bucket → upload → add files → give grants all public access → upload the files.
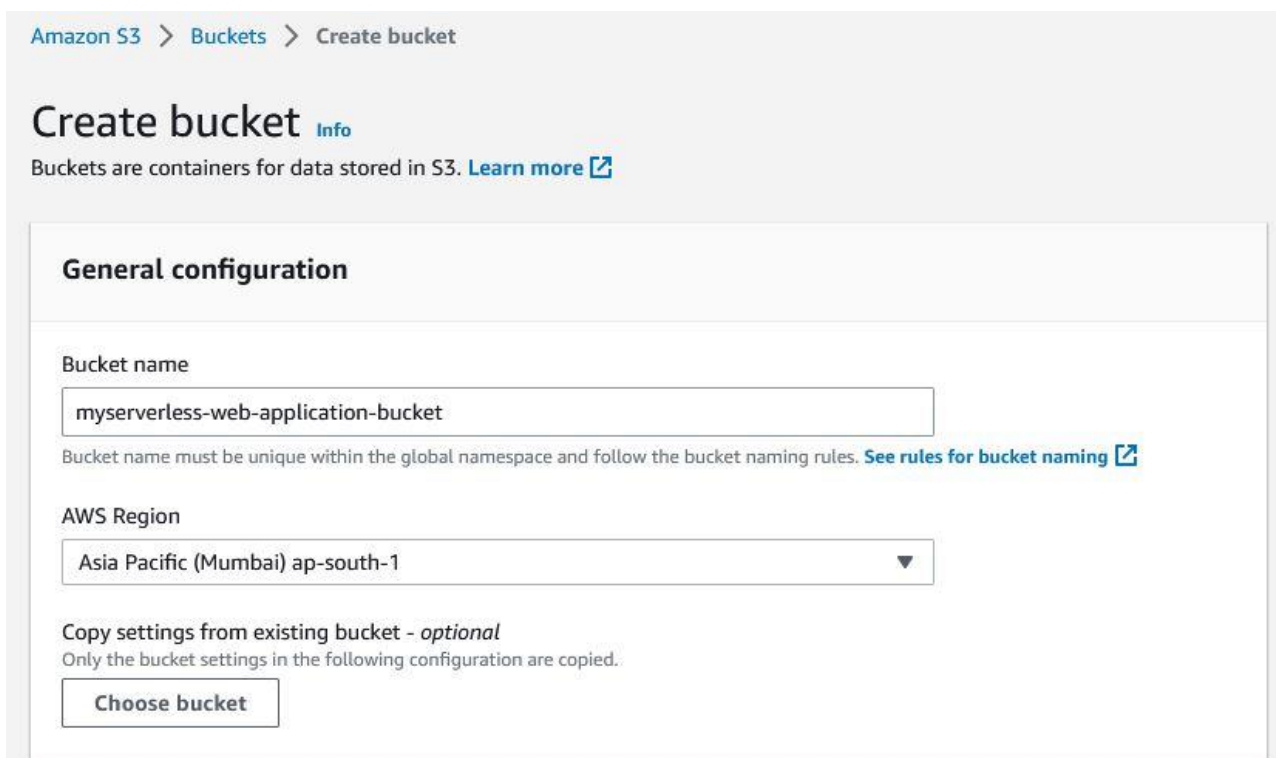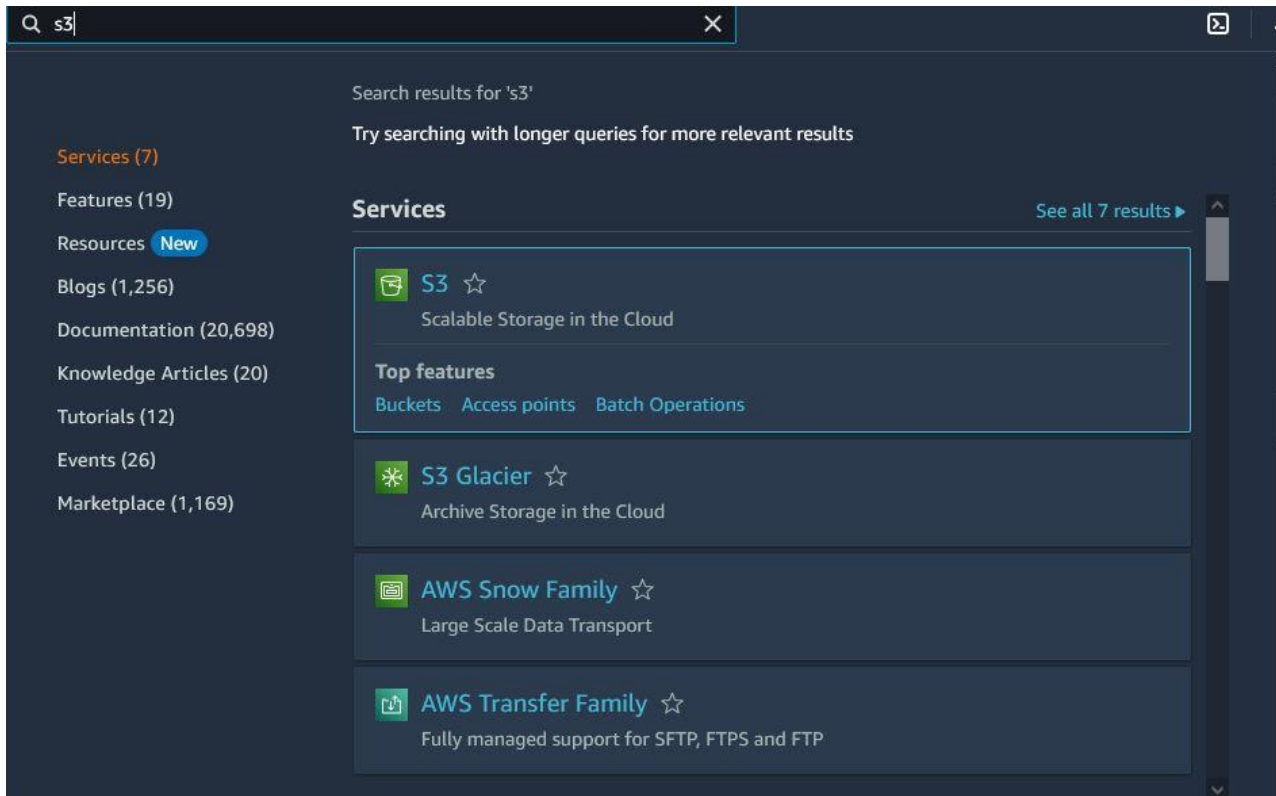
Object Ownership Info

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.

○ **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.

⚠ We recommend disabling ACLs, unless you need to control access for each object individually or to have the object writer own the data they upload. Using a bucket policy instead of ACLs to share data with users outside of your account simplifies permissions management and auditing.

Object Ownership

◉ **Bucket owner preferred**
If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.

○ **Object writer**
The object writer remains the object owner.

ⓘ If you want to enforce object ownership for new objects only, your bucket policy must specify that the bucket-owner-full-control canned ACL is required for object uploads. **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.

☐ **Block public access to buckets and objects granted through *any* access control lists (ACLs)**
S3 will ignore all ACLs that grant public access to buckets and objects.

☐ **Block public access to buckets and objects granted through *new* public bucket or access point policies**
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

☐ **Block public and cross-account access to buckets and objects through *any* public bucket or access point policies**
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

⚠ **Turning off block all public access might result in this bucket and the objects within becoming public**
AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☑ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. **Learn more** [↗]

Bucket Versioning

○ **Disable**

○ Enable

Tags (0) - *optional*

You can use bucket tags to track storage costs and organize buckets. **Learn more** [↗]

No tags associated with this bucket.

[Add tag]

Default encryption   Info

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type   Info

○ Server-side encryption with Amazon S3 managed keys (SSE-S3)

○ Server-side encryption with AWS Key Management Service keys (SSE-KMS)

○ Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
   Secure your objects with two separate layers of encryption. For details on pricing, see **DSSE-KMS pricing** on the **Management & insights** tab of the **Amazon S3 pricing page.** [↗]

Bucket Key

Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. **Learn more** [↗]

○ Disable

○ **Enable**

▼ Advanced settings

Object Lock

Store objects using a write-once-read-many (WORM) model to help you prevent objects from being deleted or overwritten for a fixed amount of time or indefinitely. **Learn more** [↗]

○ **Disable**

○ Enable
   Permanently allows objects in this bucket to be locked. Additional Object Lock configuration is

Amazon S3 > Buckets

▶ **Account snapshot**   [View Storage Lens dashboard]
  Storage lens provides visibility into storage usage and activity trends. **Learn more** ☑

Buckets (1) Info   [⟳]   [⎘ Copy content] [Empty] [Delete] [**Create bucket**]
Buckets are containers for data stored in S3. **Learn more** ☑

Q *Find buckets by name*   ⟨ 1 ⟩   ⚙

	Name	AWS Region ▽	Access ▽	Creation date ▽
○	myserverless-web-application-bucket	Asia Pacific (Mumbai) ap-south-1	Objects can be public	June 25, 2023, 08:09:10 (UTC+05:30)

Amazon S3 > Buckets > myserverless-web-application-bucket

myserverless-web-application-bucket Info

Objects   Properties   Permissions   Metrics   Management   Access Points

Objects (0)
Objects are the fundamental entities stored in Amazon S3. You can use **Amazon S3 inventory** ☑ to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. **Learn more** ☑

[⟳] [⎘ Copy S3 URI] [⎘ Copy URL] [⤓ Download] [Open ☑] [Delete] [Actions ▼]

[Create folder] [⤒ **Upload**]

Q *Find objects by prefix*   ⟨ 1 ⟩   ⚙

	Name ▲	Type ▽	Last modified ▽	Size ▽	Storage class ▽
☐					

No objects
You don't have any objects in this bucket.

[⤒ Upload]

Upload Info

Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more ↗

Drag and drop files and folders you want to upload here, or choose **Add files** or **Add folder**.

Files and folders (3 Total, 2.3 KB)

All files and folders in this table will be uploaded.

Remove | **Add files** | **Add folder**

Q Find by name   < 1 >

	Name ▲	Folder ▽	Type ▽	Size ▽
☐	index.html	-	text/html	860.0 B
☐	script.js	-	application/x-javascript	1.3 KB
☐	style.css	-	text/css	228.0 B

▼ Permissions

Grant public access and access to other AWS accounts.

Access control list (ACL)

Grant basic read/write permissions to other AWS accounts. Learn more ↗

ⓘ AWS recommends using S3 bucket policies or IAM policies for access control. Learn more ↗

Access control list (ACL)
- ⦿ Choose from predefined ACLs
- ○ Specify individual ACL permissions

Predefined ACLs
- ○ Private (recommended)
  Only the object owner will have read and write access.
- ⦿ Grant public-read access
  Anyone in the world will be able to access the specified objects. The object owner will have read and write access. Learn more ↗

⚠ **Granting public-read access is not recommended**
Anyone in the world will be able to access the specified objects. Learn more ↗

☑ I understand the risk of granting public-read access to the specified objects.

Step 2: Create a CloudFront distribution to serve the S3-hosted static files with low latency.

Go to CloudFront → distribution → create distribution → to enter the specific details→ create the distribution

Create distribution

Origin

Origin domain
Choose an AWS origin, or enter your origin's domain name.

🔍 myserverless-web-application-bucket.s3.ap-south-1.amazonaws.com ✕

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

Enter the origin path

Name
Enter a name for this origin.

myserverless-web-application-bucket.s3.ap-south-1.amazonaws.com

Origin access   Info
🔘 Public
Bucket must allow public access.

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.

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

Add header

Enable Origin Shield   Info
Origin Shield is an additional caching layer that can help reduce the load on your origin and help protect its availability.

🔘 No
⚪ Yes

Default cache behavior

Path pattern   Info

Default (*)

Compress objects automatically   Info

○ No

◉ Yes

Viewer

Viewer protocol policy

◉ HTTP and HTTPS

○ Redirect HTTP to HTTPS

○ HTTPS only

Allowed HTTP methods

◉ GET, HEAD

○ GET, HEAD, OPTIONS

○ GET, HEAD, OPTIONS, PUT, POST, PATCH, DELETE

Restrict viewer access
If you restrict viewer access, viewers must use CloudFront signed URLs or signed cookies to access your content.

◉ No

○ Yes

Cache key and origin requests

We recommend using a cache policy and origin request policy to control the cache key and origin requests.

◉ Cache policy and origin request policy (recommended)

○ Legacy cache settings

Cache policy
Choose an existing cache policy or create a new one.

| CachingOptimized | Recommended for S3 ▾ |
| Policy with caching enabled. Supports Gzip and Brotli compression. | |

Create cache policy 🗗   View policy 🗗

Origin request policy - *optional*
Choose an existing origin request policy or create a new one.

Select origin policy ▾

Create origin request policy 🗗

Step 3: Create a Hosted zone using route 53 and attach the name server with domain.

Go to Route 53 → select hosted zone → give domain name → created hosted zone and name server, then the name server attached to the domain.

Hosted zone configuration

A hosted zone is a container that holds information about how you want to route traffic for a domain, such as example.com, and its subdomains.

Domain name | Info

This is the name of the domain that you want to route traffic for.

gokultech.online

Valid characters: a-z, 0-9, ! " # $ % & ' ( ) * + , - / : ; < = > ? @ [\ ] ^ _ ` { | } . ~

Description - *optional* | Info

This value lets you distinguish hosted zones that have the same name.

The hosted zone is used for...

The description can have up to 256 characters. 0/256

Type | Info

The type indicates whether you want to route traffic on the internet or in an Amazon VPC.

○ **Public hosted zone**
A public hosted zone determines how traffic is routed on the internet.

○ **Private hosted zone**
A private hosted zone determines how traffic is routed within an Amazon VPC.

⊘ **gokultech.online was successfully created.** ✕
Now you can create records in the hosted zone to specify how you want Route 53 to route traffic for your domain.

Route 53 > Hosted zones > gokultech.online

Public **gokultech.online** Info

| Delete zone | Test record | Configure query logging |

▶ **Hosted zone details**

| Edit hosted zone |

Records (2) | DNSSEC signing | Hosted zone tags (0)

⊘ **gokultech.online was successfully created.** ✕
Now you can create records in the hosted zone to specify how you want Route 53 to route traffic for your domain.

Records (2) | DNSSEC signing | Hosted zone tags (0)

Records (2) Info

Automatic mode is the current search behavior optimized for best filter results. To change modes go to settings.

| ⟳ | Delete record | Import zone file | **Create record** |

🔍 Filter records by property or value | Type ▼ | Routing pol... ▼ | Alias ▼ | ‹ 1 › | ⚙

☐	Record ... ▽	Type ▽	Routin... ▽	Differ... ▽	Alias ▽	Value/Route traffic to ▽	TTL (s.
☐	gokultech...	NS	Simple	-	No	ns-528.awsdns-02.net. ns-1310.awsdns-35.org. ns-1735.awsdns-24.co.uk. ns-452.awsdns-56.com.	172800
☐	gokultech...	SOA	Simple	-	No	ns-528.awsdns-02.net. awsd...	900

Step 4: Once created hosted zone then go to CloudFront → edit the distribution → to enter the alternate domain name → click request certificate → redirect to AWS certified manager → certificate → request certificate → to fill the details → create certificate.

Step 5: certificate is created then status is pending, click certificate id → create DNS record in amazon route 53 → create record → wait more than 10 minutes → status is issued

Step 6: Go to Cloud front → edit the distribution → enter the custom SSL certificate → save the changes. Then go to route 53 → create record → alias on → route traffic to cloudFront distribution → policy is simple routing → create record

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

registration.gokultech.online   Remove

Add item

ⓘ To add a list of alternative domain names, 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).

*.gokultech.online (0c7fd244-d356-487f-9b27-c372006c25da)   ▼   C

⊘ *.gokultech.online ↗   Request certificate ↗

Legacy clients support - $600/month prorated charge applies. Most customers do not need this.
CloudFront allocates dedicated IP addresses at each CloudFront edge location to serve your content over HTTPS.
☐ Enabled

Security policy
The security policy determines the SSL or TLS protocol and the specific ciphers that CloudFront uses for HTTPS connections with viewers (clients).
● TLSv1.2_2021 (recommended)
○ TLSv1.2_2019

CloudFront > Distributions

Distributions (1) Info   C   Enable   Disable   Delete   **Create distribution**

🔍 Search all distributions   < 1 > ⚙

☐	ID	Description ▽	Type ▽	Domain na... ▽	Alternate ... ▽	Origins ▽	Status ▽	Last modif... ▼
☐	E1T53FAPW0NFYH	-	Production	dzy8jdlzhgira...	registration.gokulte	myserverless-web-a	⊘ Enabled	June 25, 202...

Quick create record   Switch to wizard

▼ Record 1   Delete

Record name Info   Record type Info

registration   .gokultech.online   A – Routes traffic to an IPv4 address and some AWS resources ▼
Keep blank to create a record for the root domain.

🔘 Alias

Route traffic to Info

Alias to CloudFront distribution   ▼

US East (N. Virginia)   ▼

An alias to a CloudFront distribution and another record in the same hosted zone are global and available only in US East (N. Virginia).

🔍 dzy8jdlzhgira.cloudfront.net   ✕

Use: "dzy8jdlzhgira.cloudfront.net"

registration.gokultech.online (dzy8jdlzhgira.cloudfront.net)

Step 7: Once created record to check the browser with domain name

Step 8: Create a IAM role policy in DynamoDB full access

Go to IAM → select role → use case lambda → give policy permission → create role.

Select trusted entity Info

Trusted entity type

- ⦿ **AWS service**
  Allow AWS services like EC2, Lambda, or others to perform actions in this account.

- ○ **AWS account**
  Allow entities in other AWS accounts belonging to you or a 3rd party to perform actions in this account.

- ○ **Web identity**
  Allows users federated by the specified external web identity provider to assume this role to perform actions in this account.

- ○ **SAML 2.0 federation**
  Allow users federated with SAML 2.0 from a corporate directory to perform actions in this account.

- ○ **Custom trust policy**
  Create a custom trust policy to enable others to perform actions in this account.

Use case

Allow an AWS service like EC2, Lambda, or others to perform actions in this account.

Common use cases

- ○ EC2
  Allows EC2 instances to call AWS services on your behalf.

- ⦿ Lambda
  Allows Lambda functions to call AWS services on your behalf.

Use cases for other AWS services:

Choose a service to view use case ▼

Permissions policies (Selected 1/859) Info

Choose one or more policies to attach to your new role.

[🔄]   [**Create policy** ☐]

🔍 Filter policies by property or policy name and press enter.	4 matches	‹ 1 ›	⚙

"dynamodb" ✕   [**Clear filters**]

☐	Policy name ☐ ▽	Type ▽	Description
☑	⊞ 🎁 AmazonDynamoDB...	AWS m...	Provides full access to Amazon DynamoDB via the AWS Management Co...
☐	⊞ 🎁 AmazonDynamoDB...	AWS m...	Provides read only access to Amazon DynamoDB via the AWS Managem...
☐	⊞ 🎁 AWSLambdaInvocat...	AWS m...	Provides read access to DynamoDB Streams.
☐	⊞ 🎁 AWSLambdaDynam...	AWS m...	Provides list and read access to DynamoDB streams and write permissio...

Name, review, and create

Role details

Role name
Enter a meaningful name to identify this role.

```
serverless-webapplication-role
```

Maximum 64 characters. Use alphanumeric and '+=,.@-_' characters.

Description
Add a short explanation for this role.

```
Allows Lambda functions to call AWS services on your behalf.
```

Maximum 1000 characters. Use alphanumeric and '+=,.@-_' characters.

IAM > Roles > serverless-webapplication-role

serverless-webapplication-role   Delete

Allows Lambda functions to call AWS services on your behalf.

Summary   Edit

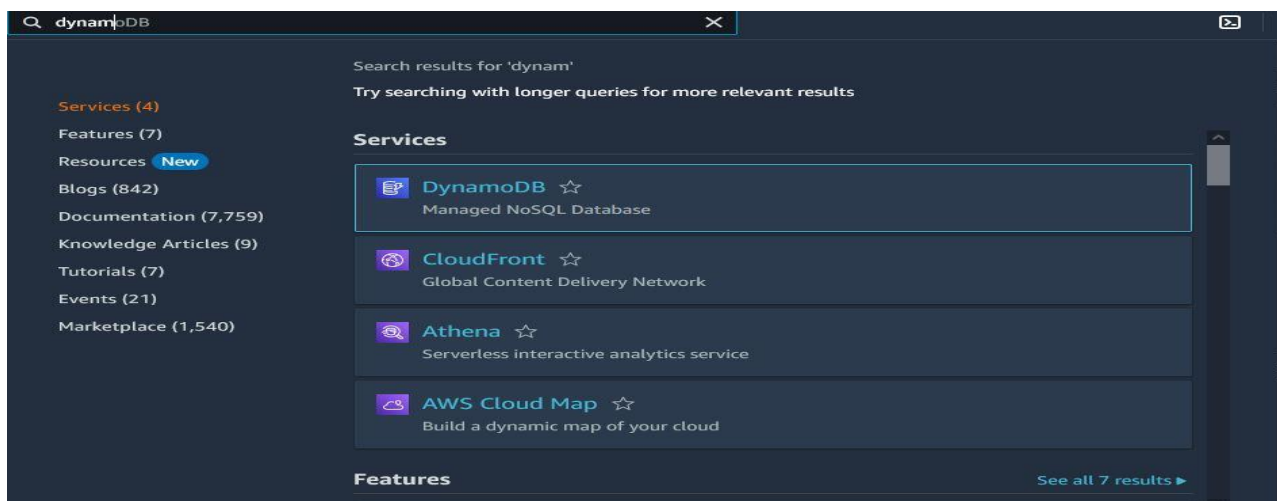Creation date   ARN
June 25, 2023, 09:33 (UTC+05:30)   📋 arn:aws:iam::207146887812:role/serverless-webapplication-role

Last activity   Maximum session duration
None   1 hour

Step 9: Create a DynamoDB table to store the items

Go to DynamoDB → create table → explore the items → create items

Dashboard
Tables
    Update settings
    Explore items
PartiQL editor
Backups
Exports to S3
Imports from S3
Reserved capacity
Settings

▼ DAX
Clusters

Database

Amazon DynamoDB
A fast and flexible NoSQL database service for any scale

DynamoDB is a fully managed, key-value, and document database that delivers single-digit-millisecond performance at any scale.

Get started

Create a new table to start exploring DynamoDB.

Create table

DynamoDB > Tables > Create 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.

    registration-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.

    email   String ▼

1 to 255 characters and case sensitive.

Step 10: Create an AWS Lambda function to handle the CRUD operation on the DynamoDB table.

Go to Lambda → select function → to fill basic configuration → attach the role → create function → to feed the python file → deploy
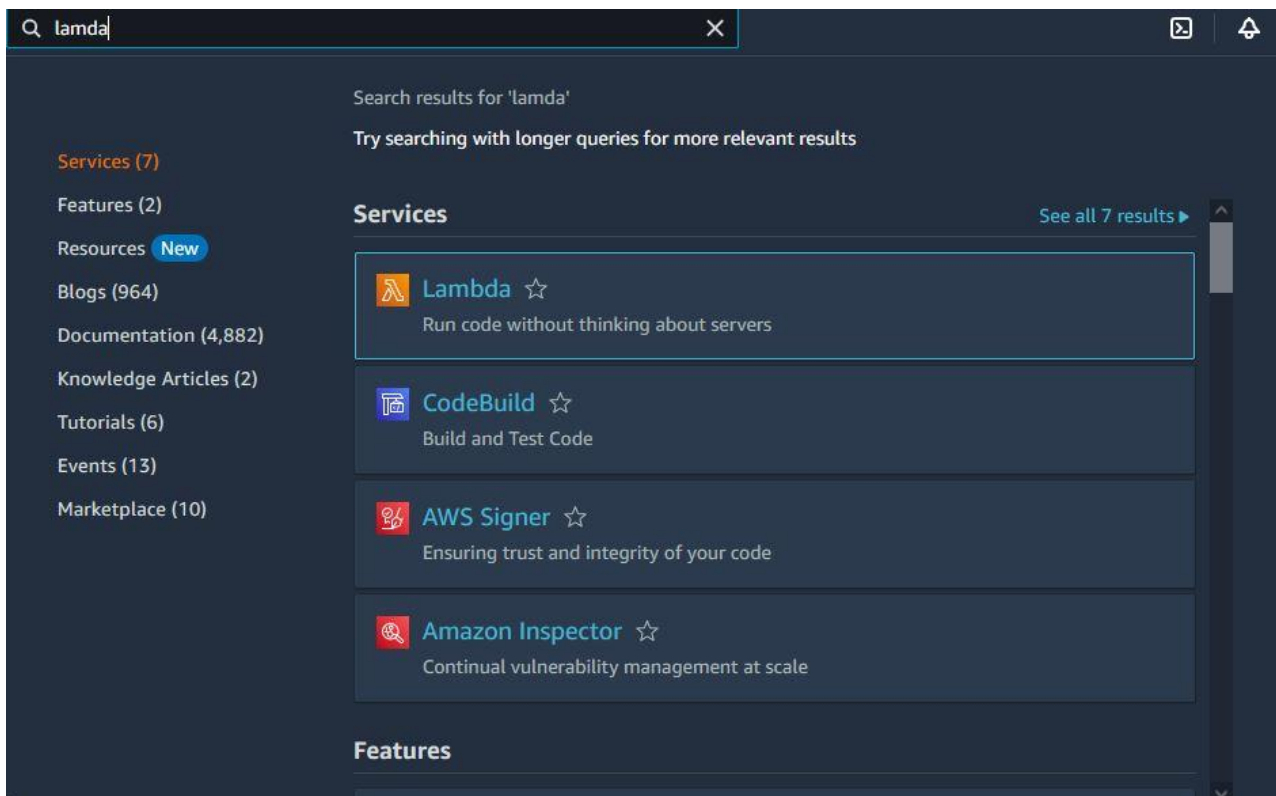
Create function Info

AWS Serverless Application Repository applications have moved to Create application.

● Author from scratch	○ Use a blueprint	○ Container image
Start with a simple Hello World example.	Build a Lambda application from sample code and configuration presets for common use cases.	Select a container image to depl function.

Basic information

Function name
Enter a name that describes the purpose of your function.

myregistration-form-function

Use only letters, numbers, hyphens, or underscores with no spaces.

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.

● x86_64

○ arm64

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
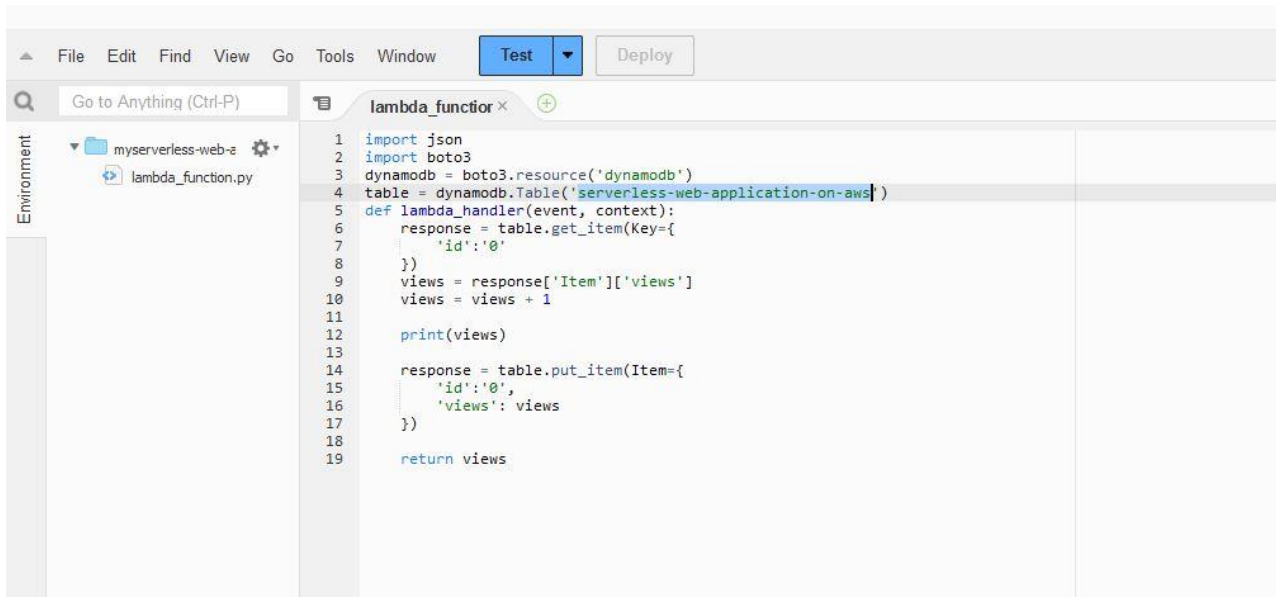
○ Create a new role from AWS policy templates

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.

serverless-webapplication-role ▼ | ⟳

View the serverless-webapplication-role role ↗ on the IAM console.

```
1   import json
2   import boto3
3   dynamodb = boto3.resource('dynamodb')
4   table = dynamodb.Table('serverless-web-application-on-aws')
5   def lambda_handler(event, context):
6       response = table.get_item(Key={
7           'id':'0'
8       })
9       views = response['Item']['views']
10      views = views + 1
11
12      print(views)
13
14      response = table.put_item(Item={
15          'id':'0',
16          'views': views
17      })
18
19      return views
```

Step 11: Test the application

To check the browser with domain name

Then refresh views count will be increased

Data will be stored in DynamoDB table