Build Amazon API Gateway Integration with Amazon S3

1. This code belongs to the SAM template file `template.yaml`

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

```
1  AWSTemplateFormatVersion: '2010-09-09'
2  Transform: 'AWS::Serverless-2016-10-31'
3  Description: >
4    module3-s3-rest-api: Sample SAM Template for module3-s3-proxy-rest-api
5
6  Globals:
7
8      # Enable Logs
9      Api:
10         MethodSettings:
11             - ResourcePath: "/*"
12               HttpMethod: "*"
13               DataTraceEnabled: True
14               LoggingLevel: INFO
15               MetricsEnabled: True
16
17 Resources:
18
19     # S3 Bucket to be integrated with our API
20     S3IntegrationBucket:
21         Type: AWS::S3::Bucket
22         Properties:
23           BucketName: !Sub 'demo-bucket-s3-proxy-${AWS::AccountId}'
24
25
26     # REST with proxy to our S3 bucket
27     APIWithS3Integration:
28         Type: AWS::Serverless::Api
29         Properties:
30             StageName: dev
31             OpenApiVersion: 3.0.3
32             DefinitionBody: # an OpenApi definition
33                 "Fn::Transform":
34                     Name: "AWS::Include"
35                     Parameters:
36                         Location: "openapi.yaml"
37             EndpointConfiguration:
38                 Type: REGIONAL
39
40     # IAM Role allowing API Gateway to read e write in our S3 bucket
41     IAMRoleForS3Integration:
42         Type: AWS::IAM::Role
43         Properties:
44             RoleName: RoleForAPIGatewayS3Integration
45             AssumeRolePolicyDocument:
46                 Version: "2012-10-17"
47                 Statement:
48                   - Effect: Allow
49                     Principal:
50                       Service:
51                         - apigateway.amazonaws.com
52                     Action:
53                       - 'sts:AssumeRole'
54             Policies:
55                 - PolicyName: PolicyForAPIGWS3Integration
56                   PolicyDocument:
57                     Version: "2012-10-17"
58                     Statement:
59                       - Effect: Allow
60                         Action: ["s3:PutObject","s3:GetObject","s3:DeleteObject", "s3:ListBucket"]
61                         Resource: [!Sub "arn:aws:s3:::demo-bucket-s3-proxy-${AWS::AccountId}/*",!Sub "arn:aws:s3:::demo-bucket-s3-proxy-${AWS::AccountId}"]
62
```

The code presented above is responsible for the creation of the following resources:

- `S3IntegrationBucket (line 23)` - This S3 bucket will be used to test our API Gateway with S3 proxy integration. We are going to store and retrieve files to it using our REST API.
- `APIWithS3Integration (line 30)` - This represents our API Gateway REST API. It will contains write and read endpoints linked to our `S3IntegrationBucket` resource. All the details of its configuration are defined in the openapi file, referenced by the `DefinitionBody` attribute.
- `IAMRoleForS3Integration (line 44)` - This is an IAM role that will be assumed by our API (`APIWithS3Integration`). It contains the necessary policies for accessing, adding and deleting files to our `S3IntegrationBucket`.

2. This code belongs to the OpenAPI definition file `openapi.yaml`. Review the code and then **copy/paste** it into the `openapi.yaml` file. Save the file.

```
1   openapi: "3.0.1"
2   info:
3     title: "Api-module-3-s3-integration"
4   paths:
5     /:
6       get:
7         responses:
8           "400":
9             description: "400 response"
10          "500":
11            description: "500 response"
12          "200":
13            description: "200 response"
14            headers:
15              Content-Length:
16                schema:
17                  type: "string"
18              Timestamp:
19                schema:
20                  type: "string"
21              Content-Type:
22                schema:
23                  type: "string"
24            content:
25              application/json:
26                schema:
27                  $ref: "#/components/schemas/Empty"
28        x-amazon-apigateway-integration:
29          credentials: "arn:aws:iam::${AWS::AccountId}:role/RoleForAPIGatewayS3Integration"
30          httpMethod: "GET"
31          uri: "arn:aws:apigateway:${AWS::Region}:s3:path/demo-bucket-s3-proxy-${AWS::AccountId}"
32          responses:
33            "4\\d{2}":
34              statusCode: "400"
35            default:
36              statusCode: "200"
37              responseParameters:
38                method.response.header.Content-Type: "integration.response.header.Content-Type"
39                method.response.header.Content-Length: "integration.response.header.Content-Length"
40                method.response.header.Timestamp: "integration.response.header.Date"
41            "5\\d{2}":
42              statusCode: "500"
43          passthroughBehavior: "when_no_match"
44          type: "aws"
45    /{filename}:
46      get:
47        parameters:
48        - name: "filename"
49          in: "path"
50          required: true
51          schema:
52            type: "string"
53        responses:
54          "400":
55            description: "400 response"
56          "500":
57            description: "500 response"
58          "200":
59            description: "200 response"
60            headers:
61              Content-Length:
62                schema:
63                  type: "string"
64              Timestamp:
65                schema:
66                  type: "string"
67              Content-Type:
68                schema:
69                  type: "string"
70            content:
71              application/json:
72                schema:
73                  $ref: "#/components/schemas/Empty"
74        x-amazon-apigateway-integration:
75          credentials: "arn:aws:iam::${AWS::AccountId}:role/RoleForAPIGatewayS3Integration"
76          httpMethod: "GET"
77          uri: "arn:aws:apigateway:${AWS::Region}:s3:path/demo-bucket-s3-proxy-${AWS::AccountId}/{key}"
78          responses:
79            "4\\d{2}":
80              statusCode: "400"
81            default:
82              statusCode: "200"
83              responseParameters:
84                method.response.header.Content-Type: "integration.response.header.Content-Type"
85                method.response.header.Content-Length: "integration.response.header.Content-Length"
86                method.response.header.Timestamp: "integration.response.header.Date"
87            "5\\d{2}":
88              statusCode: "500"
89          passthroughBehavior: "when_no_match"
90          requestParameters:
91            integration.request.path.key: "method.request.path.filename"
92          passthroughBehavior: "when_no_match"
93          type: "aws"
94      put:
95        parameters:
96        - name: "filename"
97          in: "path"
98          required: true
99          schema:
100           type: "string"
101       responses:
102         "400":
103           description: "400 response"
104         "500":
105           description: "500 response"
106         "200":
107           description: "200 response"
108           headers:
109             Content-Length:
110               schema:
111                 type: "string"
112             Timestamp:
113               schema:
114                 type: "string"
115             Content-Type:
116               schema:
117                 type: "string"
118           content:
119             application/json:
120               schema:
121                 $ref: "#/components/schemas/Empty"
122       x-amazon-apigateway-integration:
123         credentials: "arn:aws:iam::${AWS::AccountId}:role/RoleForAPIGatewayS3Integration"
124         httpMethod: "PUT"
125         uri: "arn:aws:apigateway:${AWS::Region}:s3:path/demo-bucket-s3-proxy-${AWS::AccountId}/{key}"
126         responses:
127           "4\\d{2}":
128             statusCode: "400"
129           default:
130             statusCode: "200"
131             responseParameters:
132               method.response.header.Content-Type: "integration.response.header.Content-Type"
133               method.response.header.Content-Length: "integration.response.header.Content-Length"
134               method.response.header.Timestamp: "integration.response.header.Date"
135           "5\\d{2}":
136             statusCode: "500"
137         passthroughBehavior: "when_no_match"
138         requestParameters:
139           integration.request.path.key: "method.request.path.filename"
140         passthroughBehavior: "when_no_match"
141         type: "aws"
142     delete:
143       parameters:
144       - name: "filename"
145         in: "path"
146         required: true
147         schema:
148           type: "string"
149       responses:
150         "400":
151           description: "400 response"
152         "500":
153           description: "500 response"
154         "200":
155           description: "200 response"
156           headers:
157             Content-Length:
158               schema:
159                 type: "string"
160             Timestamp:
161               schema:
162                 type: "string"
163             Content-Type:
164               schema:
165                 type: "string"
166           content:
167             application/json:
168               schema:
169                 $ref: "#/components/schemas/Empty"
170       x-amazon-apigateway-integration:
171         credentials: "arn:aws:iam::${AWS::AccountId}:role/RoleForAPIGatewayS3Integration"
172         httpMethod: "DELETE"
173         uri: "arn:aws:apigateway:${AWS::Region}:s3:path/demo-bucket-s3-proxy-${AWS::AccountId}/{key}"
174         responses:
175           "4\\d{2}":
176             statusCode: "400"
177           default:
178             statusCode: "200"
179             responseParameters:
180               method.response.header.Content-Type: "integration.response.header.Content-Type"
181               method.response.header.Content-Length: "integration.response.header.Content-Length"
182               method.response.header.Timestamp: "integration.response.header.Date"
183           "5\\d{2}":
184             statusCode: "500"
185         passthroughBehavior: "when_no_match"
186         requestParameters:
187           integration.request.path.key: "method.request.path.filename"
188         passthroughBehavior: "when_no_match"
189         type: "aws"
190 components:
191   schemas:
192     Empty:
193       title: "Empty Schema"
194       type: "object"
195 x-amazon-apigateway-binary-media-types:
196 - "plain/text"
```

The openapi.yaml file describes all the details related to our API endpoints. All the endpoints are described under the **paths** session, as following:

- **/** (line 5) - this is the root of our API. It contains a get method that will be used to list all files inside our S3 bucket.
- **/{filename}** (line 45) - this path represents all the actions that are related to specific files inside the bucket. It contains 3 methods, representing the actions described below:
  - **get**: Retrieve the content of specific file, using its key as parameter
  - **put**: Upload a new file to our S3 bucket
  - **delete**: Delete the file that contains the name specified in the path

  > The methods related to specific files (under **/{filename}**) also contains the `integration.request.path.key: "method.request.path.filename"` specification. It represents the Path Override behavior, that will be covered in the next session of this module.

Each method also contains a x-amazon-apigateway-integration object **(lines 28, 74, 122, 170)**. It specifies the integration behavior between each API Gateway method and our S3 bucket. It has the following parameters:

- `credentials`: represents the IAM role assumed by each API Gateway resource.
- `httpMethod`: the action to be performed by each API Gateway resource against our S3 bucket.
- `uri`: specifies the mapping between the method and its corresponding action performed over our S3 bucket.
- `requestParameters`: the specification of the parameters that will be necessary to perform the actions agains our S3 bucket. It is responsible for extracting information from the request path and translate it to API Gateway useful information to establish the integration with S3.
- `responses`: represents scenarios of success, client server errors and server side errors and their representations to the frontend.
- Code `type: "aws"` is responsible for configuring the integration as **"aws"** type to expose AWS service actions.

> Observe that each `uri` parameter inside every `x-amazon-apigateway-integration` object points to a specific S3 resource (`demo-bucket-s3-proxy-<AWS-ACCOUNT-ID>`). It means that every request made to our API will be executed against the `demo-bucket-s3-proxy-<AWS-ACCOUNT-ID>` bucket, so that this will be always implicit in each request)

Deploy the project

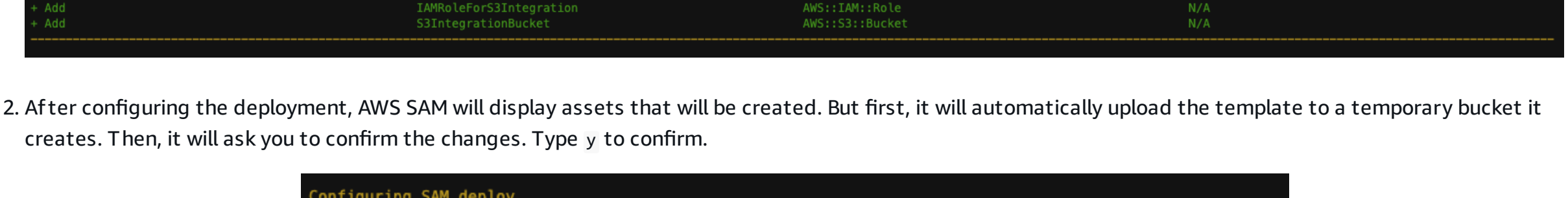
1. To initiate the deployment of this module, navigate to the `module-3/s3` directory, where the `template.yaml` file is located and then run the subsequent command:

```
1  sam build && sam deploy --capabilities CAPABILITY_NAMED_IAM --guided
```

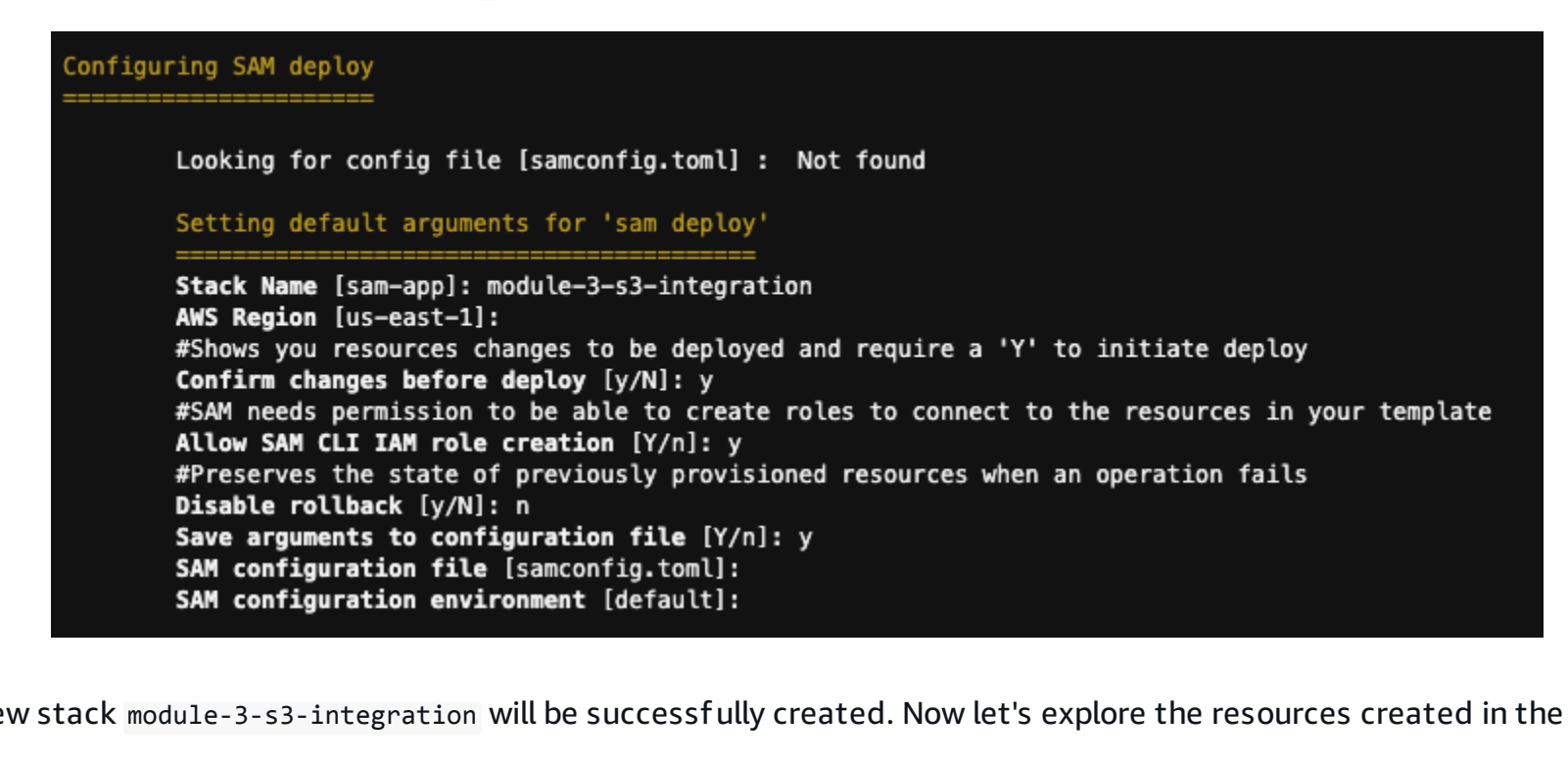
> The --capabilities parameter is required for operations that use Infrastructure as Code (e.g AWS SAM and AWS Cloudformation) to create IAM named resources. As we are creating a new IAM named role, it is necessary to specify it.

The first time that you run the `sam build && sam deploy --capabilities CAPABILITY_NAMED_IAM --guided` command, AWS SAM starts an AWS CloudFormation deployment. In this case, you need to specify what are the configurations that you want SAM to have in order to get the guided deployment. Please configure it as it is above.

- **Stack Name**: `module-3-s3-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



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.



3. After it was finished, a new stack `module-3-s3-integration` will be successfully created. Now let's explore the resources created in the console and test the API.

Test Integration

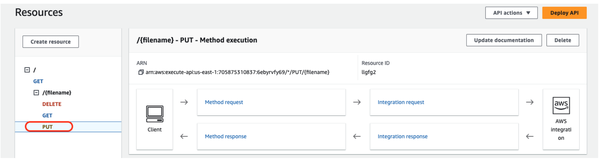
After you create your API Gateway REST API with S3 integration, you can test the API.

Test S3 Integration using API Gateway console

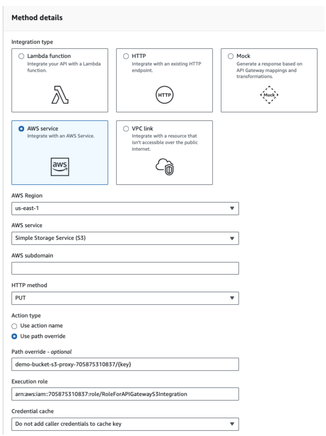
1. Open the Amazon API Gateway console and sign in. Make sure you are in the AWS region where the stack was deployed. If needed, change the AWS region using the region selector in the upper-right corner of the page.
2. Choose your REST API named `Api-module-3-s3-integration`.

Since our S3 bucket is empty, the first thing we are going to test the upload of a new file to it.

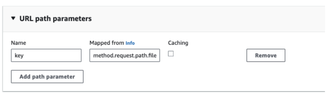
3. In the **Resources** pane, you can select the method you want to test. Click on the `PUT` method that is under `/filename`.



4. Using the tabs, navigate to **integration request** and click **Edit** to review some important configuration of our API.



5. Below at the same page, expand the **URL path parameters** session. Observe the mapping between the parameter **Key** and its correspondence to the path attribute named **filename**, that is represented as `method.request.path.filename`. This situation is defined as **Path Override**.

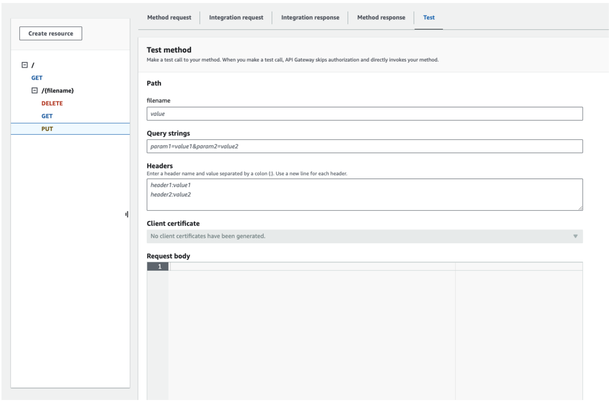


When using API Gateway integration with S3, API Gateway expects to find important parameters inside our request. Since we are defining a **PUT** command to create a new object inside our S3 bucket, it is necessary to define the new object's Key, and API Gateway will try to extract a parameter named **key** from the **PUT** request. Using this configuration for Path Override, we are just telling API Gateway that it can extract the **filename** parameter from our request and trust it to use as the necessary **Key** name.

6. After reviewing the integration request configuration, just click the **Cancel** button.

Now we are going to test our **PUT** method by writing a sample file to the S3 bucket.

7. Still under the **PUT** method, use the tabs to navigate to **Test**.



We are expected to provide the `filename` attribute under the **Path** session. It will be used for Path Overriding, so that API Gateway will use this information as the S3 object **key**.

8. Using the **filename** field, enter `sample.txt`. Below, under the **Request body** field, enter `"Hi! I am a sample test file"` and click **Test**. Looking to the response, observe the 200 status code, representing success when sending the object to S3.

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, the method call is successful, the API's resources will be deleted. If you want to, feel free to navigate to the S3 bucket to see our newly created object. You can find it under the "Resources" tab of the "module-3-s3-integration" cloudformation stack.

Now we are going to test our **GET** method to extract information about specific objects.

9. Looking at the resources panel, under **/filename** select the **GET** method and navigate to the **Test** tab. For **filename**, enter `my-nice-text-file.txt` and hit the "Test" button.

Note the response indicating that the given key does not represent an existing file, since this file was not created.

10. Now try it again, updating the **filename** field. Replace `my-nice-text-file.txt` with `sample.txt` and click **Test**. Now it should return the content from the **sample.txt** file.

Challenge:
Test the **get** method under the root path. Note that it does not require any filename parameter, because we are listing all the objects inside the S3 bucket. Also, try to test the **delete** method by deleting our **sample.txt**. Then, test the **get** methods again and check their returns

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 PUT \
  'https://<api-id>.execute-api.<region>.amazonaws.com/dev/sample-2.txt' \
  -H 'Content-Type: application/json' \
  -d 'One more test file. This time, we are using cURL to test the integration'
```

This code will create a new file inside our S3 bucket. Note that the file name is specified inside the URL path */sample-2.txt* and the file content is sent as the payload data. Now let's check the content of our file's by requesting it using cURL

3. Using your terminal, run the following command:

```
curl -X GET \
  'https://<api-id>.execute-api.<region>.amazonaws.com/dev/sample-2.txt' \
  -H 'Content-Type: application/json'
```

Congratulations! You have successfully completed this module.

Warning! Prior to move to the cleanup session, you must delete all the files inside your `demo-bucket-s3-proxy-<ACCOUNT-ID>` bucket. Otherwise, Cloudformation will prevent it