Invoke Lambda function from Java application deployed to Docker

Contents

[1. Prerequisites 2](#_Toc157252507)

[2. Import libraries 2](#_Toc157252508)

[3. Code to invoke function and how to use it 2](#_Toc157252509)

[4. AWS create user and get keys 3](#_Toc157252510)

[5. Configure AWS keys on local machine 4](#_Toc157252511)

[6. Get local configured keys to Docker 5](#_Toc157252512)

# 1. Prerequisites

For this you will need:

* AWS account
* Local AWS Cli
* Java project

# 2. Import libraries

For libraries we will need general AWS SDK and AWS SDK for lambda.

Gradle:

implementation(platform("software.amazon.awssdk:bom:2.21.1"))  
implementation 'software.amazon.awssdk:lambda'

# 3. Code to invoke function and how to use it

public String invokeFunction(LambdaClient awsLambda, String functionName, JSONObject input) {  
 InvokeResponse res = null;  
 try {  
 String json = input.toString();  
 SdkBytes payload = SdkBytes.*fromUtf8String*(json);  
  
 // Setup an InvokeRequest.  
 InvokeRequest request = InvokeRequest.*builder*()  
 .functionName(functionName)  
 .payload(payload)  
 .build();  
  
 res = awsLambda.invoke(request);  
 String value = res.payload().asUtf8String();  
 return value;  
 } catch (LambdaException e) {  
 System.*err*.println(e.getMessage());  
 }  
 return null;  
}

Parameters:

* awsLambda: can be declared as static final variable, where we need to declare the region from where our lambda function is created. Ex:

private static final LambdaClient *awsLambda* = LambdaClient.*builder*()  
 .region(Region.*EU\_CENTRAL\_1*)  
 .build();

* functionName: the name of the function we want to call
* input: data we want to send to the function

How it works: With the given parameters we need to create first an InvokeRequest where we set the function name and the payload. Then using the LambdaClient awsLambda we invoke our request. After we wait for the request to be completed and transform the received payload in a String format and return it.

Example of how to use it:

JSONObject jsonObj = new JSONObject();  
jsonObj.put("power", power);  
jsonObj.put("weight", weight);  
String res = lambdaInvoke.invokeFunction(*awsLambda*, *lambdaPowertoWeightFunction*, jsonObj);  
return Double.*valueOf*(Math.*floor*(Double.*parseDouble*(res) \* 100) / 100).describeConstable();

We create a json object where we put the date we want to send to the lambda function. In this case the result payload consists of a floating value.

# 4. AWS create user and get keys

Now if you try to run the function you most likely will get an error regarding java needing authorization to invoke this function. As presented in the aws documentation there are multiple ways to approach this issue, but now we will create a new user in our aws account, apart from the root user, give it rights to execute lambda functions and create log in keys for it.

1. From aws dashboard search IAM
2. Then go to User -> Create new User, set the name then review and create
3. Here you might need to set a one time use password, go to an incognito tab, log in with this password and set a new one.
4. Go back to User -> select your new user -> Permissions tab -> Add permissions -> Add permissions
5. Here it depends on how much permissions you want to add to this uses and how you want to manage the permissions of this account, but for the easy way of this tutorial we will attach the full access policy to this account directly. Select the Attach policies directly and search for AdministratorAccess, select it and add it the account.

A screenshot of a computer

Description automatically generated

!!! Note for a real application you might not want to give administrative access to all accounts

1. Go back to Users-> your new user
2. Select Security Credentials and scroll to Access Keys
3. Create new access key
4. Save the information about the key id and secrete
5. You should be able to see it in the user dashboard A screenshot of a computer

   Description automatically generated

# 5. Configure AWS keys on local machine

Now open your preferred terminal (CMD/PowerShell for windows) and first check your aws cli installation with this command: aws –version.

Output should look like: 

To set your account we will use the command: aws configure and put our Access Key id and Secrete Access key combination

A black background with white text

Description automatically generated

After this we should see in our user folder (C:/Users/<Your User>) a new folder .aws with 2 files in it.

A screenshot of a computer

Description automatically generated

The credentials file should contain your Key Id and Secrete. Something like this.A screenshot of a computer

Description automatically generated

Now the AWS SDK will now to use this credentials to authorize and be able to invoke the java function from your machine.

# 6. Get local configured keys to Docker

To get this configuration to docker image you will need to set 2 things:

1. enviroment variables: AWS\_ACCESS\_KEY\_ID, AWS\_SECRET\_ACCESS\_KEY

2. copy information from .aws folder to docker: on /root/.aws folder

A screenshot of a computer screen

Description automatically generated

Now the application running in docker can invoke lambda functions from AWS.