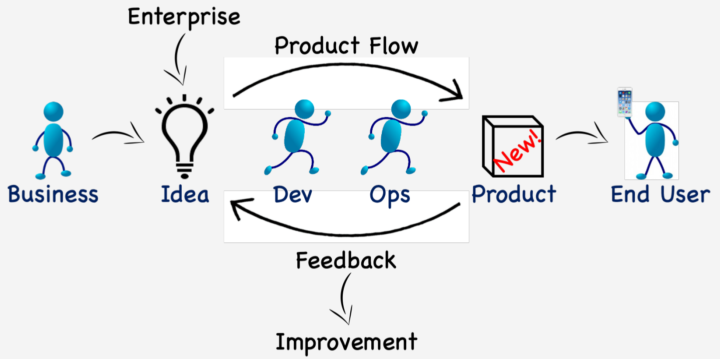
Management of cloud native messaging based applications

Lars Besselmann
EMEA Technical Sales Lead on Application Modernization

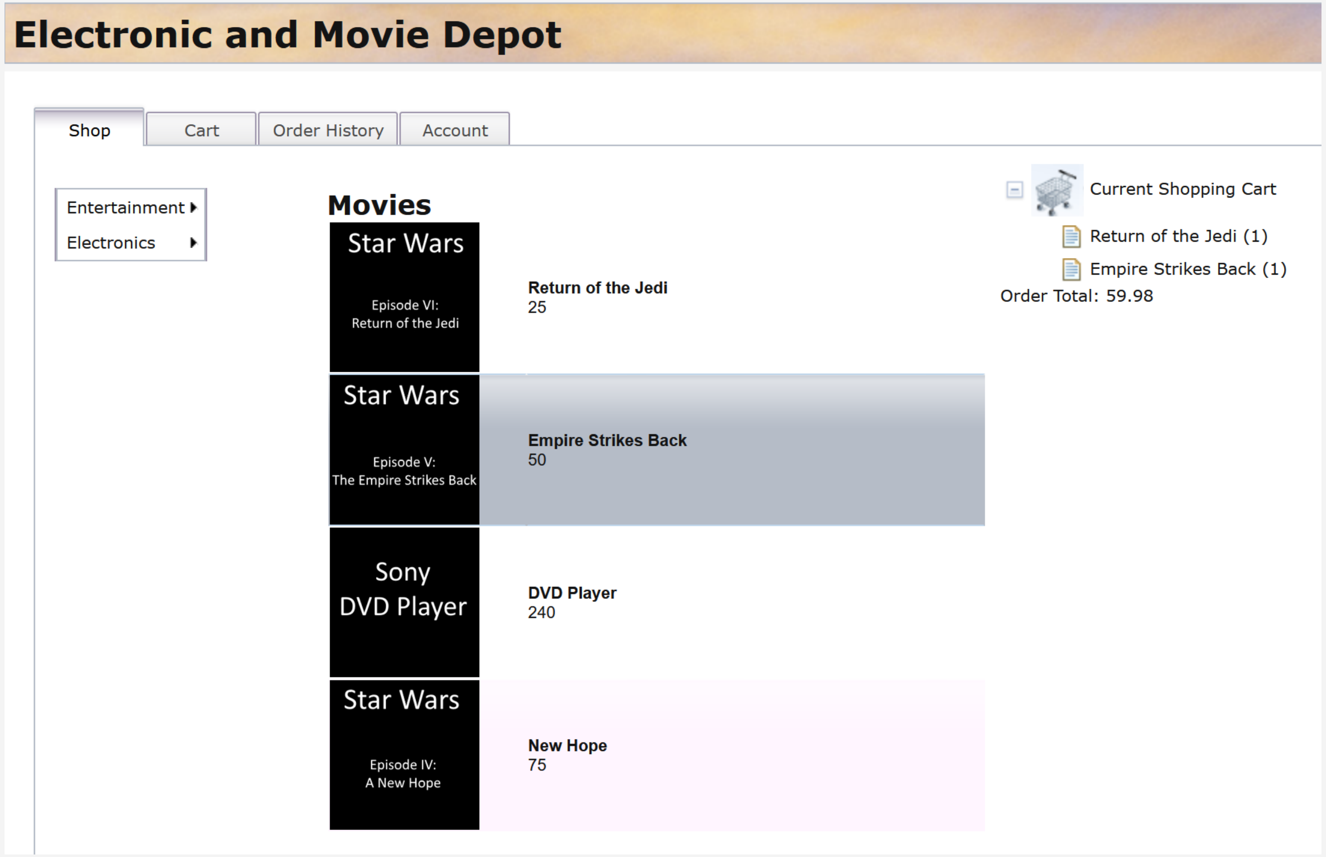
Modern(ized) applications come with new challenges



- ⬆ Infrastructure complexity
- ⬆ Number of technologies used
- ⬆ Dynamic changes in apps
- ⬆ Skills gap
- ⬇ Visibility

Application modernization, frequent releases, and use of cloud services creates a challenge for IT Operations

Demo application: Customer Order Services application



Electronic and Movie Depot

Shop | Cart | Order History | Account

Entertainment ▸
Electronics ▸

Movies

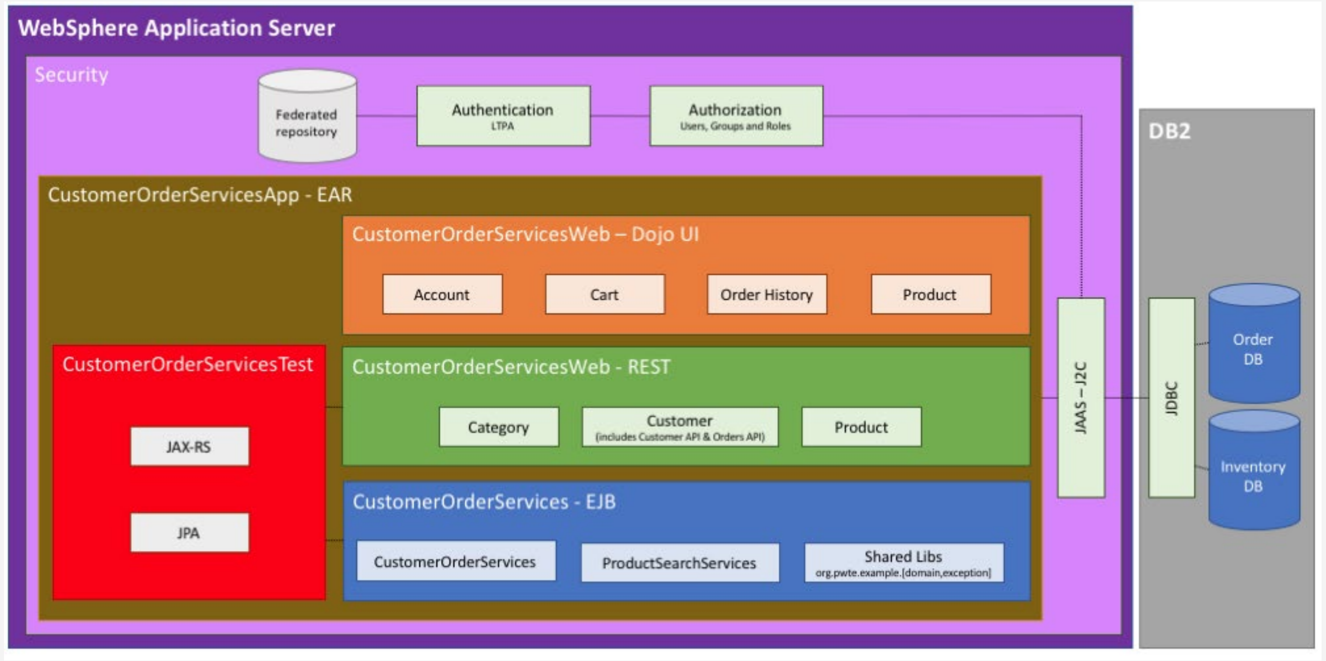
Star Wars Episode VI: Return of the Jedi	**Return of the Jedi** 25
Star Wars Episode V: The Empire Strikes Back	**Empire Strikes Back** 50
Sony DVD Player	**DVD Player** 240
Star Wars Episode IV: A New Hope	**New Hope** 75

Current Shopping Cart
- Return of the Jedi (1)
- Empire Strikes Back (1)

Order Total: 59.98

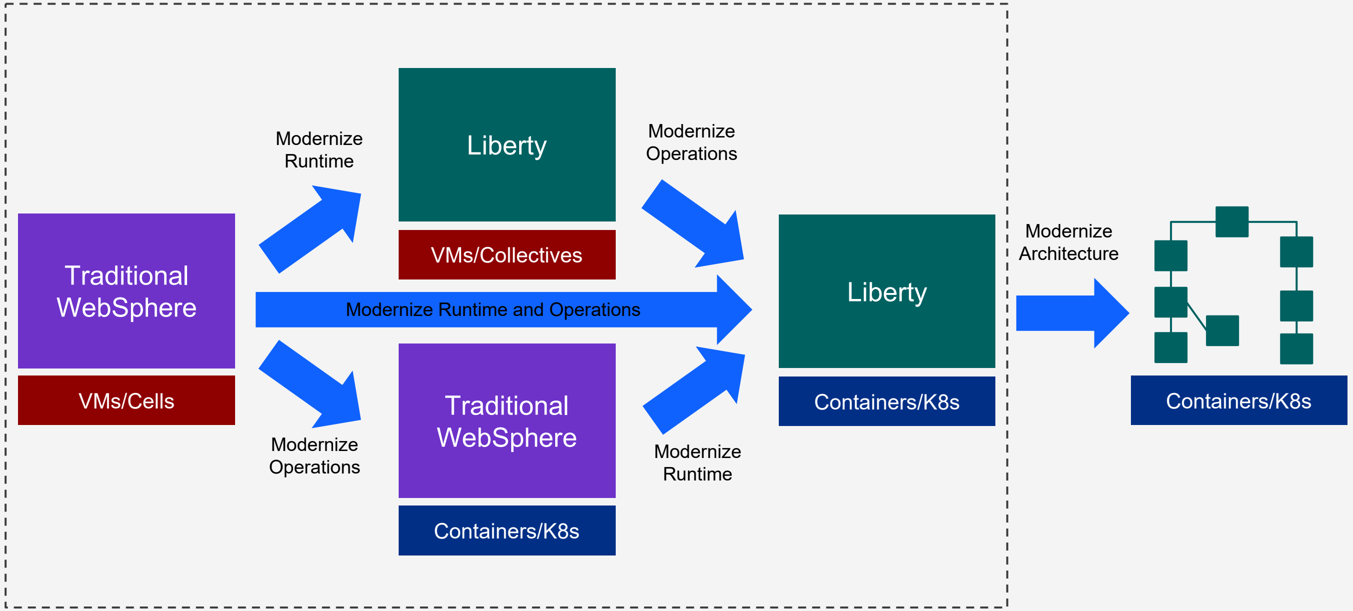
Demo application: Customer Order Services application

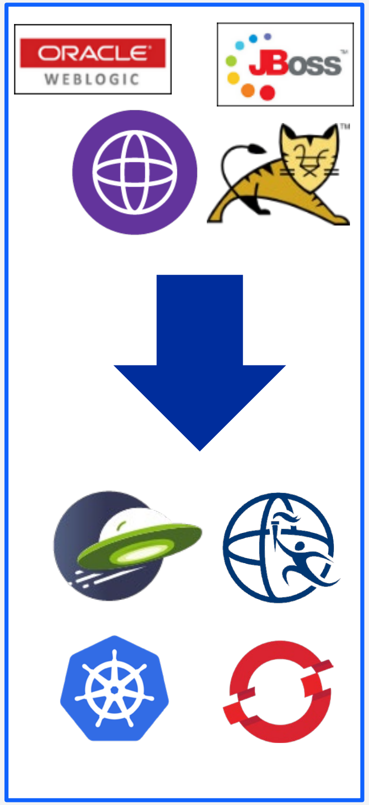
- The Customer Order Services application is a simple store-front shopping application, built during the early days of the Web 2.0 movement.
- Users interact directly with a browser-based interface and manage their cart to submit orders.
- This application is built using the traditional 3-Tier Architecture model, with an HTTP server, an application server, and a supporting database.


WebSphere Application Server
Security
Federated repository
Authentication
LTPA
Authorization
Users, Groups and Roles
CustomerOrderServicesApp - EAR
CustomerOrderServicesWeb – Dojo UI
Account
Cart
Order History
Product
CustomerOrderServicesTest
JAX-RS
JPA
CustomerOrderServicesWeb - REST
Category
Customer
(includes Customer API & Orders API)
Product
CustomerOrderServices - EJB
CustomerOrderServices
ProductSearchServices
Shared Libs
org.pwte.example.[domain,exception]
JAAS – J2C
JDBC
DB2
Order DB
Inventory DB

[Modernization Playbook – Modernizing application runtimes (ibm-cloud-architecture.github.io)](#)

Containerize the application on the **light-weight** cloud native runtime **Liberty** via **IBM Cloud Transformation Advisor**

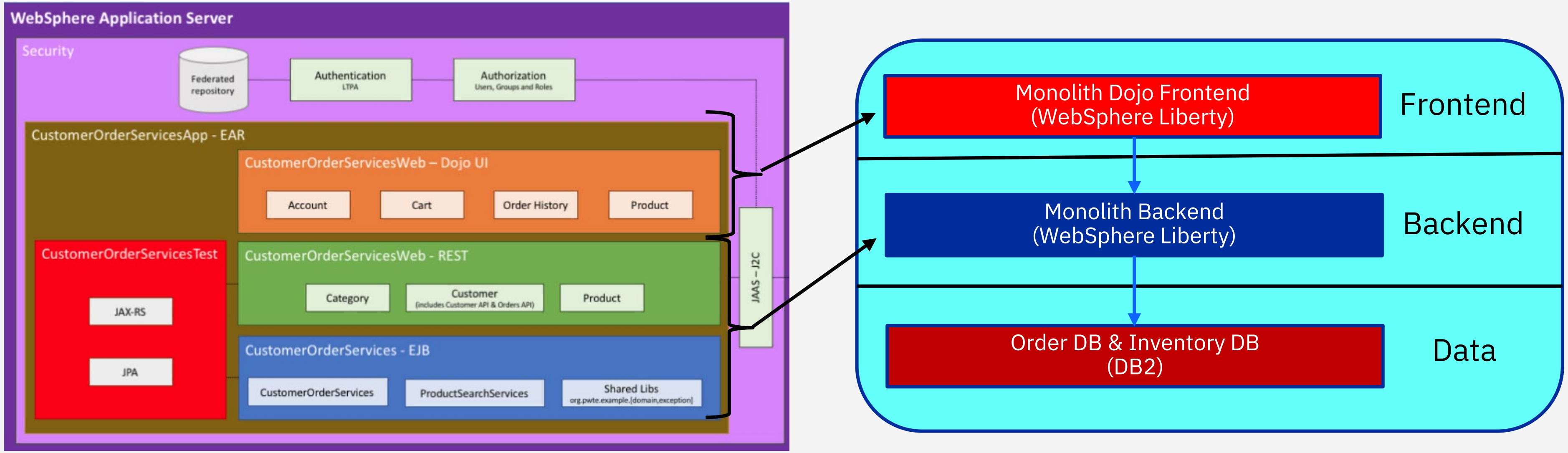




Optimum modernization depends on workload needs! http://ibm.biz/cloudta

Put the frontend into a separate container

Once the monolith is containerized, the application owners may choose to separate the frontend application code (DOJO based) from the monolith's EAR file into a separately deployable component. With the front and backend separated they can be developed, tested and deployed independently.



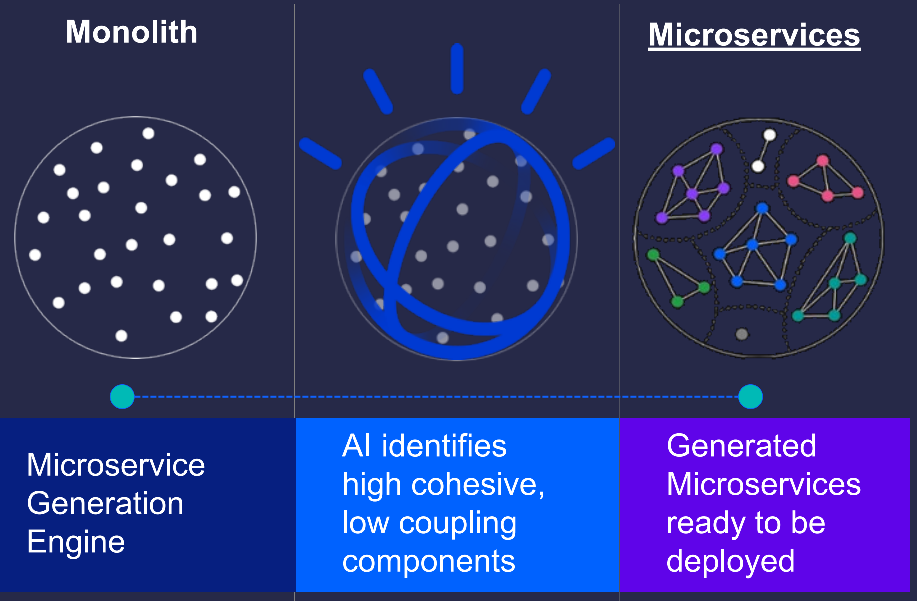
Refactor the backend part via IBM® Mono2Micro™

AI assisted transformation of monoliths into microservices, with reduced risk and lower cost.

Mono2Micro

- **Visualizes runtime and data dependencies** between classes
- **Identifies dead code** within your application
- Uses AI to **generate recommendations**, **semantic analysis** and **code** needed for refactoring
- Applies microservices best practices to recommend partitioning

Available now. Fully supported. ibm.biz/Mono2Micro


Monolith
Microservices
Microservice Generation Engine
AI identifies high cohesive, low coupling components
Generated Microservices ready to be deployed

User can interactively refine recommended Microservices

Generates code for communication between microservices

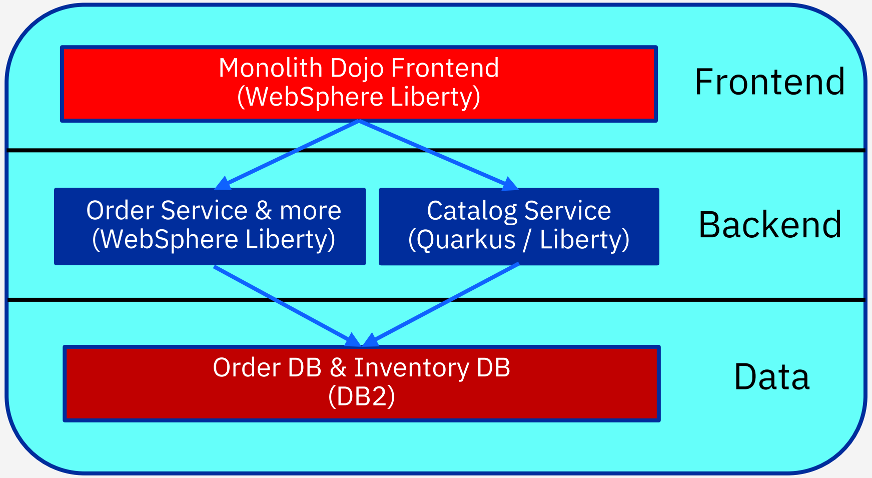
Refactor the database to reduce dependencies

Mono2Micro was used to identify the Catalog service as a good candidate to become it's own service.

- The Catalog service handles information about the items in the store such as prices, titles, ratings and descriptions.
- The remaining monolith contains the order service and more.
- To avoid dependencies between the two services, each service has its own database to keep its own data.

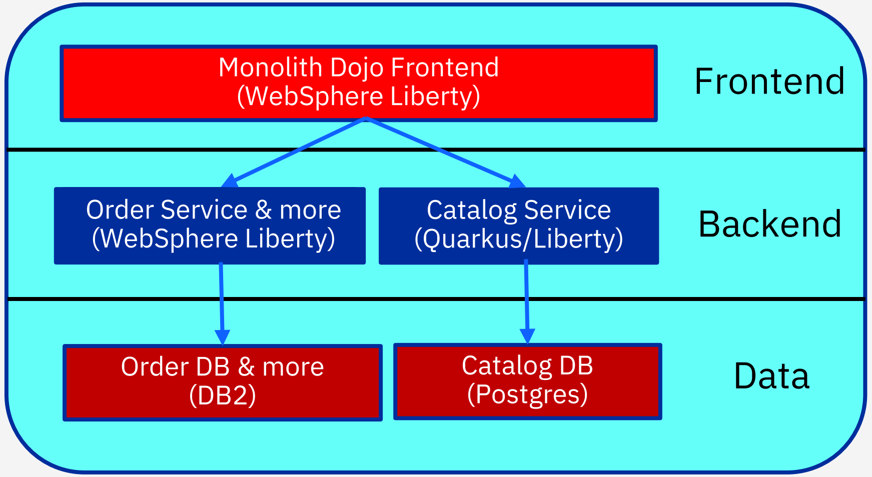
Refactor the backend

Split the backend into separate services



Refactor the datastore

Split the database into different databases



Introduce **Kafka messaging** to keep the databases in sync

The challenge: There is some data required to by both backend services and therefore available in both databases.
Solution: To keep common data such as the price in sync, **Kafka** will be used.

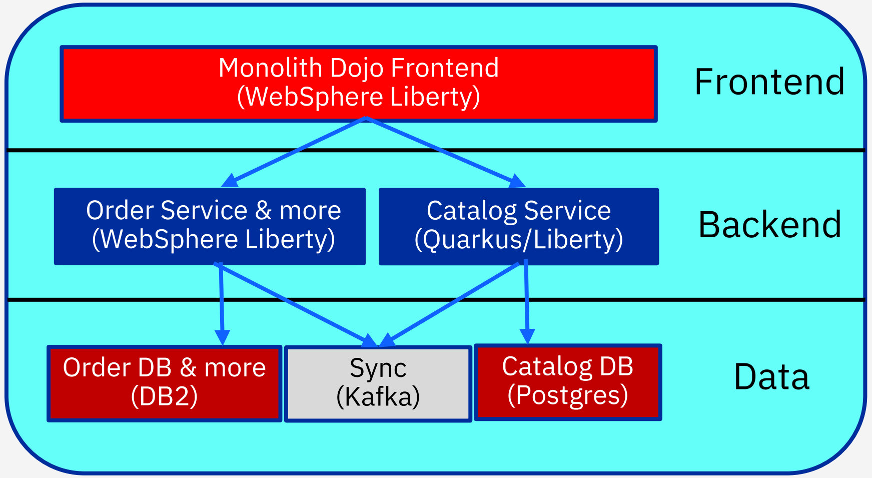
The application uses MicroProfile and Kafka to send events asynchronously.

The Event Producer
The Catalog Service stores the changed price into the Catalog DB and produces a Kafka event that the price has changed.

The Event Consumer
The Order Service consumes the Kafka event and caches the changed prices in its own Db2 database in a new column.

Implement a synchornization

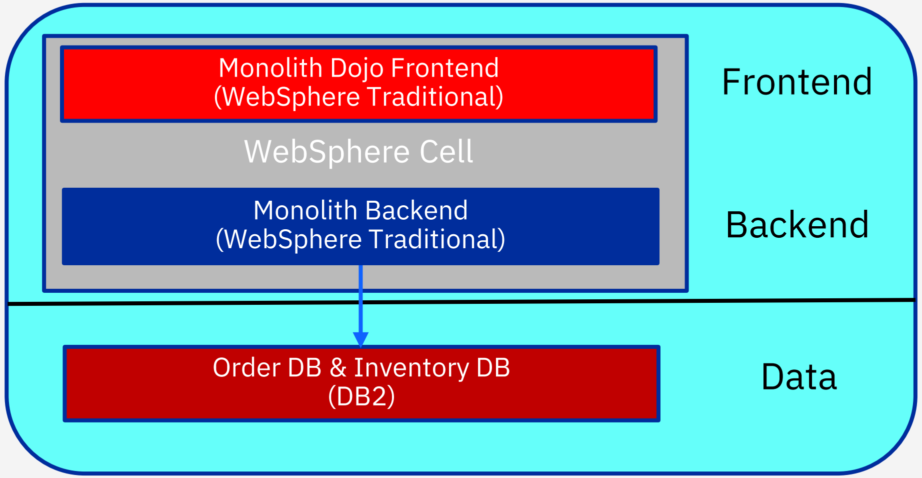
Synchonize the databases via Kafka


Monolith Dojo Frontend (WebSphere Liberty)
Frontend
Order Service & more (WebSphere Liberty)
Catalog Service (Quarkus/Liberty)
Backend
Order DB & more (DB2)
Sync (Kafka)
Catalog DB (Postgres)
Data

Finding: Modernization made the application more complex

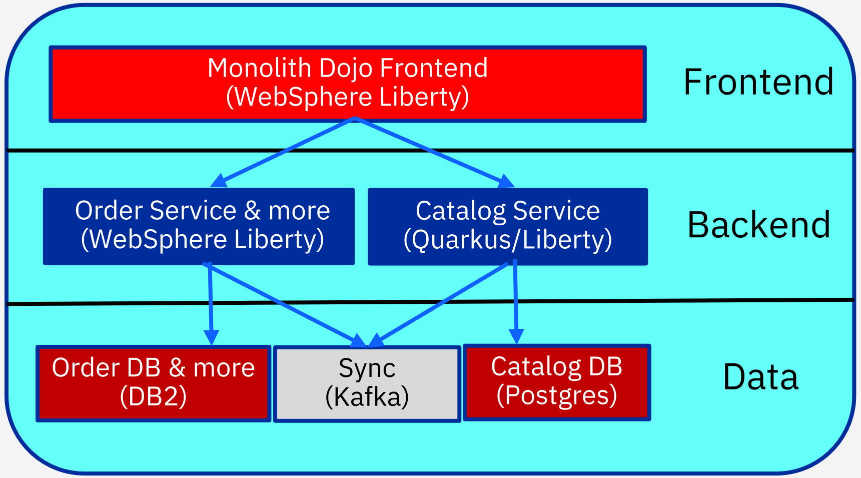
Before: traditional WAS application

(Technologies: traditional WAS, DB2)



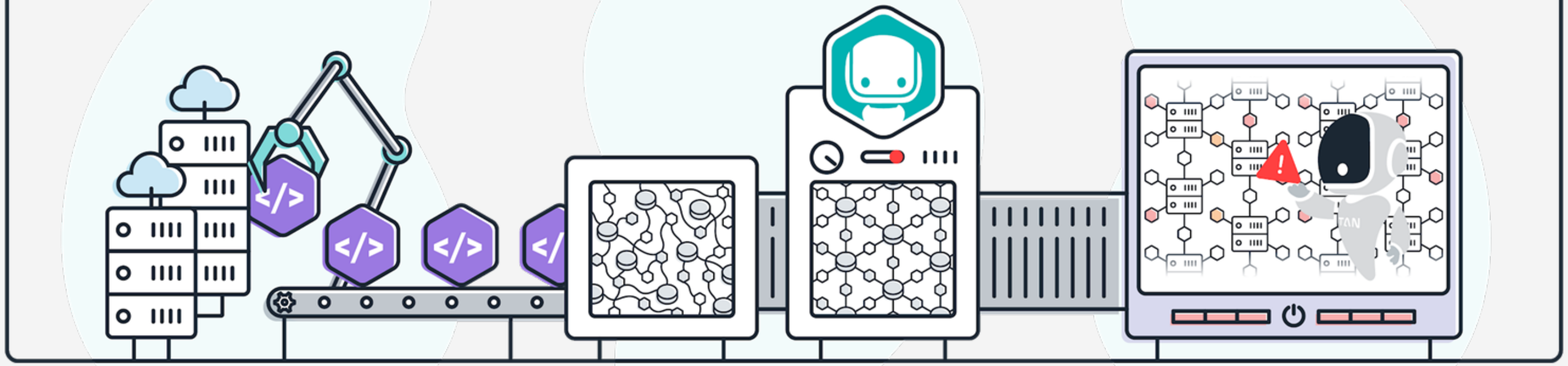
After: Refactored application

(Technologies: Liberty, Quarkus, DB2, Postgres, Kafka)



Instana's enterprise observability platform

Automation **Context** **Intelligent Action**



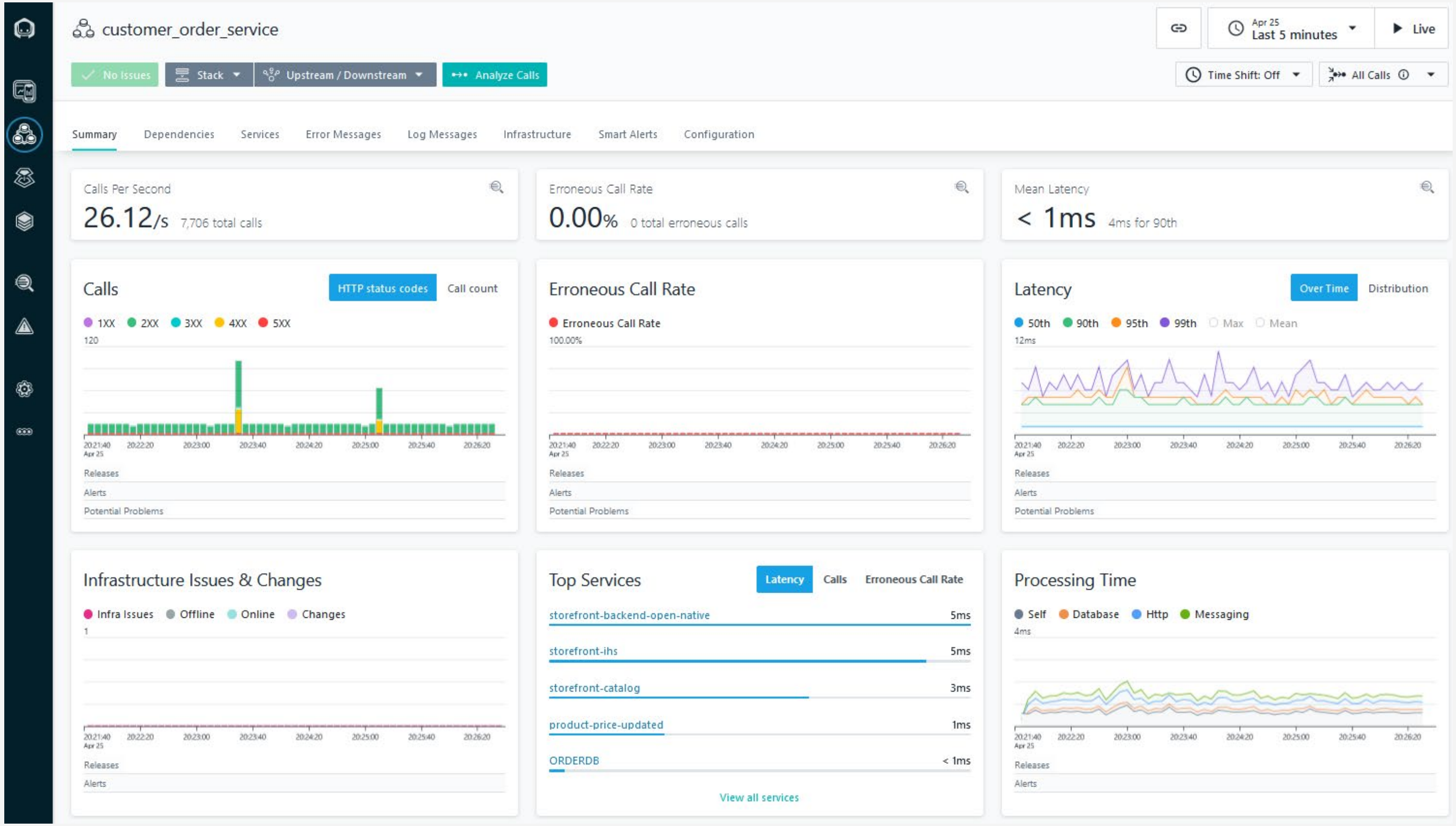
✓ **Mitigate Risk** ✓ **Protect Revenue** ✓ **Gain Efficiency**

AUTOMATION
Gain full observability in dynamic environments with auto discovery & configuration

CONTEXT
Understand all application inter-dependencies to diagnose issues and determine impact

INTELLIGENT ACTION
Proactively detect and remediate issues with an understanding of contributing factors

Instana application perspective



Calls Per Second
26.12/s 7,706 total calls

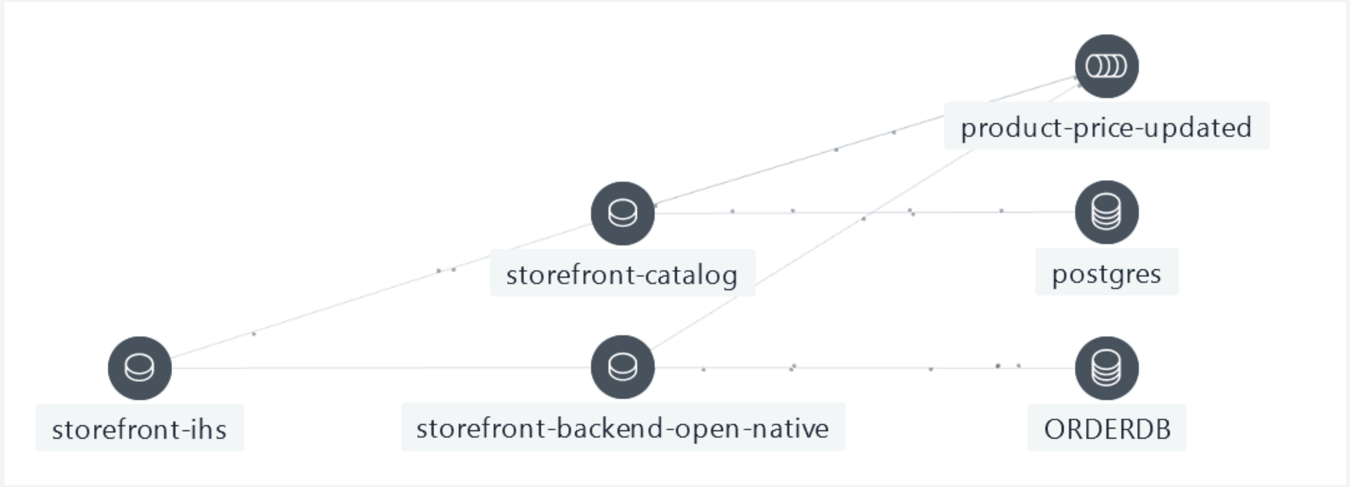
Erroneous Call Rate
0.00% 0 total erroneous calls

Mean Latency
< 1ms 4ms for 90th

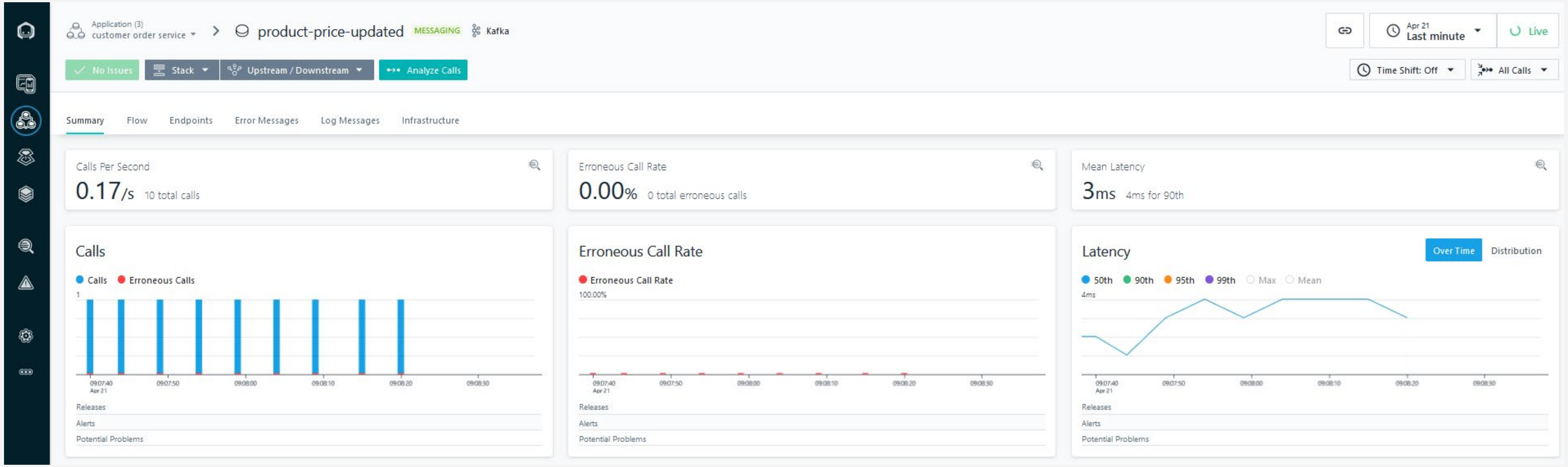
Top Services

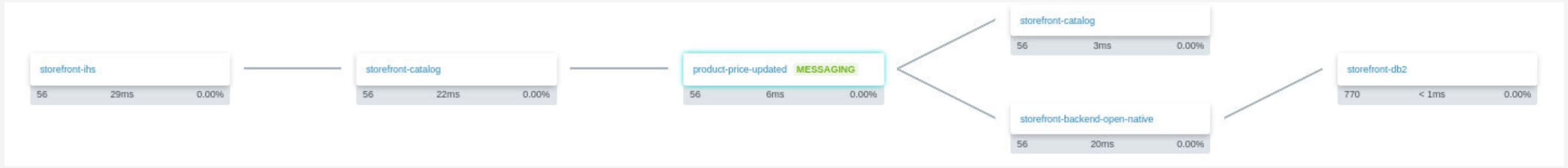
storefront-backend-open-native	5ms
storefront-ihs	5ms
storefront-catalog	3ms
product-price-updated	1ms
ORDERDB	< 1ms

Instana dependencies diagram

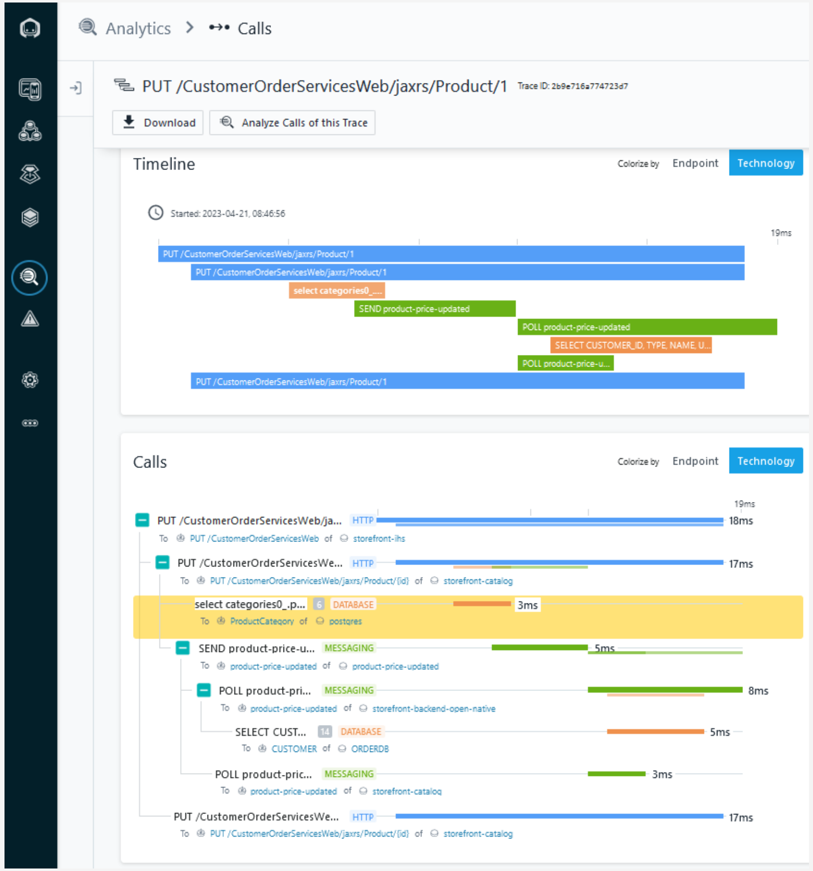


Instana metrics for Kafka





Instana tracing - Databases



Analytics > Calls

PUT /CustomerOrderServicesWeb/jaxrs/Product/1 Trace ID: 2b9e716a774723d7

Download | Analyze Calls of this Trace

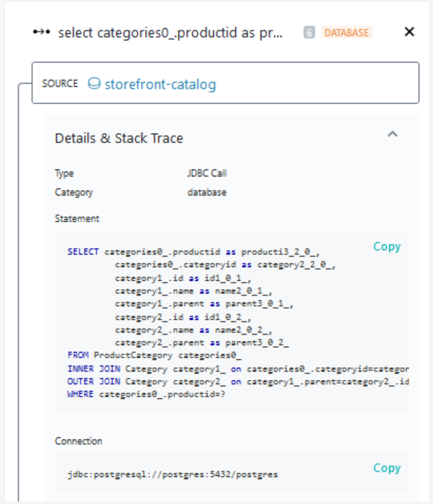
Timeline — Colorize by: Endpoint | Technology

Started: 2023-04-21, 08:46:56 — 19ms

- PUT /CustomerOrderServicesWeb/jaxrs/Product/1
- PUT /CustomerOrderServicesWeb/jaxrs/Product/1
- select categories0_...
- SEND product-price-updated
- POLL product-price-updated
- SELECT CUSTOMER_ID, TYPE, NAME, U...
- POLL product-price-u...
- PUT /CustomerOrderServicesWeb/jaxrs/Product/1

Calls — Colorize by: Endpoint | Technology

Call	Technology	Duration	To	of
PUT /CustomerOrderServicesWeb/ja...	HTTP	18ms	PUT /CustomerOrderServicesWeb	storefront-ihs
PUT /CustomerOrderServicesWe...	HTTP	17ms	PUT /CustomerOrderServicesWeb/jaxrs/Product/{id}	storefront-catalog
select categories0_.p... (6)	DATABASE	3ms	ProductCategory	postgres
SEND product-price-u...	MESSAGING	5ms	product-price-updated	product-price-updated
POLL product-pri...	MESSAGING	8ms	product-price-updated	storefront-backend-open-native
SELECT CUST... (14)	DATABASE	5ms	CUSTOMER	ORDERDB
POLL product-pric...	MESSAGING	3ms	product-price-updated	storefront-catalog
PUT /CustomerOrderServicesWe...	HTTP	17ms	PUT /CustomerOrderServicesWeb/jaxrs/Product/{id}	storefront-catalog



select categories0_.productid as pr... 6 DATABASE

SOURCE storefront-catalog

Details & Stack Trace

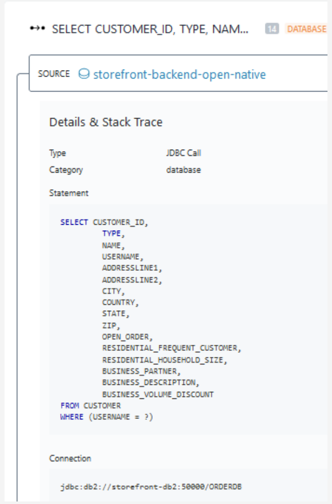
Type	JDBC Call
Category	database

Statement

```
SELECT categories0_.productid as producti3_2_0_,
         categories0_.categoryid as category2_2_0_,
         category1_.id as id1_0_1_,
         category1_.name as name2_0_1_,
         category1_.parent as parent3_0_1_,
         category2_.id as id1_0_2_,
         category2_.name as name2_0_2_,
         category2_.parent as parent3_0_2_
FROM ProductCategory categories0_
INNER JOIN Category category1_ on categories0_.categoryid=categor
OUTER JOIN Category category2_ on category1_.parent=category2_.id
WHERE categories0_.productid=?
```

Connection

```
jdbc:postgresql://postgres:5432/postgres
```



SELECT CUSTOMER_ID, TYPE, NAM... 14 DATABASE

SOURCE storefront-backend-open-native

Details & Stack Trace

Type	JDBC Call
Category	database

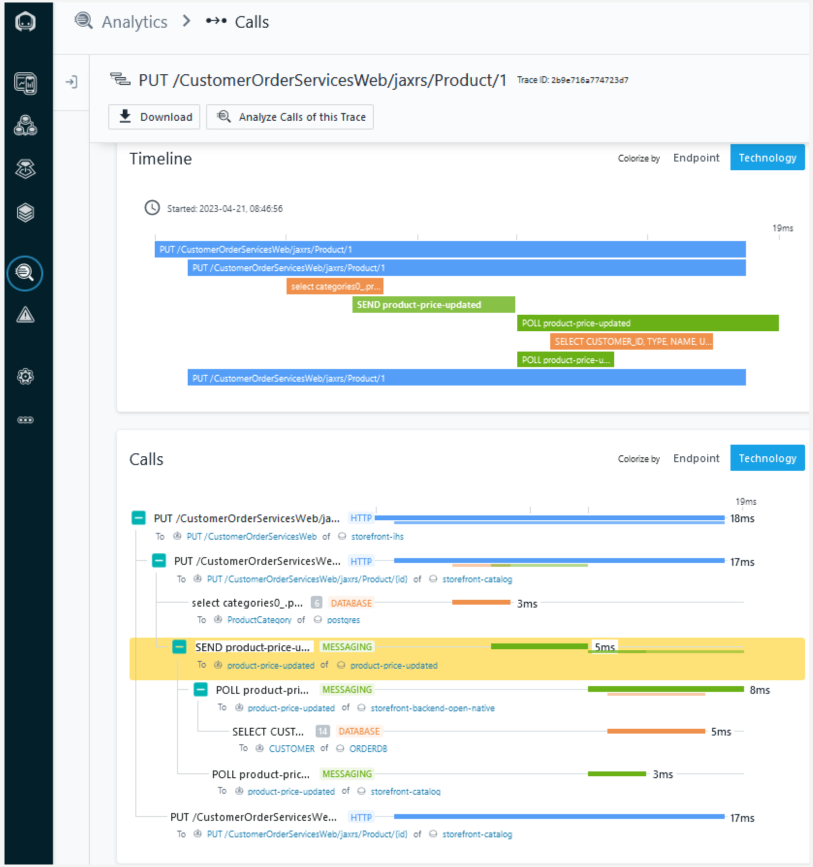
Statement

```
SELECT CUSTOMER_ID,
         TYPE,
         NAME,
         USERNAME,
         ADDRESSLINE1,
         ADDRESSLINE2,
         CITY,
         COUNTRY,
         STATE,
         ZIP,
         OPEN_ORDER,
         RESIDENTIAL_FREQUENT_CUSTOMER,
         RESIDENTIAL_HOUSEHOLD_SIZE,
         BUSINESS_PARTNER,
         BUSINESS_DESCRIPTION,
         BUSINESS_VOLUME_DISCOUNT
FROM CUSTOMER
WHERE (USERNAME = ?)
```

Connection

```
jdbc:db2://storefront-db2:50000/ORDERDB
```

Instana tracing - messaging



PUT /CustomerOrderServicesWeb/jaxrs/Product/1 Trace ID: 2b9e716a774723d7

Timeline

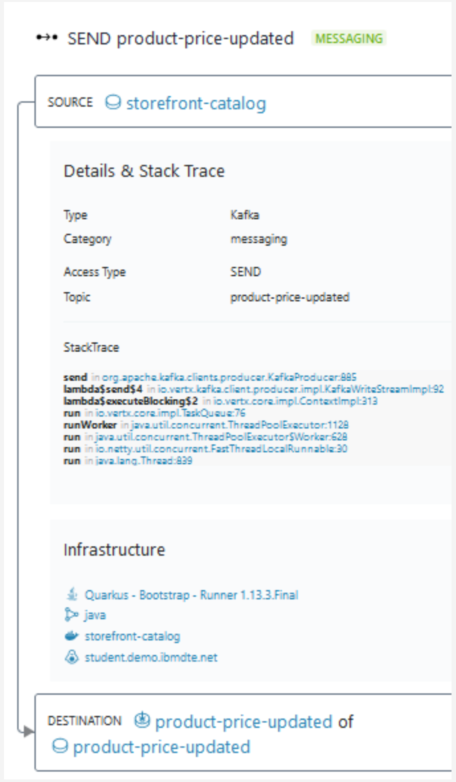
Started: 2023-04-21, 08:46:56

19ms

- PUT /CustomerOrderServicesWeb/jaxrs/Product/1
- PUT /CustomerOrderServicesWeb/jaxrs/Product/1
- select categories0_.pr...
- SEND product-price-updated
- POLL product-price-updated
- SELECT CUSTOMER_ID, TYPE, NAME, U...
- POLL product-price-u...
- PUT /CustomerOrderServicesWeb/jaxrs/Product/1

Calls

```
19ms
PUT /CustomerOrderServicesWeb/ja...   HTTP        18ms
  To PUT /CustomerOrderServicesWeb of storefront-ihs
  PUT /CustomerOrderServicesWe...     HTTP        17ms
    To PUT /CustomerOrderServicesWeb/jaxrs/Product/{id} of storefront-catalog
    select categories0_.p...  6  DATABASE         3ms
      To ProductCategory of postgres
    SEND product-price-u...     MESSAGING         5ms
      To product-price-updated of product-price-updated
      POLL product-pri...       MESSAGING         8ms
        To product-price-updated of storefront-backend-open-native
        SELECT CUST...  14  DATABASE              5ms
          To CUSTOMER of ORDERDB
      POLL product-pric...      MESSAGING         3ms
        To product-price-updated of storefront-catalog
  PUT /CustomerOrderServicesWe...     HTTP        17ms
    To PUT /CustomerOrderServicesWeb/jaxrs/Product/{id} of storefront-catalog
```



SEND product-price-updated MESSAGING

SOURCE storefront-catalog

Details & Stack Trace

Type	Kafka
Category	messaging
Access Type	SEND
Topic	product-price-updated

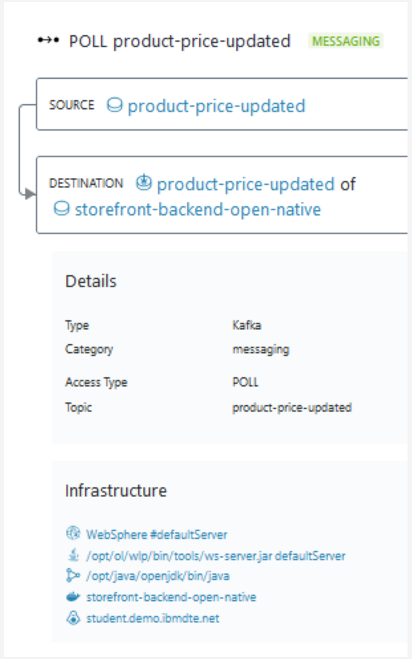
StackTrace

```
send in org.apache.kafka.clients.producer.KafkaProducer:885
lambda$send$4 in io.vertx.kafka.client.producer.impl.KafkaWriteStreamImpl:92
lambda$executeBlocking$2 in io.vertx.core.impl.ContextImpl:313
run in io.vertx.core.impl.TaskQueue:76
runWorker in java.util.concurrent.ThreadPoolExecutor:1128
run in java.util.concurrent.ThreadPoolExecutor$Worker:628
run in io.netty.util.concurrent.FastThreadLocalRunnable:30
run in java.lang.Thread:839
```

Infrastructure

- Quarkus - Bootstrap - Runner 1.13.3.Final
- java
- storefront-catalog
- student.demo.ibmdte.net

DESTINATION product-price-updated of product-price-updated



POLL product-price-updated MESSAGING

SOURCE product-price-updated

DESTINATION product-price-updated of storefront-backend-open-native

Details

Type	Kafka
Category	messaging
Access Type	POLL
Topic	product-price-updated

Infrastructure

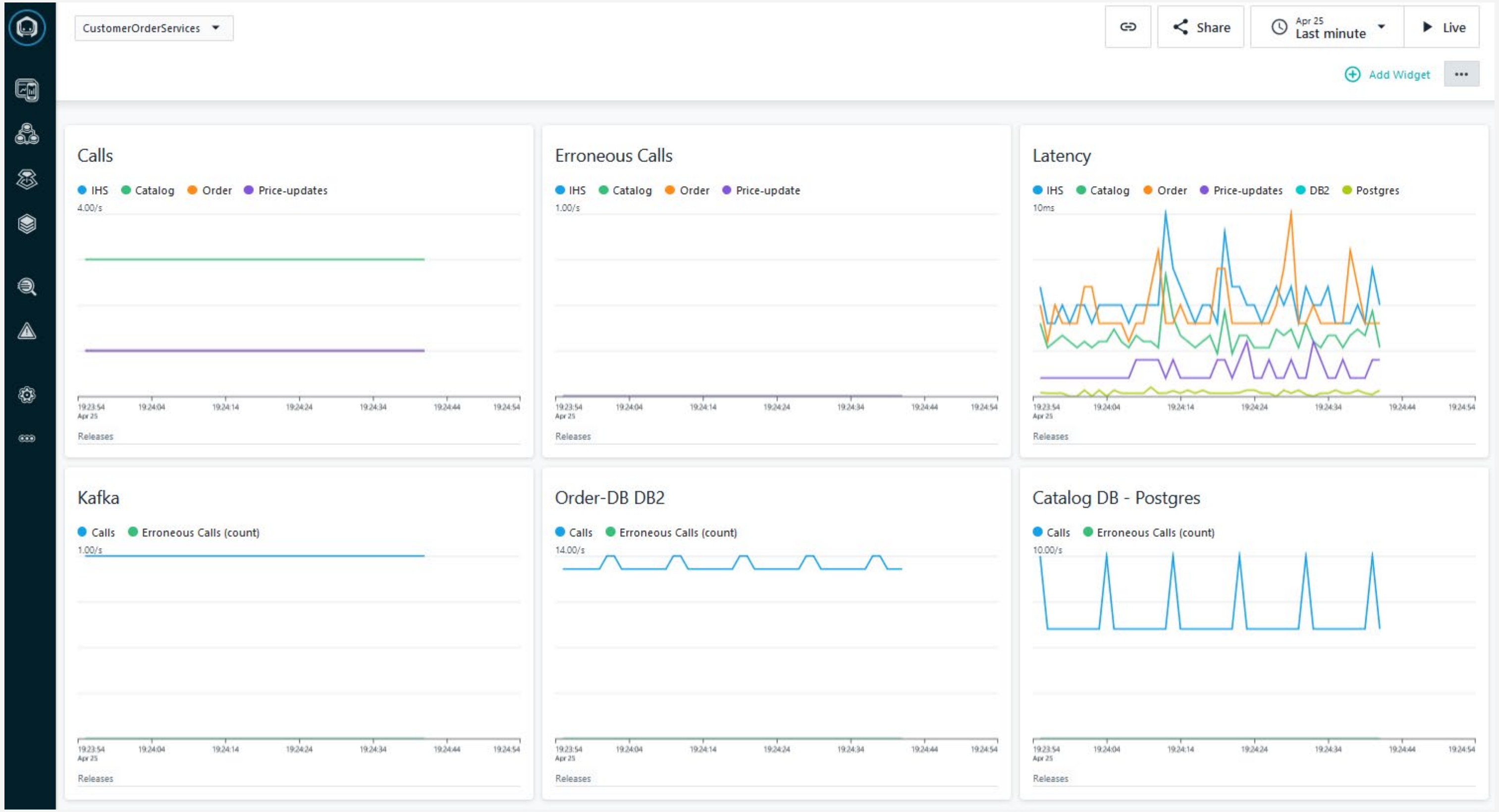
- WebSphere #defaultServer
- /opt/ol/wlp/bin/tools/ws-server.jar defaultServer
- /opt/java/openjdk/bin/java
- storefront-backend-open-native
- student.demo.ibmdte.net

Troubleshooting

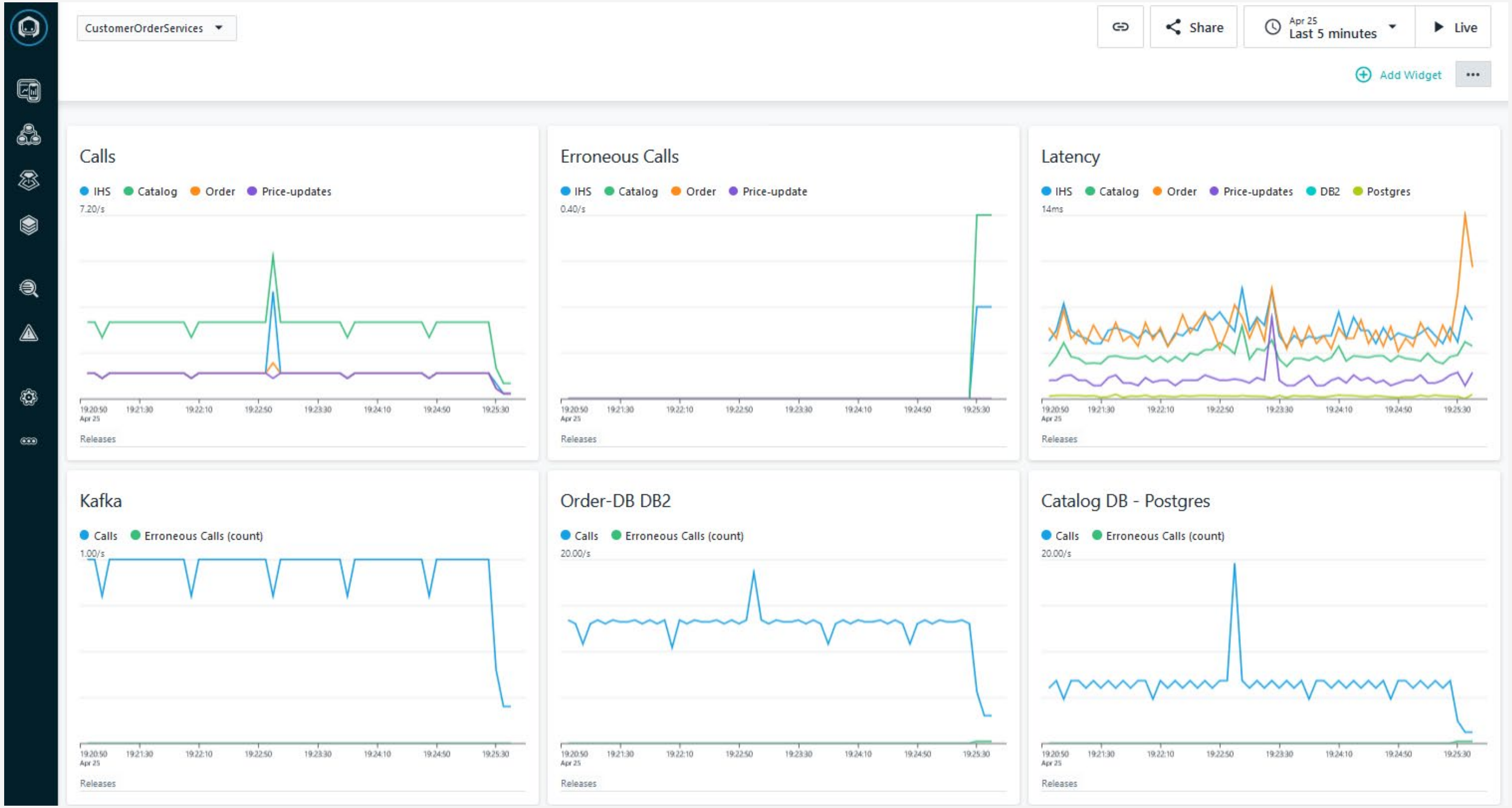
We are covering two scenarios:

- Scenario 1:
  Invalid deployment results into database errors
- Scenario 2:
  Invalid deployment results into HTTP errors and messaging problems

Initial state in dashboard

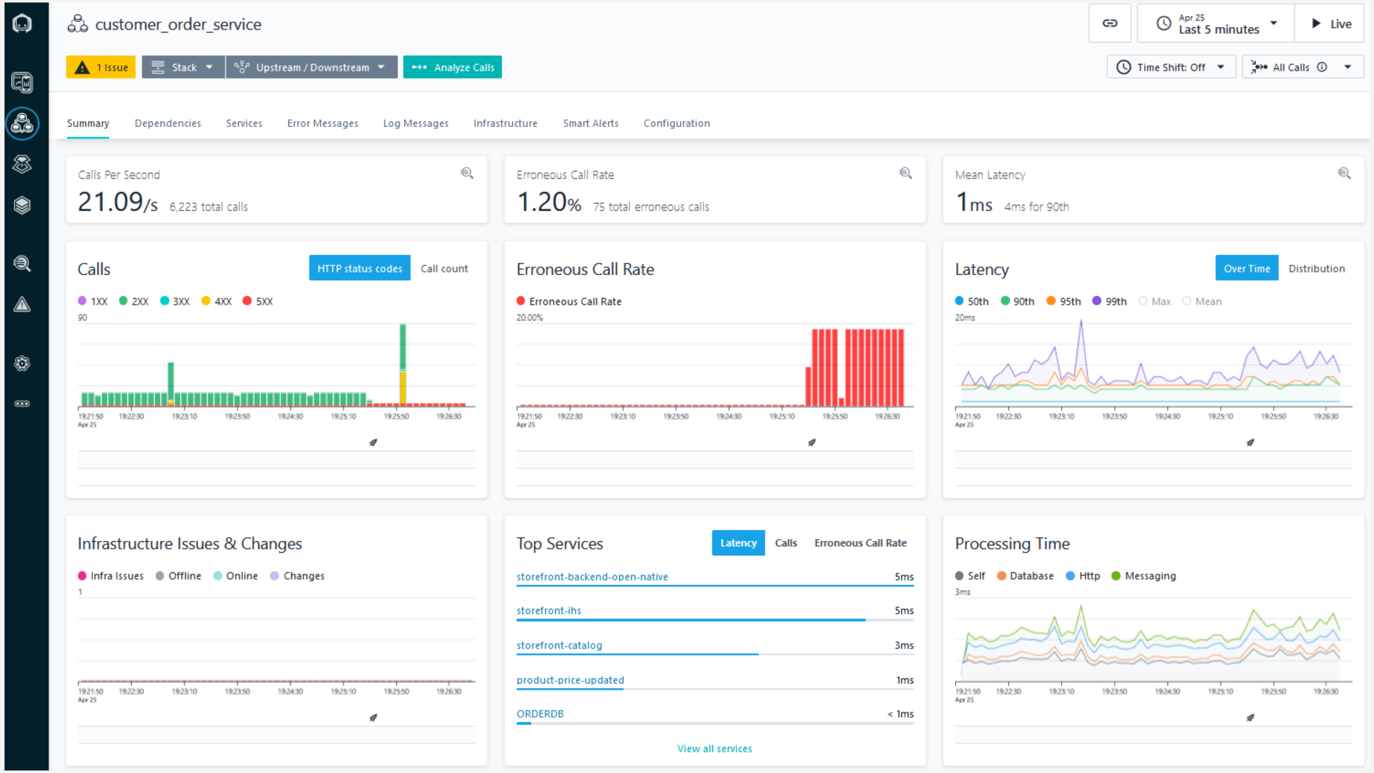


Troubleshooting Scenario 1 – Dashboard indicates problem



Troubleshooting Scenario 1 – Application Perspective

Problem likely caused by new deployment



customer_order_service

1 Issue · Stack · Upstream / Downstream · Analyze Calls

Apr 25 Last 5 minutes · Live · Time Shift: Off · All Calls

Summary · Dependencies · Services · Error Messages · Log Messages · Infrastructure · Smart Alerts · Configuration

Calls Per Second: 21.09/s — 6,223 total calls

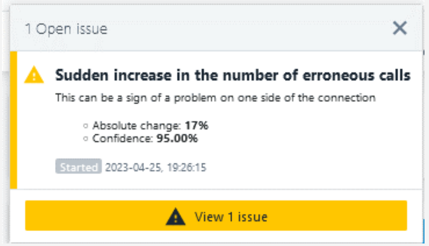
Erroneous Call Rate: 1.20% — 75 total erroneous calls

Mean Latency: 1ms — 4ms for 90th

Top Services (Latency · Calls · Erroneous Call Rate)

storefront-backend-open-native	5ms
storefront-ihs	5ms
storefront-catalog	3ms
product-price-updated	1ms
ORDERDB	< 1ms

View all services



1 Open issue

Sudden increase in the number of erroneous calls

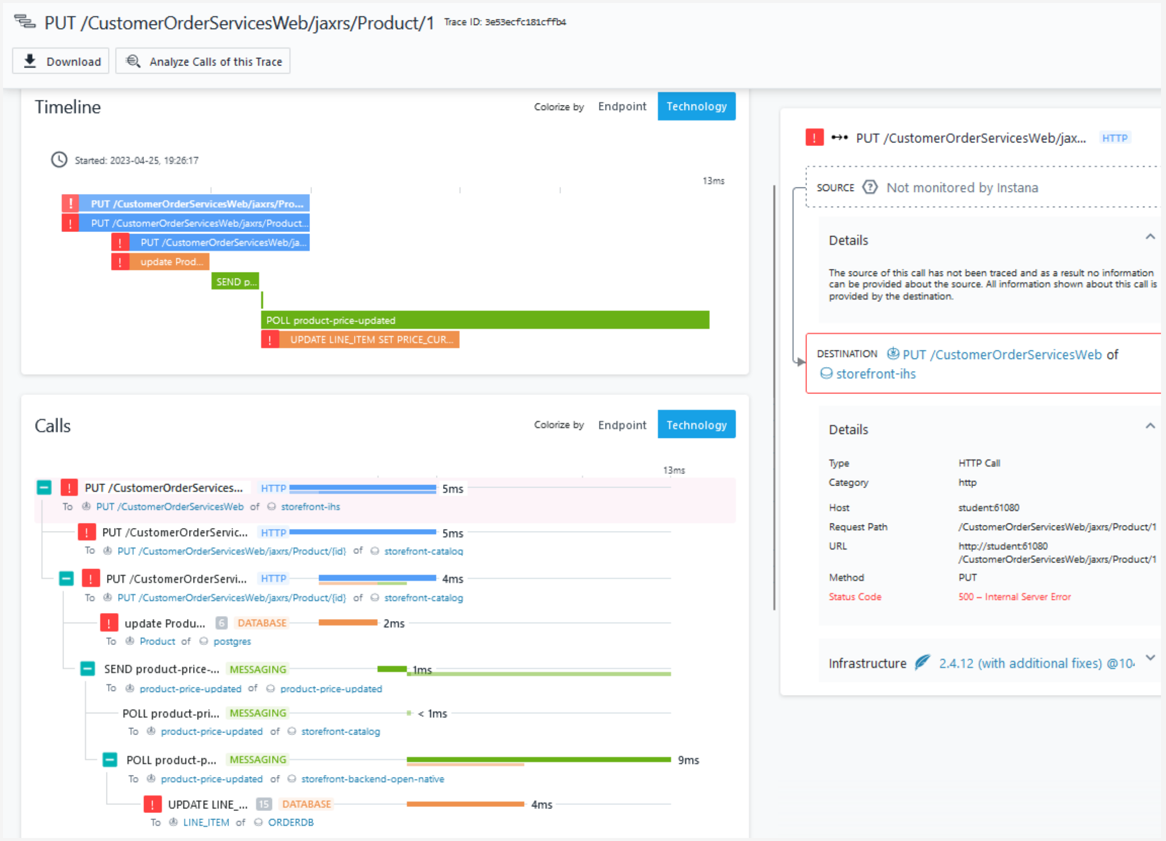
This can be a sign of a problem on one side of the connection

- Absolute change: **17%**
- Confidence: **95.00%**

Started 2023-04-25, 19:26:15

View 1 issue

Troubleshooting Scenario 1 – Track request



PUT /CustomerOrderServicesWeb/jaxrs/Product/1 Trace ID: 3e53ecfc181cffb4

Download
Analyze Calls of this Trace

Timeline

Colorize by Endpoint Technology

Started: 2023-04-25, 19:26:17

13ms
PUT /CustomerOrderServicesWeb/jaxrs/Pro...
PUT /CustomerOrderServicesWeb/jaxrs/Product...
PUT /CustomerOrderServicesWeb/ja...
update Prod...
SEND p...
POLL product-price-updated
UPDATE LINE_ITEM SET PRICE_CUR...

Calls

Colorize by Endpoint Technology
13ms

```
PUT /CustomerOrderServices...  HTTP  5ms
  To PUT /CustomerOrderServicesWeb of storefront-ihs
PUT /CustomerOrderServic...  HTTP  5ms
  To PUT /CustomerOrderServicesWeb/jaxrs/Product/{id} of storefront-catalog
PUT /CustomerOrderServi...  HTTP  4ms
  To PUT /CustomerOrderServicesWeb/jaxrs/Product/{id} of storefront-catalog
update Produ...  6  DATABASE  2ms
  To Product of postgres
SEND product-price-...  MESSAGING  1ms
  To product-price-updated of product-price-updated
POLL product-pri...  MESSAGING  < 1ms
  To product-price-updated of storefront-catalog
POLL product-p...  MESSAGING  9ms
  To product-price-updated of storefront-backend-open-native
UPDATE LINE_...  15  DATABASE  4ms
  To LINE_ITEM of ORDERDB
```

PUT /CustomerOrderServicesWeb/jax... HTTP

SOURCE Not monitored by Instana

Details

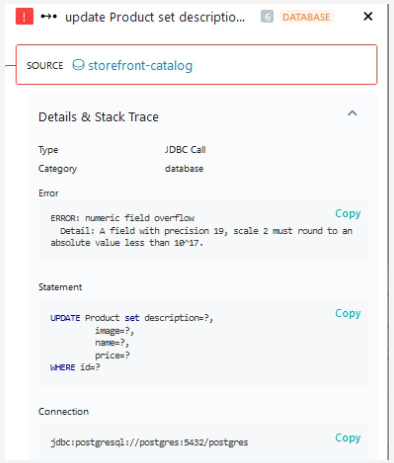
The source of this call has not been traced and as a result no information can be provided about the source. All information shown about this call is provided by the destination.

DESTINATION PUT /CustomerOrderServicesWeb of storefront-ihs

Details

Type	HTTP Call
Category	http
Host	student:61080
Request Path	/CustomerOrderServicesWeb/jaxrs/Product/1
URL	http://student:61080 /CustomerOrderServicesWeb/jaxrs/Product/1
Method	PUT
Status Code	500 – Internal Server Error

Infrastructure 2.4.12 (with additional fixes) @10.



update Product set descriptio... 6 DATABASE

SOURCE storefront-catalog

Details & Stack Trace

Type	JDBC Call
Category	database

Error

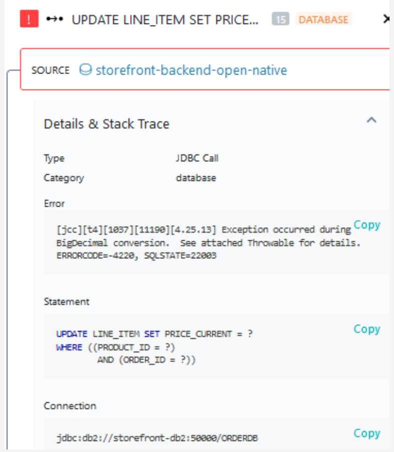
```
ERROR: numeric field overflow
  Detail: A field with precision 19, scale 2 must round to an
absolute value less than 10^17.
```

Statement

```
UPDATE Product set description=?,
         image=?,
         name=?,
         price=?
WHERE id=?
```

Connection

```
jdbc:postgresql://postgres:5432/postgres
```



UPDATE LINE_ITEM SET PRICE... 15 DATABASE

SOURCE storefront-backend-open-native

Details & Stack Trace

Type	JDBC Call
Category	database

Error

```
[jcc][t4][1037][11190][4.25.13] Exception occurred during
BigDecimal conversion.  See attached Throwable for details.
ERRORCODE=-4220, SQLSTATE=22003
```

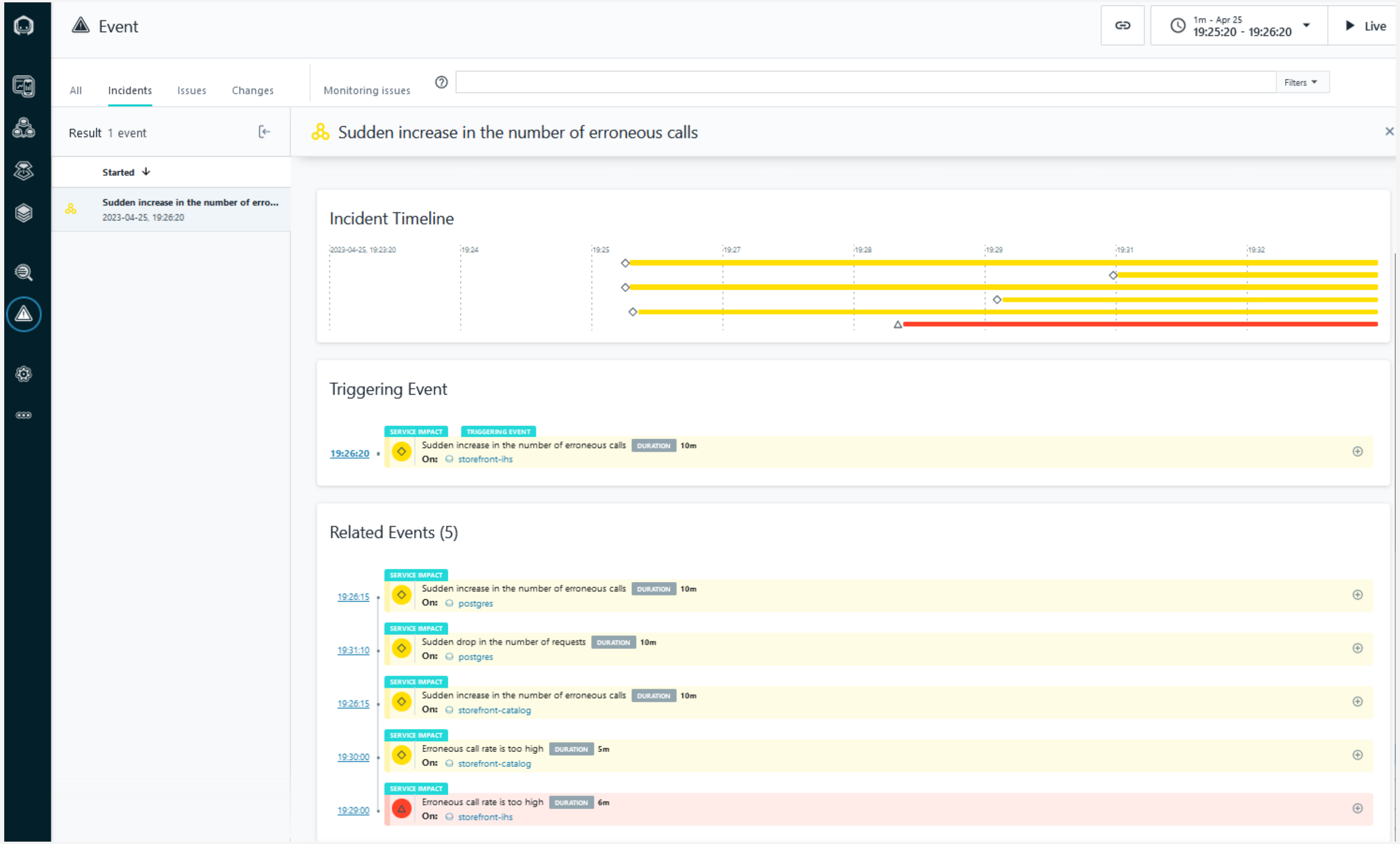
Statement

```
UPDATE LINE_ITEM SET PRICE_CURRENT = ?
WHERE ((PRODUCT_ID = ?)
        AND (ORDER_ID = ?))
```

Connection

```
jdbc:db2://storefront-db2:50000/ORDERDB
```

Troubleshooting Scenario 1 – Review events and alerts



Sudden increase in the number of erro...
2023-04-25, 19:26:20

Sudden increase in the number of erroneous calls

Incident Timeline

Triggering Event

19:26:20 · SERVICE IMPACT · TRIGGERING EVENT · Sudden increase in the number of erroneous calls · DURATION 10m · On: storefront-ihs

Related Events (5)

19:26:15 · SERVICE IMPACT · Sudden increase in the number of erroneous calls · DURATION 10m · On: postgres

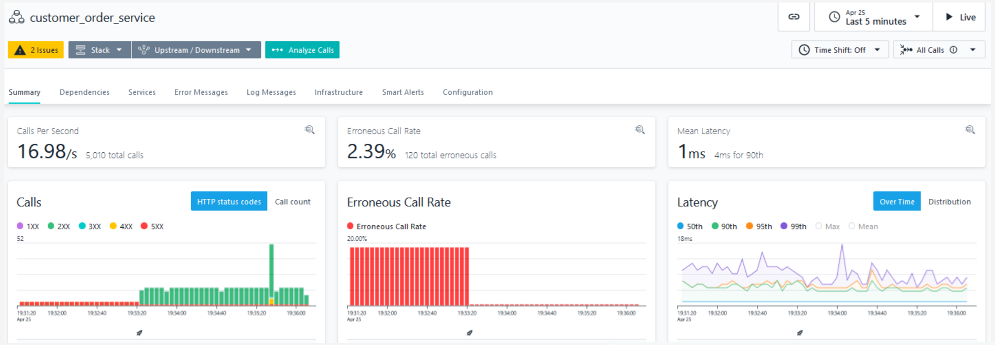
19:31:10 · SERVICE IMPACT · Sudden drop in the number of requests · DURATION 10m · On: postgres

19:26:15 · SERVICE IMPACT · Sudden increase in the number of erroneous calls · DURATION 10m · On: storefront-catalog

19:30:00 · SERVICE IMPACT · Erroneous call rate is too high · DURATION 5m · On: storefront-catalog

19:29:00 · SERVICE IMPACT · Erroneous call rate is too high · DURATION 6m · On: storefront-ihs

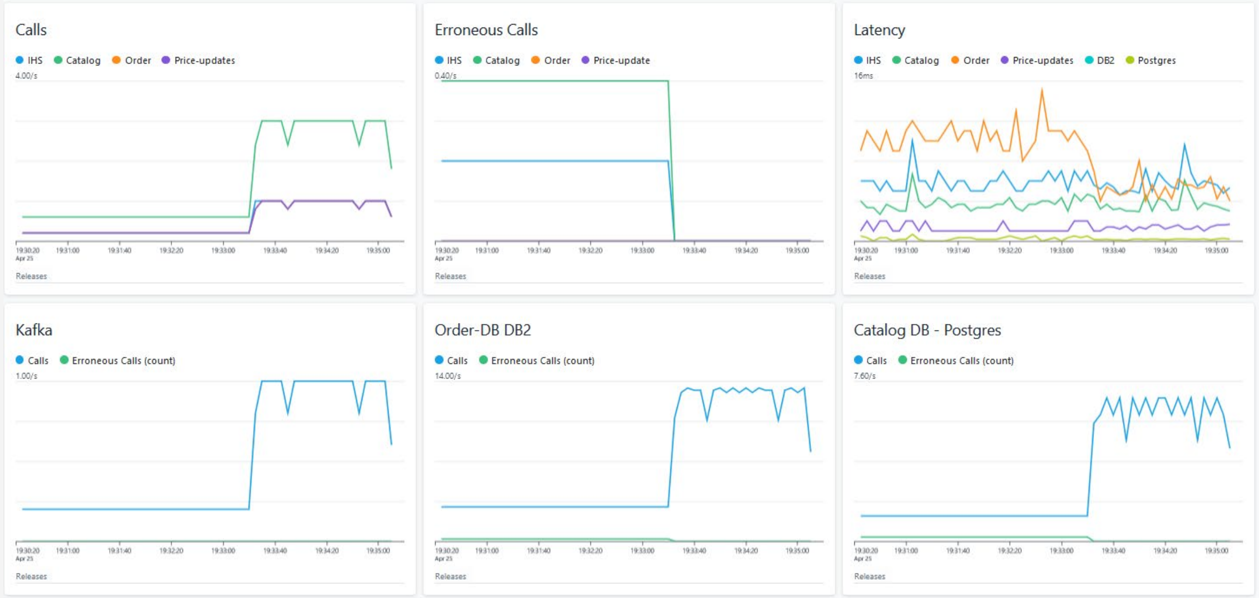
Troubleshooting Scenario 1 – Resolved by fix deployment



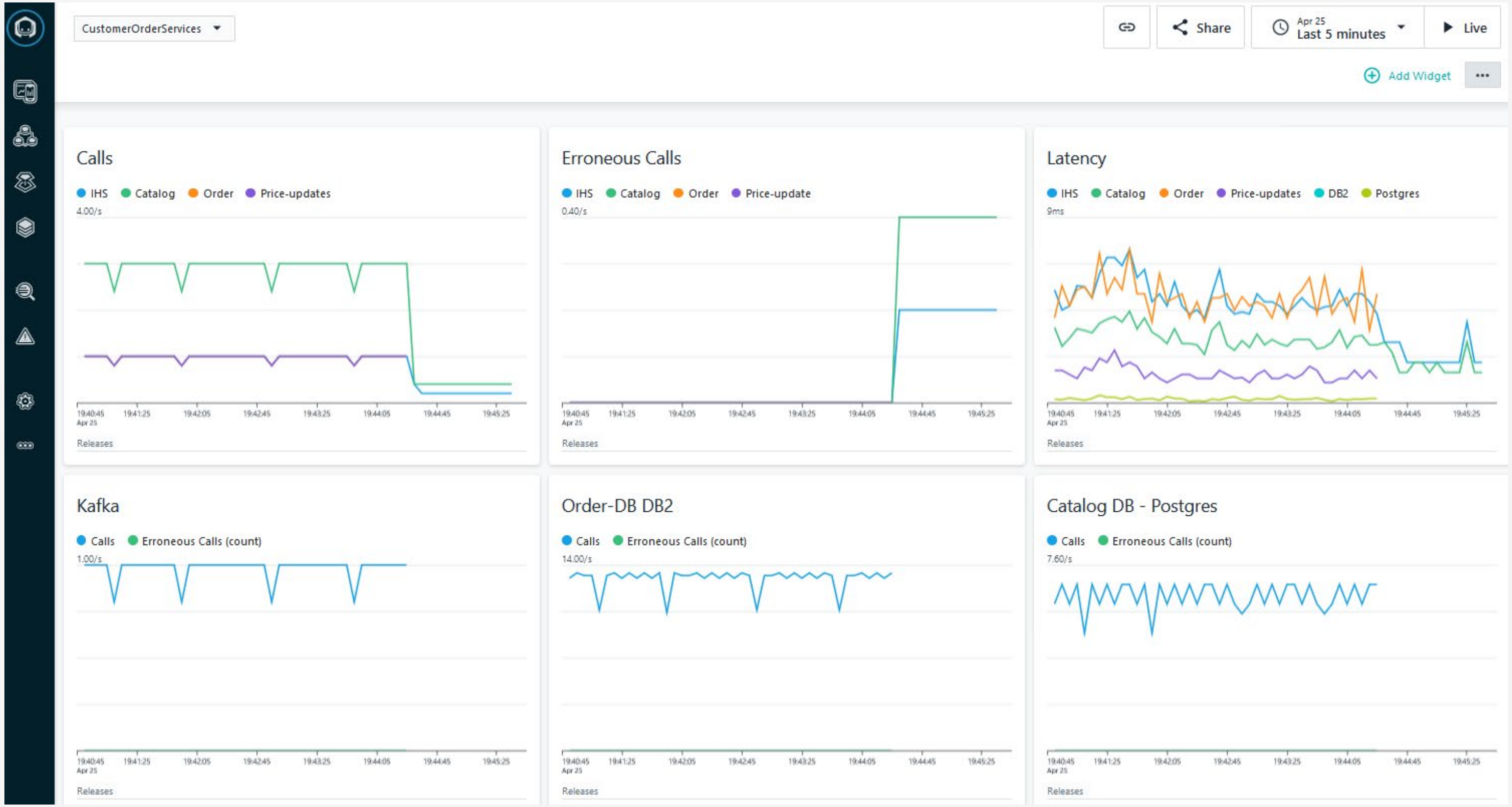
Calls Per Second 16.98/s 5,010 total calls

Erroneous Call Rate 2.39% 120 total erroneous calls

Mean Latency 1ms 4ms for 90th

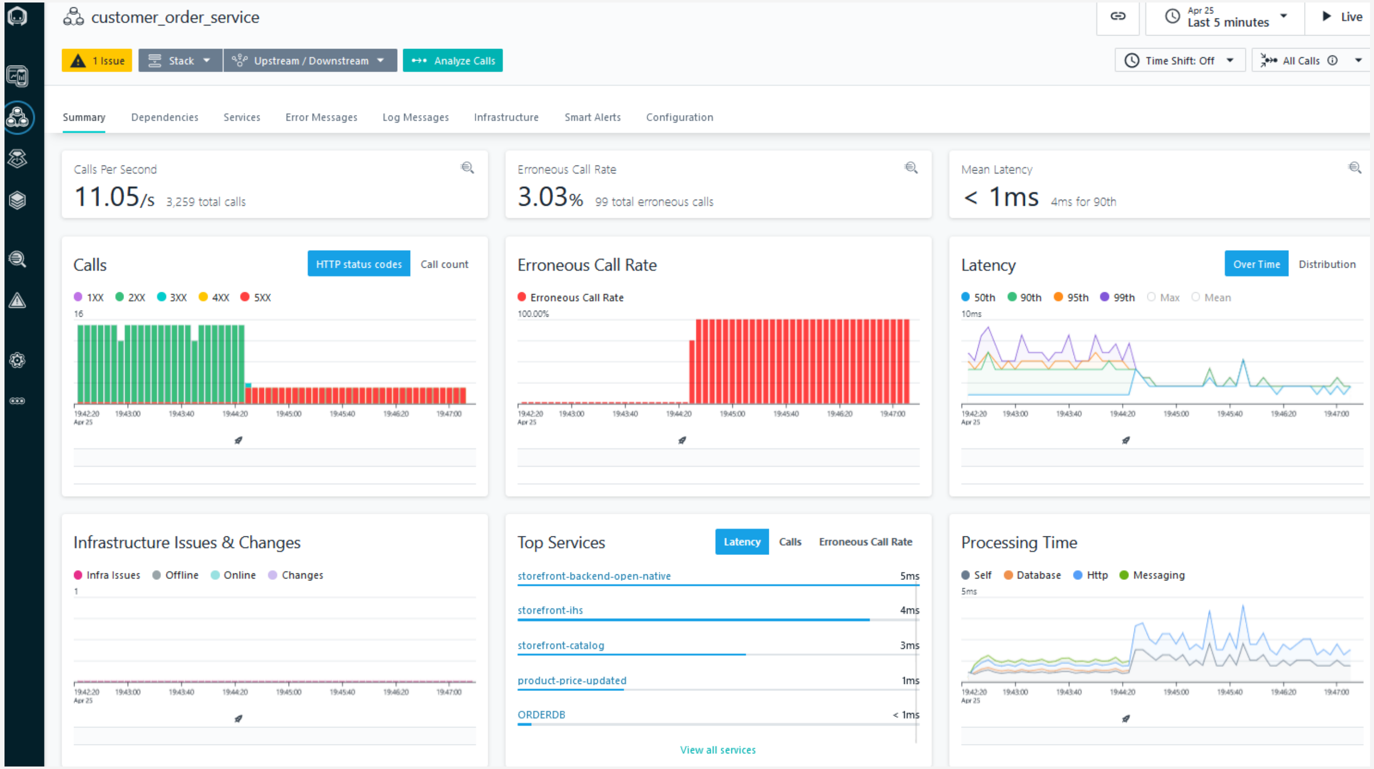


Troubleshooting Scenario 2 – Dashboard indicates problem



Troubleshooting Scenario 2 – Application Perspective

Problem likely caused by new deployment

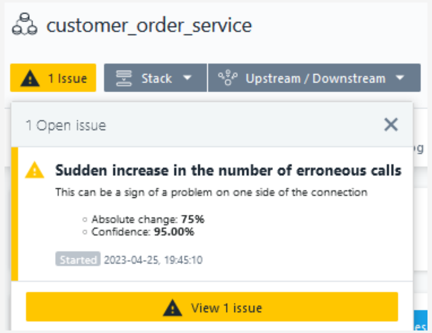


Calls Per Second: 11.05/s — 3,259 total calls

Erroneous Call Rate: 3.03% — 99 total erroneous calls

Mean Latency: < 1ms — 4ms for 90th

Top Services	Latency
storefront-backend-open-native	5ms
storefront-ihs	4ms
storefront-catalog	3ms
product-price-updated	1ms
ORDERDB	< 1ms



1 Open issue

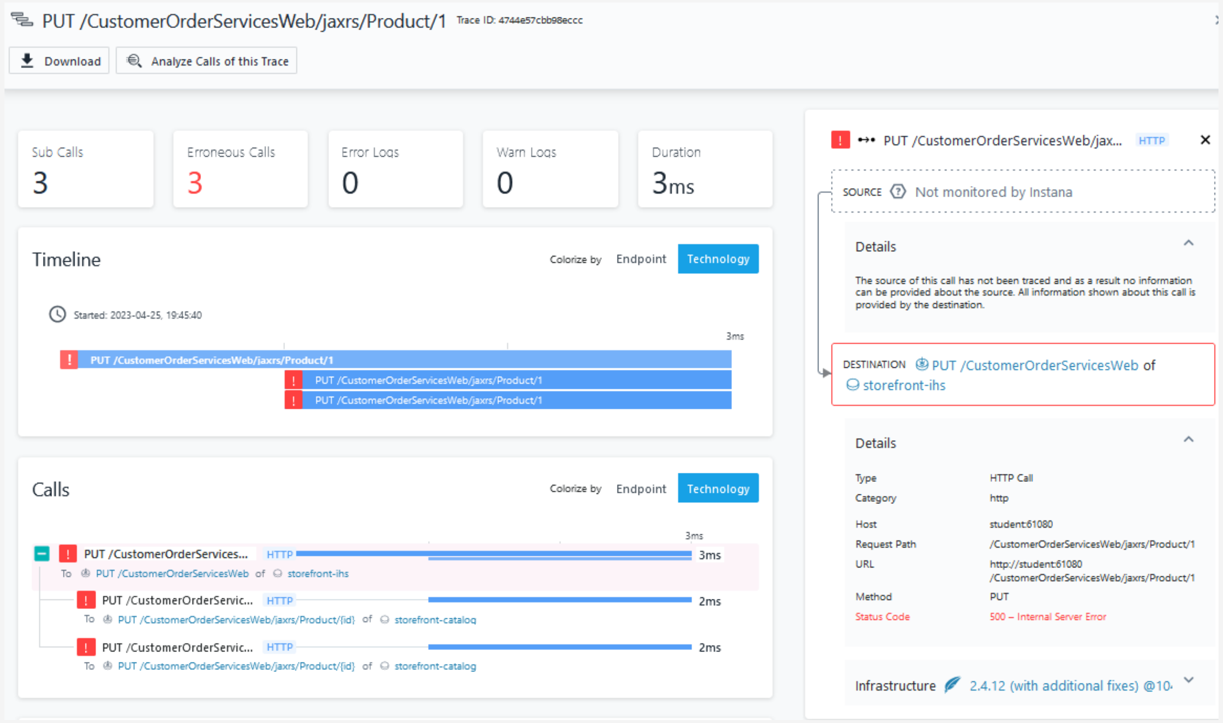
Sudden increase in the number of erroneous calls

This can be a sign of a problem on one side of the connection

- Absolute change: **75%**
- Confidence: **95.00%**

Started 2023-04-25, 19:45:10

Troubleshooting Scenario 2 – Track request



PUT /CustomerOrderServicesWeb/jaxrs/Product/1 Trace ID: 4744e57cbb98eccc

Download | Analyze Calls of this Trace

Sub Calls	Erroneous Calls	Error Logs	Warn Logs	Duration
3	3	0	0	3ms

Timeline — Colorize by Endpoint | Technology

Started: 2023-04-25, 19:45:40

3ms

- PUT /CustomerOrderServicesWeb/jaxrs/Product/1
  - PUT /CustomerOrderServicesWeb/jaxrs/Product/1
  - PUT /CustomerOrderServicesWeb/jaxrs/Product/1

Calls — Colorize by Endpoint | Technology

3ms

- PUT /CustomerOrderServices... HTTP 3ms
  To PUT /CustomerOrderServicesWeb of storefront-ihs
  - PUT /CustomerOrderServic... HTTP 2ms
    To PUT /CustomerOrderServicesWeb/jaxrs/Product/{id} of storefront-catalog
  - PUT /CustomerOrderServic... HTTP 2ms
    To PUT /CustomerOrderServicesWeb/jaxrs/Product/{id} of storefront-catalog

PUT /CustomerOrderServicesWeb/jax... HTTP

SOURCE Not monitored by Instana

Details

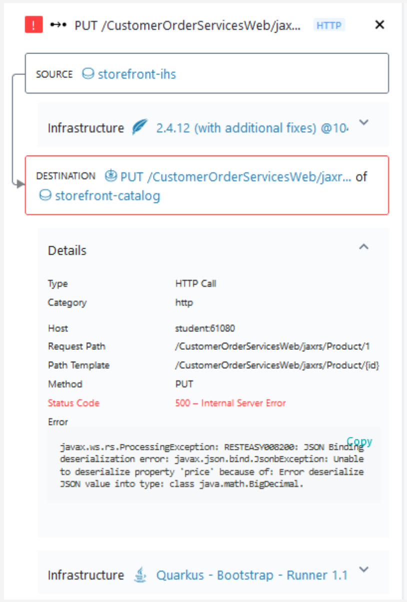
The source of this call has not been traced and as a result no information can be provided about the source. All information shown about this call is provided by the destination.

DESTINATION PUT /CustomerOrderServicesWeb of storefront-ihs

Details

Type	HTTP Call
Category	http
Host	student:61080
Request Path	/CustomerOrderServicesWeb/jaxrs/Product/1
URL	http://student:61080 /CustomerOrderServicesWeb/jaxrs/Product/1
Method	PUT
Status Code	500 – Internal Server Error

Infrastructure 2.4.12 (with additional fixes) @10·



PUT /CustomerOrderServicesWeb/jax... HTTP

SOURCE storefront-ihs

Infrastructure 2.4.12 (with additional fixes) @10·

DESTINATION PUT /CustomerOrderServicesWeb/jaxr... of storefront-catalog

Details

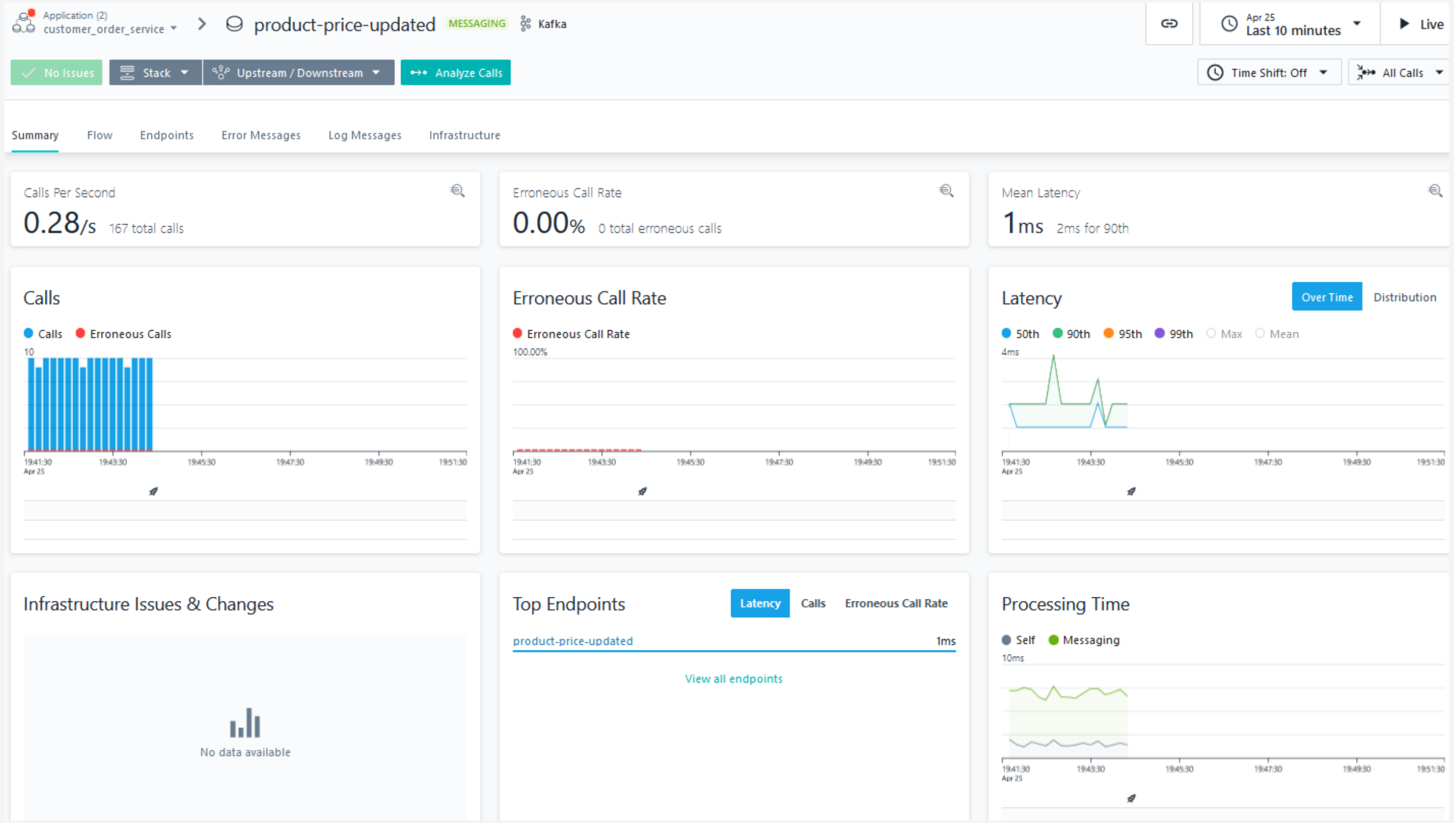
Type	HTTP Call
Category	http
Host	student:61080
Request Path	/CustomerOrderServicesWeb/jaxrs/Product/1
Path Template	/CustomerOrderServicesWeb/jaxrs/Product/{id}
Method	PUT
Status Code	500 – Internal Server Error
Error	

```
javax.ws.rs.ProcessingException: RESTEASY008200: JSON Binding
deserialization error: javax.json.bind.JsonbException: Unable
to deserialize property 'price' because of: Error deserialize
JSON value into type: class java.math.BigDecimal.
```

Copy

Infrastructure Quarkus - Bootstrap - Runner 1.1

Troubleshooting Scenario 2 – Impact for Kafka



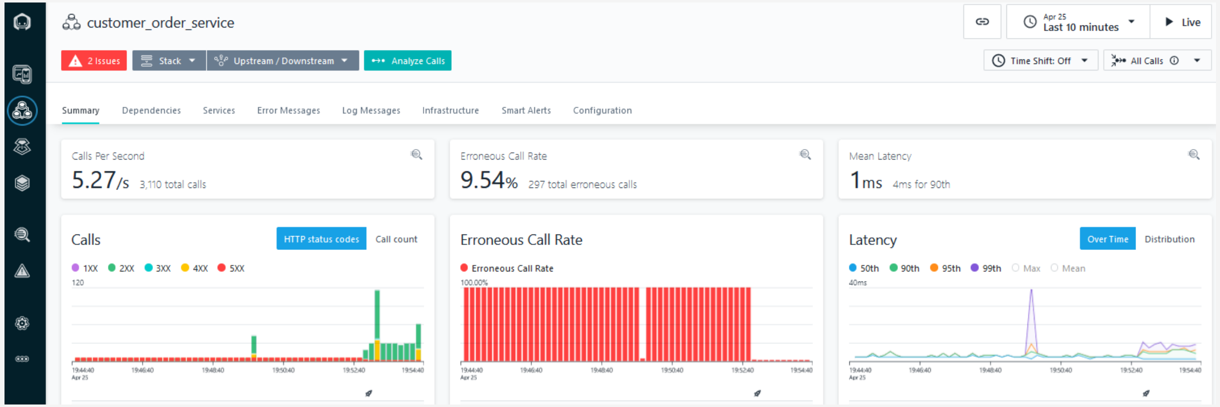
Calls Per Second 0.28/s 167 total calls

Erroneous Call Rate 0.00% 0 total erroneous calls

Mean Latency 1ms 2ms for 90th

product-price-updated 1ms

Troubleshooting Scenario 2 – Resolved by fix deployment



customer_order_service

Apr 25 Last 10 minutes | Live

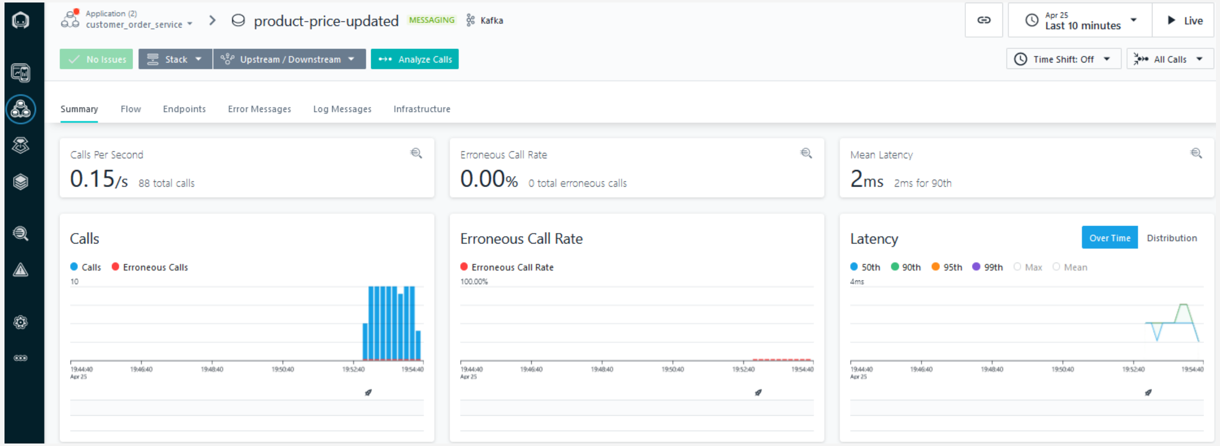
2 Issues | Stack | Upstream / Downstream | Analyze Calls | Time Shift: Off | All Calls

Summary | Dependencies | Services | Error Messages | Log Messages | Infrastructure | Smart Alerts | Configuration

Calls Per Second 5.27/s 3,110 total calls

Erroneous Call Rate 9.54% 297 total erroneous calls

Mean Latency 1ms 4ms for 90th



Application (2) customer_order_service > product-price-updated MESSAGING Kafka

Apr 25 Last 10 minutes | Live

No Issues | Stack | Upstream / Downstream | Analyze Calls | Time Shift: Off | All Calls

Summary | Flow | Endpoints | Error Messages | Log Messages | Infrastructure

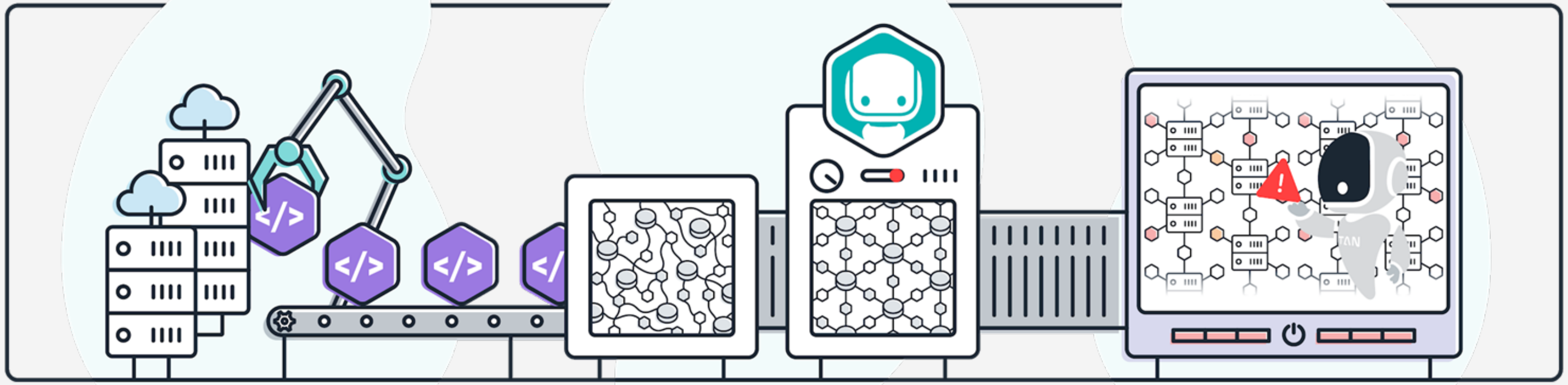
Calls Per Second 0.15/s 88 total calls

Erroneous Call Rate 0.00% 0 total erroneous calls

Mean Latency 2ms 2ms for 90th

Instana can help to keep control over cloud native apps

Automation   Context   Intelligent Action



✓ **Mitigate Risk**   ✓ **Protect Revenue**   ✓ **Gain Efficiency**

AUTOMATION
Gain full observability in dynamic environments with auto discovery & configuration

CONTEXT
Understand all application inter-dependencies to diagnose issues and determine impact

INTELLIGENT ACTION
Proactively detect and remediate issues with an understanding of contributing factors

Questions and Answers

Quarterly Liberty Update

Liberty 22.0.0.10-13 Update - Replay

- Session#1: Jan 19, 2023 from 1-2:30pm ET - https://ibm.biz/Liberty-Jan19
- Session#2: Jan 26, 2023 from 9-10am ET - https://ibm.biz/Liberty-Jan26

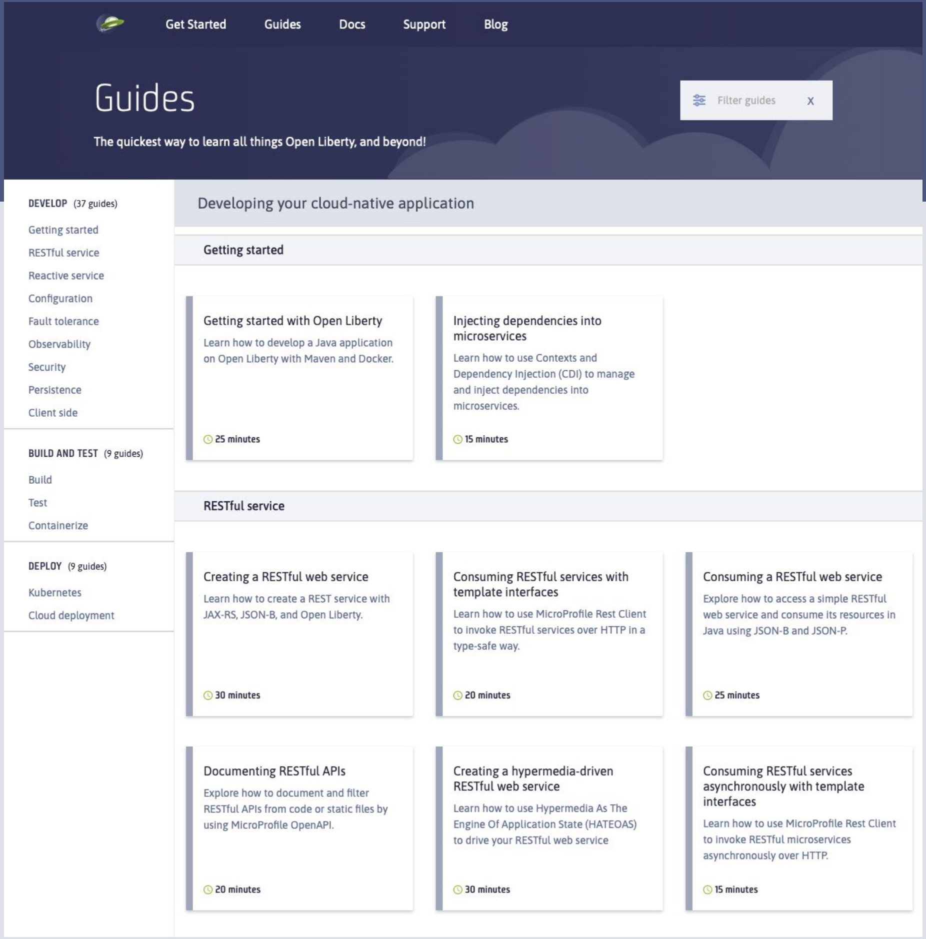
Liberty 23.0.0.1-3 Update

- Session#1: April 13, 2023 from 1-2:30pm ET - https://ibm.biz/Liberty-Apr13
- Session#2: April 20, 2023 from 9-10:30am ET - https://ibm.biz/Liberty-Apr20

Broadcast on Expert TV: http://ibm.biz/IBMExpertTV-LetsCode

Open Liberty guides

- Hands-on learning in ~20 minutes
- 55 guides
  - MicroProfile & Jakarta EE
  - Open Shift, Docker, Kubernetes Istio
- Latest Guide
  - *Containerizing microservices with Podman*



Guides

The quickest way to learn all things Open Liberty, and beyond!

DEVELOP (37 guides)
Getting started
RESTful service
Reactive service
Configuration
Fault tolerance
Observability
Security
Persistence
Client side

BUILD AND TEST (9 guides)
Build
Test
Containerize

DEPLOY (9 guides)
Kubernetes
Cloud deployment

Developing your cloud-native application

Getting started

Getting started with Open Liberty
Learn how to develop a Java application on Open Liberty with Maven and Docker.
25 minutes

Injecting dependencies into microservices
Learn how to use Contexts and Dependency Injection (CDI) to manage and inject dependencies into microservices.
15 minutes

RESTful service

Creating a RESTful web service
Learn how to create a REST service with JAX-RS, JSON-B, and Open Liberty.
30 minutes

Consuming RESTful services with template interfaces
Learn how to use MicroProfile Rest Client to invoke RESTful services over HTTP in a type-safe way.
20 minutes

Consuming a RESTful web service
Explore how to access a simple RESTful web service and consume its resources in Java using JSON-B and JSON-P.
25 minutes

Documenting RESTful APIs
Explore how to document and filter RESTful APIs from code or static files by using MicroProfile OpenAPI.
20 minutes

Creating a hypermedia-driven RESTful web service
Learn how to use Hypermedia As The Engine Of Application State (HATEOAS) to drive your RESTful web service
30 minutes

Consuming RESTful services asynchronously with template interfaces
Learn how to use MicroProfile Rest Client to invoke RESTful microservices asynchronously over HTTP.
15 minutes

Liberty References

Read

Why choose Liberty for Microservices
ibm.biz/6ReasonsWhyLiberty

Choosing the right Java runtime
ibm.biz/ChooseJavaRuntime

How to approach application modernization
ibm.biz/ModernizeJavaApps

Watch

Explore the latest on WebSphere and Liberty
ibm.biz/LibertyTV

View our recent Expert TV episode all about Liberty
ibm.biz/Liberty101

Learn more about Liberty in containers and Operator-based deployment
ibm.biz/LibertyContainerOperator

Experience

Try Liberty as a beginner
openliberty.io/guides/getting-started.html

Learn Liberty, MicroProfile, Containers, Kubernetes, Hands-on
openliberty.io/guides

Try Cloud-hosted guides (no pre-req's to install)
ibm.biz/HostedLibertyGuides

Modernization Tool References

Read

What’s new in Transformation Advisor: ibm.biz/WhatsNewTA242

What’s new in Migration Tools: ibm.biz/WhatsNewMigTools

Introduction to Mono2Micro: ibm.biz/Intro2Mono2Micro

Watch

Explore subjects related to App Modernization (including demos of Transformation Advisor, Mono2Micro, and the WebSphere Migration Tools) ibm.biz/AppTransformersTV

Experience

Assess your application estate with Transformation Advisor ibm.biz/cloudta

Make changes confidently using WebSphere Migration Toolkit ibm.biz/WASMigToolkit

Test drive the Mono2Micro refactoring experience ibm.biz/Mono2Micro

Learning Collections and courses

Liberty

Learning Collection - WebSphere Liberty
https://www.ibm.com/training/collection/ibmwebsphereli bertyadministration

Administering WebSphere Application Server Liberty Profile V9
https://www.ibm.com/training/course/WA190G

The road to Liberty
https://developer.ibm.com/articles/road-to-liberty-websphere-modernization-journey/

Modernizing applications to use WebSphere Liberty
https://developer.ibm.com/learningpaths/app-mod-liberty/

WebSphere Traditional

Administrator: IBM WebSphere Application Server V9
https://www.ibm.com/training/path/ibmwebsphereap plicationserverv9

WebSphere Application Server Administration
https://www.ibm.com/training/course/ZA590G

WebSphere Hybrid Edition and Modernization Tools

Learning Collection - IBM WebSphere Hybrid Edition
https://www.ibm.com/training/collection/ibmwebsphere hybridedition

Architect: IBM WebSphere Hybrid Edition
https://www.ibm.com/training/path/ibmwebspherehybri dedition

Introduction to IBM Mono2Micro
https://developer.ibm.com/learningpaths/intro-ibm-mono2micro/