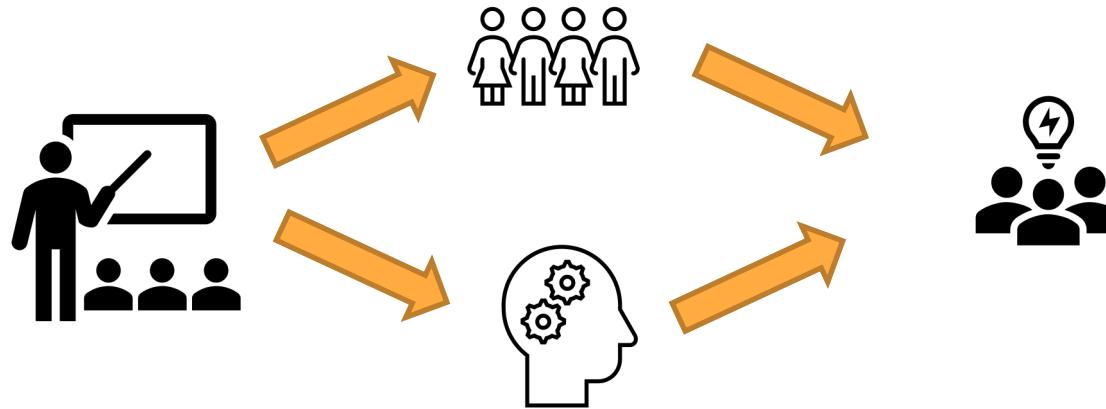


Arquitetando e testando aplicações serverless na AWS

Isabel Mendes
Paulo Siécola

Motivação

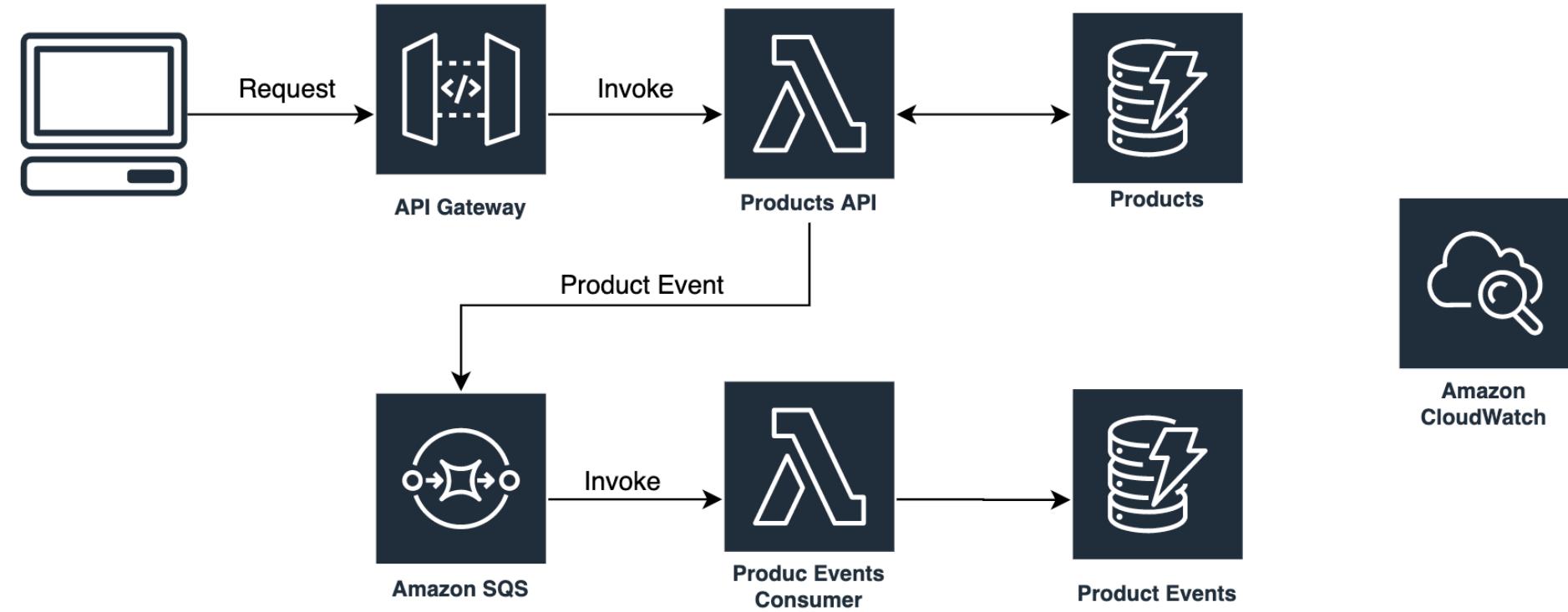


Story

- Arquitetura
- Plano de
Testes

Casos de
testes na AWS

Arquitetura do projeto



Requests

POST create_product X GET get_product + ...

▶ create_product ↎

POST https://9x38ag4qf1.execute-api.us-west-2.amazonaws.com/v1/products

Params Authorization Headers (10) Body Pre-request Script Tests Settings

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

```
1 {  
2   "id": "{{id}}",  
3   "name": "{{product_name}}",  
4   "code": "{{product_code}}",  
5   "model": "{{product_model}}",  
6   "price": 700.0  
7 }  
8  
9
```

Body Cookies Headers (8) Test Results (1/1)

Pretty Raw Preview Visualize JSON

```
1 [ {  
2   "message": "New product created"  
3 }]
```

POST create_product X GET get_product + ...

▶ get_product ↎

GET https://9x38ag4qf1.execute-api.us-west-2.amazonaws.com/v1/products/67

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

Query Params

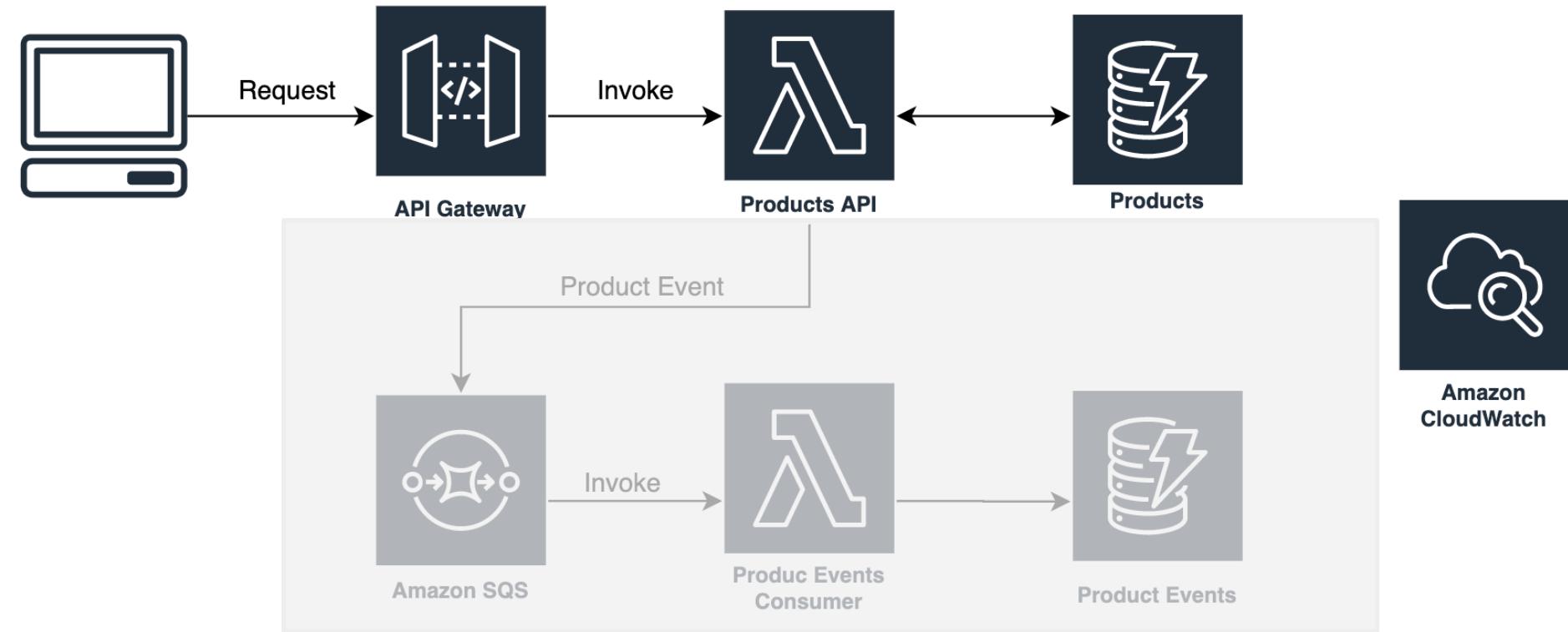
KEY
Key

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```
1 {  
2   "Product": {  
3     "id": 67,  
4     "name": "product_nob9kc",  
5     "code": "CODE_1h2f09",  
6     "model": "MOD_7uibig",  
7     "price": 700.0  
8   }  
9 }
```

Criando e buscando um produto



Testes e monitoramentos no API Gateway

Amazon API Gateway APIs > products (9x38ag4qf1) > Dashboard Show all hints

APIs Custom Domain Names VPC Links

API: products

- Resources
- Stages
- Authorizers
- Gateway Responses
- Models
- Resource Policy
- Documentation
- Dashboard**
- Settings
- Usage Plans
- API Keys

Invoke this API at: <https://9x38ag4qf1.execute-api.us-west-2.amazonaws.com/v1/>

Stage v1 From 8/2/20 To 8/16/20

API Calls

250
200
150
100
50
0

Aug 02 Aug 05 Aug 08 Aug 11 Aug 14

Latency

15000
10000
5000
0

Aug 02 Aug 05 Aug 08 Aug 11 Aug 14

Integration Latency

15000
10000
5000
0

Aug 02 Aug 05 Aug 08 Aug 11 Aug 14

4xx Error

15
10
5
0

Aug 02 Aug 05 Aug 08 Aug 11 Aug 14

5xx Error

8
6
4
2
0

Aug 02 Aug 05 Aug 08 Aug 11 Aug 14

Testes e monitoramento no Lambda Products API

Lambda > Functions > storeProduct

ARN - arn:aws:lambda:us-west-2:946835467386:function:storeProduct

storeProduct

Throttle

Qualifiers ▾

Actions ▾

StoreProduct ▾



Test

Save

Configuration

Permissions

Monitoring

CloudWatch metrics

[View traces in X-Ray](#)

[View logs in CloudWatch](#)

Add to dashboard

2020-08-09 (00:00:00) - 2020-08-09 (23:59:59) ▾



Invocations

Count

115

58

1

03:00

09:00

15:00

21:00

Invocations

Duration

Milliseconds

10.8k

5.42k

7.87

03:00

09:00

15:00

21:00

Duration Minimum Duration Average Duration Maximum

Error count and success rate (%)

Count

5

2.5

0

No unit

75

50

Errors

Success rate (%)

Testes e monitoramento no Lambda Products API

storeProduct

CloudWatch Logs Insights [Info](#)

CloudWatch Logs Insights provides a query language for analyzing log entries. The following tables list the most recent and most expensive function invocations.

Add to dashboard 1h 3h 12h 1d 3d **1w** custom ▾

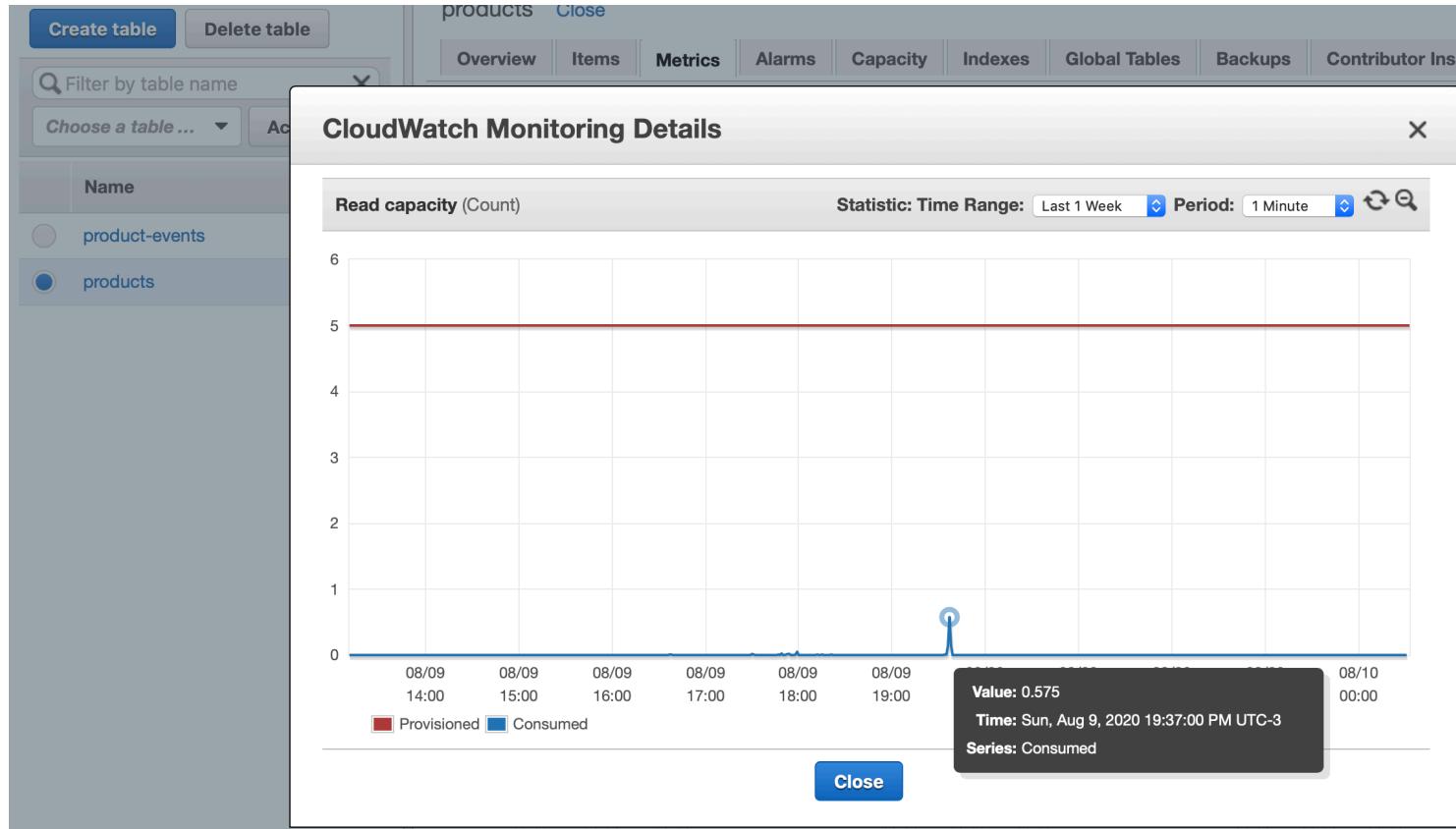
#	Timestamp	RequestID	LogStream	DurationInMS	BilledDurationInMS	N
▶ 1	2020-08-16T14:36:45.050Z	2cd8fa4-b6ea-4208-8415-596921af2926	2020/08/16[\$LATEST]860378e0ec6d4196bba6b10e8a2a2475	10595.72	10600	5
▶ 2	2020-08-09T22:40:40.725Z	59a24236-b1d7-4d51-bc9e-4e4acceb802e	2020/08/09[\$LATEST]4b4976f58bae41c89ac93ee44ca418a9	133.84	200	5
▶ 3	2020-08-09T22:40:33.656Z	3bf98c4a-1654-456d-ad3e-7964bb4a7492	2020/08/09[\$LATEST]4b4976f58bae41c89ac93ee44ca418a9	19.96	100	5
▶ 4	2020-08-09T22:40:18.556Z	bcc49e24-1317-457f-869a-a9e887c73a37	2020/08/09[\$LATEST]4b4976f58bae41c89ac93ee44ca418a9	7.87	100	5
▶ 5	2020-08-09T22:39:41.156Z	56de9843-e591-f478-9c65-05ed28b2f1a8	2020/08/09[\$LATEST]4b4976f58bae41c89ac93ee44ca418a9	25.06	100	5
▶ 6	2020-08-09T22:39:11.278Z	3e0b0c35-e2bc-4c7a-b3f0-b9da1e73c6d3	2020/08/09[\$LATEST]4b4976f58bae41c89ac93ee44ca418a9	19.27	100	5
▶ 7	2020-08-09T22:38:13.443Z	5c0957e1-7e29-4871-bb4a-dfc0833ec958	2020/08/09[\$LATEST]4b4976f58bae41c89ac93ee44ca418a9	127.37	200	5
▶ 8	2020-08-09T22:38:12.635Z	c8dbd73f-f130-4dbe-95e4-5d60f6c6461e	2020/08/09[\$LATEST]4b4976f58bae41c89ac93ee44ca418a9	77.36	100	5
▶ 9	2020-08-09T22:38:11.860Z	b9c41e4c-80c8-425b-88ff-12e473f4c051	2020/08/09[\$LATEST]4b4976f58bae41c89ac93ee44ca418a9	102	200	5

Most expensive invocations in GB-seconds (memory assigned * billed duration)					
#	Timestamp	RequestID	LogStream	BilledDurationInMS	MemorySetInMB
▶ 1	2020-08-09T19:40:03.972Z	55a9f623-e844-479a-a7d3-99e88a456a9e	2020/08/09[\$LATEST]ce9be915bd0744b19ba216971e7cbfc7	10900	512
▶ 2	2020-08-09T21:52:56.318Z	0e7bf245-082f-4d62-ac57-64ad2b5df2a	2020/08/09[\$LATEST]89a7642b7262421cb24c1de90d95aa5	10600	512
▶ 3	2020-08-16T14:36:45.050Z	2cd8fa4-b6ea-4208-8415-596921af2926	2020/08/16[\$LATEST]860378e0ec6d4196bba6b10e8a2a2475	10600	512
▶ 4	2020-08-09T20:47:37.829Z	16e553f4-6762-4ba7-b2c3-88510f359d9	2020/08/09[\$LATEST]1ffa1d7f8d6524fa8d333e9a0f274d18	10400	512
▶ 5	2020-08-09T20:30:44.309Z	46582a1f-cb4f-4448-bd5f-a858957125d6	2020/08/09[\$LATEST]f301517d14c471c974db4e08e587b7	10100	512
▶ 6	2020-08-09T21:12:44.389Z	1655390d-c462-42c6-b5e9-38624801b43	2020/08/09[\$LATEST]f5d979b7398141749c3a212d76fec89	8000	512
▶ 7	2020-08-09T22:17:10.428Z	bdedf23e-0741-4616-9caa-ecb3874407c	2020/08/09[\$LATEST]4b4976f58bae41c89ac93ee44ca418a9	7800	512
▶ 8	2020-08-09T22:23:18.891Z	7950f88c-a80a-44d9-9809-da5c6815f580	2020/08/09[\$LATEST]4b4976f58bae41c89ac93ee44ca418a9	3900	512
▶ 9	2020-08-09T22:33:43.155Z	a9e561fd-f97f-49e9-8344-feb893e64a61	2020/08/09[\$LATEST]4b4976f58bae41c89ac93ee44ca418a9	600	512

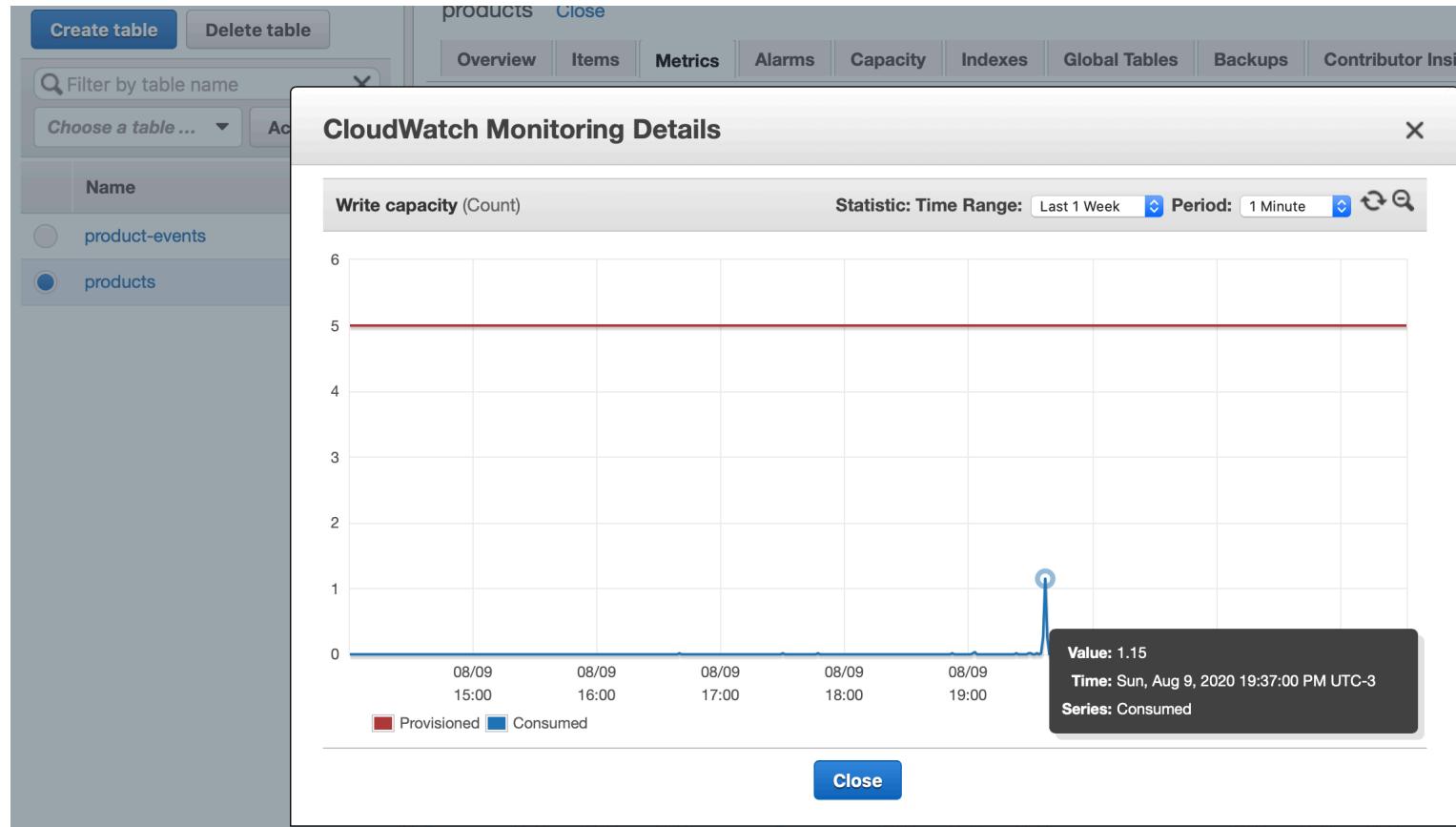
Testes e monitoramento no Lambda Product API

▶	2020-08-09T19:36:49.294-03:00	REPORT RequestId: 373af4b-8c2b-4466-91a9-64778e74b8ad Duration: 337.00 ms Billed Duration: 400 ms Memory Size: 512 MB Max Memory Used: 152 MB
▶	2020-08-09T19:36:50.184-03:00	START RequestId: 8d24b9c3-cfd6-4433-9cd0-d718f2c34100 Version: \$LATEST
▶	2020-08-09T19:36:50.295-03:00	Product created - productId: 76 - code: CODE_5fc31g
▶	2020-08-09T19:36:50.415-03:00	Product event published - messageId: 5a2a315c-b544-4354-8eef-b5f289d0d607
▶	2020-08-09T19:36:50.415-03:00	END RequestId: 8d24b9c3-cfd6-4433-9cd0-d718f2c34100
▶	2020-08-09T19:36:50.415-03:00	REPORT RequestId: 8d24b9c3-cfd6-4433-9cd0-d718f2c34100 Duration: 228.57 ms Billed Duration: 300 ms Memory Size: 512 MB Max Memory Used: 152 MB
▼	2020-08-09T19:36:51.367-03:00	START RequestId: 5e568749-d8c6-4022-957c-36a2956efc59 Version: \$LATEST
▼	2020-08-09T19:36:51.451-03:00	Product created - productId: 40 - code: CODE_vmy3gc
▼	2020-08-09T19:36:51.583-03:00	Product event published - messageId: 70571cb5-1cd5-43ff-b929-ad6c6c9115f3
▼	2020-08-09T19:36:51.583-03:00	END RequestId: 5e568749-d8c6-4022-957c-36a2956efc59
▼	2020-08-09T19:36:51.583-03:00	REPORT RequestId: 5e568749-d8c6-4022-957c-36a2956efc59 Duration: 212.91 ms Billed Duration: 300 ms Memory Size: 512 MB Max Memory Used: 152 MB
▶	2020-08-09T19:36:52.574-03:00	START RequestId: 328570e8-600a-4d0d-abd1-d00690be9260 Version: \$LATEST
▶	2020-08-09T19:36:52.694-03:00	Product created - productId: 51 - code: CODE_z2r6pm
▶	2020-08-09T19:36:52.915-03:00	Product event published - messageId: 5d5806dd-bda3-4b78-86da-479a046763d2

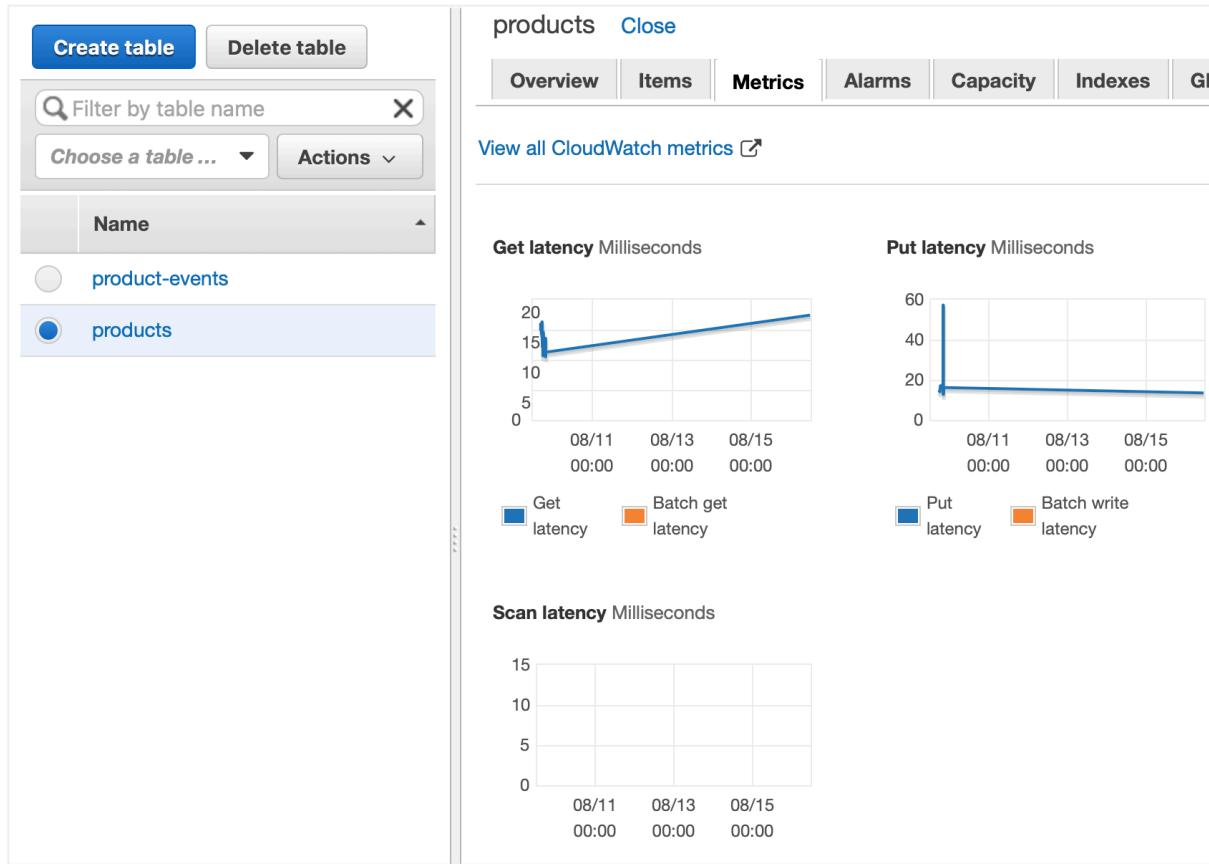
Testes e monitoramento no DynamoDB Products



Testes e monitoramento no DynamoDB Products



Testes e monitoramento no DynamoDB Products



Testes e monitoramento no DynamoDB Products

The screenshot shows the AWS DynamoDB console interface. On the left, there's a sidebar with buttons for 'Create table' and 'Delete table', a search bar 'Filter by table name', and a dropdown 'Choose a table ...'. Below these are two entries: 'product-events' and 'products', with 'products' selected and highlighted in blue. On the right, the main area is titled 'products' with a 'Close' button. It features a navigation bar with tabs: Overview, Items (which is selected), Metrics, Alarms, Capacity, Indexes, Global Tables, Backups, and CloudWatch Metrics. Below the navigation bar is a secondary navigation with 'Create item' and 'Actions'. Underneath is a search section titled 'Scan: [Table] products: id' with dropdowns for 'Scan' and '[Table] products: id', a '+ Add filter' button, and a 'Start search' button. The main content area displays a table with columns: id, code, model, name, and price. The data rows are:

	id	code	model	name	price
	1	{{procut_code}}	MOD_48uf6w	product_bnu8y	700
	2	CODE_ox4u0g	MOD_ocpvt	product_ro3qio	700
	3	CODE_5rry73	MOD_7b3xg	product_xbr3v	700
	4	COD4	Model4	MacBook	400
	5	CODE_n2zena	MOD_itzdlf	product_9fgvvm	700
	6	CODE_9cdvz	MOD_9iu0ol	product_4ht6o	700
	7	CODE_zwbijh	MOD_v39mh9	product_3kgg3	700

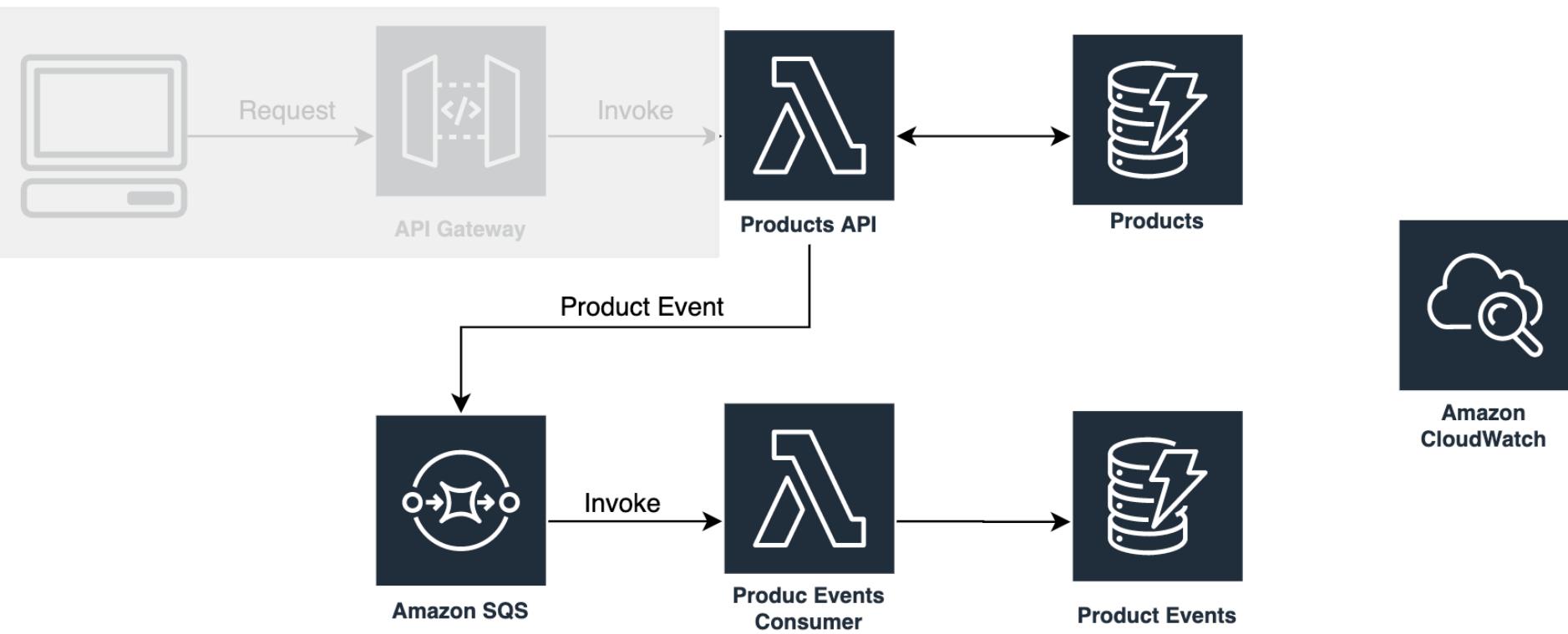
Testes e monitoramento no DynamoDB Products

Edit item

Text DynamoDB JSON

```
1 {  
2     "code": "CODE_ox4u0g",  
3     "id": 2,  
4     "model": "MOD_ocpvt",  
5     "name": "product_ro3qio",  
6     "price": 700  
7 }
```

Criando os eventos



Testes e monitoramento no SQS

Amazon SQS > Queues > product-events

product-events

Edit

Delete

Purge

Send and receive messages

SNS subscriptions

Lambda triggers

Dead-letter queue

Monitoring

Tagging

Access policy

Encryption

Add to dashboard

2020-08-09 (18:00:00) - 2020-08-09 (23:59:59) ▾



Approximate Age Of Oldest Message

Seconds

3.06k

1.53k

18:00 19:00 20:00 21:00 22:00 23:00

ApproximateAgeOfOldestMessage

Approximate Number Of Messages Delayed

Count

1

0.5

18:00 19:00 20:00 21:00 22:00 23:00

ApproximateNumberOfMessagesDelayed

Approximate Number Of Messages Not Visible

Count

0.059

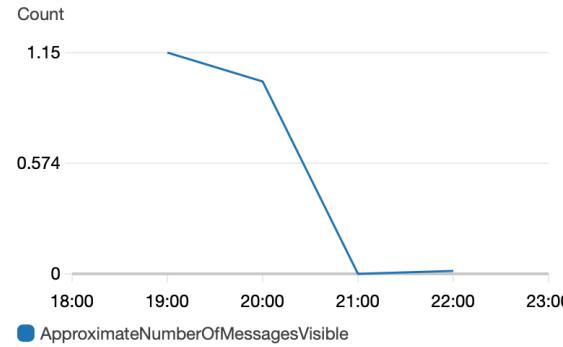
0.029

18:00 19:00 20:00 21:00 22:00 23:00

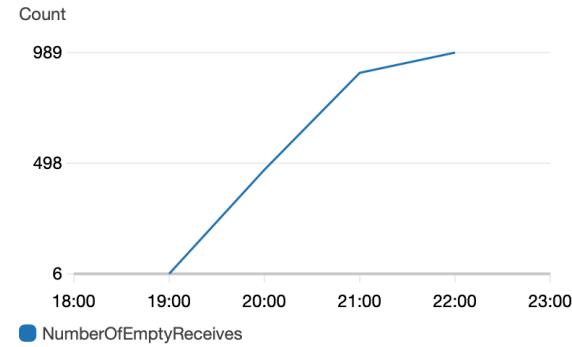
ApproximateNumberOfMessagesNotVisible

Testes e monitoramento no SQS

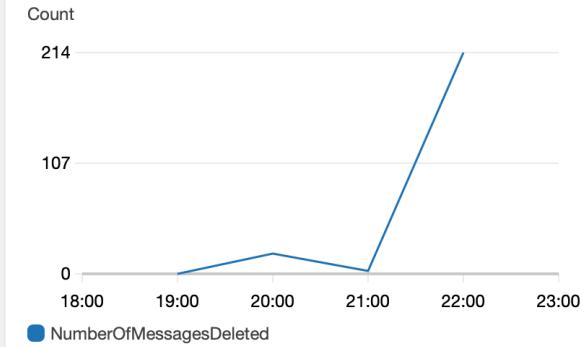
Approximate Number Of Messages Visible



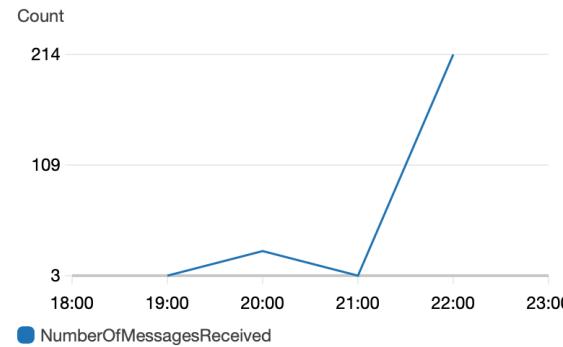
Number Of Empty Receives



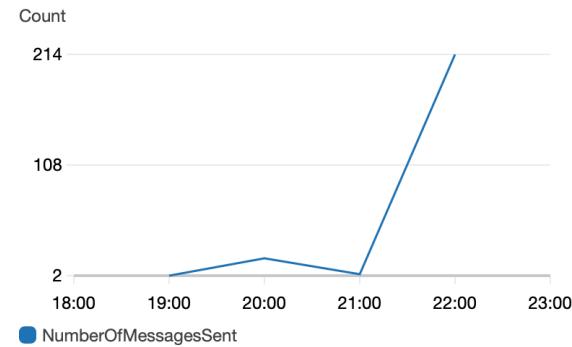
Number Of Messages Deleted



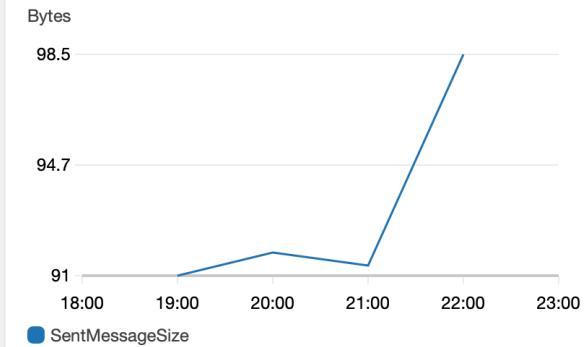
Number Of Messages Received



Number Of Messages Sent



Sent Message Size



Testes e monitoramento no DynamoDB Product Events

The screenshot shows the AWS DynamoDB console interface. On the left, a sidebar lists tables: 'Create table' (button), 'Delete table' (button), 'Filter by table name' (input field), 'Choose a table ...' (button), and 'Actions' (button). Below these are two table entries: 'product-events' (selected, indicated by a blue circle) and 'products'. The main area is titled 'product-events' with a 'Close' button. It features a navigation bar with tabs: Overview, Items, Metrics, Alarms, Capacity, Indexes, Global Tables, Backups, Contributor Insights, and Triggers. The 'Items' tab is selected. Below the tabs, there are 'Create item' and 'Actions' buttons. A search bar at the top says 'Scan: [Table] product-events: id'. A dropdown menu shows 'Scan' is selected. A search input field contains '[Table] product-events: id'. Below it is a 'Add filter' button and a 'Start search' button. The main table area has columns: id, code, event, productId, and timestamp. Six items are listed:

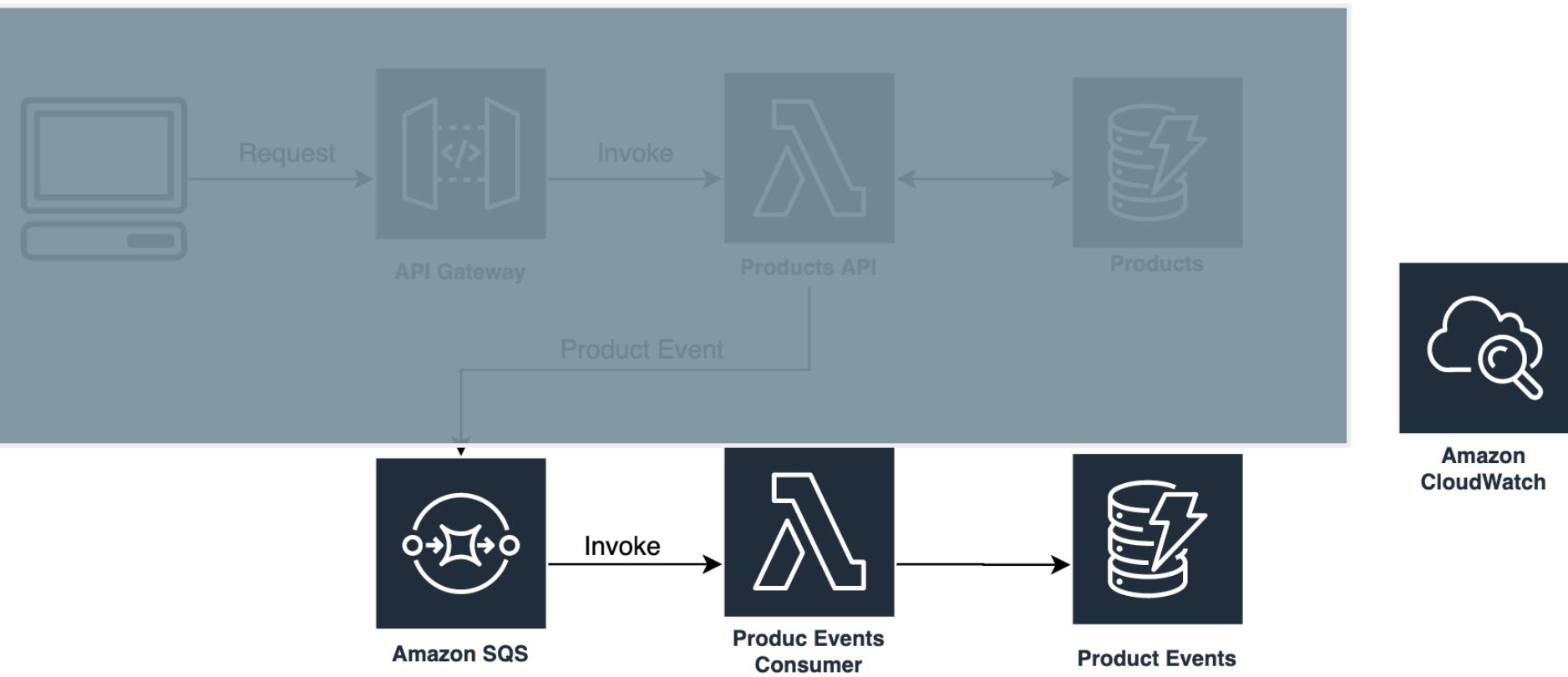
	id	code	event	productId	timestamp
<input type="checkbox"/>	001c4a1f-adfc-45a9-8787-57fe544cf742	CODE_puqtv	CREATED	88	1597012627973
<input type="checkbox"/>	014c5a02-6bbd-4e3c-a557-53556496000e	CODE_n2zena	RETRIEVED	5	1597012642509
<input type="checkbox"/>	04f741d5-4d24-4312-afee-d7baff4b6276	CODE_n2zena	RETRIEVED	5	1597012687333
<input type="checkbox"/>	060d1d20-ade4-4d24-bed4-0a5b54a89547	COD6	RETRIEVED	6	1597006182290
<input type="checkbox"/>	06769ce9-aa1a-4573-88c7-833604b7fd8d	CODE_piz0kt	CREATED	45	1597012630946
<input type="checkbox"/>	094ade95-d99d-4db5-b75b-3858cff745ed	CODE_n2zena	CREATED	5	1597012504998

Testes e monitoramento no DynamoDB Product Events

The screenshot shows the AWS DynamoDB console interface. On the left, there's a sidebar with buttons for 'Create table' and 'Delete table', and a search bar labeled 'Filter by table name'. Below that is a list of tables: 'product-events' (selected) and 'products'. The main area is titled 'product-events' and shows the 'Items' tab selected. At the top, there are tabs for Overview, Metrics, Alarms, Capacity, Indexes, Global Tables, Backups, Contributor Insights, and Triggers. Below these tabs, there are buttons for 'Create item' and 'Actions'. A search bar at the top says 'Scan: [Table] product-events: id'. The results table has columns: id, code, event, productId, and timestamp. There are five items listed:

	id	code	event	productId	timestamp
1	74237de7-ff24-487c-96ea-e504167456a0	COD5	CREATED	5	1597002003071
2	bc4e9a78-f7cd-47f6-9d23-3de7f272508e	COD5	RETRIEVED	5	1597006334340
3	7df51013-bb32-48b7-8b62-29bfe878db08	COD5	RETRIEVED	5	1597006408740
4	72875449-d806-490a-8c91-078b4d8aaf5c	COD5	RETRIEVED	5	1597006439941
5	86b758fd-99a3-42d0-a981-dbca2456c95	COD5	RETRIEVED	5	1597006441400

Arquitetura incompleta – parte 1



O que consigo testar?

Amazon SQS > Queues

Queues (1)					Edit	Delete	Purge	Actions ▾	
					Send and receive messages				
					Subscribe to Amazon SNS topic				
					Configure Lambda function trigger				
Name	▲	Type	▼	Created	▼	Messages available	▼	Messages in flight	▼
product-events		Standard		8/9/2020, 16:24:55		0		0	

O que consigo testar?

The screenshot shows an IDE interface with two tabs open: `ProductEventsHandler.java` and `ProductEvent.java`. A tooltip titled "Definition of ProductEvent" is displayed over the code in `ProductEvent.java`.

```
ProductEventsHandler.java
import com.amazonaws.services.dynamodbv2.AmazonDynamoDB;
import com.amazonaws.services.dynamodbv2.document.Document;
import com.amazonaws.services.dynamodbv2.document.PutItemOutcome;
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.LambdaLogger;
import com.amazonaws.services.lambda.runtime.RequestHandler;
import com.amazonaws.services.lambda.runtime.events.SQSEvent;
import com.google.gson.Gson;

public class ProductEventsHandler implements RequestHandler<SQSEvent, String> {
    private static final String DYNAMODB_TABLE_NAME = "Products";
    private static final String SQS_QUEUE_NAME = "ProductsQueue";

    @Override
    public void handleRequest(SQSEvent input, Context context) {
        LambdaLogger logger = context.getLogger();
        logger.info("Received SQS message: " + input);

        for (SQSEvent.SQSMessage msg : input.getMessages()) {
            Gson gson = new Gson();
            ProductEvent productEvent = gson.fromJson(msg.getBody(), ProductEvent.class);
            logger.log("Product event received - messageId: " + msg.getMessageId() + " - productId: " +
                    productEvent.getProductId() + " - event: " + productEvent.getEvent());
            saveProductEvent(msg.getMessageId(), productEvent);
        }
    }

    private void saveProductEvent(String messageId, ProductEvent productEvent) {
        AmazonDynamoDB dynamoDB = AmazonDynamoDBClientBuilder.defaultClient();
        PutItemOutcome outcome = dynamoDB.putItem(DYNAMODB_TABLE_NAME, productEvent);
        logger.info("Put item outcome: " + outcome);
    }
}
```

```
ProductEvent.java
public class ProductEvent {
    private int productId;
    private String code;
    private String event;
    private long timestamp;

    public String toString() {
        Gson gson = new GsonBuilder().setPrettyPrinting().create();
        return gson.toJson(this);
    }

    public int getProductId() {
        return productId;
    }

    public void setProductId(int productId) {
        this.productId = productId;
    }

    public String getCode() {
        return code;
    }

    public void setCode(String code) {
        this.code = code;
    }

    public String getEvent() {
        return event;
    }

    public void setEvent(String event) {
        this.event = event;
    }

    public long getTimestamp() {
        return timestamp;
    }

    public void setTimestamp(long timestamp) {
        this.timestamp = timestamp;
    }
}
```

O que consigo testar?

Amazon SQS > Queues > product-events > Send and receive messages

Send and receive messages

Send messages to and receive messages from a queue.

Send message [Info](#)

[Clear content](#)[Send message](#)

Message body

Enter the message to send to the queue.

```
{  
  "productId": 123456,  
  "code": "COD123456",  
  "event": "CREATED",  
  "timestamp": 1597590829000  
}
```



Delivery delay [Info](#)



Should be between 0 seconds and 15 minutes.

► Message attributes - *Optional* [Info](#)

O que consigo testar?

The screenshot shows the AWS Lambda console interface for the 'product-events' table. On the left, there's a sidebar with 'Create table' and 'Delete table' buttons, a search bar for 'Filter by table name', and a dropdown for 'Choose a table ...'. Below these are sections for 'Name' (with 'product-events' selected) and 'products'. The main area is titled 'product-events' and includes tabs for Overview, Items, Metrics, Alarms, Capacity, Indexes, Global Tables, Backups, Contributor Insights, and Triggers. The 'Items' tab is active. Below the tabs, there are 'Create item' and 'Actions' buttons. A search bar at the top says 'Scan: [Table] product-events: id'. Underneath, there's a filter bar with 'Scan' selected, a dropdown for 'product-events: id', a 'Filter' section for 'productId' (set to 'Number = 123456'), an 'Add filter' button, and a 'Start search' button. At the bottom, there's a table with columns: id, code, event, productId, and timestamp. One row is shown: id '9fbebba5-dc79-4264-9fd-3b700091c244', code 'COD123456', event 'CREATED', productId '123456', and timestamp '1597590829000'.

	id	code	event	productId	timestamp
	9fbebba5-dc79-4264-9fd-3b700091c244	COD123456	CREATED	123456	1597590829000

O que consigo testar?

CloudWatch

Dashboards

Alarms

ALARM 0

INSUFFICIENT 0

OK 4

Billing

Logs

Log groups

Insights

Metrics

Events

Rules

Event Buses

ServiceLens

Service Map

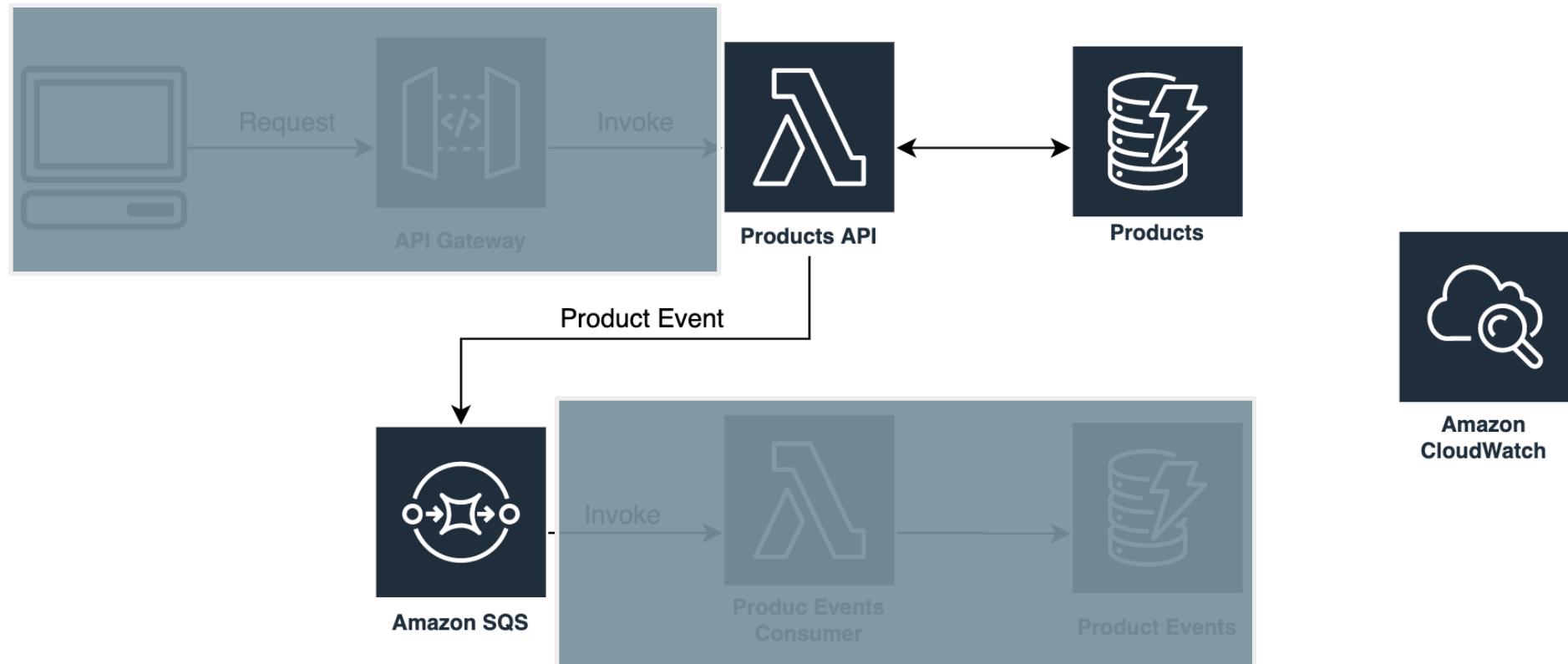
CloudWatch > CloudWatch Logs > Log groups > /aws/lambda/product-event-consumer > 2020/08/16/[&LATEST]81d7f6f3454b4589b9a0630b5094c80f Switch to the original interface.

Log events

Filter events Clear 1m 30m 1h 12h Custom (3h) ⚖️ ⚙️

Timestamp	Message
There are older events to load. Load more.	
▶ 2020-08-16T12:14:51.611-03:00	START RequestId: c280688c-13e6-58ad-85e0-3f3f5c7dac08 Version: \$LATEST
▼ 2020-08-16T12:14:51.973-03:00	Product event received - messageId: 9fbbeba5-dc79-4264-9f9d-3b700091c244 - productId: 123456 - event: CREATED
▼ 2020-08-16T12:15:00.496-03:00	END RequestId: c280688c-13e6-58ad-85e0-3f3f5c7dac08
▶ 2020-08-16T12:15:00.496-03:00	REPORT RequestId: c280688c-13e6-58ad-85e0-3f3f5c7dac08 Duration: 8884.28 ms Billed Duration: 8900 ms Memory Size: No newer events at this moment. Auto retry paused. Resume

Arquitetura incompleta – parte 2



O que posso testar?

Lambda > Functions > getProduct

ARN - arn:aws:lambda:us-west-2:946835467386:function:getProduct

getProduct

Configuration Permissions Monitoring

Throttle Qualifiers Actions ▾

getProduct ▾

- Saved Test Events
- getProduct
- Configure test events

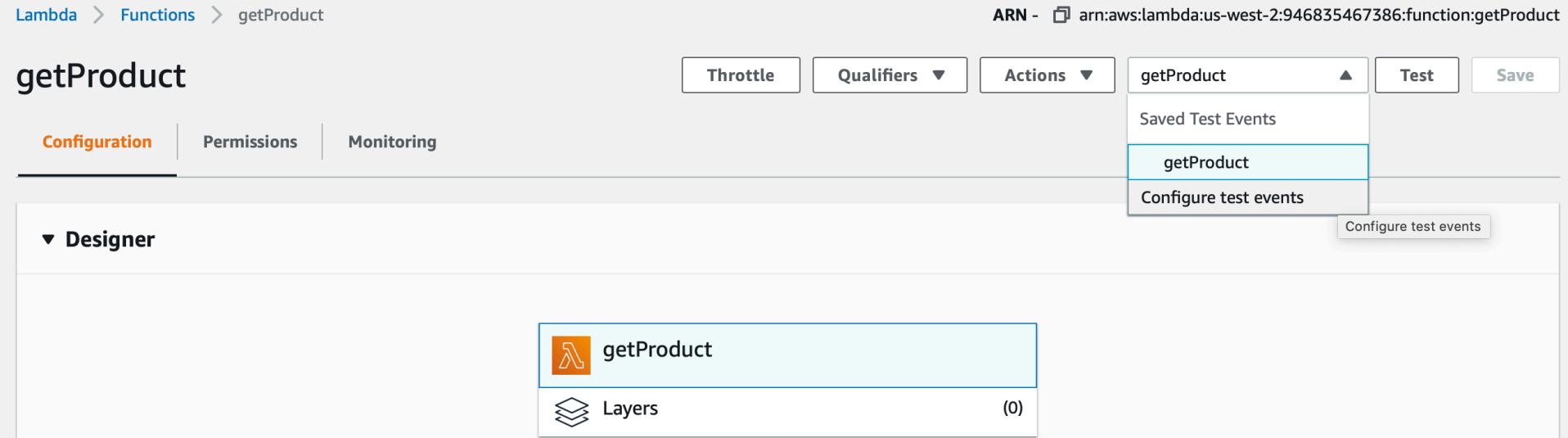
Test Save

▼ Designer

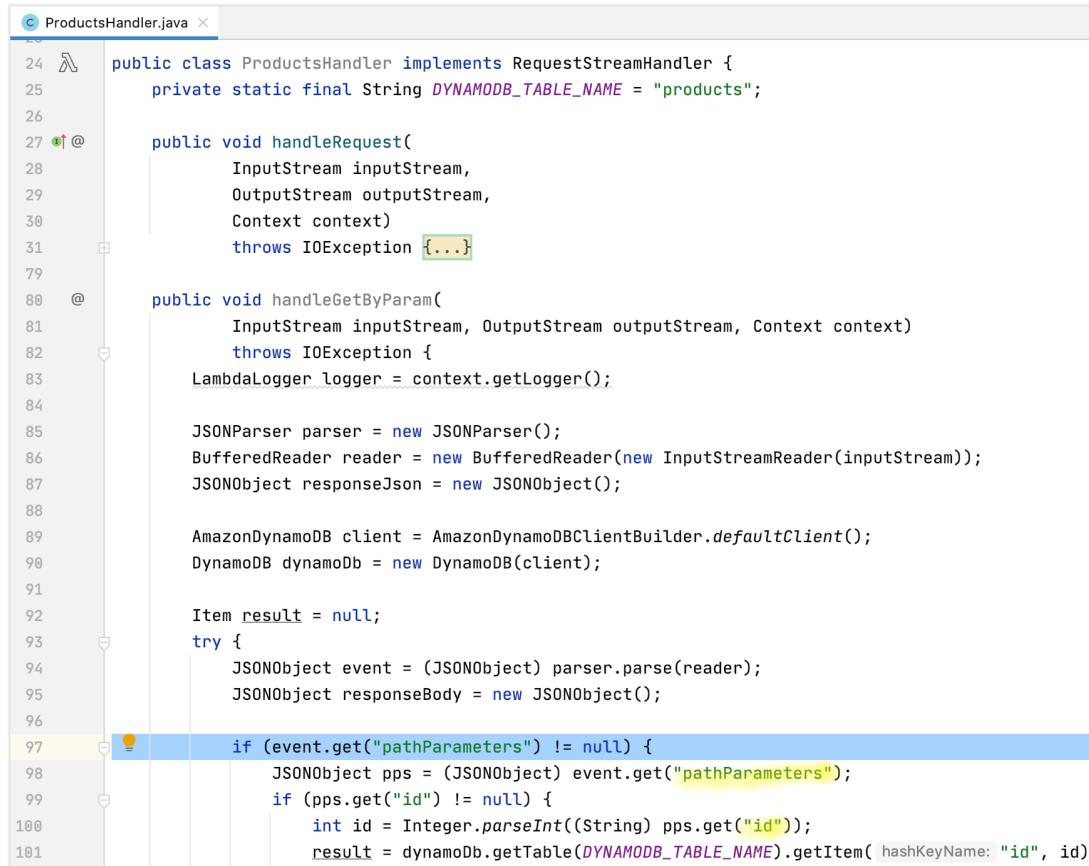
 **getProduct**

 Layers (0)

Configure test events



O que posso testar?



```
ProductsHandler.java
24 public class ProductsHandler implements RequestStreamHandler {
25     private static final String DYNAMODB_TABLE_NAME = "products";
26
27     @Override
28     public void handleRequest(
29             InputStream inputStream,
30             OutputStream outputStream,
31             Context context)
32             throws IOException {...}
33
34     @Override
35     public void handleGetByParam(
36             InputStream inputStream, OutputStream outputStream, Context context)
37             throws IOException {
38         LambdaLogger logger = context.getLogger();
39
40         JSONParser parser = new JSONParser();
41         BufferedReader reader = new BufferedReader(new InputStreamReader(inputStream));
42         JSONObject responseJson = new JSONObject();
43
44         AmazonDynamoDB client = AmazonDynamoDBClientBuilder.defaultClient();
45         DynamoDB dynamoDb = new DynamoDB(client);
46
47         Item result = null;
48         try {
49             JSONObject event = (JSONObject) parser.parse(reader);
50             JSONObject responseBody = new JSONObject();
51
52             if (event.get("pathParameters") != null) {
53                 JSONObject pps = (JSONObject) event.get("pathParameters");
54                 if (pps.get("id") != null) {
55                     int id = Integer.parseInt((String) pps.get("id"));
56                     result = dynamoDb.getTable(DYNAMODB_TABLE_NAME).getItem(hashKeyName: "id", id);
57                 }
58             }
59         } catch (IOException e) {
60             logger.error("Error processing request: " + e.getMessage());
61         }
62         responseBody.put("result", result);
63         responseBody.toString();
64         outputStream.write(responseBody.toString().getBytes());
65     }
66 }
```

O que posso testar?

The screenshot shows the AWS Lambda console interface. On the left, there's a sidebar with 'aws' logo, 'Services' dropdown, and 'Resources' dropdown. Below that, the navigation path is 'Lambda > Functions > getProduc...'. The main area has a large title 'getProduct' and tabs for 'Configuration' (which is active) and 'Permissions'. Under 'Configuration', there's a 'Designer' section. On the right, a modal window titled 'Configure test event' is open. It contains a text area with the following content:

```
A function can have up to 10 test events. The events are persisted and test your function with the same events.

 Create new test event
 Edit saved test events
```

Below this is a 'Saved test event' section with a dropdown menu containing 'getProduct'. The code editor shows the following JSON:1 {
2 "pathParameters": {
3 "id": "1"
4 }
5 }

O que posso testar?

getProduct

Throttle Qualifiers Actions getProduct Test S...

Execution result: succeeded ([logs](#))

▼ Details

The area below shows the result returned by your function execution. [Learn more](#) about returning results from your function.

```
{  
  "headers": {  
    "x-custom-header": "custom header"  
  },  
  "body": "{\"Product\":{\"id\": 1, \"name\": \"product_bnu8y\", \"code\": \"{{procut_code}}\", \"model\": \"MOD_48uf6w\", \"price\": 700.0}, \"statusCode\": 200}  
}
```

Summary

Code SHA-256	Request ID
IzKbsG63TGzC0owapH5G7Oh+00peEDdYbavVz4xiOes=	d93a7997-761b-413b-99b1-7029b18813a1
Duration	Billed duration
316.96 ms	400 ms
Resources configured	Max memory used
512 MB	142 MB

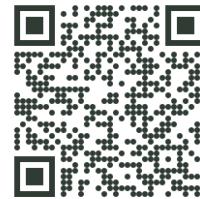
Log output

The section below shows the logging calls in your code. These correspond to a single row within the CloudWatch log group corresponding to this Lambda function. [Click here](#) to view the CloudWatch log group.

```
START RequestId: d93a7997-761b-413b-99b1-7029b18813a1 Version: $LATEST  
Product retrieved - productId: 1 - code: {{procut_code}}Product event published - messageId: 064cc643-0e4e-48d4-92b4-ecbeafb3f9baEND  
RequestId: d93a7997-761b-413b-99b1-7029b18813a1  
REPORT RequestId: d93a7997-761b-413b-99b1-7029b18813a1 Duration: 316.96 ms Billed Duration: 400 ms Memory Size: 512 MB Max Memory Used: 142 MB
```

Fontes

SCAN ME



Products events consumer



Products API

Obrigado!



LinkedIn



GitHub