3-1 Custom Authorizer

Tuesday, August 24, 2021

12:58 PM

What is a custom authorizer?

A custom authorizer uses lambda behind the scenes.

It tells API Gateway to call a specific lambda function, pass some data, some information from the incoming request to that function. The function then has to run some code to validate or to identify that user.

Authorizers 
Authorizers enable you to control access to your APIs using Amazon Cognito user Pools or a Lambda function. 
+ Create New Authorizer 
Create Authorizer 
Name • 
Type * O 
Lambda 
Lambda Function • O 
Lambda Invoke Role O 
Lambda Event Payload * O 
Token Source' O 
Authorization Caching O 
Enabled 
Create 
Cognito 
Request 
Token Validation O 
TTL (seconds) 
Cancel Return IAM Policy 
"Effect": "Allow", 
"Action": "execute-api" 
Return Principal ID (User Id) 
mentation 
Return Context Object 
"YourData": "YourValue", 
Age": "28" 

We need these

**Code**

1 
2 
3 
4 
5 
6 
7 
8 
9 
10 
11 
12 
13 
14 
15 
16 
17 
18 
19 
20 
21 
22 
23 
24 
25 
26 
27 
28 
29 
30 
31 
32 
33 
34 
35 
36 
37 
38 
39 
41 
43 
45 
exports. handler = 
const token = 
/ (Use token 
if(token = 
(event, context, callback) 
event . authorizationToken; 
'allow') { 
const policy = genpolicy( 'allow', event.meth0dArn); 
// methodArn some information we get from API passed into this function 
const principalld = 
alsdjovdnkf' ; 
const context = { 
simpleAuth: true 
const response = 
principalld: principalld, 
policyDocument: policy, 
context: context 
callback(null, response); 
else if (token — 
'deny•) { 
const policy = genP01icy( 'deny' , event . methodArn); 
// methodArn some information we get from API passed into this function 
const principalld = 
alsdjovdnkf' ; 
const context = { 
simpleAuth: true 
const response = { 
principalld: principalld, 
policyDocument: policy, 
context: context 
callback(null, response) ; 
else { 
callback( ' Unauthorized ' ) ; 
function genP01icy(effect, resource) { 
const policy = 
policy .Version = 
2912-10-17' ; 
policy. Statement = [I; 
const stmt = 
stmt. Action = 
execute-api : Invoke ; 
stmt . Effect = effect; 
stmt. Resource = resource; 
policy. Statement . push ( stmt) ; 
return policy; The following input data is provided to you: 
"type" ; "TOKEN" , 
"authorizationToken" : "<cal ler- suppli ed- token>" , 
"methodArn" : "arn :aws : execute -api : : : capild 
<caller- supplied-token> is the token you actually receive. 
You configure how to extract the token from the incoming 
request in API gateway. 
methodArn simply refers to the endpoint on which this 
authorizer was triggered. The following output data has to be provided by your 
function (via 
callback() 
"principalld": "yyyyyyyy% 
"policyoocunent": { 
"Version" : 
"2012-10-17", 
"Statement": [ 
// The principal user identification 
"Action": "execute-api : Invoke" , 
"Allowl Deny" , 
"Effect": 
"Resource : 
" "arn: aus :execute-api : cregionld>: caccountld>: 
"context": { 
"stringKey": "value", 
"numberKey": "I" , 
"booleanKey" : 
"true" 
principalld simply is the user identifier. 
policyDocument is a JS object which uses the IAM policy 
structure (as shown in the above example). 
context is the only optional attribute. It simply is an object 
of key-value pairs of your choice. 