**Note: For now, you can ignore the README.md file automatically generated with SAM CLI and the samconfig.toml file since those topics will be explained in a dedicated guide)**

Explanation //

**Purpose:**

We will deploy a simple API Gateway API and one single lambda function. The API gateway resource will use a GET method and it will be the lambda backend will be implemented as lambda proxy-integration, the lambda function will also use request validators to check if two required query string parameters exists within the request.

Note that the ResquestParameters property will just define which parameters are required but it will not define a Request validator and you will have to enable it manually, alternatively you can define a request validator if using Explicit API resource definition.

The function will just print/return a simple hello world in different languages depending on the path, query parameters and body request.

Remember also that the API will be created IMPLICITLY, that means that since we are not explicitly defining an AWS::Serverless::Api resource SAM framework knows that it need to create one so the function of type AWS::Serverless::Function can exist, however is important to keep in mind that implicit resource creation although faster will limit our ability to use advance features or configurations.

**General considerations:**

Examples on how to call the endpoint/resource is within the postman collection in which you can see the URL, path/query parameters and any relevant information.

We will be using async/await syntax and the moment library just to get familiar with the use of external packages within the Serverless lambda architecture.

Note that you **can** send a body within a GET request in local testing, but it is not recommended and NOT possible in some production scenarios, use POST instead since some proxies, services or firewalls could not work with a body included withing a GET request, and this is the case when using API gateway.

You can find more about SAM here:

<https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-getting-started.html>

**Instructions:**

**Note: This first two steps are a brief overview of what I did to create the SAM template/project you don't need to do it and in the guide folder there is a file in which I explain in detail the features of SAM and CFN.**

**1-run:** sam init

(Remember that you need to have SAM CLI installed on your local environment)

<https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/install-sam-cli.html>

SAM needs some prerequisites to run, most of them are optional since they are to use some local testing features of SAM within docker containers on your local environment, but I recommend to follow the complete installation in the link above.

**2-Answer the prompts with the desired configuration, in my case it was:**

-AWS Quick Start Templates

-Hello world Example

-N

-13 (node 16.x)

-1 (zip)

-1 (Hello world Example)

-N

-y

-LambdaBasicsSAM

**The following steps are so you can deploy the SAM template and test it locally (local testing only available if you followed the SAM CLI prerequisites and installed docker).**

**1- To test the function by itself locally sit on the root directory of the SAM project and run:**

**sam local invoke "multiHellorWorld" -e .\events\multiHelloWorldTestEvent.json**

This will invoke the logical name of the function that is defined in the template.yaml file and will use the **-e** flag to specify a test event which you can find inside the events folder (remember that you need to allow docker to share the volumes of your pc, generally you get prompted automatically, otherwise I will include a link in the references/important-links.txt file)

**2-With the use we have specified on our template.yaml file an Api gateway resource will be created implicitly**

<https://github.com/awslabs/serverless-application-model/blob/master/docs/internals/generated_resources.rst#api>

To test the function and the api locally run the following command.

**sam local start-api --debug** or **sam local start-api**

**Note: You can create resources implicitly not implicitly and also, implicitly and not implicitly at the same time, I will detail these cases in the main demo of the repo.**

After you run the command, It will give you an Ip with a port, use postman to test the path which is used to call our function:

Example (this path is specified in the template.yaml file within the definition of our resources):

**Running on http://127.0.0.1:3000/helloMulti/{name}**

**Note: Within the repo there is a folder called postman which will contain an exported JSON of the collections used for each demo, you can import it into your postman if you want, remember to create an environment for the collection and create the variable host within it with the value that SAM gave you in the previous step (http://127.0.0.1:3000/helloMulti/{name})**

**3-sam package --template-file template.yaml --output-template-file sam.yaml --s3-bucket <bucket-name>**

**Note: the --s3-bucket flag should only be used if you want to specify a bucket in line otherwise it will pick the one in the samconfig.toml file, if neither the inline command nor the toml file define a bucket then a default bucket will be created automatically. (This bucket is used to store some necessary files for CFN to work)**

This command will transform your template.yaml file to another SAM yaml file which will have some variables resolved like the implicit resources or the URI of the function code that will no longer be on our local computer but an s3 bucket which you must create before running this command (I will explain all the details about SAM in the SAM guide)

**4-To deploy the stack run the following command:**

**sam deploy --template-file <the file created on step 3> --stack-name <YOUR STACK NAME>**

**5-Once you run the command**

Go and check CloudFormation console to see the status of the stack you are creating, alternatively you can see the outputs in the terminal. (Remember that CFN as well as the SAM frameworks offers useful tools for debugging)

**6-Once everything gets created** **successfully**

Go and check that a new Api was created and test it. I will not cover how to test it since the documents within the guide folder should give you enough understanding on how to do it.

**7-After testing**

Delete the stack from CFN and the bucket so you keep your environment in a clean status (This is optional since most of the demos will be built on top of each other and redeploying a Sam file will just update the resources from a stack if the stack name is the same)

**Additionally:** Keep in mind that this was a brief demo, and the purpose was not to go full detail into CFN or SAM and instead you should be focusing only on the behavior of the lambda and the Api Gateway services deployed through this demo.