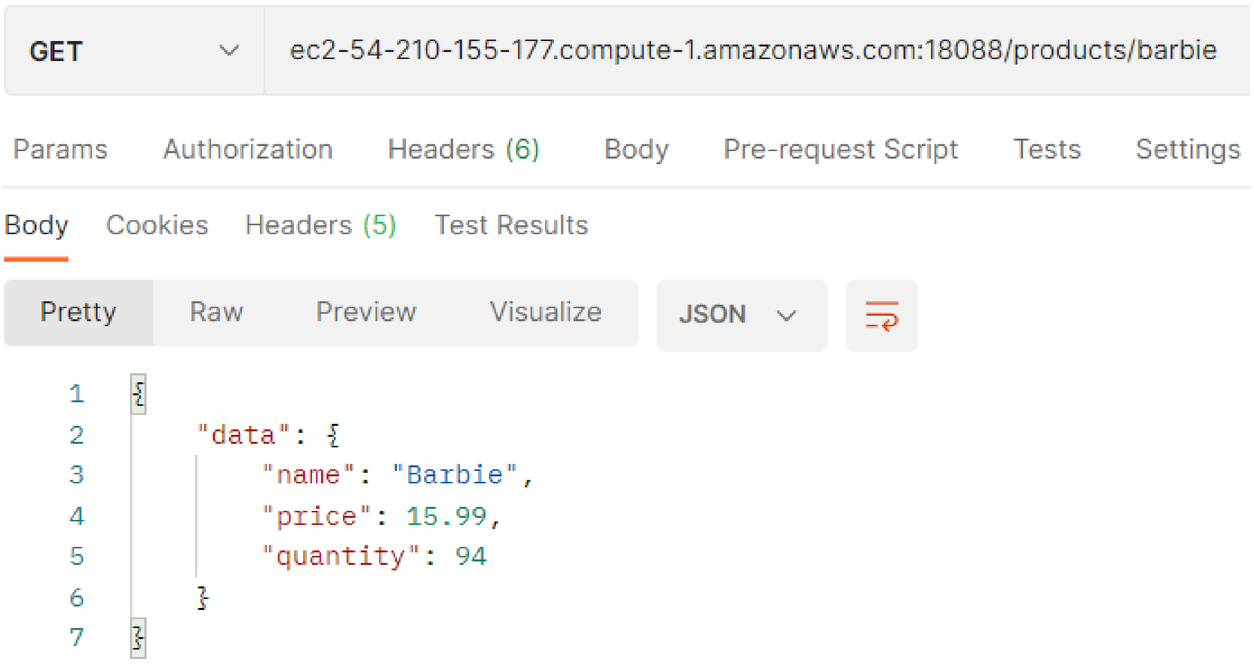
Application Tests

*All requests are send to front-end service at port 18088.

*Certain fault-tolerance test are done by requesting each order replioca

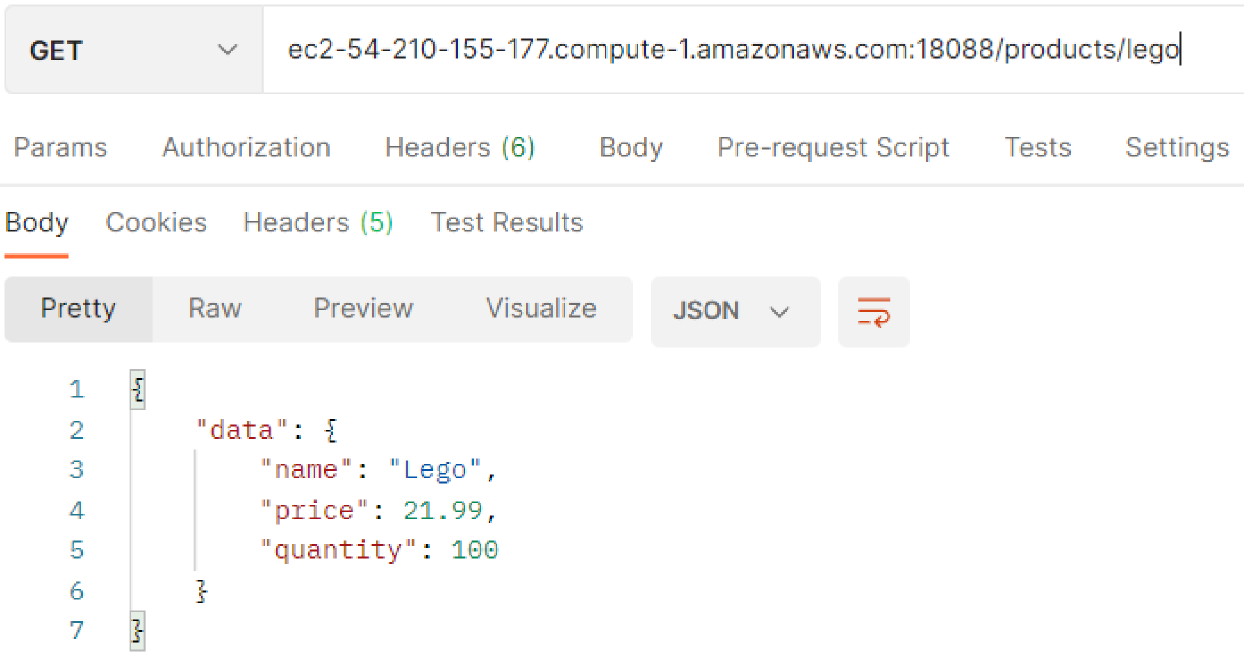
Query:

Query for products:



GET ec2-54-210-155-177.compute-1.amazonaws.com:18088/products/barbie

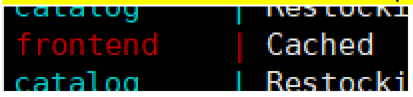
```
{
    "data": {
        "name": "Barbie",
        "price": 15.99,
        "quantity": 94
    }
}
```



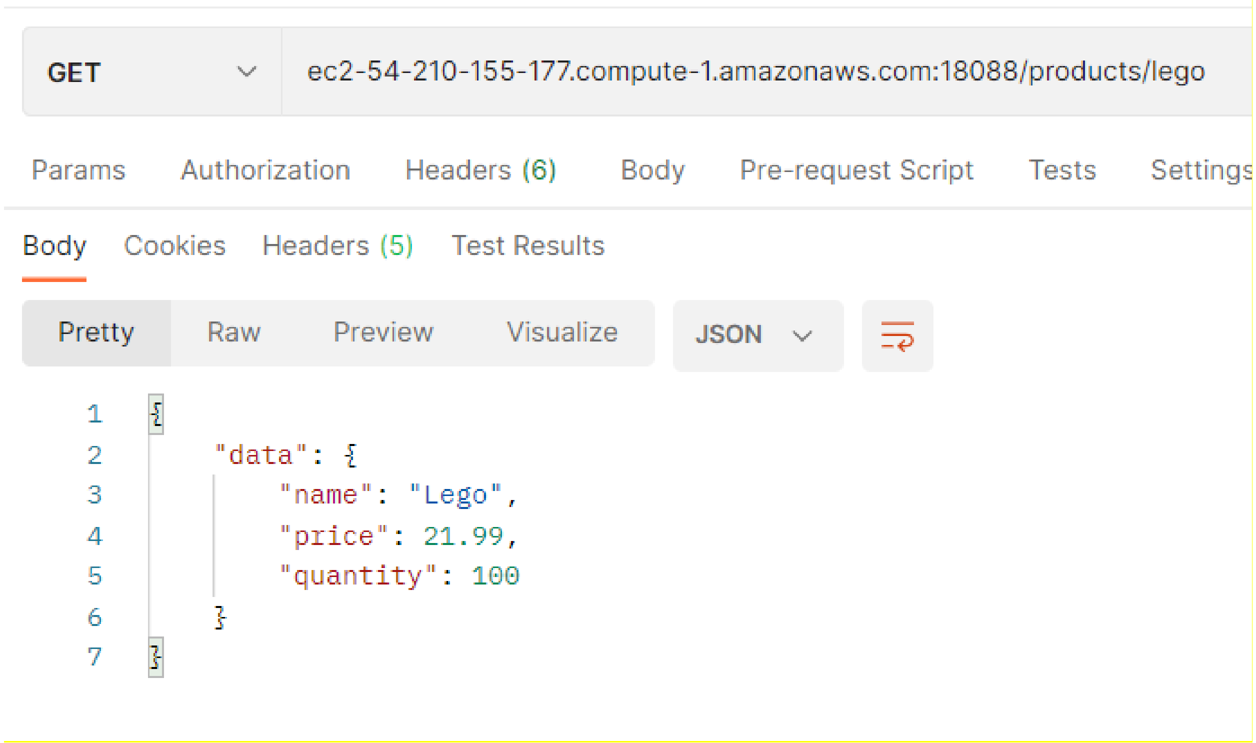
GET ec2-54-210-155-177.compute-1.amazonaws.com:18088/products/lego

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

Server Reads cache at second request:

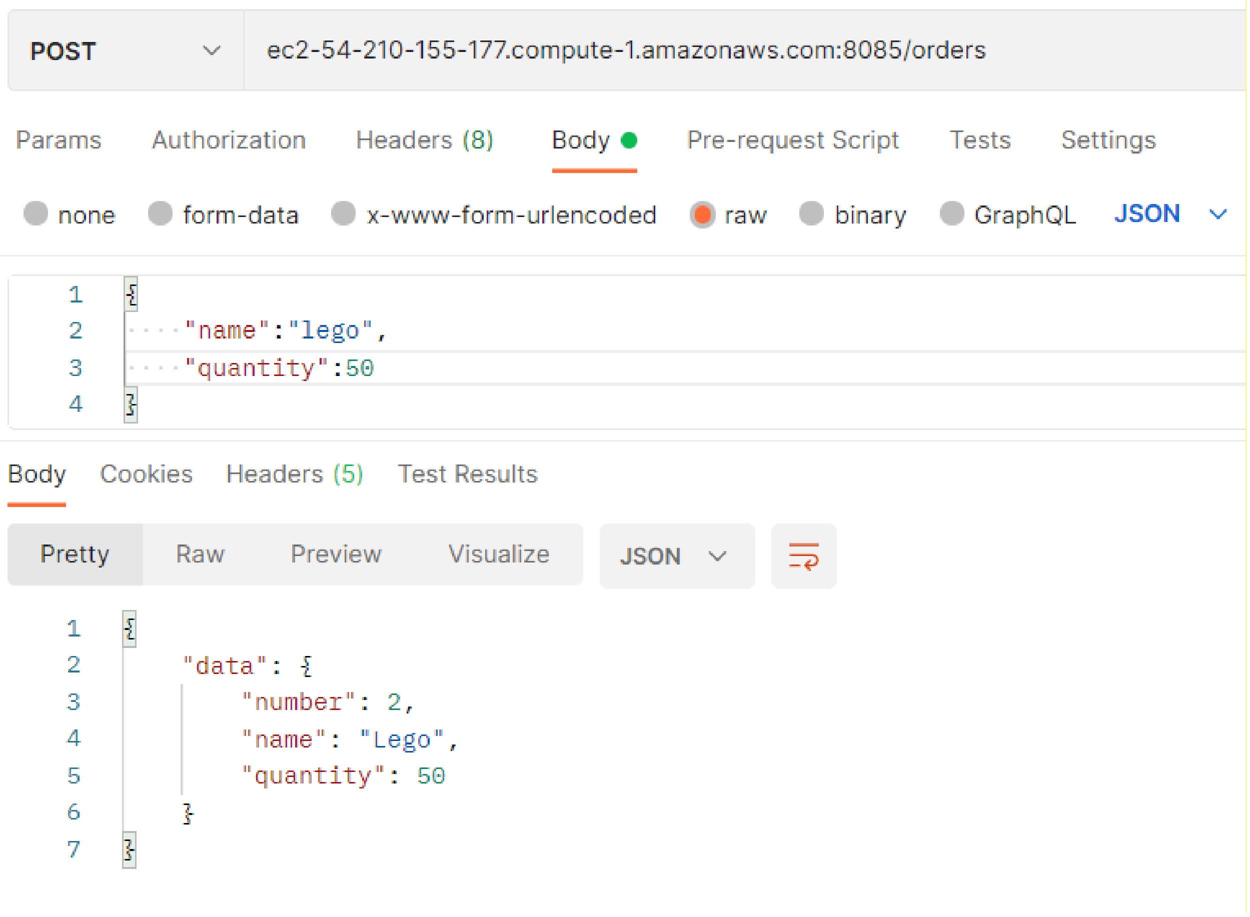


```
catalog    | Restocki
frontend   | Cached
catalog    | Restocki
```


GET ec2-54-210-155-177.compute-1.amazonaws.com:18088/products/lego

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

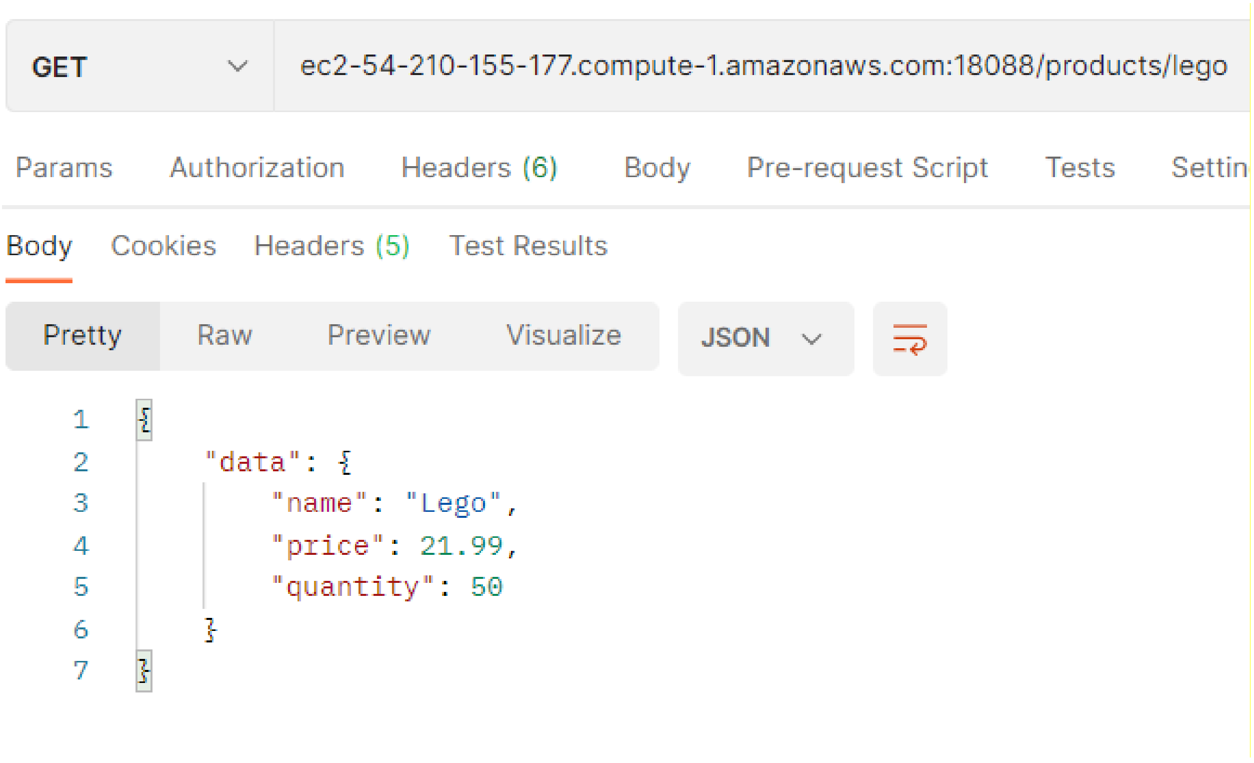
Server can successfully process order:


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

```
{
    "name":"lego",
    "quantity":50
}
```

```
{
    "data": {
        "number": 2,
        "name": "Lego",
        "quantity": 50
    }
}
```

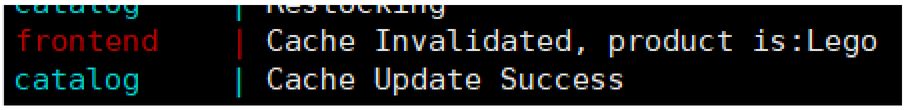
Subsequent query will reflect stock change:



ec2-54-210-155-177.compute-1.amazonaws.com:18088/products/lego

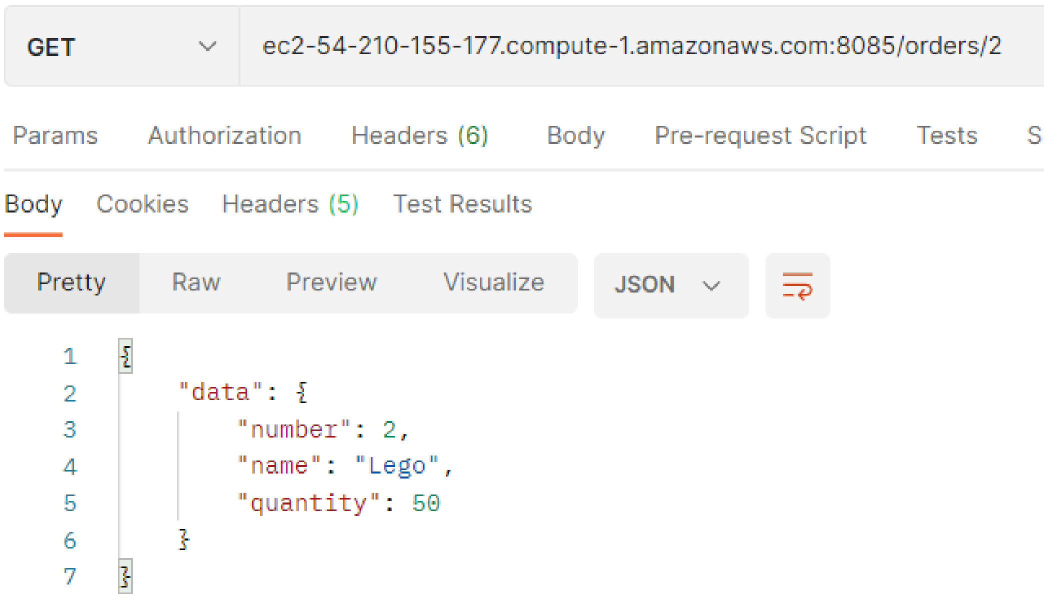
```
{
    "data": {
        "name": "Lego",
        "price": 21.99,
        "quantity": 50
    }
}
```

Frontend server should invalidate its query:



```
frontend      | Cache Invalidated, product is:Lego
catalog       | Cache Update Success
```

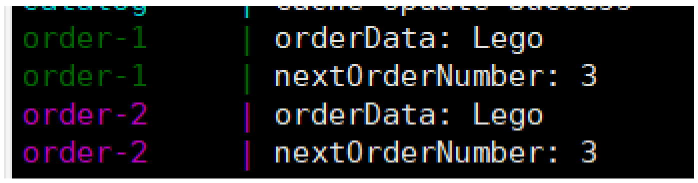
Frontend server should be able to retrieve previously succeed order.



ec2-54-210-155-177.compute-1.amazonaws.com:8085/orders/2

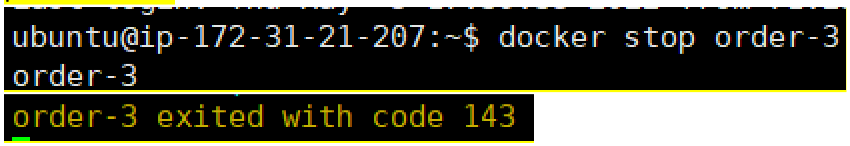
```
{
    "data": {
        "number": 2,
        "name": "Lego",
        "quantity": 50
    }
}
```

The order is replicated across all order replicas



```
order-1          | orderData: Lego
order-1          | nextOrderNumber: 3
order-2          | orderData: Lego
order-2          | nextOrderNumber: 3
```

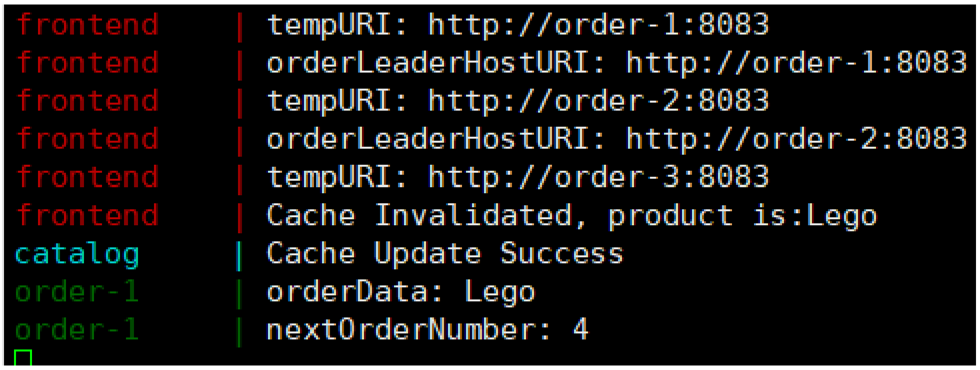
The server can process order with leader replica failed, the leader should find new leader and continue process order.



```
ubuntu@ip-172-31-21-207:~$ docker stop order-3
order-3
```

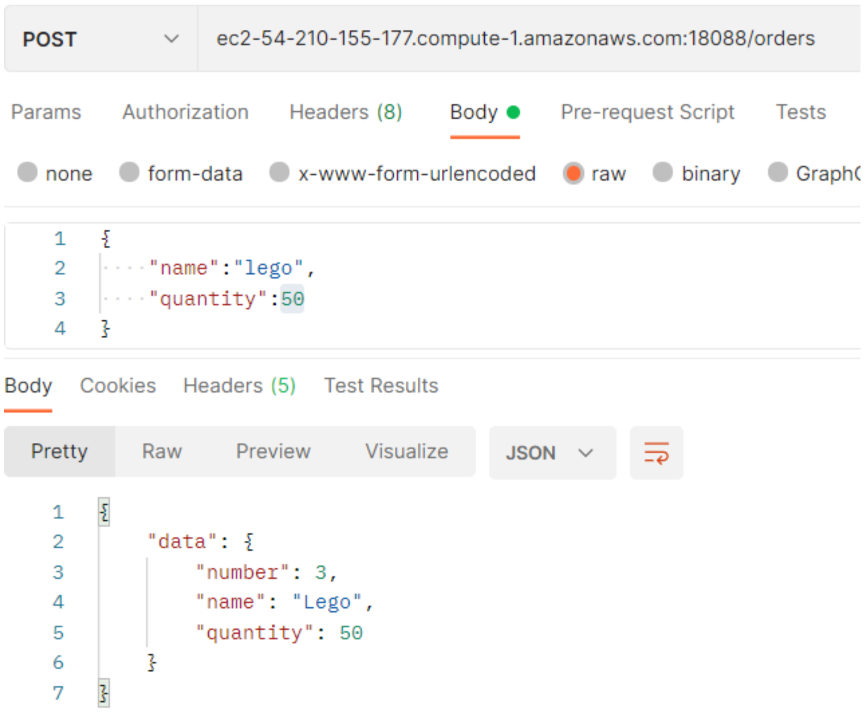
```
order-3 exited with code 143
```

Updated leader:



```
frontend        | tempURI: http://order-1:8083
frontend        | orderLeaderHostURI: http://order-1:8083
frontend        | tempURI: http://order-2:8083
frontend        | orderLeaderHostURI: http://order-2:8083
frontend        | tempURI: http://order-3:8083
frontend        | Cache Invalidated, product is:Lego
catalog         | Cache Update Success
order-1         | orderData: Lego
order-1         | nextOrderNumber: 4
```

New order

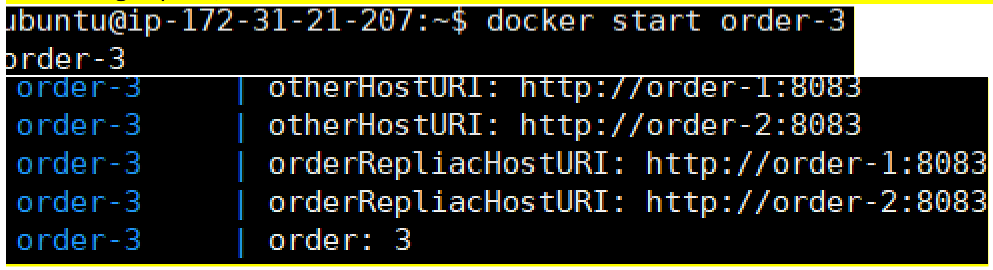


POST ec2-54-210-155-177.compute-1.amazonaws.com:18088/orders

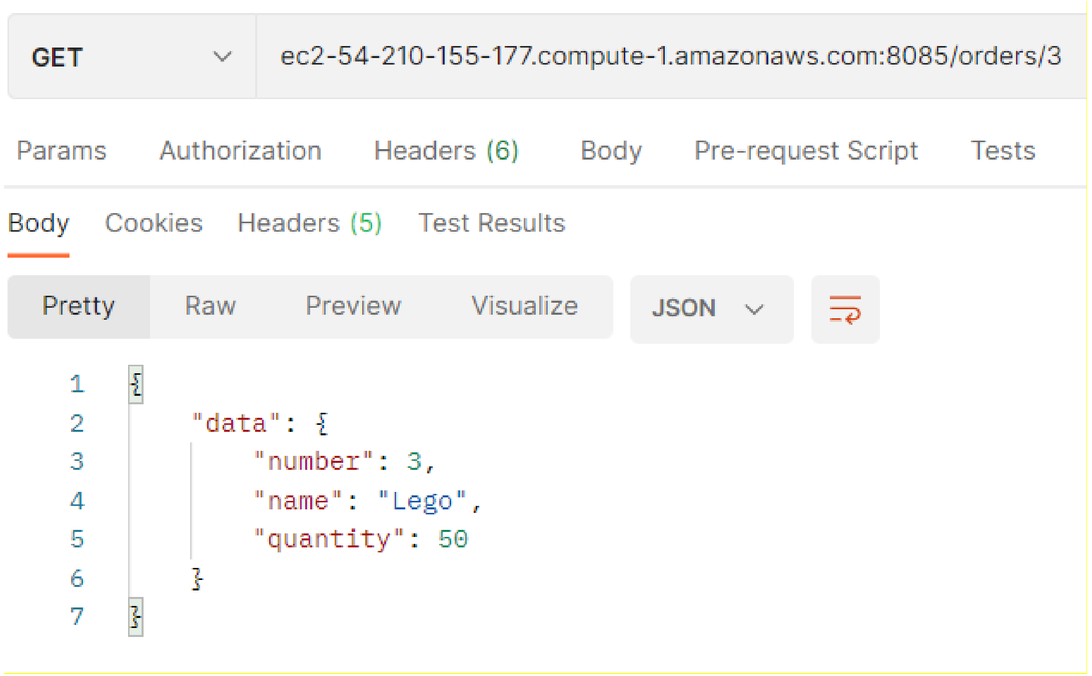
```
{
    "name":"lego",
    "quantity":50
}
```

```
{
    "data": {
        "number": 3,
        "name": "Lego",
        "quantity": 50
    }
}
```

Restarting replica, the order service should recover and retrieve order from other replicas



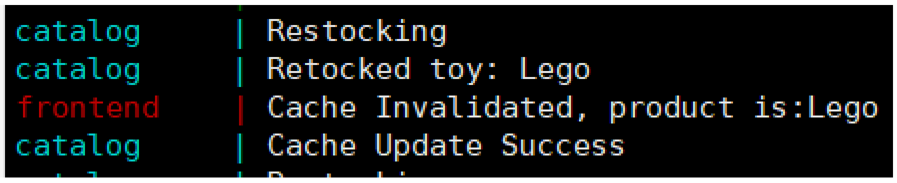
```
buntu@ip-172-31-21-207:~$ docker start order-3
rder-3
order-3          | otherHostURI: http://order-1:8083
order-3          | otherHostURI: http://order-2:8083
order-3          | orderRepliacHostURI: http://order-1:8083
order-3          | orderRepliacHostURI: http://order-2:8083
order-3          | order: 3
```



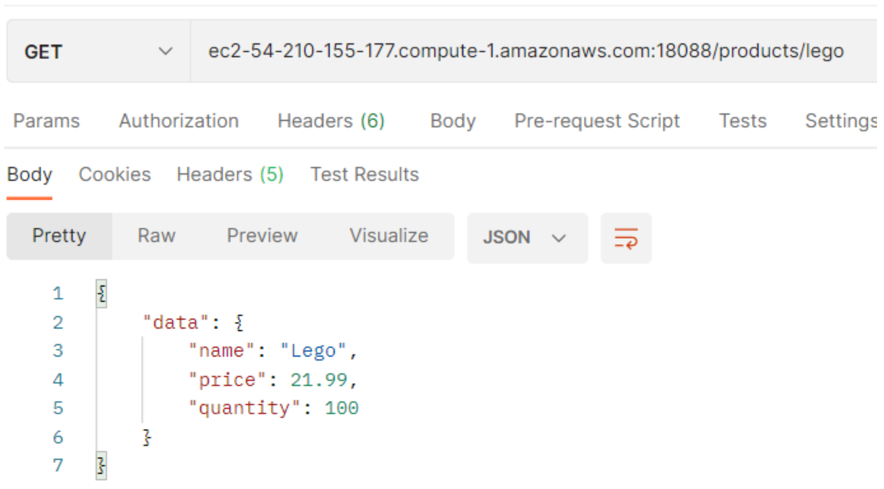
GET ec2-54-210-155-177.compute-1.amazonaws.com:8085/orders/3

```
{
    "data": {
        "number": 3,
        "name": "Lego",
        "quantity": 50
    }
}
```

Catalog service will restock after two order with 50 in quantity and update frontend service cache:



```
catalog          | Restocking
catalog          | Retocked toy: Lego
frontend         | Cache Invalidated, product is:Lego
catalog          | Cache Update Success
```



GET ec2-54-210-155-177.compute-1.amazonaws.com:18088/products/lego

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