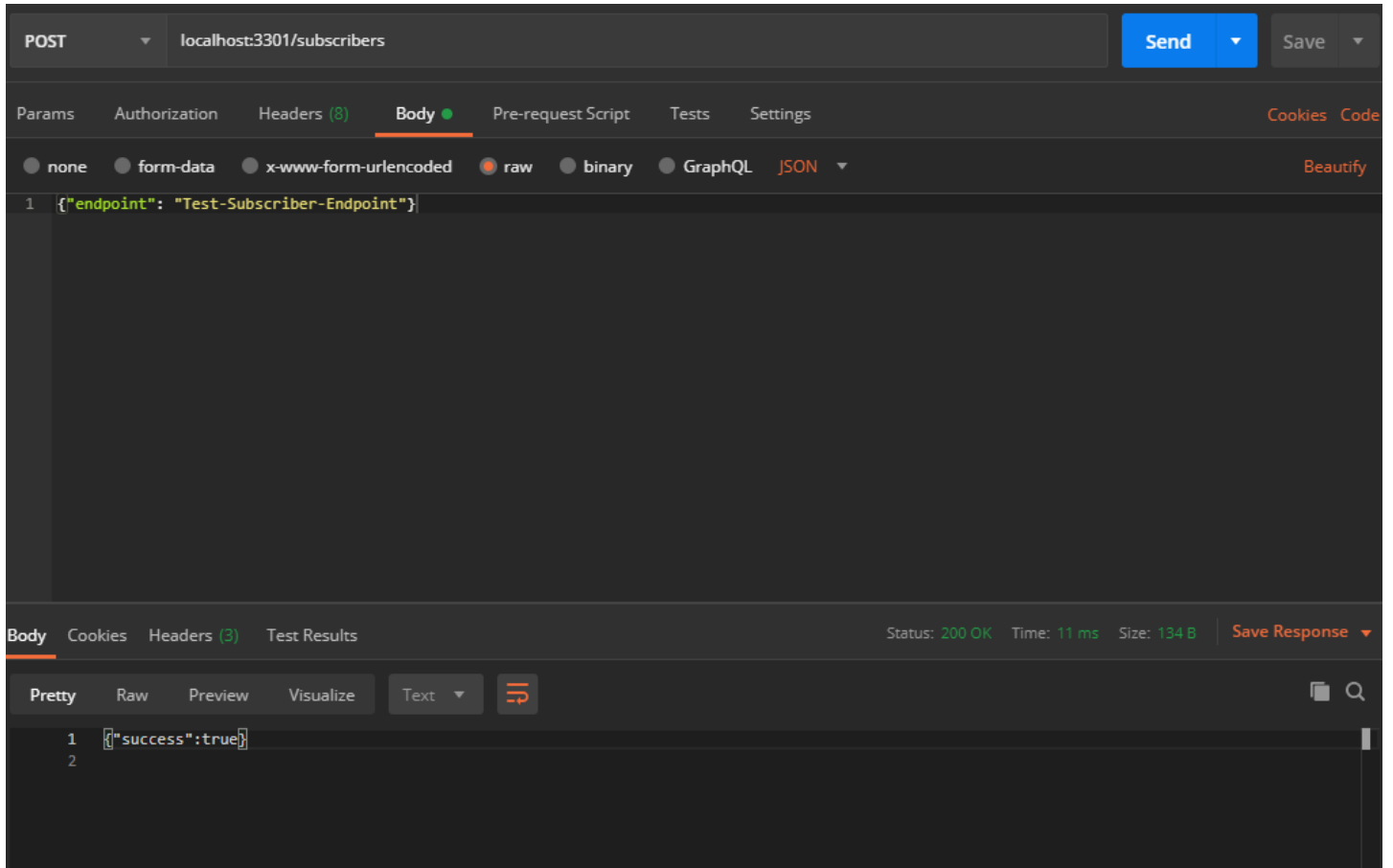
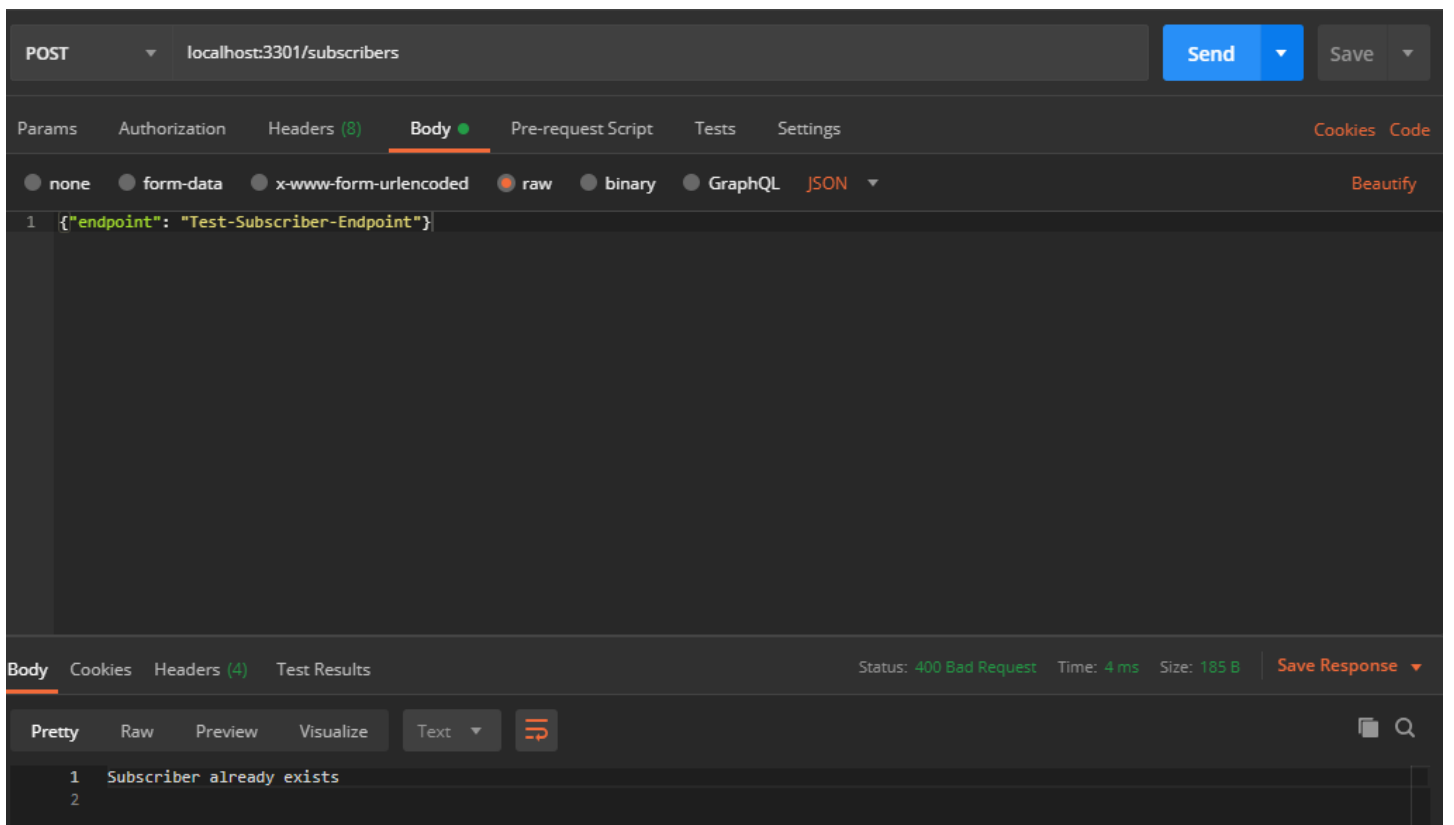


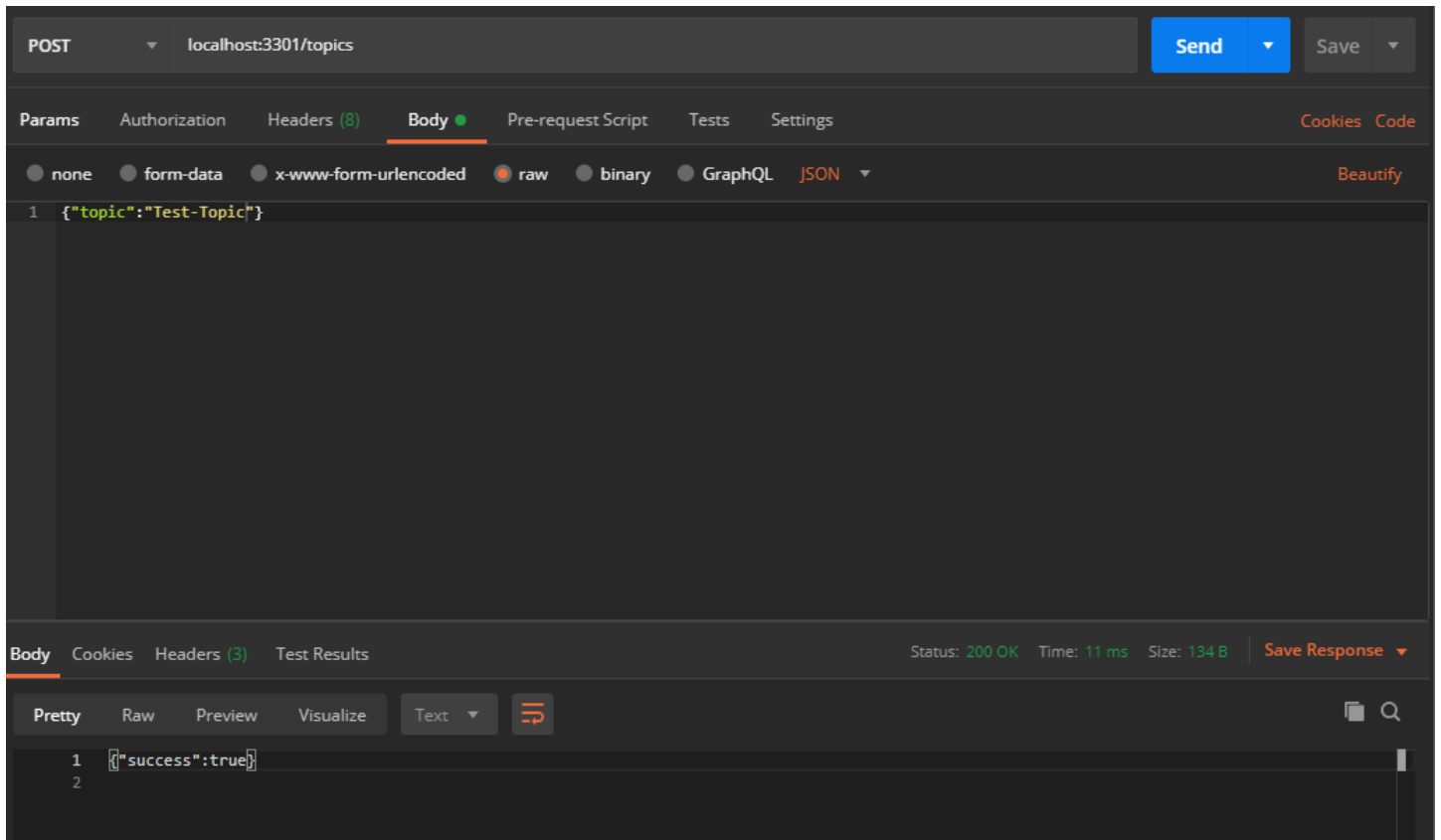
1) Create subscriber



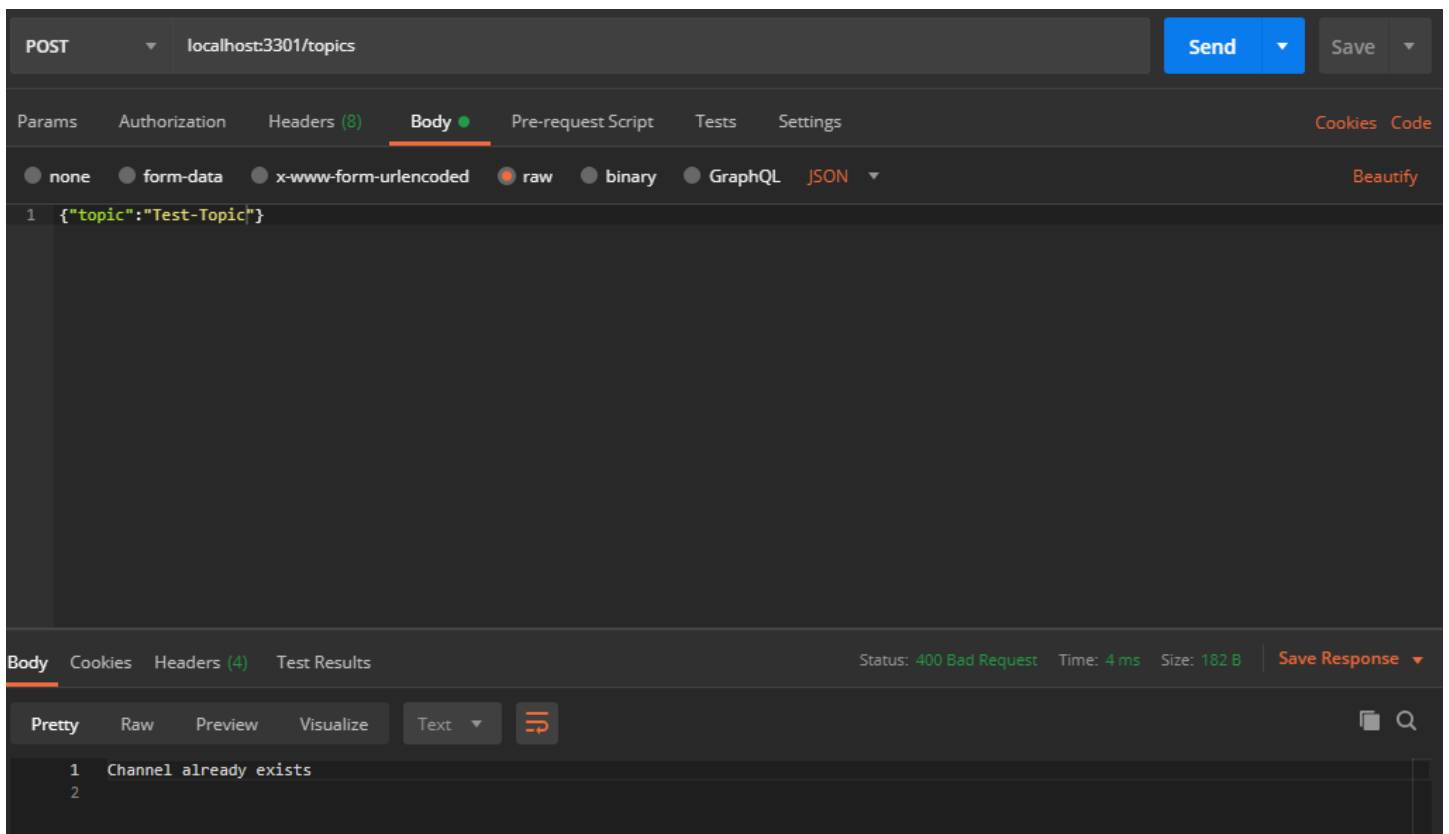
2) Create duplicate subscriber



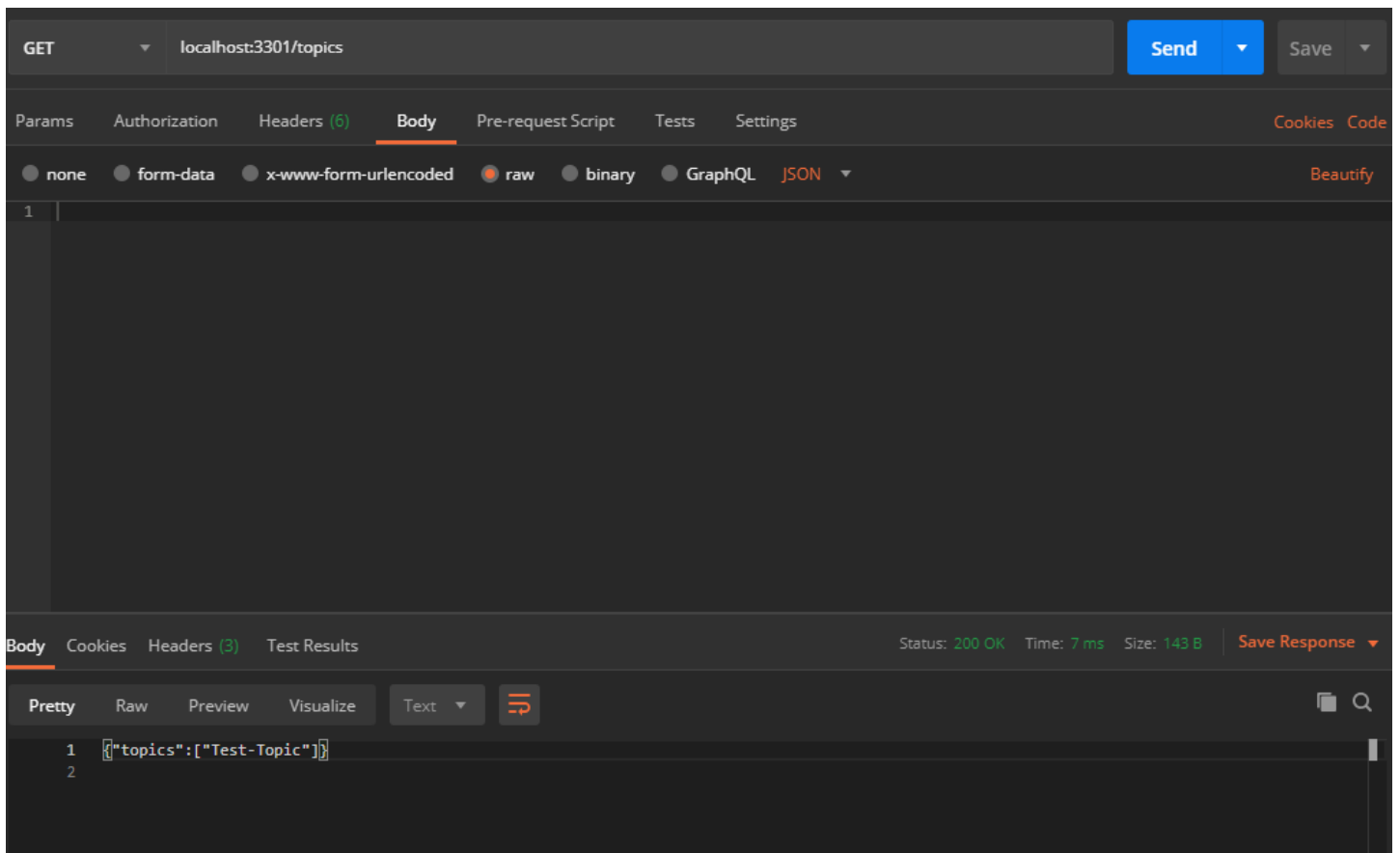
3) Create topic



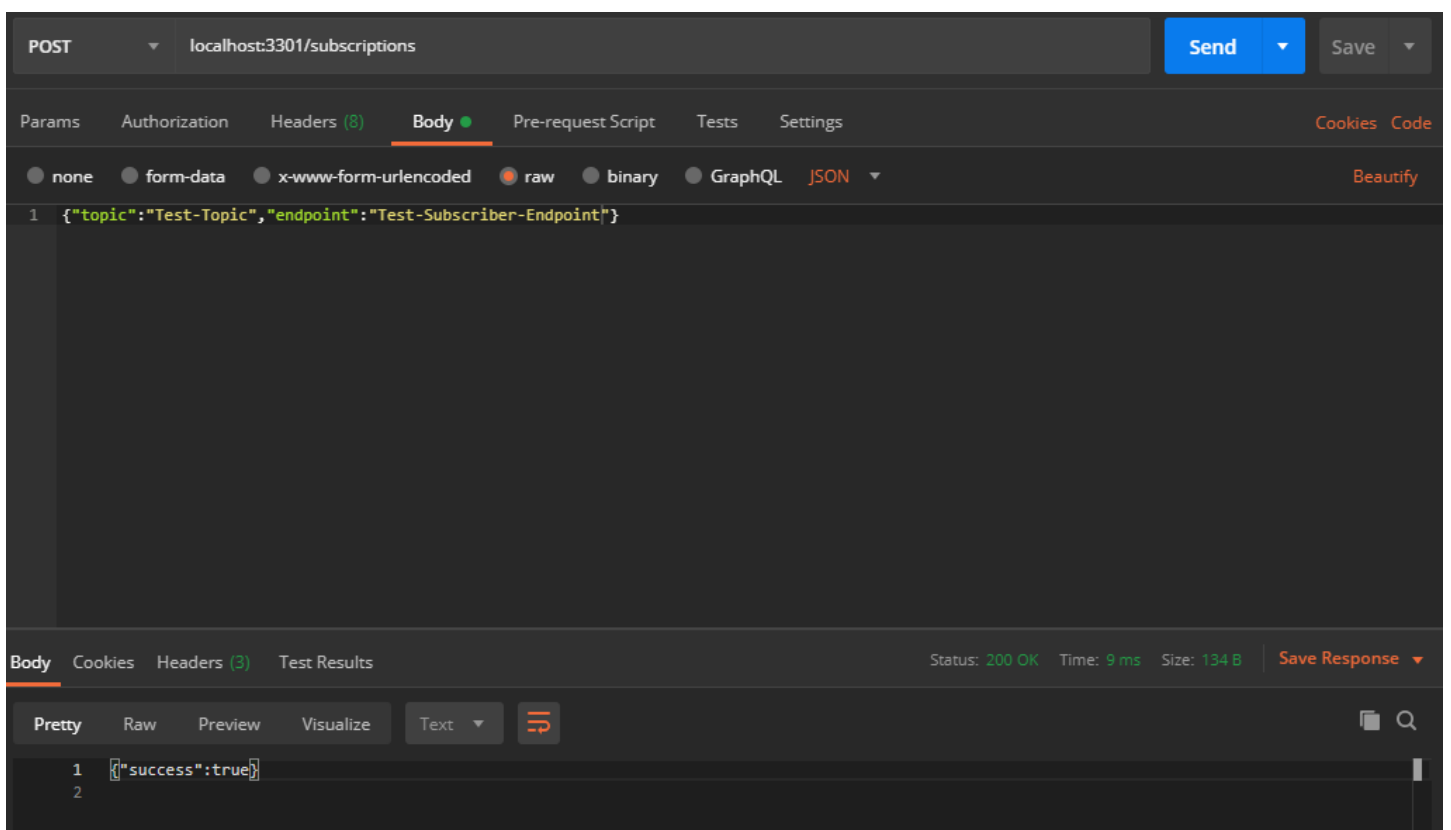
4) Create duplicate topic



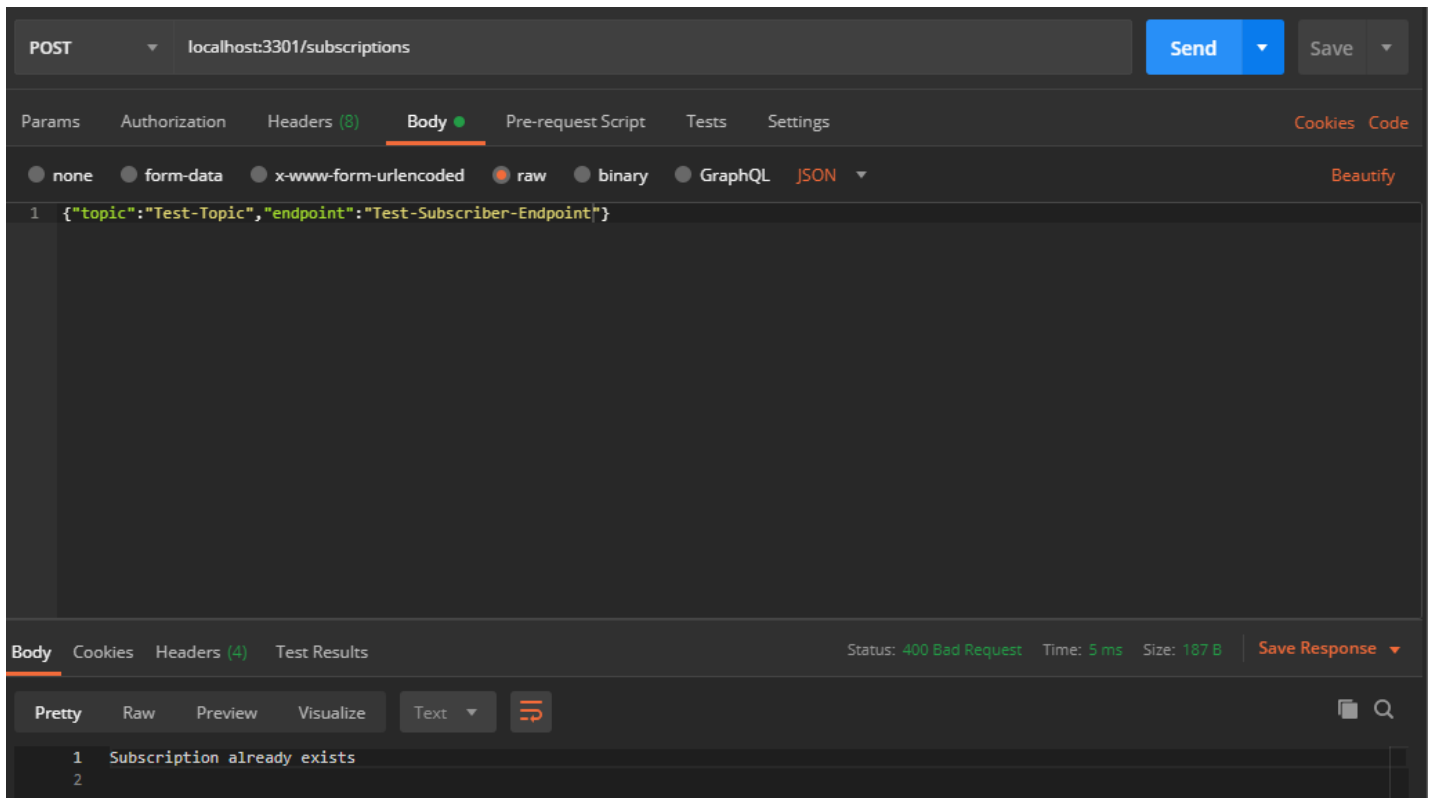
5) List topics



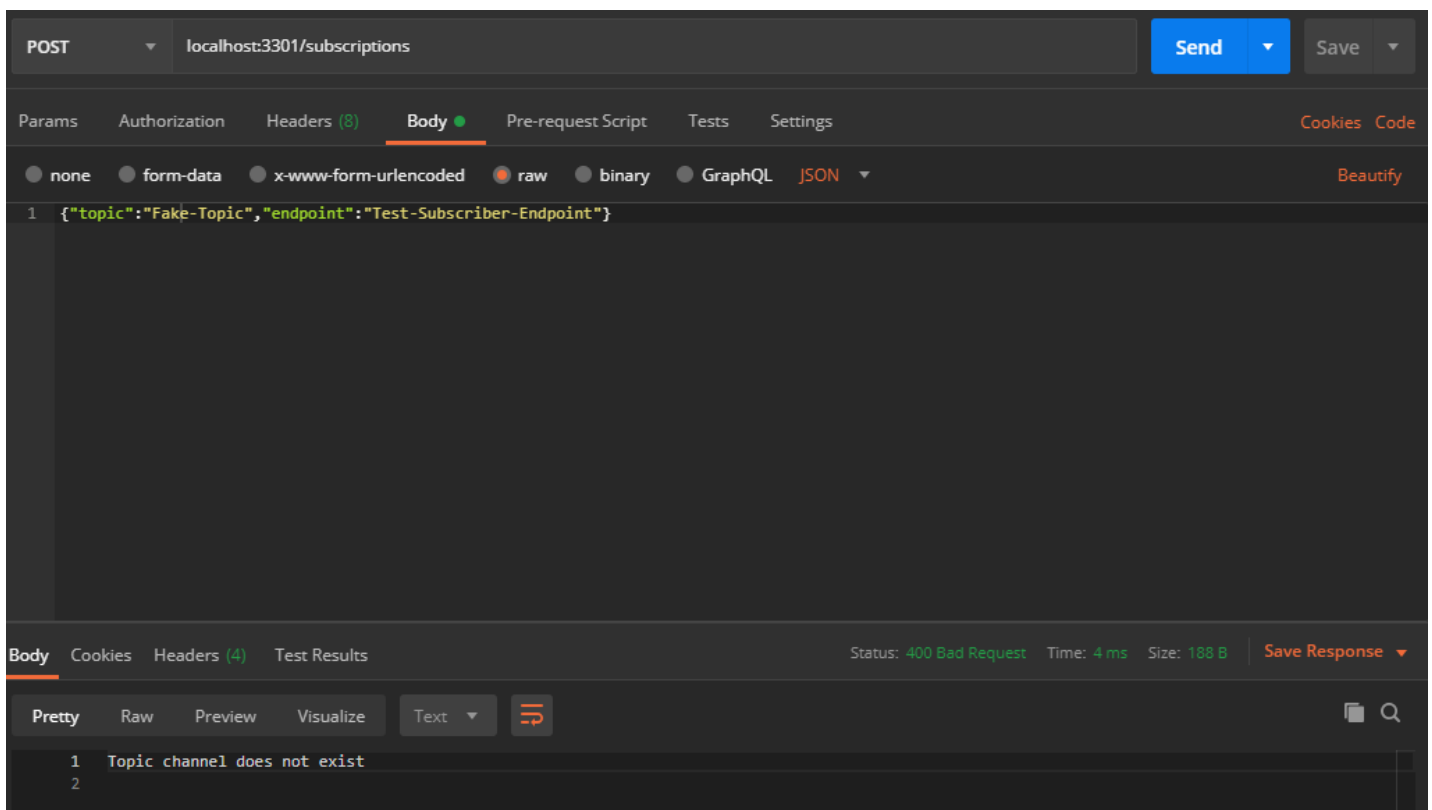
6) Subscribe subscriber to topic



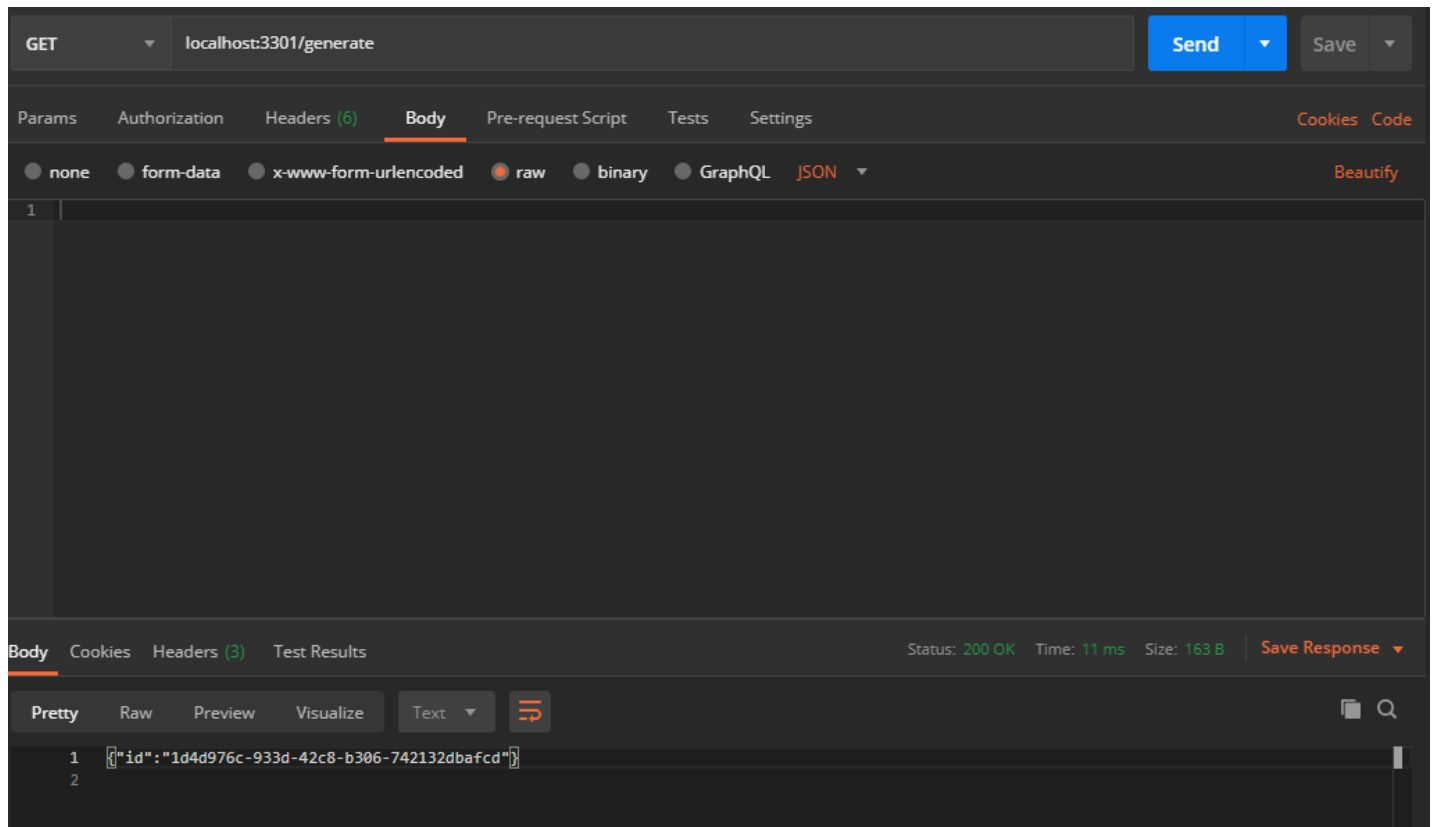
7) Attempt repeat subscription



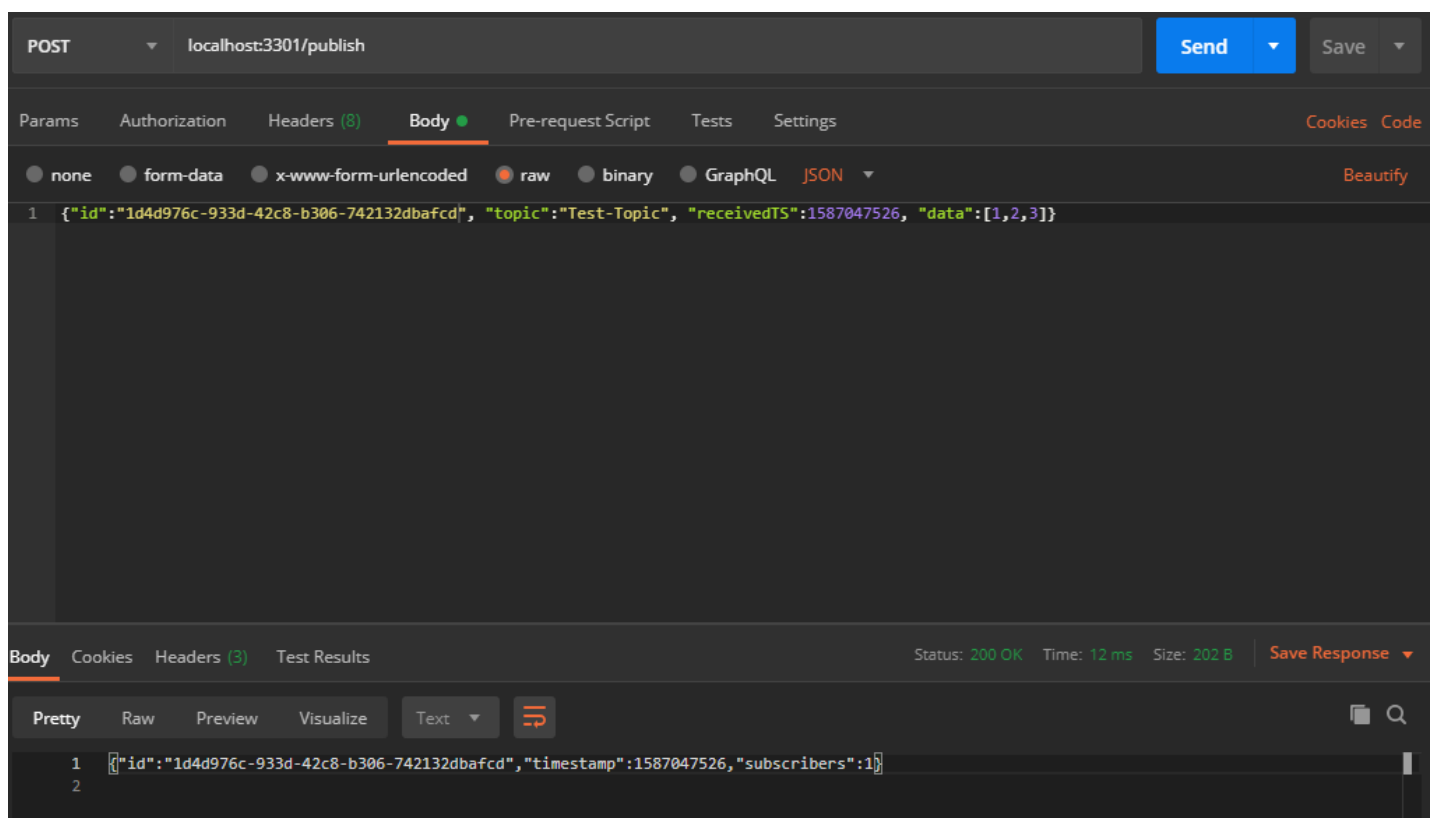
8) Attempt incorrect subscription



9) Generate UUID



10) Create message



11) Attempt duplicate message creation

The screenshot shows a Postman interface for a POST request to `localhost:3301/publish`. The request body is a JSON object: `{"id":"1d4d976c-933d-42c8-b306-742132dbafcd", "topic":"Test-Topic", "receivedTS":1587047526, "data":[1,2,3]}`. The status bar indicates a `400 Bad Request` with a time of `4 ms` and size of `208 B`. The response body, viewed in 'Text' format, contains the message: `1 Duplicate message. UUID already exists in system`.

12) Create message with given timestamp

The screenshot shows a Postman interface for a POST request to `localhost:3301/publish`. The request body is a JSON object: `{"topic":"Test-Topic", "receivedTS":1000000000, "data":[0,0,0]}`. The status bar indicates a `200 OK` with a time of `12 ms` and size of `202 B`. The response body, viewed in 'Text' format, contains the message: `1 {"id":"041ea8fa-f03c-45bb-87d9-7dac35456702","timestamp":1000000000,"subscribers":1}`.

13) Restart system

The screenshot shows a REST client interface with a POST request to `localhost:3301/publish`. The request body is a JSON object: `{"topic": "Test-Topic", "receivedTS": 1000000000, "data": [0, 0, 0]}`. The response tab shows an error: "Could not get any response". Below the error, a message states: "There was an error connecting to <http://localhost:3301/publish>". A section titled "Why this might have happened:" lists several potential causes:

- The server couldn't send a response: Ensure that the backend is working properly
- Self-signed SSL certificates are being blocked: Fix this by turning off 'SSL certificate verification' in *Settings > General*
- Proxy configured incorrectly: Ensure that proxy is configured correctly in *Settings > Proxy*
- Request timeout: Change request timeout in *Settings > General*

14) Pull messages

The screenshot shows a REST client interface with a GET request to `localhost:3301/pull`. The request body is a JSON object: `{"endpoint": "Test-Subscriber-Endpoint"}`. The response is a 200 OK status with a response time of 14 ms and a size of 333 B. The response body is a JSON array of messages:

```
[{"messages": [{"id": "1d4d976c-933d-42c8-b306-742132dbafcd", "attributes": [], "receivedTS": 1587047526, "data": [1, 2, 3]}, {"id": "041ea8fa-f03c-45bb-87d9-7dac35456702", "attributes": [], "receivedTS": 1000000000, "data": [0, 0, 0]}]}
```

15) Acknowledge message

POST localhost:3301/ack

Send Save

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies Code

none form-data x-www-form-urlencoded **raw** binary GraphQL JSON Beautify

```
1 {"id":"1d4d976c-933d-42c8-b306-742132dbafcd","endpoint":"Test-Subscriber-Endpoint"}
```

Body Cookies Headers (3) Test Results Status: 200 OK Time: 17 ms Size: 134 B Save Response

Pretty Raw Preview Visualize Text

```
1 {"success":true}
2
```

16) Pull unacknowledged message

GET localhost:3301/pull

Send Save

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings Cookies Code

none form-data x-www-form-urlencoded **raw** binary GraphQL JSON Beautify

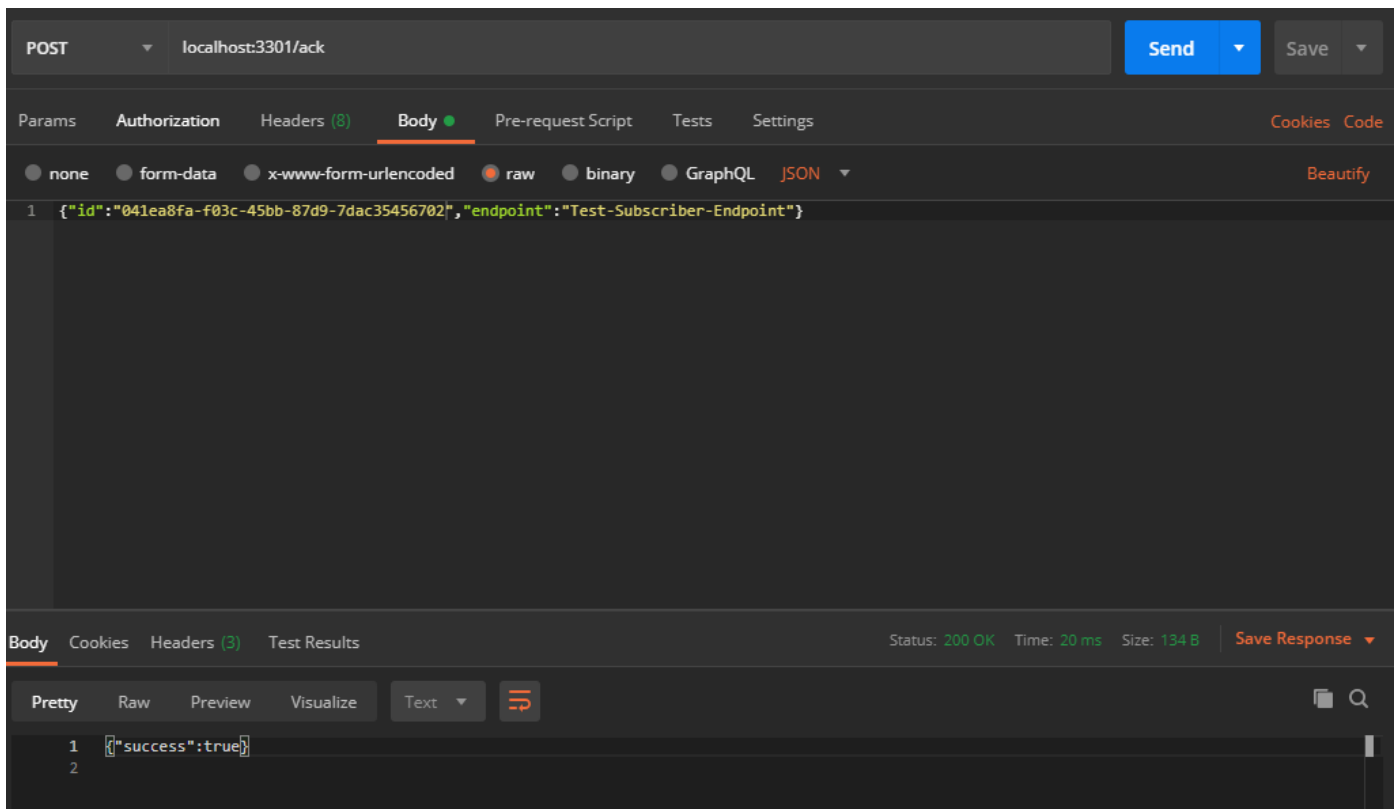
```
1 {"endpoint": "Test-Subscriber-Endpoint"}
```

Body Cookies Headers (3) Test Results Status: 200 OK Time: 5 ms Size: 233 B Save Response

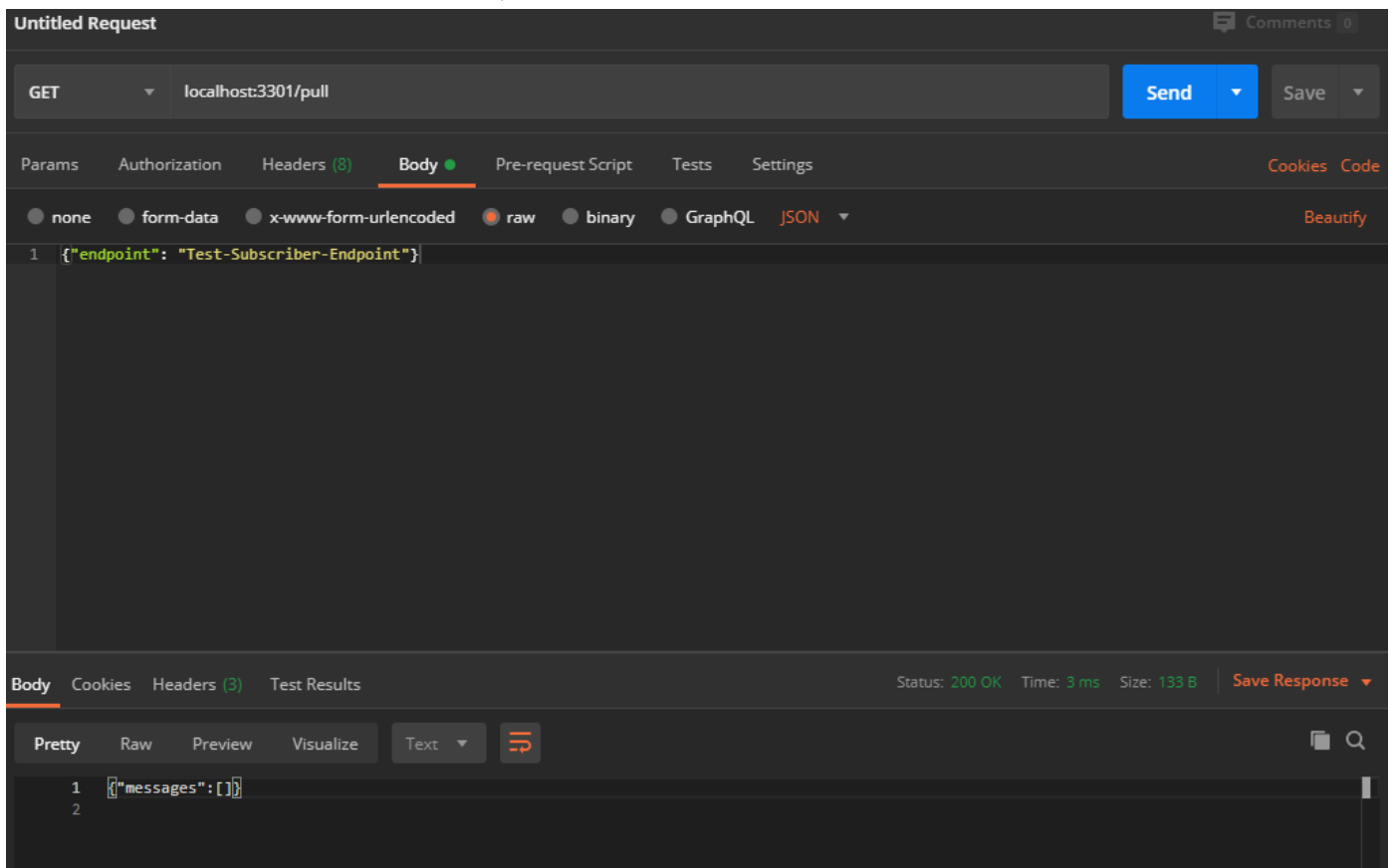
Pretty Raw Preview Visualize Text

```
1 {"messages":[{"id":"041ea8fa-f03c-45bb-87d9-7dac35456702","attibutes":[],"receivedTS":1000000000,"data":[0,0,0]}]}
2
```


17) Acknowledge second message



18) Pull messages



19) Delete topic

The screenshot shows a REST client interface with the following details:

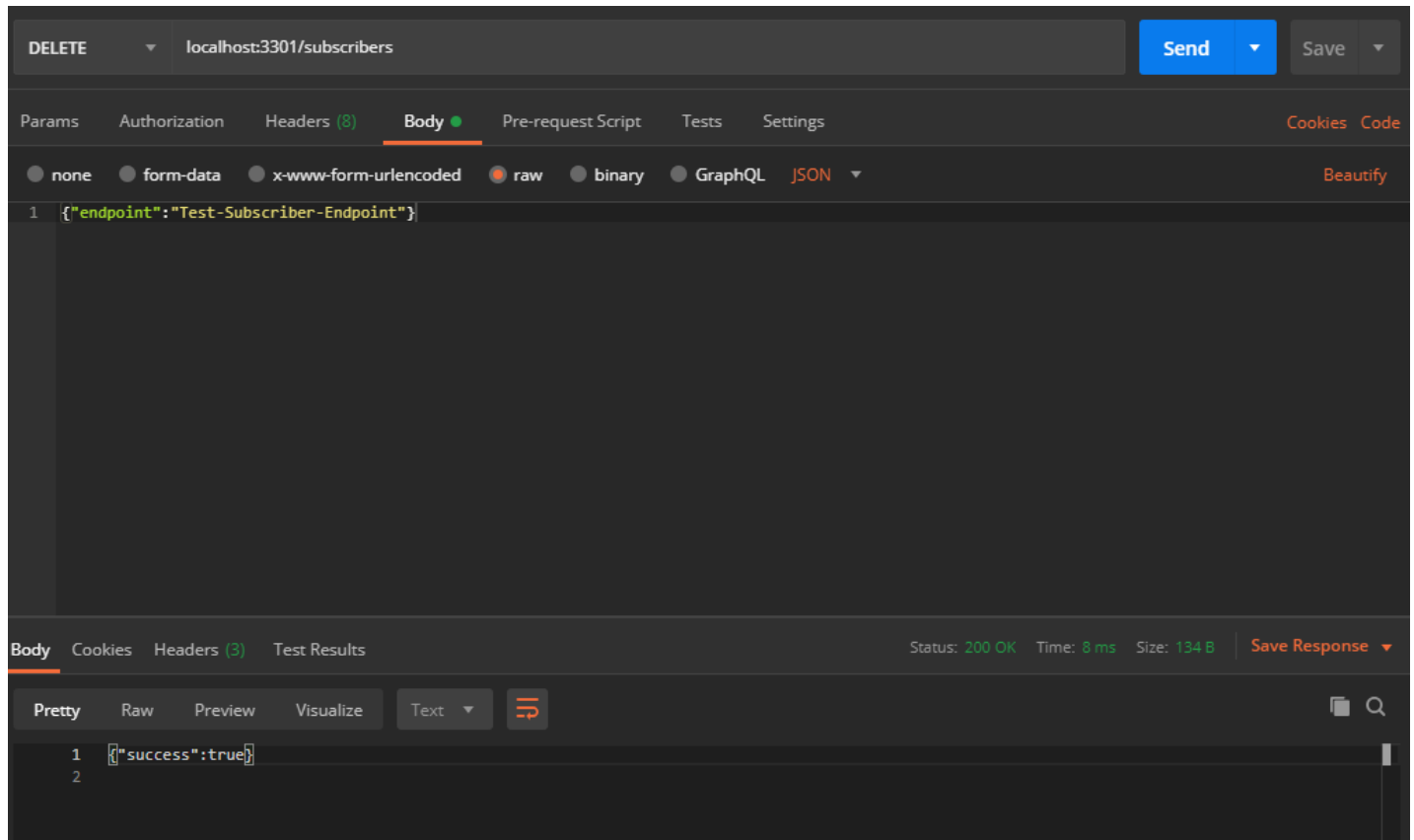
- Method:** DELETE
- URL:** localhost:3301/topics
- Body:** `{"topic": "Test-Topic"}`
- Status:** 200 OK
- Time:** 10 ms
- Size:** 134 B
- Response Body:** `{"success": true}`

20) Delete non-existent topic

The screenshot shows a REST client interface with the following details:

- Method:** DELETE
- URL:** localhost:3301/topics
- Body:** `{"topic": "Test-Topic"}`
- Status:** 400 Bad Request
- Time:** 3 ms
- Size:** 182 B
- Response Body:** `Channel does not exist`

21) Delete subscriber



22) Delete non-existent subscriber

