Simulating Lambda Locally

In this lesson, you will learn how to run the application locally using Docker.

WE'LL COVER THE FOLLOWING

Local Lambda runtime environments

Deploying a new version of a Lambda function takes only a minute or two, which is amazing when you consider how many things need to be created in the background, but this is still too slow for a smooth development flow. Together with Docker, SAM can also simulate the Lambda runtime environment locally, so you can experiment and debug code much faster than when deploying to AWS.

Run the following command from your project directory (`code/app`), which contains the application template (`template.yaml`):

```
sam local start-api --host=0.0.0.0
```

> If you use your local machine to run code, you can simply run the following command:
>
> ```
> sam local start-api
> ```

After you press the **Run** button, you can click on the URL next to `Your app can be found at:` and add `hello` to it to view the application.

Environment Variables

Key:	Value:
AWS_ACCESS_KEY_ID	Not Specified...
AWS_SECRET_ACCE...	Not Specified...
BUCKET_NAME	Not Specified...
AWS_REGION	Not Specified...


```
{
  "body": "{\"message\": \"hello world\"}",
  "resource": "/{proxy+}",
  "path": "/path/to/resource",
  "httpMethod": "POST",
  "isBase64Encoded": false,
  "queryStringParameters": {
    "foo": "bar"
  },
  "pathParameters": {
    "proxy": "/path/to/resource"
  },
  "stageVariables": {
    "baz": "qux"
  },
  "headers": {
    "Accept": "text/html,application/xhtml+xml,application/xml;q=0.9,image/webp,*/*;q=0.8",
    "Accept-Encoding": "gzip, deflate, sdch",
    "Accept-Language": "en-US,en;q=0.8",
    "Cache-Control": "max-age=0",
    "CloudFront-Forwarded-Proto": "https",
    "CloudFront-Is-Desktop-Viewer": "true",
    "CloudFront-Is-Mobile-Viewer": "false",
    "CloudFront-Is-SmartTV-Viewer": "false",
    "CloudFront-Is-Tablet-Viewer": "false",
    "CloudFront-Viewer-Country": "US",
    "Host": "1234567890.execute-api.us-east-1.amazonaws.com",
    "Upgrade-Insecure-Requests": "1",
    "User-Agent": "Custom User Agent String",
    "Via": "1.1 08f323deadbeefa7af34d5feb414ce27.cloudfront.net (CloudFront)",
    "X-Amz-Cf-Id": "cDehVQoZnx43VYQb9j2-nvCh-9z396Uhbp027Y2JvkCPNLmGJHqlaA==",
    "X-Forwarded-For": "127.0.0.1, 127.0.0.2",
    "X-Forwarded-Port": "443",
    "X-Forwarded-Proto": "https"
  },
  "requestContext": {
    "accountId": "123456789012",
    "resourceId": "123456",
    "stage": "prod",
    "requestId": "c6af9ac6-7b61-11e6-9a41-93e8deadbeef",
    "requestTime": "09/Apr/2015:12:34:56 +0000",
    "requestTimeEpoch": 1428582896000,
    "identity": {
      "cognitoIdentityPoolId": null,
      "accountId": null,
      "cognitoIdentityId": null,
      "caller": null,
      "accessKey": null,
      "sourceIp": "127.0.0.1",
```

```
      "cognitoAuthenticationType": null,
      "cognitoAuthenticationProvider": null,
      "userArn": null,

      "userAgent": "Custom User Agent String",
      "user": null
    },
    "path": "/prod/path/to/resource",
    "resourcePath": "/{proxy+}",
    "httpMethod": "POST",
    "apiId": "1234567890",
    "protocol": "HTTP/1.1"
  }
}
```

`sam local start-api` should start up a local API Gateway emulation and a local Lambda execution environment, and print out the details similar to the output below:

```
$ sam local start-api
2019-02-10 15:22:17 Found credentials in shared credentials file: ~/.aws/c
redentials
2019-02-10 15:22:17 Mounting HelloWorldFunction at http://127.0.0.1:3000/h
ello [GET]
2019-02-10 15:22:17 You can now browse to the above endpoints to invoke yo
ur functions. You do not need to restart/reload SAM CLI while working on y
our functions, changes will be reflected instantly/automatically. You onl
y need to restart SAM CLI if you update your AWS SAM template
2019-02-10 15:22:17 * Running on http://127.0.0.1:3000/ (Press CTRL+C to q
uit)
```

If you get an error stating that Docker is not installed (for example '*Error: Running AWS SAM projects locally requires Docker. Have you got it installed?*'), check if your Docker Desktop software is running. SAM needs Docker to simulate the Lambda environment locally. The error will ask about Docker not being installed even if it is installed but just not running.

If you run locally, check the URL printed in the console and add `hello` to it (the sample template maps a Lambda function to the `/hello` URL). Based on the log in the output above, the URL should be `http://127.0.0.1:3000/hello` which you will be able to open in your browser to see the function running.

> ### SAM and downloading Docker images
>
> To save disk space, SAM skips downloading the Docker images for simulating Lambda runtimes during initial installation. The first time

> you try to execute a function in a simulated runtime, it will retrieve the
>
> relevant image and store it locally. This means that the initial request might take a while to execute. Subsequent requests will be quick, though.

Rebuilding and invoking the Lambda function

This simulation will automatically reload the function code if you rebuild the project.

For example, open `hello-world/app.js` in the widget above and change the message from `hello world` to something else. Press the **Run** button to register the changes.

Then run `sam build` in a new terminal (keep the simulated function running) after changing the directory to `usercode/app` using the `cd` command and access the URL again. You should see the updated message.

You can also use `sam local` to send events to individual Lambda functions (even for stacks that do not have an API Gateway component). To do that, use `invoke` followed by the logical name of the function from the stack template. If you do not specify an event from the command line, SAM will wait for the event on the console input. Alternatively, you can pass `--event` followed by a file name containing the test event.

Now that we're logging incoming events, it's easy to just take sample events from remote CloudWatch logs and replay them locally in a simulated environment for debugging. List the logs, copy an event (look for a JSON structure), and then save it to `event.json`. Press the **Run** button.

Run the following command from the project directory (`usercode/app`) in to send the event to the function running in the simulated Lambda environment:

```
sam local invoke HelloWorldFunction --event event.json
```

Isn't that great? Hopefully, you are enjoying the course so far. In the next lesson, you'll explore how to debug functions.