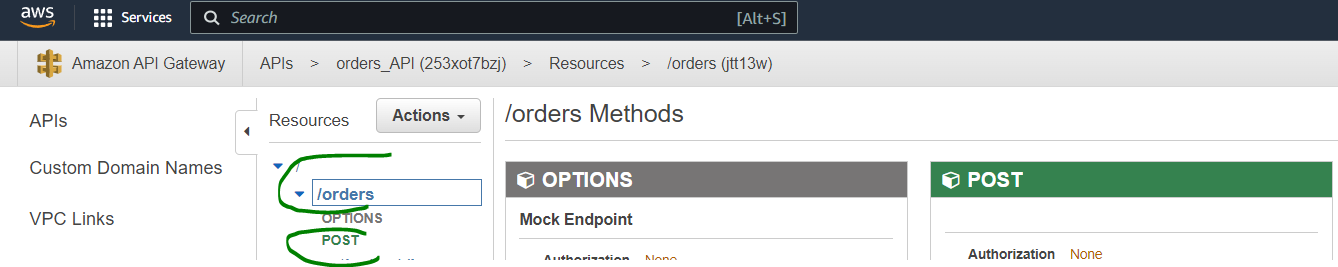
In Api Gateway, created a Resource -> orders.

Under this resource, created the POST method.

This will publish a new order into the Kinesis data stream and the order should flow through the ETL pipeline – S3 landing area, S3 staging area and Redshift.

We are using Lambda proxy integration, meaning we are not going to use Integration Request / Integration Response to modify the data going into or coming out of the Lambda function.



In the *Method Request*, we will be using the below Parameters:

API Key Required -> true

This helps us to control who uses our API and also how much they use. More of this covered later.

Request Validator -> body

This way we reject invalid messages upfront before even passing it on to the Lambda function.

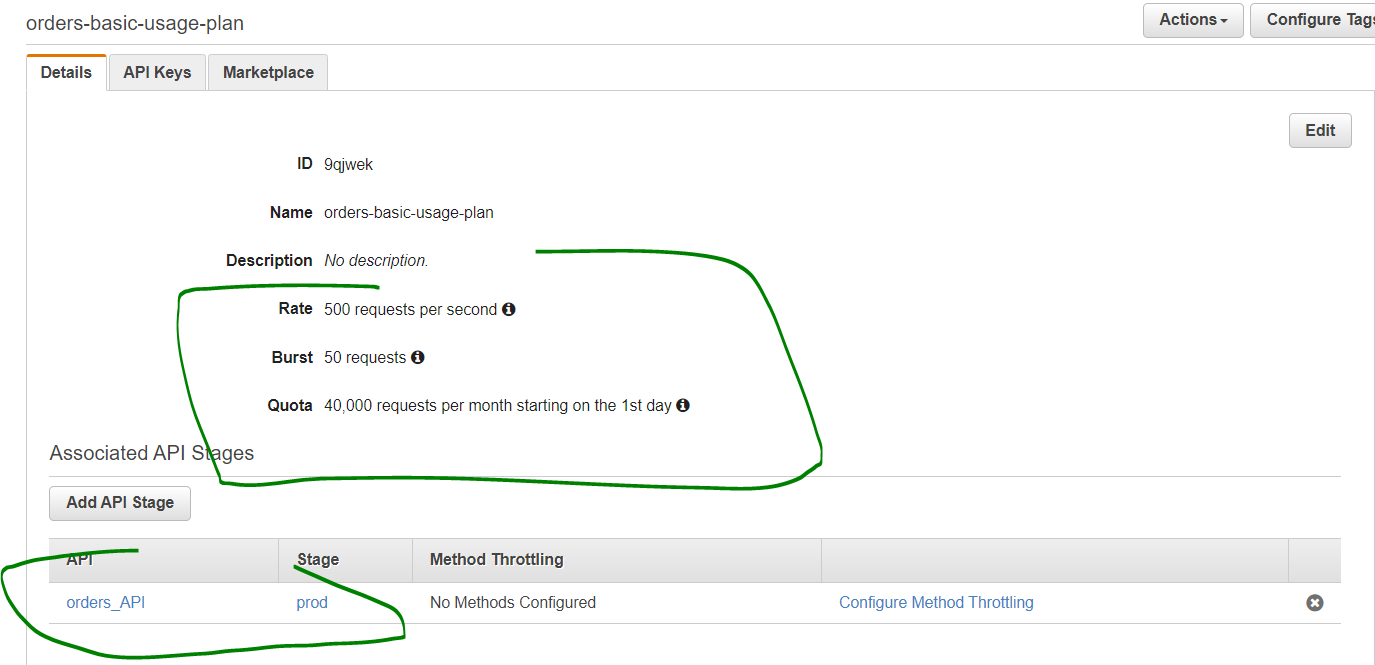
A model is created and this is used in the Request Body section.



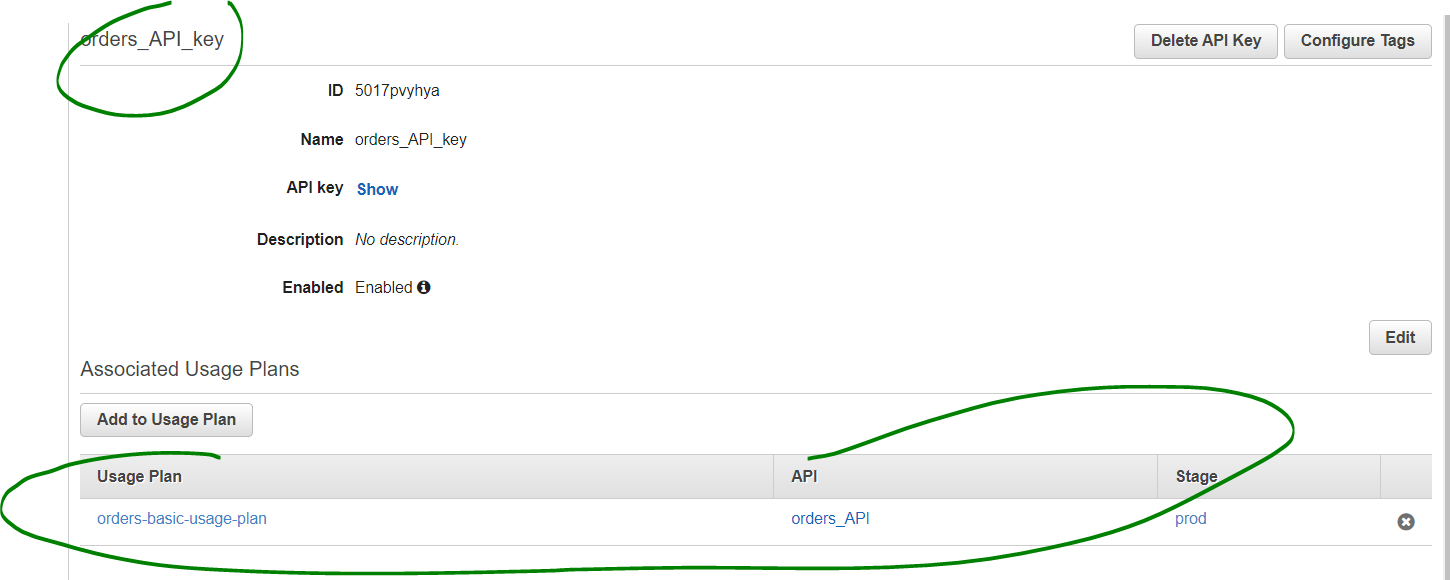
Now, if we pass a message with one of the “required” attributes missing or a different data type, the message will be rejected right away before being sent to Lambda.

*Usage Plan and API Key:*

Create an usage plan like below which helps to limit API usage and associate this with our API / deployment stage -> a stage gets created when you deploy an API.



Then create an API Key and associate this with the usage plan. Since the usage plan is associated with out API, the API Key gets associated with our API. Any one trying to access our API must pass the API Key as well now. This way we get to control who accesses our API.



*CORS:*

Normally APIs cannot be accessed from within another application / website unless the API is explicitly configured to allow access. This is where CORS comes in.

For every method within the API, click on the method -> Actions -> Enable CORS. When you do this, in the “Method Response” section, a header Access-Control-Allow-Origin will be added. In the lambda function, as part of the return object, include this header and its value also. The value is \*, if you want the API to be accessible from anywhere.

*Lambda function invoked by the POST 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:*

Pass a message like this:

{

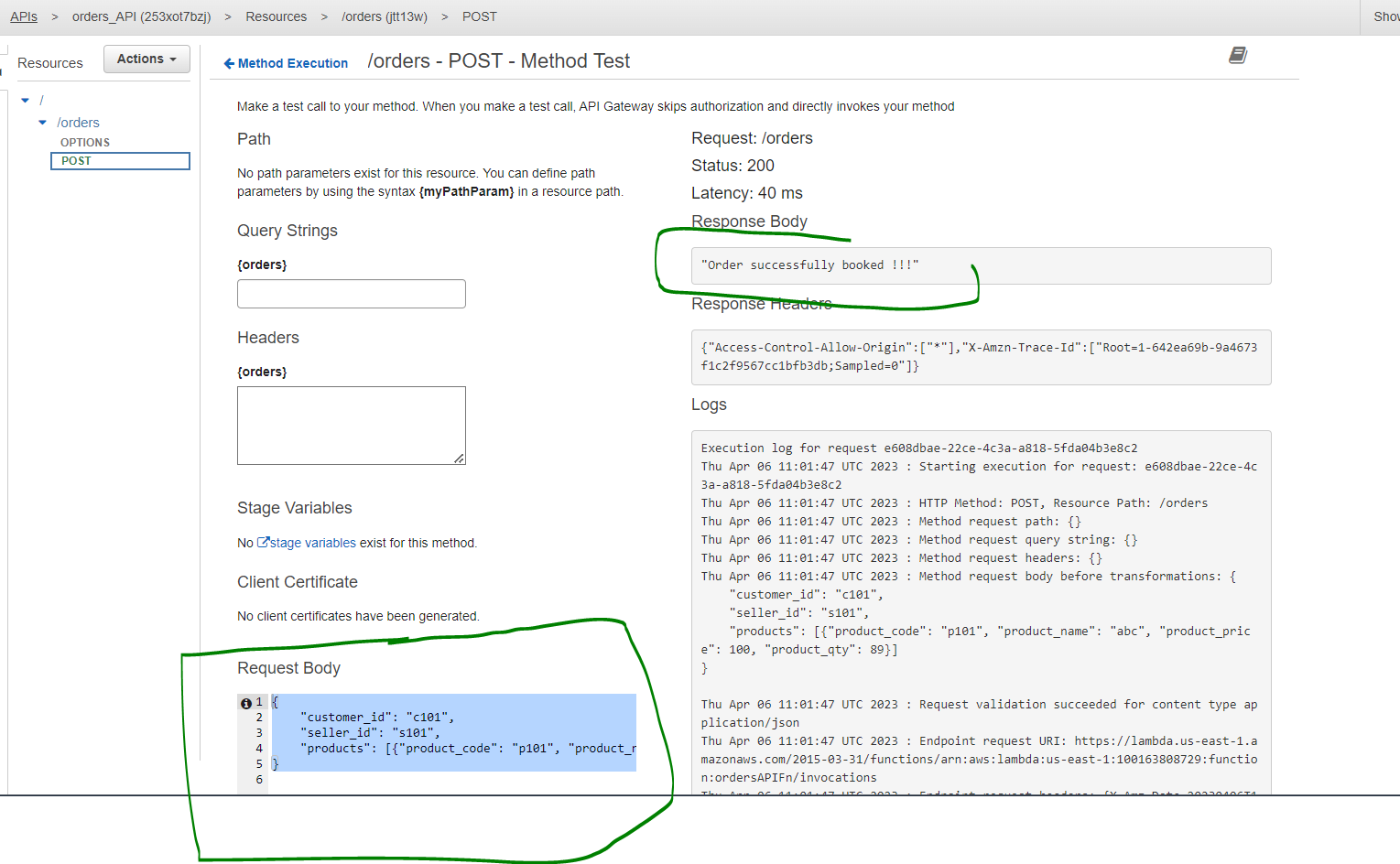
"customer\_id": "c101",

"seller\_id": "s101",

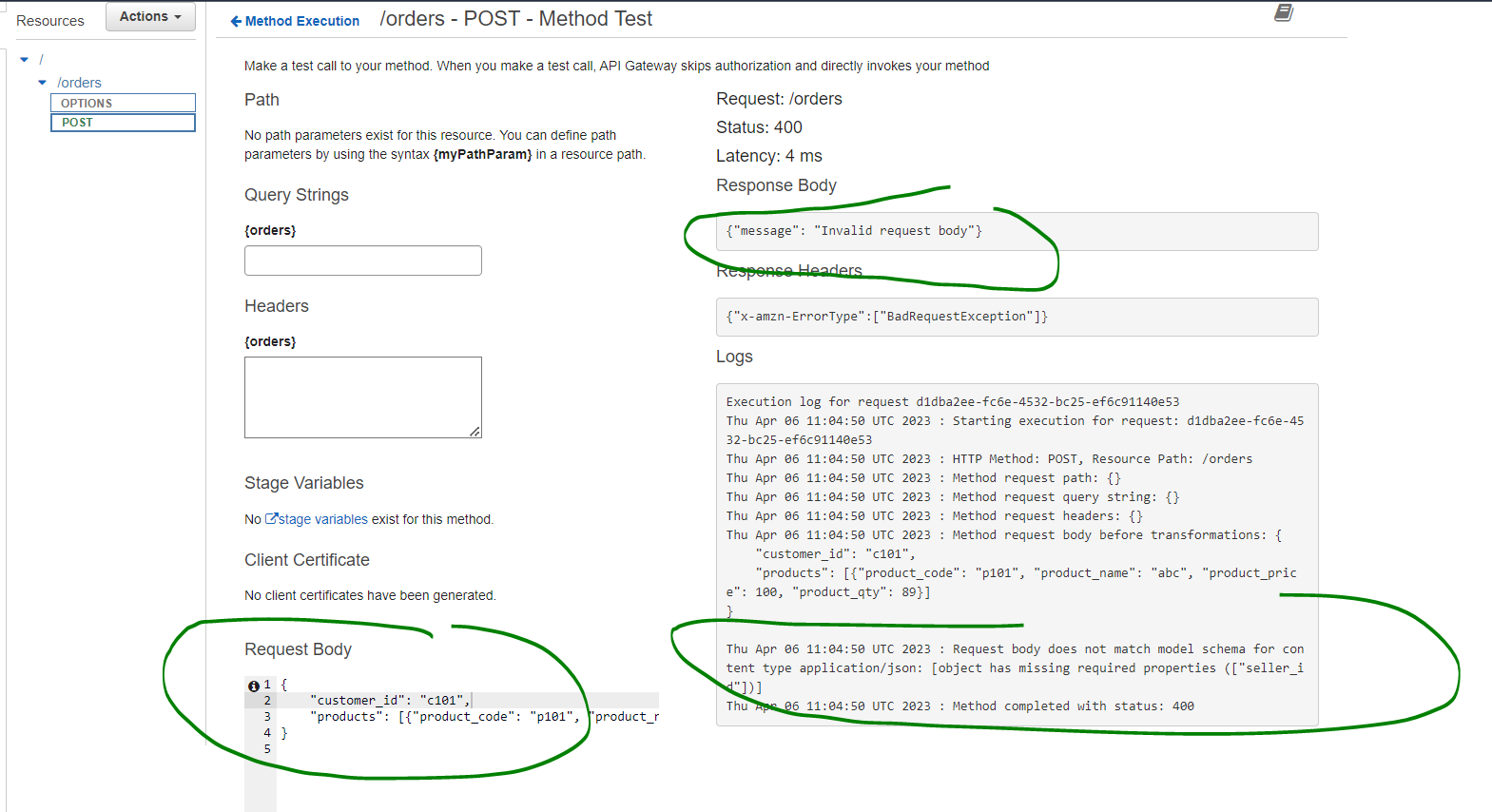
"products": [{"product\_code": "p101", "product\_name": "abc", "product\_price": 100, "product\_qty": 89}]

}

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.

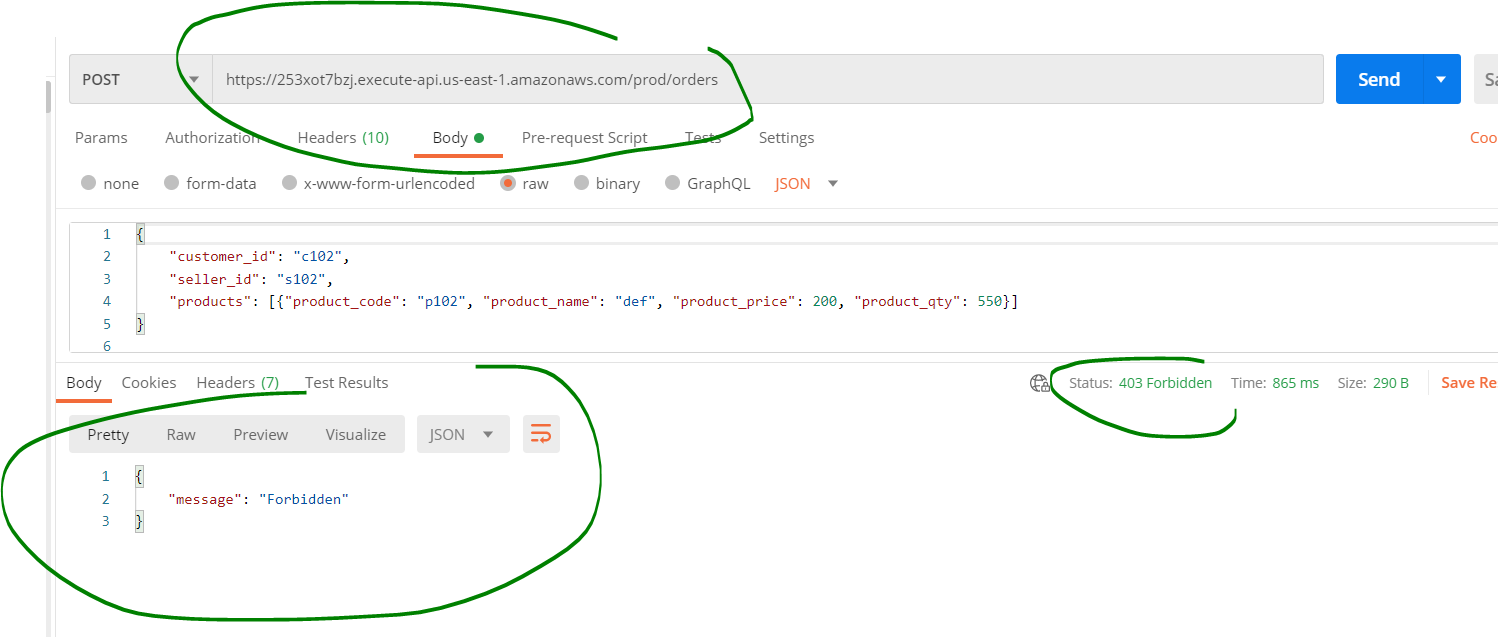


Pass an incorrect message and verify it is getting rejected:



*From outside API Gateway -> POSTMAN:*

Deploy the API. Copy the Invoke URL and put it in POSTMAN. Change the method to POST. Now if you do not pass the API Key you will get an error like below:



Now, pass the API Key in the Headers section.

Verify that you are getting a Success message and also verify the message is going through the ETL Pipeline:

