The PATCH method under the resource order\_id will update an order. We will not be using Lambda proxy integration here as well.

The lambda function associated with the method will check if the order\_id to be updated is a valid order by looking up on the dynamodb orders table. If not, the update request will be rejected. If yes, the updated order details are captured in the API Gateway and published into the Kinesis data stream so that the record flows through the entire pipeline and reaches Redshift. There is no point in updating the DynamoDB table alone here.

*Method Request:*

We set the API Key Required to true and use the ordesPostModel that was used in the POST request to validate the request body. Note that, in the PATCH request as well the request body is going to be the same as in POST request. The order\_id which is the additional detail needed in the PATCH request is a part of the main URL.

*Integration Request:*

We are not using Lambda Proxy Integration here. We will be using the below mapping template to convert the API gateway request into format needed by lambda:

#set($inputRoot = $input.path('$'))

{

"order\_id" : "$input.params('order\_id')",

"input" : {

"customer\_id" : "$inputRoot.customer\_id",

"seller\_id" : "$inputRoot.seller\_id",

"products" : $inputRoot.products

}

}

The query string parameters, headers, path parameters are captured using input.params and the request body details are captured using inputRoot. We have used “input” as the key for the request body here. This can be anything like say “body” also. The lambda function has to use the same key to retrieve these attributes. Also, we need not always capture the request body under another key like “input” or “body”. It can be at the level of order\_id itself. Again, lambda that is invoked by the Api Gateway has to use the right key to access the attributes.

Also, note that I have not used double-quotes for $inputRoot.products. When I used double-quotes, I was getting “unable to parse json” error message.

*Integration Response:*

We use the below mapping template to customize the output message from lambda to API Gateway:

#set($inputRoot = $input.path('$'))

$input.json('$.body')

Under Header Mappings, we set the value of Access-Control-Allow-Origin to ‘\*’ since we have enabled CORS and *Method Response* expects a value for this.

*Lambda function invoked by the PATCH method:*

The lambda function, when it gets a message with valid structure, publishes the data into Kinesis data stream. Remember, messages with invalid structure, are rejected at the Method Request section of the API Gateway itself using Request Validator (body).

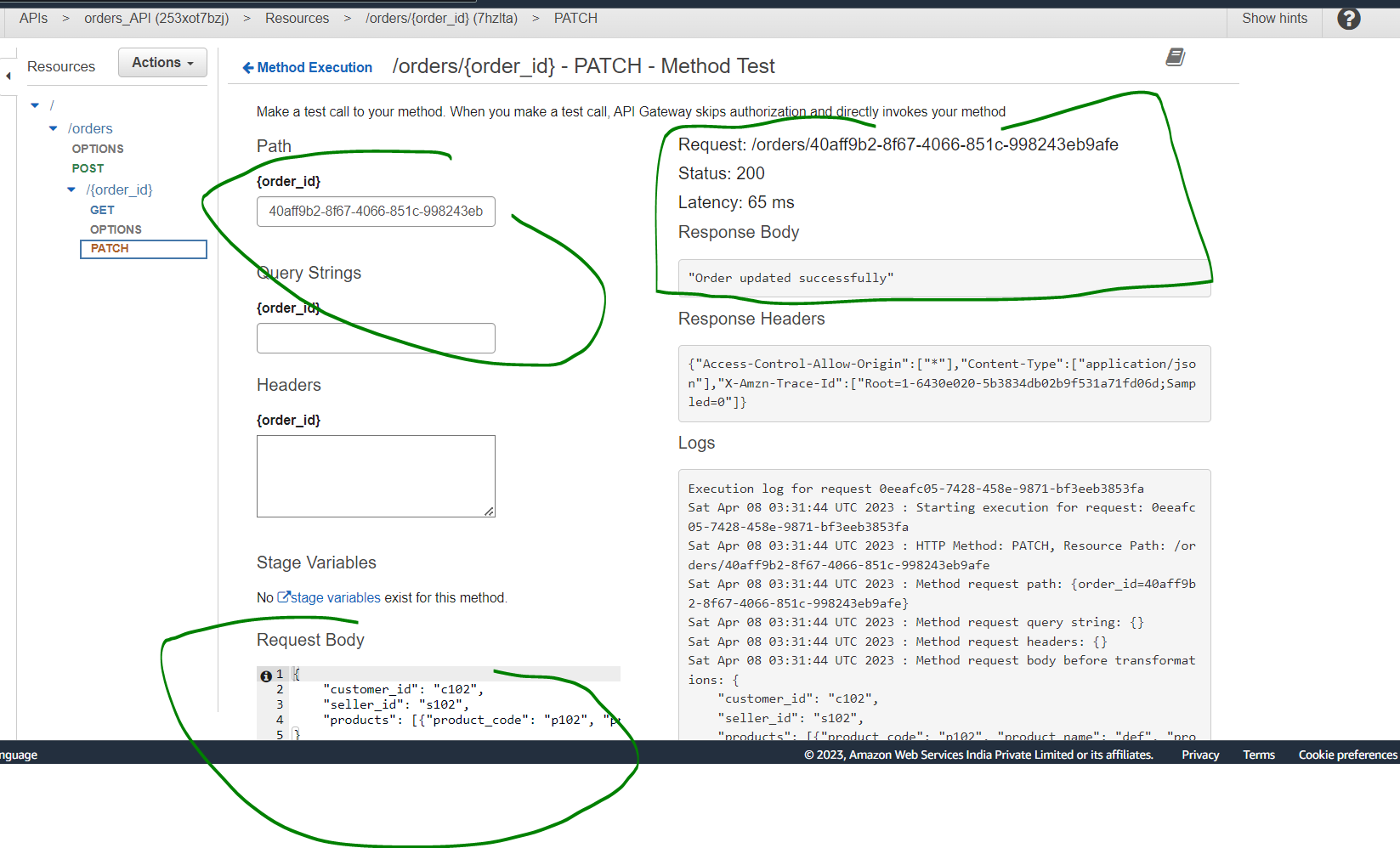
Refer the “Code” folder for the put\_record code.

**TESTING:**

*From within API Gateway:*

Verify you get a Success message from Lambda. Also, verify that the message has passed through the ETL pipeline -> S3 landing area, S3 staging area and Redshift. In Redshift, you have Upsert logic. So a new record should not be inserted. Existing record should be updated with changed values. dwh\_update\_timestamp should have a value now. For new records, this column will be blank / NULL. The dynamodb orders table must also have the updated attribute values for the order.

Note that the put\_item method is able to update the dynamodb record as well automatically (if an order\_id is found) and there is no need to write a separate update\_item code.



*From POSTMAN:*

