Cognizant | Academy
Business Focused. Learner Centric.

Microservices Architecture

Final Case Study

Instructions

1. Implement the core services **customer** and **Item**
2. Implement the composite service **sales order**
3. Ensure to have implemented the following components
    - Eureka – Discovery service
    - Client side load balancing
    - Hystrix Circuit Breaker
    - Centralized configuration
4. Deploy all the services locally in the VM.

Microservices

- Item service (core)
- Customer service (core)
- Sales order service (composite)

Please take a back-up of the source codes from the VM to avoid losing the content.

Customer Service

1. Get customers – Return all customer details in the table
   Get url: /customers

2. Create a customer by sending in customer details.
   Get url: /customer
   a. When "create customer" method is invoked. Insert the details in the customer table and publish "CustomerCreated" event along with the customer details(customer id, email, firstname, last name).
   b. Sales order service has to subscribe to the "CustomerCreated" event. For further details look into Sales Order Service slide.

Table:
1. Customer – id, email, first_name, last_name

Item Service

1. Get Items – Return all items in the table
   Get url: /items
2. Get a item detail if item name is sent as parameter
   Get url: /items/{itemname}

Table:

1. item – id, name, description, price

Sales Order Service

1. Sales order customer – event subscription

   a. When a "**CustomerCreated**" event is published, sales order service needs to subscribe to it. Fetch the customer details(customer id, email, first name and last name) and insert it into the local customer table.

Table: Customer_SOS (cust_id, cust_first_name, cust_last_name, cust_email)

Sales Order Service

1. Create Order – create an order and return an order id

   Post url: /orders

   Input: Order Description, Order Date, customer id, list of item names

   Output: Order Id, Order Description, Order line items.

   a. validate customer by verifying the table "customer_sos" with cust_id.

   b. validate items by calling item service with item name

   c. create order by inserting the order details in order table and items for the order details in the order_line_item table.

   Table:

   1. **sales_order** – id, order_date, cust_id, order_desc, total_price
   2. **order_line_item** – id, item_name, item_quantity, order_id

Sales Order Service (contd)…

1. Lookup Order By Customer– Get all orders for customer

   Get url: /orders?customerId={customerId}

   Input: Customer Id

   Output: Order Id, Order Desc

   a. validate customer by verifying the table "customer_sos" with cust_id.

   b. retrieve all order summary for customer.

2. Get Order details – Get order deails

   Get url: /orders/{orderId}

   Input: Customer Id

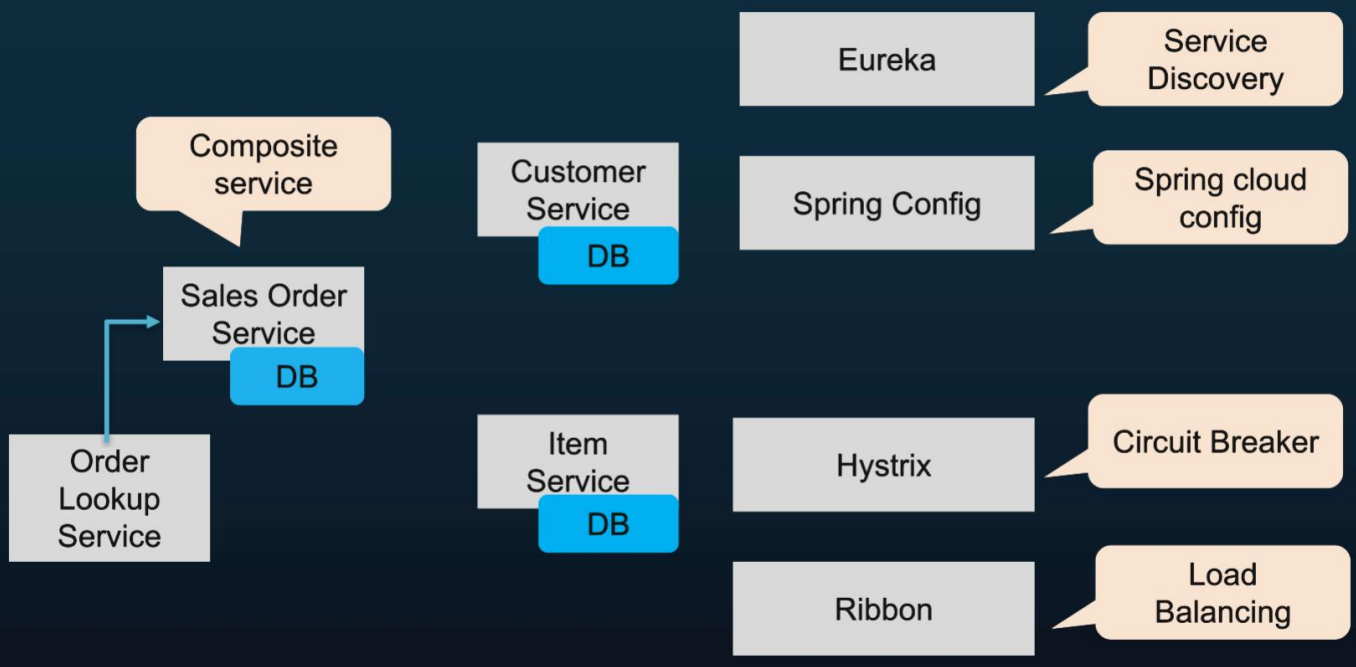
   Output: Order Id, Order Desc, Order line items.

Order Lookup service

search for customer orders(Using Postman).

- Invokes APIs on Sales order service to be able to lookup orders based on customer.
- Fetch all order summaries for a customer in a JSON.
- Pass Order Id and fetch the order details.

MS Architecture to implement



DB – can use H2, MySql, Mongo, Redis

Non-functionals and potential issues

- Provide security for all APIs across all services based on configured username/password (each application and user would have a separate username/password).

- Access another service from a different application might require CORS enabled.

Thank You!