## **Complete Following at AWS Cloud9 IDE to understand how AWS SAM works**

## **Step 1: Download a Sample AWS SAM Application**

Command to run:

sam init

Follow the on-screen prompts. For this tutorial we recommend you choose AWS Quick Start Templates, the runtime of your choice, and the Hello World Example.

Example output:

-----------------------

Generating application:

-----------------------

Name: sam-app

Runtime: python3.7

Dependency Manager: pip

Application Template: hello-world

Output Directory: .

Next steps can be found in the README file at ./sam-app/README.md

**What AWS SAM is doing:**

This command creates a directory with the name you provided as the project name. The contents of the project directory are similar to the following (these contents are created when one of the Python runtimes and the Hello World Example are chose):

sam-app/

├── README.md

├── events/

│ └── event.json

├── hello\_world/

│ ├── \_\_init\_\_.py

│ ├── app.py #Contains your AWS Lambda handler logic.

│ └── requirements.txt #Contains any Python dependencies the application requires, used for sam build

├── template.yaml #Contains the AWS SAM template defining your application's AWS resources.

└── tests/

└── unit/

├── \_\_init\_\_.py

└── test\_handler.py

There are three especially important files:

* template.yaml: Contains the AWS SAM template that defines your application's AWS resources.
* hello\_world/app.py: Contains your actual Lambda handler logic.
* hello\_world/requirements.txt: Contains any Python dependencies that the application requires, and is used for sam build.

## **Step 2: Build Your Application**

Command to run:

First change into the project directory (that is, the directory where the template.yaml file for the sample application is located; by default is sam-app), then run this command:

sam build

Example output:

Build Succeeded

Built Artifacts : .aws-sam/build

Built Template : .aws-sam/build/template.yaml

Commands you can use next

=========================

[\*] Invoke Function: sam local invoke

[\*] Deploy: sam deploy --guided

**What AWS SAM is doing:**

The AWS SAM CLI comes with abstractions for a number of Lambda runtimes to build your dependencies, and copies the source code into staging folders so that everything is ready to be packaged and deployed. The sam build command builds any dependencies that your application has, and copies your application source code to folders under .aws-sam/build to be zipped and uploaded to Lambda.

You can see the following top-level tree under .aws-sam:

.aws\_sam/

└── build/

├── HelloWorldFunction/

└── template.yaml

HelloWorldFunction is a directory that contains your app.py file, as well as third-party dependencies that your application uses.

## **Step 3: Deploy Your Application to the AWS Cloud**

Command to run:

sam deploy --guided

Follow the on-screen prompts. You can just respond with Enter to accept the default options provided in the interactive experience.

Example output:

Deploying with following values

===============================

Stack name : sam-app

Region : us-east-1

Confirm changeset : False

Deployment s3 bucket : sam-bucket

Capabilities : ["CAPABILITY\_IAM"]

Parameter overrides : {}

Initiating deployment

=====================

Waiting for changeset to be created..

CloudFormation stack changeset

---------------------------------------------------------------------------------------------------------------------------------------------------

Operation LogicalResourceId ResourceType

---------------------------------------------------------------------------------------------------------------------------------------------------

+ Add HelloWorldFunctionHelloWorldPermissionProd AWS::Lambda::Permission

+ Add ServerlessRestApiDeployment47fc2d5f9d AWS::ApiGateway::Deployment

+ Add ServerlessRestApiProdStage AWS::ApiGateway::Stage

+ Add ServerlessRestApi AWS::ApiGateway::RestApi

\* Modify HelloWorldFunctionRole AWS::IAM::Role

\* Modify HelloWorldFunction AWS::Lambda::Function

---------------------------------------------------------------------------------------------------------------------------------------------------

2019-11-21 14:33:24 - Waiting for stack create/update to complete

CloudFormation events from changeset

-------------------------------------------------------------------------------------------------------------------------------------------------

ResourceStatus ResourceType LogicalResourceId ResourceStatusReason

-------------------------------------------------------------------------------------------------------------------------------------------------

UPDATE\_IN\_PROGRESS AWS::IAM::Role HelloWorldFunctionRole -

UPDATE\_COMPLETE AWS::IAM::Role HelloWorldFunctionRole -

UPDATE\_IN\_PROGRESS AWS::Lambda::Function HelloWorldFunction -

UPDATE\_COMPLETE AWS::Lambda::Function HelloWorldFunction -

CREATE\_IN\_PROGRESS AWS::ApiGateway::RestApi ServerlessRestApi -

CREATE\_COMPLETE AWS::ApiGateway::RestApi ServerlessRestApi -

CREATE\_IN\_PROGRESS AWS::ApiGateway::RestApi ServerlessRestApi Resource creation Initiated

CREATE\_IN\_PROGRESS AWS::ApiGateway::Deployment ServerlessRestApiDeployment47fc2d5 Resource creation Initiated

f9d

CREATE\_IN\_PROGRESS AWS::Lambda::Permission HelloWorldFunctionHelloWorldPermis Resource creation Initiated

sionProd

CREATE\_IN\_PROGRESS AWS::Lambda::Permission HelloWorldFunctionHelloWorldPermis -

sionProd

CREATE\_IN\_PROGRESS AWS::ApiGateway::Deployment ServerlessRestApiDeployment47fc2d5 -

f9d

CREATE\_COMPLETE AWS::ApiGateway::Deployment ServerlessRestApiDeployment47fc2d5 -

f9d

CREATE\_IN\_PROGRESS AWS::ApiGateway::Stage ServerlessRestApiProdStage -

CREATE\_IN\_PROGRESS AWS::ApiGateway::Stage ServerlessRestApiProdStage Resource creation Initiated

CREATE\_COMPLETE AWS::ApiGateway::Stage ServerlessRestApiProdStage -

CREATE\_COMPLETE AWS::Lambda::Permission HelloWorldFunctionHelloWorldPermis -

sionProd

UPDATE\_COMPLETE\_CLEANUP\_IN\_PROGRES AWS::CloudFormation::Stack sam-app -

S

UPDATE\_COMPLETE AWS::CloudFormation::Stack sam-app -

-------------------------------------------------------------------------------------------------------------------------------------------------

Stack sam-app outputs:

---------------------------------------------------------------------------------------------------------------------------------------------------

OutputKey-Description OutputValue

---------------------------------------------------------------------------------------------------------------------------------------------------

HelloWorldFunctionIamRole - Implicit IAM Role created for Hello World arn:aws:iam::123456789012:role/sam-app-

function HelloWorldFunctionRole-104VTJ0TST7M0

HelloWorldApi - API Gateway endpoint URL for Prod stage for Hello World https://0ks2zue0zh.execute-api.us-east-1.amazonaws.com/Prod/hello/

function

HelloWorldFunction - Hello World Lambda Function ARN arn:aws:lambda:us-east-1:123456789012:function:sam-app-

HelloWorldFunction-1TY92MJX0BXU5

---------------------------------------------------------------------------------------------------------------------------------------------------

Successfully created/updated stack - sam-app in us-east-1

**What AWS SAM is doing:**

This command deploys your application to the AWS cloud. It take the deployment artifacts you build with the sam build command, packages and uploads them to an Amazon S3 bucket created by AWS SAM CLI, and deploys the application using AWS CloudFormation. In the output of the deploy command you can see the changes being made to your AWS CloudFormation stack.

If your application created a HTTP endpoint, the Outputs generated by sam deploy also show you the endpoint URL for your test application. You can use curl to send a request to your application using that endpoint URL. For example:

curl https://<restapiid>.execute-api.us-east-1.amazonaws.com/Prod/hello/

You should see output like the following after successfully deploying your application:

{"message": "hello world"}

If you see {"message": "hello world"} after executing the curl command, it means that you've successfully deployed your serverless application to AWS, and are calling your live Lambda function. Otherwise, see the [Troubleshooting](https://github.com/awsdocs/aws-sam-developer-guide/blob/master/doc_source/serverless-getting-started-hello-world.md#serverless-getting-started-hello-world-troubleshooting) section later in this tutorial.

## **Step 4: Testing Your Application Locally**

When you're developing your application, you might also find it useful to test locally. The AWS SAM CLI provides the sam local command to run your application using Docker containers that simulate the execution environment of Lambda.

### **Test your Lambda function directly**

Command to run:

sam local invoke "HelloWorldFunction" -e events/event.json

Example output:

2019-07-01 14:08:42 Found credentials in shared credentials file: ~/.aws/credentials

2019-07-01 14:08:42 Invoking app.lambda\_handler (python3.7)

Fetching lambci/lambda:python3.7 Docker container image...............................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................................

2019-07-01 14:09:39 Mounting /<working-development-path>/sam-app/.aws-sam/build/HelloWorldFunction as /var/task:ro,delegated inside runtime container

START RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72 Version: $LATEST

END RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72

REPORT RequestId: 52fdfc07-2182-154f-163f-5f0f9a621d72 Duration: 3.51 ms Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 22 MB

{"statusCode":200,"body":"{\"message\": \"hello world\"}"}

**What AWS SAM is doing:**

The invoke command directly invokes your Lambda functions, and can pass input event payloads that you provide. With this command, you pass the event payload in the file event.json that's provided by the sample application.

## **Troubleshooting**

### **SAM CLI error: "no such option: --app-template"**

When executing sam init, you see the following error:

Error: no such option: --app-template

This means that you are using an older version of the AWS SAM CLI that does not support the --app-template parameter. To fix this, you can either update your version of AWS SAM CLI to 0.33.0 or later, or omit the --app-template parameter from the sam init command.

### **SAM CLI error: "no such option: --guided"**

When executing sam deploy, you see the following error:

Error: no such option: --guided

This means that you are using an older version of the AWS SAM CLI that does not support the --guided parameter. To fix this, you can either update your version of AWS SAM CLI to 0.33.0 or later, or omit the --guided parameter from the sam deploy command.

### **SAM CLI error: "Failed to create managed resources: Unable to locate credentials"**

When executing sam deploy, you see the following error:

Error: Failed to create managed resources: Unable to locate credentials

This means that you have not set up AWS credentials to enable the AWS SAM CLI to make AWS service calls. To fix this, you must set up AWS credentials. For more information, see [Setting Up AWS Credentials](https://github.com/awsdocs/aws-sam-developer-guide/blob/master/doc_source/serverless-getting-started-set-up-credentials.md).

### **SAM CLI error: "Running AWS SAM projects locally requires Docker. Have you got it installed?"**

When executing sam local invoke, you see the following error:

Error: Running AWS SAM projects locally requires Docker. Have you got it installed?

This means that you do not have Docker properly installed. Docker is required to test your application locally. To fix this, follow the instructions for installing Docker for your development host.

For instructions on installing Docker on your development host, go to [Installing the AWS SAM CLI](https://github.com/awsdocs/aws-sam-developer-guide/blob/master/doc_source/serverless-sam-cli-install.md), choose the appropriate platform, and follow the instructions in the section titled Install Docker.

## **Clean Up**

If you no longer need the AWS resources you created by running this tutorial, you can remove them by deleting the AWS CloudFormation stack that you deployed.

To delete the AWS CloudFormation stack created with this tutorial using the AWS Management Console, follow these steps:

1. Sign in to the AWS Management Console and open the AWS CloudFormation console at [https://console.aws.amazon.com/cloudformation](https://console.aws.amazon.com/cloudformation/).
2. In the left navigation pane, choose Stacks.
3. In the list of stacks, choose sam-app (or the name of stack you created).
4. Choose Delete.

When done, the status of the of the stack will change to DELETE\_COMPLETE.

Alternatively, you can delete the AWS CloudFormation stack by executing the following AWS CLI command:

aws cloudformation delete-stack --stack-name sam-app --region region

### **Verify Deleted Stack**

For both methods of deleting the AWS CloudFormation stack, you can verify it was deleted by going to the [https://console.aws.amazon.com/cloudformation](https://console.aws.amazon.com/cloudformation/), choosing Stacks in the left navigation pane, and choosing Deleted in the dropdown to the right of the search text box. You should see your stack name sam-app (or the name of the stack you created) in the list of deleted stacks.