Rating

This Lambda function stores and retrieves page feedback using DynamoDB. It is already deployed in the MUI AWS account. Request credentials if you need to update dev for testing, or to deploy a new prod version.

If you wish to deploy your own instance for testing, follow the steps below.

Prerequisites

Create an AWS profile in ~/.aws/credentials called "claudia" with credentials corresponding to an IAM user with AmazonAPIGatewayAdministrator, AWSLambdaFullAccess and IAMFullAccess policies. You can do that with `aws configure --profile claudia`.

Create a table in DynamoDB, with a `string` partition key called `id`, and a sort key called `page`. You can do that from the DynamoDB web console, or using the AWS CLI command line. Here is an example command that will create the `feedback-dev` table with the minimal provisioned throughput:

```
aws dynamodb create-table --profile claudia --region us-east-1 \
  --attribute-definitions AttributeName=id,AttributeType=S
AttributeName=page,AttributeType=S \
  --key-schema AttributeName=id,KeyType=HASH AttributeName=page,KeyType=RANGE \
  --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=1 \
  --query TableDescription.TableArn --output text \
  --table-name feedback-dev
```

You will need to repeat this command to create a table for production, for example `feedback-prod`.

For on-demand throughput, replace:

```
  --provisioned-throughput ReadCapacityUnits=5,WriteCapacityUnits=1 \
```

with:

```
  --billing-mode PAY_PER_REQUEST \
```

The project includes an IAM access policy that will grant the lambda function access to the tables. You can edit the policies/access-dynamodb.json file to change the access permissions. These are only applied on create (`yarn setup`). Alternatively, to avoid inadvetantly pushing changes, use the `--policies` flag with `yarn setup` to refer to a copy of this directory, and exclude it in your `~/.gitignore`.

> ⚠️ *You will need to update the "Resource" key in this file with the value returned after creating each table.*

Get started

> ⚠️ *When setting up for the first time, you will need to delete the included `claudia.json` file that is specific to the MUI installation. Alternatively, if making changes to the function that you intend to submit back, then to avoid inadvetantly commiting changes to `claudia.json`, use `--config` with each command to create and use a local config file, and exclude this file in your `~/.gitignore`.*

To set this up, first [set up the credentials](), then:

1. run `yarn install` (from the root workspace) to install the dependencies
2. Navigate into the directory of this README, e.g. `cd docs/packages/feedback`
3. run `yarn setup` to create the lambda function on AWS under the default name. This will also ask you for table names for development and production. If you used the above AWS command, they will be `feedback-dev` and `feedback-dev` respectively.
4. Test the API using the example requests below

For subsequent updates, use the `npm run deploy` command.

Stage variables

The table name, stored in the API Gateway stage variables, is passed to each request processor in the `request.env` key-value map. Check out index.js to see it in use.

The value is set during the first deployment, using `--configure-table-dev` & `--configure-table-prod`. This works using a post-deploy step (check out the last line of index.js for the actual setup, and Configuring stage variables using post-deployment steps for more information about the API).

The API

- `POST` to `/feedback` - stores a new rating data object
- `GET` from `/feedback/{id}` - returns all ratings with id `{id}`
- `GET` from `/rating/average` - returns average ratings for all pages

Testing

Claudia will print the API URL after it is created (typically something in the format `https://[API ID].execute-api.[REGION].amazonaws.com/<version>`). Replace `<API-URL>` with that value in the examples below:

You can test the API by using `curl` (or using a fancier client like Postman). Below are some examples with `curl`.

Create new feedback

This will create a feedback entry from the data stored in example.json. Change the data in the file to create ratings:

```
curl -H "Content-Type: application/json" -X POST --data @example.json <API-URL>/feedback
```

Add the UUID returned to `example.json` with key `id` to store more feedback under the same id.

Retrieve feedback

This will get the feedback stored for ID d6890562-3606-4c14-a765-da81919057d1

```
curl <API-URL>/feedback/d6890562-3606-4c14-a765-da81919057d1
```

Retrieve average ratings

This will get the average feedback stored for all pages

```
curl <API-URL>/feedback/average
```

Testing with the documentation

Create the file `docs/.env.local` containing an environment variable `FEEDBACK_URL` with your API URL without the version. For example:

```
FEEDBACK_URL=https://abcd123ef4.execute-api.us-east-1.amazonaws.com
```

If already running, restart the local docs site. Feedback should now be posted to your deployment.