Repo Registry (AWS S3/CloudFront,API, Lambda, DDB)

A tiny, production-ready serverless app where anyone can **submit a GitHub repo** and **browse all submissions**.  
Frontend is a static site on **S3 behind CloudFront (OAC, HTTPS)**; the API is **API Gateway → Lambda (Python) → DynamoDB**.

**Stack**

* **CloudFront + S3**: static hosting, HTTPS, private bucket with OAC
* **API Gateway (REST)**: /projects (**GET**, **POST**)
* **AWS Lambda (Python)**: request handling, validation, JSON responses
* **DynamoDB**: submissions store
  + Partition key: Project (constant "PROJECT")
  + Sort key: {createdAtEpochSeconds}#{uuid} (newest-first queries)

**Environment**

Set on the Lambda function:

* TABLE=Projects
* PK\_ATTR=Project
* SK\_ATTR={createdAtEpochSeconds}#{uuid}
* ALLOWED\_ORIGIN=https://<your-cloudfront-domain>

## Create the DynamoDB table

1. DynamoDB → **Create table**
2. **Table name**: Projects
3. **Partition key**: Project (String)
4. **Sort key**: {createdAtEpochSeconds}#{uuid} (String)
5. Capacity mode: **On-demand**
6. Create

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Create the Lambda (Python)

1. Lambda → **Create function** → Author from scratch
   * Name: projecthandler
   * Runtime: **Python 3.11/3.12**
2. Paste the contents of lambda/lambda\_function.py.
3. **Environment variables** (Configuration → Environment variables):
   * TABLE = Projects
   * PK\_ATTR = Project
   * SK\_ATTR = {createdAtEpochSeconds}#{uuid}
   * ALLOWED\_ORIGIN = https://<your-cloudfront-domain>
4. **Permissions** (Execution role) → attach a policy allowing:

{

"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action": ["dynamodb:PutItem", "dynamodb:Query"],

"Resource": "arn:aws:dynamodb:<REGION>:<ACCOUNT\_ID>:table/Projects"

}

]

}

1. Deploy the function.

A screenshot of a computer

AI-generated content may be incorrect.A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

## Create the REST API (API Gateway)

1. API Gateway → **Create API** → **REST API**
2. **Resources**: create resource /projects
3. Add methods on /projects (all with **Lambda proxy**):
   * **OPTIONS** → Lambda (or MOCK that returns CORS headers)
   * **GET** → Lambda projecthandler
   * **POST** → Lambda projecthandler
4. **Enable CORS** for /projects (or ensure responses include CORS headers)
5. **Deploy API** to a stage, e.g., prod
6. Note the **Invoke URL**:  
   https://<restid>.execute-api.<region>.amazonaws.com/prod

A screenshot of a computer

AI-generated content may be incorrect.

## Host the frontend (S3 + CloudFront, OAC)

1. S3 → **Create bucket** (private), e.g., your-site-bucket
2. Upload index.html, app.js, styles.css
3. CloudFront → **Create distribution**
   * **Origin**: your S3 bucket (not the website endpoint)
   * **Origin access**: **Origin Access Control (OAC)**
   * **Default root object**: index.html
   * Save → copy/paste the **generated bucket policy** into your bucket’s **Bucket policy**
4. (Optional) Custom domain → attach ACM cert in **us-east-1**
5. Note your **CloudFront domain**: https://<xxxx>.cloudfront.net
6. A screenshot of a computer

   AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer screen

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.