

Per Microservices Tests

1. Catalog

- a. Goal: The catalog service should initialize every product in its stock with 100 in quantity for each. Client should be able to query products of interest through REST GET requests.
Step: Query catalog service at Port 8082 with two products: barbie and Lego.
Expected: The query service should return product data for barbie and Lego with 100 quantity for each.

The screenshot displays two sequential REST client requests and their responses. The first request is a GET to `ec2-52-23-170-194.compute-1.amazonaws.com:8082/query/barbie`, which returns a JSON object containing product information for Barbie, including a price of 15.99 and a quantity of 100. The second request is a GET to `ec2-52-23-170-194.compute-1.amazonaws.com:8082/query/lego`, returning a similar JSON object for Lego with a price of 21.99 and a quantity of 100. Both requests resulted in a 200 OK status.

Request 1: GET /query/barbie

KEY	VALUE	DESCRIPTION
Key	Value	Description

Response 1 (JSON):

```
{  "data": {    "name": "Barbie",    "price": 15.99,    "quantity": 100  }}
```

Request 2: GET /query/lego

KEY	VALUE	DESCRIPTION
Key	Value	Description

Response 2 (JSON):

```
{  "data": {    "name": "Lego",    "price": 21.99,    "quantity": 100  }}
```

- b. Goal: The catalog service should return error message when a product that does not exists.
Step: Query catalog service at Port 8082 with nonexistence product: test.

Expected: Error message stating that product does not exist.

GET ec2-52-23-170-194.compute-1.amazonaws.com:8082/query/test

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "error": {
3     "code": 404,
4     "message": "Product Not Found"
5   }
6 }
```

- c. Goal: The catalog service should be able to process order requested by other services.
Step: Request order at catalog service at port 8082
Expected: An order should be successfully processed with order data processed, and order number should be -1 because order number is processed by order service

POST ec2-52-23-170-194.compute-1.amazonaws.com:8082/orders

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

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

```
1 {
2   "name": "barbie",
3   "quantity": 6
4 }
```

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "data": {
3     "number": -1,
4     "name": "Barbie",
5     "quantity": 6
6   }
7 }
```

- d. Goal: The catalog service should be able to respond error message when an product does not exist in stock
Step: Request order at catalog service at port 8082

Expected: An error message should be returned

POST ec2-52-23-170-194.compute-1.amazonaws.com:8082/orders

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

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

```
1 {
2   ... "name": "test",
3   ... "quantity": 6
4 }
```

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "error": {
3     "code": 404,
4     "message": "Product Not Found"
5   }
6 }
```

- e. Goal: The stock should be updated according to successful orders

Step: Query catalog service with purchased product: barbie

Expected: Product data with updated quantity that reflects prior purchase

GET ec2-52-23-170-194.compute-1.amazonaws.com:8082/query/barbie

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Query Params

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "data": {
3     "name": "Barbie",
4     "price": 15.99,
5     "quantity": 94
6   }
7 }
```

- f. Goal: The catalog service should report insufficient stock when requesting quantity is greater than stock, the same test also applies to stock that is 0
- Step: Request order with quantity greater than 94

Expected: Error message indicating that there is insufficient stocks.

POST ec2-52-23-170-194.compute-1.amazonaws.com:8082/orders

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

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

```
1 {
2   ... "name": "barbie",
3   ... "quantity": 100
4 }
```

Body Cookies Headers (5) Test Results Status: 200 OK

Pretty Raw Preview Visualize JSON

```
1 {
2   "error": {
3     "code": 406,
4     "message": "Product insufficient stock"
5   }
6 }
```

- g. Goal: catalog service should be able to restock commodity when quantity reaches zero
Step: Order a product so that its quantity becomes zero, query current stock, after 10 seconds, query the product

Expected: the product quantity should be 0 immediately after ordering and after 10 seconds the quantity has restocked to 100

POST ec2-52-23-170-194.compute-1.amazonaws.com:8082/orders

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

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

```
1 {
2   ... "name": "lego",
3   ... "quantity": 100
4 }
```

Body Cookies Headers (5) Test Results Status: 200

Pretty Raw Preview Visualize JSON

```
1 {
2   "data": {
3     "number": -1,
4     "name": "Lego",
5     "quantity": 100
6   }
7 }
```

GET

ec2-52-23-170-194.compute-1.amazonaws.com:8082/query/lego

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

Query Params

	KEY	VALUE	DESCRIPTION
	Key	Value	Description

BodyCookiesHeaders (5)Test Results

PrettyRawPreviewVisualizeJSON

```
1  {
2    "data": {
3      "name": "Lego",
4      "price": 21.99,
5      "quantity": 0
6    }
7  }
```

GET

ec2-52-23-170-194.compute-1.amazonaws.com:8082/query/lego

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

Query Params

	KEY	VALUE	DESCRIPTION
	Key	Value	Description

BodyCookiesHeaders (5)Test Results

PrettyRawPreviewVisualizeJSON

```
1  {
2    "data": {
3      "name": "Lego",
4      "price": 21.99,
5      "quantity": 100
6    }
7  }
```

2. Order

- The order service should be able to process order requests and return the order information

Send order request to order service at port 8083

Expecting order data successfully from response

The screenshot shows a REST client interface with a POST request to `ec2-54-210-155-177.compute-1.amazonaws.com:8085/orders`. The request body is a JSON object: `{ "name": "lego", "quantity": 100 }`. The response status is 200 OK. The response body is displayed in JSON format: `{ "data": { "number": 1, "name": "Lego", "quantity": 100 } }`.

- b. The order service should be able query a successful order placed before.
Send order query to order service at port 8083
Expecting previously placed order information

The screenshot shows a REST client interface with a GET request to `ec2-54-210-155-177.compute-1.amazonaws.com:8083/orders/1`. The request does not have a body. The response status is 200 OK. The response body is displayed in JSON format: `{ "data": { "number": 1, "name": "Lego", "quantity": 100 } }`.

- c. The order service should be able to replicate order information across three replicas
Send order query to other order services at port 8084 and 8085

Expecting same order information as before

GET

ec2-54-210-155-177.compute-1.amazonaws.com:8084/orders/1

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

This request does not have a body

BodyCookiesHeaders (5)Test Results

Status: 200 OK

PrettyRawPreviewVisualizeJSON

1

{

2

"data": {

3

"number": 1,

4

"name": "Lego",

5

"quantity": 100

6

}

7

}

GET

ec2-54-210-155-177.compute-1.amazonaws.com:8085/orders/1

ParamsAuthorizationHeaders (6)BodyPre-request ScriptTestsSettings

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

This request does not have a l

BodyCookiesHeaders (5)Test Results

PrettyRawPreviewVisualizeJSON

1

{

2

"data": {

3

"number": 1,

4

"name": "Lego",

5

"quantity": 100

6

}

7

}