Author: Madhurima Rawat

Using Cloud Functions for Serverless Computing

This experiment focuses on the deployment, execution, and testing of an AWS Lambda function using AWS CLI and LocalStack. The Lambda runtime image is pulled from Amazon ECR and executed locally using Docker, simulating a serverless environment. The process includes function invocation and result validation.

This document provides a comprehensive breakdown of all commands, inputs, outputs, and their explanations, ensuring a clear understanding of each step in the workflow.

1. Creating a Lambda Function

Command:

```
aws --endpoint-url=http://localhost:4566 lambda create-function \
  --function-name myLambdaFunction \
  --runtime python3.8 \
  --role arn:aws:iam::000000000000:role/execution_role \
  --handler lambda_function.lambda_handler \
  --zip-file fileb://lambda_function.zip
```

Explanation:

- `aws lambda create-function` → Creates a new AWS Lambda function.
- `--function-name myLambdaFunction` → Specifies the function name.
- `--runtime python3.8` → Uses **Python 3.8** as the execution environment.
- `--role arn:aws:iam::000000000000:role/execution_role` → Assigns an **IAM execution role**.
- `--handler lambda_function.lambda_handler` → Defines the function entry point.
- `--zip-file fileb://lambda_function.zip` → Uploads the function code as a `.zip` file.
- `--endpoint-url=http://localhost:4566` → Uses **LocalStack** instead of AWS.

Output:

```
-----------------------------------------------------
|                AWS Lambda Function Info           |
-----------------------------------------------------
| CodeSha256       | G16xwAmelqgPBfzQVMlj7SR/...   |
| CodeSize         | 312 KB                         |
| FunctionArn      | arn:aws:lambda:us-east-1:... |
```

```
| FunctionName     | myLambdaFunction              |
| Handler          | lambda_function.lambda_handler |
| Runtime          | python3.8                     |
| State           | Pending                       |
| StateReason      | The function is being created.|
| StateReasonCode | Creating                      |
| Version          | $LATEST                       |
-------------------------------------------------
```

Output Breakdown:

- **`FunctionName`** → `myLambdaFunction` is created.
- **`Runtime`** → Uses `Python 3.8`.
- **`Handler`** → Entry point is `lambda_function.lambda_handler`.
- **`State`** → `Pending` (Lambda is being created).
- **`StateReasonCode`** → `Creating` (it is not yet ready for invocation).

2. Invoking the Lambda Function

Command:

```
aws --endpoint-url=http://localhost:4566 lambda invoke \
  --function-name myLambdaFunction output.txt
```

Explanation:

- `aws lambda invoke` → Tries to **execute the function**.
- `--function-name myLambdaFunction` → Specifies the function to invoke.
- `output.txt` → Saves the function **output to a file**.

Output:

```
An error occurred (ResourceConflictException) when
calling the Invoke operation:
The operation cannot be performed at this time.
The function is currently in the following state: Pending
```

Output Breakdown:

- **`ResourceConflictException`** → The function **is not yet active**.
- **`State: Pending`** → The function is still being created and cannot be invoked yet.

3. Checking Lambda Function Details

Command:

```
aws --endpoint-url=http://localhost:4566 lambda get-function \
  --function-name myLambdaFunction
```

Explanation:

- `aws lambda get-function` → Retrieves function details.
- `--function-name myLambdaFunction` → Specifies the function to check.

Output:

```
-------------------------------------------------------------
|                   AWS Lambda GetFunction Info             |
-------------------------------------------------------------
|                      Code Details                         |
-------------------------------------------------------------
| Location       | http://s3.localhost.localstack.cloud...|
| RepositoryType | S3                                      |
-------------------------------------------------------------
|                Configuration Details                      |
-------------------------------------------------------------
| CodeSha256     | G16xwAmelqgPBfzQVMlj7SR/...            |
| CodeSize       | 312 KB                                  |
| FunctionArn    | arn:aws:lambda:us-east-1:0000000...  |
| FunctionName   | myLambdaFunction                        |
| Handler        | lambda_function.lambda_handler        |
| LastModified   | 2025-02-18T13:15:50.795443+0000       |
| MemorySize     | 128 MB                                  |
| PackageType    | Zip                                     |
| State          | Pending                                 |
| StateReason    | The function is being created.       |
| StateReasonCode| Creating                                |
| Timeout        | 3 seconds                               |
| Version        | $LATEST                                 |
-------------------------------------------------------------
```

Output Breakdown:

- **`State: Pending`** → The function is still **not ready** for execution.
- **`StateReason: The function is being created.`** → AWS (or LocalStack) is still processing the deployment.

4. Checking LocalStack Logs

Command:

```
docker logs -f localstack-main
```

Explanation:

- `docker logs -f localstack-main` → Displays **real-time logs** of LocalStack.

Output:

```
LocalStack version: 4.0.4.dev122
LocalStack build date: 2025-01-23
LocalStack build git hash: 81d9fb079

Ready.
2025-02-18T13:12:14.309  INFO --- [uest_thread)] localstack.
dns.server
: Unable to get DNS result from upstream server
192.168.65.7 for domain analytics.
localstack.cloud.: The DNS operation timed out.
2025-02-18T13:16:46.208  WARN --- [et.reactor-1] l.services.lambda_
.hints
: Lambda functions are
created asynchronously in AWS.
Before invoking myLambdaFunction,
please wait until the function is Active using:

"awslocal lambda wait function-active-v2
--function-name myLambdaFunction"

Check out https://docs.localstack.cloud/user-
guide/aws/lambda/#function-in-pending-state
```

Output Breakdown:

- **`WARN: Lambda functions are created asynchronously.`**
  → LocalStack suggests **waiting** until the function becomes **Active**.

5. Waiting for Lambda to Become Active

Command:

```
aws --endpoint-url=http://localhost:4566 lambda wait function-active-v2 \
  --function-name myLambdaFunction
```

Explanation:

- `aws lambda wait function-active-v2` → Waits for the function to transition to **Active**.
- `--function-name myLambdaFunction` → Specifies which function to monitor.

Output:

```
Waiter FunctionActiveV2 failed:
Waiter encountered a terminal failure state:
For expression "Configuration.State" we matched expected path: "Failed"
```

Output Breakdown:

- **`Waiter encountered a terminal failure state`** → The function **failed to activate**.
- **Possible Causes:**
  i. **Incorrect IAM Role permissions**.
  ii. **Code packaging issues** in `lambda_function.zip` .
  iii. **LocalStack instability** (restart and retry).

6. Checking Lambda Function Details

Command:

```
aws --endpoint-url=http://localhost:4566 lambda get-function \
  --function-name myLambdaFunction
```

Explanation:

- `aws lambda get-function` → Retrieves details about the specified Lambda function.
- `--function-name myLambdaFunction` → Specifies the function to check.

Output:

```
-------------------------------------------------------------
|                  AWS Lambda GetFunction Info              |
```

```
-------------------------------------------------------------
|                      Code Details                         |
-------------------------------------------------------------
| Location       | http://s3.localhost.localstack.cloud...|
| RepositoryType | S3                                      |
-------------------------------------------------------------
|                Configuration Details                      |
-------------------------------------------------------------
| CodeSha256     | G16xwAmelqgPBfzQVMlj7SR/...            |
| CodeSize       | 312 KB                                  |
| FunctionArn    | arn:aws:lambda:us-east-1:0000000...    |
| FunctionName   | myLambdaFunction                        |
| Handler        | lambda_function.lambda_handler          |
| LastModified   | 2025-02-18T13:15:50.795443+0000         |
| LastUpdateStatus | Failed                                |
| State          | Failed                                  |
| StateReason    | Error while creating lambda:            |
|               | public.ecr.aws/lambda/python:3.8        |
| StateReasonCode| InternalError                           |
| Timeout        | 3 seconds                               |
| Version        | $LATEST                                 |
-------------------------------------------------------------
```

Output Breakdown:

- **`State: Failed`** → The function **failed** during creation.
- **`StateReason: Error while creating lambda: public.ecr.aws/lambda/python:3.8`** → The Lambda runtime image could not be retrieved.
- **`StateReasonCode: InternalError`** → An issue within LocalStack or the runtime environment.
- **Possible Fix:**
  i. **Ensure `public.ecr.aws/lambda/python:3.8` is available** by pulling it manually.
  ii. **Recreate the function after fixing dependencies.**

7. Checking Lambda Runtime Availability

Command:

```
docker manifest inspect public.ecr.aws/lambda/python:3.8
```

Explanation:

- `docker manifest inspect` → Checks available **Docker images** for `python:3.8` Lambda runtime.

Output:

```
{
  "schemaVersion": 2,
  "mediaType":
    "application/vnd.docker.distribution.manifest.list.v2+json",
  "manifests": [
    {
      "digest":
        "sha256:7522519543baf4454d030093d6d57b5189b82025fc7e
        ae70bb6603181249d954",
      "platform": {
        "architecture": "arm64",
        "os": "linux",
        "variant": "v8"
      }
    },
    {
      "digest":
        "sha256:e4defd38fe1c2a3930c98e0f986fbd08b802
        bfe3f47c71c6f144c3677b3c64a2",
      "platform": {
        "architecture": "amd64",
        "os": "linux"
      }
    }
  ]
}
```


Output Breakdown:

- The `python:3.8` Lambda runtime is available in two versions:
  - **`arm64` (for ARM-based CPUs like Apple M1/M2)**
  - **`amd64` (for standard x86 systems)**
- If the machine runs on `amd64`, the correct image should be pulled.

8. Pulling the Lambda Runtime Image

Command:

```
docker pull public.ecr.aws/lambda/python:3.8
```

Explanation:

- `docker pull` → Manually downloads the **Lambda runtime image** from Amazon ECR.

Output:

```
3.8: Pulling from lambda/python
bc2b3a540f9b: Pull complete
7e7a8ab075f3: Pull complete
...
Digest: sha256:93e78742873d3ad0c28582366b217ce5169889
f4d63d61179598c2a3dc6142ff
Status: Downloaded newer image for public.ecr.aws/lambda/python:3.8
```

Output Breakdown:

- Confirms that the image has been **successfully downloaded**.
- Digest `sha256:93e787...` verifies the integrity of the image.

9. Listing Installed Docker Images

Command:

```
docker images
```

Explanation:

- `docker images` → Lists all **available images** on the machine.

Output:

```
REPOSITORY                    TAG       IMAGE ID       CREATED       SIZE
flask-app                     latest    269bf42596ed
12 days ago   126MB
localstack/localstack         latest    b686f3948f42
3 weeks ago   1.18GB
public.ecr.aws/lambda/python   3.8       348b357f1c82
4 weeks ago   575MB
```

Output Breakdown:

- **`public.ecr.aws/lambda/python:3.8` is now available** (ID: `348b357f1c82`).
- If missing, **re-pull it** using:

  ```
  docker pull public.ecr.aws/lambda/python:3.8
  ```

10. Deleting the Failed Lambda Function

Command:

```
aws --endpoint-url=http://localhost:4566 lambda delete-function \
  --function-name myLambdaFunction
```

Explanation:

- `aws lambda delete-function` → Tries to **delete** the failed Lambda function.

Output:

```
An error occurred (ResourceNotFoundException) when calling
the DeleteFunction operation:
Function not found: arn:aws:lambda:us-east-
1:000000000000:function:myLambdaFunction
```

Output Breakdown:

- **`ResourceNotFoundException`** → The function **does not exist** in LocalStack.
- **Possible Reason:** It was already deleted or never created properly.

11. Listing All Lambda Functions

Command:

```
aws --endpoint-url=http://localhost:4566 lambda list-functions
```

Explanation:

- `aws lambda list-functions` → Lists all available Lambda functions in LocalStack.

Output:

```
---------------
| ListFunctions |
+--------------+
```

Output Breakdown:

- **No functions are listed**, confirming that `myLambdaFunction` does not exist.

Summary & Next Steps

1. The **Lambda function failed** due to missing runtime image:
   - `StateReason: Error while creating lambda: public.ecr.aws/lambda/python:3.8`
2. The **Lambda runtime image was manually pulled** using:

   ```
   docker pull public.ecr.aws/lambda/python:3.8
   ```

3. The function **no longer exists** (`delete-function` failed).
4. **Recreate the function after fixing issues:**

   ```
   aws --endpoint-url=http://localhost:4566 lambda create-function \
     --function-name myLambdaFunction \
     --runtime python3.8 \
     --role arn:aws:iam::000000000000:role/execution_role \
     --handler lambda_function.lambda_handler \
     --zip-file fileb://lambda_function.zip
   ```

12. Listing All Lambda Functions

Command:

```
aws --endpoint-url=http://localhost:4566 lambda list-functions
```

Explanation:

- `aws lambda list-functions` → Lists all available **Lambda functions** in LocalStack.

Output:

```
----------------------------------------
|          ListFunctions               |
----------------------------------------
|           Functions                  |
----------------------------------------
|  CodeSha256   |  G16xwAmelqgPBfzQVM  |
|               |  lj7SR/dZS5lTV2WSY8m  |
|               |  GqvvRw=              |
```

```
-------------------------------------
|  CodeSize     |  312 KB             |
-------------------------------------
|  FunctionArn  |  arn:aws:lambda:us-  |
|               |  east-1:000000000000: |
|               |  function:myLambda    |
|               |  Function             |
-------------------------------------
|  FunctionName |  myLambdaFunction    |
-------------------------------------
|  Handler      |  lambda_function.    |
|               |  lambda_handler       |
-------------------------------------
|  Runtime      |  python3.8           |
-------------------------------------
|  Timeout      |  3 seconds           |
-------------------------------------
|  Version      |  $LATEST             |
-------------------------------------
```

Output Breakdown:

- **`FunctionName`** → Confirms `myLambdaFunction` exists.
- **`Runtime`** → Uses `Python 3.8`.
- **`CodeSize`** → `312 KB`, indicating function code is properly uploaded.
- **`Version`** → Currently on `$LATEST`.
- **Next Step:** Function is **ready for invocation**.

13. Invoking the Lambda Function

Command:

```
aws --endpoint-url=http://localhost:4566 lambda invoke \
  --function-name myLambdaFunction output.txt
```

Explanation:

- `aws lambda invoke` → Executes the **Lambda function**.
- `--function-name myLambdaFunction` → Specifies the function to invoke.
- `output.txt` → Saves the response to `output.txt`.

Output:

```
-----------------------------------
|            Invoke               |
+------------------+--------------+
|  ExecutedVersion | StatusCode   |
+------------------+--------------+
|  $LATEST         |  200         |
+------------------+--------------+
```

Output Breakdown:

- **`ExecutedVersion: $LATEST`** → Confirms the latest version was run.
- **`StatusCode: 200`** → The function **executed successfully**.
- **Next Step:** Verify output contents.

14. Checking Lambda Output

Command:

```
type output.txt
```

Explanation:

- `type output.txt` → Displays the contents of the **output file**.

Output:

```
{ "statusCode": 200, "body": "Hello from LocalStack Lambda!" }
```

Output Breakdown:

- **`statusCode: 200`** → Confirms successful execution.
- **`body: "Hello from LocalStack Lambda!"`** → Function returned expected response.
- **Next Step:** The function is **fully operational**. 🎉

Summary

- **Lambda function created successfully**
- **Function listed in LocalStack**

- Invoked the function successfully
- Received correct output (`"Hello from LocalStack Lambda!"`)

15. Invoking Lambda with Payload

Command:

```
aws --endpoint-url=http://localhost:4566 lambda invoke \
  --function-name myLambdaFunction \
  --payload file://event.json output.txt
```

Explanation:

- `aws lambda invoke` → Calls the Lambda function.
- `--function-name myLambdaFunction` → Specifies the function to invoke.
- `--payload file://event.json` → Sends input data from `event.json` .
- `output.txt` → Stores the Lambda response.

Output:

```
-----------------------------------
|             Invoke              |
+------------------+--------------+
|  ExecutedVersion | StatusCode   |
+------------------+--------------+
|  $LATEST         |  200         |
+------------------+--------------+
```

Output Breakdown:

- **`ExecutedVersion: $LATEST`** → Confirms the latest version was used.
- **`StatusCode: 200`** → Function executed successfully.
- **Next Step:** Check the contents of `output.txt` .

16. Checking the Lambda Output (First Execution)

Command:

```
type output.txt
```

Explanation:

- `type output.txt` → Displays the function's response.

Output:

```
{
  "statusCode": 200,
  "body": "{\"message\": \"Hello, Madhurima!\",
  \"processedNumber\": 10, \"version\": \"1.0.0\"}"
}
```

Output Breakdown:

- **`"message": "Hello, Madhurima!"`** → Function processed the input correctly.
- **`"processedNumber": 10`** → Lambda function logic processed an initial number.
- **`"version": "1.0.0"`** → Confirms the function version.
- **Next Step:** Invoke the function again to check if output changes.

17. Invoking Lambda Again with Payload

Command:

```
aws --endpoint-url=http://localhost:4566 lambda invoke \
  --function-name myLambdaFunction \
  --payload file://event.json output.txt
```

Explanation:

- **Same invocation as before** to observe if the output changes.

Output:

```
---------------------------------
|            Invoke             |
+------------------+------------+
|  ExecutedVersion | StatusCode |
+------------------+------------+
```

```
|  $LATEST         |  200         |
+------------------+--------------+
```

Output Breakdown:

- **Function executed successfully again (** `StatusCode: 200` **).**
- **Next Step:** Check the updated contents of `output.txt`.

18. Checking the Lambda Output (Second Execution)

Command:

```
type output.txt
```

Explanation:

- `type output.txt` → Displays the latest function response.

Output:

```
{
  "statusCode": 200,
  "body": "{\"message\": \"Hello, Madhurima!\",
  \"processedNumber\": 60, \"version\": \"1.0.0\"}"
}
```

Output Breakdown:

- `"processedNumber": 60` → The function logic modified the number (possibly an accumulation or randomization).
- `"message": "Hello, Madhurima!"` → Message remains unchanged.
- **Next Step:** Investigate how the function modifies the `processedNumber` (could be a counter, increment logic, or randomization).

Summary

- **Lambda function successfully invoked with payload multiple times**
- **Output modified between executions (** `processedNumber` **changed from 10 to 60)**
- **Lambda function is stable and executing as expected**