Build Amazon SNS Integration with AWS SAM and OpenAPI

You can integrate an API method directly with an Amazon SNS topic without the need for a compute layer such as Lambda in the middle. This is achieved through the **AWS service integration** type.

Use AWS SAM and OpenAPI to create an API Gateway REST API with a direct Amazon SNS integration

1. Using AWS Cloud9 console, return to the root folder `module-3/SNS`
2. This code belongs in your SAM template file `template.yaml`

   Review the code and then **copy/paste** it into the `template.yaml` file.

```
AWSTemplateFormatVersion: '2010-09-09'
Transform: 'AWS::Serverless-2016-10-31'
Description: >
  module3-sns: Sample SAM Template for module3-sns

Globals:

  #Enable Logs
  Api:
    MethodSettings:
      - ResourcePath: "/*"
        HttpMethod: "*"
        DataTraceEnabled: True
        LoggingLevel: INFO
        MetricsEnabled: True

Resources:

    #Create the SNS Topic
    SNSTopicForAPIGW:
      Type: AWS::SNS::Topic
      Properties:
        TopicName: MyAPIGWSNSTopic

    #Create the IAM role
    IAMRoleForSNSIntegration:
      Type: AWS::IAM::Role
      Properties:
        AssumeRolePolicyDocument:
          Version: "2012-10-17"
          Statement:
            - Effect: Allow
              Principal:
                Service:
                  - apigateway.amazonaws.com
              Action:
                - 'sts:AssumeRole'
        Path: /
        Policies:
          - PolicyName: PolicyForAPIGWSQSIntegration
            PolicyDocument:
              Version: "2012-10-17"
              Statement:
                - Effect: Allow
                  Action: 'sns:Publish'
                  Resource: !Ref SNSTopicForAPIGW

    # Create the API Gateway
    API1WithSNSIntegration:
        Type: AWS::Serverless::Api
        Properties:
            StageName: dev
            OpenApiVersion: 3.0.3
            DefinitionBody: # an OpenApi definition
              "fn::Transform":
                Name: "AWS::Include"
                Parameters:
                  Location: "openapi.yaml"
        EndpointConfiguration:
            Type: REGIONAL

Outputs:
    APIGatewayEndpoint:
        Description: API Gateway Endpoint
        Value:
          Fn::Sub: https://${API1WithSNSIntegration}.execute-api.${AWS::Region}.amazonaws.com/dev/email-me
    SNSTopicARN:
        Description: ARN for SNS topic
        Value:
          Fn::Sub: ${SNSTopicForAPIGW}
```

The above template contains a mixture of both AWS SAM and CloudFormation resources. As SAM templates are an extenstion of CloudFormation, its possible to mix resources from both types as upon deployment, the SAM resources are converted to CloudFormation resources. Within this template, we have created an SNS topic, an IAM role that will give API Gateway permission to publish messages to the SNS service and an API Gateway. After a successful deployment, the template will output the API Gateway invoke URL as well as the ARN of the SNS topic.

3. This code belongs in your OpenAPI definition file `openapi.yaml`

   Review the code and then **copy/paste** it into the `openapi.yaml` file

```
openapi: "3.0.1"
info:
  title: "module-3-sns-integration"
  description: "API Gateway example for using SNS as direct service integration"
  version: "1.0"

paths:
  /email-me:
    post:
      consumes:
      - "application/json"
      produces:
      - "application/json"
      responses:
        "200":
          description: "200 response"
          schema:
            $ref: "#/definitions/Empty"
      x-amazon-apigateway-integration:
        uri: "arn:aws:apigateway:${AWS::Region}:sns:path//"
        credentials:
          Fn::Sub: ${IAMRoleForSNSIntegration.Arn}
        httpMethod: "POST"
        responses:
          default:
            statusCode: "200"
        requestTemplates:
          application/json:
            Fn::Sub: "Action=Publish&TopicArn=${util.urlEncode('${SNSTopicForAPIGW}')}&Message=$util.urlEncode($input.body)"
        requestParameters:
          integration.request.header.Content-Type: "'application/x-www-form-urlencoded'"
        passthroughBehavior: "never"
        type: "aws"

```

Lines **(19-33)** are particulary important as these are what create the direct service integration for SNS. The x-amazon-apigateway-integration is an OpenAPI extension that allows us to integrate our APIs to AWS backends.

> **Info:** The SNS Publish API call expects a POST request where the relevant parameters are passed in the body as form data.

Line **(29)** defines our integration request template, which transforms the incoming HTTP request, including the request body into the URL encoded format that the SNS Publish API expects. Within this template, we also define the intended action (which is to call the Publish API) as well as the topic we want to publish the message to, and finally what message we want to send.

Line **(31)** sets the Content-Type header that is sent to the backend. As the SNS endpoint expects the request to be URL encoded, we set this value to 'application/x-www-form-urlencoded'.

Deploy the project

1. To deploy the API Gateway and SNS resources to your AWS account, run the following commands from the application root `module-3/SNS`, where the `template.yaml` file for the sample application is located:

```
sam build && sam deploy --guided
```

The first time that you run the `sam deploy --guided` command, AWS SAM starts an AWS CloudFormation deployment. In this case, you need to say what are the configurations that you want SAM to have in order to get the guided deployment. You can configure it as below.

- **Stack Name**: `module-3-sns-integration`
- **AWS Region**: Put the chosen region to run the workshop. **e.g.** `us-east-1`
- **Confirm changes before deploy**: Y
- **Allow SAM CLI IAM role creation**: Y
- **Disable rollback**: N
- **Save arguments to configuration file**: Y
- **SAM configuration file** and **SAM configuration environment** leave blank

```
Configuring SAM deploy
======================

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

        Setting default arguments for 'sam deploy'
        =========================================
        Stack Name [sam-app]: module-3-sns-integration
        AWS Region [eu-west-1]:
        #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
        #Preserves the state of previously provisioned resources when an operation fails
        Disable rollback [y/N]: N
        Save arguments to configuration file [Y/n]: Y
        SAM configuration file [samconfig.toml]:
        SAM configuration environment [default]:
```

2. After configuring the deployment, AWS SAM will display assets that will be created. But first, it will automatically upload the template to a temporary bucket it creates. Then, it will ask you to confirm the changes. Type `y` to confirm.

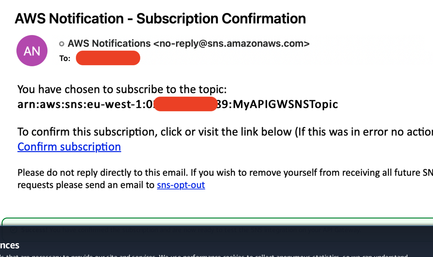
```
---------------------------------------------------------------------------------------------------------
Outputs
---------------------------------------------------------------------------------------------------------
Key                 APIGatewayEndpoint
Description         API Gateway Endpoint
Value               https://[redacted].execute-api.eu-west-1.amazonaws.com/dev/email-me

Key                 SNSTopicARN
Description         ARN for SNS topic
Value               arn:aws:sns:eu-west-1:02[redacted]89:MyAPIGWSNSTopic
---------------------------------------------------------------------------------------------------------
```

3. Once the deployment has been successful, you will see an '**Outputs**' section that contains the API Gateway invoke URL and the ARN of the SNS topic you just created - **take note of the SNS topic ARN** as you will need this for the next step.
4. With our SNS topic created, you now need to create a subscription in order to receive the messages sent to this topic. For the purposes of this demo, you will use your email address as the notification endpoint. Return to your Cloud9 environment and paste in the below command, being sure to replace **SNS Topic ARN** with the ARN provided in the Outputs section of the SAM deployment and **YOUR EMAIL HERE** with an email address you have ownership of.

```
aws sns subscribe --topic-arn <SNS Topic ARN> --protocol email --notification-endpoint <YOUR EMAIL HERE>
```

5. After running the above command, you should get the response 'Pending Confirmation'. To confirm the subscription, navigate to the email address you used to subscribe to the topic and look for an email from 'no-reply@sns.amazonaws.com'. From this email, click '**Confirm subscription**'.

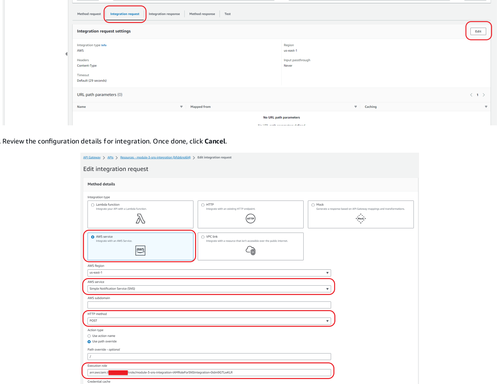


Test the Integration

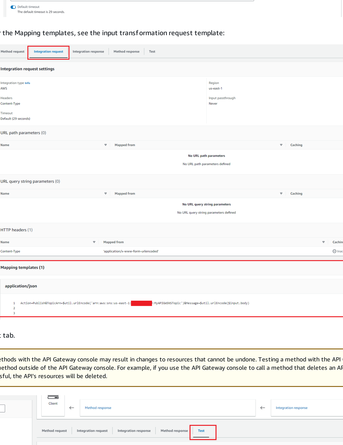
Now the API has been deployed and subscription confirmed, its time to test the integration.

Test SNS Integration using API Gateway console

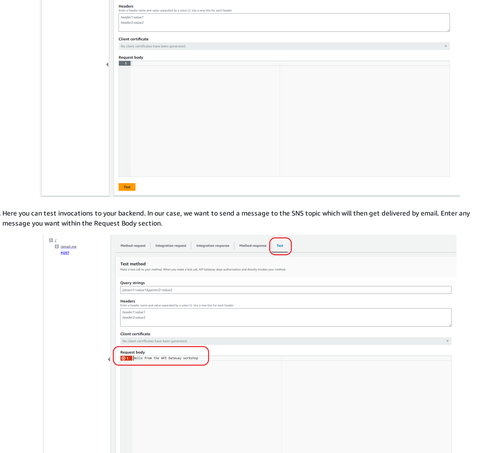
1. Navigate to the API Gateway console and select the API that has been created. It will be named **module-3-sns-integration**.
2. Select the POST method from the resources section to the left of the screen to bring up the API request flow window.
3. Click in **Integration Request** and then **Edit**.



4. Review the configuration details for integration. Once done, click **Cancel**.

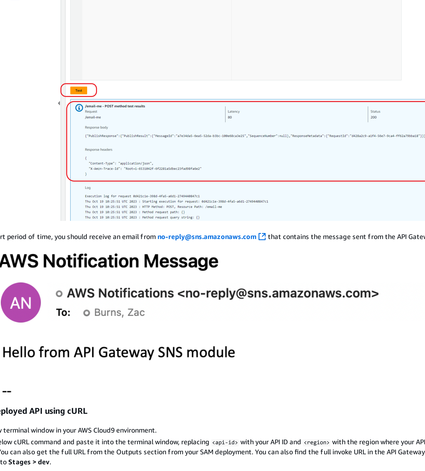


5. Scroll down, and look for the Mapping templates, see the input transformation request template:



6. Scroll up, select the Test tab.

> **Warning!** Testing methods with the API Gateway console may result in changes to resources that cannot be undone. Testing a method with the API Gateway console is the same as calling the method outside of the API Gateway console. For example, if you use the API Gateway console to call a method that deletes an API's resources, if the method call is successful, the API's resources will be deleted.



7. Here you can test invocations to your backend. In our case, we want to send a message to the SNS topic which will then get delivered by email. Enter any message you want within the Request Body section.

8. Hit the Test button. From this, you should receive a Status code of 200 meaning the message was successfully delivered to the SNS topic.

9. After a short period of time, you should receive an email from **no-reply@sns.amazonaws.com** that contains the message sent from the API Gateway.

Test the deployed API using cURL

1. Open a new terminal window in your AWS Cloud9 environment.
2. Copy the below cURL command and paste it into the terminal window, replacing `<api-id>` with your API ID and `<region>` with the region where your API is deployed. You can also get the full URL from the Outputs section from your SAM deployment. You can also find the full invoke URL in the API Gateway console by navigating to **Stages > dev**.

```
curl -X POST \
'https://<api-id>.execute-api.<region>.amazonaws.com/dev/email-me' \
-H 'Content-Type: application/json' \
-d 'Hello from API Gateway workshop'
```

> **Info:** We need to ensure we send the 'application/json' Content-Type header due to the fact our Request Body passthrough settings are set to **Never** meaning the request will not be sent to the backend unless the request is using this supported Content Type.

3. The **Response Body** output should be:

```
{
    "PublishResponse": {
        "PublishResult": {
            "MessageId": "f9f3a5e7-58b4-5e4d-840c-b97923a2ee97",
            "SequenceNumber": null
        },
        "ResponseMetadata": {
            "RequestId": "d2dd09c6-99e0-5213-8e39-5558e2f612c9"
        }
    }
}
```

4. You will receive an email from **no-reply@sns.amazonaws.com** that contains the message sent from the API Gateway.

> **Challenge**
> Find the payload logs **before** and **after** the **request** and **response** transformations in **CloudWatch**.