

# Deploy AWS Serverless Application Model (SAM)

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

# Get Security Credentials from AWS Console to login via local system

The screenshot shows the AWS IAM console interface. The left sidebar contains navigation links for Identity and Access Management (IAM), including Dashboard, Access management, Users, Roles, Policies, Identity providers, Account settings, Access reports, Access analyzer, Archive rules, Analyzers, Settings, Credential report, Organization activity, and Service control policies (SCPs). The main content area displays the 'Summary' page for the user 'swetpodd@in.ibm.com'. The 'Security credentials' tab is selected, showing 'Sign-in credentials' and 'Access keys'. The 'Access keys' section includes a 'Create access key' button and a table of existing access keys. The first access key is highlighted with a red circle.

Access key ID	Created	Last used	Status
AKIA6DVO2TOQP5AIPJ	2021-05-21 17:56 UTC+0530	2021-06-07 22:25 UTC+0530 with sts in us-east-1	Active

# Login to AWS Console via local

\$ aws configure

```
GMX+04828W744@DESKTOP-5KTEPAC MINGW64 ~/Desktop/AWS_Cloud/Assignments/SAM
$ aws configure
AWS Access Key ID [*****LKN6]: AKIA6DIVO2TOULIMRH62
AWS Secret Access Key [*****5ib0]: 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
Default region name [us-east-2]:
Default output format [json]:
```

## Check if you logged-in successfully

`$ aws iam list-users`

```
GMX+04828W744@DESKTOP-5KTEPAC MINGW64 ~/Desktop/AWS_Cloud/Assignments/SAM
$ aws iam list-users
{
  "Users": [
    {
      "Path": "/",
      "UserName": "swetpodd@in.ibm.com",
      "UserId": "AIDA6DIVO2TOYGPW52CK7",
      "Arn": "arn:aws:iam::969097008349:user/swetpodd@in.ibm.com",
      "CreateDate": "2021-05-21T12:26:00+00:00"
    }
  ]
}
```

Install AWS SAM CLI on Windows operating system and check the SAM version

`$ sam --version`

```
Microsoft Windows [Version 10.0.19042.985]  
(c) Microsoft Corporation. All rights reserved.  
  
C:\Users\SwetaPoddar>sam --version  
SAM CLI, version 1.24.1
```

## Step 1: Download a sample AWS SAM application

Run the below command to  
create the SAM template and  
enter corresponding template  
values

**\$ sam init**

```
C:\Users\SwetaPoddar>sam init
Which template source would you like to use?
  1 - AWS Quick Start Templates
  2 - Custom Template Location
Choice: 1
What package type would you like to use?
  1 - Zip (artifact is a zip uploaded to S3)
  2 - Image (artifact is an image uploaded to an ECR image repository)
Package type: 1
Which runtime would you like to use?
  1 - nodejs14.x
  2 - python3.8
  3 - ruby2.7
  4 - go1.x
  5 - java11
  6 - dotnetcore3.1
  7 - nodejs12.x
  8 - nodejs10.x
  9 - python3.7
 10 - python3.6
 11 - python2.7
 12 - ruby2.5
 13 - java8.al2
 14 - java8
 15 - dotnetcore2.1
Runtime: 5
Which dependency manager would you like to use?
  1 - maven
  2 - gradle
Dependency manager: 1
Project name [sam-app]: wave-18
Cloning from https://github.com/aws/aws-sam-cli-app-templates
AWS quick start application templates:
  1 - Hello World Example: Maven
  2 - EventBridge Hello World: Maven
  3 - EventBridge App from scratch (100+ Event Schemas): Maven
  4 - Step Functions Sample App (Stock Trader): Maven
Template selection: 1

-----
Generating application:
-----
Name: wave-18
Runtime: java11
Dependency Manager: maven
Application Template: hello-world
Output Directory: .

Next steps can be found in the README file at ./wave-18/README.md
```

## Step 2: Build your application

To build the serverless application as a .zip file archive, declare PackageType: Zip for the serverless function.

- Go inside the `<wave-18>` folder & Run the below command

**\$ sam build**

```
C:\Users\SwetaPoddar\wave-18>sam build
Building codeuri: C:\Users\SwetaPoddar\wave-18\HelloWorldFunction runtime: java11 metadata: {} functions: ['HelloWorldFunction']
Running JavaMavenWorkflow:CopySource
Running JavaMavenWorkflow:MavenBuild
Running JavaMavenWorkflow:MavenCopyDependency
Running JavaMavenWorkflow:MavenCopyArtifacts

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
```

## Step 3: Deploy your application to the AWS Cloud

\$ sam deploy --guided

```
C:\Users\SwetaPoddar\wave-18>sam deploy --guided
Configuring SAM deploy
=====

Looking for config file [samconfig.toml] : Not found

Setting default arguments for 'sam deploy'
=====
Stack Name [sam-app]:
AWS Region [us-east-2]:
#Shows you resources changes to be deployed and require a 'Y' to initiate deploy
Confirm changes before deploy [y/N]: Y
#SAM needs permission to be able to create roles to connect to the resources in your template
Allow SAM CLI IAM role creation [Y/n]: Y
HelloWorldFunction may not have authorization defined, Is this okay? [y/N]: Y
Save arguments to configuration file [Y/n]: Y
SAM configuration file [samconfig.toml]:
SAM configuration environment [default]:

Looking for resources needed for deployment: Not found.
Creating the required resources...
Successfully created!

Managed S3 bucket: aws-sam-cli-managed-default-samclisourcebucket-t13kj0pq4at7
A different default S3 bucket can be set in samconfig.toml

Saved arguments to config file
Running 'sam deploy' for future deployments will use the parameters saved above.
The above parameters can be changed by modifying samconfig.toml
Learn more about samconfig.toml syntax at
https://docs.aws.amazon.com/serverless-application-model/latest/developerguide/serverless-sam-cli-config.html

Uploading to sam-app/476d62b39fcb245566d475781587145f 732820 / 732820 (100.00%)

Deploying with following values
=====
Stack name      : sam-app
Region         : us-east-2
Confirm changeset : True
Deployment S3 bucket : aws-sam-cli-managed-default-samclisourcebucket-t13kj0pq4at7
Capabilities    : ["CAPABILITY_IAM"]
Parameter overrides : {}
Signing Profiles : {}

Initiating deployment
=====
Uploading to sam-app/57079b81f31b5e8efb9c86d11bd9b8.template 1226 / 1226 (100.00%)

Waiting for changeset to be created..

CloudFormation stack changeset
-----
Operation LogicalResourceId ResourceType Replacement
-----
+ Add HelloWorldFunctionHelloWorldPermissionProd AWS::Lambda::Permission N/A
+ Add HelloWorldFunctionRole AWS::IAM::Role N/A
+ Add HelloWorldFunction AWS::Lambda::Function N/A
+ Add ServerlessRestApiDeployment47fc2d5f9d AWS::ApiGateway::Deployment N/A
+ Add ServerlessRestApiProdStage AWS::ApiGateway::Stage N/A
+ Add ServerlessRestApi AWS::ApiGateway::RestApi N/A
```



Changeset created successfully. arn:aws:cloudformation:us-east-2:969097008349:changeSet/samcli-deploy1623339159/345c35e4-ff93-4511-97dd-b6a0427fb2e3

Previewing CloudFormation changeset before deployment

Deploy this changeset? [y/N]: Y

2021-06-10 21:03:57 - Waiting for stack create/update to complete

CloudFormation events from changeset

ResourceStatus	ResourceType	LogicalResourceId	ResourceStatusReason
CREATE_IN_PROGRESS	AWS::IAM::Role	HelloWorldFunctionRole	Resource creation Initiated
CREATE_IN_PROGRESS	AWS::IAM::Role	HelloWorldFunctionRole	-
CREATE_COMPLETE	AWS::IAM::Role	HelloWorldFunctionRole	-
CREATE_IN_PROGRESS	AWS::Lambda::Function	HelloWorldFunction	-
CREATE_COMPLETE	AWS::Lambda::Function	HelloWorldFunction	-
CREATE_IN_PROGRESS	AWS::Lambda::Function	HelloWorldFunction	Resource creation Initiated
CREATE_IN_PROGRESS	AWS::ApiGateway::RestApi	ServerlessRestApi	-
CREATE_COMPLETE	AWS::ApiGateway::RestApi	ServerlessRestApi	-
CREATE_IN_PROGRESS	AWS::ApiGateway::RestApi	ServerlessRestApi	Resource creation Initiated
CREATE_COMPLETE	AWS::ApiGateway::Deployment	ServerlessRestApiDeployment47fc2d5f9d	-
CREATE_IN_PROGRESS	AWS::ApiGateway::Deployment	ServerlessRestApiDeployment47fc2d5f9d	Resource creation Initiated
CREATE_IN_PROGRESS	AWS::Lambda::Permission	HelloWorldFunctionHelloWorldPermissionProd	Resource creation Initiated
CREATE_IN_PROGRESS	AWS::Lambda::Permission	HelloWorldFunctionHelloWorldPermissionProd	-
CREATE_IN_PROGRESS	AWS::ApiGateway::Deployment	ServerlessRestApiDeployment47fc2d5f9d	-
CREATE_COMPLETE	AWS::ApiGateway::Stage	ServerlessRestApiProdStage	-
CREATE_IN_PROGRESS	AWS::ApiGateway::Stage	ServerlessRestApiProdStage	Resource creation Initiated
CREATE_IN_PROGRESS	AWS::ApiGateway::Stage	ServerlessRestApiProdStage	-
CREATE_COMPLETE	AWS::Lambda::Permission	HelloWorldFunctionHelloWorldPermissionProd	-
CREATE_COMPLETE	AWS::CloudFormation::Stack	sam-app	-

CloudFormation outputs from deployed stack

Outputs

Key	HelloWorldFunctionIamRole
Description	Implicit IAM Role created for Hello World function
Value	arn:aws:iam::969097008349:role/sam-app-HelloWorldFunctionRole-17JFJEUCT09I3
Key	HelloWorldApi
Description	API Gateway endpoint URL for Prod stage for Hello World function
Value	https://e8gpv6tq9d.execute-api.us-east-2.amazonaws.com/Prod/hello/
Key	HelloWorldFunction
Description	Hello World Lambda Function ARN
Value	arn:aws:lambda:us-east-2:969097008349:function:sam-app-HelloWorldFunction-gyjx6CLTdPkB

Successfully created/updated stack - sam-app in us-east-2



Now back to AWS  
Console..

# Step 01: Search for “*Lambda*”

This will show you the console where application is deployed..

The screenshot shows the AWS Lambda console interface. The left sidebar contains navigation links: Dashboard, Applications, Functions, Additional resources, and Related AWS resources. The main content area displays a list of functions under the heading 'Functions (1)'. A red circle highlights the function 'sam-app-HelloWorldFunction-gyjx6CITdPkB' in the 'Function name' column. The table also shows the package type as 'Zip', runtime as 'Java 11 (Corretto)', code size as '715.6 kB', and last modified time as '7 minutes ago'. A search bar at the top of the console allows filtering by tags and attributes or searching by keyword.

Function name	Description	Package type	Runtime	Code size	Last modified
sam-app-HelloWorldFunction-gyjx6CITdPkB		Zip	Java 11 (Corretto)	715.6 kB	7 minutes ago

## Step 02 : Click on the application

The screenshot displays the AWS Lambda console interface. At the top, the navigation bar includes the AWS logo, 'Services', a search bar, and user information. The breadcrumb trail shows 'Lambda > Functions > sam-app-HelloWorldFunction-gyjx6CITdPkB'. The main heading is 'sam-app-HelloWorldFunction-gyjx6CITdPkB', with buttons for 'Throttle', 'Copy ARN', and 'Actions'. A blue notification bar states: 'This function belongs to an application. [Click here](#) to manage it.'

The 'Function overview' section is expanded, showing a diagram of the function's configuration. The diagram includes a Lambda function icon labeled 'sam-app-HelloWorldFunction-gyjx6CITdPkB' with '(0)' layers, and an 'API Gateway' trigger icon with an '+ Add trigger' button. An '+ Add destination' button is also present. To the right, a metadata panel lists: 'Description: -', 'Last modified: 9 minutes ago', 'Function ARN: arn:aws:lambda:us-east-2:969097008349:function:sam-app-HelloWorldFunction-gyjx6CITdPkB', and 'Application: [sam-app](#)'.

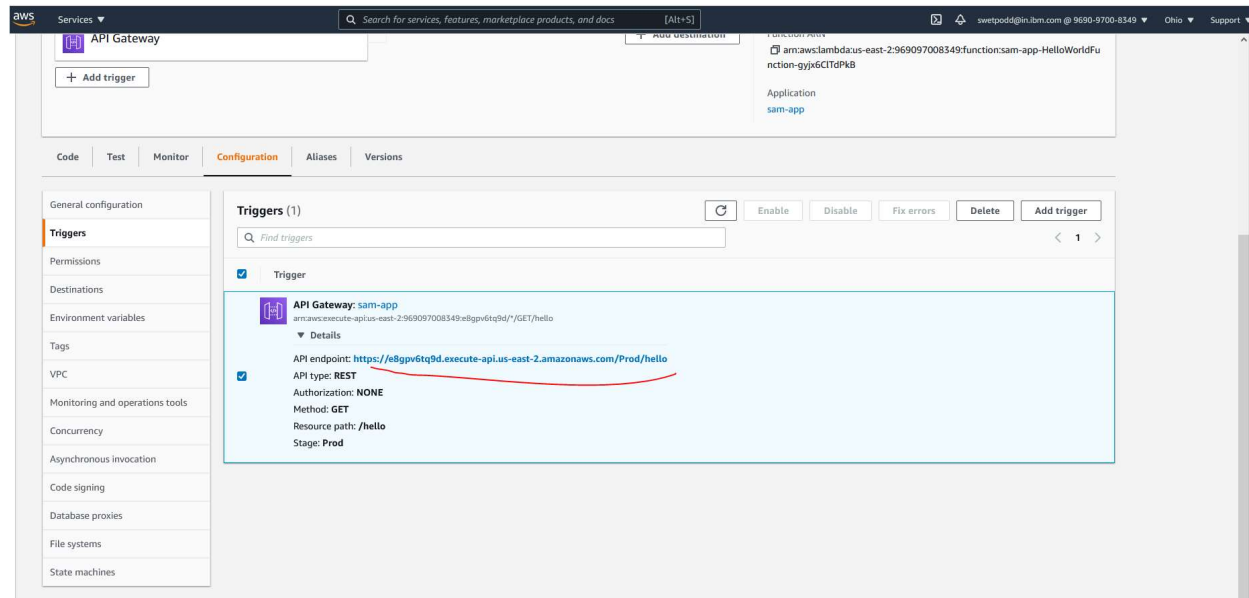
At the bottom, a tabbed interface shows 'Code' as the active tab, with other tabs for 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Code source' section is visible, featuring an 'Upload from' button.

Step 03 : Click on  
"Configuration"  
tab.

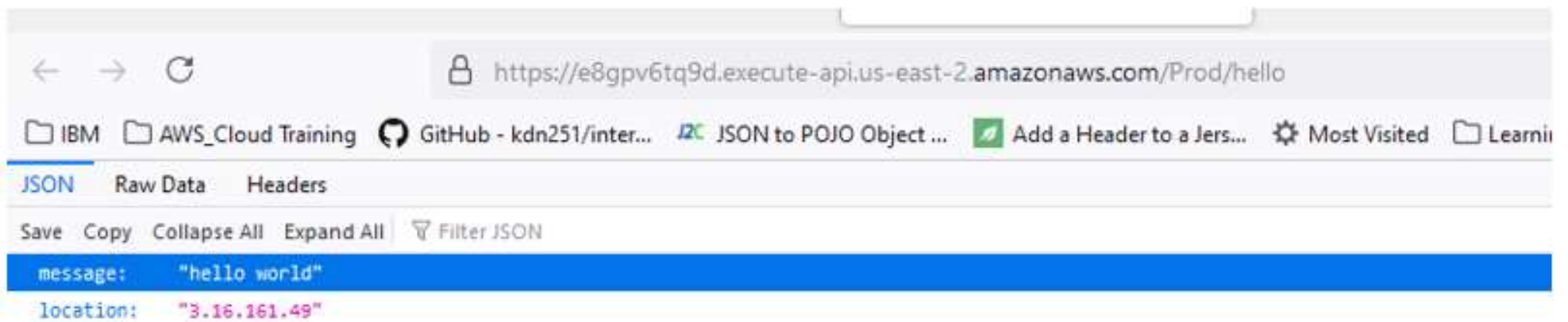
The screenshot displays the AWS API Gateway console interface. At the top, the 'API Gateway' header is visible with a search bar and user information. Below the header, there's a 'Configuration' tab selected among others like 'Code', 'Test', 'Monitor', 'Aliases', and 'Versions'. The left sidebar contains a navigation menu with options such as 'General configuration', 'Triggers', 'Permissions', 'Destinations', 'Environment variables', 'Tags', 'VPC', 'Monitoring and operations tools', 'Concurrency', 'Asynchronous invocation', 'Code signing', 'Database proxies', 'File systems', and 'State machines'. The main content area shows the 'Triggers (1)' section, which includes a search bar and a table of triggers. A single trigger is listed with the following details:

- API Gateway:** sam-app
- Function ARN:** arn:aws:execute-api:us-east-2:969097008349:ebgpv6tq9d/\*GET/hello
- Details:**
  - API endpoint: <https://ebgpv6tq9d.execute-api.us-east-2.amazonaws.com/Prod/hello>
  - API type: REST
  - Authorization: NONE
  - Method: GET
  - Resource path: /hello
  - Stage: Prod

Step 04 : Now open  
—“*API endpoint*” in  
the browser.



After successfully deploying the application, below output will show on browser



Step 05 : Now  
change path  
in  
“*template.yml*”

```
Resources:
  HelloWorldFunction:
    Type: AWS::Serverless::Function # More info about Function Resource: https://github.com/aws/aws-lambda-python/blob/master/doc/lambda\_function.md
    Properties:
      CodeUri: HelloWorldFunction
      Handler: helloworld.App::handleRequest
      Runtime: java11
      MemorySize: 512
      Environment: # More info about Env Vars: https://github.com/aws/aws-lambda-python/blob/master/doc/lambda\_function.md#environment-object
        Variables:
          PARAM1: VALUE
      Events:
        HelloWorld:
          Type: Api # More info about API Event Source: https://github.com/aws/aws-lambda-python/blob/master/doc/lambda\_function.md#api
          Properties:
            Path: /welcome
            Method: get

Outputs:
  # ServerlessRestApi is an implicit API created out of Events key under Serverless::Function
  # Follow link (ctrl + click) ther implicit resources you can reference within SAM
  # https://github.com/aws/aws-lambda-python/blob/master/doc/lambda\_function.md#api
  HelloWorldApi:
    Description: "API Gateway endpoint URL for Prod stage for Hello World function"
    Value: !Sub "https://${ServerlessRestApi}.execute-api.${AWS::Region}.amazonaws.com/Prod/welcome/"
  HelloWorldFunction:
    Description: "Hello World Lambda Function ARN"
    Value: !GetAtt HelloWorldFunction.Arn
  HelloWorldFunctionIamRole:
    Description: "Implicit IAM Role created for Hello World function"
    Value: !GetAtt HelloWorldFunctionRole.Arn
```



Step 06: Make changes in

*"HelloWorldFunction\src\main\java\helloworld"*

```
/**
 * Handler for requests to Lambda function.
 */
public class App implements RequestHandler<APIGatewayProxyRequestEvent, APIGatewayProxyResponseEvent> {

    public APIGatewayProxyResponseEvent handleRequest(final APIGatewayProxyRequestEvent input, final Context context) {
        Map<String, String> headers = new HashMap<>();
        headers.put("Content-Type", "application/json");
        headers.put("X-Custom-Header", "application/json");

        APIGatewayProxyResponseEvent response = new APIGatewayProxyResponseEvent()
            .withHeaders(headers);

        try {
            final String pageContents = this.getPageContents("https://checkip.amazonaws.com");
            String output = String.format("{ \"message\": \"Sweta- welcome to the Stack Route!\", \"location\": \"%s\" }", pageContents);

            return response
                .withStatusCode(200)
                .withBody(output);
        } catch (IOException e) {
            return response
                .withBody("{}")
                .withStatusCode(500);
        }
    }

    private String getPageContents(String address) throws IOException{
        URL url = new URL(address);
        try(BufferedReader br = new BufferedReader(new InputStreamReader(url.openStream()))){
            return br.lines().collect(Collectors.joining(System.lineSeparator()));
        }
    }
}
```

Step 07: Now “*build*” & “*deploy*” application like before to apply changes.  
Step 08 : Run it with changed URL path



# The End

Compiled By,

*SWETA PODDAR*

