**VENDOR ORDER DOCUMENTATION**

**Technical Documentation**

**<IBM-Vendor Order>**

**Company Name**

**S3 Innovate**

**Date**

**31/08/2023**

Table of Contents

[Introduction: 3](#_Toc144484757)

[Solution Overview: 3](#_Toc144484758)

[System Architecture: 3](#_Toc144484759)

[ Python-Kafka-Client: 3](#_Toc144484760)

[ VendorOrder-Service: 4](#_Toc144484761)

[o JSON Service: 4](#_Toc144484762)

[o PDF Service: 4](#_Toc144484763)

[ DHL-Web-VendorOrder: 4](#_Toc144484764)

[ Database: 4](#_Toc144484765)

[Security & privacy: 5](#_Toc144484766)

[Performance and Scalability: 5](#_Toc144484767)

[Stability: 6](#_Toc144484768)

# Introduction:

The following documentation is based on IBM vendor order technical architecture that provides an overview of the application flow. It aims to guide developers and clients in understanding the application design for the IBM architecture.

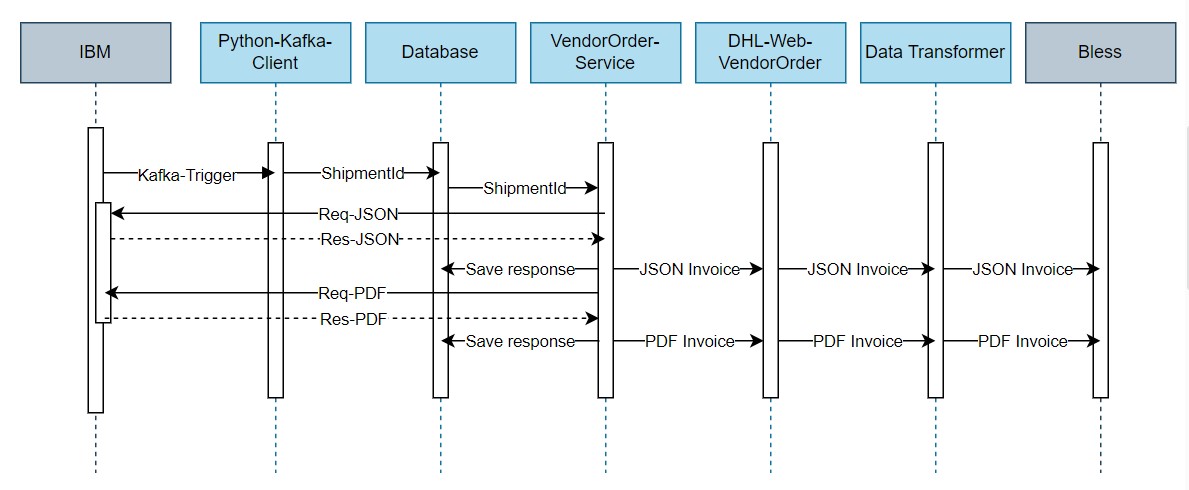
# Solution Overview:

The IBM Vendor Order application's core functionality involves retrieving data from IBM, performing transformations in accordance with the Bless system's requirements, and subsequently forwarding it to Bless. The data to be processed falls into two categories: JSON data and PDF data. The processing of PDF data is contingent upon the successful completion of the JSON data processing phase.

# System Architecture:

This sequence diagram will give an overview of the flow we are using for the vendor order.

## 



The diagram provides a comprehensive overview of the key components and their interactions within the IBM Vendor Order application:

* Python-Kafka-Client:This application serves as the initial data consumer, fetching data from IBM via Kafka. It receives payloads in the following format.

{

"eventName": "SHIPMENT\_PREPARED",

"shipmentID": "20233641789863",

"docType": "TBD",

"process": "SPO",

"shipToCountryCode": "3-digit code"

}

If the event name is "SHIPMENT\_PREPARED," it extracts the shipment ID and stores it in the database.

* VendorOrder-Service: This application consists of two services. Once is for Json data process and anther is for PDF data process.
  + JSON Service: This component retrieves unprocessed shipment IDs from the database and submits requests to IBM for invoices using these shipment IDs. Upon receiving the response from IBM, the application extracts vital information such as shipment tracking numbers, invoices, and document names. It then transmits this data to the Web-VendorOrder application through an API. If a token is acquired as part of this process, it is subsequently updated in the database to mark the request as processed.
  + PDF Service: This module is responsible for retrieving unprocessed documents. It relies on the JSON invoice data being processed first. Once the JSON data is processed, this service requests PDFs from IBM using the associated document names. After obtaining the PDF response from IBM, it forwards the PDF data to the Web-VendorOrder application via a separate API. Similar to the JSON service, if a token is obtained during this procedure, it is recorded in the database to indicate that the request has been processed.
* DHL-Web-VendorOrder: This application is responsible for receiving data from the VendorOrder service via APIs. It performs data transformations to align with the Bless system's requirements and then forwards the transformed data to Bless. Additionally, the application responds with a token to the caller, which is obtained from the Bless system.
* Database: The database component plays a critical role in maintaining data for processing. The schema for the database includes tables like DT\_IBM\_SHIPMENT, where various data related to the processing workflow is stored. The schema ensures data integrity and uniqueness, facilitating efficient data management.

Below is the schema.

CREATE TABLE dt\_ibm\_shipment (

Id INT AUTO\_INCREMENT PRIMARY KEY,

shipment\_id VARCHAR(50),

createddate DATETIME DEFAULT CURRENT\_TIMESTAMP,

COMINV\_status BIT DEFAULT 0,

COMINV\_proces\_date DATETIME,

COMINV\_retry\_count INT DEFAULT 0,

shipment\_traking\_no VARCHAR(255),

invoice\_number VARCHAR(255),

file\_names VARCHAR(255),

COMINV\_DOC\_status BIT DEFAULT 0,

COMINV\_DOC\_proces\_date DATETIME,

COMINV\_DOC\_retry\_count INT DEFAULT 0

);

ALTER TABLE dt\_ibm\_shipment

ADD CONSTRAINT shipment\_id\_unique UNIQUE (shipment\_id);

# Security & privacy:

To bolster security measures and enforce access control for unauthorized users, the Web-VendorOrder application implements JWT (JSON Web Token) authentication. This robust authentication method is applied to every request, serving as a means of identifying and validating the legitimacy of the user making the request.

# Performance and Scalability:

The IBM Vendor Order system is designed around a microservices architecture, promoting efficiency and scalability. Key components like the Python-Kafka-Client microservice are specialized for distinct tasks, such as ingesting data from IBM and storing it in the database. This modularity allows for independent scaling of different components based on their resource requirements, ensuring optimal performance.

Additionally, the application incorporates a data retry mechanism to handle any unexpected errors that may occur during processing. Importantly, this mechanism includes limitations to prevent indefinite retry attempts, maintain control over resource utilization.

# Stability:

To enhance reliability, the Vendor Order application leverages containerization through technologies like Docker. This approach isolates microservices, reducing the impact of failures on individual components and enhancing system stability.

The application also prioritizes error handling and logging mechanisms. Comprehensive error handling ensures that issues are captured, while extensive logs provide valuable insights for diagnosing and resolving problems.

It's worth noting that reliability is a continuous pursuit for the application, requiring ongoing monitoring, testing, and refinement to maintain a robust and dependable system.