Published in Better Programming

Valerio Uberti Follow
Apr 15 · 4 min read · Listen

Create a Simple AWS Lambda Using TypeScript

Use TypeScript for my Lambda functions

Hi everyone, today I would like to talk about how to write a simple AWS Lambda

Using Typescript for your lambda will have some benefits such as:

1. Code completion hint while programming with your IDE
2. Compilation time errors checks
3. Knowledge about the structure of your input and ouputs

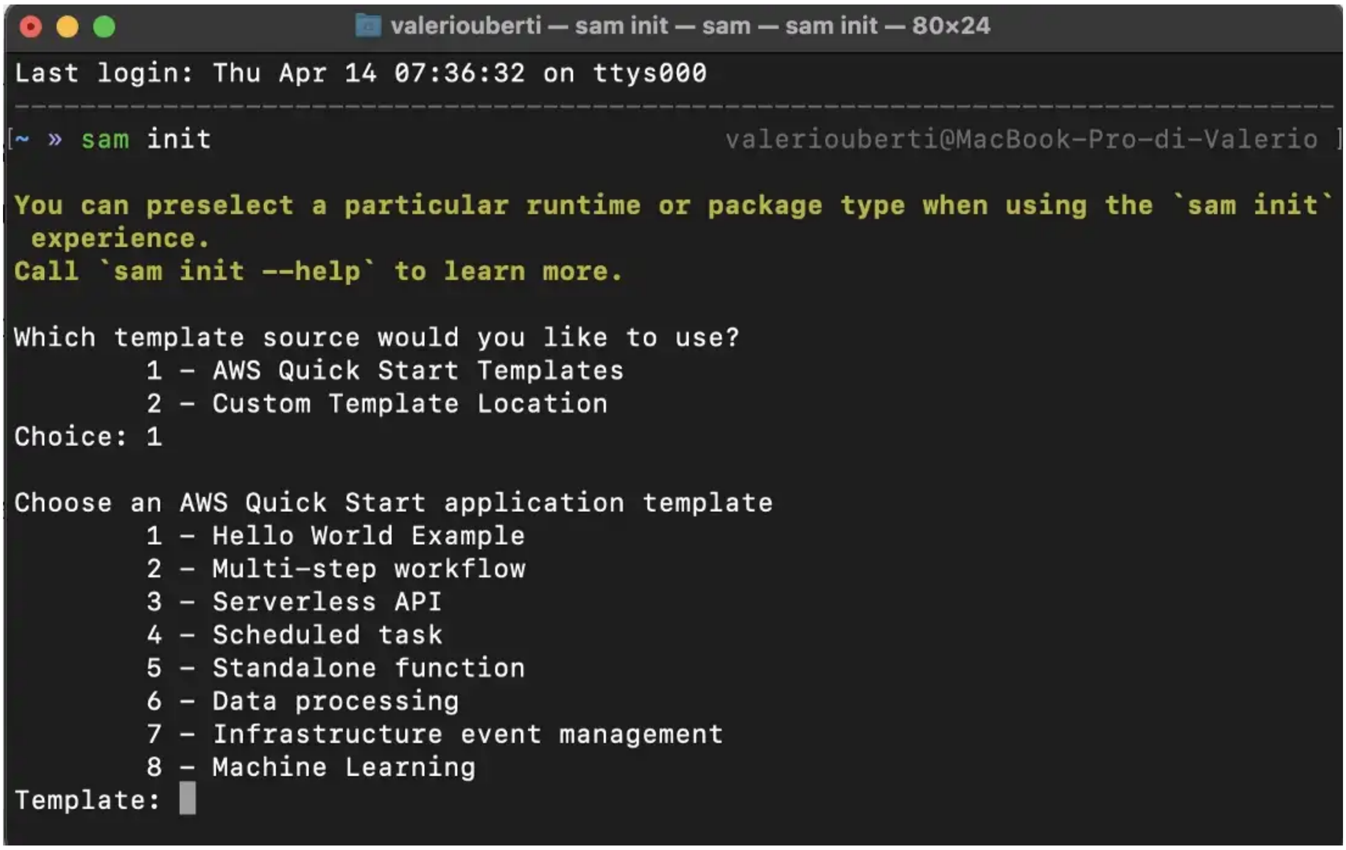
Okay, enough talk. Let's jump to the example!

Preparation

Before you begin creating a sample lambda project using SAM CLI tool:

```
sam init
```

In your terminal you should see something like this:



```
Last login: Thu Apr 14 07:36:32 on ttys000
--------------------------------------------------------------------------------
[~ » sam init                                   valeriouberti@MacBook-Pro-di-Valerio ]

You can preselect a particular runtime or package type when using the `sam init`
 experience.
Call `sam init --help` to learn more.

Which template source would you like to use?
        1 - AWS Quick Start Templates
        2 - Custom Template Location
Choice: 1

Choose an AWS Quick Start application template
        1 - Hello World Example
        2 - Multi-step workflow
        3 - Serverless API
        4 - Scheduled task
        5 - Standalone function
        6 - Data processing
        7 - Infrastructure event management
        8 - Machine Learning
Template: 
```

```
├── README.md
├── events
│   └── event.json
├── hello-world
│   ├── app.js
│   ├── package.json
│   └── tests
│       └── unit
│           └── test-handler.js
└── template.yaml
```

First of all, begin with renaming `hello-word` folder to `src` and delete `tests` folder (we will not use it for this tutorial). When you finish, go inside your terminal and start to add some dependencies.

Start with AWS dependency using the command:

```
npm install aws-sdk
```

After that it’s time to typescript dependencies:

```
npm install --save-dev typescript @types/aws-lambda @types/node
```

1. `typescript`
2. `@type/aws-lambda` , which contains AWS types for the code completion and typing checks
3. `@type/node` , which is used for built-in types

Once you have installed all dependencies you can run:

```
tsc --init
```

```
{
  "compilerOptions": {
    "module": "CommonJS",
    "target": "ES2017",
    "noImplicitAny": true,
    "preserveConstEnums": true,
    "outDir": "./built",
    "sourceMap": true
  }
}
```

TypeScript compiler needs `tsconfig.json` to figure out how to transform TypeScript to JavaScript.

Change the code

Ok, now that the configuration is finished and let's change the code.
Rename `app.js` to `app.ts` and delete the code inside.

Then, write the Lamba handler like this:

```
import {
  APIGatewayProxyEvent,
  APIGatewayProxyResult }
from "aws-lambda/trigger/api-gateway-proxy";

export const lambdaHandler = async (
   event: APIGatewayProxyEvent
): Promise<APIGatewayProxyResult> => {

  const queries = JSON.stringify(event.queryStringParameters);

  return {
    statusCode: 200,
    body: `Queries: ${queries}`
  }
}
```

This is a simple handler that reads the query string parameters of my function and displays them for output. Cool right?

```
  surname: string
  age: number
}
```

The purpose of the function is to pass an input which is a `Person` object and display the information contained as an output.

Let’s code:

```
import {
  APIGatewayProxyEvent,
  APIGatewayProxyResult }
from "aws-lambda/trigger/api-gateway-proxy";

interface Person {
  name: string
  surname: string
  age: number
}

export const lambdaHandler = async (
   event: APIGatewayProxyEvent
): Promise<APIGatewayProxyResult> => {

  const person: Person = JSON.parse(event.body);

  return {
    statusCode: 200,
    body: JSON.stringify({'Person' : person})
  }
}
```

Deploy and test

Now it’s time to test the Lambda function. First of all, you have to compile the Lambda and check if everything works.

Go to `package.json` file and add this step into the script section:

```
.
```

```
.
.
```

and execute typing:

```
npm run compile
```

If everything works, the Lambda function is compiled. Let’s test! Go to `template.yml` and adjust it to suit your app. Mine is like this:

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: AWS::Serverless-2016-10-31
Description: Sample Lambda Function in Typescript
Globals:
  Function:
  Timeout: 3
Resources:
  TypescriptFunction:
    Type: AWS::Serverless::Function
    Properties:
     CodeUri: src/built
     Handler: app.lambdaHandler
     Runtime: nodejs14.x
     Events:
      Typescript:
       Type: Api
        Properties:
         Path: /hello
         Method: get
.
.
.
```

when you finish run:

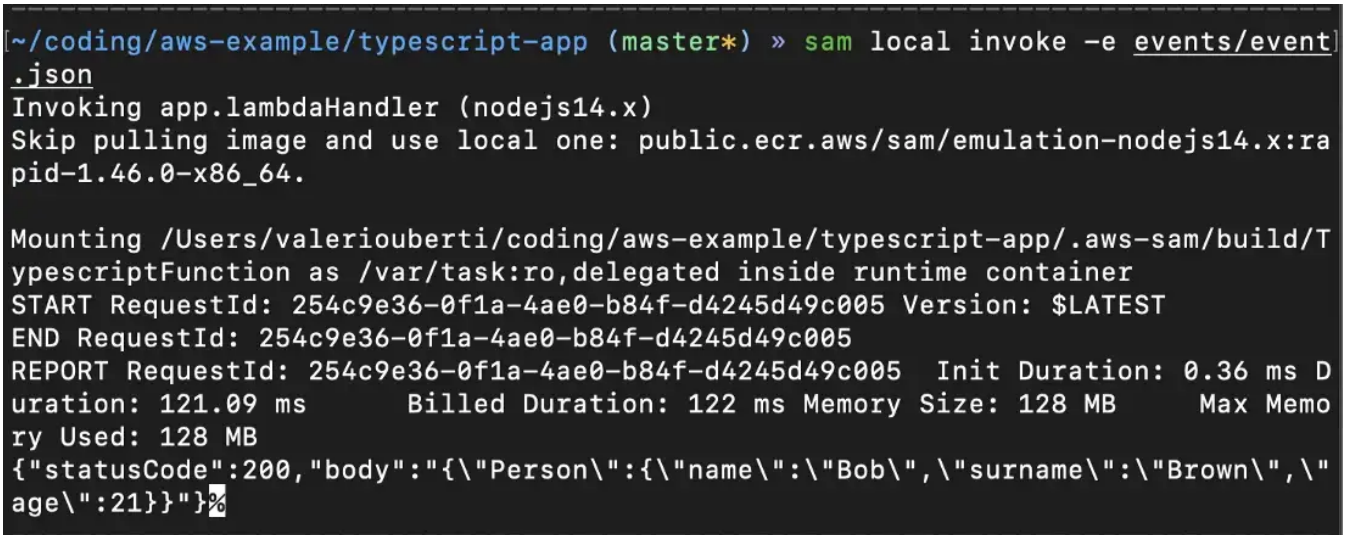
```
sam build
```

```
"body": "{\"name\": \"Bob\", \"surname\":\"Brown\",\"age\":21}",
.
.
.
```

Run your Lambda function in your local machine:

```
sam local invoke -e events/event.json
```

If everything works you should see in your terminal output like this:



```
~/coding/aws-example/typescript-app (master*) » sam local invoke -e events/event
.json
Invoking app.lambdaHandler (nodejs14.x)
Skip pulling image and use local one: public.ecr.aws/sam/emulation-nodejs14.x:ra
pid-1.46.0-x86_64.

Mounting /Users/valeriouberti/coding/aws-example/typescript-app/.aws-sam/build/T
ypescriptFunction as /var/task:ro,delegated inside runtime container
START RequestId: 254c9e36-0f1a-4ae0-b84f-d4245d49c005 Version: $LATEST
END RequestId: 254c9e36-0f1a-4ae0-b84f-d4245d49c005
REPORT RequestId: 254c9e36-0f1a-4ae0-b84f-d4245d49c005  Init Duration: 0.36 ms D
uration: 121.09 ms      Billed Duration: 122 ms Memory Size: 128 MB     Max Memo
ry Used: 128 MB
{"statusCode":200,"body":"{\"Person\":{\"name\":\"Bob\",\"surname\":\"Brown\",\"
age\":21}}"}%
```

Your lambda is now ready to be deployed to AWS.

Conclusion

The sample code can be found at:

GitHub - valeriouberti/aws-lambda-typescript: A simple example of an AWS Lambda Function in...

A simple example of an AWS Lambda Function in Typescript - GitHub - valeriouberti/aws-lambda-typescript: A simple...

github.com

And the compiler helps me to write the correct types and avoid potentials problems before deploying.