



## Single Calculator Microservice using CLI

1. In our previous lab, we built a single Calculator Microservice using AWS Console and AWS Lambda.
2. In this lab, we will again use AWS Lambda as a service to build our Single Calculator Microservice, but this time through AWS CLI.
3. So, first create a folder and open it in VS code then create these shown in the snapshot.
4. Create a text file to store all the commands and then create an index.js file where all the code will be created for our lambda function then we need to create a zip file out of our index.js file.

The screenshot shows a VS Code interface with the following details:

- Folders: Using CLI**: A sidebar showing three files: commands.txt, event.json, and index.js.
- Welcome**: The main editor tab bar.
- commands.txt**: An empty file.
- index.js**: The active file, showing the following code:

```
1  export const handler = async (event) => {
2      console.log('Received event:', JSON.stringify(event, null, 2));
3
4      // Check if event.body exists and is not empty before parsing
5      if (!event.body) {
6          return {
7              statusCode: 400,
8              body: JSON.stringify({
9                  error: "Request body is missing or empty"
10             })
11         };
12     }
13
14     let payload;
15     try {
16         // Try parsing the body
17         payload = JSON.parse(event.body);
18     } catch (error) {
19         // If parsing fails, return an error response
20         return {
21             statusCode: 400,
22             body: JSON.stringify({
23                 error: "Invalid JSON format in request body",
24                 details: error.message
25             })
26         };
27     }
}
```

5. Now just run the command to zip our file, below you can see that our file has been zipped.

```
PS D:\AWS Serveless\Single Calculator Microservice Expose https methods\Using CLI> Compress-Archive index.js functi
● on.zip
○ PS D:\AWS Serveless\Single Calculator Microservice Expose https methods\Using CLI>
```

6. Then we ran this command to get the execution role for our lambda function.

```

PS D:\AWS Serveless\Single Calculator Microservice Expose https methods\Using CLI> aws iam get-role --role-name lambda-exec
>>
Role:
  Arn: arn:aws:iam::878893308172:role/lambda-exec
  AssumeRolePolicyDocument:
    Statement:
      - Action: sts:AssumeRole
        Effect: Allow
        Principal:
          Service: lambda.amazonaws.com
        Version: '2012-10-17'
      CreateDate: '2024-11-12T08:14:46+00:00'
      MaxSessionDuration: 3600

```

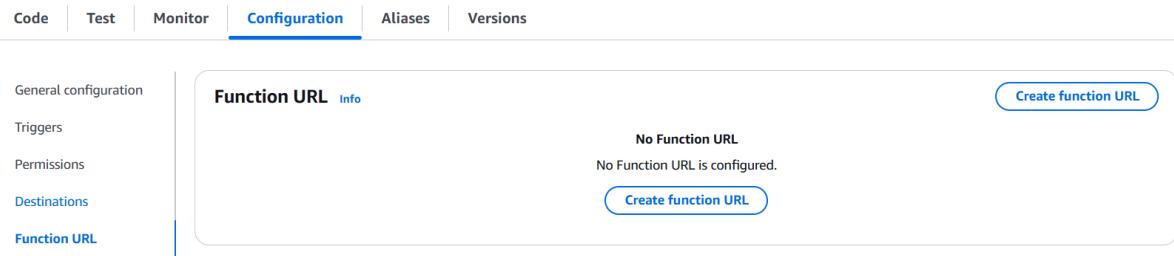
- After that we ran the create lambda command to create our function.

```

>>   --role Arn: arn:aws:iam::878893308172:role/lambda-exec
PS D:\AWS Serveless\Single Calculator Microservice Expose https methods\Using CLI> aws lambda create-function \
>>   --function-name calculator \
>>   --runtime nodejs18.x \
>>   --zip-file fileb://function.zip \
>>   --handler index.handler \
>>   --role arn:aws:iam::878893308172:role/lambda-exec
Architectures:
- x86_64
CodeSha256: AETC4kxMsU4+pmu4owbpEq552ai8hJI97M//Fr69dw8=
CodeSize: 735
Description: ''
EphemeralStorage:
  Size: 512

```

- On the console too we can see that our function has been created. Now we need to create the function URL for our function. For that go inside it and from the configuration tab choose function URL and click on Create.



- First you need to choose the Auth type to None and scroll down to expand additional settings. There you need to enable the CORS feature and click on save.

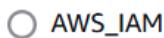
## Configure Function URL

### Function URL Info

Use function URLs to assign HTTP(S) endpoints to your Lambda function.

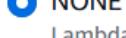
#### Auth type

Choose the auth type for your function URL. [Learn more](#)



**AWS\_IAM**

Only authenticated IAM users and roles can make requests to your function URL.



**NONE**

Lambda won't perform IAM authentication on requests to your function URL. The URL endpoint will be public.

▼ Additional settings

**Invoke mode**

Choose how your function returns responses. [Learn more](#)

**BUFFERED (default)**  
The invocation results are available when the payload is complete. Response payload max size: 6 MB

**RESPONSE\_STREAM**  
Stream the invocation results. Streaming responses incur additional costs. Refer to the documentation for payload size limitations. [Learn more](#)

**Configure cross-origin resource sharing (CORS)**  
Use CORS to allow access to your function URL from any domain. You can also use CORS to control access for specific HTTP headers and methods in requests to your function URL. [Learn more](#)

**Allow origin**

Add the origins that can access your function URL. You can list any number of specific origins. Alternatively, you can grant access to all origins with the wildcard character (\*). For example: https://www.example.com, https://\*, or \*.

[Add new value](#)

10. Now you will see that your function has the function URL.

calculator

Throttle Copy ARN Actions ▾

▼ Function overview [Info](#)

[Export to Infrastructure Composer](#) [Download](#)

[Diagram](#) [Template](#)

**calculator**

Layers (0)

+ Add trigger + Add destination

Description  
-

Last modified  
4 minutes ago

Function ARN  
[arn:aws:lambda:us-east-1:878893308172:function:calculator](#)

Function URL [Info](#)  
<https://mpucj5w3okshvmtcyb37qux2jq0fsogp.lambda-url.us-east-1.on.aws>

11. Like before open the Postman tool on your laptop. Paste the function here and choose the POST method then choose the body, raw and JSON. Write the query and click on Send button you will see that your calculator is working.

POST <https://mpucj5w3okshvmtcyb37qux2jq0fsogp.lambda-url.us-east-1.on.aws/>

No environment

Save Share

POST <https://mpucj5w3okshvmtcyb37qux2jq0fsogp.lambda-url.us-east-1.on.aws/>

Send

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify

```
1 {  
2   "a": 2,  
3   "b": 5,  
4   "op": "+"  
5 }
```

Body Cookies Headers (6) Test Results

200 OK 596 ms 331 B

Pretty Raw Preview Visualize JSON

```
1 {  
2   "processed": true,  
3   "result": 7  
4 }
```