Java Spring Boot Microservices on Azure

Micro-credential Program

Contents

[Use Case Documentation 4](#_Toc74221850)

[Associate Details 4](#_Toc74221851)

[GitHub Repo 4](#_Toc74221852)

[Steps to Run Applications 4](#_Toc74221853)

[To run applications locally: 4](#_Toc74221854)

[To run applications in Azure as Webapp: 4](#_Toc74221855)

[K-Point Details 4](#_Toc74221856)

[Approach and Implementation Justifications 5](#_Toc74221857)

[Use of Synchronous and Asynchronous Communication 5](#_Toc74221858)

[GoF Patterns 6](#_Toc74221859)

[Creational Design Pattern 6](#_Toc74221860)

[Factory Pattern 6](#_Toc74221861)

[Singleton Pattern 6](#_Toc74221862)

[Template Pattern 6](#_Toc74221863)

[Structural Design Pattern 6](#_Toc74221864)

[Façade Pattern 6](#_Toc74221865)

[Adapter Pattern 6](#_Toc74221866)

[Behavior Design Pattern 6](#_Toc74221867)

[Iterator Pattern 6](#_Toc74221868)

[Microservices Patterns 7](#_Toc74221869)

[Single Responsibility Principal and Decomposition 7](#_Toc74221870)

[API Gateway 7](#_Toc74221871)

[Observability Pattern 7](#_Toc74221872)

[Health Check 7](#_Toc74221873)

[Distributed Tracing 7](#_Toc74221874)

[Cross Cutting Concern Pattern 7](#_Toc74221875)

[Externalized Configurations 7](#_Toc74221876)

[Circuit Breaker Pattern 7](#_Toc74221877)

[Technology Stack & Libraries 8](#_Toc74221878)

[Testing 8](#_Toc74221879)

[Screenshots - Local System Execution 9](#_Toc74221880)

[All Micro services Running Locally without any Errors 9](#_Toc74221881)

[Posting Request in Swagger UI for Order Service 10](#_Toc74221882)

[Circuit Breaker and Retry Functionalities in Order Service 11](#_Toc74221883)

[Data Received in Downstream Systems – Package Service 11](#_Toc74221884)

[Data Received in Downstream Systems – Courier Service 11](#_Toc74221885)

[Retry and Fallback 12](#_Toc74221886)

[Circuit Breaker 12](#_Toc74221887)

[Screenshots – Applications Running in Azure 13](#_Toc74221888)

[Order Service with Swagger 13](#_Toc74221889)

[Order Service with API Gateway 14](#_Toc74221890)

[Screenshots - Azure Application Logs 15](#_Toc74221891)

[Order Service 15](#_Toc74221892)

[Package Service 15](#_Toc74221893)

[Courier Service 16](#_Toc74221894)

# Use Case Documentation



# Associate Details

|  |  |
| --- | --- |
| Employee ID | 413643 |
| Employee Name | Linesh Mehta |

# GitHub Repo

<https://github.com/lineshmehta5/cognizant-azure-mc-program>

# Steps to Run Applications

There are a total of 3 applications namely:

* Order Service
* Package Service
* Courier/Shipping Service

All the applications are Java/SpringBoot applications and use Maven for building.

### To run applications locally:

* Uncomment **server.port** property in application.properties
* Run **mvn clean install**

### To run applications in Azure as Webapp:

* mvn com.microsoft.azure:azure-webapp-maven-plugin:1.14.0:config
* mvn azure-webapp:deploy

# K-Point Details

|  |  |
| --- | --- |
| Video Title | 413643-LineshMehta-CloudNativeMSDevelopment-SpringBootAzure |
| Video Link | <https://cognizant.kpoint.com/app/video/gcc-6240fcde-0084-4801-bd1e-9c7c463ace0b?list=my> |

# Approach and Implementation Justifications

### 12 Factor App

*The microservices/apps have been designed using the 12 Factor principals.*

# Use of Synchronous and Asynchronous Communication

### Queues

### HTTP Rest Calls

# GoF Patterns

## Creational Design Pattern

### Factory Pattern

In the solution for this use case, we have extensively used the Factory Pattern using Spring and Java for managing our boiler plate functionalities like Interfaces, Class Extensions and Wrappers (Long, Boolean, valueOf etc.)

### Singleton Pattern

In the solution for this use case, we have used Spring driven Singleton Pattern for all autowired beans.

### Template Pattern

In the solution for this use case, we have used Spring driven template pattern for API and Queue calls i.e. RestTemplate and JMSTemplate

## Structural Design Pattern

### Façade Pattern

*The Order Service Controller is implemented as the Façade Pattern as it collates the interaction with the downstream systems through it being the single point of entry for the transaction.*

### Adapter Pattern

*The Order Service Layer is implemented using the Adapter Pattern as it manages the interaction between the Order Service and Package Service and does the required data translation.*

## Behavior Design Pattern

### Iterator Pattern

*For Iterating through the Order/Items Data across all the 3 services*

### Decorator Pattern

*For implementing Circuit Breaker Pattern*

# Microservices Patterns

## Single Responsibility Principal and Decomposition

Microservices is all about making services loosely coupled and I have achieved this by applying the single responsibility principle and decomposed the services based on their business capability. Hence, I designed 3 independent microservices i.e., Order Service, Package Service and Shipping Service.

## API Gateway

The API Gateway pattern helps resolve issues related to data transformations/protocols and consumer channels. By using Azure API Gateway – it becomes a single point of entry for the Order Service and hence abstracts the producer details allowing client or channel-based transformation of data/protocol and offloading functionalities like security.

## Observability Pattern

### Health Check

Implemented Health Checks using Spring Boot Actuator for the monitoring of API health.

### Distributed Tracing

Implemented Distributed Tracing using Sleuth for tracing a request end-to-end to troubleshoot problems.

## Cross Cutting Concern Pattern

### Externalized Configurations

Implemented this using this through Azure App Service Configurations

### Circuit Breaker Pattern

Implemented this using Resilience4J APIs.

# Technology Stack & Libraries

* JDK 11
* Spring Boot 2.5 and Standard Starters
* Spring Boot Actuator
* Spring Retry
* Resilence4J Circuit Breaker
* Swagger 2.9
* Apache Commons
* Lombok
* Azure JMS Starter
* Azure SQL Starter

# Testing

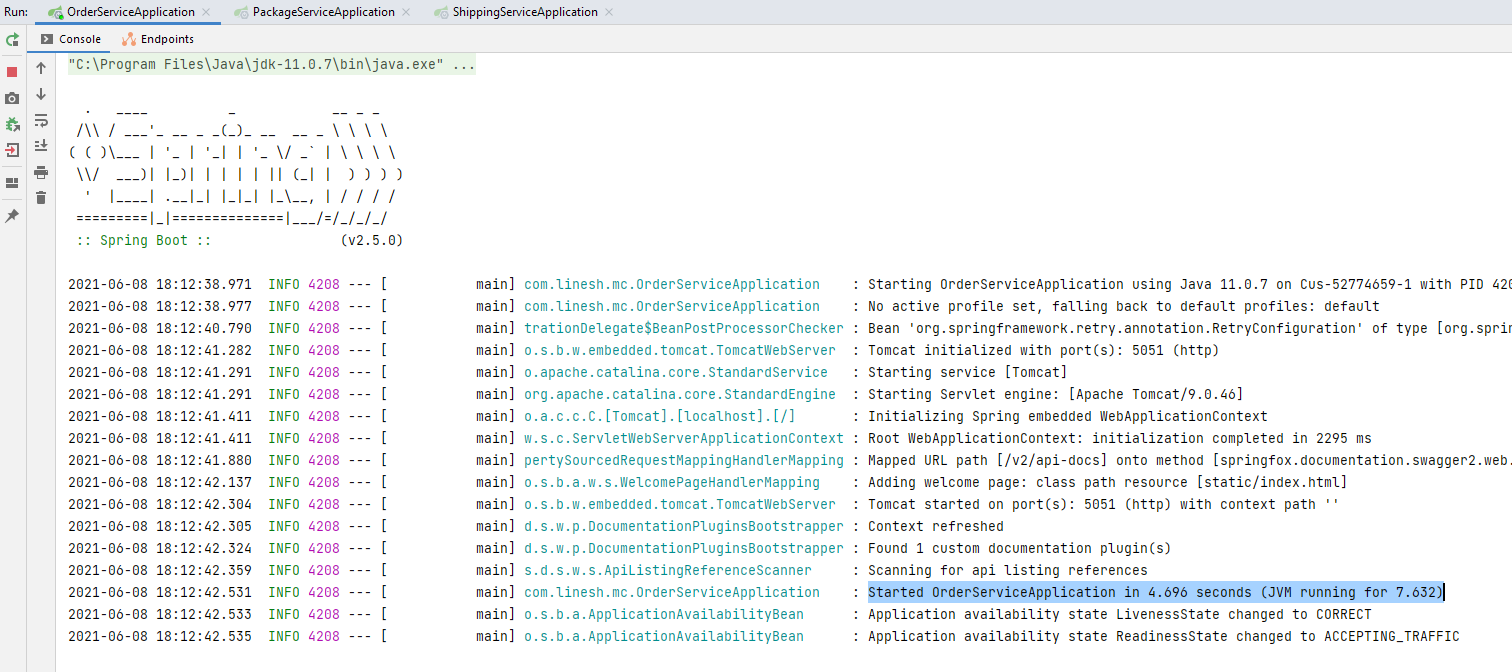
*JUnits with Mocking Framework*

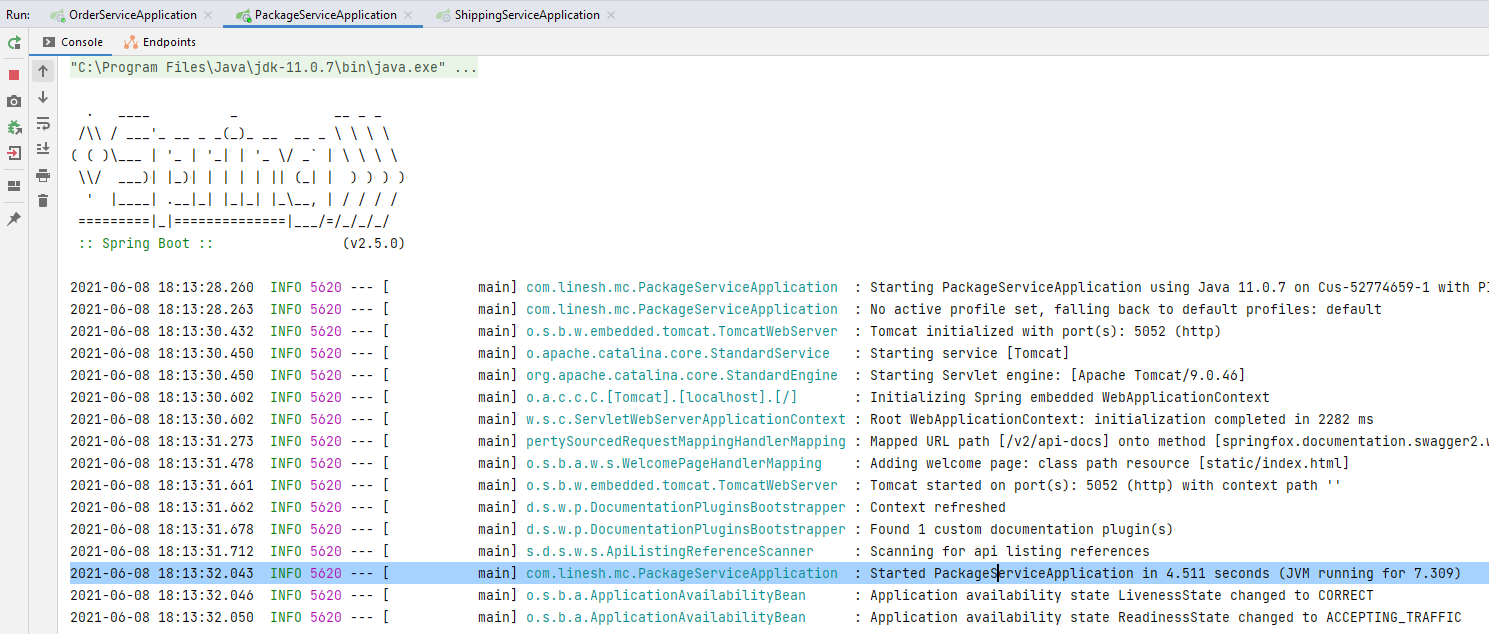
# Code Quality

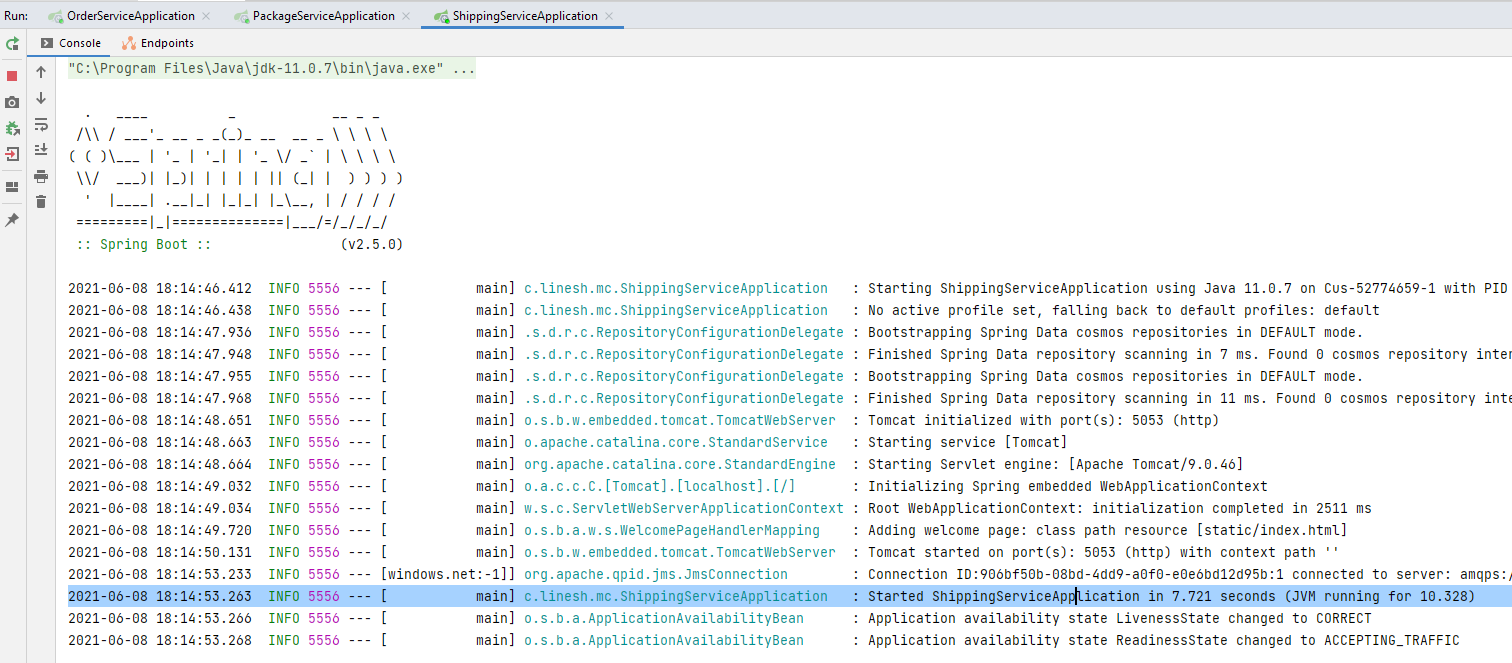
*Static code analysis using Sonar*

# Screenshots - Local System Execution

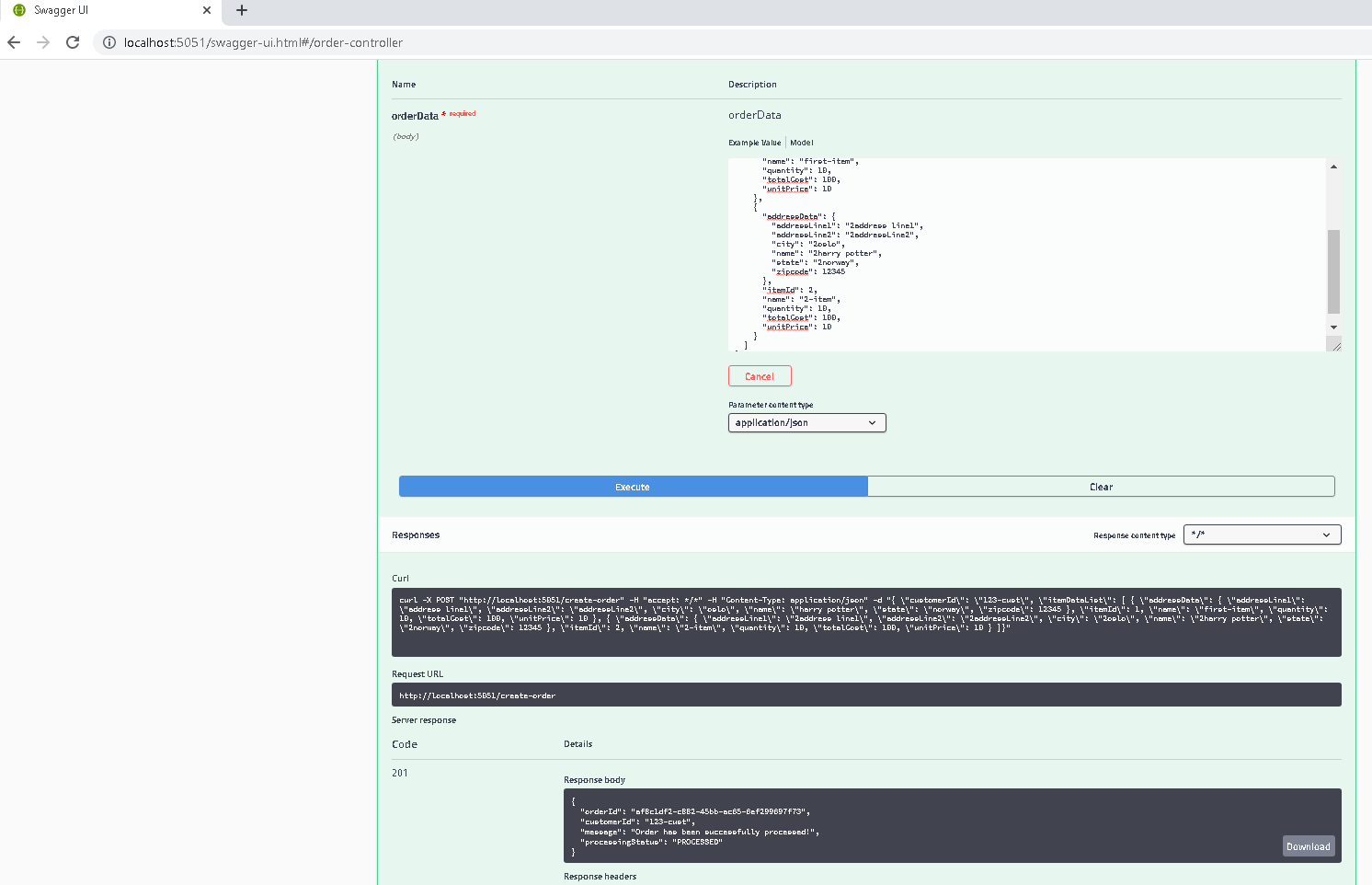
### All Micro services Running Locally without any Errors



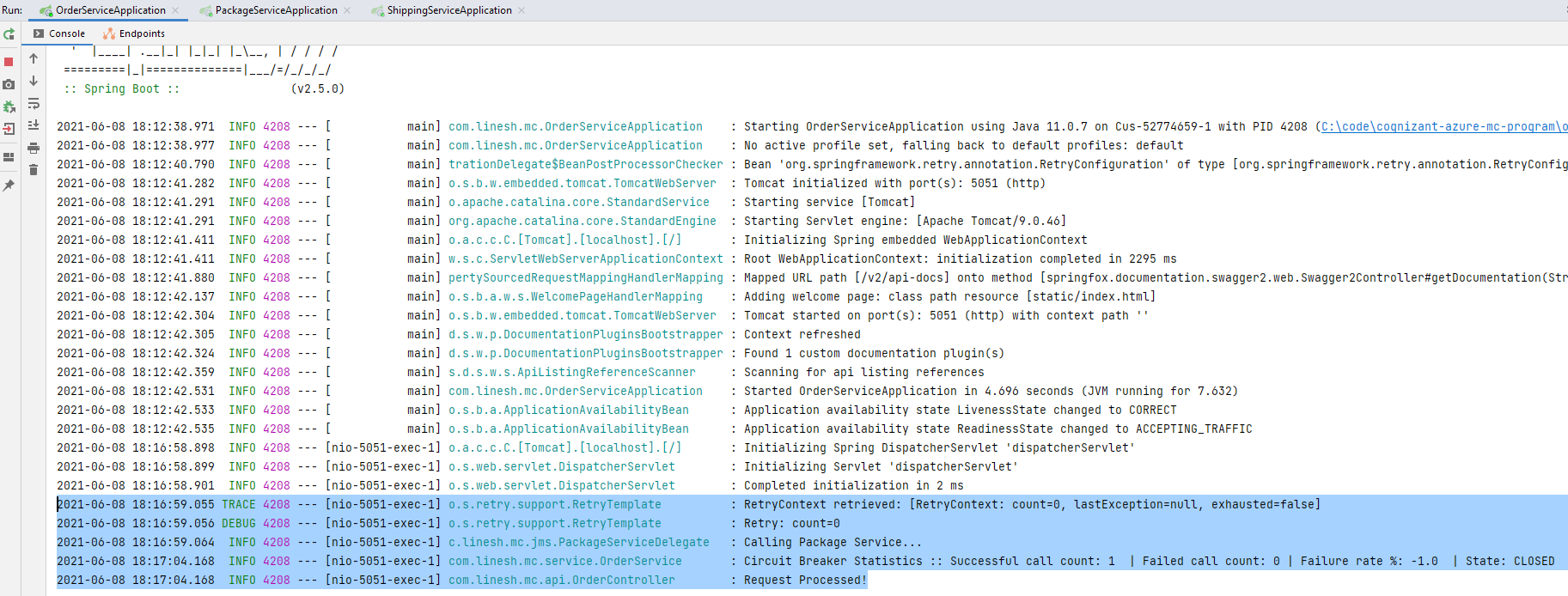




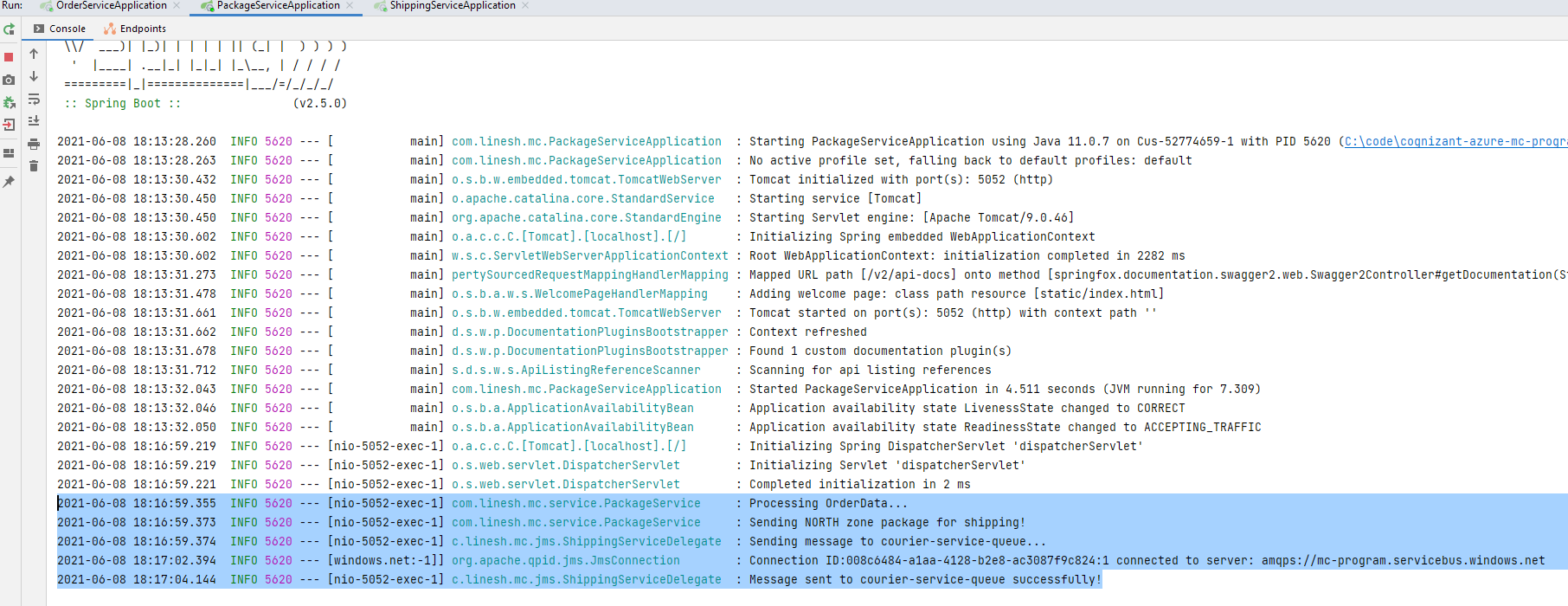
### Posting Request in Swagger UI for Order Service



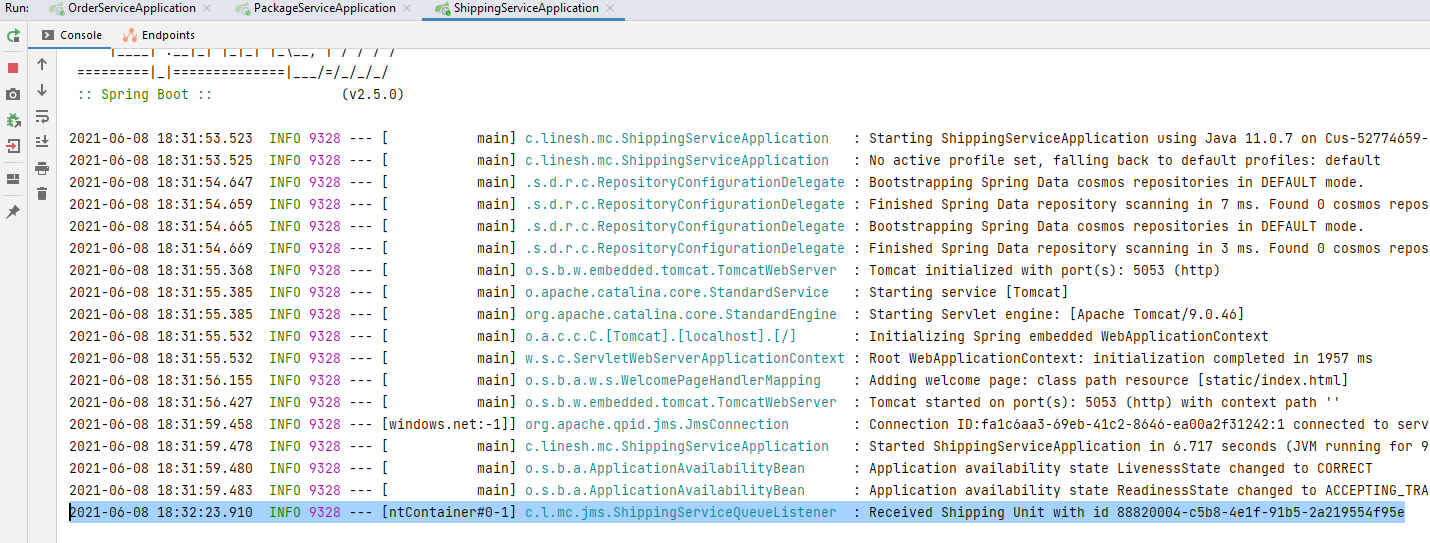
### Circuit Breaker and Retry Functionalities in Order Service



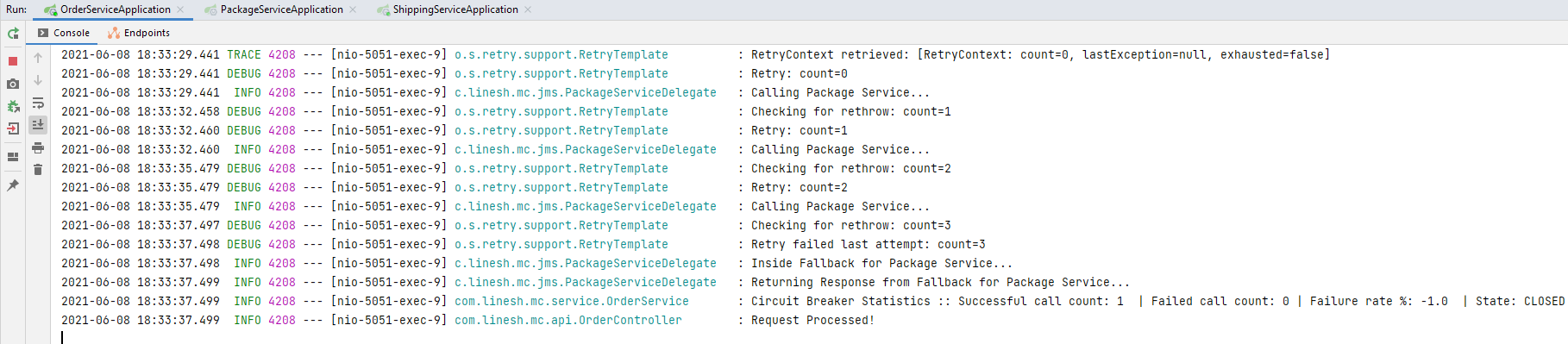
### Data Received in Downstream Systems – Package Service



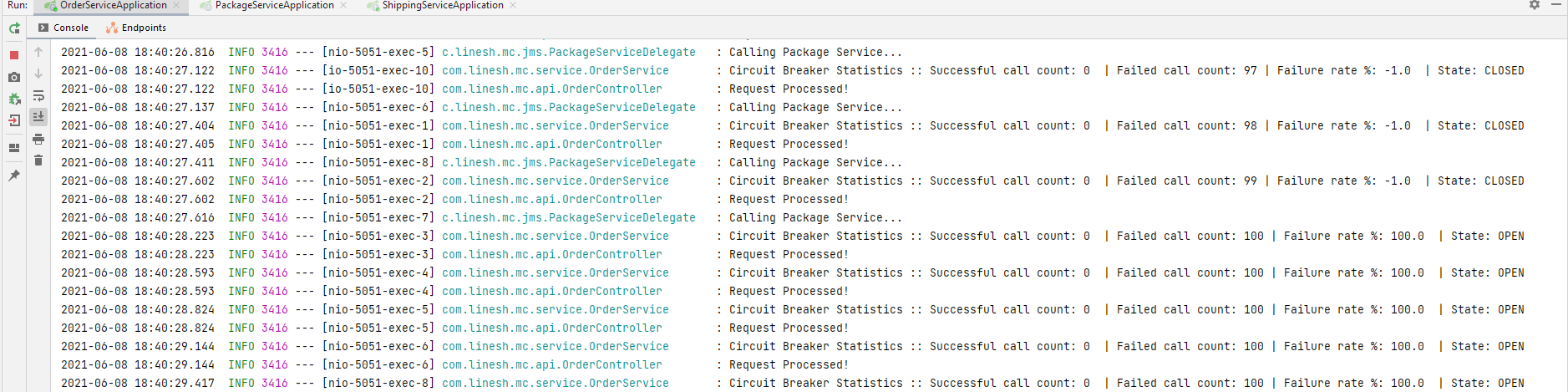
### Data Received in Downstream Systems – Courier Service



### Retry and Fallback

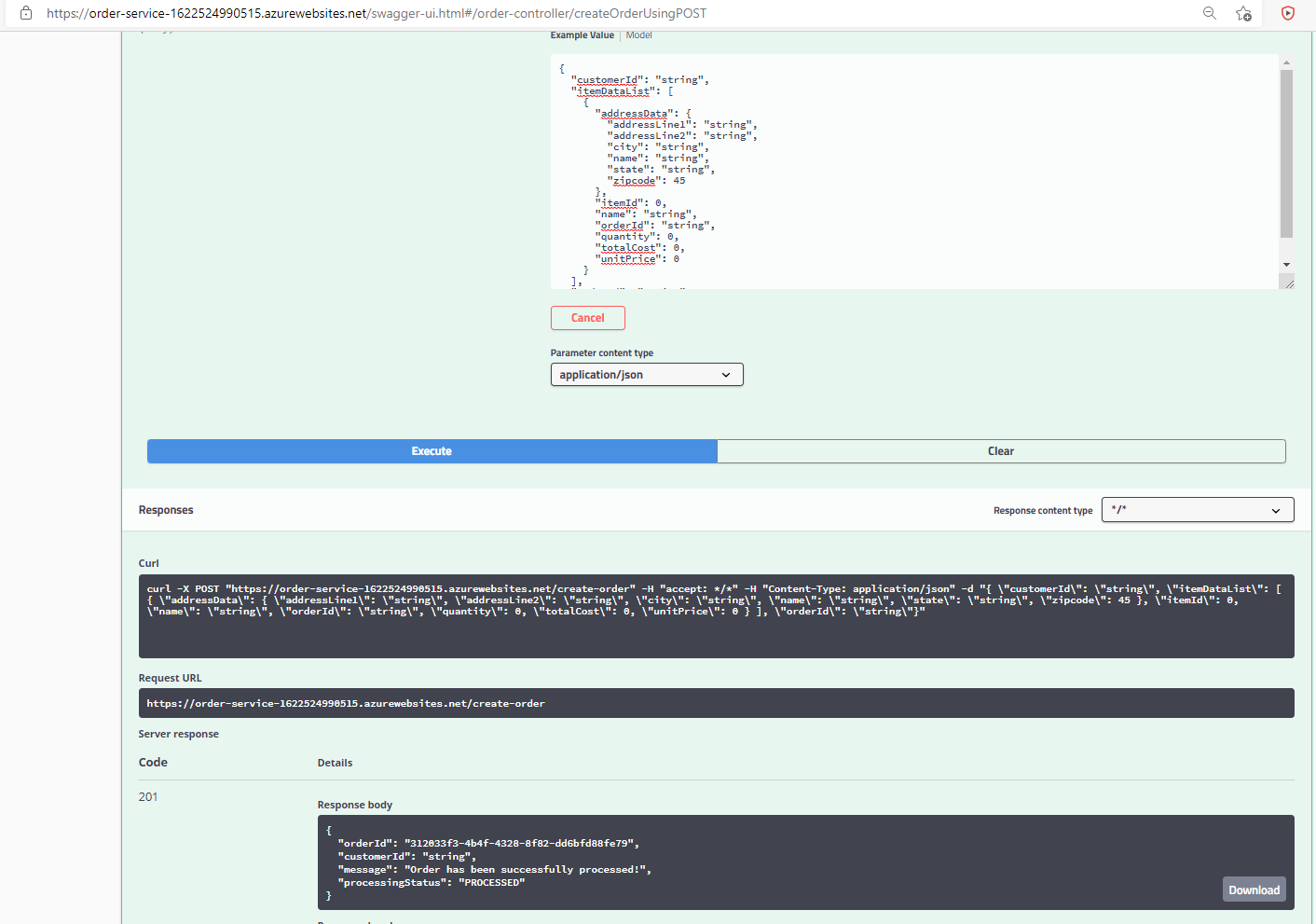


### Circuit Breaker

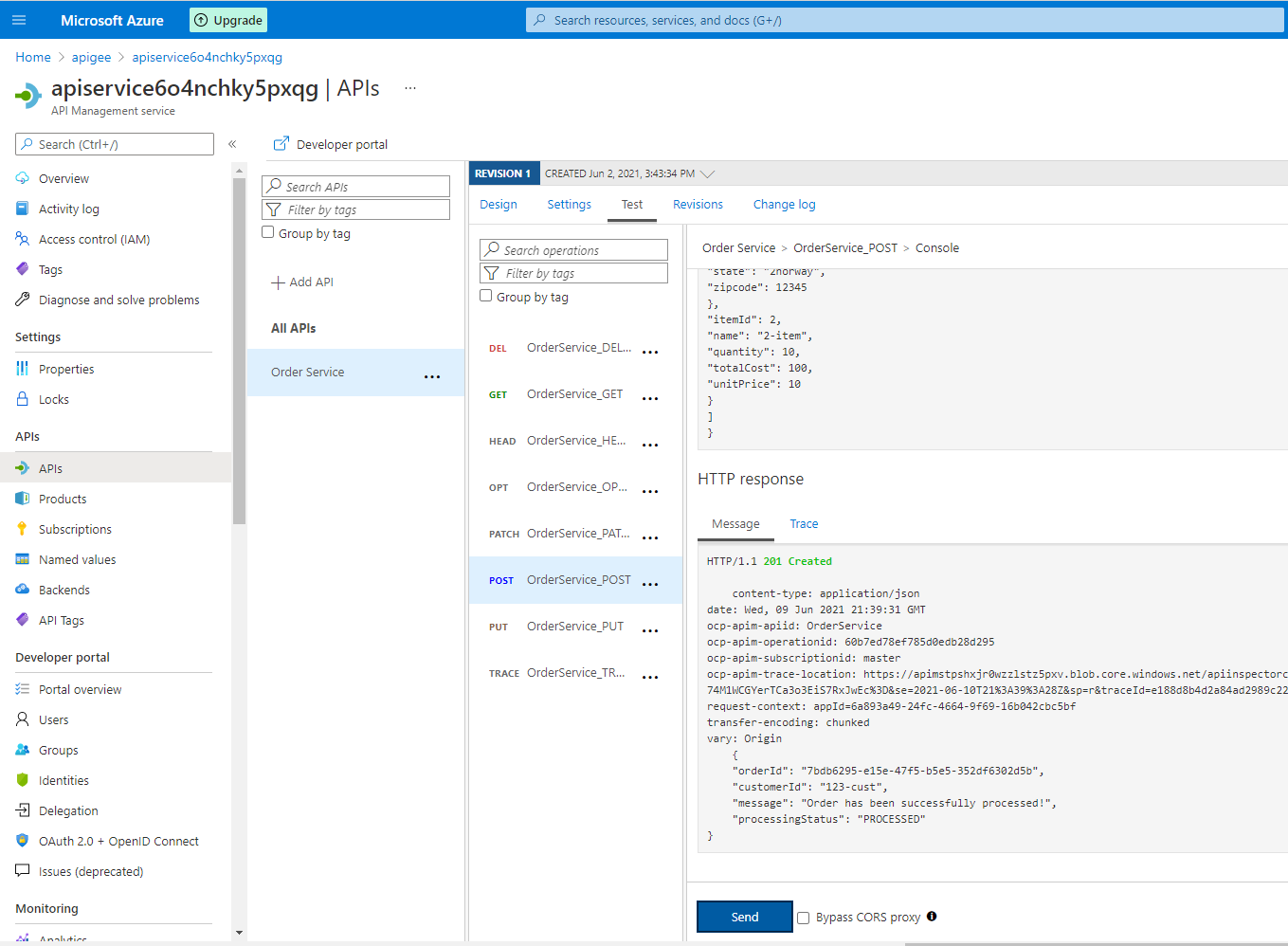


# Screenshots – Applications Running in Azure

### Order Service with Swagger

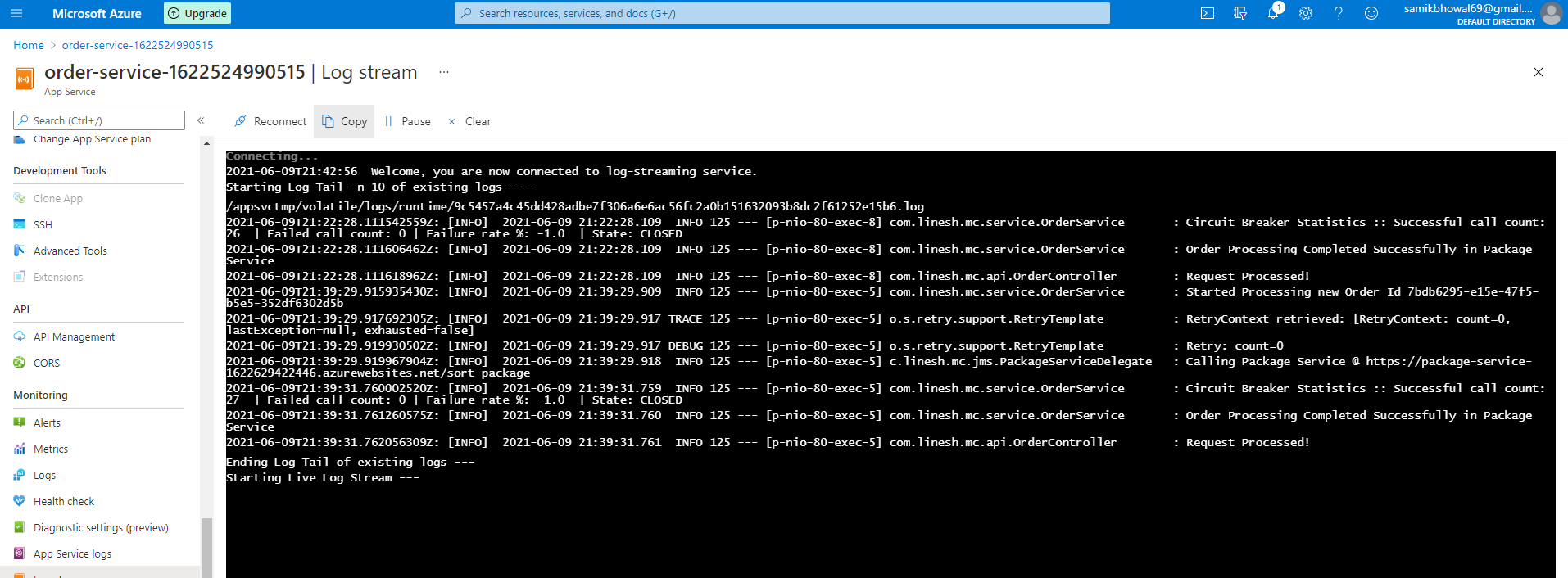


### Order Service with API Gateway

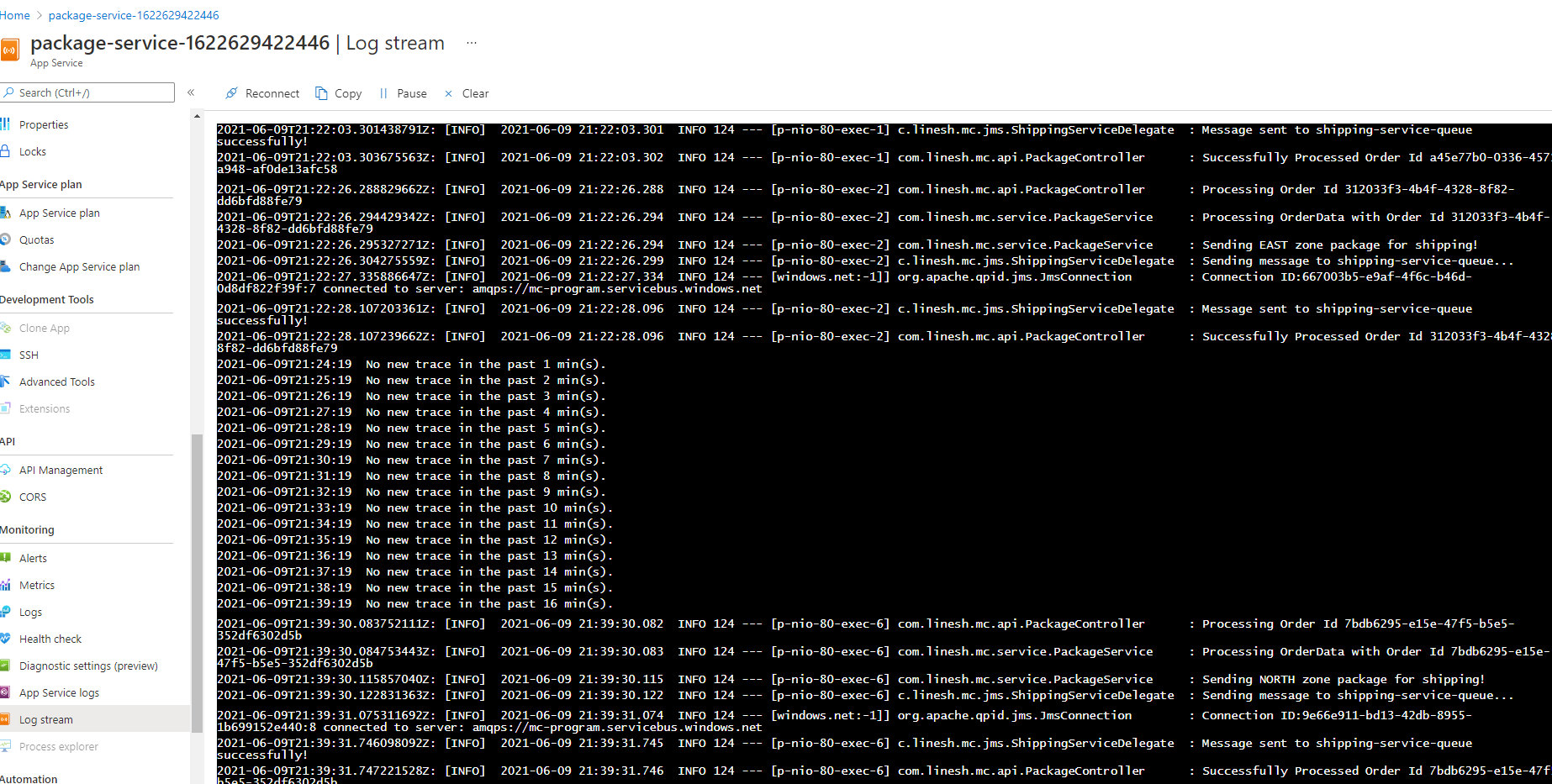


# Screenshots - Azure Application Logs

### Order Service



### Package Service



### Courier Service

