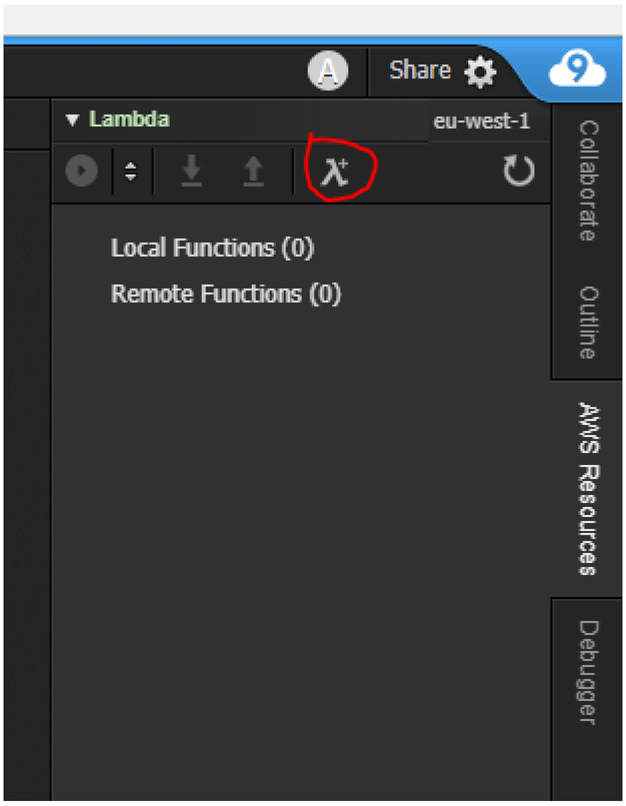
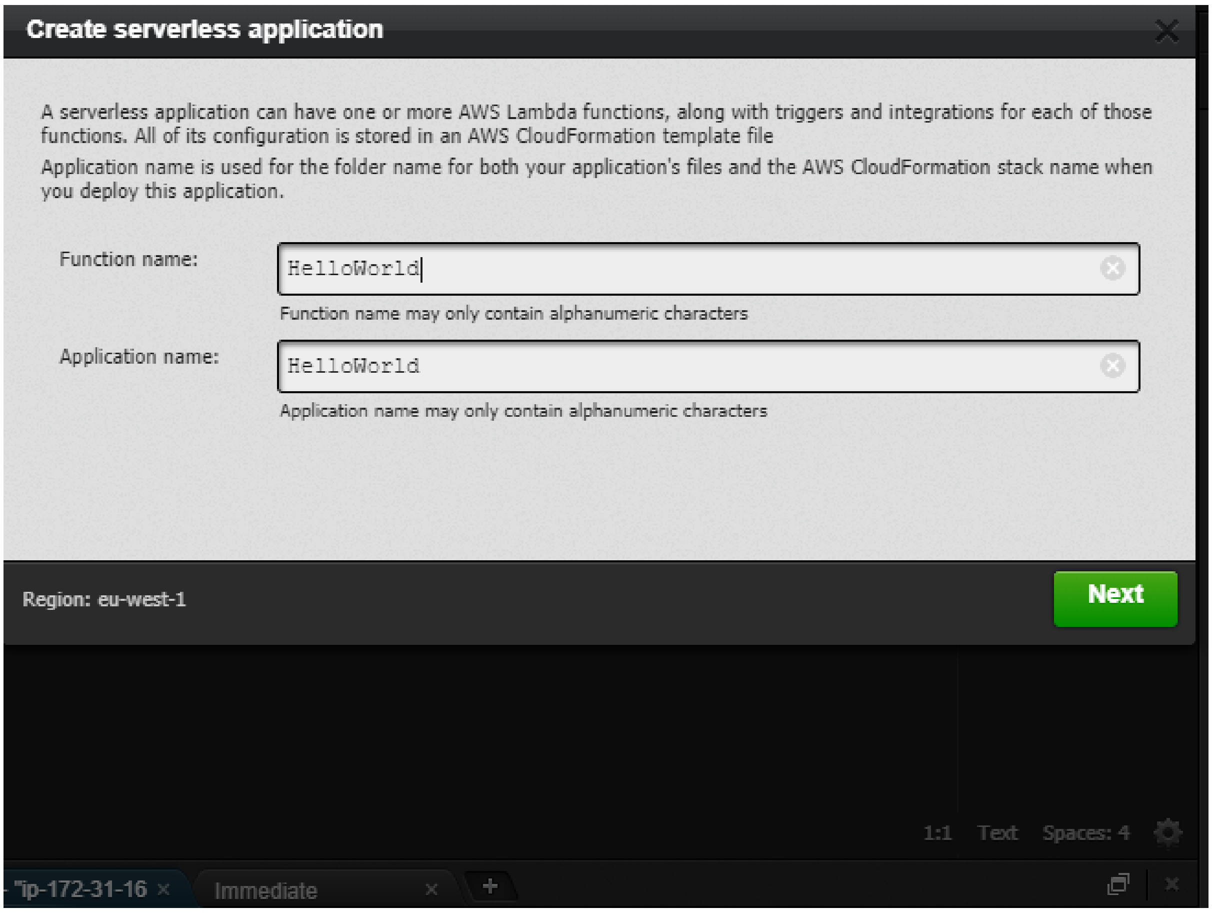
Building HelloWorld

Let’s start off with organizing our work and creating a new folder for the mandatory “Hello World” example:

Create a new lambda function by using the AWS Resources menu and selecting the lambda+ icon:





Create serverless application

A serverless application can have one or more AWS Lambda functions, along with triggers and integrations for each of those functions. All of its configuration is stored in an AWS CloudFormation template file

Application name is used for the folder name for both your application's files and the AWS CloudFormation stack name when you deploy this application.

Function name: HelloWorld

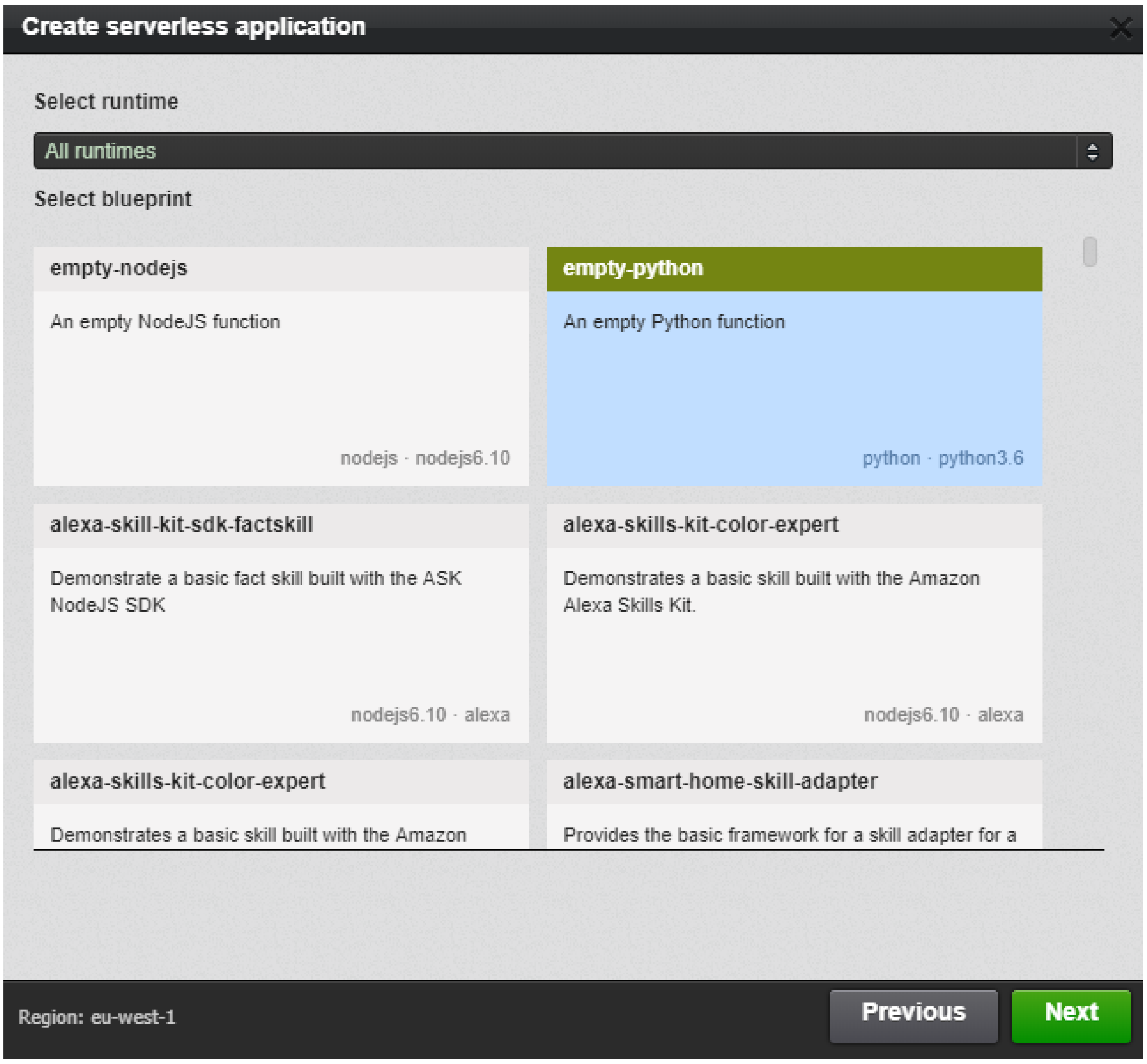
Function name may only contain alphanumeric characters

Application name: HelloWorld

Application name may only contain alphanumeric characters

Region: eu-west-1

The function needs to be developed. Let's selected Python 3.6+ as the run time to write the Hello World example as Python code:



Create serverless application

Select runtime

All runtimes

Select blueprint

empty-nodejs
An empty NodeJS function
nodejs · nodejs6.10

empty-python
An empty Python function
python · python3.6

alexa-skill-kit-sdk-factskill
Demonstrate a basic fact skill built with the ASK NodeJS SDK
nodejs6.10 · alexa

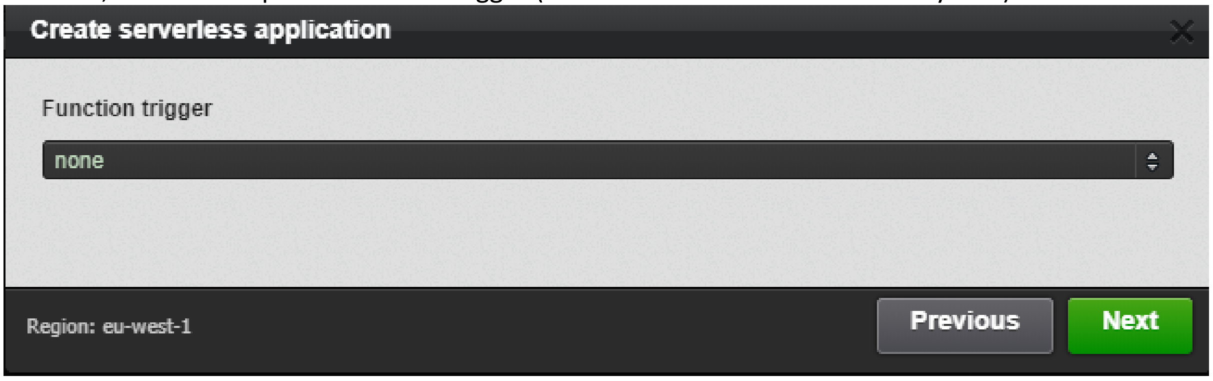
alexa-skills-kit-color-expert
Demonstrates a basic skill built with the Amazon Alexa Skills Kit.
nodejs6.10 · alexa

alexa-skills-kit-color-expert
Demonstrates a basic skill built with the Amazon

alexa-smart-home-skill-adapter
Provides the basic framework for a skill adapter for a

Region: eu-west-1

For now, we don't require a function trigger (we will test the function manually first):



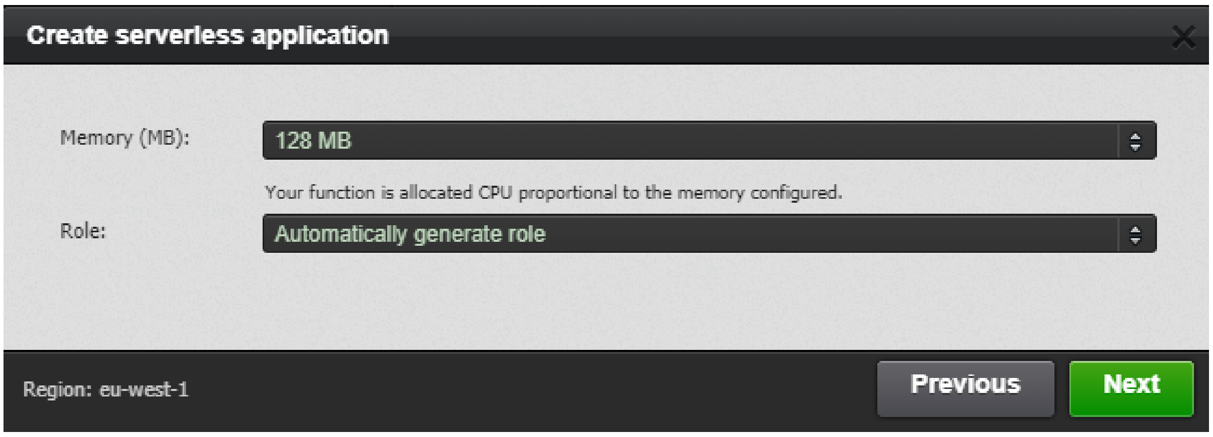
Create serverless application

Function trigger

none

Region: eu-west-1

Also, default memory settings and a default generated role should be enough - we're not doing any heavy lifting, nor are we using any asynchronous functionality:


Create serverless application

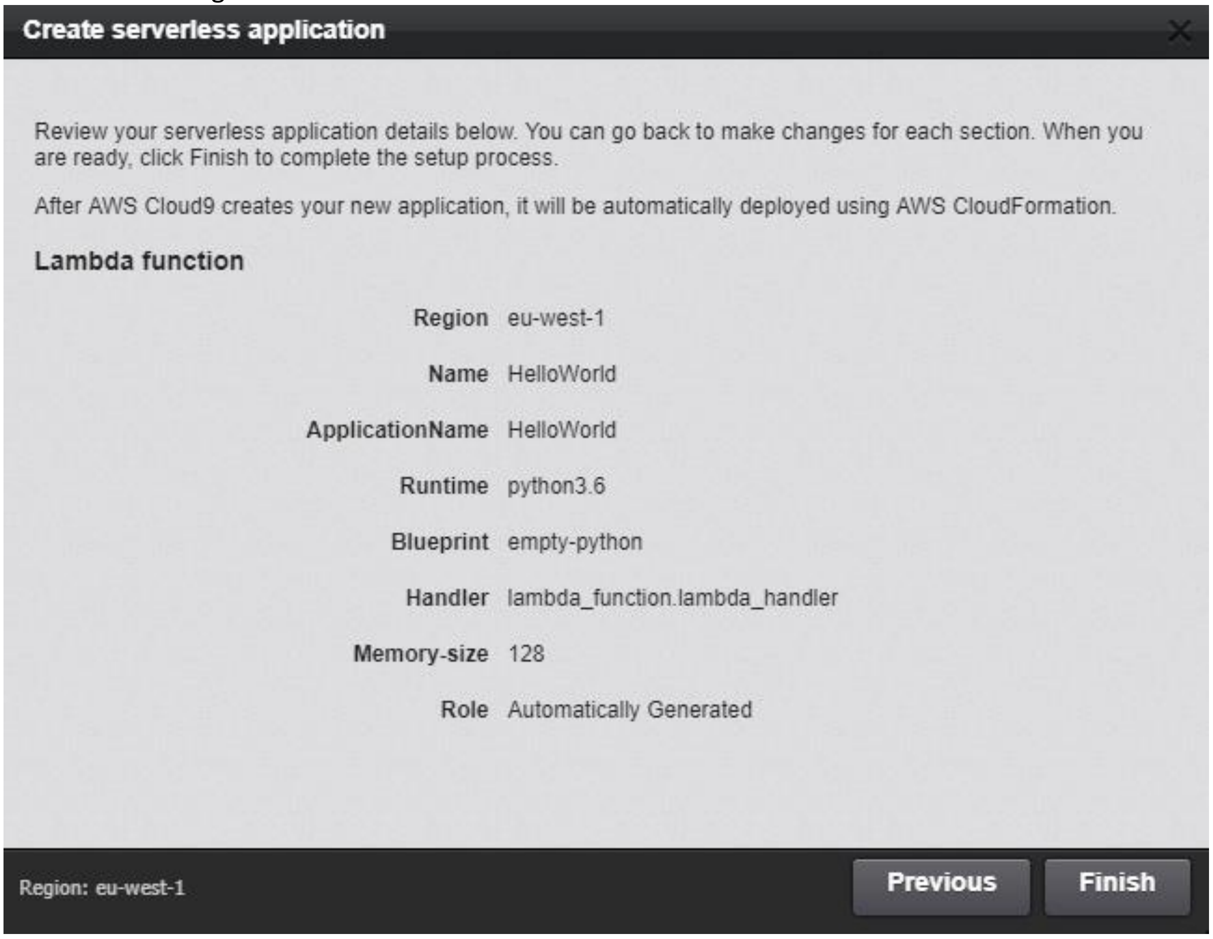
Memory (MB): 128 MB

Your function is allocated CPU proportional to the memory configured.

Role: Automatically generate role

Region: eu-west-1

Review the settings and finish:


Create serverless application

Review your serverless application details below. You can go back to make changes for each section. When you are ready, click Finish to complete the setup process.

After AWS Cloud9 creates your new application, it will be automatically deployed using AWS CloudFormation.

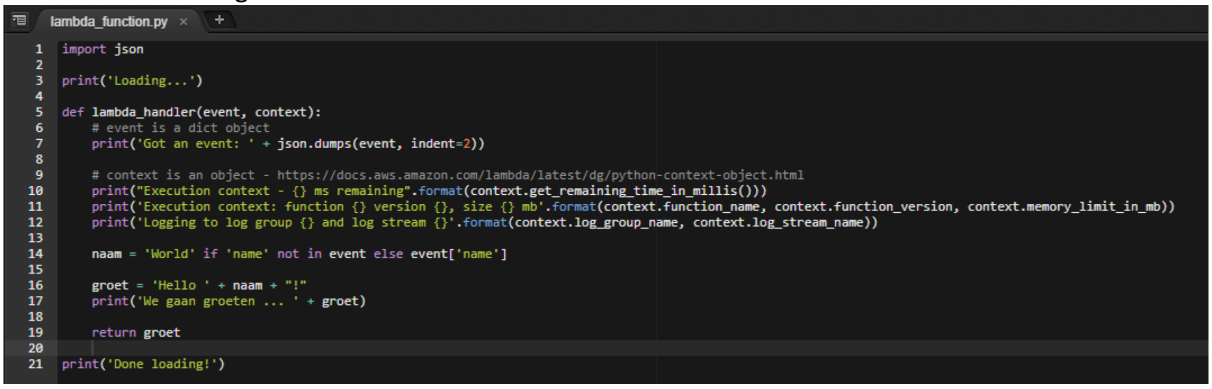
Lambda function

Region	eu-west-1
Name	HelloWorld
ApplicationName	HelloWorld
Runtime	python3.6
Blueprint	empty-python
Handler	lambda_function.lambda_handler
Memory-size	128
Role	Automatically Generated

Region: eu-west-1

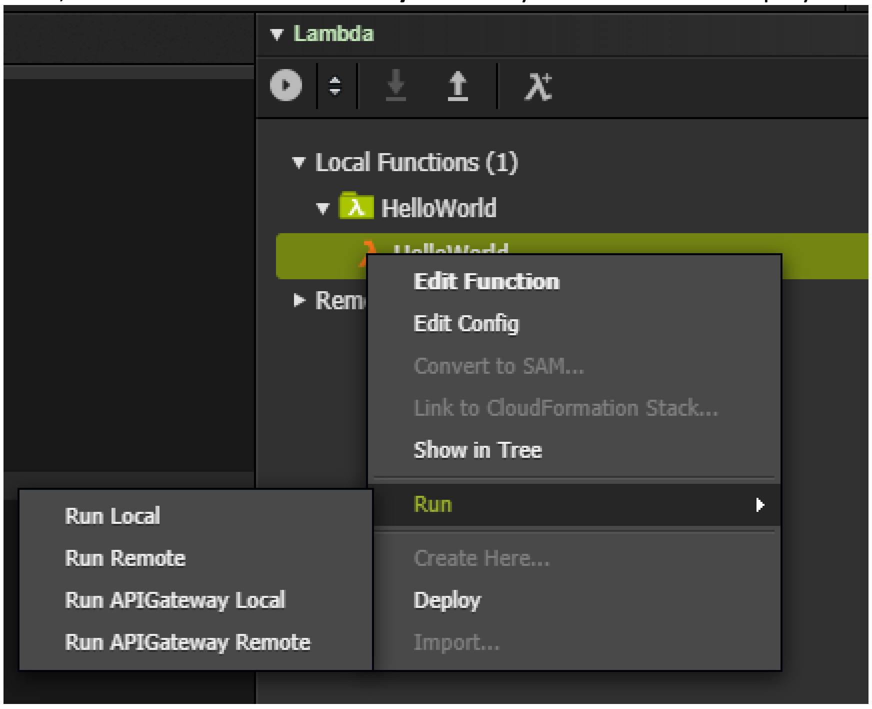
From the setting above, you should note that the Handler is "lambda_function.lambda_handler". What does this mean? Well, if the function is invoked, it will look for a Python **source file** named "lambda_function" (.py, actually) and try and execute the **function** "lambda_handler" inside this file. To isolate this project from other projects in the same IDE, Cloud9 will create a so-called "Virtual Environment" for Python, where all its dependencies will be stored and not pollute the global namespace - reducing the chances of conflicts.

As you can see, the AWS Lambda framework will provide the function with an event and a context: the event is the event that triggered the invocation of the function, the context consists of AWS specific environment information, the so-called execution context. Let’s examine both objects passed in and return a greeting using “Hello “ plus the name if present in the input event, otherwise name should be defaulting to “World” “.

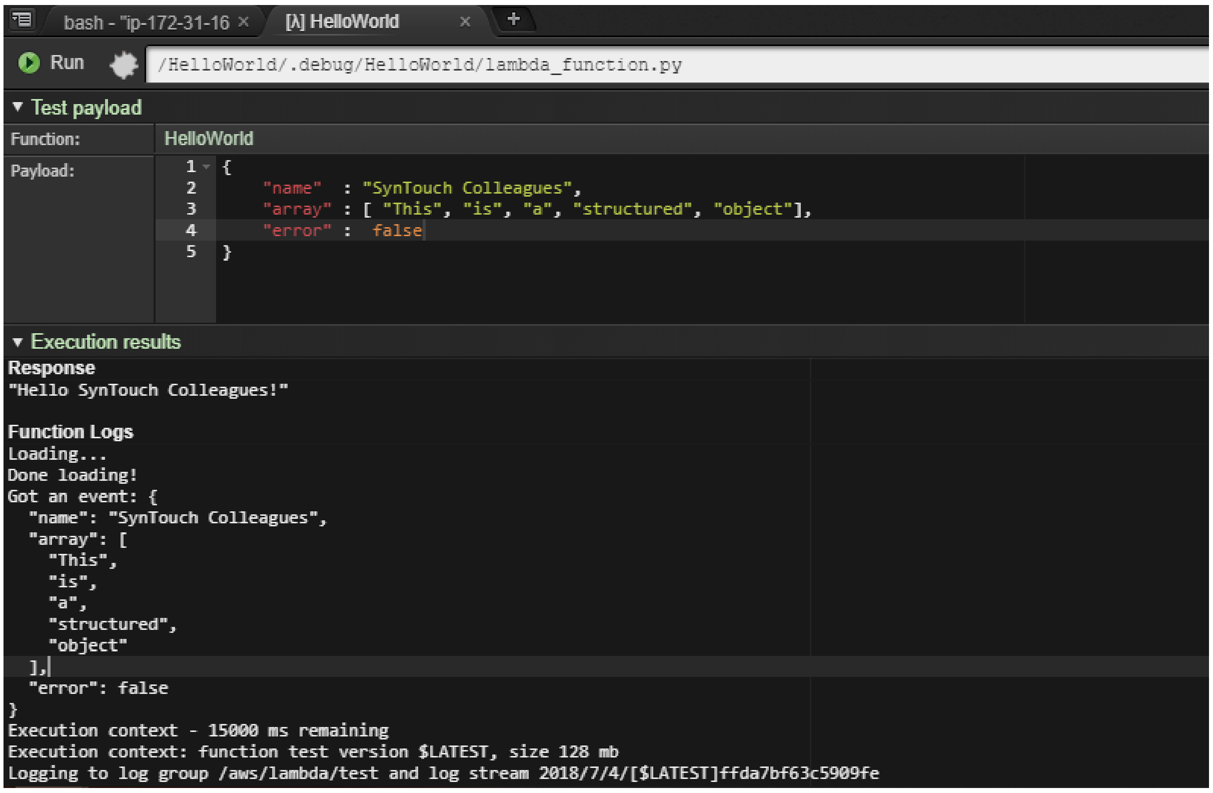


```python
import json

print('Loading...')

def lambda_handler(event, context):
    # event is a dict object
    print('Got an event: ' + json.dumps(event, indent=2))

    # context is an object - https://docs.aws.amazon.com/lambda/latest/dg/python-context-object.html
    print("Execution context - {} ms remaining".format(context.get_remaining_time_in_millis()))
    print('Execution context: function {} version {}, size {} mb'.format(context.function_name, context.function_version, context.memory_limit_in_mb))
    print('Logging to log group {} and log stream {}'.format(context.log_group_name, context.log_stream_name))

    naam = 'World' if 'name' not in event else event['name']

    groet = 'Hello ' + naam + "!"
    print('We gaan groeten ... ' + groet)

    return groet

print('Done loading!')
```

Now, first test the function **locally** - before you even need to deploy it to the real cloud:



This opens a window that lets you specify an **event payload**. As we are expecting a name property at a minimum set this property in the event payload:



```
{
    "name"  : "SynTouch Colleagues",
    "array" : [ "This", "is", "a", "structured", "object"],
    "error" :  false
}
```

Response

```
"Hello SynTouch Colleagues!"
```

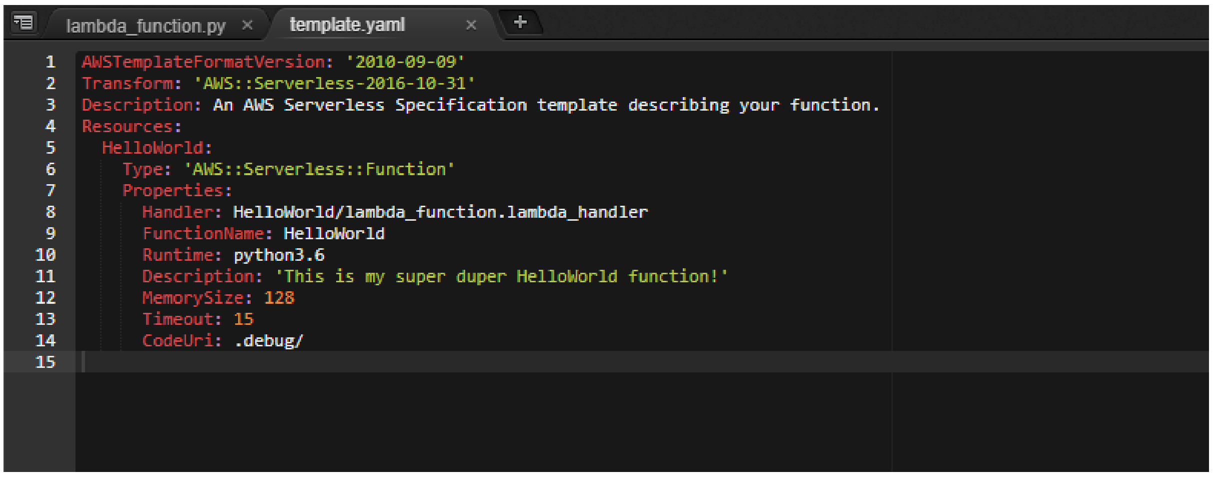
Function Logs

```
Loading...
Done loading!
Got an event: {
  "name": "SynTouch Colleagues",
  "array": [
    "This",
    "is",
    "a",
    "structured",
    "object"
  ],
  "error": false
}
Execution context - 15000 ms remaining
Execution context: function test version $LATEST, size 128 mb
Logging to log group /aws/lambda/test and log stream 2018/7/4/[$LATEST]ffda7bf63c5909fe
```

When supplying a name attribute in the JSON event, the contents are used for the greeting. You can see the function loading (initializing), finish the initialization and then executing on the supplied event. As we're running locally, the actual names ("test" for the function name) may be different from the real run-time environment values.

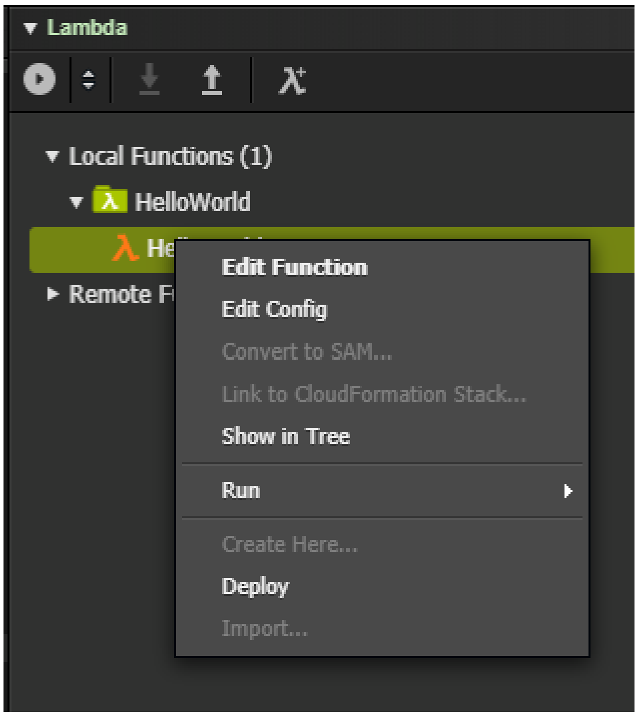
What Cloud9 is doing behind the scenes, is creating a SAM (Serverless Application Model) template to provide all resources required. Actually, this is an extension to the CloudFormation service, which creates or updates an application stack based on an input template ('infrastructure' as code!)

Before deployment, we need to make a small change to the template that is used to provision the stack (if we don't we cannot control the actual name assigned to the function completely): add a FunctionName attribute to the HelloWorld function and set this to HelloWorld:

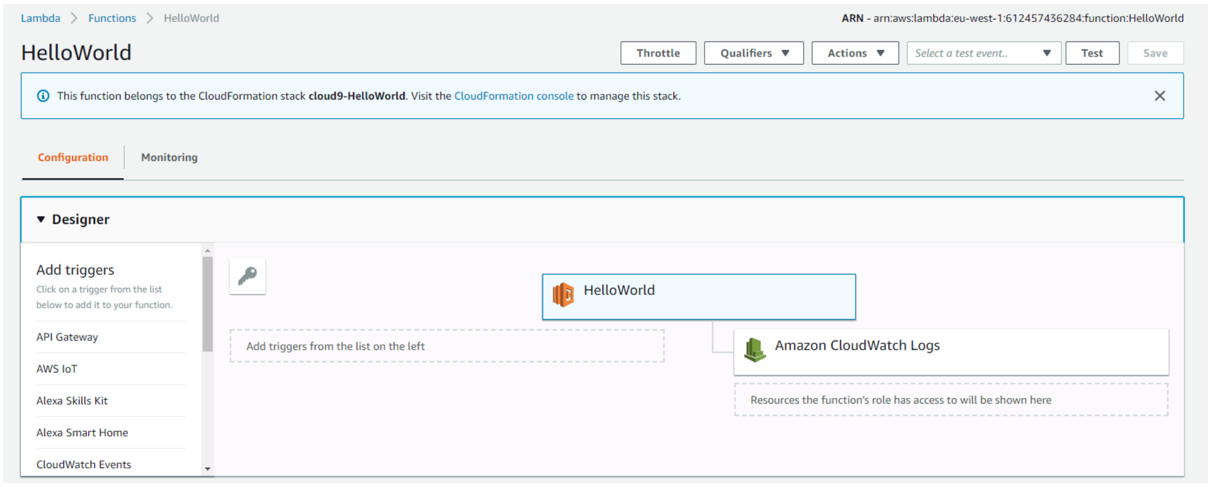


```
AWSTemplateFormatVersion: '2010-09-09'
Transform: 'AWS::Serverless-2016-10-31'
Description: An AWS Serverless Specification template describing your function.
Resources:
  HelloWorld:
    Type: 'AWS::Serverless::Function'
    Properties:
      Handler: HelloWorld/lambda_function.lambda_handler
      FunctionName: HelloWorld
      Runtime: python3.6
      Description: 'This is my super duper HelloWorld function!'
      MemorySize: 128
      Timeout: 15
      CodeUri: .debug/
```

Now the code is ready to rock & roll!



After some seconds, a stack has been provisioned using CloudFormation. You can examine the actual HelloWorld Lambda function from the AWS Lambda console:



ARN - arn:aws:lambda:eu-west-1:612457436284:function:HelloWorld

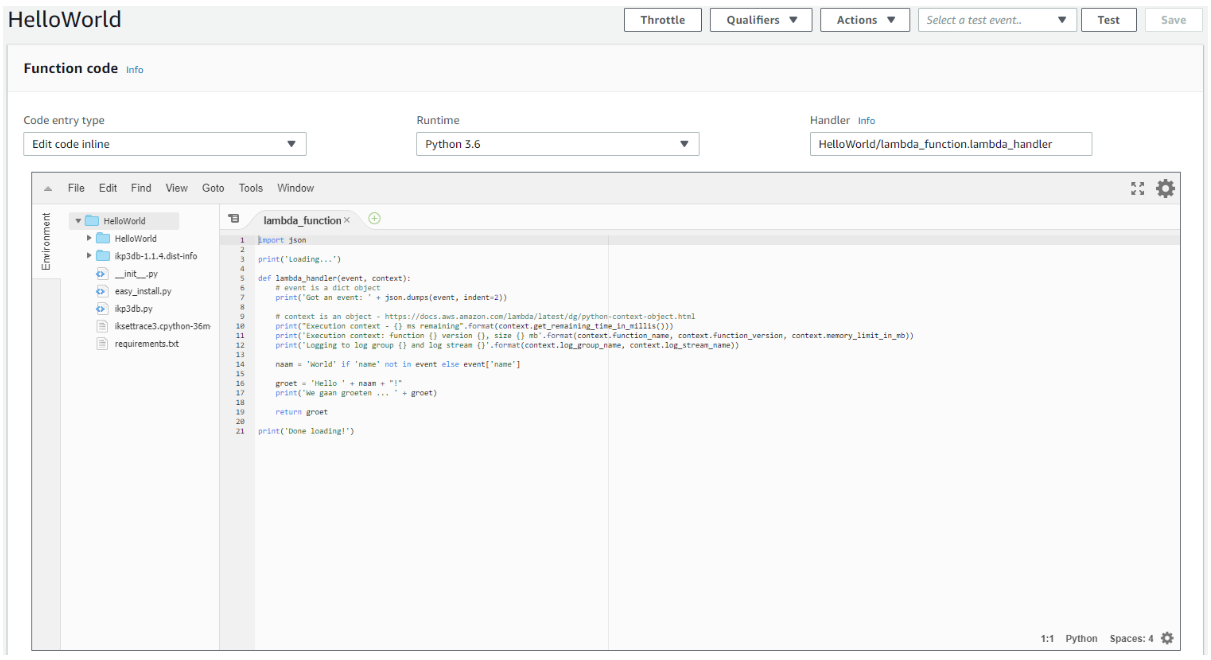
This function belongs to the CloudFormation stack **cloud9-HelloWorld**. Visit the CloudFormation console to manage this stack.

Add triggers
Click on a trigger from the list below to add it to your function.

Add triggers from the list on the left

Resources the function's role has access to will be shown here

Examine and edit the code:



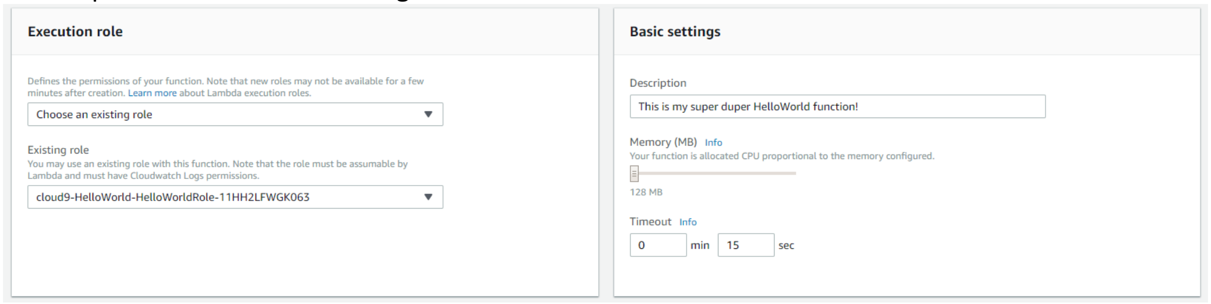
HelloWorld

Function code

Code entry type	Edit code inline
Runtime	Python 3.6
Handler	HelloWorld/lambda_function.lambda_handler

```python
import json

print('Loading...')

def lambda_handler(event, context):
    # event is a dict object
    print('Got an event: ' + json.dumps(event, indent=2))

    # context is an object - https://docs.aws.amazon.com/lambda/latest/dg/python-context-object.html
    print("Execution context - {} ms remaining".format(context.get_remaining_time_in_millis()))
    print('Execution context: function {} version {}, size {} mb'.format(context.function_name, context.function_version, context.memory_limit_in_mb))
    print('Logging to log group {} and log stream {}'.format(context.log_group_name, context.log_stream_name))

    naam = 'World' if 'name' not in event else event['name']

    groet = 'Hello ' + naam + "!"
    print('We gaan groeten ... ' + groet)

    return groet

print('Done loading!')
```

And inspect other relevant settings:



Execution role

Defines the permissions of your function. Note that new roles may not be available for a few minutes after creation. Learn more about Lambda execution roles.

Choose an existing role

Existing role

You may use an existing role with this function. Note that the role must be assumable by Lambda and must have Cloudwatch Logs permissions.

cloud9-HelloWorld-HelloWorldRole-11HH2LFWGK063

Basic settings

Description: This is my super duper HelloWorld function!

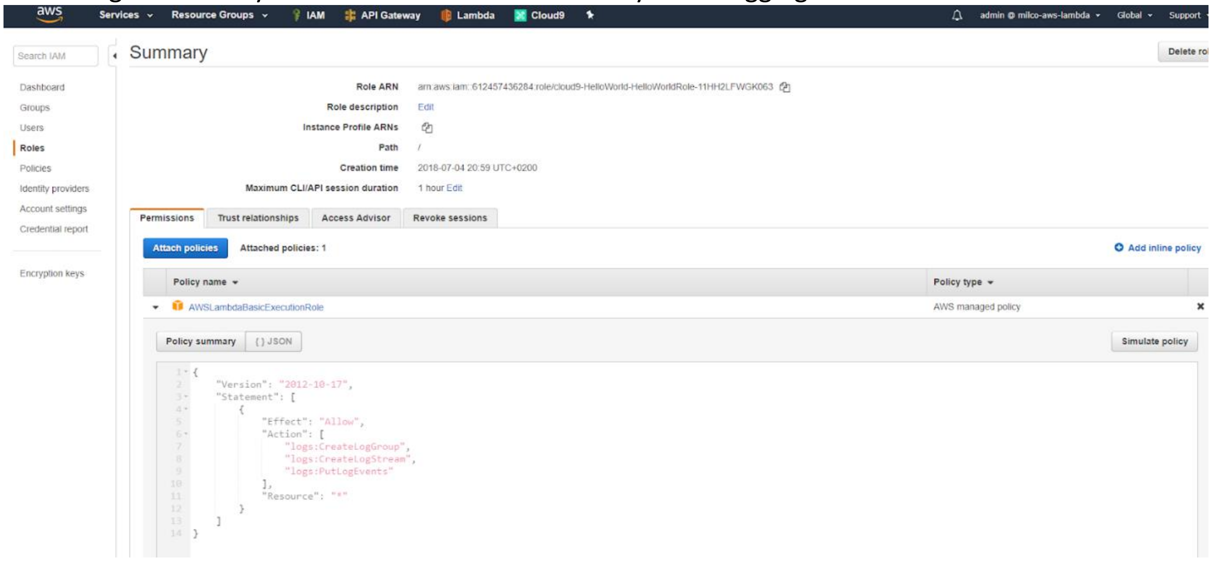
Memory (MB) Info

Your function is allocated CPU proportional to the memory configured.

128 MB

Timeout Info: 0 min 15 sec

The role generated by IAM allows the function to only write logging to CloudWatch:



Summary

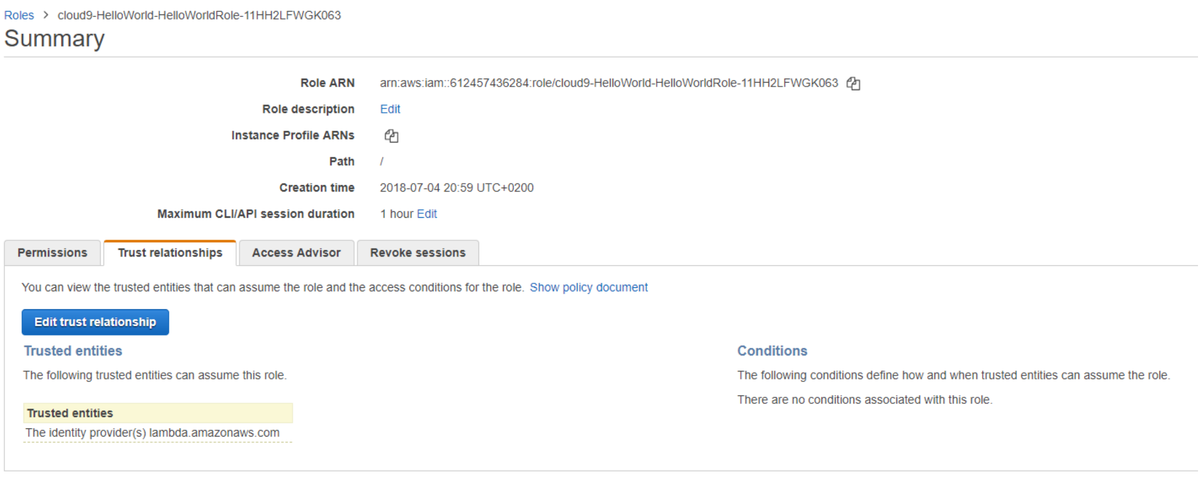
Role ARN	arn:aws:iam::612457436284:role/cloud9-HelloWorld-HelloWorldRole-11HH2LFWGK063
Role description	Edit
Instance Profile ARNs	
Path	/
Creation time	2018-07-04 20:59 UTC+0200
Maximum CLI/API session duration	1 hour Edit

Attached policies: 1

Policy name	Policy type
AWSLambdaBasicExecutionRole	AWS managed policy

```json
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        }
    ]
}
```

And only AWS Lambda is allows to assume this role:



Roles > cloud9-HelloWorld-HelloWorldRole-11HH2LFWGK063

Summary

Role ARN	arn:aws:iam::612457436284:role/cloud9-HelloWorld-HelloWorldRole-11HH2LFWGK063
Role description	Edit
Instance Profile ARNs	
Path	/
Creation time	2018-07-04 20:59 UTC+0200
Maximum CLI/API session duration	1 hour Edit

You can view the trusted entities that can assume the role and the access conditions for the role. Show policy document

Trusted entities

The following trusted entities can assume this role.

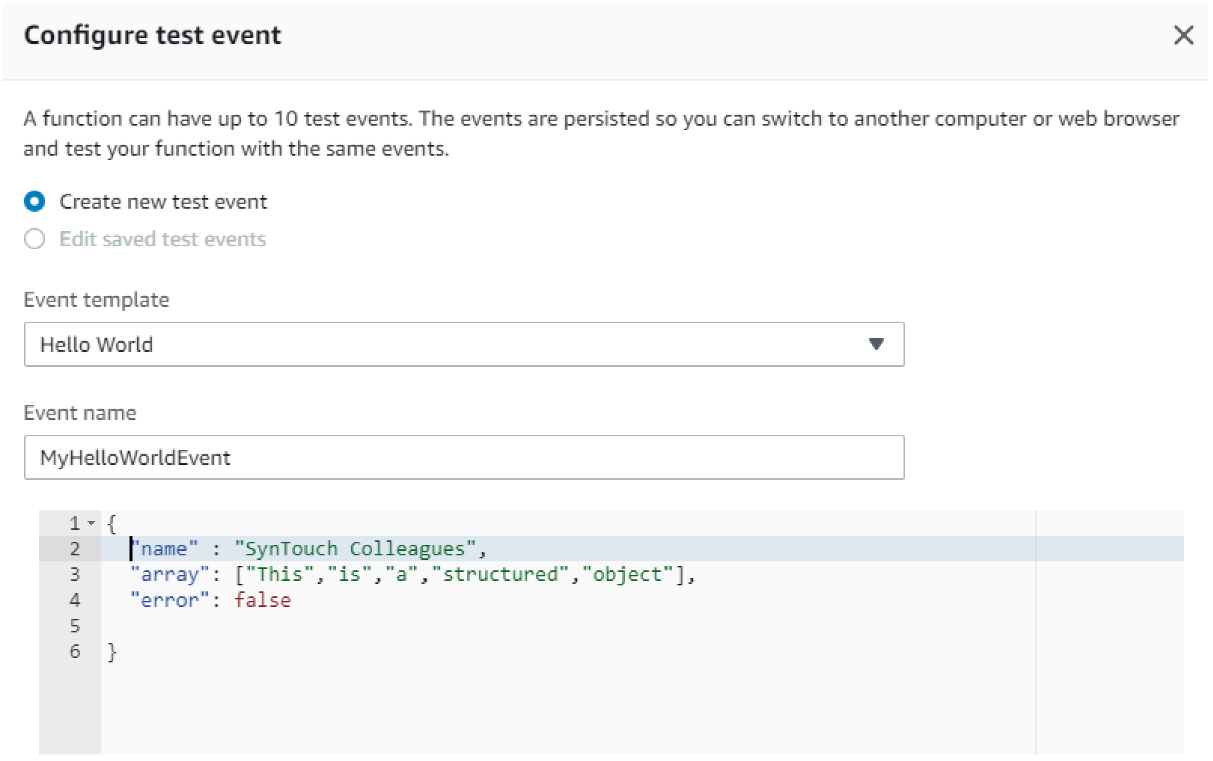
Trusted entities
The identity provider(s) lambda.amazonaws.com

Conditions

The following conditions define how and when trusted entities can assume the role.

There are no conditions associated with this role.

Return to the AWS HelloWorld Lambda function and create a new test event by using the Test button:



Configure test event

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

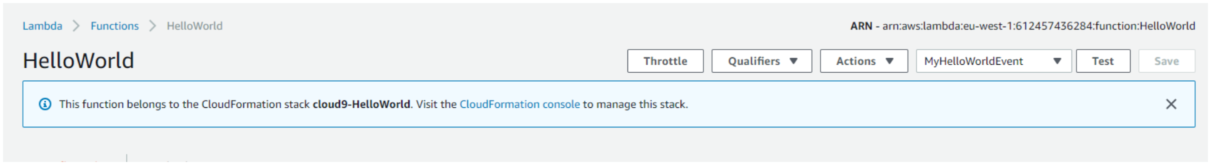
(● Create new test event) Edit saved test events

Event template: Hello World

Event name: MyHelloWorldEvent

```
{
  "name" : "SynTouch Colleagues",
  "array": ["This","is","a","structured","object"],
  "error": false

}
```

Save the event and test!



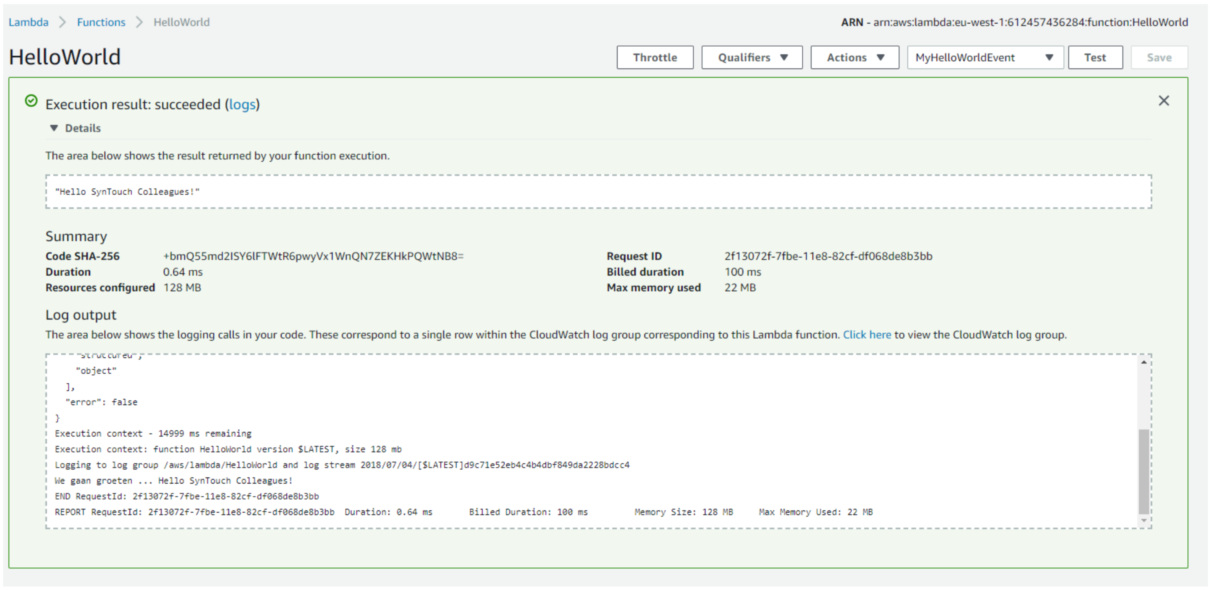
Lambda > Functions > HelloWorld

ARN - arn:aws:lambda:eu-west-1:612457436284:function:HelloWorld

HelloWorld

This function belongs to the CloudFormation stack **cloud9-HelloWorld**. Visit the CloudFormation console to manage this stack.

The execution logs provide you with the actual output, but also with useful information about the duration, the duration you will be billed for (rounded up to 100 ms) and the actual memory consumed:



ARN - arn:aws:lambda:eu-west-1:612457436284:function:HelloWorld

HelloWorld

Execution result: succeeded (logs)

▼ Details

The area below shows the result returned by your function execution.

```
"Hello SynTouch Colleagues!"
```

Summary

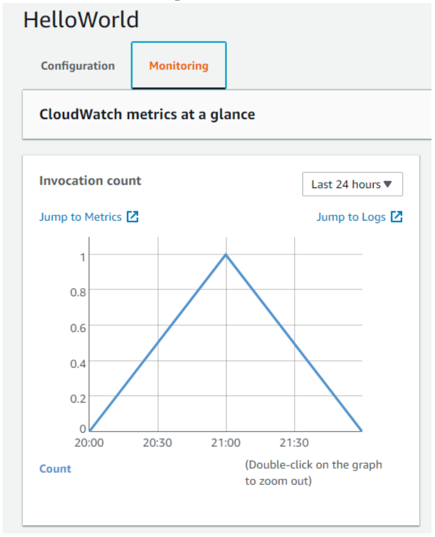
Code SHA-256	+bmQ55md2lSY6lFTWtR6pwyVx1WnQN7ZEKHkPQWtNB8=	**Request ID**	2f13072f-7fbe-11e8-82cf-df068de8b3bb
Duration	0.64 ms	**Billed duration**	100 ms
Resources configured	128 MB	**Max memory used**	22 MB

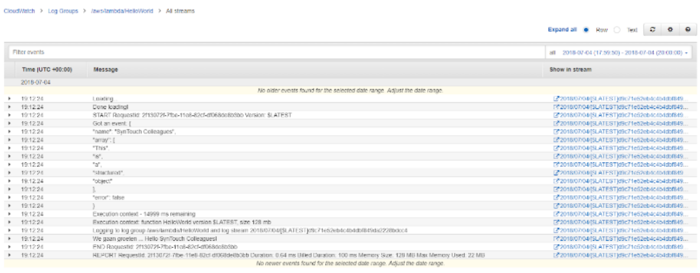
Log output

The area below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. Click here to view the CloudWatch log group.

```
    "object"
  ],
  "error": false
}
Execution context - 14999 ms remaining
Execution context: function HelloWorld version $LATEST, size 128 mb
Logging to log group /aws/lambda/HelloWorld and log stream 2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849da2228bdcc4
We gaan groeten ... Hello SynTouch Colleagues!
END RequestId: 2f13072f-7fbe-11e8-82cf-df068de8b3bb
REPORT RequestId: 2f13072f-7fbe-11e8-82cf-df068de8b3bb  Duration: 0.64 ms      Billed Duration: 100 ms        Memory Size: 128 MB     Max Memory Used: 22 MB
```

The monitoring tab returns miscellaneous statistics on your function, but also offers access to the cloudwatch log files where this time period was logged:





CloudWatch > Log Groups > /aws/lambda/HelloWorld > All streams
Expand all
Row
Text
Filter events
all 2018-07-04 (17:59:50) - 2018-07-04 (20:00:00)

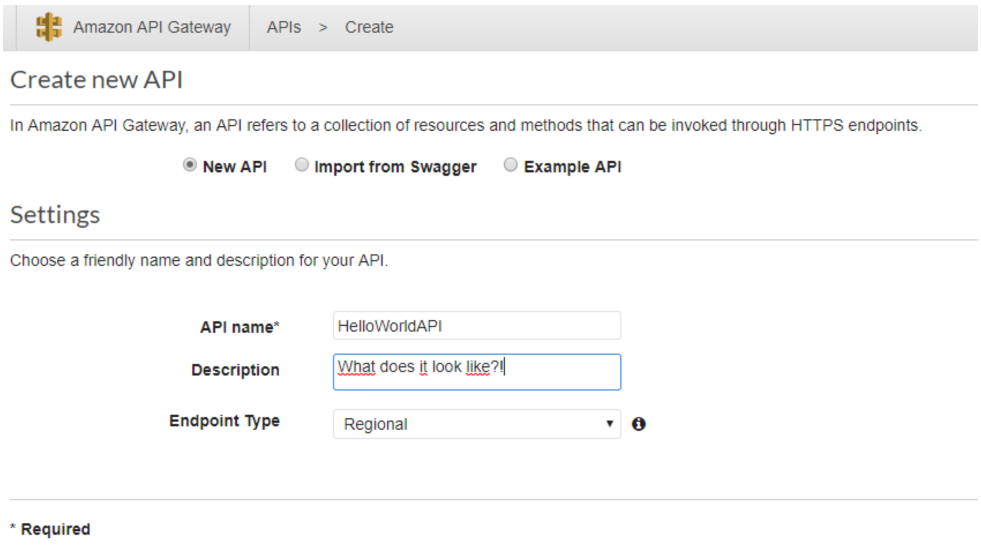
Time (UTC +00:00)	Message	Show in stream
2018-07-04		
	No older events found for the selected date range. Adjust the date range.	
19:12:24	Loading...	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	Done loading!	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	START RequestId: 2f13072f-7fbe-11e8-82cf-df068de8b3bb Version: $LATEST	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	Got an event: {	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	"name": "SynTouch Colleagues",	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	"array": [	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	"This",	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	"is",	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	"a",	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	"structured",	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	"object"	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	],	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	"error": false	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	}	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	Execution context - 14999 ms remaining	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	Execution context: function HelloWorld version $LATEST, size 128 mb	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	Logging to log group /aws/lambda/HelloWorld and log stream 2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849da2228bdcc4	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	We gaan groeten ... Hello SynTouch Colleagues!	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	END RequestId: 2f13072f-7fbe-11e8-82cf-df068de8b3bb	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
19:12:24	REPORT RequestId: 2f13072f-7fbe-11e8-82cf-df068de8b3bb Duration: 0.64 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 22 MB	2018/07/04/[$LATEST]d9c71e52eb4c4b4dbf849...
	No newer events found for the selected date range. Adjust the date range.	

Expose HelloWorld as a WebAPI

Now that we have defined the mandatory HelloWorld as a serverless function in AWS Lambda, let's go the extra mile and expose this to the GBI ("Grote Boze Internet") as an WebAPI to be called over HTTP.

Make sure that you're still in the eu-west-1 (Ireland) region, as this is where your Lambda function lives.

Define a new API in API Gateway:



Amazon API Gateway APIs > Create

Create new API

In Amazon API Gateway, an API refers to a collection of resources and methods that can be invoked through HTTPS endpoints.

[✓ New API] Import from Swagger Example API

Settings

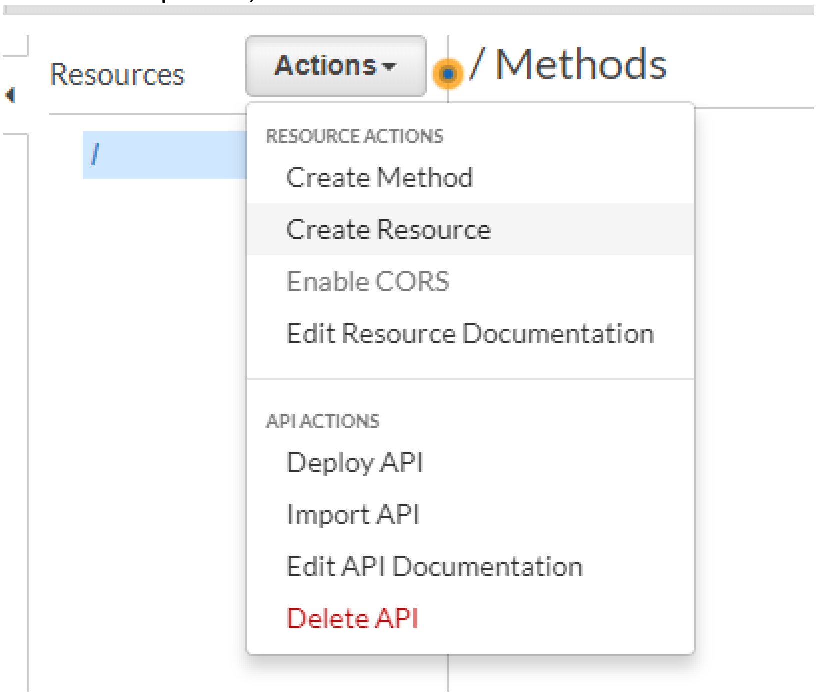
Choose a friendly name and description for your API.

API name*	HelloWorldAPI
Description	What does it look like?!
Endpoint Type	Regional

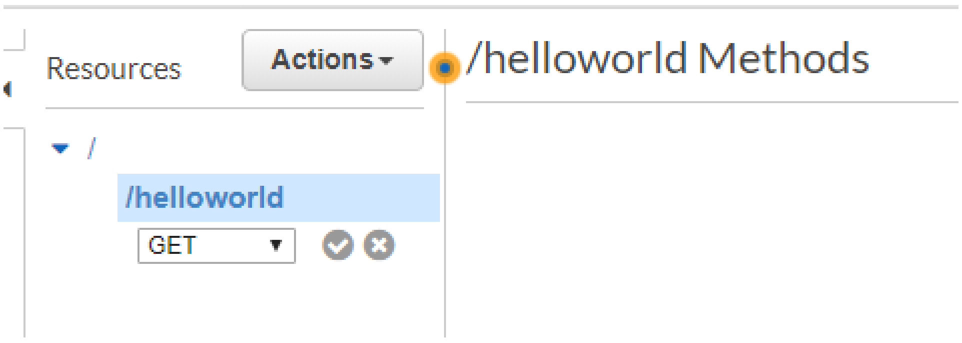
* Required

We'll define the API as a GET on resource helloworld where the name is to be provided as a query parameter …

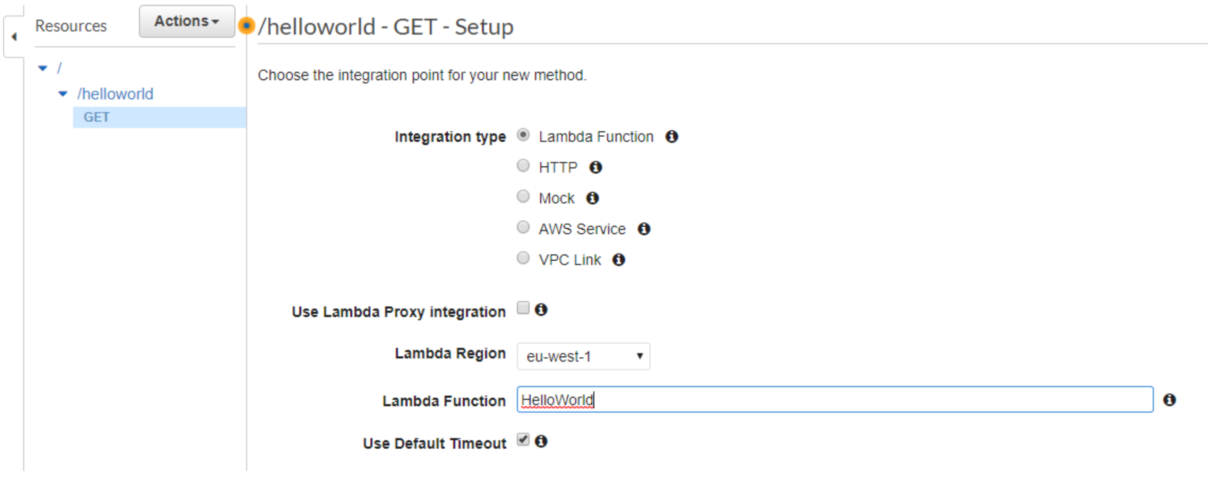
From the drop down, choose "Create Resource":



Select the /helloworld resource and choose "Create Method" from the dropdown list; select the GET HTTP method:



If you press the “Okay” icon, you’re presented with a configuration form to connect your exposed endpoint. Of course we’ll be connecting this to the HelloWorld lambda function:



/helloworld - GET - Setup

Choose the integration point for your new method.

Integration type: [✓ Lambda Function] HTTP Mock AWS Service VPC Link

Use Lambda Proxy integration: ☐

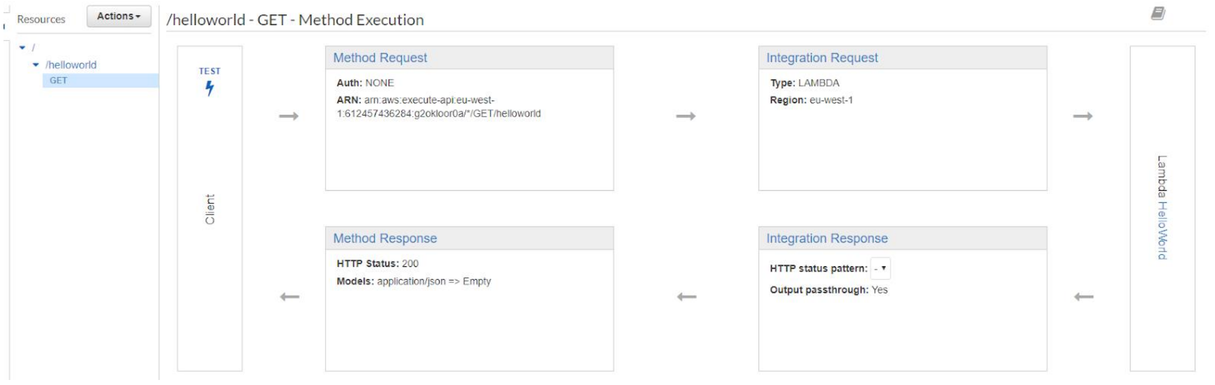
Lambda Region: eu-west-1

Lambda Function: HelloWorld

Use Default Timeout: ✓

When asked, authorize API Gateway to invoke your Lambda function.

The API Gateway represents the integration with the backend as a number of phases:



/helloworld - GET - Method Execution

Method Request
Auth: NONE
ARN: arn:aws:execute-api:eu-west-1:612457436284:g2okloor0a/*/GET/helloworld

Integration Request
Type: LAMBDA
Region: eu-west-1

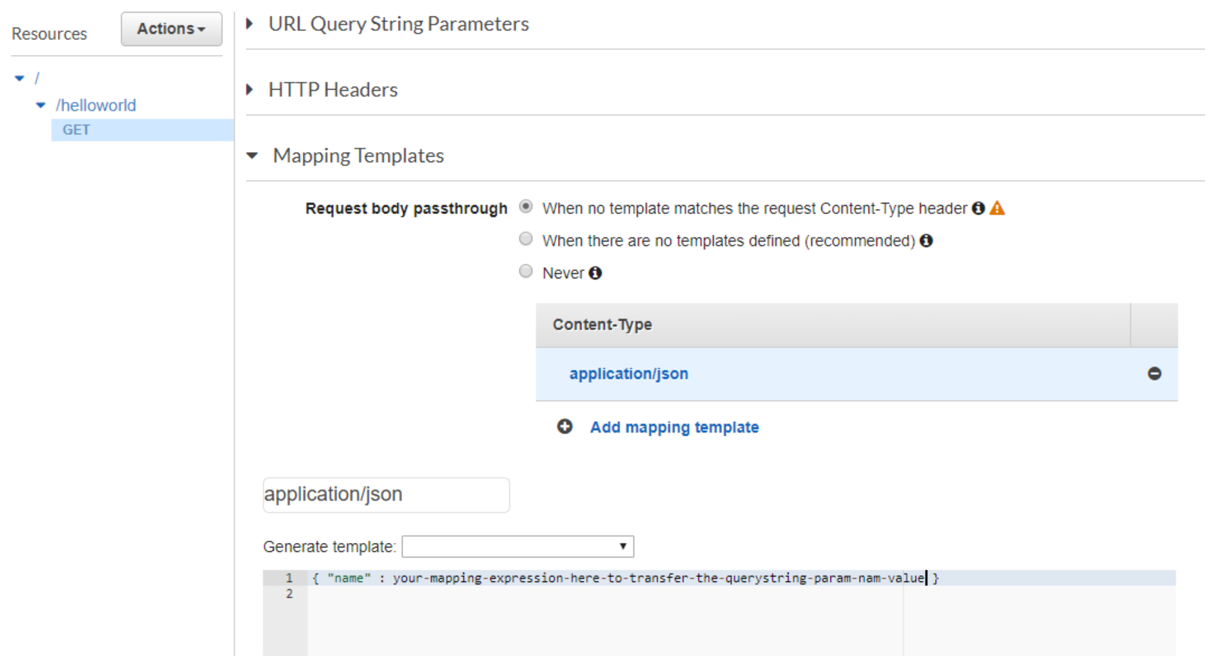
Method Response
HTTP Status: 200
Models: application/json => Empty

Integration Response
HTTP status pattern: -
Output passthrough: Yes

In the first stage (Method Request), the expected payload structure, headers and query string parameters can be defined. The request can be rejected if it does not satisfy the defined requirement. For our current purpose, this is not relevant.
The next step, Integration Request, is used to transform the received request to the request the backend (in this case our HelloWorld lambda function) will receive.

Here, we need to find a way to transfer the contents of the name querystring parameter to a name element in the JSON payload for the backend service. API Gateway supports the Apache Velocity templating engine for performing these tasks. For an overview of the available mapping functions, see https://docs.aws.amazon.com/apigateway/latest/developerguide/api-gateway-mapping-template-reference.html

Define a new template to be used for application/json content types; insert your expression replacing your-mapping-expression-here-to-transfer-the-querystring-param-nam-value:



Request body passthrough [✓ When no template matches the request Content-Type header] When there are no templates defined (recommended) Never

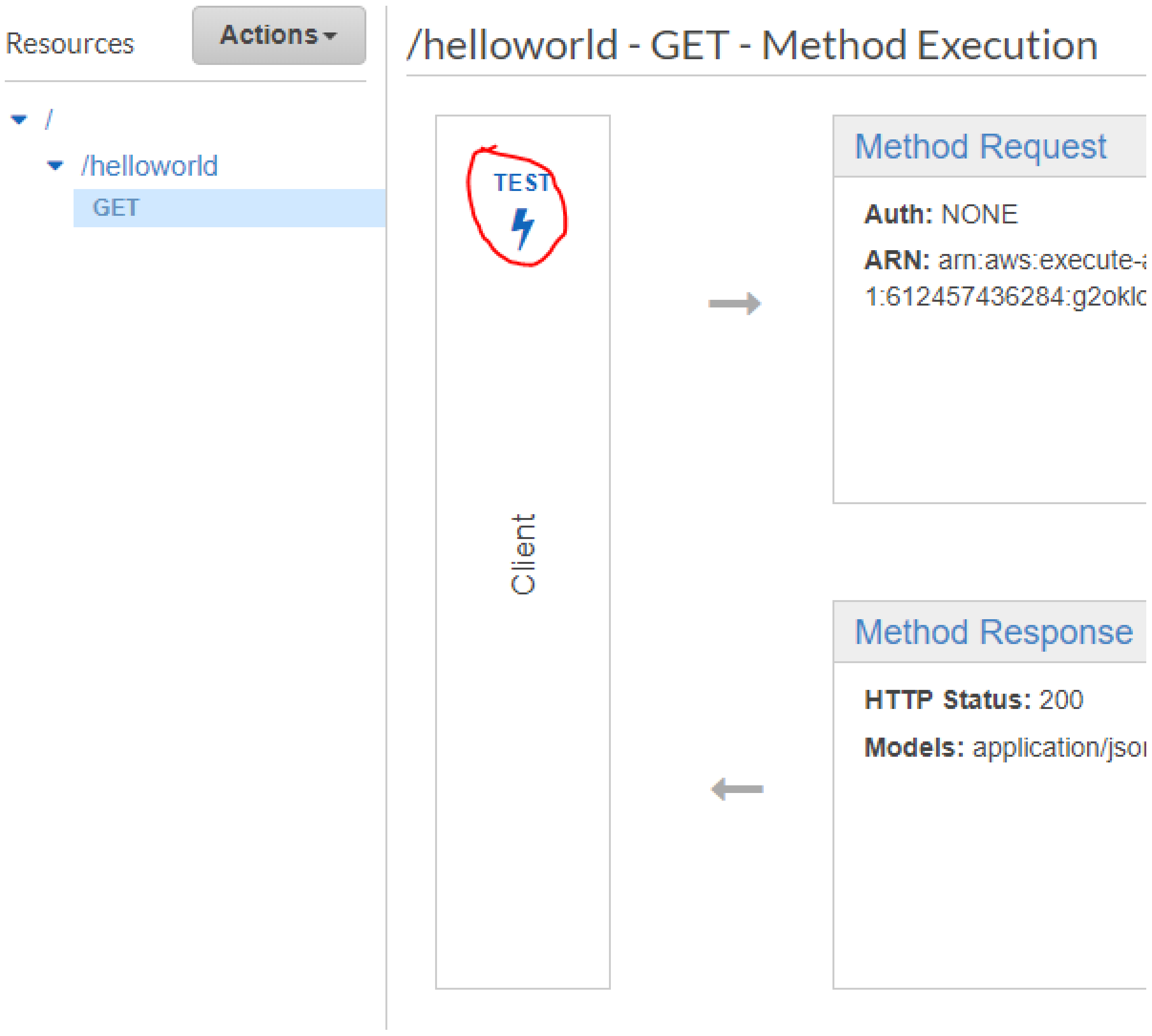
Content-Type
application/json

application/json

Generate template:

```
{ "name" : your-mapping-expression-here-to-transfer-the-querystring-param-nam-value }
```

Navigate back to your API, select the GET method on helloworld and invoke the TEST button on the client:



Resources

Actions

/

/helloworld

GET

/helloworld - GET - Method Execution

TEST

Client

Method Request

Auth: NONE

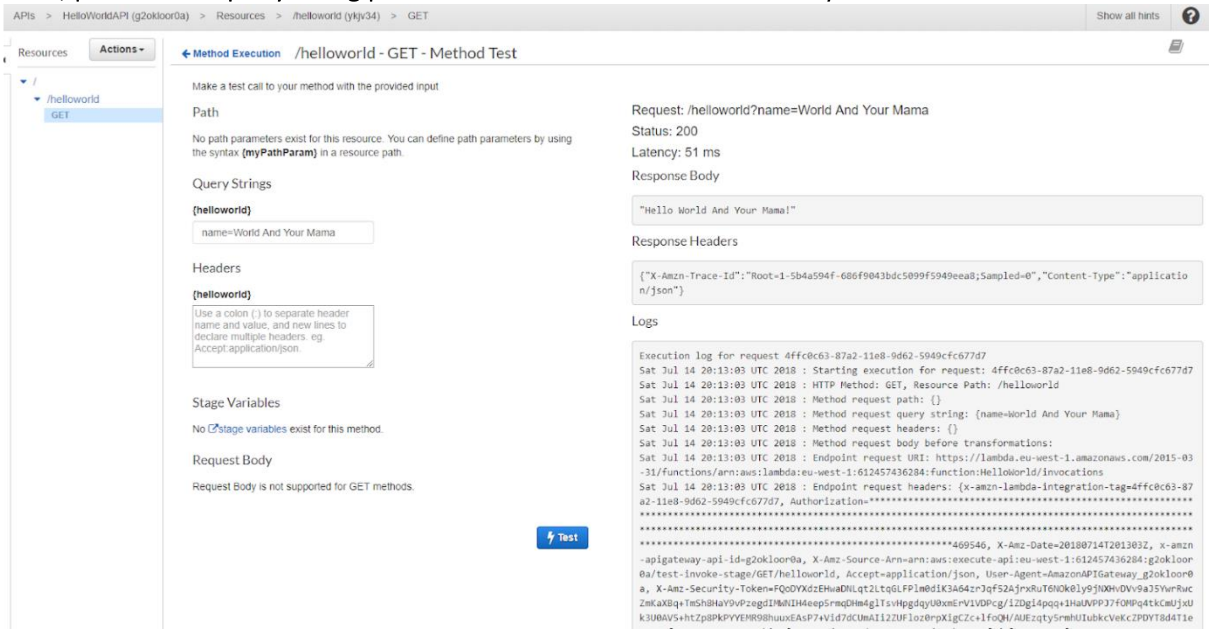
ARN: arn:aws:execute-a
1:612457436284:g2oklo

Method Response

HTTP Status: 200

Models: application/jsoi

Next, provide the query string parameter name and set to a value of your choice. TEST!



APIs > HelloWorldAPI (g2okloor0a) > Resources > /helloworld (ykjv34) > GET

Show all hints

Resources

Actions

/

/helloworld

GET

Method Execution /helloworld - GET - Method Test

Make a test call to your method with the provided input

Path

No path parameters exist for this resource. You can define path parameters by using the syntax **{myPathParam}** in a resource path.

Query Strings

{helloworld}

name=World And Your Mama

Headers

{helloworld}

Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg. Accept:application/json.

Stage Variables

No stage variables exist for this method.

Request Body

Request Body is not supported for GET methods.

Test

Request: /helloworld?name=World And Your Mama

Status: 200

Latency: 51 ms

Response Body

```
"Hello World And Your Mama!"
```

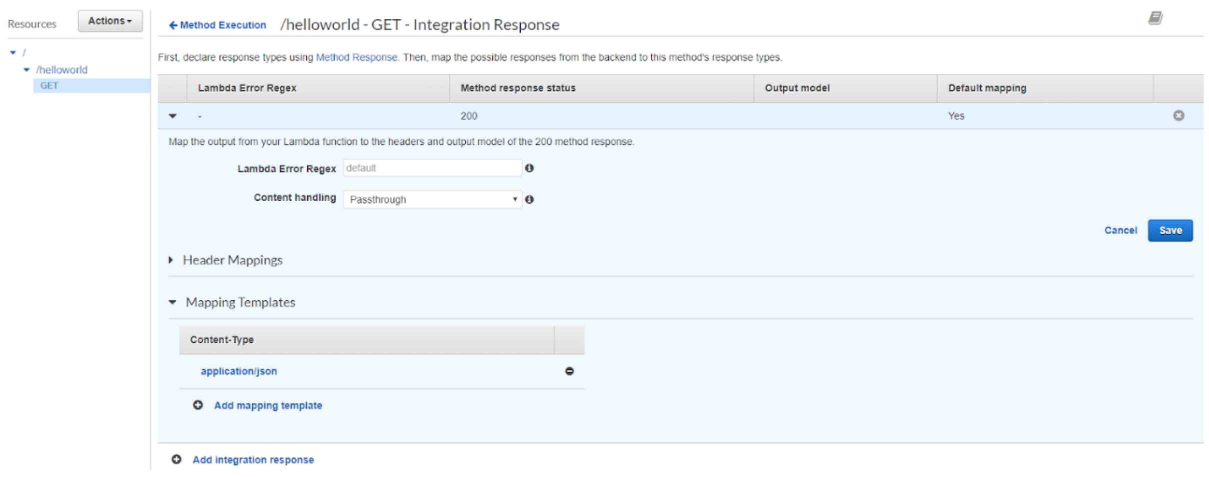
Response Headers

```
{"X-Amzn-Trace-Id":"Root=1-5b4a594f-686f9043bdc5099f5949eea8;Sampled=0","Content-Type":"applicatio
n/json"}
```

Logs

```
Execution log for request 4ffc0c63-87a2-11e8-9d62-5949cfc677d7
Sat Jul 14 20:13:03 UTC 2018 : Starting execution for request: 4ffc0c63-87a2-11e8-9d62-5949cfc677d7
Sat Jul 14 20:13:03 UTC 2018 : HTTP Method: GET, Resource Path: /helloworld
Sat Jul 14 20:13:03 UTC 2018 : Method request path: {}
Sat Jul 14 20:13:03 UTC 2018 : Method request query string: {name=World And Your Mama}
Sat Jul 14 20:13:03 UTC 2018 : Method request headers: {}
Sat Jul 14 20:13:03 UTC 2018 : Method request body before transformations:
Sat Jul 14 20:13:03 UTC 2018 : Endpoint request URI: https://lambda.eu-west-1.amazonaws.com/2015-03
-31/functions/arn:aws:lambda:eu-west-1:612457436284:function:HelloWorld/invocations
Sat Jul 14 20:13:03 UTC 2018 : Endpoint request headers: {x-amzn-lambda-integration-tag=4ffc0c63-87
a2-11e8-9d62-5949cfc677d7, Authorization=*************************************************************
****************************************************************************************************
****************************************************************************************************
**********************************************************469546, X-Amz-Date=20180714T201303Z, x-amzn
-apigateway-api-id=g2okloor0a, X-Amz-Source-Arn=arn:aws:execute-api:eu-west-1:612457436284:g2okloor
0a/test-invoke-stage/GET/helloworld, Accept=application/json, User-Agent=AmazonAPIGateway_g2okloor0
a, X-Amz-Security-Token=FQoDYXdzEHwaDNLqt2LtqGLFPlm0diK3A64zrJqf52AjrxRuT6NOk0ly9jNXHvDVv9aJ5YwrRwc
ZmKaXBq+TmSh8HaY9vPzegdIMWNIH4eep5rmqDHm4glTsvHpgdqyU0xmErV1VDPcg/iZDgi4pqq+1HaUVPPJ7fOMPq4tkCmUjxU
k3U0AV5+htZp8PkPYYEMR98huuxEAsP7+Vid7dCUmAIi2ZUFloz0rpXigCZc+lfoQH/AUEzqty5rmhUIubkcVeKcZPDYT8d4T1e
```

As you can see from the logs (and the response), the query string parameter is mapped to the name element in the payload of the event. However, the response is still returned as plain old text - since the function simply returns text. This can be easily solved using the Integration Response mapping template, in a similar fashion we have applied to the request:



First, declare response types using Method Response. Then, map the possible responses from the backend to this method's response types.

Lambda Error Regex	Method response status	Output model	Default mapping
-	200		Yes

Map the output from your Lambda function to the headers and output model of the 200 method response.

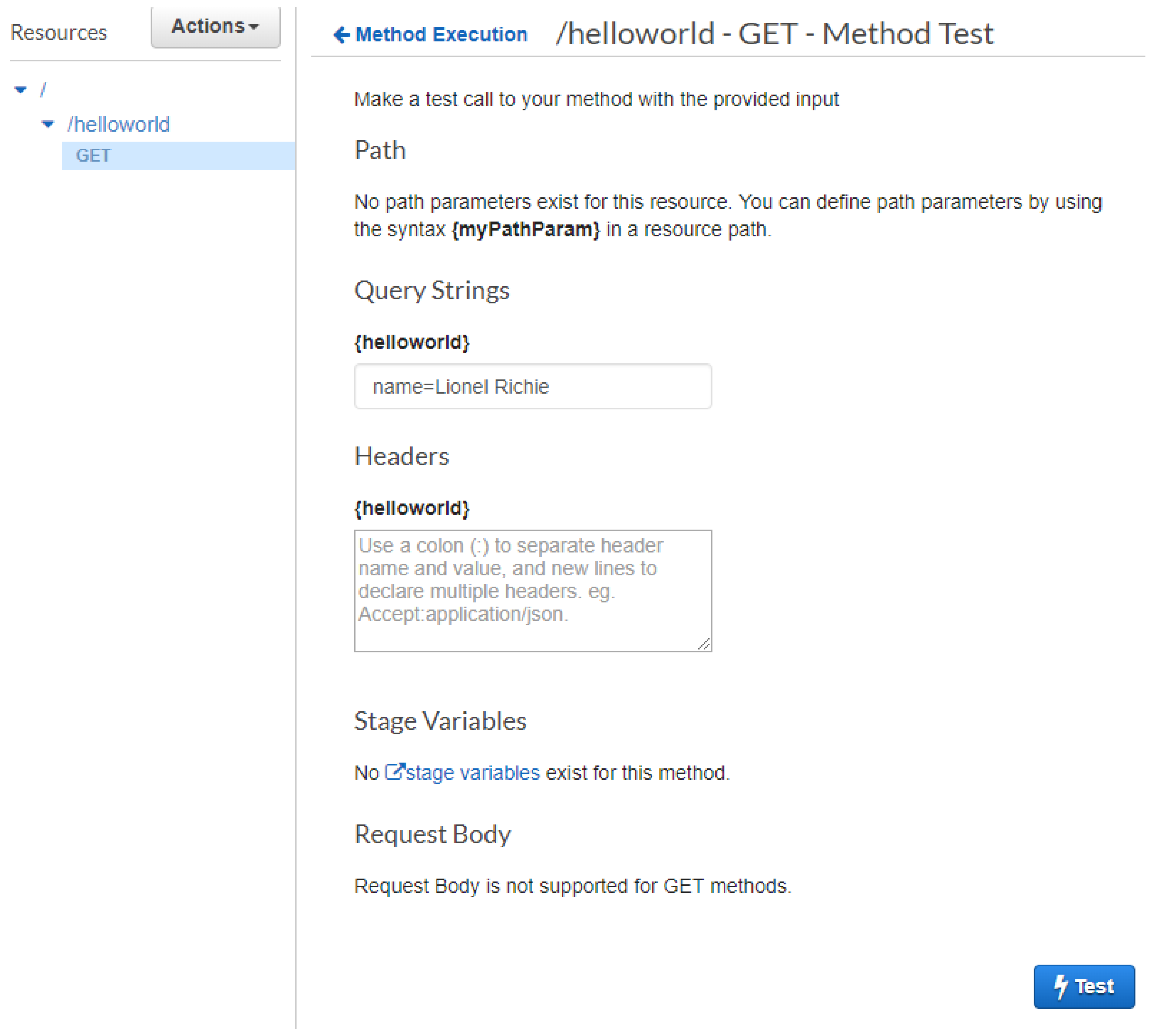
Lambda Error Regex: default

Content handling: Passthrough

Add a mapping template to take the entire response body and assign this to the JSON message element; the resulting response should be like:

{ “message” : “*The-actual-response-from-your-lambda-should-go-here…*”}

Test the GET method again:



/helloworld - GET - Method Test

Make a test call to your method with the provided input

Path

No path parameters exist for this resource. You can define path parameters by using the syntax **{myPathParam}** in a resource path.

Query Strings

{helloworld}

name=Lionel Richie

Headers

{helloworld}

Use a colon (:) to separate header name and value, and new lines to declare multiple headers. eg. Accept:application/json.

Stage Variables

No stage variables exist for this method.

Request Body

Request Body is not supported for GET methods.

Now is properly responding JSON !

```
{
  "message": "Hello Lionel Richie!"
}
```

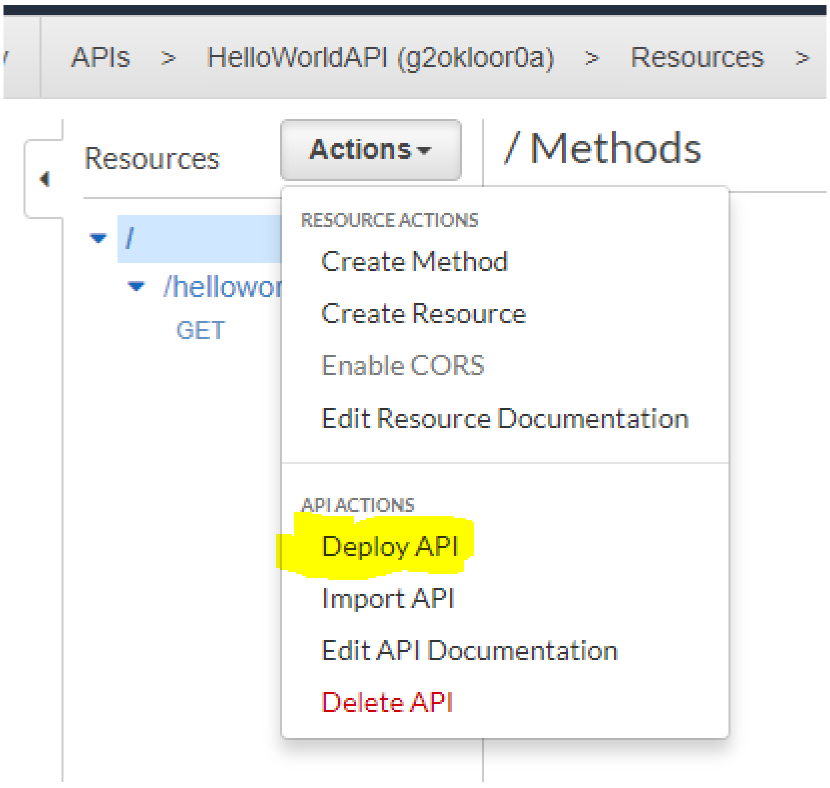
Response Headers

```
{"X-Amzn-Trace-Id":"Root=1-5b4a5ba8-7adcdff64f418131abe48e20;Sampled=0","Content-Type":"applicatio
n/json"}
```

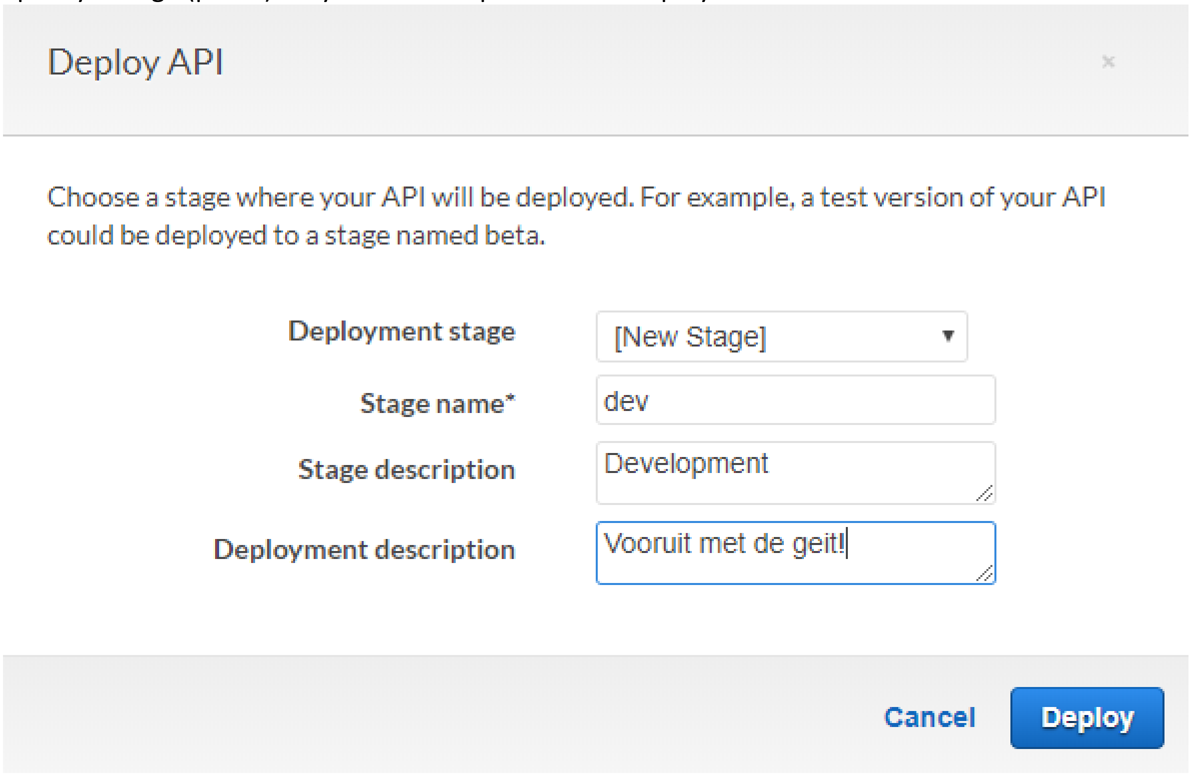
Check the Logs for the steps executed upon receiving the request!

Making the API accessible

The final step in the process is actually deploying the API to the world:



Specify a stage (prefix) for your API and perform the deploy:

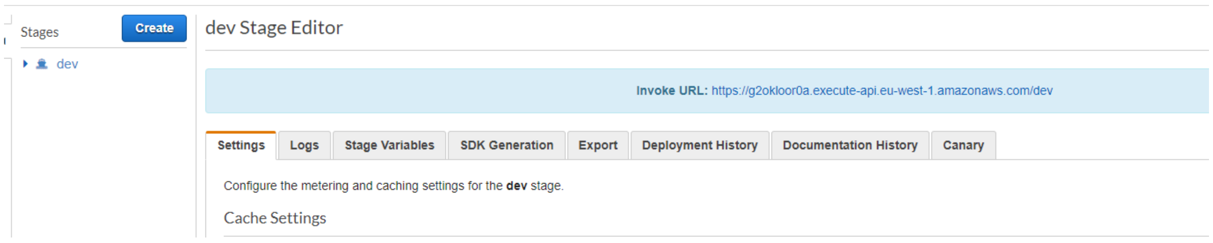


Deploy API

Choose a stage where your API will be deployed. For example, a test version of your API could be deployed to a stage named beta.

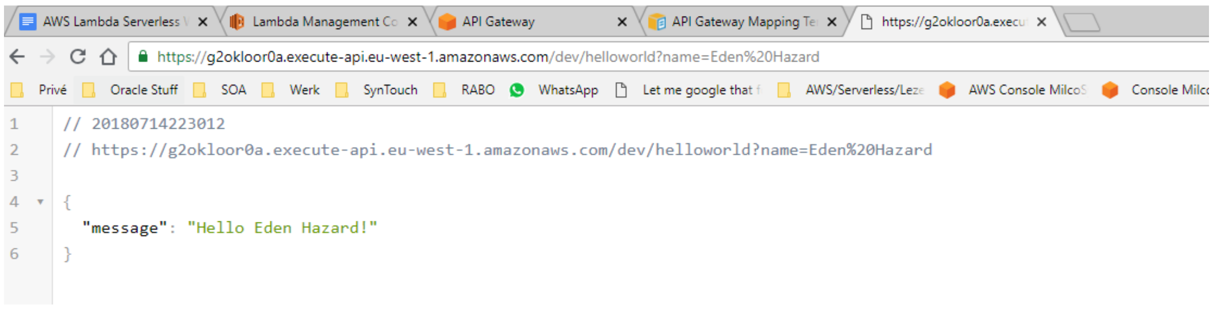
Deployment stage	[New Stage]
Stage name*	dev
Stage description	Development
Deployment description	Vooruit met de geit!

Examine the confirmation:


Stages
Create
dev
dev Stage Editor
Invoke URL: https://g2okloor0a.execute-api.eu-west-1.amazonaws.com/dev

Configure the metering and caching settings for the **dev** stage.
Cache Settings

As you can see, here the stage “dev” is exposed at “https://g2okloor0a.execute-api.eu-west-1.amazonaws.com/dev”. The resource for our API is helloworld and the GET request requires a name parameter, so you can test the api by going to https://g2okloor0a.execute-api.eu-west-1.amazonaws.com/dev/helloworld?name=Eden%20Hazard



https://g2okloor0a.execute-api.eu-west-1.amazonaws.com/dev/helloworld?name=Eden%20Hazard

```
// 20180714223012
// https://g2okloor0a.execute-api.eu-west-1.amazonaws.com/dev/helloworld?name=Eden%20Hazard

{
  "message": "Hello Eden Hazard!"
}
```

Well done!