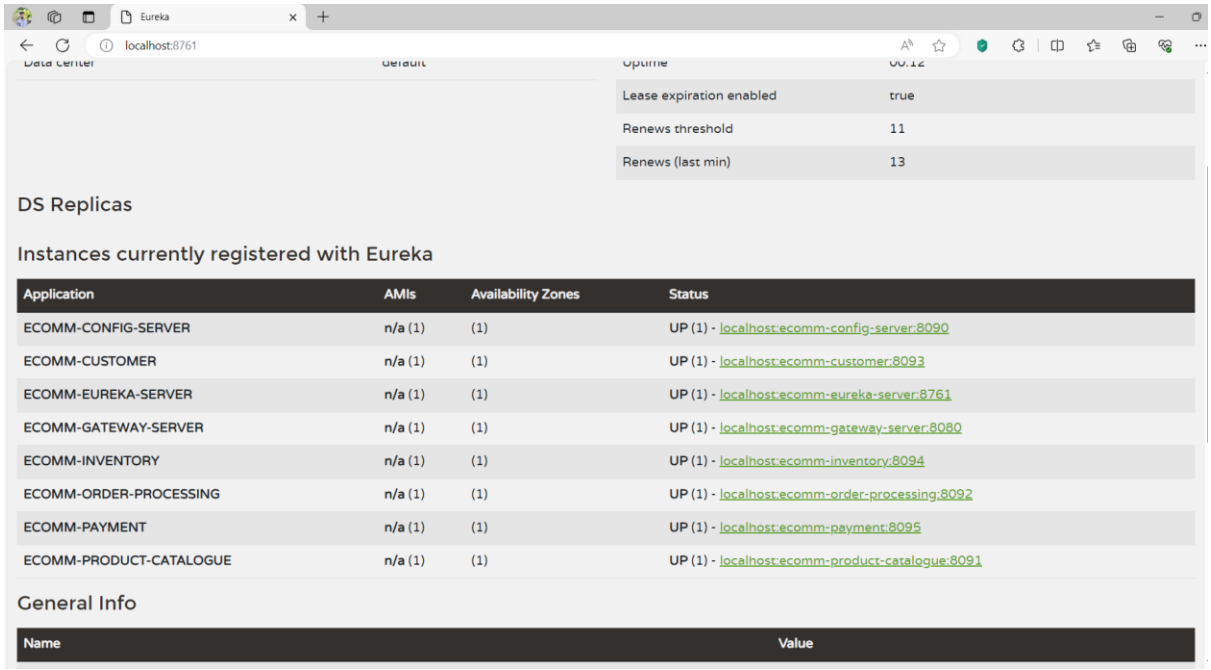
Ecommerce Application Repository

1) Config server - https://github.com/jeffinjude/ecomm-config-server
2) Config server props - https://github.com/jeffinjude/ecomm-config-server-props
3) Eureka server - https://github.com/jeffinjude/ecomm-eureka-server
4) Gateway server - https://github.com/jeffinjude/ecomm-gateway-server
5) Customer service - https://github.com/jeffinjude/ecomm-customer
6) Product catalogue service - https://github.com/jeffinjude/ecomm-product-catalogue
7) Inventory service - https://github.com/jeffinjude/ecomm-inventory
8) Order processing service - https://github.com/jeffinjude/ecomm-order-processing
9) Payment service - https://github.com/jeffinjude/ecomm-payment
10) Kubernetes config files - https://github.com/jeffinjude/ecomm-k8s-deployment
11) Ecomm resources - https://github.com/jeffinjude/ecomm-resources

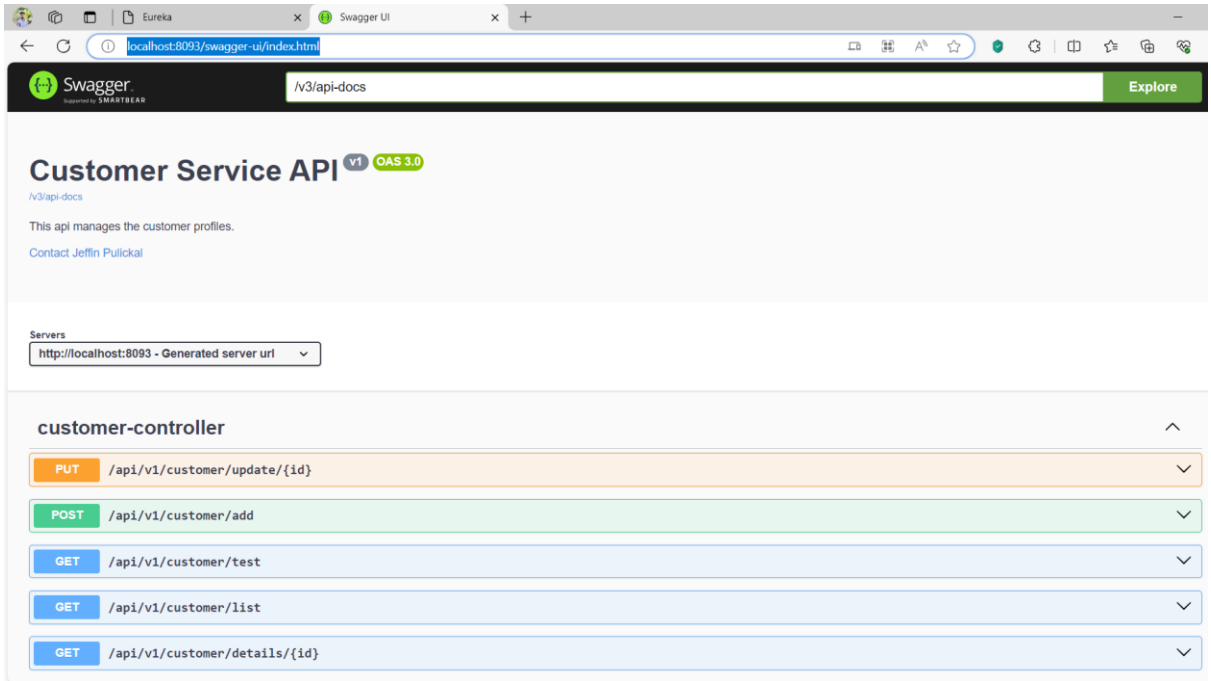
Ecommerce Application Setup

1) Start the apps in the following order. Make sure to ensure config, eureka and gateway server are completely started to prevent errors in the business microservices.
    - ecomm-config-server
    - ecomm-eureka-server
    - ecomm-gateway-server
    - ecomm-customer
    - ecomm-product-catalogue
    - ecomm-inventory
    - ecomm-order-processing
    - ecomm-payment
2) Access eureka server on localhost:8761 to list out all the registered services.

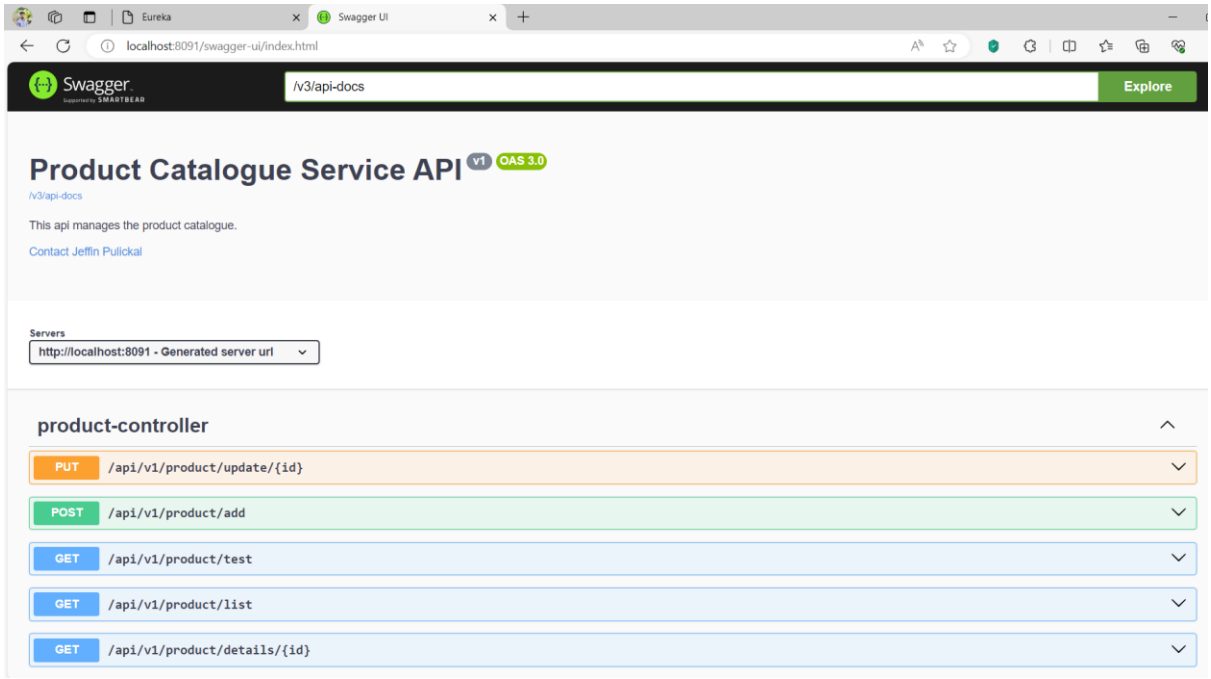


3) Access swagger endpoints of the services.

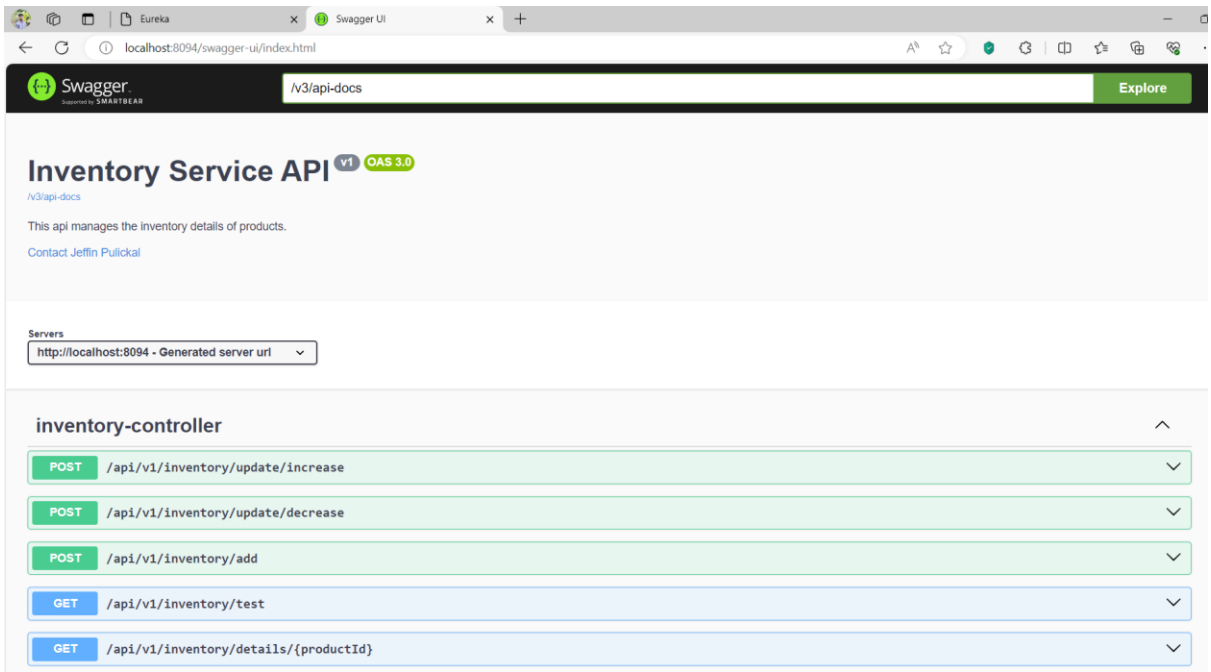
- Customer Service - http://localhost:8093/swagger-ui/index.html



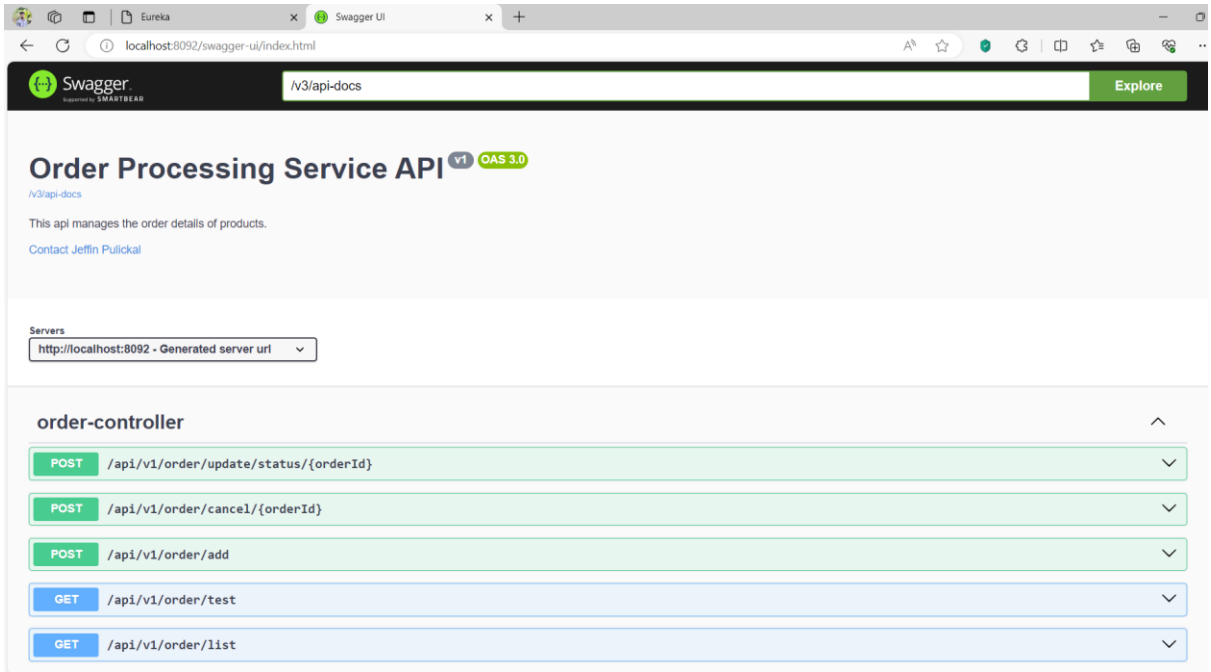
- Product catalogue service - http://localhost:8091/swagger-ui/index.html



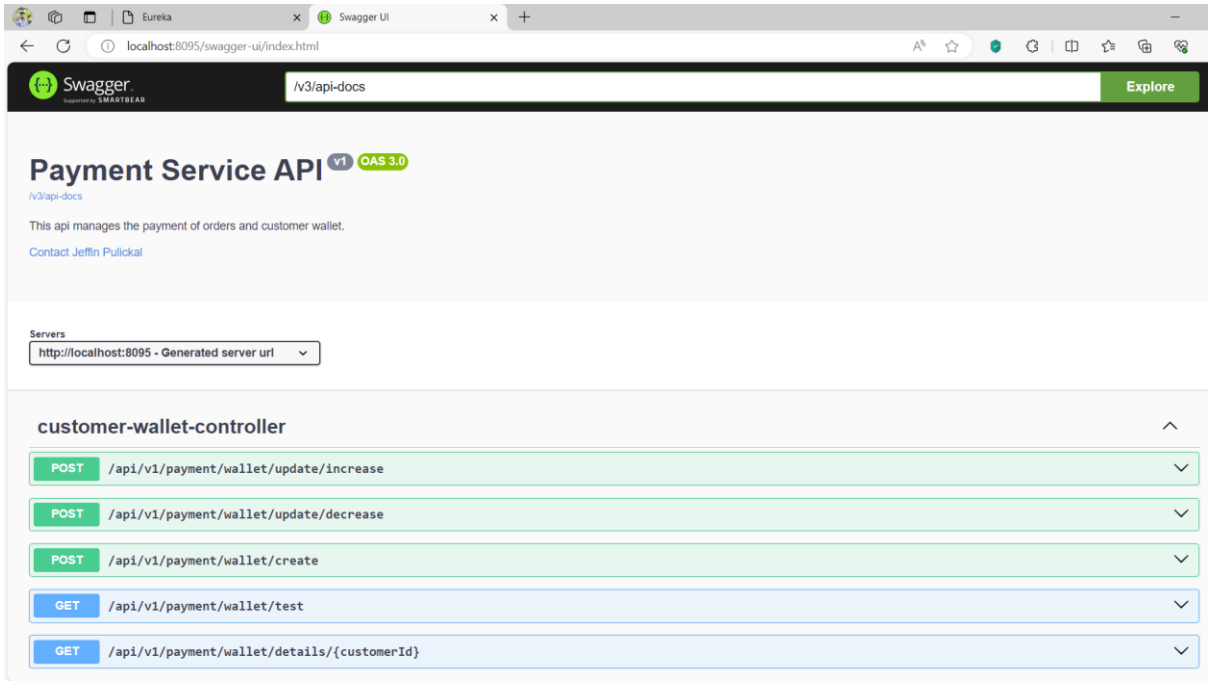
- Inventory service - http://localhost:8094/swagger-ui/index.html



- Order processing service - http://localhost:8092/swagger-ui/index.html



- Payment service - http://localhost:8095/swagger-ui/index.html



4) Run the zipkin server. Give command 'java -jar <zipkinjar>'. The zipkin server will be accessible at http://localhost:9411/zipkin/.
5) Run kafka.
    - Run zookeeper
      .\bin\windows\zookeeper-server-start.bat .\config\zookeeper.properties
    - Start kafka server
      .\bin\windows\kafka-server-start.bat .\config\server.properties
    - Creata a kafka topic named 'ecomm-order-status'.

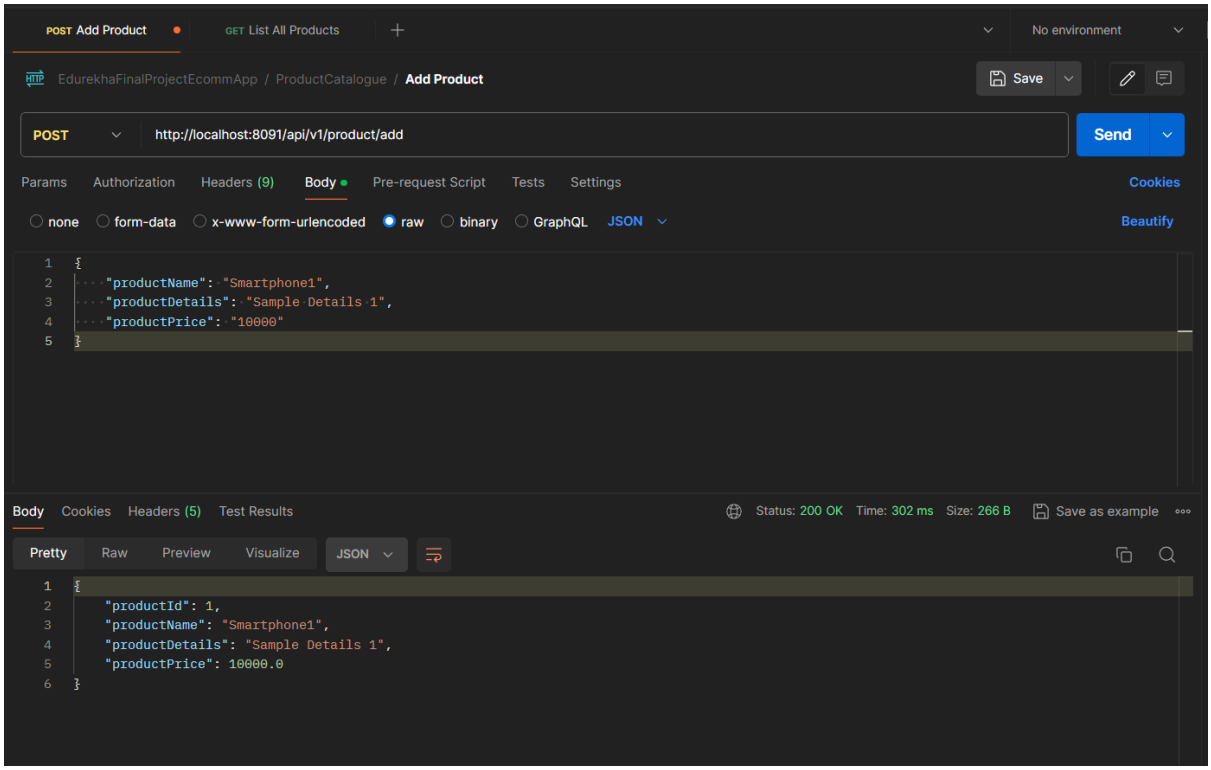
kafka-topics.bat --create --bootstrap-server localhost:9092 --replication-factor 1 --partition 1 --topic ecomm-order-status

- Verify kafka topic

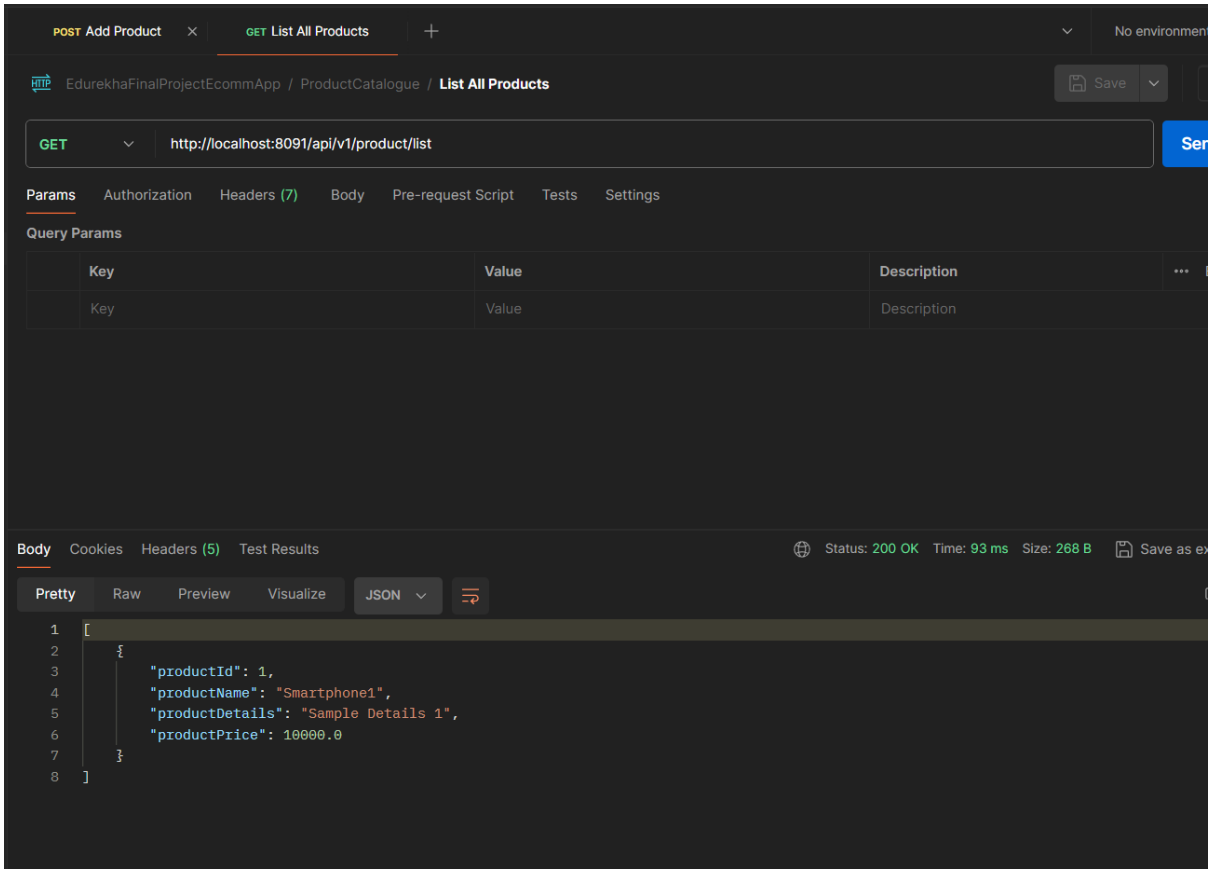
```
C:\kafka\bin\windows>kafka-topics.bat --list --bootstrap-server localhost:9092
__consumer_offsets
ecomm-order-status
test

C:\kafka\bin\windows>
```

Sample Business Flow

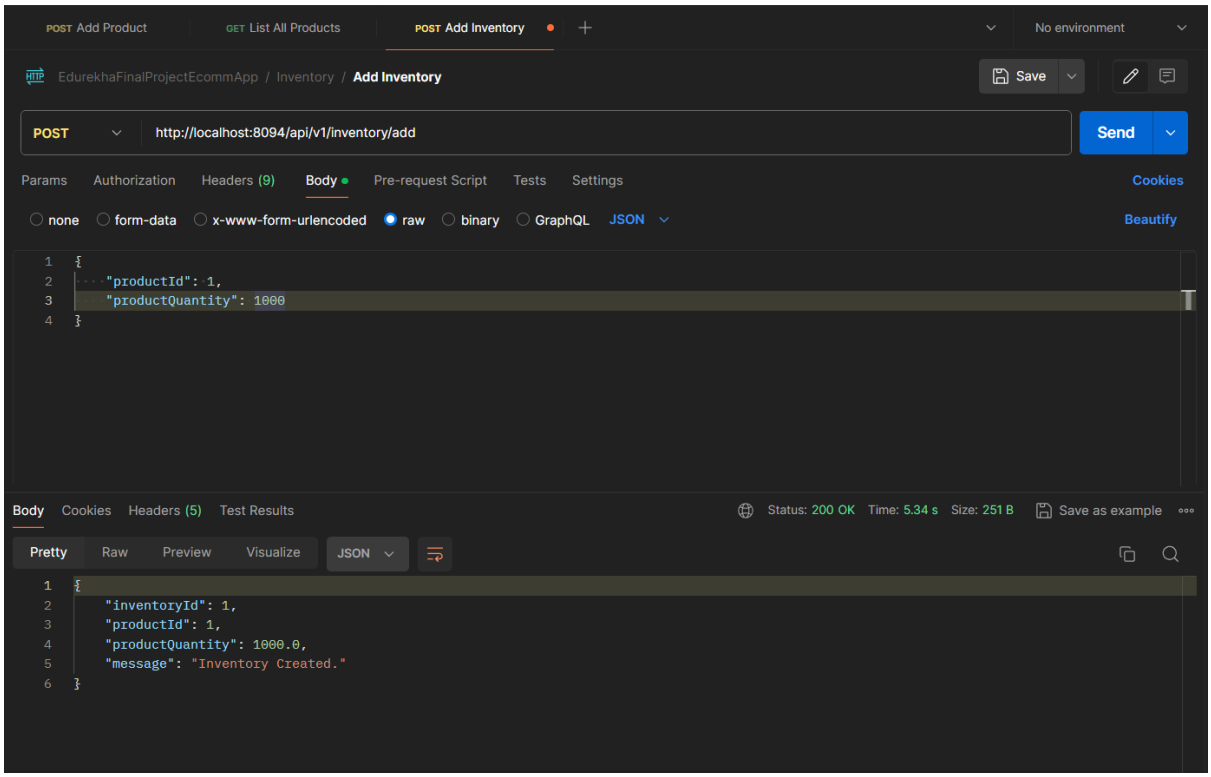
1) Create a product.



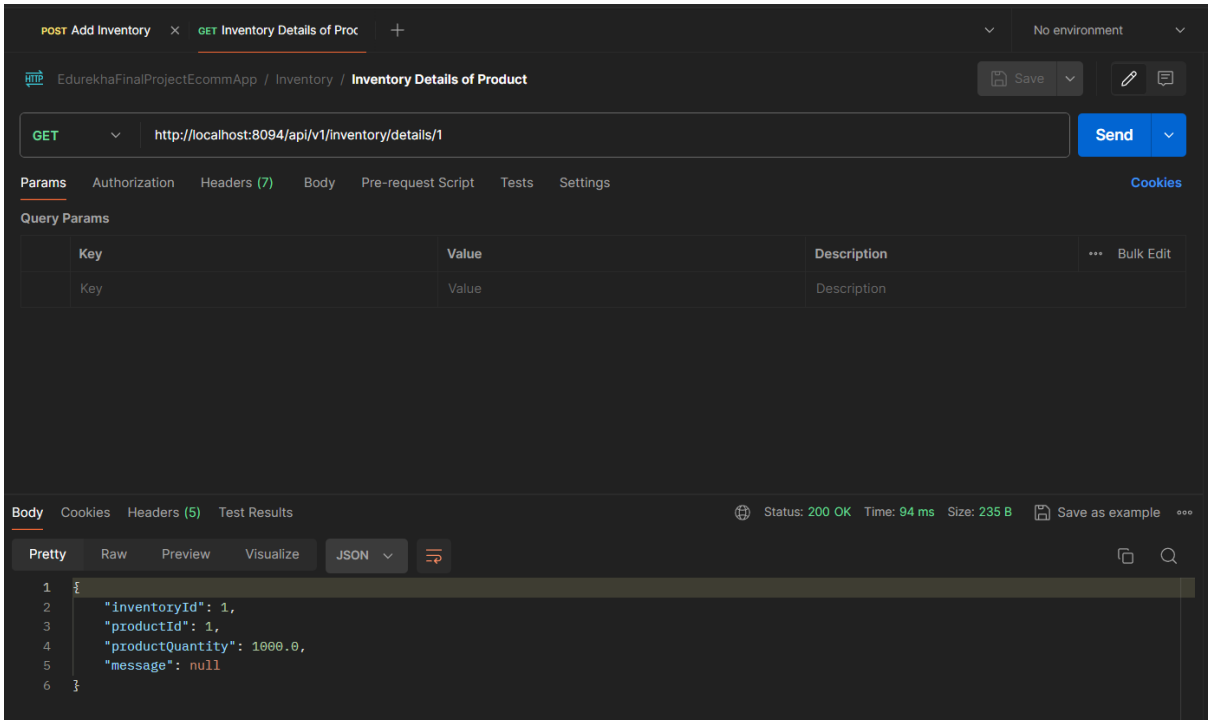
2) Verify the product is created by listing out the products in the DB.



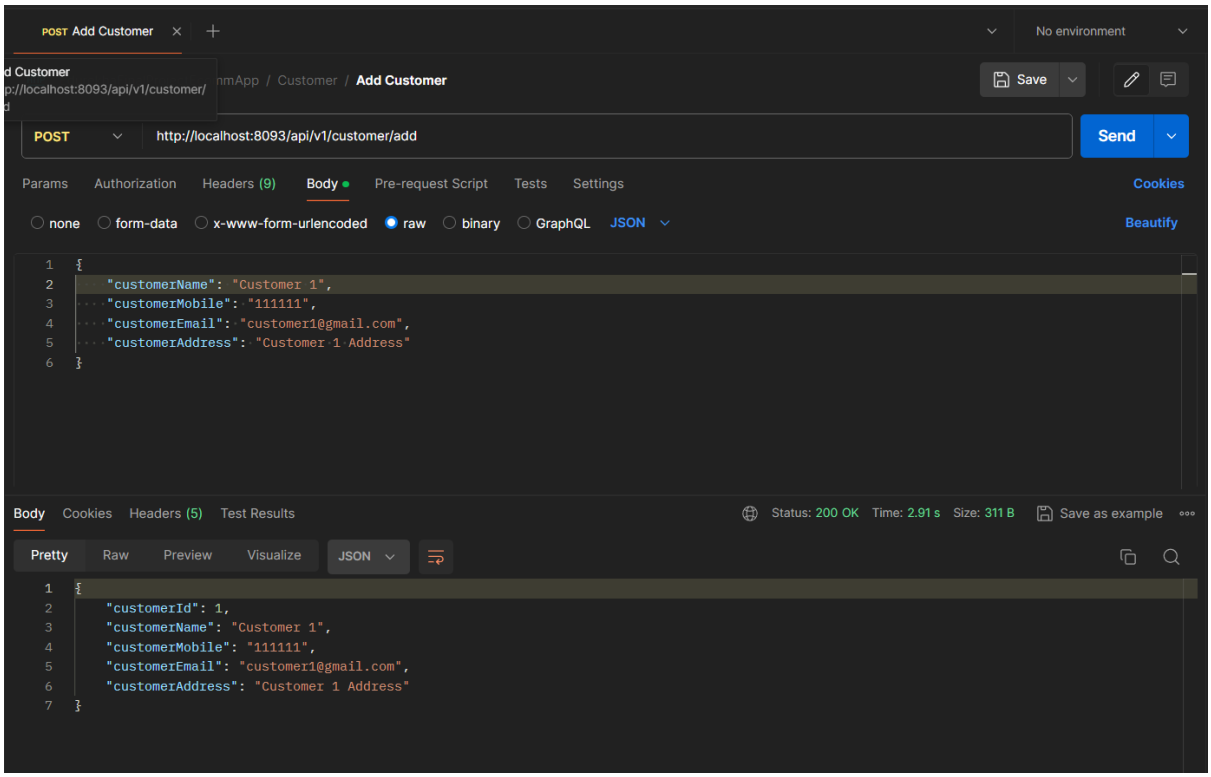
3) Create an inventory for the product.



4) Verify the inventory details.



5) Create a customer profile.



6) Verify the customer details.

POST Add Customer | GET Customer Details | No environment

EdurekhaFinalProjectEcommApp / Customer / **Customer Details**

GET http://localhost:8080/ecomm-customer/api/v1/customer/details/1 Send

Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies

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

This request does not have a body

Body Cookies Headers (3) Test Results Status: 200 OK Time: 405 ms Size: 263 B Save as example

Pretty Raw Preview Visualize JSON


```
{
    "customerId": 1,
    "customerName": "Customer 1",
    "customerMobile": "111111",
    "customerEmail": "customer1@gmail.com",
    "customerAddress": "Customer 1 Address"
}
```


7) Create a customer payment wallet.

POST Create Wallet | No environment

EdurekhaFinalProjectEcommApp / Payment / **Create Wallet**

POST http://localhost:8095/api/v1/payment/wallet/create Send

Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies

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

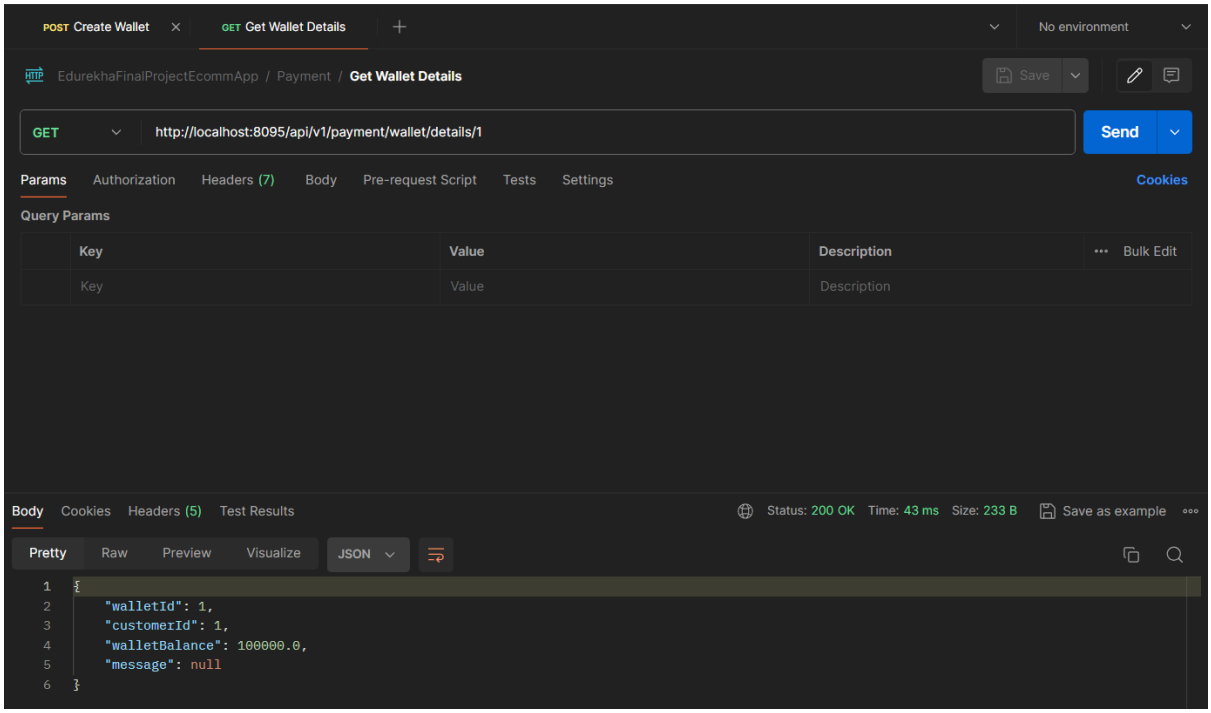

```
{
    "customerId": 1,
    "walletBalance": 100000
}
```


Body Cookies Headers (5) Test Results Status: 200 OK Time: 2.56 s Size: 233 B Save as example

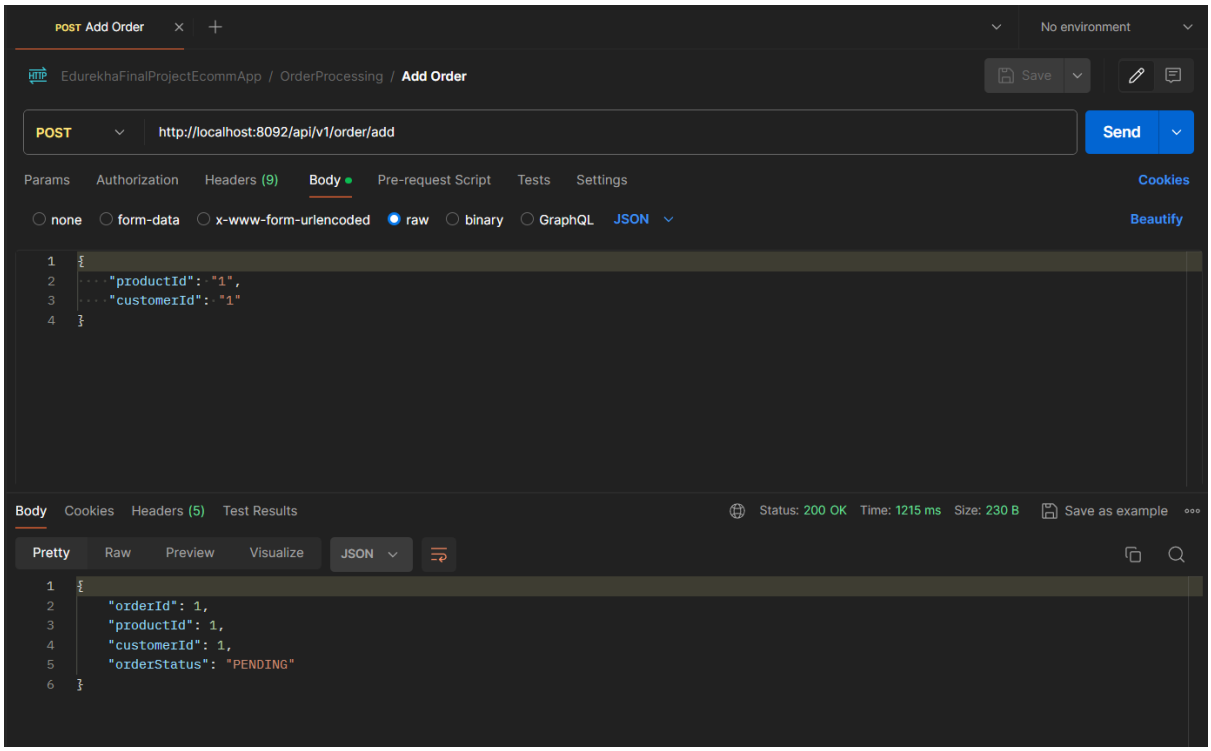
Pretty Raw Preview Visualize JSON


```
{
    "walletId": 1,
    "customerId": 1,
    "walletBalance": 100000.0,
    "message": null
}
```


8) Verify the available wallet balance of the customer.



9) Create an order for customer – Customer1 places order for Product1.



10) Verify the order details. The order will be in PENDING status until a payment is done.

POST Add Order | GET List All Orders | No environment

EdurekhaFinalProjectEcommApp / OrderProcessing / **List All Orders**

GET http://localhost:8092/api/v1/order/list

Send

Params | Authorization | Headers (7) | Body | Pre-request Script | Tests | Settings | Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body | Cookies | Headers (5) | Test Results — Status: 200 OK Time: 397 ms Size: 463 B — Save as example

Pretty | Raw | Preview | Visualize | JSON


```
[
    {
        "customer": {
            "customerId": 1,
            "customerName": "Customer 1",
            "customerMobile": "111111",
            "customerEmail": "customer1@gmail.com",
            "customerAddress": "Customer 1 Address"
        },
        "product": {
            "productId": 1,
            "productName": "Smartphone1",
            "productDetails": "Sample Details 1",
            "productPrice": 10000.0
        },
        "orderStatus": "PENDING"
    }
]
```


11) Make payment for the Order1. Note that all endpoints can also be accessed from the spring cloud gateway server url by appending appropriate service name.

POST Make Order Payment - | No environment

EdurekhaFinalProjectEcommApp / Payment / **Make Order Payment - Gateway Server Url**

POST http://localhost:8080/ecomm-payment/api/v1/payment/order-payment/make-payment

Send

Params | Authorization | Headers (9) | Body | Pre-request Script | Tests | Settings | Cookies

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

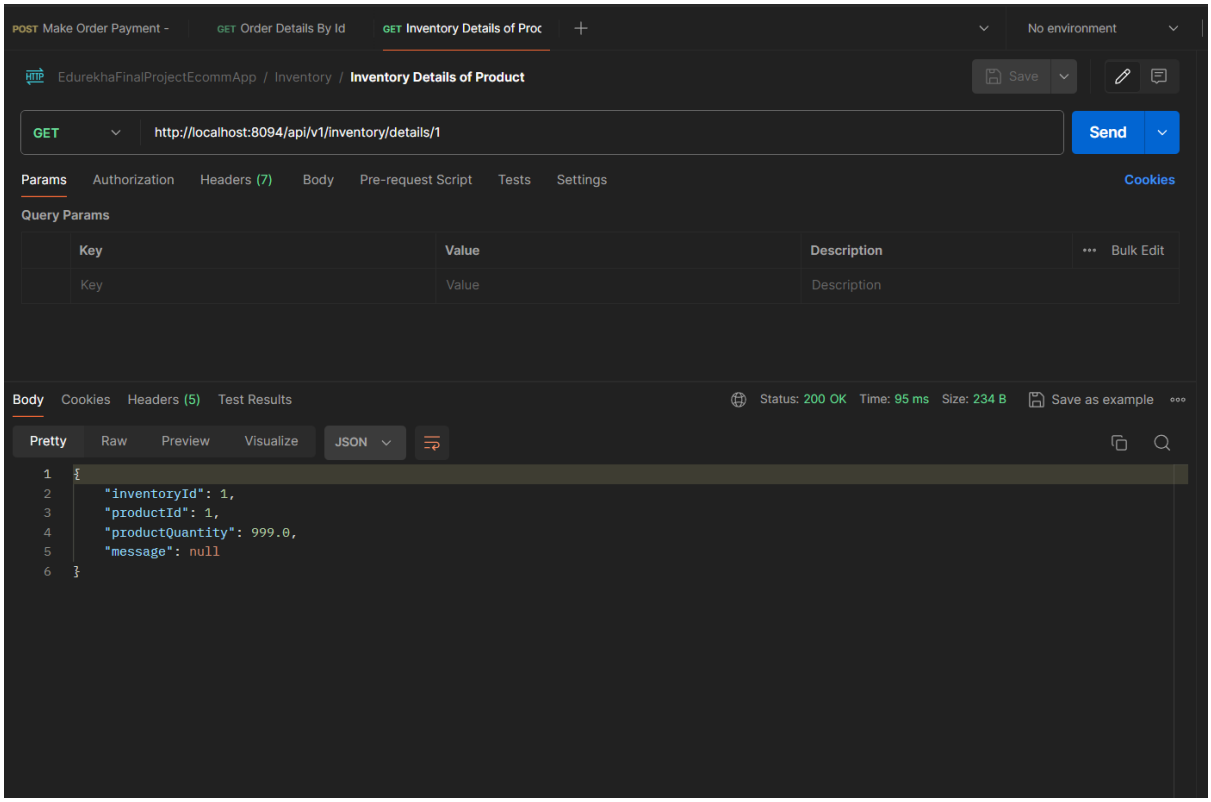

```
{
    "orderId": 1
}
```


Body | Cookies | Headers (3) | Test Results — Status: 200 OK Time: 3.70 s Size: 208 B — Save as example

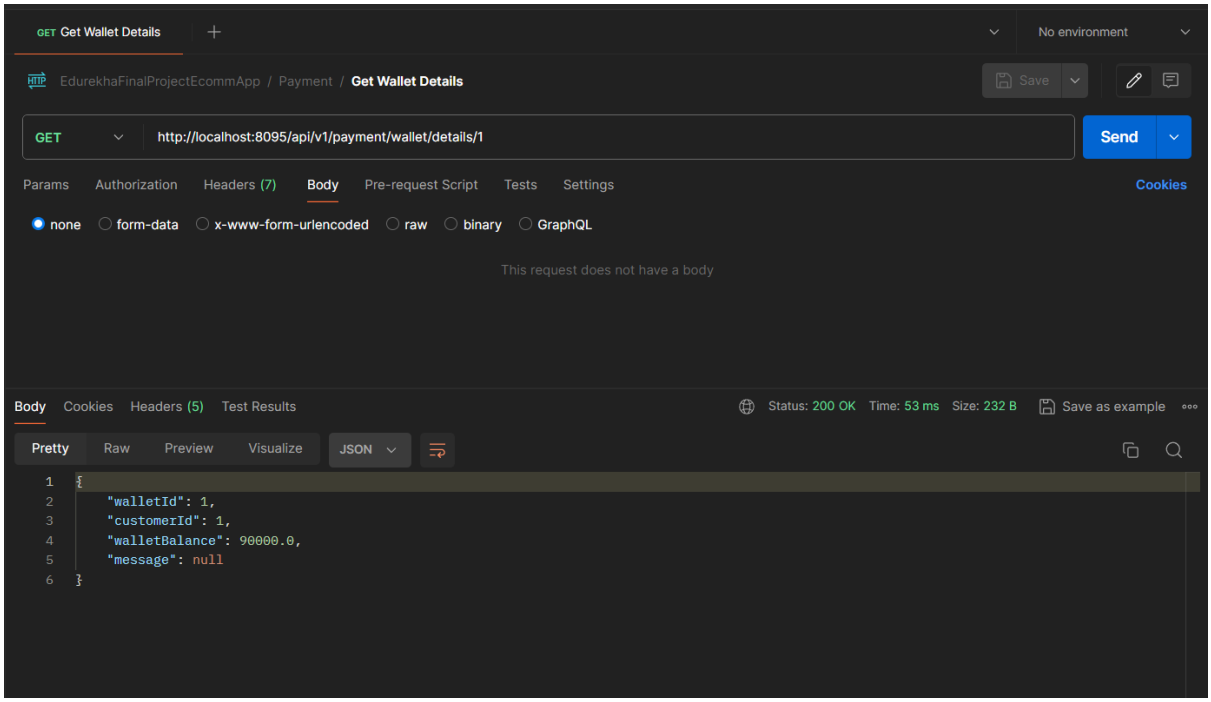
Pretty | Raw | Preview | Visualize | JSON


```
{
    "paymentId": 1,
    "orderId": 1,
    "paymentAmount": 10000.0,
    "paymentStatus": "SUCCESS",
    "message": null
}
```


12) Verify the inventory is reduced by 1.



13) Verify that product price (Rs 10000) is debited from the customer wallet. So the available balance should be Rs 90000.



14) Once payment is success payment microservice will send message to kafka that order is confirmed. This message will be consumed by order processing service which updates the product status as ‘CONFIRMED’.

POST Make Order Payment - | GET Order Details By Id | GET Inventory Details of Proc | No environment

EdurekhaFinalProjectEcommApp / OrderProcessing / **Order Details By Id**

Save

GET http://localhost:8092/api/v1/order/details/1 **Send**

Params | Authorization | Headers (7) | Body | Pre-request Script | Tests | Settings | Cookies

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body | Cookies | Headers (5) | Test Results — Status: 200 OK Time: 135 ms Size: 463 B | Save as example

Pretty | Raw | Preview | Visualize | JSON


```
{
    "customer": {
        "customerId": 1,
        "customerName": "Customer 1",
        "customerMobile": "111111",
        "customerEmail": "customer1@gmail.com",
        "customerAddress": "Customer 1 Address"
    },
    "product": {
        "productId": 1,
        "productName": "Smartphone1",
        "productDetails": "Sample Details 1",
        "productPrice": 10000.0
    },
    "orderStatus": "CONFIRMED"
}
```

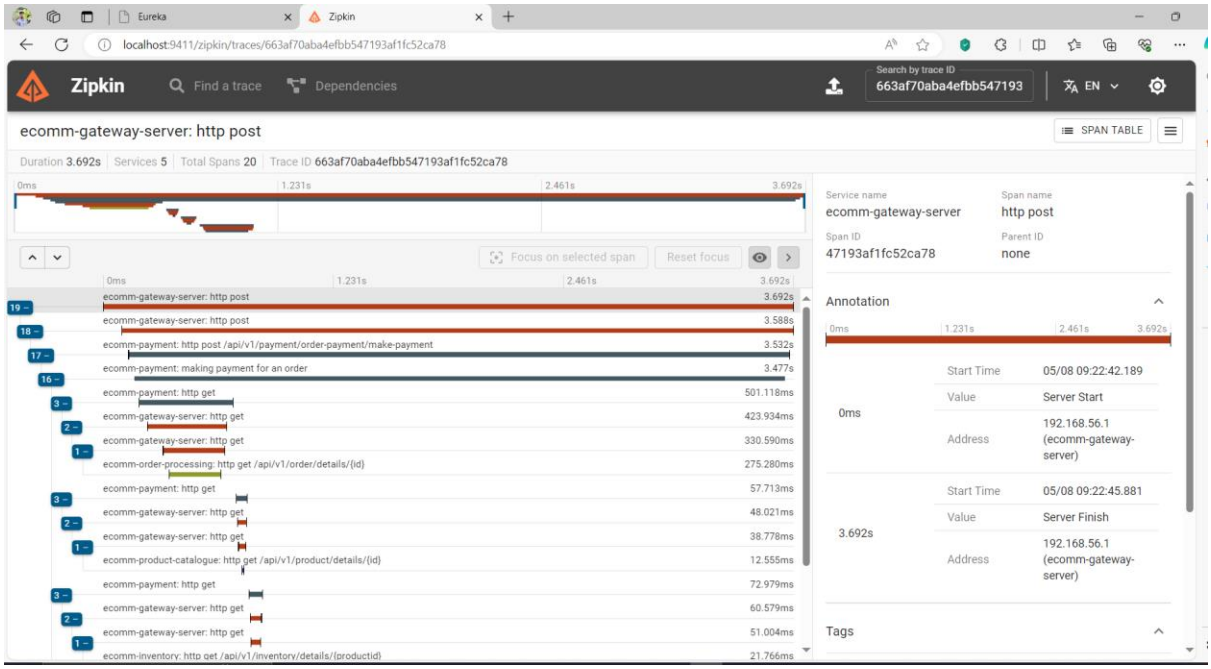

Observability

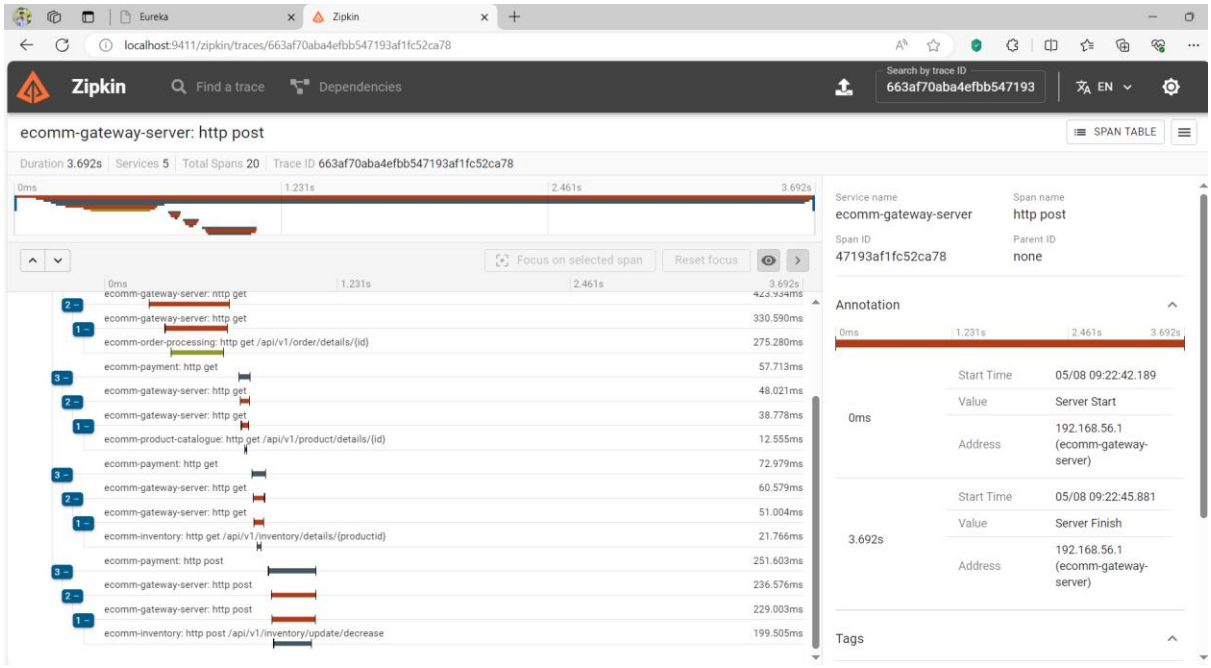
1) The trace id and span id for the above business flow is generated and can be verified in the zipkin server.

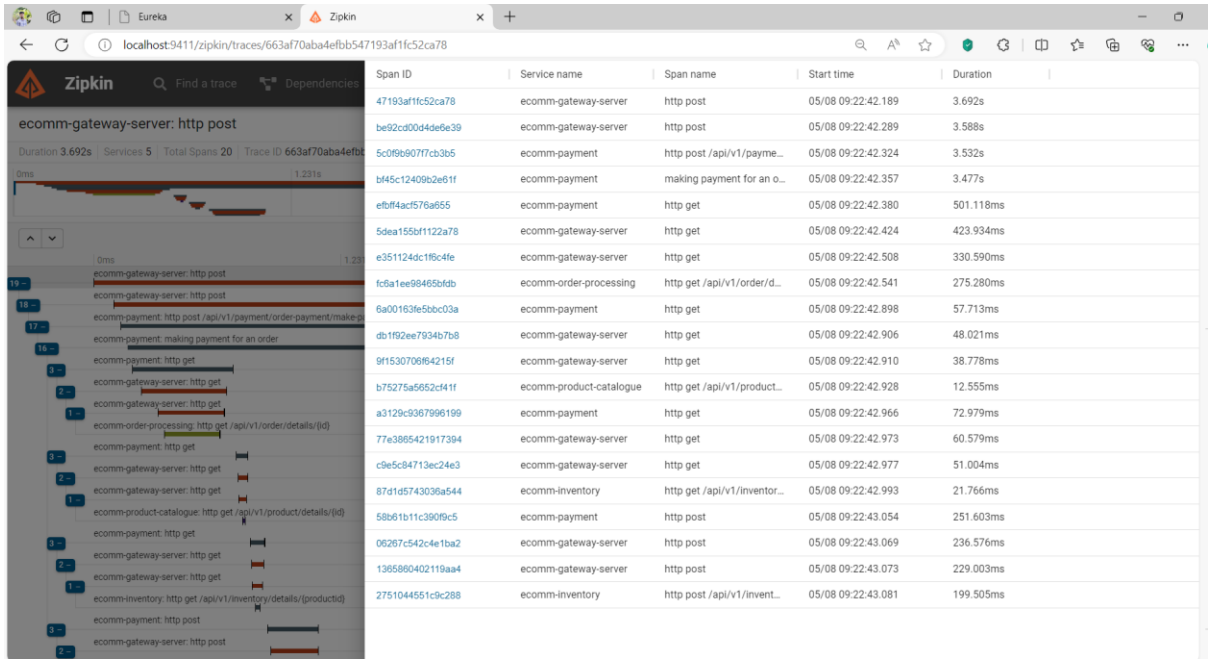
Console | Console | Console | Console | Console | Console | Console

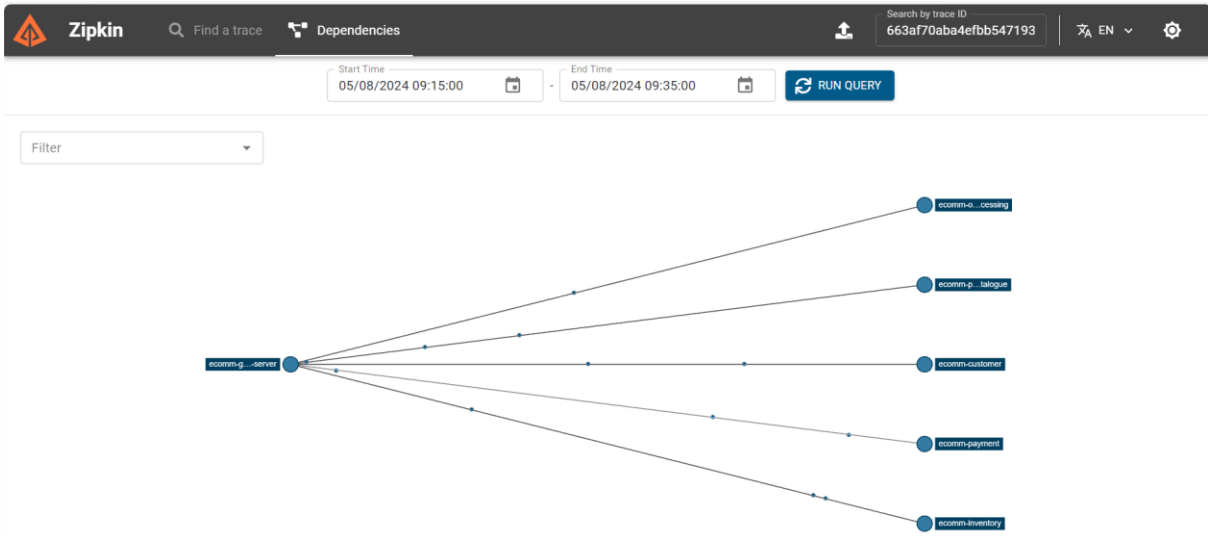
EcommPaymentApplication [Java Application] [pid: 1852]

```
2024-05-08T09:22:42.957+05:30  INFO [ecomm-payment,663af70aba4efbb547193af1fc52ca78,bf45c12409b2e61f] 1852 --- [ecomm-payment] [nio-8095-exec-9] [663af70aba4efbb5
2024-05-08T09:22:42.961+05:30  INFO [ecomm-payment,663af70aba4efbb547193af1fc52ca78,bf45c12409b2e61f] 1852 --- [ecomm-payment] [nio-8095-exec-9] [663af70aba4efbb5
2024-05-08T09:22:43.040+05:30  INFO [ecomm-payment,663af70aba4efbb547193af1fc52ca78,bf45c12409b2e61f] 1852 --- [ecomm-payment] [nio-8095-exec-9] [663af70aba4efbb5
2024-05-08T09:22:43.052+05:30  INFO [ecomm-payment,663af70aba4efbb547193af1fc52ca78,bf45c12409b2e61f] 1852 --- [ecomm-payment] [nio-8095-exec-9] [663af70aba4efbb5
2024-05-08T09:22:43.313+05:30  INFO [ecomm-payment,663af70aba4efbb547193af1fc52ca78,bf45c12409b2e61f] 1852 --- [ecomm-payment] [nio-8095-exec-9] [663af70aba4efbb5
2024-05-08T09:22:43.315+05:30  INFO [ecomm-payment,663af70aba4efbb547193af1fc52ca78,bf45c12409b2e61f] 1852 --- [ecomm-payment] [nio-8095-exec-9] [663af70aba4efbb5
2024-05-08T09:22:43.390+05:30  INFO [ecomm-payment,663af70aba4efbb547193af1fc52ca78,bf45c12409b2e61f] 1852 --- [ecomm-payment] [nio-8095-exec-9] [663af70aba4efbb5
2024-05-08T09:22:43.401+05:30  INFO [ecomm-payment,663af70aba4efbb547193af1fc52ca78,bf45c12409b2e61f] 1852 --- [ecomm-payment] [nio-8095-exec-9] [663af70aba4efbb5
2024-05-08T09:22:43.472+05:30  INFO [ecomm-payment,663af70aba4efbb547193af1fc52ca78,bf45c12409b2e61f] 1852 --- [ecomm-payment] [nio-8095-exec-9] [663af70aba4efbb5
2024-05-08T09:22:43.657+05:30  INFO [ecomm-payment,663af70aba4efbb547193af1fc52ca78,bf45c12409b2e61f] 1852 --- [ecomm-payment] [nio-8095-exec-9] [663af70aba4efbb5
        acks = -1
        auto.include.jmx.reporter = true
        batch.size = 16384
```

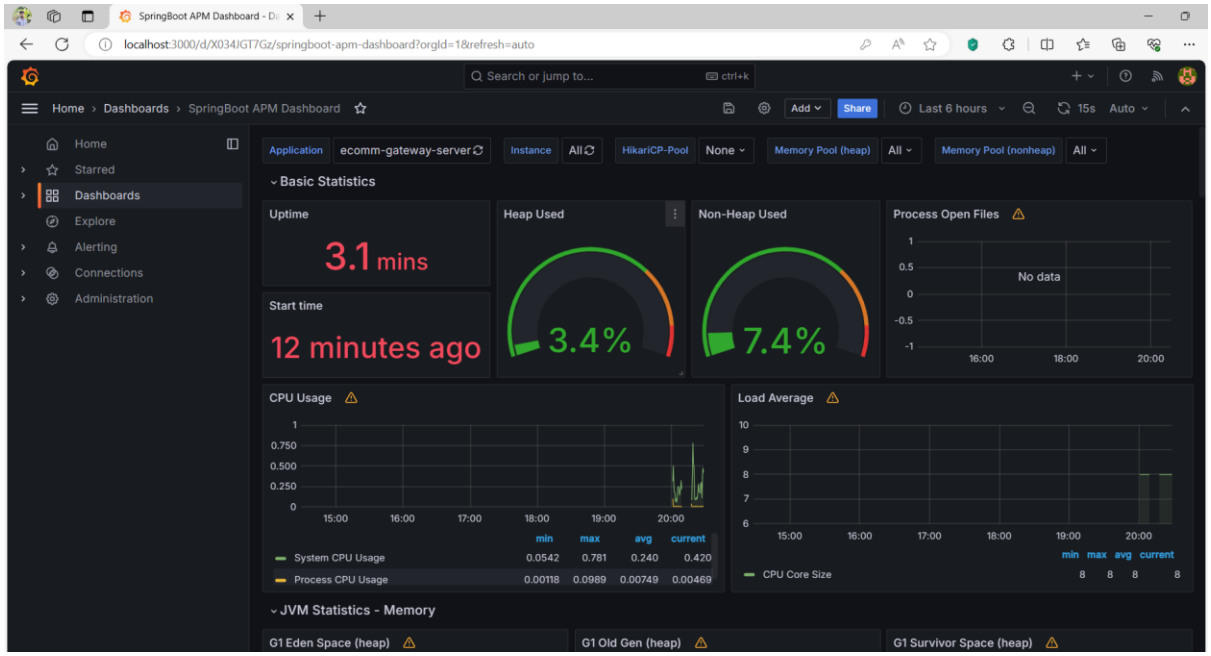

Eureka
Zipkin
localhost:9411/zipkin/traces/663af70aba4efbb547193af1fc52ca78
Zipkin
Find a trace
Dependencies
Search by trace ID
663af70aba4efbb547193
EN
ecomm-gateway-server: http post
SPAN TABLE
Duration 3.692s | Services 5 | Total Spans 20 | Trace ID 663af70aba4efbb547193af1fc52ca78
0ms
1.231s
2.461s
3.692s
Focus on selected span
Reset focus
ecomm-gateway-server: http post 3.692s
ecomm-gateway-server: http post 3.588s
ecomm-payment: http post /api/v1/payment/order-payment/make-payment 3.532s
ecomm-payment: making payment for an order 3.477s
ecomm-payment: http get 501.118ms
ecomm-gateway-server: http get 423.934ms
ecomm-gateway-server: http get 330.590ms
ecomm-order-processing: http get /api/v1/order/details/{id} 275.280ms
ecomm-payment: http get 57.713ms
ecomm-gateway-server: http get 48.021ms
ecomm-gateway-server: http get 38.778ms
ecomm-product-catalogue: http get /api/v1/product/details/{id} 12.555ms
ecomm-payment: http get 72.979ms
ecomm-gateway-server: http get 60.579ms
ecomm-gateway-server: http get 51.004ms
ecomm-inventory: http get /api/v1/inventory/details/{productid} 21.766ms
Service name
ecomm-gateway-server
Span name
http post
Span ID
47193af1fc52ca78
Parent ID
none
Annotation
0ms
Start Time 05/08 09:22:42.189
Value Server Start
Address 192.168.56.1 (ecomm-gateway-server)
3.692s
Start Time 05/08 09:22:45.881
Value Server Finish
Address 192.168.56.1 (ecomm-gateway-server)
Tags

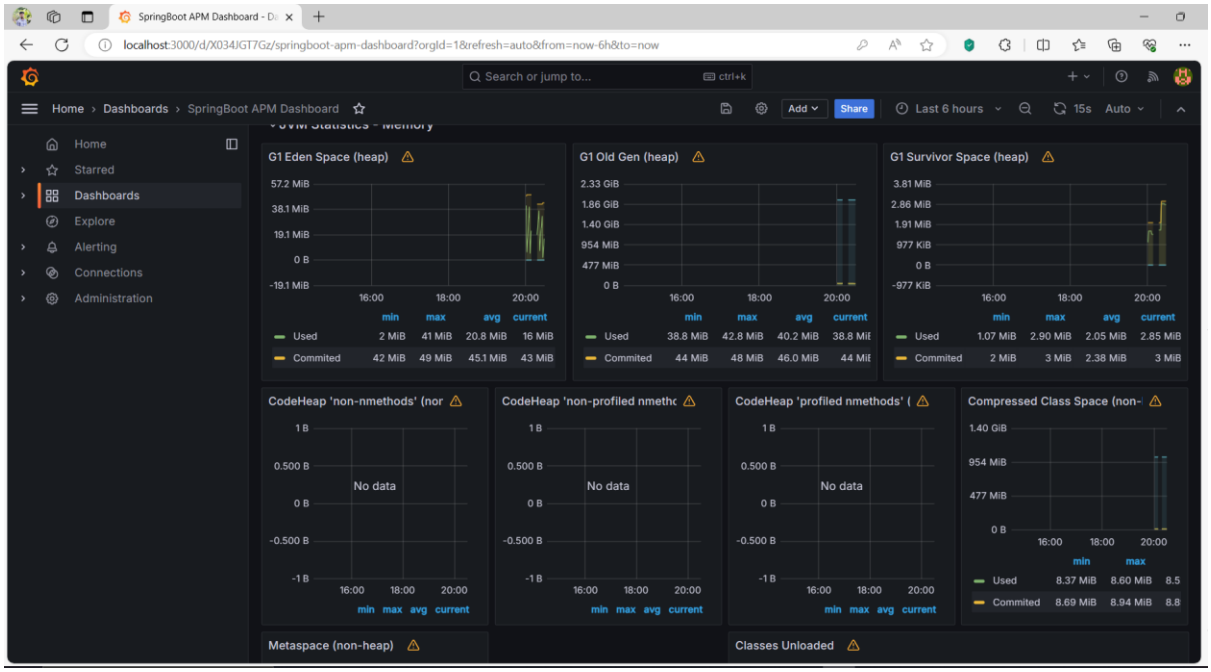

Eureka
Zipkin
localhost:9411/zipkin/traces/663af70aba4efbb547193af1fc52ca78
Zipkin
Find a trace
Dependencies
Search by trace ID
663af70aba4efbb547193
EN
ecomm-gateway-server: http post
SPAN TABLE
Duration 3.692s | Services 5 | Total Spans 20 | Trace ID 663af70aba4efbb547193af1fc52ca78
0ms
1.231s
2.461s
3.692s
Focus on selected span
Reset focus
ecomm-gateway-server: http get 423.934ms
ecomm-gateway-server: http get 330.590ms
ecomm-order-processing: http get /api/v1/order/details/{id} 275.280ms
ecomm-payment: http get 57.713ms
ecomm-gateway-server: http get 48.021ms
ecomm-gateway-server: http get 38.778ms
ecomm-product-catalogue: http get /api/v1/product/details/{id} 12.555ms
ecomm-payment: http get 72.979ms
ecomm-gateway-server: http get 60.579ms
ecomm-gateway-server: http get 51.004ms
ecomm-inventory: http get /api/v1/inventory/details/{productid} 21.766ms
ecomm-payment: http post 251.603ms
ecomm-gateway-server: http post 236.576ms
ecomm-gateway-server: http post 229.003ms
ecomm-inventory: http post /api/v1/inventory/update/decrease 199.505ms
Service name
ecomm-gateway-server
Span name
http post
Span ID
47193af1fc52ca78
Parent ID
none
Annotation
0ms
Start Time 05/08 09:22:42.189
Value Server Start
Address 192.168.56.1 (ecomm-gateway-server)
3.692s
Start Time 05/08 09:22:45.881
Value Server Finish
Address 192.168.56.1 (ecomm-gateway-server)
Tags

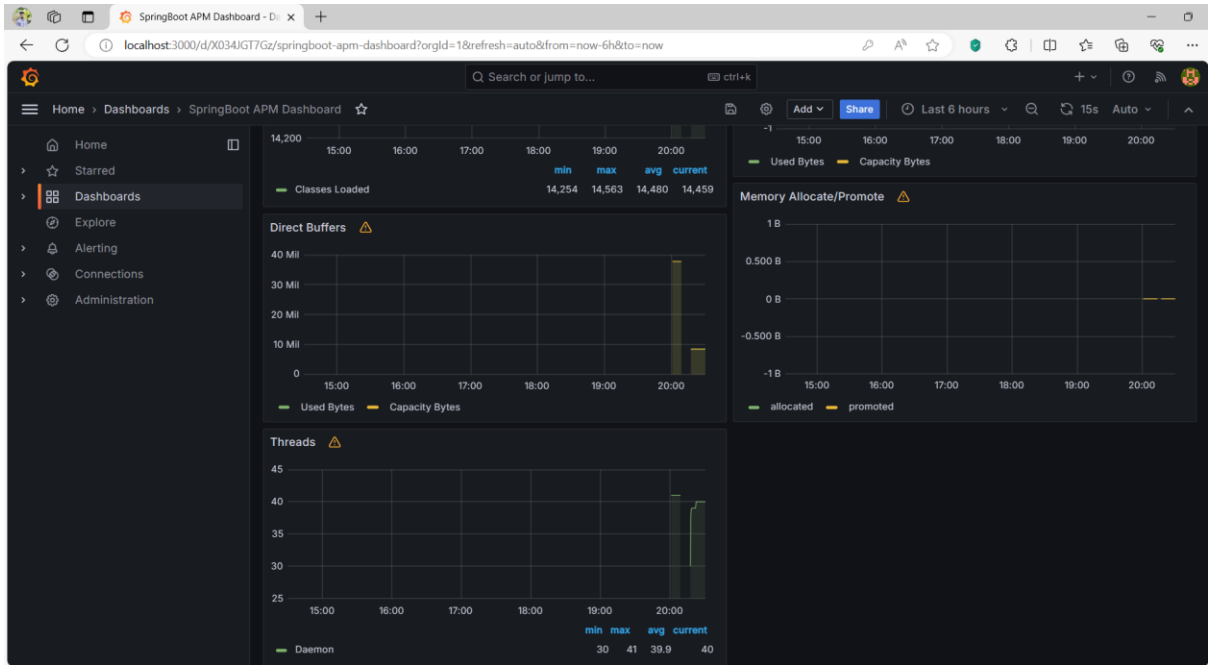




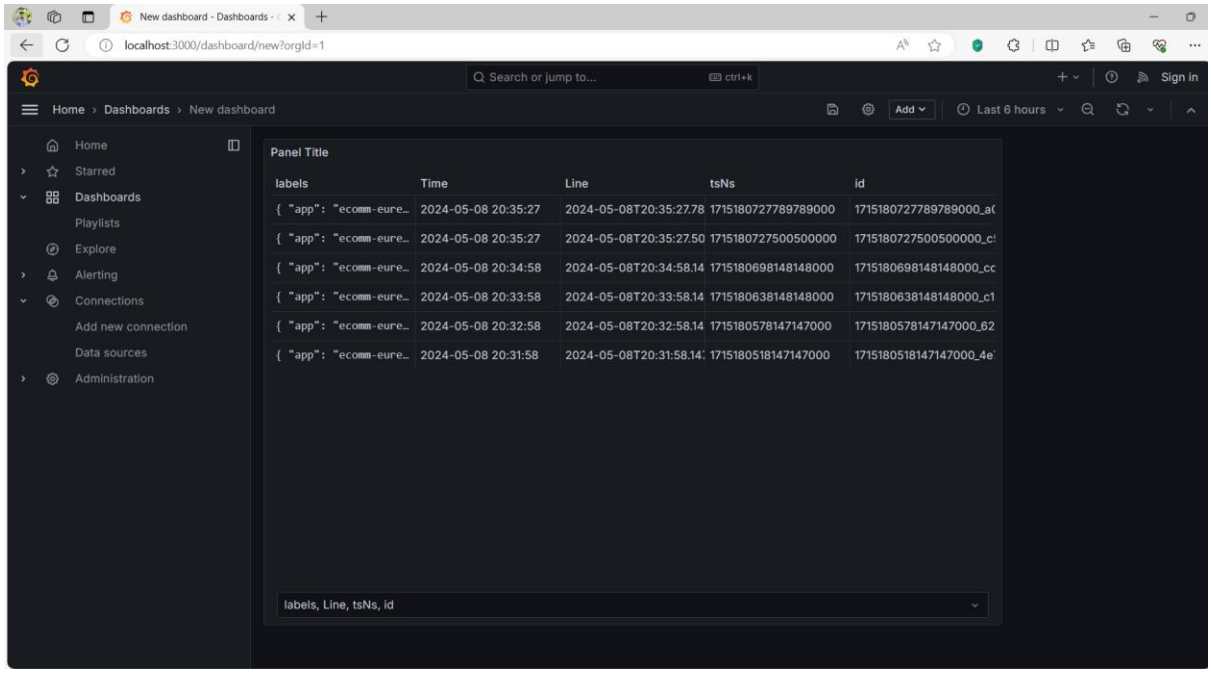
2) Graphana dashboard is configured to show the metrics from Prometheus:


SpringBoot APM Dashboard - D
localhost:3000/d/X034JGT7Gz/springboot-apm-dashboard?orgId=1&refresh=auto
Search or jump to...
ctrl+k
Home › Dashboards › SpringBoot APM Dashboard
Add
Share
Last 6 hours
15s
Auto
Home
Starred
Dashboards
Explore
Alerting
Connections
Administration
Application
ecomm-gateway-server
Instance
All
HikariCP-Pool
None
Memory Pool (heap)
All
Memory Pool (nonheap)
All
Basic Statistics
Uptime
3.1 mins
Start time
12 minutes ago
Heap Used
3.4%
Non-Heap Used
7.4%
Process Open Files
No data
CPU Usage
min max avg current
System CPU Usage 0.0542 0.781 0.240 0.420
Process CPU Usage 0.00118 0.0989 0.00749 0.00469
Load Average
min max avg current
CPU Core Size 8 8 8 8
JVM Statistics - Memory
G1 Eden Space (heap)
G1 Old Gen (heap)
G1 Survivor Space (heap)


SpringBoot APM Dashboard - D
localhost:3000/d/X034JGT7Gz/springboot-apm-dashboard?orgId=1&refresh=auto&from=now-6h&to=now
Search or jump to...
ctrl+k
Home › Dashboards › SpringBoot APM Dashboard
Add
Share
Last 6 hours
15s
Auto
Home
Starred
Dashboards
Explore
Alerting
Connections
Administration
G1 Eden Space (heap)
min max avg current
Used 2 MiB 41 MiB 20.8 MiB 16 MiB
Commited 42 MiB 49 MiB 45.1 MiB 43 MiB
G1 Old Gen (heap)
min max avg current
Used 38.8 MiB 42.8 MiB 40.2 MiB 38.8 MiB
Commited 44 MiB 48 MiB 46.0 MiB 44 MiB
G1 Survivor Space (heap)
min max avg current
Used 1.07 MiB 2.90 MiB 2.05 MiB 2.85 MiB
Commited 2 MiB 3 MiB 2.38 MiB 3 MiB
CodeHeap 'non-nmethods' (nor
No data
CodeHeap 'non-profiled nmethc
No data
CodeHeap 'profiled nmethods' (
No data
Compressed Class Space (non-
min max
Used 8.37 MiB 8.60 MiB 8.5
Commited 8.69 MiB 8.94 MiB 8.8
Metaspace (non-heap)
Classes Unloaded



3) Log aggregation is done using loki. The loki datastore is configured to display app logs in dashboard.



Deployment

1) The app is built using dockerfile and pushed to docker hub.
2) Minikube is used to deploy the application to a Kubernetes cluster in local:

```
C:\Users\jeffi\JeffinData\Work\TCS\Wings2_MicroservicesArchitect\EdurekhaMicroservicesArchitectureCourse_TCS\FinalProjectWorkspace\ecomm-k8s-deployment>kubectl get all
NAME                                              READY   STATUS    RESTARTS       AGE
pod/ecomm-customer-app-55b8457947-b6zw2           1/1     Running   7 (44s ago)    49m
pod/ecomm-inventory-app-7444b8655b-sc2k7          1/1     Running   1 (4m8s ago)   12m
pod/ecomm-order-processing-app-54f48cf78b-zxlb6   1/1     Running   0              12m
pod/ecomm-payment-app-78c45c9975-5df4w            1/1     Running   0              11m
pod/ecomm-product-catalogue-app-5f7b5f79bb-zxhst  1/1     Running   6 (44s ago)    37m
pod/ecommconfig-0                                 1/1     Running   8 (51s ago)    3h28m
pod/ecommeureka-0                                 1/1     Running   8 (109s ago)   175m
pod/ecommgateway-0                                1/1     Running   8 (40m ago)    3h18m

NAME                                  TYPE           CLUSTER-IP      EXTERNAL-IP   PORT(S)          AGE
service/ecomm-customer-svc            ClusterIP      10.98.209.187   <none>        8093/TCP         49m
service/ecomm-inventory-svc           ClusterIP      10.99.128.182   <none>        8094/TCP         12m
service/ecomm-order-processing-svc    ClusterIP      10.111.220.21   <none>        8092/TCP         12m
service/ecomm-payment-svc             ClusterIP      10.99.130.105   <none>        8095/TCP         11m
service/ecomm-product-catalogue-svc   ClusterIP      10.107.244.58   <none>        8091/TCP         37m
service/ecommconfig                   ClusterIP      None            <none>        8090/TCP         3h28m
service/ecommconfig-lb                NodePort       10.100.62.81    <none>        8090:30494/TCP   3h28m
service/ecommeureka                   ClusterIP      None            <none>        8761/TCP         175m
service/ecommeureka-lb                NodePort       10.107.88.224   <none>        8761:30000/TCP   175m
service/ecommgateway                  ClusterIP      None            <none>        8080/TCP         3h18m
service/ecommgateway-lb               LoadBalancer   10.101.199.22   <pending>     8080:30116/TCP   3h18m
service/kubernetes                    ClusterIP      10.96.0.1       <none>        443/TCP          2d3h

NAME                                          READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/ecomm-customer-app            1/1     1            1           49m
deployment.apps/ecomm-inventory-app           1/1     1            1           12m
deployment.apps/ecomm-order-processing-app    1/1     1            1           12m
deployment.apps/ecomm-payment-app             1/1     1            1           11m
deployment.apps/ecomm-product-catalogue-app   1/1     1            1           37m

NAME                                                DESIRED   CURRENT   READY   AGE
replicaset.apps/ecomm-customer-app-55b8457947       1         1         1       49m
replicaset.apps/ecomm-inventory-app-7444b8655b      1         1         1       12m
replicaset.apps/ecomm-order-processing-app-54f48cf78b   1         1         1       12m
replicaset.apps/ecomm-payment-app-78c45c9975        1         1         1       11m
replicaset.apps/ecomm-product-catalogue-app-5f7b5f79bb   1         1         1       37m

NAME                            READY   AGE
statefulset.apps/ecommconfig    1/1     3h28m
statefulset.apps/ecommeureka    1/1     175m
statefulset.apps/ecommgateway   1/1     3h18m

C:\Users\jeffi\JeffinData\Work\TCS\Wings2_MicroservicesArchitect\EdurekhaMicroservicesArchitectureCourse_TCS\FinalProjectWorkspace\ecomm-k8s-deployment>
```